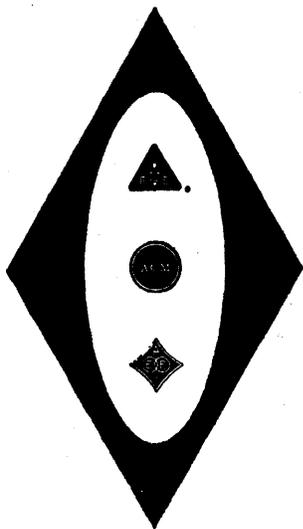


Proceedings of the

WESTERN JOINT COMPUTER CONFERENCE

March 3-5, 1959

San Francisco, Calif.



Sponsors:

THE INSTITUTE OF RADIO ENGINEERS

Professional Group on Electronic Computers

THE AMERICAN INSTITUTE OF ELECTRICAL ENGINEERS

Committee on Computing Devices

THE ASSOCIATION FOR COMPUTING MACHINERY

PROCEEDINGS OF THE WESTERN JOINT COMPUTER CONFERENCE

PAPERS PRESENTED AT
THE JOINT IRE-AIEE-ACM COMPUTER CONFERENCE
SAN FRANCISCO, CALIF., MARCH 3-5, 1959

Sponsors
THE INSTITUTE OF RADIO ENGINEERS
Professional Group on Electronic Computers
THE AMERICAN INSTITUTE OF ELECTRICAL ENGINEERS
Committee on Computing Devices
THE ASSOCIATION FOR COMPUTING MACHINERY

Published by
The Institute of Radio Engineers
1 East 79th Street, New York 21, N. Y.
for the
Joint Computer Committee

ADDITIONAL COPIES

Additional copies may be purchased from the following sponsoring societies at \$6.00 per copy. Checks should be made payable to any one of the following societies:

INSTITUTE OF RADIO ENGINEERS

1 East 79th Street, New York 21, N. Y.

AMERICAN INSTITUTE OF ELECTRICAL ENGINEERS

33 West 39th Street, New York 18, N. Y.

ASSOCIATION FOR COMPUTING MACHINERY

2 East 63rd Street, New York 21, N. Y.

Copyright © 1959

THE INSTITUTE OF RADIO ENGINEERS

LIST OF EXHIBITORS

AERONUTRONIC SYSTEMS, INC.....	Santa Ana, Calif.
AMPEX CORP.....	Redwood City, Calif.
AMP INC.....	Harrisburg, Pa.
BECKMAN/BERKELEY DIVISION OF BECKMAN INSTRUMENTS, INC.....	Richmond, Calif.
BENDIX COMPUTER DIVISION OF BENDIX AVIATION CORP.....	Los Angeles, Calif.
E. L. BERMAN CO.....	San Francisco, Calif.
BRYANT COMPUTER PRODUCTS DIVISION.....	Springfield, Vt.
C. P. CLARE & CO.....	Los Angeles, Calif.
COLLINS RADIO CO.....	Burbank, Calif.
COMPUTER CONTROL CO., INC.....	Los Angeles, Calif.
DATA INSTRUMENTS DIVISION OF TELECOMPUTING CORP.....	North Hollywood, Calif.
DIGITAL EQUIPMENT CORP.....	Maynard, Mass.
ELECTRODATA DIVISION OF BURROUGHS CORP.....	Pasadena, Calif.
ELECTRONIC ASSOCIATES, INC.....	Long Branch, N. J.
ELECTRONIC ENGINEERING CO. OF CALIFORNIA....	Santa Ana, Calif.
FERRENTI ELECTRIC, INC.....	Hempstead, N. Y.
FRIDEN, INC.....	San Leandro, Calif.
GENERAL ELECTRIC CO., LIGHT MILITARY ELECTRONICS DEPT.....	Johnson City, N. Y.
HUGHES AIRCRAFT CO.....	Culver City, Calif.
INTERNATIONAL BUSINESS MACHINES CORP.....	New York, N. Y.
LABORATORY FOR ELECTRONICS, INC.....	Boston, Mass.
LIBRASCOPE, INC.....	Glendale, Calif.
F. L. MOSELEY CO.....	Pasadena, Calif.
G. E. MOXON SALES.....	Beverly Hills, Calif.
PACIFIC TELEPHONE & TELEGRAPH CO.....	San Francisco, Calif.
PHILCO CORP.....	Philadelphia, Pa.
RADIO CORP. OF AMERICA.....	Camden, N. J.
REMINGTON RAND DIVISION OF SPERRY RAND CORP.	New York, N. Y.
RESEARCH & ENGINEERING, THE MAGAZINE OF DATAMATION.....	Los Angeles, Calif.
RESE ENGINEERING, INC.....	Philadelphia, Pa.
ROYAL McBEE CORP.....	Port Chester, N. Y.
SOROBAN ENGINEERING, INC.....	Melbourne, Fla.
SPRAGUE ELECTRIC CO.....	North Adams, Mass.
STROMBERG-CARLSON—SAN DIEGO.....	San Diego, Calif.
TALLY REGISTER CORP.....	Seattle, Wash.
TELEMETER MAGNETICS, INC.....	Los Angeles, Calif.
THE THOMPSON-RAMO-WOOLDRIDGE PRODUCTS CO.	Los Angeles, Calif.
JOHN WILEY & SONS, INC.....	New York, N. Y.

NATIONAL JOINT COMPUTER COMMITTEE

Chairman

H. H. Goode
Bendix Systems Division
Ann Arbor, Mich.

Vice-Chairman

P. Armer
RAND Corporation
Santa Monica, Calif.

Secretary-Treasurer

Miss M. R. Fox
National Bureau of Standards
Department of Commerce
Washington 25, D.C.

IRE Representatives

R. D. Elbourn
National Bureau of Standards
Department of Commerce
Washington 25, D. C.

H. H. Goode
Bendix Systems Division
Ann Arbor, Mich.

W. Buchholz
IBM Product Dev. Lab.
P.O. Box 390
Poughkeepsie, N. Y.

L. Nofrey
General Electric Computer Dept.
Phoenix, Ariz.

AIEE Representatives

R. A. Imm
IBM Corporation
Rochester, Minn.

G. Glinski
Dept. of Electrical Engineering
University of Ottawa
Ottawa, Ont., Can.

C. A. R. Kagan
Engineering Research Center
Western Electric Company, Inc.
Princeton, N. J.

S. Rogers
c/o Convair, Mail Zone 6-156
San Diego 12, Calif.

ACM Representatives

P. Armer
RAND Corporation
Santa Monica, Calif.

J. D. Madden
System Development Corp.
Santa Monica, Calif.

F. M. Verzuh
Massachusetts Institute of Technology
Cambridge, Mass.

H. R. J. Grosch
IBM Corporation
New York, N. Y.

Ex-Officio Representatives

R. W. Hamming (ACM)
Bell Telephone Laboratories
Murray Hill, N. J.

W. H. Ware (IRE)
RAND Corporation
Santa Monica, Calif.

M. Rubinoff (AIEE)
Philco Corporation
Government and Industrial Division
Philadelphia, Pa.

The Headquarters Representatives

J. Moshman
Council for Economic and Industry Research, Inc.
Arlington, Va.

L. G. Cumming
The Institute of Radio Engineers
New York, N. Y.

R. S. Gardner
American Institute of Electrical Engineers
33 West 39th Street
New York, N. Y.

WESTERN JOINT COMPUTER CONFERENCE COMMITTEE

General Chairman	R. R. Johnson, General Electric Co., Palo Alto, Calif.
Vice-Chairman	R. W. Melville, Stanford Research Inst., Menlo Park, Calif.
Secretary-Treasurer	H. G. Asmus, General Electric Co., Palo Alto, Calif.
Technical Program	R. W. Melville, <i>Chairman</i> J. Paivinen, General Electric Co., Palo Alto, Calif. A. S. Zukin, Lockheed Aircraft Corp., Palo Alto, Calif. D. Teichroew, Stanford University, Stanford, Calif.
Publications	B. J. Bennett, <i>Chairman</i> , IBM Corp., San Jose, Calif. E. Lowe, IBM Corp., San Jose, Calif. D. Willard, IBM Corp., San Jose, Calif. A. Bagley, Hewlett-Packard Co., Palo Alto, Calif.
Publicity	G. A. Barnard, III, <i>Chairman</i> , AMPEX, Redwood City, Calif. C. Elkind, <i>Vice-Chairman</i> , Stanford Research Inst., Menlo Park, Calif. W. C. Estler, <i>Consultant</i> , Palo Alto, Calif.
Exhibits	H. K. Farrar, <i>Chairman</i> , Pacific Telephone & Telegraph Co., San Francisco, Calif. G. H. Warfel, Bank of America, San Francisco, Calif.
Field Trips	K. F. Tiede, <i>Chairman</i> , Livermore Radiation Lab., Livermore, Calif.
Registration	R. M. Bennett, Jr., <i>Chairman</i> , IBM Corp., San Jose, Calif. F. Chiang, IBM Corp., San Jose, Calif.
Printing	L. D. Krider, <i>Chairman</i> , Livermore Radiation Lab., Livermore, Calif. R. Abbott, Livermore Radiation Lab., Livermore, Calif.
Women's Activities	Mrs. J. Teasdale, <i>Chairman</i> , General Electric Co., San Jose, Calif. Mrs. R. Waters, General Electric Co., San Jose, Calif. Mrs. E. Majors, General Electric Co., San Jose, Calif.
Local Arrangements	R. C. Douthitt, <i>Chairman</i> , Remington-Rand, El Cerrito, Calif. W. Gerkin, Remington Rand, San Francisco, Calif.
Mailing	E. T. Lincoln, <i>Chairman</i> , Stanford Research Inst., Menlo Park, Calif.

TABLE OF CONTENTS

Foreword	7
New Horizons in Systems	Darwin E. Ellett 8
A Multiloop Transfluxor Memory	D. G. Hammel, W. L. Morgan, and R. D. Sidnam 14
Design and Analysis of MAD Transfer Circuitry	D. R. Bennion and H. D. Crane 21
A Twistor Matrix Memory for Semipermanent Information	Duncan H. Looney 36
A Card Changeable Nondestructive Readout Twistor Store	J. J. DeBuske, J. Janik, Jr., and B. H. Simons 41
Square-Loop Magnetic Logic Circuits	Edward P. Stabler 47
Relative Merits of General and Special Purpose Computers for Information Retrieval	A. Opler and N. Baird 54
A Specialized Library Index Search Computer	B. Kessel and A. DeLucia 57
Programmed Interpretation of Text as a Basis for Information-Retrieval Systems	L. Doyle 60
A Theory of Information Retrieval	Clinton M. Walker 63
The Role of USAF Research and Development in Information Retrieval and Machine Translation	Robert F. Samson 66
Computing Educated Guesses	E. S. Spiegelthal 70
A Memory of 314 Million Bits Capacity with Fast and Direct Access—Its Systems and Economic Considerations	N. Bishop and A. I. Dumey 74
Information Retrieval on a High-Speed Computer	A. R. Barton, V. L. Schatz, and L. N. Caplan 77
The Next Twenty Years in Information Retrieval: Some Goals and Predictions	Calvin N. Mooers 81
Simulation of an Information Channel on the IBM 704 Computer	E. G. Newman and L. O. Nippe 87
A Compiler with an Analog-Oriented Input Language	M. L. Stein, J. Rose, and D. B. Parker 92
Automatic Design of Logical Networks	T. C. Bartee 103
The Role of Digital Computers in the Dynamic Optimization of Chemical Reactions	R. E. Kalman and R. W. Koepcke 107
Simulation of Human Problem-Solving	W. G. Bouricius and J. M. Keller 116
The Role of the University in Computers, Data Processing, and Related Fields	Louis Fein 119
The RCA 501 Assembly System	H. Bromberg, T. M. Hurewitz, and K. Kozarsky 127
A Program to Draw Multilevel Flow Charts	Lois M. Haibt 131
A Compiler Capable of Learning	Richard F. Arnold 137
Special-Purpose, Electronic Data Systems—The Solution to Industrial and Commercial Automation	William V. Crowley 143
The Residue Number System	Harvey L. Garner 146
System Evaluation and Instrumentation for Military Special-Purpose Digital Computer Systems	A. J. Strassman and L. H. Kurkjian 153
Automatic Failure Recovery in a Digital Data-Processing System	R. H. Doyle, R. A. Meyer, and R. P. Pedowitz 159
A High-Speed Data Translator for Computer Simulation of Speech and Television Devices	E. E. David, Jr., M. V. Mathews, and H. S. McDonald 169
Some Experiments in Machine Learning	Howard Campaigne 173
Some Communication Aspects of Character-Sensing Systems	Clyde C. Heasley, Jr. 176
An Approach to Computers That Perceive, Learn, and Reason	Peter H. Greene 181
Automatic Data Processing in the Tactical Field Army	A. B. Crawford 187
Data Transmission Equipment Concepts for FIELDDATA	W. F. Luebbert 189
A High-Accuracy, Real-Time Digital Computer for Use in Continuous Control Systems	W. J. Milan-Kamski 197
The Man-Computer Team in a Space Ecology	J. Stroud and J. McLeod 202
The RCA 501 High-Speed Printers—The Story of a Product Design	C. Eckel and D. Flechtner 204
A Digital Computer for Industrial Process Analysis and Control	Edward L. Braun 207
The Burroughs 220 High-Speed Printer System	F. W. Bauer and P. D. King 212
The ACRE Computer—A Digital Computer for a Missile Checkout System	Richard I. Tanaka 217
IBM 7070 Data-Processing System	J. Svigals 222
An Organizational Approach to the Development of an Integrated Data-Processing Plan	George J. Fleming 231
Developing a Long-Range Plan for Corporate Methods and the Dependence on Electronic Data Processing	Norman J. Ream 234
A General Approach to Planning for Management Use of EDPM Equipment	Gomer H. Redmond 240
Dynamic Production Scheduling of Job-Shop Operations on the IBM 704 Data-Processing Equipment	L. N. Caplan and V. L. Schatz 244
Numerical Methods for High-Speed Computers—A Survey	George E. Forsythe 249
More Accurate Linear Least Squares	Richard E. von Holdt 255
The CORDIC Computing Technique	Jack Volder 257
Monte Carlo Calculations in Statistical Mechanics	W. W. Wood and J. D. Jacobson 261
Real-Time Digital Analysis and Error-Compensating Techniques	Wally Ito 269
Automatic Digital Matrix Structural Analysis	M. Chirico, B. Klein, and A. Owens 272
A New Approach to High-Speed Logic	W. D. Rowe 277
Information Retrieval Study	Robert Cochran 283
Communication Across Language Barriers	W. F. Whitmore 286
Symbolic Language Translation	Eugene C. Gluesing 288
A Generalized Scanner for Pattern- and Character-Recognition Studies	W. H. Highleyman and L. A. Kamensky 291
File Searching Using Variable Length Keys	Rene De La Briandais 295
Program Design to Achieve Maximum Utilization in a Real-Time Computing System	A. Frederick Rosene 299
Pattern and Character Recognition Systems—Picture Processing by Nets of Neuron-Like Elements	L. A. Kamensky 304
The Social Responsibility of Engineers and Scientists	F. B. Wood 310
Emergency Simulation of the Duties of the President of the United States	Louis L. Sutro 314
Can Computers Help Solve Society's Problems?	Jerome Rothstein 323
The Measurement of Social Change	Richard L. Meier 327
Simulation of Sampled-Data Systems Using Analog-to-Digital Converters	Michael S. Shumate 331
FOXY 2: A Transistorized Analog Memory for Functions of Two Variables	L. J. Kamm, P. C. Sherertz, and L. E. Steffen 338
A Time-Sharing Analog Computer	John V. Reihing, Jr. 341
Computers—The Answer to Real-Time Flight Analysis	Guenther Hintze 350
Industry's Role in Supporting High-School Science Programs	J. O. Paivinen, Chairman 358

Foreword

The Western Joint Computer Conference returned to San Francisco, Calif., for the second time in 1959. The dynamic growth in our industry and the computer industry growth in the San Francisco area is evidenced by the doubling of attendance since 1956.

The 1959 Western Joint Computer Conference presented a view of "New Horizons with Computer Technology." These new horizons are appearing with the new circuits and devices that are arising from the maturing research efforts of our growing businesses. Similarly, new vistas are opening as a result of the knowledge gained through the application and utilization of computers throughout our entire economy.

These annual conferences are intended to provide an opportunity for the professional people in the computer industry to discuss their technical and business interests and accomplishments. The papers herein present the record of these discussions.

R. R. JOHNSON
Conference Chairman

New Horizons in Systems

DARWIN E. ELLETT†

IT HAS been our experience in the Air Materiel Command, and I imagine this is true of everyone engaged in data systems modernization and automation, that the further we advance, the greater potential we find for achievement. A prime characteristic of our objectives would appear to be mobility, for as we move closer to each established milestone, we inevitably find a new horizon beyond. So, in keeping with my subject, I propose to review some suggested issues and opportunities for future exploration and development.

The field of data systems design and automation is at once broad and complex, with its many interlocking facets of concept, principle, methodology, equipping, and technique, and its deep penetration into every phase of management and operational activities. Further, it is a very dynamic field. There are interacting advancements afoot in all of these facets and all of these phases, and there is, therefore, a wide range of new horizons available for examination.

In the light of this situation, my comments necessarily follow from a process of selection and limitation. I have chosen those areas which appear to us to be of the broadest application and the most basic concern.

As a first order of limitation, I shall confine my remarks to management systems from the viewpoint of the Air Materiel Command. I shall further limit the subject to a discussion of four steps in the management process:

- 1) The expression of the mission of the Command.
- 2) The establishment of the work processes for the accomplishment of this mission.
- 3) The establishment and organization of the management rules associated with these work processes.
- 4) The translation of these rules, where appropriate, into the procedural detail required for machine application.

Although I am basing my remarks on our experiences in the Air Materiel Command, I think it is safe to say that the fundamental management decisions are substantially the same in all enterprises. Therefore, the issues upon which I shall build my case are offered as being of possible general application. In this connection, since I am basically describing a possible course for our own future efforts, my intent is simply to add whatever stimulus I may to your thinking, rather than to propound ready-made answers.

The immediate reaction to my selection of specific topics may be that I am outside the lawful hunting

ground of the data systems designer. Traditionally, this is true. I think most of us work under written or implicitly accepted legislation which decrees that the establishment and ordering of work processes is a separate task, and that management retains the prerogative of deciding and specifying how it shall manage.

Let it be understood at once that I am not about to make a case for major adjustment in the organizational chart, with the data systems planner to come into a new ascendancy. Rather, I am making the proposition that we cannot do all that we should, if in our data systems work we go no deeper than a reappraisal and updating of accounting and computational methodology, for this level is to a large degree no more than a reflection of the basic work flow itself.

In developing a case for positive action, I should like to begin with a brief review of our own data systems modernization program. Since early 1954, we have been vigorously pursuing a command-wide program for modernizing data systems through the use of electronic data processing and modern communications. Our primary aim has been data systems integration. I shall not attempt to define integration to the satisfaction of everyone, but I should like to devote a moment or two to pointing up what such an objective has meant to our program. Simply stated, it has meant that our program could be no less in scope than the command itself. While we do pursue a policy of extensive decentralization, all of our activities and operating facilities are themselves completely interdependent; they are as indivisible as airpower itself.

Let me further enlarge on this point by a brief description of our specific management responsibilities and relationships.

The Air Materiel Command is the agency responsible for materiel support of all Air Force organizations and activities in accordance with established Air Force plans and programs. This involves:

- 1) Managing the production and delivery into active service of the air weapons themselves.
- 2) Determination of requirements and control of world-wide distribution of the million and a half line items of spare assemblies, parts, and supporting equipment and supplies in today's inventories.
- 3) Operation of a formidable array of industrial facilities which must physically accomplish the procurement, storage, transportation, and repair of these items.

In the execution of these responsibilities, we do not have a situation, as have many industrial concerns, where each operating division, or plant, can be permanently identified with a separate product, or range of

† Colonel, USAF; Chief, Data Systems Plans Div., Air Materiel Command, Wright-Patterson AFB, Ohio.

products, and can therefore engage in separate forecasting, workload planning, budgeting, production, and distribution. To the contrary, it is in the nature of the Air Force mission, and the highly dynamic state of research and development, that air weapons will constantly enter and leave the active inventory, and during their life cycles, will enjoy different relative precedences at different points in time. All command resources, in funds, materiel of common usage, and facilities, must be applied in the manner most effective for appropriate support of all weapons, in accordance with the current combat force structure, and the precedences of units and weapons within this structure.

This condition brings to bear two severe criteria for the data systems planner:

- 1) In the light of the very dynamic operational and technological environment, there must be a very high capability for rapid and coordinated reaction and adjustment down through all of the levels of materiel support programming and program execution, from weapon production, through scheduling of spares acquisition and allocation, to workloading of the industrial facilities.

- 2) The managers of all components of the command, however organized and wherever located within our fourteen major management centers across the country, are jointly engaged in managing a common enterprise in support of a common workload. This means that there is complete interdependency of management decision and action, and therefore interdependency for management information. Thus, we have, in fact, a single management process and the need for a single, all-inclusive data system.

With an awareness of this need, our data systems development program has moved out on this general basis:

- 1) We have defined and published this objective of systems integration, and the essential detail of planning criteria it engenders.

- 2) We have initiated and are well along on a systems design and application program which provides for the identification and separate modernization of data subsystems.

There are about 30 of these subsystems within our total data system. Examples range from the computation of requirements for aircraft spare parts, to inventory control of ground vehicles, to cost accounting and labor distribution in the maintenance shops. The objective of command-wide integration implies the need for command-wide procedural standardization, and our development program is geared to this end. A formal development project has been established for each subsystem. Each such project is guided by a Headquarters steering group made up of representatives of all offices of management or systems interest and monitored by a representative of the Data Systems Plans Division. Since our materiel operation is decentralized, most of the projects cover depot-level applications. Therefore,

much of the procedures design work, and the machine programming, test, and initial computer application are assigned to a "pilot" Air Force Depot in each case. When this work is completed, all other Air Force Depots concerned implement the standard product as developed. (Air Materiel Areas and Air Force Depots are equivalents for our purposes here. Each of these activities holds world-wide materiel management responsibility for an assigned segment of the total inventory. In addition, each is the site of a part of the industrial facility complex, and as such engages in materiel processing, that being primarily major repair and wholesale storage, for the various materiel management elements.)

As one can see, under this approach we are taking the data system apart, updating its components, and putting it back together again in a new composite. We are doing this through central control against an end of integration, and through decentralized but standard modernization of components.

We can report that this approach works, and it works rather well if the objective is to make rapid improvements in data handling. However, it will not quickly achieve integrated systems, as will become evident in this discussion. We have been attacking the subsystems on a "pay-off-as-we-go" basis, in an order of priority which affords the greatest rate of improvement in our combat support capability and in resources management. Many of our priority efforts have reached the operational stage, with acceptable results; others are nearly there.

About a dozen large-scale and 30-odd small- and medium-scale computers are in active use at Headquarters and at our Air Materiel Areas and Air Force Depots. We have in being a world-wide punch-card transceiver net which links these centers and most of the major Air Force Bases, and we are well along toward significant further improvement in our communication systems.

Beyond these immediate data systems improvements, there has been increasing awareness of the need and opportunity for greater management integration. In recent months, the command has undergone considerable adjustment in management organization, moving into an aggregation of responsibilities which parallels closely the present array of decision points and resulting data flow.

While our data system work has been by no means the sole (and quite possibly not even the primary) cause of this reorganization, it is safe to say that it has contributed an important and positive influence. At the same time, and notwithstanding these significant achievements, if we are to attain and retain the ultimate in an integrated system, there is much yet to be done, and this brings us back to the further efforts I defined at the beginning of this discussion.

Before I move into a brief review of these issues, I should like to point out that much of what I shall say is based primarily on personal thoughts and observations,

and should not be regarded as a chronicle of what the Air Force now officially proposes to do. I am simply proposing to share with you, as fellow workers in the systems field, some of the ideas that have evolved in attempting to project my thinking beyond today's state-of-the-art and potential achievement in the near term.

Let us begin with my first step, "The expression of the mission of the Command." As presently stated, the Air Materiel Command does have an abstract single ultimate objective. In a very simplified version, it is: To produce and deliver air weapons and thereafter to support them through their life cycles, in accordance with Air Force plans and programs and priorities, and as indicated earlier, we have a single integrated enterprise to accomplish this mission. Although it has a multiplicity of functions, and a complex far-flung organization, it is nevertheless of single purpose and unified by nature. Now, in the modernization of data systems over the past few years, it has become increasingly evident that if the many decision points in this total system are to be equated, then the mission itself, to which they should all relate, must be stated quantitatively. There have, of course, been many advancements over the years toward a more precise definition of the end product of the work of the Command. For example, we have used and continue to use a desired in commission rate for each of the various types of aircraft and missiles in the inventory. And to some extent, we have used a desired percentage of positive supply action as an objective of the enterprise. We use and have used an array of other objectives, such as a desired percentage of the aircraft and missiles in commission that are equipped to perform their wartime mission, or the sortie capability desired at a given point in time by various units, with an expression of the quantitative materiel requirements at a particular site to achieve this. None of these objectives or definitions of the products of the Command are entirely satisfactory when viewed from the standpoint of the data system designer, who is attempting to relate appropriately all decision points to each other and to the mission of the organization. This is so because the objective has never been defined as a single all-inclusive quantified expression. I realize that to achieve this is a tremendous undertaking, but without it, no final evaluation of all of the work processes as to their essentiality and quality can be achieved nor can the best supporting automated data system be attained. Our systems people are not now actively working in this particular area, but we realize that to do so will become increasingly more important with the passage of time.

Next, the establishment and description of the physical work processes necessary to the accomplishment of this mission. By referring again to our single ultimate purpose, and the unity of our enterprise, we come to this conclusion: The entire operation can be considered in total as a world-wide production line, composed of an unbroken series of work steps, from original input of Air Force programs to final delivery of the last item of ma-

teriel to the using forces. These are the action steps which march through all phases of planning, programming, and budgeting, and the direction, execution, and evaluation of results of these programs. And although there is a positive interrelationship between the work steps and the rules associated with them, I believe the general pattern of work must be planned before setting down the rules by which it will be directed and reported.

I do not know how many such steps would be involved in the typical situation, nor do I know how many there are in our own system, but obviously there are a considerable number, possibly several thousand. In whatever event, a description of these steps, in their logical sequence and without regard to current organization or division of responsibility, is a delineation of the live system itself, and is therefore the foundation for the most effective data system. Since the advent of the electronic computer, and of course even before that, we have all given much attention to the data needs of management, and have been much concerned with meeting these requirements. I am now suggesting that there is a more basic task at hand, if we are to explore the full potential of today's equipment and techniques. We must equip ourselves with a systematic view of the total work process in which the data system is grounded, which is at once the source of all its inputs and the destination of all outputs. For the relatively small and simple enterprise, operating at one location, this may be no great problem; all of the elements may literally be already in sight. However, for the large-scale activity, which has expanded with the industrial revolution and has tended to grow up in segments, through the necessary multiplication of management, this is no small task.

I can perhaps elaborate a bit on the true extent of such an effort, and further establish our need for this systematic view, by expanding the discussion of a work step. We find that in the operating system, the work required is a mixture of data processing and materiel processing, interwoven and interlocked. In the early stages of the cycle, particularly those involved in planning, programming, and budgeting, only data processing is involved. However, beyond the first point at which materiel is committed to manufacture, there is an interdependent blend of materiel processing and data processing in terms of work flow. From this observation, we have drawn the conclusion that data processing is so closely related to the physical work process that it is inseparable of design if the ultimate in integration is to be achieved. This alone is sufficient reason for a view which includes all work steps of whatever specific nature.

And this is by no means the whole story. If we follow this logic a bit further, we come to a rather startling revelation. While we have said that data processing is an integral part of the total work process, we are still confronted with the fact that the basic job is to manufacture, deliver, store, repair, use, and dispose of materiel. To go back to one of our earliest axioms: data processing

is not an end in itself, paper can never be the final product. This means that data processing serves only to provide a control and a report; a control to direct a physical action with reference to the materiel itself, and a report of action accomplished which in turn leads to direction of the next succeeding materiel work step, another action, another report, and so on throughout the entire system. Of course, there is a very definite layering effect in data processing. The execution of one data step may lead only to another, down through any number, but the whole series must eventually emerge into direction of a materiel processing action in our business or it has no purpose.

If all of this is so, then it must surely follow that much of the complexity and volume of data processing must be charged to the number of separate control directions to be given and reports to be rendered, and the extent of their repetition.

Therefore, can we not postulate from this that the road to data systems simplification must finally lead through simplification of the materiel work process? The simpler and better organized the ultimate job, the simpler the control and reporting system.

Beyond this, lies factory automation, which I think most of us have historically regarded as a not-too-familiar cousin. As the physical work itself is automated in longer and longer runs, there must be a corresponding decrease in the number of points at which individual energizing instructions must be entered from the central management data system. It should be possible in the foreseeable future to introduce original instructions at point of entry of materiel into the automated production run at hand, and thereupon, to pass control to the local automatic governing mechanism, with no need for report or new instruction until the materiel is delivered at the other end of the line. The longer and, therefore, the fewer number of separate production runs, the lesser demands upon the central management data system. Here again a case for reviewing the steps involved in both data processing and materiel processing as a single entity, with improvement in one basic to performance in the other.

Our case can be extended through examination of one final issue in this cycle of events. By automation of the materiel production process in long sequences, we do, as I have indicated, reduce the need for step-by-step control from the central management data system. But the fact remains that this control must go on. We have simply transferred it to the local automatic control device for the production run in question. Now, two facts become self-evident. First, this local control device is itself by nature a data processor, moving through a series of orders and counts, which are quantitative of expression, and second, it is energized by the input of production schedules which can only come from the central management data system. Again, the ordering and processing of the work itself is basic to a correct design of the data system, and automation of materiel

production is by definition to a large degree automation of management data processing.

In summary on this point, we come to the inevitable conclusion that we must eventually cease to view the data system as something apart; we must step up to a plane from which we can attack the entire operating system as an inseparable entity.

I should like to turn now to my third point: the establishment and organization of management rules associated with the work processes. This subject reflects to a considerable degree the same issues as those we found in reviewing the subject of work steps, for a management rule as I am using the term is basically an extension of the statement of a work step to include the conditions, actions, and exceptions which will apply under each set of possible circumstances. For each work step there must be governing management rules, and in composite these steps and their rules are a procedural extension of management objectives, plans, and policies. The incidence of management rules will, of course, depend entirely upon the incidence of variable conditions. In some situations, a standard and unvarying condition will exist and the only management rule needed is an implied, "Do this work step." Such work steps are most frequently found in straight production-line situations where there is no choice and therefore no local decision.

On the other hand, there are of course many situations where multiple alternatives present themselves. Since each decision will have impact on all of the steps that follow, and, therefore, on the functioning of the entire system, it is absolutely essential that all possible selections must be carefully established and described within a frame of reference which can be no less than the total operation.

I think I can develop this point and perhaps point up an interesting area for exploration by categorizing these management rules in two groups.

The first group would include all rules which are made necessary by the nature of the enterprise itself. It would hardly be realistic to assume that any large-scale activity could be straight-lined from original plan and policy to final accomplishment, with absolutely no need for intermediate decision and selection of alternatives along the way. There are too many variables and unpredictable in the surrounding environment alone to permit such a condition. I believe I can demonstrate this rather obvious fact by a simple example, in terms of our own Air Force materiel support, and in terms of work steps and management rules which are a part of data processing.

In the course of our accomplishment of materiel support, one of the major sequential work steps might be, "Distribute this commodity from storage point 'A' to customer 'B'." Obviously, such a statement immediately begets subordinate steps. One might be, "Process the customer's requisition." Now, as you can readily see, we have stepped off into the field of data systems, for a good part of processing a requisition is data processing.

We now come therefore to work steps and management rules which are of basic importance to the data system designer. One such work step might be, "Compare quantity requested with inventory balance." Here the management rules set in. For example, if the balance is adequate, take one action. If there is stock, but this order brings the balance below minimum levels for retention, take another. If there is no stock, there is a third type of action involved. No matter how precisely future systems may permit the planning of requirements, unpredictable events in manufacture, delivery, or usage may operate to cause this condition. Therefore, if automation is to occur, management must make rules in advance to cover all situations that may arise due to the relationships of the enterprise to its surrounding environment, and management must assume responsibility for failure in this area.

The second group would include what we might call deficit rules; those made necessary by faults in design of the system itself.

In a rather obvious example, one element of the data system might prescribe a work flow which terminates in the rendition of reports from a number of similar activities. Another element of the system must receive and act upon these reports. If these reports are in standard format and content from all reporting agencies, and as needed by the receiving activity, then it is possible to proceed immediately to the next work step of operational value. But, if reports are not consistent, there is a need to provide management rules and further work steps which cover actions to be taken to discover and eliminate the various discrepancies. This might involve returning incorrect or incomplete reports to the sender, and suspending the remainder. It might involve making assumptions which would permit the process to continue. In any event, it causes the establishment and building into the system of a set of management rules which serve no more than a deficiency purpose, and for which responsibility must rest with the systems designer himself.

If we design and create the system in segments, there is every reason to believe that with due care we shall eliminate the internal need for such deficit rules within each segment. However, with anything less than perfect coordination of detail and rate of progress among those designing the various segments, such deficiencies will inevitably occur. The direct result: inefficiencies in the segments, cumulative inefficiencies in the entire system, and finally inefficiency and loss of effectiveness and economy in meeting the basic mission. I believe that at this point we can add this review of management rules to our earlier observations on work steps, and draw a rather obvious conclusion: to secure the most effective results, we must design and bring into being a system which extends from original input programs to final product delivery in an unbroken series of work steps and governing management rules, with physical material processing and data processing steps and rules, and

their automation, regarded as inseparable and completely interdependent. The composition of such a system must be limited entirely to rules and steps which actually advance the work at hand; and as a first order of business, the system deficiencies must be sought out and eliminated. This latter statement raises a further issue: the efficient organization of these work steps and rules. We have just observed that steps and rules entered by the systems designer as a deficiency action must be eliminated. This leaves the province of work step establishment and type one rule formulation entirely to management, and this is as it should be. However, once this has been accomplished, the system designer must step forward and assume responsibility for their most effective organization. This division of effort has always been true of manual methods; it is even more pertinent as we move deeper into process automation. This organization comes to final fruition in our detailed procedural flow and our computer routines, and the outer limits of effective automation will depend upon our ability to absorb and consolidate these routines in larger and larger composites.

Thus, we find ourselves in a situation where we must pursue an objective of attaining and maintaining complete systems integration, accompanied by maximum effective automation. We must ask all of the many elements of management to establish work steps and formulate rules in the light of all others, and we must ask our systems designers to organize all of these steps for the most effective execution.

Given this statement of need, we must now cast about for a control vehicle through which we can approach this task, and here we come upon a familiar subject. I refer to the various directive and procedural manuals and similar publications which are an integral part of any management system, in which all of these work steps and rules are traditionally expressed, and which must continue to serve in one form or another as our documents of reference.

At this point, we are confronted with very real considerations which are at once friend and foe: the English language, and the human mind.

You are all well aware of the limitations of the English language as a device for definitive and precise communication in procedural publications. If this is true in a limited area, consider the problems involved in attempting to formulate, correlate, and record with precision, all work steps and management rules for the whole of a large-scale system. And consider further the very limited ability of the human mind to comprehend, analyze, correlate, and evaluate all of these management rules under such conditions.

Fortunately, we believe that there is the very crude beginning of a solution at hand, one which requires a great deal of imaginative and painstaking effort, but which offers great potential. This is central and standard control of the body of publication knowledge. I am sure that this approach has occurred to others in the

field of data system design, probably including at least some of you here today.

One of our consulting firms has just completed considerable initial research in this area, utilizing some of our existing publications for analysis. The results indicate that while this is certainly no simple task, it is entirely possible, and we are now preparing to extend our efforts into further research and probably later into a major attack on some of our more critical areas. Specifically those which are identified with our priority data subsystem development projects.

This attack is guided by four basic principles:

Principle No. 1—Work steps and management rules can be removed from the usual narrative form and expressed as independent entities, often in tabular form, and without consideration of data systems problem organization or the type of processing equipment to be employed. This principle in itself will work to bring greater precision to the formulation and application of management rules. Further, and most important, such an expression will serve to close partially the language gap between management and data systems designers. There is some relief from the need for joint logical flow-charting; management, as the creator of steps and rules, need not concern itself directly with the most efficient organization of rules for machine processing.

Principle No. 2—Techniques are required which will permit the efficient processing, correlation, and control of the total body of publication knowledge, as a basis for coordinated formulation of all work steps and management rules. This can be accomplished through the numerical coding of each English language information element, set, and system as they are used in procedural publications to describe these steps and rules. This permits the correlation of English words and phrases which are dissimilar in expression but have common meaning, and their reduction to specific and standard terms, which are in turn susceptible to consolidated filing and processing.

Principle No. 3—There can be established a central information repository containing all steps and management rules, expressed in terms of standard elements and sets of information, and indexed to all applicable components of the system, and to related publications. This library would serve as a control filter between management policy and the data system, by providing evaluation of the total impact of a proposed change to any rule or information element, a must in the interests of sustained integration.

Principle No. 4—Such a file is itself a very likely candidate for automation. This assumes a rather extensive file, and the need for frequent reference, and it has been our experience that our own system would certainly meet these criteria.

In summary to this point, we have proposed a quantitative expression of the objective, the restating of all contributing work steps, and management rules, in the most logical and effective sequence across the entire

system, their conversion to more precise and standard terms, and a central file and process which will permit management and systems designers to review a change to any sequential step or rule in relation to all others, an impossible feat in today's large enterprise, but possibly not so in the future.

Finally, we come to our last new horizon for today's consideration: automatic machine programming. This subject may seem a bit apart from those I have been discussing, but I believe I can establish a close liaison. As you all know, programming is a very difficult task requiring a very scarce skill, and it is one of the major limiting factors of our rate of progress in computer application. In the light of this situation, we, like other computer users, have moved actively into the field of automatic, or general, programming. We define this as a technique whereby the computer is instructed in basic English verbs, and in turn the machine codes instructions for itself when fed appropriately designed data to be processed. The criticality of this area has been such that we are applying some of our best talent in a major effort at improvement. While we are still far from final achievement, we have made considerable progress, and are quite optimistic about future potential. In addition to practical relief from the programming task, achievement here has been considered essential to insuring the necessary degree of data systems standardization and integration among our computer centers.

Beyond these direct systems benefits, let me now point up the close relationship between this field and those I discussed earlier. The specific situation is this: if we can achieve a high degree of direct correlation between the input to automatic programming and the standard expression of work steps and management rules, we will have forged the final link in the chain, and will have a highly controlled and responsive means for progressing from original management policy and decision to procedural implementation. We can then close the circuit by providing test decks and other analytical means for insuring that the information system has in fact produced the results desired by management in original decision and directive.

This, then, is our view of where we must go in the future if we are to reap the full benefits inherent in the use of advanced equipments and techniques, and are to finally achieve the best in management control through systems integration and automation.

In summary and analysis of the issues I have been discussing, it occurs to me that the basic substance has been this: The future of our systems work holds almost unlimited potential for gains in industrial efficiency, and that resulting benefits will continue to justify the expenditure of the required resources to achieve them.

However, I cannot leave the subject at this. I suggest for your sincere consideration that if we as systems people proceed solely within this goal of efficiency, we are guilty of introspection; we may miss completely the basic point and true objective of all our efforts to im-

prove the welfare of human beings. If there is a real reason for promoting this efficiency, it can be none other than the betterment of human living, whether through better defense of the nation, improvement of the standard of living, relief from manual drudgery, or whatever the specific and valid aim.

In pursuing our goals of greater integration and automation by attempting to predict all eventualities and to prearrange solutions, we as systems workers,

must never lose sight of this purpose, lest in our enthusiasm we do harm to our real intent. We must assume major responsibility for insuring that every step forward meets the criterion of even greater human contribution all up and down the line, and therefore even greater human dignity. Progress in systems design and automation must always be measured within this larger perspective, and must forever be conditioned, and perhaps even limited, by this governing need.

A Multiload Transfluxor Memory

D. G. HAMMEL[†], W. L. MORGAN[‡], AND R. D. SIDNAM[†]

INTRODUCTION

IN the field of computing machinery there is an ever-increasing demand for the development of random-access digital memory-storage units that operate at higher speeds and provide greater storage capacities. In 1951 a digital memory-storage unit that had a capacity of 1000 words and performed the basic memory cycle in about 200 μ sec was quite sufficient to satisfy the needs of a large-scale data-processing system. Today memory units of large-scale data-processing systems are being designed to provide as much as 90,000 words of storage capacity and to perform the basic memory cycle in about 4 μ sec. The trend is obvious, but the means of achieving desired results are often cumbersome.

The development of a superior memory-storage device is presently a major consideration of many prominent research activities. An important feature of the most promising schemes is the ability of the storage device to perform a nondestructive readout. This means that the state of the storage device is not destroyed whenever a readout is performed. This is not the case with present-day magnetic core memories which destroy the stored information when the core is read, and consequently require that the information be written back into the memory if retention is desired. This in effect gives the nondestructive storage medium a 2:1 speed advantage over the destructive storage device.

The read/write speeds of the memories being developed with present-day techniques are approaching their maximum, and any further increases can be achieved only with decreased reliability and increased costs. These memories are capable of executing only one access at a time and therefore restrict the digital com-

puters to functioning sequentially. The ability of a memory to perform more than one access simultaneously would be a major advancement in the computer field; this would be equivalent to increasing the read/write speed of the memory cycle. But a much more significant aspect to this mode of operation is that it would make possible the practical realization of truly parallel computers, computers capable of simultaneously and independently sharing the same memory or memories and hence able to communicate at the computer speed. A storage system which holds promise of fulfilling all these desirable features is a multiload transfluxor memory.

The multiload transfluxor is a multiapertured magnetic memory element employing the same type of high-remanent ferrite used in the ordinary memory cores. Thus the transfluxor is built upon a strong foundation of practical and theoretical ferrite core knowledge. The wealth of experience that has been accumulated during the development and use of coincident current magnetic core memories is applicable. In addition the transfluxor offers many properties heretofore unobtainable.

THE TWO-HOLE TRANSFLUXOR MEMORY

The original Rajchman and Lo transfluxor is a two-hole ferrite core.¹ The following is a brief explanation of the device. (See Fig. 1.)

The large hole is used for the writing operation and the small hole for reading. There are two parts to the writing cycle: a block pulse and a set pulse. The block pulse is a large pulse, either positive or negative, which saturates the entire core in one direction. When the core is blocked, the flux direction on both sides of the small

[†] RCA, Defense Electronic Products Div., Moorestown, N. J.
[‡] RCA, Astro-Electronic Products Div., Princeton, N. J.

¹ A complete explanation of the device may be found in J. A. Rajchman and A. W. Lo, "The transfluxor," Proc. IRE, vol. 44, pp. 321-332; March, 1956.

hole is the same [Fig. 1(b)]. The set pulse is a restricted smaller pulse of opposite polarity which reverses a portion of the flux in leg one and all the flux in leg 2. This is the set condition of the core [Fig. 1(c)]. The transfluxor is read by applying sine waves or pulses of alternating polarity to the small hole to sense the presence or absence of a set condition. If the core is set, the following occurs. First, a prime pulse, which produces a clockwise path around the small hole [see Fig. 1(d)], is applied to leg 3. A drive pulse is then applied to leg 3 to produce a counterclockwise flux path. The reversal of the direction of the flux path around the small hole by the prime and drive pulses generates an emf which is sensed by a winding on leg 3 [see Fig. 1(e) and 1(h)]. The set condition of the core and subsequent readout of a sense voltage is the equivalent of writing and reading a "1."

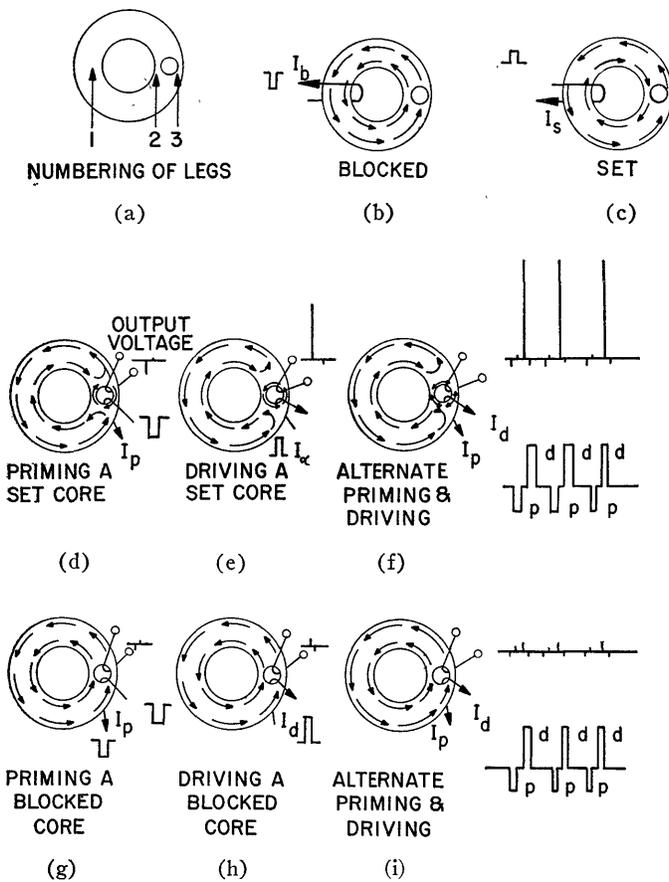


Fig. 1—Transfluxor operation.

In order to write a "0" in the core, the write operation must be modified so that the core remains blocked. This is accomplished by an inhibit winding through the large hole. At the same time the set pulse is applied, a half-current pulse of opposite polarity is applied to the inhibit winding. This half-pulse thus nullifies the set pulse and the core remains blocked. With the transfluxor blocked, the prime-drive pulses will not reverse any flux in the vicinity of the small hole and consequently will not generate any sense voltage [see Fig.

1(g)–1(i)]. Thus the generation or nongeneration of a sense voltage is the equivalent of reading a "1" or a "0" from the core.

The applicable addressing techniques for transfluxor memories are similar to those used for magnetic core memories. Each of the two transfluxor holes may have its own set of selection wires as shown in Fig. 2. Because there are separate addressing systems for both reading and writing, and because there is negligible interaction between aperture signals, it is possible to write in one location of the array while simultaneously reading in another location.

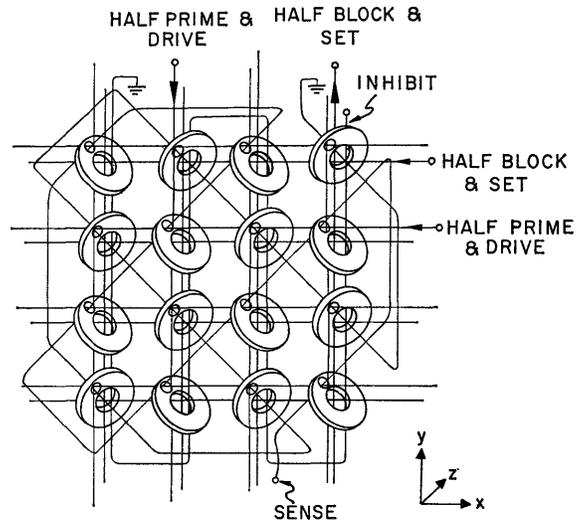


Fig. 2—Coincident current transfluxor array.

An important feature of the transfluxor is its ability to perform a nondestructive readout. This ability of the transfluxor memory system offers significant advantages to the digital computer. Because of the nondestructive nature of the transfluxor, there is no need to restore the transfluxor to its original state after interrogation. This results in the elimination of a major portion of the read/write cycle normally associated with memory systems, and the timing requirements are appreciably simplified. In effect the speed of the memory cycle is increased by approximately 2:1.

This nondestructive read characteristic is also significant in that there are no ruinous effects when a transient error occurs during a read operation. True, the information in a memory location may be read incorrectly due to a transient noise but this does not affect the contents of that memory location since the data do not have to be rewritten from the output. A computer that has a nondestructive memory can easily cope with transient noise by immediately repeating the read operation upon detection of a read error. The ability to read from the memory without destroying its contents is a desirable feature especially in real-time computer applications where the execution of the stored program must be reiterated indefinitely.

THE MULTILOAD TRANSFLUXOR

One of the attributes of the transfluxor is signal isolation between the windings of the small hole and the large hole. Through the proper physical placement of additional small holes in the ferrite body and a separate set of addressing wires and sense windings in each small hole, it is possible to obtain more than one independent output from the transfluxor without seriously affecting the signal isolation between holes. Thus it is possible to consider three, four, five, or more holes in the transfluxor, offering the designer an extremely versatile memory storage component. This new component is called the multiload transfluxor.

For the purpose of discussion, consider a multiload transfluxor with a large hole and two small holes. If only one of the small holes is used for readout, the performance of the multiload transfluxor is identical to the two-holed transfluxor. A possible configuration of a dual-load transfluxor and its flux patterns with only one of the small holes being pulsed for readout is shown in Fig. 3(a)-3(d).

The cores shown in Fig. 3 are pulsed by the coincident-current address method. This means that two pulses, one on the *X* wire and one on the *Y* wire, are needed to supply the total current required to generate the proper flux condition in the core. These flux patterns are nearly the same as those for the more conventional transfluxor. If the prime-drive cycle is simultaneously initiated in both small holes, the resulting flux distributions are shown in Fig. 4(a) and 4(b). Notice that there is minimum interaction between the holes. This permits the timing of the priming and driving pulses for each hole to be independent of the other.

The multiload transfluxor as used in a memory system has a separate set of addressing wires and sense windings in each small hole. Each set of wires is tied to independent address registers and loads. All address registers are capable of addressing randomly any core in the array and any core may be addressed by all the registers. Therefore the information stored in one core may be read out simultaneously to any or all loads, and any number of cores may be read independently into different loads.

With only one large aperture in each multiload transfluxor, there can be only one write/addressing source for each word. This limitation is imposed on the large aperture because each inhibit winding is common to every bit in its memory plane; so it can only represent one addressing source. Of course, multiple writing could be made possible by eliminating the inhibit winding, but this scheme involves many more control circuits. A simple and quite satisfactory method of accomplishing multiple writes is to use a split-memory system. The split-memory system is discussed later in this paper.

There are two techniques for selecting cores or core registers mentioned in this discussion: coincident-current selection and external word-selection (end firing).

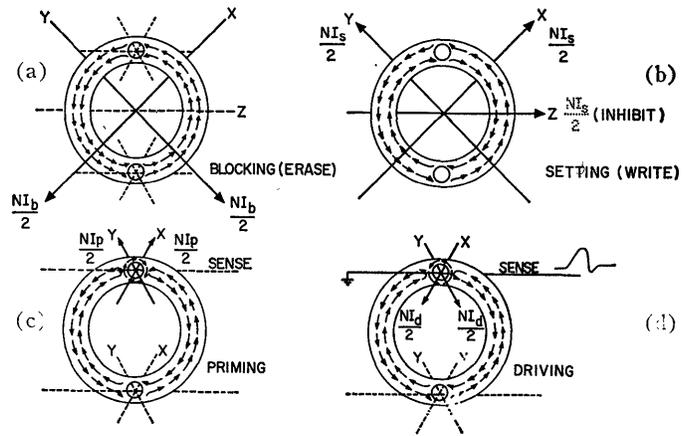


Fig. 3—Flux patterns of a dual-load transfluxor with only one small aperture being pulsed; single-load readout.

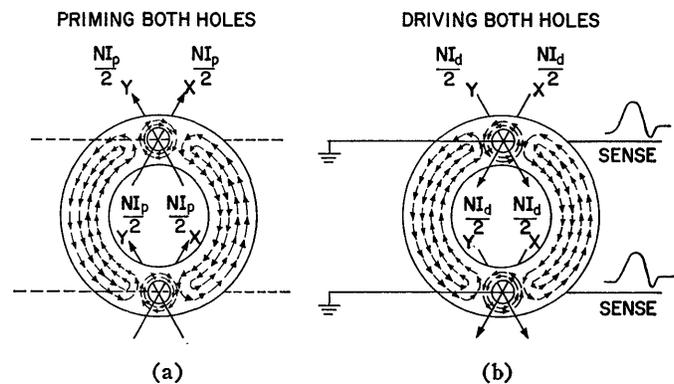


Fig. 4—Flux patterns of a dual-load transfluxor with both small apertures being pulsed; double-load readout.

The coincident-current technique is illustrated in Fig. 5. It is used here to address three cores simultaneously from three independent sources.

The wire leads in the array are designated by a code which describes the addressing source, the addressing axis, and the row or column number of that axis. For instance, *LY3* indicates that the origin of the lead is the third column in the *Y* axis of the *L* addressing source.

In this figure the *U* addressing source transmits half-current priming signals on the *UX4* and *UY3* leads. Accordingly, the upper apertures of cores 41, 42, 44, 45, 13, 23, and 33 all receive half-current pulses, but this half current is not sufficient to significantly disturb the flux pattern around the small aperture. This is a basic requirement of any coincident-current magnetic-core memory. However, core 43 receives half currents on both the *UX* and *UY* axis leads and the coincident summing of these currents causes a flux reversal in the vicinity of the small aperture. Thus, core 43, and only core 43, is primed for reading by the *U* addressing source. Similarly, the *L* addressing source selects core 13 for reading also, while *M* addressing source selects core 22 for writing.

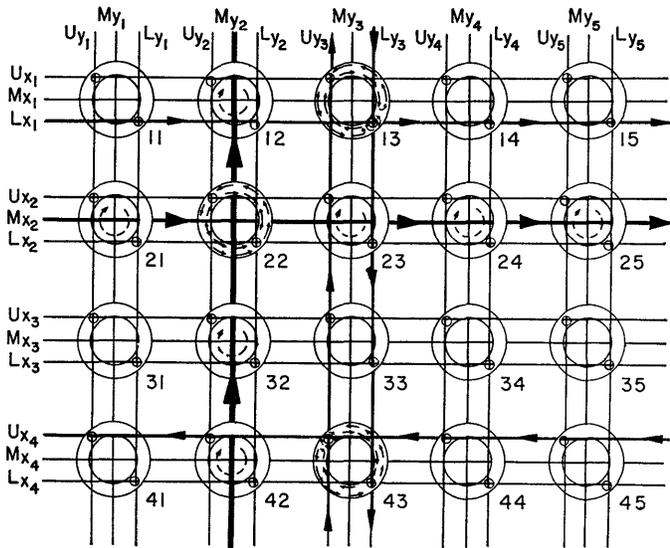


Fig. 5—The effects of simultaneously pulsing three cores; not shown: inhibit wires, sense wires to loads 1 and 2. This is only one plane of the memory.

In contrast to the coincident-current technique of selecting a word by summing the half currents of the X and Y axis leads, the external word-selection technique involves one wire that carries the full current to the selected word. A multiload transfluxor memory using the external word selection is shown in Fig. 6. Memory planes, composed of printed multiload transfluxor plates similar to those of a ferrite-plate memory, are stacked with the selection wires threaded in the word dimension. There are three apertures per bit, with one selection wire through each aperture, thus requiring only 3 physical wires per bit. All other windings can be printed on the ferrite plate, since they are all in the X-Y plane. These printed windings consist of one continuous inhibit winding per plate through each large aperture and two continuous sense windings per plate, one through each upper small aperture and the other through each lower small aperture. The X and Y addressing signals are supplied to an external circuit using either magnetic or transistor switches which provides the gating to select a specific word.

The advantages of this technique are as follows:

- 1) There is only one selection wire required per hole of a transfluxor. This eases the congestion and reduces the task of threading wires;
- 2) The noise problem due to the half-current selection pulses is eliminated; and
- 3) The specifications on the current amplitudes of block and drive pulses are less critical than in the normal array.

The multiload transfluxor for most memory design purposes imposes no additional restrictions than are required for toroidal cores. In fact, other selection methods may prove more advantageous than the ones described.

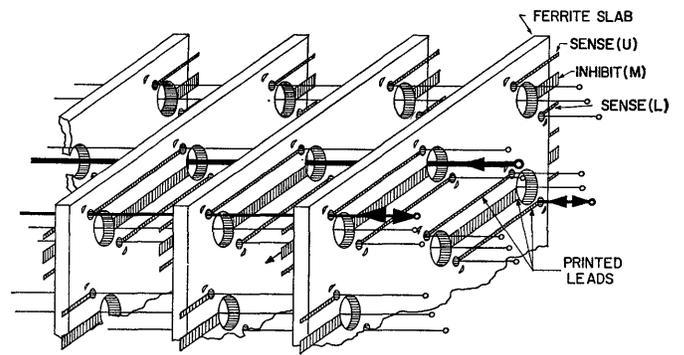


Fig. 6—Switch-driven memory-employing transfluxors.

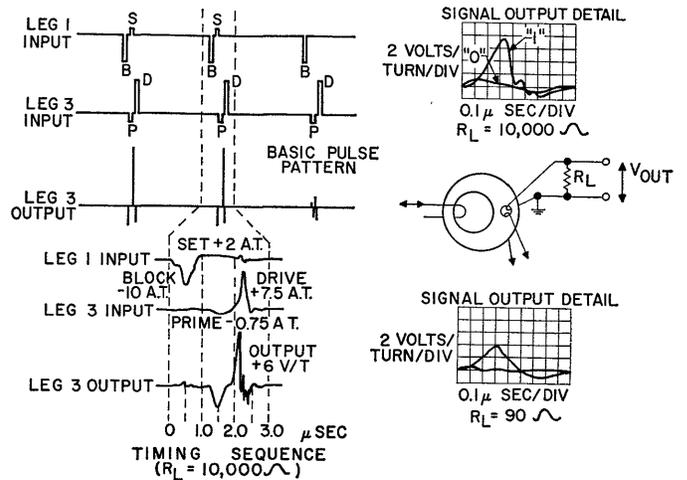


Fig. 7—A test program for a transfluxor of the large size.

LABORATORY TEST DATA

A considerable amount of experience has been gained by RCA in studies on the transfluxor itself and in development of a transfluxor memory for an airborne digital computer. The characteristics of transfluxors which make it ideally suited for high-speed digital memories were laboratory tested. The results are described here briefly; some data and waveforms are shown in Fig. 7.

The test setup used a large core (364-mil O.D.) which requires rather large currents. In practical memories smaller cores will be used with more reasonable currents. A memory is now being constructed using transfluxors with an outside diameter of 200 mils.

The currents used in these test circuits were abnormally large to accentuate certain conditions. The block pulse, for instance, was maintained at 10 amp turns to observe clearly the effect of the set pulse amplitude upon the output. The pulse pattern illustrated in Fig. 7 shows a driven output of 6 volts per turn (peak) with a pulse width of 0.18 μ sec (at 50 per cent of pulse amplitude) and a switching time of 0.30 μ sec. With such a large output it may be possible to eliminate the sense amplifier in certain applications.

Under laboratory conditions the prime-drive (non-destructive readout) cycle on a single core has been completed within $0.8 \mu\text{sec}$. A block-set sequence has been completed in $2.5 \mu\text{sec}$. Faster speeds appear to be possible, but could not be observed due to limitations imposed by the testing equipment.

The nondestructive readout properties were tested by setting one core and blocking another. A stream of bipolar (prime and drive) pulses were applied to both cores. After approximately 3,600,000,000 readouts, the outputs were the same as at the start of the test.

One of the problems common to ferrite devices is their temperature dependence. In most memories this problem is avoided by placing the entire core assembly in a temperature-maintained environment such as a 40°C oven. Typically a $\pm 3^\circ\text{C}$ range is maintained. Work has been done by Abbott and Suran² and by Bennion and Crane³ on stabilizing the logical transfluxor for wide temperature variations, but a temperature-maintained oven still appears advisable for coincident current memories.

COMPUTER MEMORY APPLICATIONS

There are four basic paths of communication between the memory and other sections of a digital computer:

- 1) New data are transmitted to the memory from the input unit;
- 2) Stored data are transmitted from the memory to the central computer for processing;
- 3) Results data are transmitted to the memory from the central computer;
- 4) Answer data are transmitted from the memory to the output unit.

In general, present-day computer memories can perform only one of these functions per memory cycle. This means that while the memory is communicating over any one of these basic paths, the other three communication paths are stymied. This inherent interference is a limiting factor in realizing the full operating potential of a digital computer.

The introduction of a computer memory that has the ability to communicate over all four of these paths simultaneously would offer effectively higher speeds of operation and other advantages. Such a memory is now possible through the use of the multiloop transfluxor.

Conventional Computer Organization

One suggested organization of a conventional digital computer utilizing a three-hole transfluxor memory is shown in the block diagram of Fig. 8. This memory unit is controlled independently from each of three sources: the output unit, the input unit, and the central computer. The interconnections between these three sources

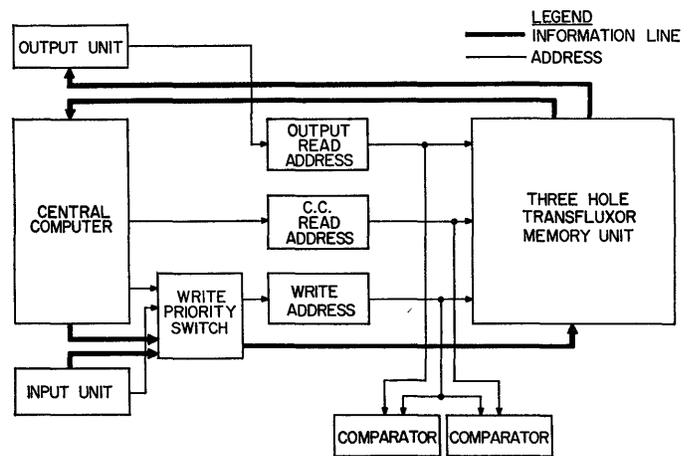


Fig. 8—Computer memory organization using a three-hole transfluxor memory.

and the memory unit are as follows. One of the two small apertures in each transfluxor is used for reading data from the memory to the output unit. The other small aperture in each core is used for reading data from the memory to the central computer. The large aperture in each transfluxor is used to write data into the memory unit from either the input unit or the central computer. Since both of these sources could simultaneously initiate memory-write instructions, a switching and priority unit must be provided to make a decision as to which should take precedence and control the flow of information accordingly. Until the execution of the selected write instruction is completed, the other write instruction is temporarily held up.

The logical operation of this proposed digital computer presents another minor restriction in that it is not permissible to read and write in the same memory location simultaneously. Two comparator checking circuits are required to check for coincidence of like read and write addresses. When like read and write addresses are detected, the procedure is to interrupt the performance of one of the instructions until the other is completed.

Except for these two restrictions, the output unit, the input unit, and the central computer are able to execute all read and write instructions independently, simultaneously, and asynchronously. They can function independently because each source is capable of selecting its own memory location regardless of the performance of the other source. They can function simultaneously because there is no problem of interference when data are read from two memory locations and written into another memory location at the same time. In fact, there is no interference when both read sources simultaneously select the same memory location. They can function asynchronously because good signal isolation makes different timing sources permissible.

Split Memory

The inability of a computer to write into the memory from both the input unit and the central computer

² H. W. Abbott and J. J. Suran, "Temperature characteristics of the transfluxor," IRE TRANS. ON ELECTRON DEVICES, vol. ED-4, pp. 113-119; April, 1957.

³ D. R. Bennion and H. D. Crane, "Design and analysis of MAD transfer circuitry," this issue, pp. 21-36.

simultaneously may be solved by another approach more compatible with the philosophy of parallel operations. It is a split-memory system shown in the block diagram of Fig. 9. In this memory system, one portion stores only the data from the input unit and the remaining portion stores only the central computer data. Thus there are effectively two separate memory units for writing information, but only one for reading by the central computer and output unit.

An important point to emphasize here is that the memory can be divided into two or more sections in order to accommodate simultaneous inputs to the memory from two or more sources. This division can be made in such a way as to satisfy the relative storage needs of the inputs. For instance, the memory can be divided so that 40 per cent of it is addressable for writing by the computer, 30 per cent of it is addressable for writing by a tape input, and 30 per cent of it is addressable for writing by a magnetic drum. It is obvious that there is a great number of possible arrangements of such a memory system.

An advantage of this digital computer organization is that there is no interference between the write instructions of the input unit and the central computer and thus no need for a switching and priority unit. As a result there is a definite saving in computer operating time. However, this advantage is gained only at the expense of a somewhat increased demand for memory-storage space and the introduction of two more comparator-checking circuits.

Parallel Operations

Some of the possible uses of a three-hole transfluxor have been given above. Additional small holes in the multiload transfluxor further increase its versatility. The possibility of having many small holes for reading purposes permits the realization of heretofore unachievable system designs. In real-time computer systems, it is desirable to be able to read large quantities of data continuously to output devices without interfering with other memory communications. In a missile control center, for example, there is a need for high-speed printers, tote-board display units, CRT display units, and tape units. Data transmitted to the CRT display units must be renewed at a rate sufficiently fast to eliminate flicker. The tote-board display must be able to exhibit pertinent information that is frequently updated to keep personnel abreast of the happenings. Concurrently, the tape units and high-speed printers must also be able to record in real time. If all of these output devices require access to the same data, the memory must consist of multiload transfluxors which have one small aperture for each output device. However, if the data required for the various output devices are different, the transfluxor memory unit can be split for reading in a manner similar to that discussed previously for writing. In this case, the memory would consist of three-hole transfluxors with one small aperture of each con-

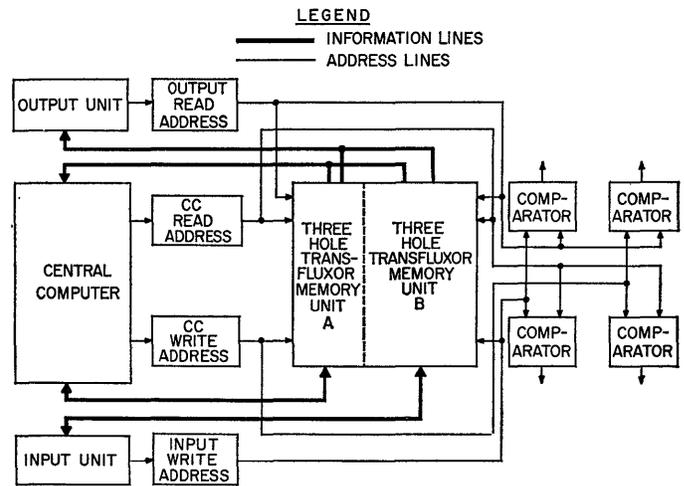


Fig. 9—Split memory system variation of Fig. 8.

nected to the one applicable output and the other small aperture connected to the central computer. With either configuration there is complete independence of reading by any or all output devices.

The development of the multiload transfluxor memory will make it practical to design and develop central computers that operate on multiple addresses in parallel. This means that instead of the conventional method, which must address each word required by the problem in sequence, all words may be addressed simultaneously in a transfluxor memory. Some novel computer designs based on this principle follow. A computer arithmetic unit could be designed to accept both the augend and addend data simultaneously. Computer logic based on a parallel three-address instruction could permit simultaneous reading of two operands to the arithmetic unit and writing of previous adder results into the memory unit. It is also conceivable to develop a sorting unit that simultaneously accepts and sorts three, four, or more inputs. Such a device would be a welcome addition to business computers where a major per cent of the time is spent sorting.

Common Memory

It is becoming increasingly important in the fields of science, computer checking, and real-time applications, where the speed of computation exceeds that permitted by one high-speed computer, to devise systems which link two computers together in order to handle adequately complex problems. One method, which is in use at Holloman Air Force Base, accomplishes this by using a conventional memory as the communicating link between computers. The basic operation of this type of system is illustrated in Fig. 10. Separate transfer memories *A* and *B* serve as links between the two computers. Computer *A* writes into transfer memory *A* any data which are needed by Computer *B*. Computer *B* may then read these data, process them, and either store the result in its own memory or in transfer memory *B* from which computer *A* may then read the data out. This

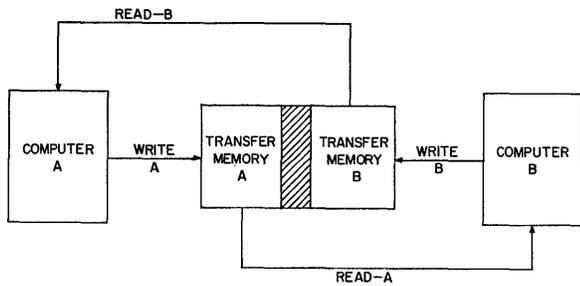


Fig. 10—Two-computer communication link using a transfer memory organization.

two-computer system offers certain advantages in speed over the operation of a single computer.

Another method of linking two computers is being used at the National Bureau of Standards. In this system one computer, called the Secondary, is a slave to the other computer, called the Principal. A similar organization of units within one computer network is used in the IBM 709 computer. Here the Data-Synchronizing Unit is comparable to the Secondary, and the Central Processing Unit is comparable to the Principal. The CPU can assign the DSU an input-output task and then proceed with its own program. The DSU proceeds independently, with the exception of memory access which it must share with the CPU, until it completes its assignment. In this system the program for the DSU is essentially wired in and the memory is time-shared. The National Bureau of Standards system is shown functionally in Fig. 11. The Principal has the authority to assign part of a problem to the Secondary. However, for the Secondary to perform its task it must be supplied with assignment instructions and data. This transfer of data interrupts the operation of the computer A memory and thus is time-consuming.

Now consider the advantages of using a "common multiload transfluxor memory" as the communication link between two computers. The functional linkage of such a system is shown in Fig. 12. The common memory is split into three sections to permit the execution of independent writing operations by both computers and any input-output devices. However, the memory is not split for reading, so both computers and other devices can have independent access to any data in the common memory. This common memory will permit three read and three write operations to occur simultaneously.

The significance of this system organization is that the common memory, in addition to acting as a communication link between computers, also serves as the memory-storage unit for each computer. As a result, there is no transfer time involved in communicating between computers. This saving of time is achieved because each computer independently has direct access to any data pertinent to the functioning of the other computer.

Another significant feature of this organization is that the common memory will need less storage capacity than the total amount required by two individual com-

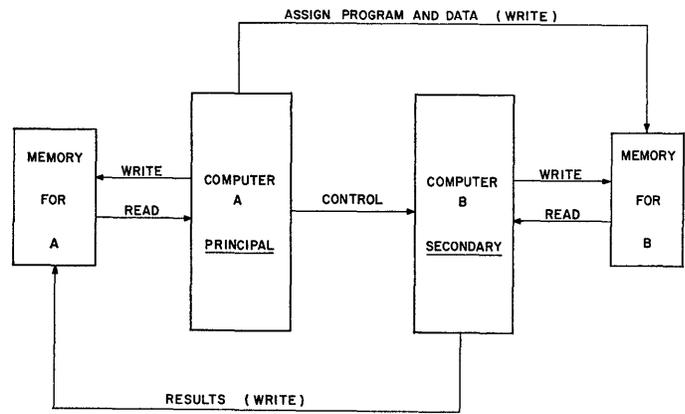


Fig. 11—Two-computer communication link using a Principal-Secondary computer organization.

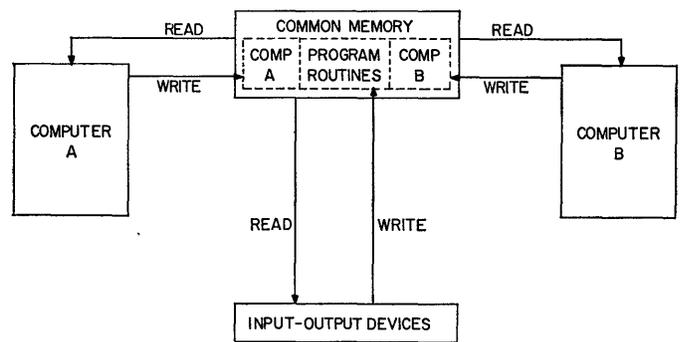


Fig. 12—Two-computer communication link using a common multiload transfluxor memory organization.

puters. This reduction in memory storage capacity can be realized because

- 1) No transfer memory-storage units are required since each computer has direct access to any data pertinent to the other computer; and
- 2) Basic subroutines, stored constants, and input data are available to both computers, since both have access to all the memory-storage locations. Thus the duplication of common data is eliminated.

The concept of a master-slave relationship can be carried over to a common memory system but with increased flexibility. The necessity of assigning large blocks of data and several tasks at once to the Secondary in order to keep it operating continuously is practically eliminated. Since all data are directly available to both computers, only the initial address of several stored programs needs to be assigned. In addition the roles of Principal and Secondary can be interchanged at will so that either computer can assign a task to the other.

The system organization as proposed in Fig. 12 requires a four-hole transfluxor memory unit. With a five or more hole transfluxor memory, it is possible to expand this organization to incorporate three or more computers. The more computers in the network, the more efficient the common memory becomes.

CONCLUSION

The basic mechanisms of transfluxor operation have been shown and a few examples given on how this versatile component may be used.

The multiload transfluxor is constructed by placing additional reading apertures in the core and wiring each hole for separate addressing. This permits many read operations to take place throughout the memory at the same time. Each readout is delivered to its own independent load. The nondestructive read property eliminates the rewrite time associated with conventional

core memories; this feature permits cutting the read time in half.

It appears that, by utilizing these properties, considerably more flexibility and speed can be built into a transfluxor memory. The independent operation of the various parts of the memory would facilitate communication between sections of a computer or between two computers and would permit all parts of a computer network to operate without delays due to memory time-sharing. This would be a major advance in computer design.

Design and Analysis of MAD Transfer Circuitry*

D. R. BENNION† AND H. D. CRANE†

I. INTRODUCTION

THIS is the second in a series of papers¹ concerned with a technique for performing combinatorial and sequential digital logic with magnetic elements and connecting wires only. These elements are termed MAD's (Multi-Aperture Devices). For clarity, in the first paper the basic techniques were described in terms of simple circuit structures which do not represent the best that can be achieved in the way of operational properties. The object of this paper is: 1) to present circuit techniques for significantly improving the circuit operation; and 2) to present experimental and analytic results which are pertinent to an understanding of the coupling loop operation.

The basic coupling loop and clock cycle are briefly reviewed in the next section.

II. REVIEW OF BASIC COUPLING LOOP AND CLOCK CYCLE

The circuits discussed here use only POSITIVE MAD elements¹ (although the results are applicable to circuits using other types of MAD elements). Each element has at least two small apertures, one used for an output winding and one for an input winding. The output winding of one element connects with the input winding of another to form a coupling loop, and in this way a pair of electrically connected elements is formed. As information is shifted along a chain of elements, each

element alternately plays the role of a receiver and transmitter.

Output Aperture

An element can be in either the Set (binary *one*) or Clear (binary *zero*) state, Fig. 1. Typical ϕ_T - F_T (where F_T is the driving mmf $N_T I_T$) curves for the output aperture of a transmitter for these two states are illustrated in Fig. 1(c). If the element is in the Set state, Fig. 1(b), then flux changes locally about the output aperture in response to small values of mmf F_T , whereas if the element is in the Clear state, Fig. 1(a), flux can change only about a path enclosing both the output and central apertures. Because of the longer path length in the latter case, larger switching mmfs are required. (Subscript T indicates that this winding is connected with the transmitter end of a coupling loop.)

Input Aperture

A receiver element is always cleared to its *zero* state before transmission into it. It operates then only along a ϕ_R - F_R Clear curve (where F_R is the mmf $N_R I_R$) which is essentially the same as the Clear-state curve for the transmitter, since the relevant path lengths are the same, Fig. 2(a) and 2(b).

An important property of a system in which windings connect with apertures, as indicated here, is that once an element is Set, it is impossible to Clear it from any aperture winding. In Fig. 2(c), an element is shown Set, as a result of current I_R . In Fig. 2(d), the same element is shown after a subsequent "negative set" current, $-I_R$. Note that as a result of the negative set current flux changes only *locally* about the input aperture *without* disturbing the flux about the output aperture.

* This work was carried out at the Stanford Research Inst., Menlo Park, Calif., under the sponsorship of the Burroughs Corp. Res. Center, Paoli, Pa.

† Stanford Res. Inst., Menlo Park, Calif.

¹ H. D. Crane, "A high-speed logic system using magnetic elements and connecting wire only," Proc. IRE, vol. 47, pp. 63-73; January, 1959.

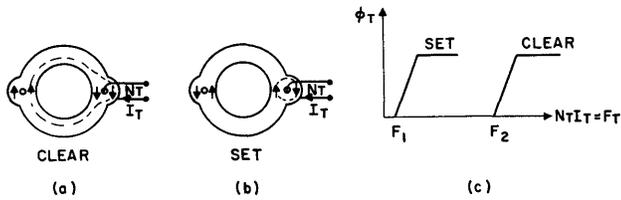


Fig. 1—Output aperture properties.

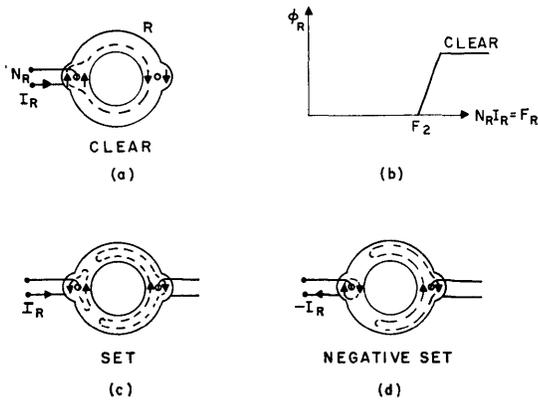


Fig. 2—Input aperture properties.

Coupling Loop

To form a coupling loop, the output winding of one element is connected to the input winding of the next (Fig. 3). If the transmitter is in the *zero* state, then transmission is trivial in the sense that the receiver is already in the *zero* (reference) state. It is necessary then that the Advance current I_A be controlled in value so that in this case no flux switches in either element. With the transmitter in the *one* state, it is desirable that I_R (and hence I_A) be as large as possible to facilitate flux transfer. For best transmission then, it is desirable to make I_A as large as allowable. The largest allowable value I_A is determined by the *zero* state condition, and is equal to

$$I_A^m = I_T^m + I_R^m = \frac{F_2}{N_T} + \frac{F_2}{N_R}$$

where F_2 is the Clear-state threshold mmf (Figs. 1 and 2). In this case, assuming that the current divides into $I_T = F_2/N_T$, and $I_R = F_2/N_R$, both devices are brought just to their thresholds F_2 , but no flux is switched. Thus a *zero* is "transmitted" as a *zero*. If the transmitter is in the Set state, however, the transmitter has a much lower threshold, F_1 , so that $I_R \gg I_T$ and the Advance current I_A causes flux switching in both the transmitter and the receiver. Thus the receiver is set to the *one* state.

Clock Cycle

A clock cycle for operating a two-MAD-per-bit shift register is briefly reviewed, Fig. 4. The elements are considered in two groups, arbitrarily labelled O (odd)

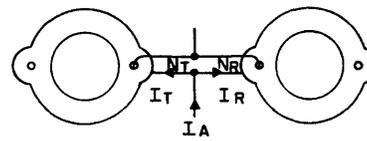


Fig. 3—Basic coupling loop.

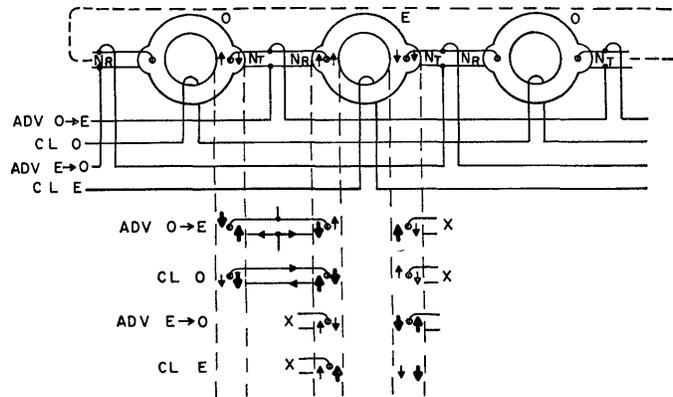


Fig. 4—Clock cycle.

and E (even). Information is stored in one group while the other is being cleared. The information is then shifted to the cleared group, while the other group is cleared, etc.

To get started in the cycle, assume the data are initially stored in the O elements. The ADV O→E pulse shifts the information to the E elements. The O elements are then explicitly cleared by the CL O pulse. The ADV E→O pulse shifts the information back to the O elements, and the E elements are then cleared by the CL E signal, etc. Thus the basic cycle is

$$\dots, \text{ADV O} \rightarrow \text{E}, \text{CL O}, \text{ADV E} \rightarrow \text{O}, \text{CL E}, \dots$$

By tracing the details of the data transmission, one may see how the MAD elements and basic clock cycle work together to yield unilateral transmission. As previously indicated, transmission of a *zero* is trivial. Transfer of a *one* is followed in detail in Fig. 4. The statements made with reference to this particular *one* transfer are true as well for every O→E loop simultaneously transferring a *one*.

Flux switched in any leg as a result of a particular pulse is indicated by a heavy arrow. The ADV O→E pulse switches flux locally about the output aperture of the O element and causes the E element to be set. The CL O pulse then clears the O element and in so doing switches flux through the output winding. This results in a loop current flow that negatively sets [see Fig. 2(d)] the E element (receiver) *without* affecting the flux state about the output aperture of the E element. Note that neither the ADV O→E nor CL O pulse causes any flux to be switched in the output leg of the E element (indicated by the crosses opposite the output winding), eliminating thereby the need for the conventional series coupling diode. ADVANCE E→O shifts the binary *one*

to the next O element in the direction of transmission, and CL E then clears the E element. Note again that neither the ADV E→O nor CL E pulses cause any flux to be switched in the input winding of the E element, eliminating the need for the conventional shunt diode to prevent backward transmission. Thus unilateral data flow is achieved.

Unity Turns Ratio Operation

Coupling loops can be successfully operated under conditions where $N_T \geq N_R$. The condition $N_T = N_R$ is particularly interesting for shift register circuits since the coupling loops are inherently symmetrical, and shifting may therefore be made to occur in either direction by control of the clock sequencing. Furthermore, the special case $N_T = N_R = 1$ results in very simple mechanical assemblies. The condition $N_T < N_R$ does not appear to have any particular advantage, but the condition $N_T > N_R$ is important as a means for obtaining flux gain when bi-directional coupling loops are not required. Operation with unity-turns-ratio, *i.e.*, $N_T = N_R$, is discussed in Sections V and VII.

In order to simplify the mathematical relations all circuit descriptions will be for symmetrical unity-turns-ratio coupling loops. The extension to the more general case $N_T > N_R$ is relatively straightforward.

Advance Current Range

It is well known that magnetic circuits that operate in the region of threshold are inherently slower and less tolerant to clock current variations than would be similar circuits not so limited. It is important therefore to determine and to take advantage, insofar as possible, of all techniques for improving these allowable operating ranges.

Relations for advance current range are derived below for the coupling loop circuit of Fig. 3, using a very simple model in which we assume 1) that the *one*, *zero* ϕ_T - F_T curves have vertical steps at threshold F_1 and F_2 (Fig. 5); 2) that if a transmitter element is in the *zero* state, then the circuit must be limited so that as a result of the Advance pulse, the receiver is not brought over its Clear state threshold F_2 ; and 3) that if the transmitter is in the *one* state, then as a result of the Advance pulse, the receiver element and transmitter element are completely switched. For this latter condition, the receiver must receive a net drive of at least F_2 and the transmitter a net drive of at least F_1 . Although this model is extremely inadequate (see Sections VII and VIII), it is very useful for comparative estimating purposes.

For the circuit of Fig. 5(a), the maximum value of Advance current I_A^{\max} is determined by the *zero* transfer condition. In this case, $I_A^{\max} = 2(F_2/N)$ where F_2/N is the current required in each branch to just bring its corresponding element to its Clear state mmf threshold F_2 .

The minimum value of Advance current, I_A^{\min} , is de-

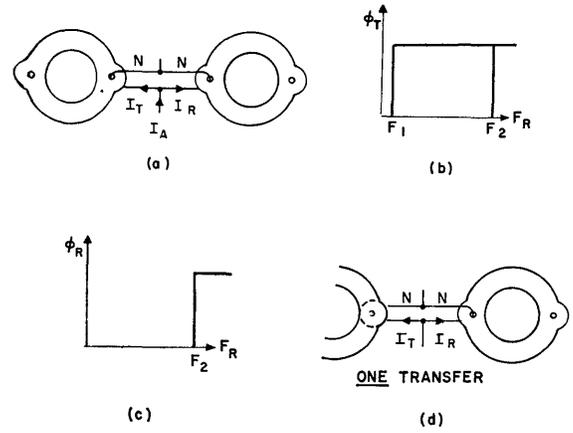


Fig. 5—Switching model for basic coupling loop.

termined by the *one* transfer condition. The equivalent circuit of Fig. 5(d) can best be used for visualizing the relations, so that

$$I_A^{\min} = I_T^{\min} + I_R^{\min} = \frac{F_1}{N} + \frac{F_2}{N}.$$

Therefore, the percentage range R for the Advance current, defined as

$$R = \frac{(I_A^{\max} - I_A^{\min}) \times 100}{I_A^{\text{av}}}$$

is equal to

$$R = 2 \left[\frac{F_2 - F_1}{3F_2 + F_1} \right] \times 100. \quad (1)$$

For comparative purposes, it is interesting to consider the limiting case when $F_1 = 0$. Under these conditions, the limiting value of range, R^0 , is

$$R^0 = 67 \text{ per cent.} \quad (1a)$$

In the sections to follow, circuits that exhibit significantly greater range will be introduced and the expressions for range determined for these may be compared with the relation derived here.

It may be noted that in these circuits only the Advance current range is of concern since the Clear current range is essentially unlimited, as long as it is above the minimum value required for adequate clearing of the elements.

Switching Speed

It is well known that the rate of switching of a “square loop” magnetic material is approximately proportional to the amount of (excess) drive, over and above the threshold value. Thus, two thin rings (of the same material) of radii r_1 and r_2 would switch at the same rate if driven with mmf’s in the ratio r_1/r_2 .

For the switching problem at hand, we will estimate the switching speed with the Advance current set in the

TABLE I
SUMMARY OF RANGE AND SPEED RELATIONS DERIVED IN TEXT

Figure	Condition	R [Range in fraction of 100 per cent]	R^0 [R for $F_1 = 0$]	Number of switching thresholds n measured at I_A^{sv}	$n^0 - 1$ (n^0 is value for $F_1 = 0$)
5		$2 \left(\frac{F_2 - F_1}{3F_2 + F_1} \right)$	$\frac{2}{3}$	$\frac{3F_2 + F_1}{2(F_1 + F_2)}$	$\frac{1}{2}$
9	let $F_2^B = F_2 + F_B$	$2 \left(\frac{F_2^B - F_1}{3F_2^B + F_1} \right)$	$\frac{2}{3}$	$\frac{3F_2 + F_B + F_1}{2(F_1 + F_2)}$	$\frac{1}{2} \left(\frac{F_2^B}{F_B} \right)$
10	$N \geq 4N_B$	$2 \left[\frac{(F_2 - F_1)N + 2F_1N_B}{(3F_2 + F_1)N - (4F_2 + 2F_1)N_B} \right]$	$\frac{2N}{3N - 4N_B}$	$\left[\frac{(3F_2 + F_1)N - (4F_2 + 2F_1)N_B}{2(F_1 + F_2)(N - 2N_B)} \right]$	$\frac{N}{2(N - 2N_B)}$
	$N \leq 4N_B$	$2 \left[\frac{F_2N - (2F_2 + F_1)N_B}{F_2N + F_1N_B} \right]$	$\frac{2(N - 2N_B)}{N}$	$\left[\frac{F_2N + F_1N_B}{2(F_1 + F_2)N_B} \right]$	$\frac{N - 2N_B}{2N_B}$
	$N = 4N_B$	$2 \left[\frac{2F_2 - F_1}{4F_2 + F_1} \right]$	1	$\left[\frac{4F_2 + F_1}{2(F_1 + F_2)} \right]$	1
11	$F_{dc} \geq \left(\frac{4N_B}{N} - 1 \right) F_2$	$2 \left[\frac{(F_2 - F_{dc} - F_1)N + 2F_1N_B}{(3F_2 - 3F_{dc} + F_1)N - (4F_2 - 4F_{dc} + 2F_1)N_B} \right]$	$\frac{2N}{3N - 4N_B}$	$\left[\frac{(3F_2 - F_{dc} + F_1)N - (4F_2 + 2F_1)N_B}{2(F_1 + F_2)(N - 2N_B)} \right]$	$\frac{(F_2 - F_{dc})N}{2F_2(N - 2N_B)}$
	$F_{dc} \leq \left(\frac{4N_B}{N} - 1 \right) F_2$	$2 \left[\frac{(F_2 + F_{dc})N - (2F_2 + F_1)N_B}{(F_2 + F_{dc})N + (F_1 - 2F_{dc})N_B} \right]$	$2 \left[\frac{(F_2 + F_{dc})N - 2F_2N_B}{(F_2 + F_{dc})N - 2F_{dc}N_B} \right]$	$\frac{(F_2 + F_{dc})N + F_1N_B}{2(F_1 + F_2)N_B}$	$\frac{N(F_2 + F_{dc})}{2N_B F_2} - 1$
	$F_{dc} = \left(\frac{4N_B}{N} - 1 \right) F_2$	$2 \left[\frac{2F_2 - F_1}{4F_2 - 2F_{dc} + F_1} \right]$	$\frac{2F_2}{2F_2 - F_{dc}}$	$\left[\frac{4F_2 + F_1}{2(F_1 + F_2)} \right]$	1
	$N = 2N_B$ and $F_{dc} = F_2$	$2 \left[\frac{2F_2 - F_1}{2F_2 + F_1} \right]$	2	$\left[\frac{4F_2 + F_1}{2(F_1 + F_2)} \right]$	1
12	Relations same as in Fig. 10 with N replaced by $N' + 2N_B'$ and N_B replaced by N_B'				
13	Relations same as in Fig. 10 with N replaced by $2(N_1 + N_2)$ and N_B replaced by N_1				

middle of its calculated range, *i.e.*, with the Advance current set at

$$I_A^{\text{av}} \left(\text{that is, } \frac{I_A^{\text{max}} + I_A^{\text{min}}}{2} \right).$$

With the transmitter and receiver windings directly in parallel, then at every instant during the switching process,

$$N_T \frac{d\phi_T}{dt} = N_R \frac{d\phi_R}{dt}$$

where ϕ_T and ϕ_R are the switched fluxes in the T and R elements, respectively. With equal turns, $N_T = N_R$, the branch currents will divide so as to cause the same number of thresholds, n , in the transmitter and receiver. This will result in equal switching rates, *i.e.*, $\dot{\phi}_T = \dot{\phi}_R$. Furthermore, according to the ϕ - NI model of Fig. 5, I_T and I_R will be constant during the switching process (see Fig. 23). The number of thresholds in the transmitter is equal to $n_t = NI_T^{\text{av}}/F_1$ and in the receiver is equal to $n_r = NI_R^{\text{av}}/F_2$ where $I_T^{\text{av}} + I_R^{\text{av}} = I_A^{\text{av}}$. By setting $n_t = n_r = n$, then we find the relation

$$n = \frac{3F_2 + F_1}{2(F_2 + F_1)}. \quad (2)$$

In the limiting case, $F_1 = 0$, then

$$n^0 = 1.5 \text{ thresholds.} \quad (2a)$$

This value of n^0 could easily have been predicted by noting that

$$I_A^{\text{av}} = \frac{3F_2}{2N}.$$

For the ideal case assumed here, during *one* transfer, the entire Advance current flows into the receiver. Thus, $n^0 = NI_A^{\text{av}}/F_2$.

The relations for range, for n , and for switching speed proportional to $n-1$ derived here are listed in Table I, along with corresponding relations for the circuits derived below. (It should be kept in mind that speed, $1/\tau$, where τ is switching time, is proportional to the *excess* number of thresholds of drive and hence $n-1$, since n was defined as the total number of thresholds.)

Motivation

It is clear from the above relations for R and n that one way to improve the speed and advance current range is to make F_2 large relative to F_1 . This implies the use of a large diameter element, compared to the diameter of the small aperture. However, a large element is undesirable for many obvious reasons. It is fortunate, though, that the equivalent of a large element can be obtained by appropriate biasing arrangements. But even further, the biasing arrangements can improve the operation even beyond what would be expected merely of a "larger" element.

Another way to improve the speed, and in general the range as well, is to provide as much drive as possible about the output aperture of the transmitter. Under the ideal condition of threshold $F_1 = 0$, then it requires no current to switch the transmitter, and $I_R = I_A$ during *one* transmission. This is equivalent to saying that all of the mmf appearing about the transmitter output aperture in the *zero* state is transferred to the receiver via the coupling loop during *one* transmission.

Consider again the elementary coupling loop of Fig. 6 with I_A^{max} applied, Fig. 6(a) and 6(b). During *zero* transfer, both the transmitter and receiver are stressed with a threshold mmf F_2 . During *one* transfer, at least ideally, the receiver becomes stressed by $2F_2$. If appropriate circuitry can provide higher stresses in the transmitter (around the output aperture) during *zero* transfer, then the receiver stress during *one* transmission is correspondingly higher. Generally, this can be achieved in two ways, Fig. 6(c) and 6(d). In Fig. 6(c), the coupling loop applies an mmf of $2F_2$, which ordinarily would tend to set a *zero* transmitter. However, this is prevented by a bias equal to F_2 so that the net setting mmf about the central aperture in the *zero* state is still only F_2 . Furthermore, the bias F_2 is not strong enough to clear a set transmitter during *one* transfer. Thus, with this arrangement, a stress of $2F_2$ can be added to the receiver during *one* transfer, resulting in a total receiver stress of $3F_2$.

In the arrangement of Fig. 6(d), a *zero* state stress of $2F_2$ is also achieved in the transmitter, but this is obtained by applying an extra clear direction drive of magnitude F_2 on the inner leg.

Alternate schemes combining these approaches may be used as well, as indicated in Fig. 6(e), where k , which may have any value, but practically will lie in the range $1 \leq k \leq 2$, is an arbitrary constant. In this case, the net stress about the output aperture is $2F_2$, independent of k . Note that the circuit of Fig. 6(c) results from $k = 2$, and the circuit of Fig. 6(d) for $k = 1$.

The initial circuit arrangements that follow are motivated from these concepts. However, as these circuits develop other concepts arise which lead to still other circuits.

III. SCHEMES UTILIZING ONLY A SINGLE WINDING IN THE INPUT AND OUTPUT APERTURES

Transmitter Bias

With a current I_{BT} in the clear direction through a winding of N_{BT} turns linking the central aperture of the transmitter (Fig. 7), the Advance mmf $N_T I_T$ must first overcome the mmf $F_{BT} = N_{BT} I_{BT}$ before it can switch flux about the central aperture of the transmitter. This "bias," therefore, effectively increases the magnitude of the threshold F_2 by the value F_{BT} . The equivalent threshold is, therefore, $F_2' = F_2 + F_{BT}$. The threshold F_1 is not affected, however, since the bias current does not link the flux paths that are local about the output aperture. Thus, to the electrical circuit the element ap-

pears to be larger than it actually is. The magnitude of bias is limited to a value $N_{BT}I_{BT} = F_2$ or the bias mmf itself would tend to clear a Set transmitter. Thus, with maximum bias, $F_{BT} = F_2$, the element appears to be twice as large in diameter, and the effective threshold F'_2 is essentially twice F_2 .

Receiver Bias

With the transmitter in the *zero* state, the manner in which I_A initially tends to divide between transmitter and receiver branches depends upon the branch inductances L_T and L_R , which are attributable mainly to the saturation permeability of the ferrite. However, the final division of current depends only on branch wire resistances R_T and R_R . For identical elements and with turns ratio N_T/N_R , the ratio of branch inductances is $(N_T/N_R)^2$. It is desirable to make the resistance ratio the same in order to eliminate transient overshoots in the branch currents. At the same time, it is desirable for R_T/R_R to be in the ratio N_T/N_R (in the case of no transmitter bias) in order for the final mmf's applied to transmitter and receiver to be equal. The above two conditions appear to be incompatible for $N_T/N_R \neq 1$ or even for $N_T/N_R = 1$ if transmitter bias is being used. However, by application of bias $N_{BR}I_{BR}$ to the receiver (Fig. 8), an additional term is added to the receiver mmf and the above conditions can both be satisfied. In this sense then, receiver bias serves as a free parameter for simultaneous satisfaction of one additional loop condition.

For the case $N_T = N_R = N$, it is clear that receiver bias should be equal to transmitter bias for proper balance. At the same time, this leads to a symmetrical loop capable of bidirectional transmission.

It is important to note that whereas the transmitter bias must be limited to, at most, the value F_2 , the receiver bias can be of arbitrary value. This is so because the receiver current can only cause flux switching about the central aperture of the receiver, and the only concern, therefore, is with the net mmf $(N_R I_R - N_{BR} I_{BR})$ about the central aperture. During *one* transmission any flux switching about the central aperture of the transmitter in the Clear direction is detrimental.

Simultaneous Transmitter and Receiver Bias

With bias $F_B = F_{BT} = F_{BR}$ applied, the maximum value of I_A^{\max} is [Fig. 9(a)]

$$I_A^{\max} = \frac{2(F_2 + F_B)}{N} = \frac{2F_2^B}{N}$$

where $F_2^B = F_2 + F_B$, and from Fig. 9(b), I_A^{\min} is

$$I_A^{\min} = \frac{F_1}{N} + \frac{F_2 + B}{N} = \frac{F_2^B + F_1}{N}$$

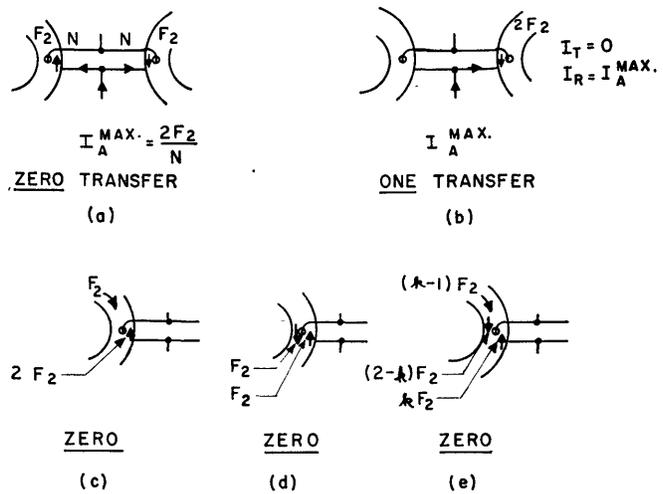


Fig. 6—Basic biasing arrangements.

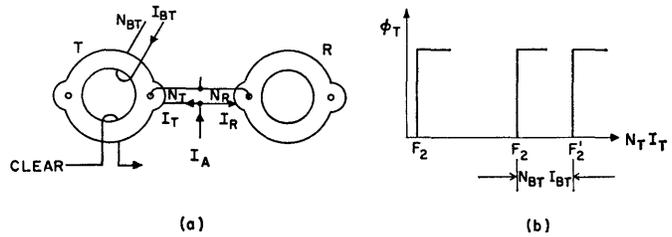


Fig. 7—Transmitter bias.

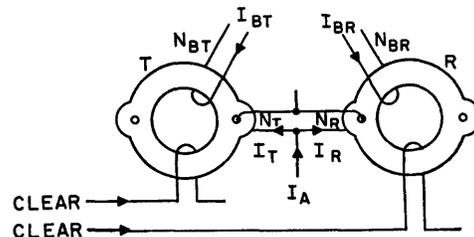


Fig. 8—Receiver bias.

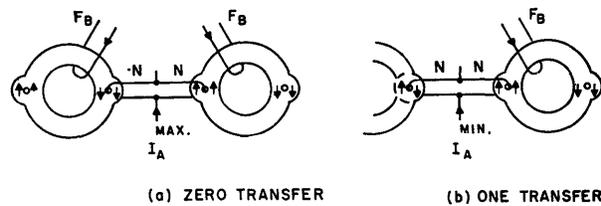


Fig. 9—Circuit using separate transmitter and receiver bias.

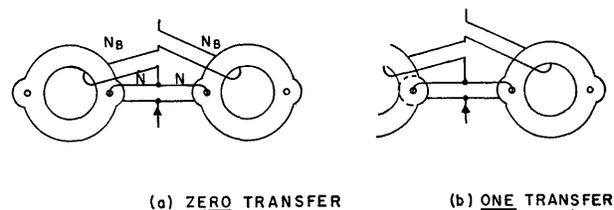


Fig. 10—Circuit using compatible transmitter and receiver bias.

Therefore, the range R is

$$R_{SB} = 2 \left[\frac{F_2^B - F_1}{3F_2^B + F_1} \right] \times 100 \quad (3)$$

and, again for the limiting case $F_1 = 0$,

$$R_{SB}^0 = 67 \text{ per cent} \quad (3a) \quad \text{or}$$

where the subscript SB implies that this range is for circuit with simple bias. The switching factor n can be found in the same way as for (2); thus,

$$n_{SB} = \frac{NI_T^{av}}{F_1} = \frac{NI_R^{av} - F_B}{F_2} = \frac{3F_2 + F_B + F_1}{2(F_2 + F_1)} \quad (4)$$

and

$$n_{SB}^0 = 1.5 + \frac{F_B}{2F_2} \quad (4a)$$

It is interesting to note that with bias mmf F_B applied, the range is increased exactly as though F_2 were replaced by a larger element of dimension $F_2^B = F_2 + F_B$. However, the switching speed is improved beyond just the simple substitution of F_2^B for F_2 in (2).

Thus, by the use of bias in this manner, the switching properties are significantly improved. This bias could be provided from a dc source, or from a pulse source. In the next section, it is indicated that still further advantage is obtained by having the bias mmf provided by the Advance current itself.

Compatible Bias

In order to keep the number of current sources to a minimum and also to insure that the receiver bias operates only during the Advance current (if, by design, it should be greater than F_2), it is desirable to have the Advance current itself provide the bias as indicated in Fig. 10.

In this circuit arrangement, there are two conditions for I_A^{\max} . For $N \geq 4N_B$, then I_A^{\max} is limited by the *zero* transfer, in which case

$$I_A^{\max} = \left[\frac{F_2 + N_B I_A^{\max}}{N} \right]_T + \left[\frac{F_2 + N_B I_A^{\max}}{N} \right]_R$$

or

$$I_A^{\max} = \frac{2F_2}{N - 2N_B}.$$

If I_A^{\max} were larger than this value, then the transmitter and receiver would overrun their threshold during *zero* transfer.

For $N \leq 4N_B$, then I_A^{\max} is limited by the *one* transfer, in which case

$$I_A^{\max} = \frac{F_2}{N_B}.$$

For values larger than this, the transmitter would tend to be cleared during transfer. In either case, I_A^{\min} is determined from Fig. 10(b), where

$$I_A^{\min} = \left[\frac{F_1}{N} \right]_T + F_2 + N_B I_A^{\min} \Big|_R$$

$$I_A^{\min} = \frac{F_2 + F_1}{N - N_B}.$$

The resulting relations for R_{CB} , n_{CB} , R_{CB}^0 , and n_{CB}^0 are given in Table I, where the subscripts CB imply the compatible bias use. Note that R_{CB} and n_{CB} are functions of N and N_B . Of course, with $N_B = 0$ these relations reduce to the same relations as given in (1) and (2).

Maximum R_{CB}^0 and n_{CB}^0 occur for $N = 4N_B$ in which case

$$R_{CB}^0 = 100 \text{ per cent} \quad (5)$$

$$n_{CB}^0 = 2. \quad (6)$$

The increased range obtained with compatible bias can be explained in terms of "moving thresholds." That is, with compatible bias, the effective transmitter threshold ($F_2 + N_B I_A$) is itself a function of I_A . Hence, as I_A increases from the center of its range and, therefore, tends to approach the threshold F_2 , the effective threshold value itself tends to increase, reducing considerably the overrunning effect. The same stabilization results in the case of Advance current reduction as well, as far as the receiver is concerned. Thus, as I_A decreases from the center of its range, the receiver moves further from threshold, but the effective threshold itself is decreasing. Note that with $N = 4N_B$, and with I_A^{\max} applied, the circuit of Fig. 10 exactly matches the conditions of Fig. 6(c).

Thus, significant improvement in R and n is obtained by the use of compatible bias. However, in practical circuits, N is greater than N_B , and since the minimum value of N_B is unity, single turn coupling loops cannot effectively be used.

Counter Bias

In order to improve the stabilization and, therefore, increase the Advance current range still further, it is necessary to increase the feedback effect; *i.e.*, it is necessary to make the bias "move" even faster as a function of Advance current. This may be achieved by increasing the bias turns N_B relative to the coupling loop turns N . However, for a given value of I_A and N , as the bias turns are increased the transmitter bias increases, and soon overruns 100 per cent. This effect may be compensated for by use of a dc bias of opposite sign (*i.e.*, counter bias), as indicated in Fig. 11.

As in the previous case there are two conditions for I_A^{\max} . For the case where $F_{dc} \geq ((4N_B/N) - 1)F_2$, then I_A^{\max} is determined from the *one* transfer case.

$$I_A^{\max} = \left. \frac{F_2 + N_B I_A^{\max} - F_{dc}}{N} \right]_T + \left. \frac{F_2 + N_B I_A^{\max} - F_{dc}}{N} \right]_R$$

or

$$I_A^{\max} = \frac{2(F_2 - F_{dc})}{N + 2N_B}.$$

The minimum current I_A^{\min} is determined for *one* transfer and is

$$I_A^{\min} = \left. \frac{F_1}{N} \right]_T + \left. \frac{F_2 + N_B I_A^{\min} - F_{dc}}{N} \right]_R$$

or

$$I_A^{\min} = \frac{F_2 - F_{dc} + F_1}{N - N_B}.$$

These combine into the relations for R_{dc} and n_{dc} given in the table.

For the alternate condition $F_{dc} \leq (4N_B/N - 1)F_2$, then I_A^{\max} is given simply from the *one* transfer case as

$$I_A^{\max} = \frac{F_2 + F_{dc}}{N_B}.$$

The relation for I_A^{\min} is the same as before.

The range is a maximum where $F_{dc} = ((4N_B/N) - 1)F_2$. Nominally, the dc bias is limited in magnitude to the value of F_2 . For this value of dc bias maximum range occurs for $N/N_B = 2$, and is equal to 200 per cent. The corresponding value of switching factor is $n^0 = 2$. Thus, ultimately, a 2 to 1 improvement in operating range is achieved.

This improvement in operating range is obtained at the expense of an extra dc bias. However, no extra windings are required since the dc current may be simply carried on the existing Clear windings. Furthermore, although range improvement is obtained at the expense of a new current source, dc currents are very simply regulated compared with pulse currents. Finally, the magnitude of the dc counter-bias current may be used as a fine control to aid in achieving the optimum operating point.

IV. MULTIPLE WINDINGS IN INPUT AND OUTPUT APERTURES

The circuits discussed thus far have all employed only a single winding in the input and output apertures. By relaxing this restriction, significant advantages can be obtained.

Drive on Inner Output Leg

With low-turn windings in the coupling loops, the Advance currents can become relatively large. By using the drive scheme indicated in Fig. 12, then for the same number of coupling loop turns, the advantages of com-

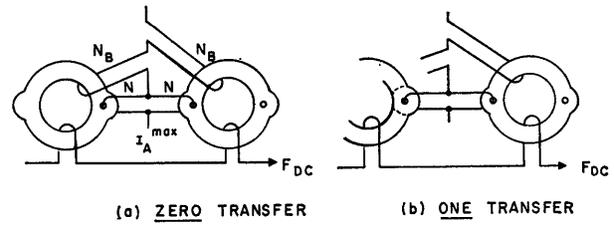


Fig. 11—Circuit using dc counter bias.

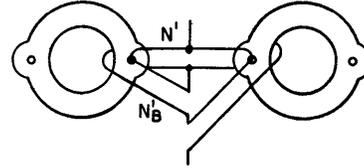


Fig. 12—Circuit using inner-leg drive.

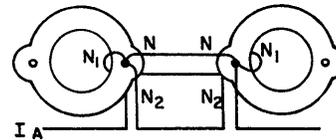


Fig. 13—Circuit using floating coupling loop.

patible bias can be obtained but with significantly lower Advance currents. In this circuit the Advance current is made to link the inner leg about the output aperture of the transmitter, as well as the coupling loop itself.

The equations for this case are identical to the equations for compatible bias if in the latter equations N is replaced by $(N' + 2N_B')$ and N_B by N_B' . Thus, for example, the case $N' = 3$, $N_B' = 1$ with the present scheme is identical to $N = 5$, $N_B = 1$ in the earlier scheme. Thus, $N' = 2$, $N_B' = 1$ yields maximum range corresponding to the case $N = 4$, $N_B = 1$, which was shown to be optimum for the earlier scheme.

Although, for a given number of coupling loop turns, there are more total turns in the small aperture, the additional drive turns may be of relatively small wire, since the main concern is only to make the coupling loop turns as large as possible to reduce coupling loop resistive losses.

Floating Coupling Loop

The circuits discussed thus far have the following two disadvantages: 1) with the coupling loop directly driven, care has to be exercised in physically connecting the two branch windings together so that proper ratios of the (parasitic) resistance and inductance are maintained and 2) since the same advance current flows through the coupling loops and bias windings, there are restrictions on the combinations of turns that may be used.

By allowing "floating" coupling loops, both of the above disadvantages are overcome. In the circuit of Fig. 13, the turns N , N_1 , and N_2 are completely inde-

pendent. The case $N_1=N_2$ is equivalent to the maximum range case $N'=2$, $N_B'=1$ in the inner leg drive case, which in turn is equivalent to the maximum range case $N=4$, $N_B=1$ for compatible bias. The equations for this case are identical to the equations for compatible bias if N is replaced by $2(N_1+N_2)$ and N_B is replaced by N_1 .

Note that with $N_1=N_2$ and with I_A^{\max} applied, the circuit of Fig. 13 exactly matches the conditions of Fig. 6(d), except that in this case the stress on the outer leg of the transmitter (and receiver) is applied directly from a drive winding instead of from coupling loop current. Current flows in the coupling loop only during *one* transfer.

V. COUPLING LOOP FLUX RELATIONS

The principal quantities of concern in these circuits are flux and current. Data are stored in particular flux patterns, and the currents are provided to move these patterns about in appropriate manner. In the earlier sections of this paper, it was implicitly assumed that the necessary flux relations were taken care of separately and that a reasonable estimate of range could be obtained by supposing that no flux switches during *zero* transfer and that complete switching occurs during *one* transfer.

In order for stable two-level operation to exist, the input (or received) flux ϕ_R at each transfer and the input flux ϕ_R' at the previous transfer must be related as indicated in Fig. 14, where the gain $G=\phi_R/\phi_R'$ is >1 in the interval $\phi_I < \phi_R' < \phi_U$ and is <1 in the interval $\phi_L < \phi_R' < \phi_I$. Thus, the transfer operation will tend to increase a "low" *one* level of flux toward ϕ_U , and to decrease a "high" *zero* level of flux toward ϕ_L . This is equivalent to saying that with such a gain relation, the operation stably protects against "zero build-up" and "one build-down."

With turns ratio greater than unity, *i.e.*, $N_T > N_R$, it is a straightforward matter to arrange a coupling loop to obtain the necessary relation between ϕ_R' and ϕ_R . Suppose that there are no losses in the coupling loop of Fig. 15(a). Then by integrating the relation $N_T \dot{\phi}_T = N_R \dot{\phi}_R$, which must hold at every instant of the switching period, we find the relation $N_T \phi_T = N_R \phi_R$, where ϕ_T and ϕ_R are the net flux changes in the transmitter and receiver. Then, provided $\phi_T = \phi_R'$,

$$\frac{\phi_R}{\phi_R'} = \frac{N_T}{N_R} > 1.$$

This relation between ϕ_R and ϕ_T is illustrated in Fig. 15(b). Notice that the linear relation holds only until ϕ_R saturates. In order finally to obtain the necessary relation of Fig. 14(b), consider the coupling loop of Fig. 15(c) in which a "clipper" core is added. This core is arranged to have a total flux linkage capacity of ϕ_C , which is a relatively small fraction of the saturation flux of the MAD's. However, the switching threshold is much

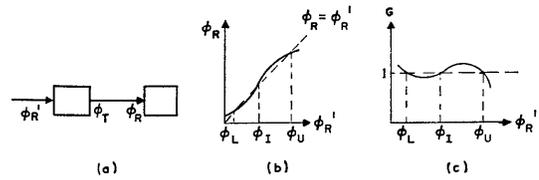


Fig. 14—Gain relations for bi-stable operation.

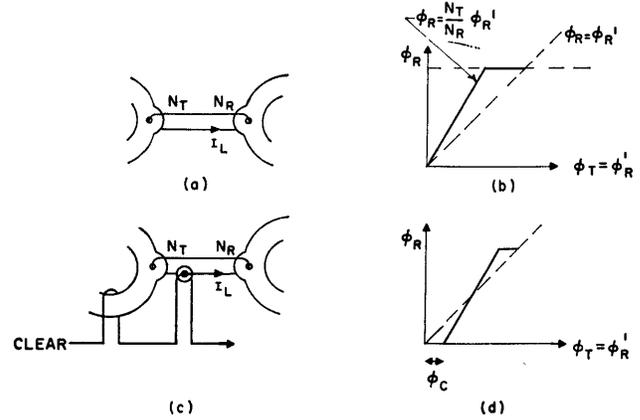


Fig. 15—Coupling loop arrangement for achieving proper gain relations.

lower than that of the receiver, and therefore, its full capacity of flux will be substantially switched before the receiver begins switching at all. In this way, a constant amount of flux is subtracted from the quantity $(N_T/N_R)\phi_T$ when $>\phi_C$; if $(N_T/N_R)\phi_T < \phi_C$, then $\phi_R = 0$. The resulting relation between ϕ_R and $\phi_T = \phi_R'$ is shown in Fig. 15(d). Notice that this curve has the proper form for bistable operation (with $\phi_L = 0$ in this case). If the flux clipper is cleared at the same time as the transmitter, none of the basic clock cycle operations of Fig. 4 is altered, and very good transfer loop operation is achieved.

The use of a flux clipper along with the condition $N_T/N_R > 1$ makes the gain properties of the transfer loop very explicit. However, a flux clipper is not actually required. In fact, it is further demonstrated in the following sections that the relation $N_T/N_R > 1$ is also not required and that successful operation can be achieved with $N_T/N_R \gtrsim 1$. The case $N_T = N_R$ is interesting because the transfer loop is symmetrical, and bi-directional shifting is possible by merely reversing the sequence of Clear pulses or sequence of Advance pulses in Fig. 4. The case $N_T = N_R = 1$ is particularly interesting because of the simple assembly schemes that are made possible. The case $N_T > N_R$ is useful where bi-directional properties are not necessary, and extra flux gain is required. No particular advantages can be seen for the case $N_T < N_R$.

An explicit clipper core is not required because a partial clipping action is performed by the parasitic resistance and inductance of the coupling loop. With the Advance current set in the middle of its range, neither the transmitter nor receiver is brought up to its thresh-

old level during *zero* transmission. Thus, during *one* transmission, a considerable amount of current must be steered to the receiver before any flux can be switched there at all. As the current begins to rise, flux linkage is lost (or stored) in the loop inductance L in the form of LI_L volt-sec, where I_L is the loop current. Any of this stored flux linkage remaining in the loop after switching stops in the receiver (whether due to the advance pulse ending or to the receiver current dropping below the effective threshold) is dissipated in the loop resistance and lost. Furthermore, there is an additional resistive loss of flux linkage in terms of $\int I_L R_L dt$ (where I_L and R_L are loop current and resistance) during the time in which the receiver is switching. Most significantly, however, for sufficiently low levels of flux, ϕ_T will be absorbed entirely into $\int I_L R_L dt + LI_L$ before I_L brings the receiver up to threshold, and therefore ϕ_T will be 100 per cent lost. Hence the plot of ϕ_R vs ϕ_T will start from zero at some value of $\phi_T > 0$, just as in Fig. 15(d).

Unlike Fig. 15(d), the curve will now not be linear, but as long as the turns ratio is just great enough to bring the curve above the $\phi_R = \phi_R'$ line at some higher value of ϕ_R' , bistable operation will be achieved. The turns ratio required is not high, 6/5 being a typical example. In fact, a high turns ratio is quite undesirable, since the excess drive available for switching is reduced. For example, if $N_T/N_R = 2$, then for a simple coupling loop with no bias, $I_{T0} = F_2/N_T$ is an upper bound on the current available for steering to the receiver during *one* transfer. The receiver mmf provided by this current would be $(N_R/N_T)F_2 = (1/2)F_2$, whereas the corresponding figure for unity turns ratio would be F_2 .

Before discussing the properties of unity-turns-ratio operation ($N_T = N_R$), let us consider some basic switching properties of magnetic cores that will be useful in later discussions.

VI. SOME BASIC SWITCHING PROPERTIES OF CONVENTIONAL CORES

Consider the flux-current relations for the conventional toroid of Fig. 16(a). Assume that the B - H curve for the material is ideally square, Fig. 16(b).

With very long setting pulses I_s , the ϕ_s - F_s curve is as indicated in Fig. 16(c), where ϕ_s is the amount of switched flux in response to a setting mmf $F_s = N_s I_s$ applied to a well-cleared core. The ratio F_b to F_a is the same as the ratio r_o to r_i (outer to inner radius). This curve may be automatically plotted by setting up a continuous pattern of alternate Clear and Set currents, in which the Set current is made to vary in amplitude from cycle to cycle. By deflection of an oscilloscope beam in the x direction in response to current I_s and in the y direction in response to switched flux ($= \int edt$), the ϕ_s - F_s curve is automatically traced. In all of the ϕ - F curves to be considered here, ϕ represents remanent flux (*i.e.*, does not include the elastic or reversible component of flux). To plot remanent flux curves, it is only necessary to energize the oscilloscope beam just after

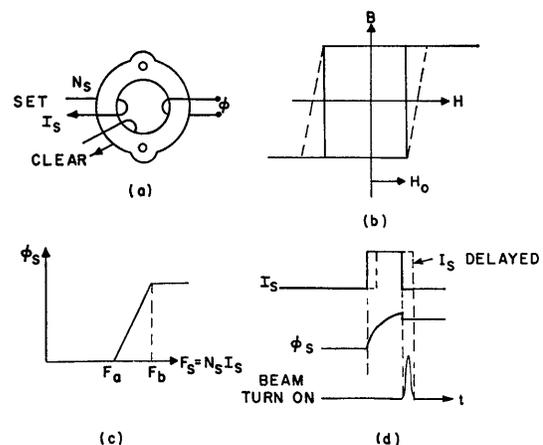


Fig. 16—Core switching experiment.

the setting current is over, by which time the elastic flux has been removed. To maintain the x deflection unchanged until such a time as the beam is energized, it is merely necessary to connect the Advance current pulse to the x deflection plates via a delay line, Fig. 16(d).

An interesting property to consider in relation to the ϕ_s - F_s curve is its dependence on the pulse width of the setting current. Consider a switching model in which it is assumed that the switching rate dB/dt in any portion of the material is proportional to the instantaneous excess drive $(H - H_0)$, where H_0 represents the threshold field. Although idealized, this model does result in the usual inverse relationship between switching time and excess field for the case of a thin ring of material. By its use, calculated ϕ_s - F_s curves for different pulse widths are shown in Fig. 17(a). For very long pulse widths (*i.e.*, $T \rightarrow \infty$), the curve reduces to that shown in Fig. 16(c). However, for a given drive F , as T decreases, a smaller and smaller amount of flux is switched; hence the ϕ_s - F_s curves are monotonically lowered as T decreases. With this model, for pulse widths greater than some critical value T_c , each curve has a linear region marked on the lower end by the mmf required to just saturate the inner radius in time T , and on the upper end by the mmf required to just start switching the material at the outer radius. The nonlinear connecting regions are approximately parabolic for relatively thin-walled cores having a ratio of outer to inner radii of about 1.3 or less.²

In Fig. 17(b) is shown an actual family of ϕ_s - F_s curves. For later comparison, these and all later curves, unless otherwise stated, are taken on an experimentally molded MAD element (having the nominal dimensions indicated in Fig. 19) treated as a conventional core. By

² It may also be noted that these curves have the identical form as for the case in which a very long switching pulse is used on a core having the same dimensions as here but for which the slope of the "rising" portion of the B - H curve of the material is a variable. With truly vertical sides, the curve $T \rightarrow \infty$ applies. With the sides less steep, as shown by dotted lines in Fig. 16(b), the family of ϕ_s - F_s curves has the identical form of Fig. 17(a).

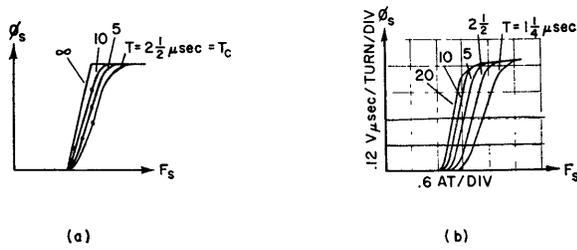


Fig. 17—Calculated and measured ϕ_s - F_s curves.

shaping about the apertures as indicated in Fig. 19, the results are substantially identical with results obtained on actual toroids of the same material. Notice that, compared with the curves of Fig. 17(a), these curves do not radiate in so pronounced a fashion from the value F_a , but rather are mainly translated horizontally to higher values of mmf as the pulse length decreases. This property is very important in MAD elements for reasons that will become clearer below.

If one looks at the corresponding switching voltage curves for this element, it becomes apparent what is causing this translation of the ϕ_s - F_s curves. This family of voltage curves vs time is indicated in Fig. 18, where the parameter is the magnitude of I_s (where I_s is a very long pulse). The curves of Fig. 18(a) are calculated using the model $dB/dt \propto (H - H_0)$. Notice that this simple model is very inadequate for predicting the front end of the voltage curves. This fact is understandable, since this model relates more to the rate of movement of existing domain walls. However, if we start with a well-cleared core in which there is a minimum of reverse domains (walls), then after the pulse I_s is turned on, it takes a time for domain walls to be established.³ This is reflected in the initial slope of the voltage curves. In any case, notice the difference in behavior of the peaks of the voltage curves as a function of I_s . In the family of actual curves, there is a large "peaking delay" at low levels of switching. With materials that exhibit significant peaking delay properties, the voltage is almost exactly zero before the time of peaking. It is straightforward to convert the voltage curves of Fig. 18(b) into the ϕ_s - F_s curves of Fig. 17(b) and see the reason for the increase in threshold of the narrow-pulse curves with peaking delay.

It may be noted that peaking delay is a property of some materials and not others. For example, there are materials for which the peaks in the switching curves of Fig. 18(b) lie almost directly over each other. For these materials, the ϕ_s - F_s curves are more like those of Fig. 17(a).

Given a core of appropriate material, it is further necessary, in order for the core to exhibit peaking delay, that the setting current I_s be applied to a well-cleared core. Generally speaking, Clear strengths of at

³ N. Menyuk and J. B. Goodenough, "Magnetic materials for digital computers, I. A theory of flux reversal in polycrystalline ferromagnetics," *J. Appl. Phys.*, vol. 6, pp. 8-18; January, 1955.

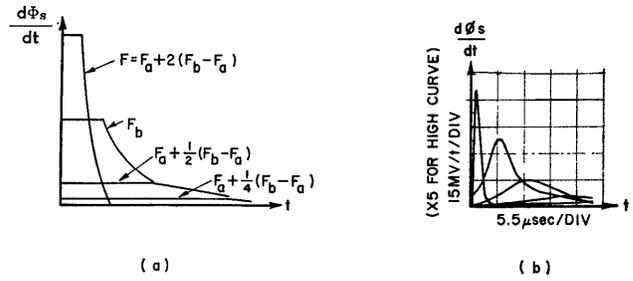


Fig. 18—Calculated and measured $d\phi_s/dt$ -time curves.

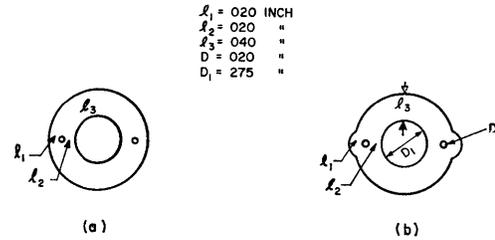


Fig. 19—Shaping of MAD elements.

least two to three times threshold are required for good peaking delay.

If a MAD is made by cutting apertures in the wall of a conventional toroid, Fig. 19(a), then regardless of the Clear magnitude, it is impossible to get all material on a major loop since the cross-sectional area $1_1 + 1_2$ is less than 1_3 for a core of unit height. Thus, even for material that is potentially capable of exhibiting peaking delay, this type of construction nullifies the effect. However, by appropriate shaping of the element, e.g., so that $1_1 + 1_2 = 1_3$, the element treated as a simple toroid will exhibit significant peaking delay.

Another very important switching property related to peaking delay is shown by the families of ϕ_s - F_s curves taken for the condition in which the core is preset before I_s is applied. The families of ϕ_s - F_s curves shown in Fig. 20, contain the magnitude of preset flux ϕ_p as the parameter. The difference between the various families of curves are that they are taken for different combinations of long and short duration preset and set pulses.

For zero preset ($\phi_p = 0$), the total switched flux ϕ_s is $\phi_s^{\max} = \phi^M$, where ϕ^M represents the total flux capacity of the core from saturation in one direction to saturation in the other. When a core has been preset, the current I_s has correspondingly less flux available to switch. Thus in Fig. 20(b), as the amount of preset flux ϕ_p increases, the maximum switchable flux $\phi_s^{\max} = \phi^M - \phi_p$. For tracing these curves automatically, a repeating cycle of Clear, Preset, and Set currents is applied to the core of Fig. 20(a).

For each curve, the Preset current is adjusted to the appropriate level and the Set current is made to vary from cycle to cycle. The oscilloscope plotting is exactly as indicated in connection with Fig. 16(d), where the beam is turned on just after the Set current is over.

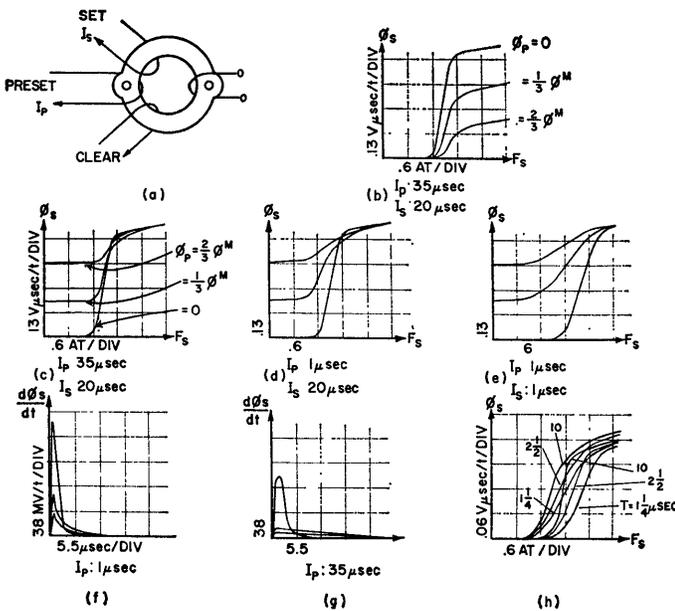


Fig. 20— ϕ_s - F_s and $d\phi_s/dt$ -time curves for various pulse lengths of I_p and I_s .

For later convenience in dealing with families of curves for MAD elements, the curves of Fig. 20(b) are redrawn in Fig. 20(c) with the final rather than initial values of flux superimposed. Actually, this is equivalent to raising the zero flux level for each curve by the magnitude ϕ_p .

When a well-cleared core is preset to a certain level of flux by a long current pulse, one can visualize the flux condition in the core to be represented by a circumferential domain wall outside of which the flux is in a clockwise (Clear) direction and inside in a counterclockwise direction. Except in the transition region, substantially all material is well saturated. Since I_s stresses the core in the same direction as I_p , it is reasonable for gentle preset (long pulse) that, regardless of preset level, the current I_s should continue the switching where I_p left off, and with essentially the same threshold, as demonstrated in Fig. 20(c). Let us now consider the effect of shortening the pulse durations. In order for a certain magnitude of flux, ϕ_p , to be preset as the preset pulse is decreased in length, the magnitude of the preset current must be increased. For significant decrease in duration, the magnitude of current must be likewise considerably increased. For a reasonably thin-walled core, this increased magnitude of current is capable of switching flux simultaneously throughout the entire core, so that the current pulse must be shut off when the proper level of preset flux is reached. In this case, it is certain that reverse domains are distributed throughout the body of the core in some random fashion. Thus it is hardly surprising that after such a preset pulse, the set current, which tends to continue the switching in the same direction, finds a much lower and less abrupt threshold. This effect is clearly seen when the curves of Fig. 20(c) and 20(d) are compared.

Let us next consider the effect of short set and preset pulses, Fig. 20(e). For $\phi_p = 0$, the ϕ_s - I_s curve is just the appropriate curve of the family of Fig. 17(b), for the given pulse duration. If we compare the curves of Fig. 20(d) and Fig. 20(e), we notice that in the latter case, the threshold moves a considerable distance to the right for zero preset, but not quite so far for nonzero preset. This is reasonable in terms of the previous discussion of the effects of a good Clear state on peaking delay. Good peaking delay occurs only when all of the material is in a well saturated condition. However, due to presetting with a short pulse, a random distribution of reverse domains is left throughout the core, resulting in very poor peaking delay after preset and hence very little increase in threshold for a short-pulse set. This point is demonstrated by the voltage-time curves (similar to those of Fig. 18) taken after a preset condition of $\phi_p = 1/2 \phi^M$ using a short preset pulse, Fig. 20(f), and a long preset pulse, Fig. 20(g). Notice the significant difference in switching times for these two families. The effect of preset is also demonstrated in the ϕ_s - F_s curves, Fig. 20(h), taken for the same preset level $1/2 \phi^M$. The group of curves radiating from the lower threshold value of F_s is for the short preset pulse; the other group is for the long preset pulse. The lowering of threshold for short preset pulses is clearly seen. Within each group, the parameter is the duration of Set pulse.

VII. SWITCHING PROPERTIES OF MAD'S

Ideal Family of Output Curves

In Fig. 1, output ϕ_T - F_T curves were shown for the two cases of a Set and Clear MAD. Actually, there exists a whole family of such curves with the amount of preset (or input) flux as the parameter (Fig. 21). In Fig. 21(a), the input current is shown linking leg l_1 about the input aperture, and the output current is shown linking leg l_4 about the output aperture. Assume all legs are of equal dimension. Let ϕ^M represent the total flux capacity in any leg from saturation in one direction to saturation in the other direction. With leg l_4 saturated in the Clear (clockwise) direction, application of I_T of sufficient magnitude will switch an amount of flux ϕ^M in leg l_4 independent of the amount of preset flux. A portion of it equal to $\phi_p (= \phi_{in})$ will switch locally about the output aperture and the remainder will switch around the main aperture. If all material is operating on an ideal rectangular hysteresis loop, the family of curves will have the form indicated in Fig. 21(b).

Actual Family of Output Curves

Actual families of output curves, for the same MAD used for the above tests of core switching properties, are shown in Fig. 22. These curves are automatically plotted by the method previously described and are taken for the same combinations of long and short current pulses indicated in Fig. 20. Notice that the effects are substantially the same as observed in the case of presetting

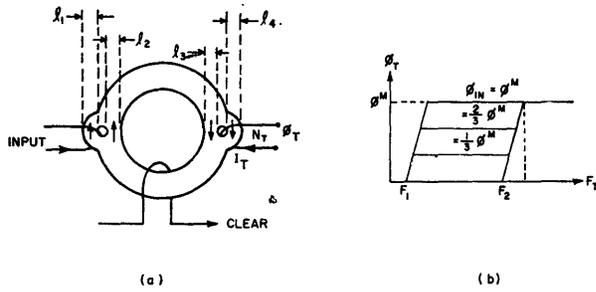


Fig. 21—Idealized family of output curves ϕ_T - F_T for a MAD.

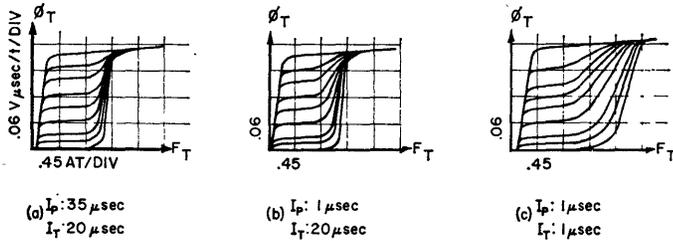


Fig. 22—Actual families of ϕ_T - F_T curves for various input and output current pulse durations.

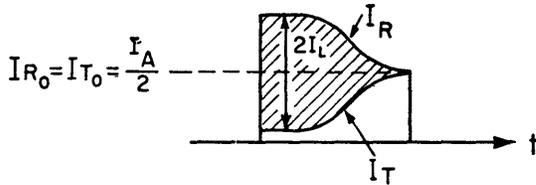


Fig. 23—Transmitter and receiver currents during one transfer.

a toroid. Actually, the short-short current pulse combination is the one of interest, because this more nearly approximates actual operation. In Sections III and IV, it was indicated that in biased circuits, the transmitter and receiver are switched with two or more thresholds of current. This operation corresponds to high-drive, short-pulse conditions of measurement. However, it is shown below that for one transfer, the transmitter current I_T and receiver current I_R are not constant in time. To this extent, the above experimental families of curves, which are plotted for rectangular input and output current pulses, do not apply. Nevertheless, they are extremely indicative of the nature of the operation.

The main significance of these curves is the lowering of the main aperture threshold for partial set levels relative to the Clear State threshold. It is demonstrated below that this property provides a mechanism for obtaining proper gain relations for unity-turns-ratio operation.

Transmitter and Receiver Currents

In the (ideal) zero transfer case, assuming unity turns ratio, the Advance current I_A divides into equal branch currents $I_{T_0} = I_{R_0} = (1/2)I_A$, where the sub "0" stands for the zero case.

During a one transfer, the Advance current divides

into unequal branch currents I_T, I_R such that at all instants of time $\dot{\phi}_T = \dot{\phi}_R$. Because of the lower threshold (neglecting the voltage drop in the loop resistance and inductance) in the transmitter, $I_R > I_T$, but always $I_R + I_T = I_A$. This situation can be characterized by a loop current I_L superimposed on the zero transfer currents so that $I_R = I_{R_0} + I_L$ and $I_T = I_{T_0} - I_L$

$$\left(\text{where } I_{R_0} = I_{T_0} = \frac{I_A}{2} \right), \text{ or } I_L = \frac{I_R - I_T}{2} .$$

During one transfer, the transmitter may be characterized as a relatively small, thick-walled core [see e.g., Figs. 9(b) and 10(b)] compared with the receiver. Thus, for given magnitudes of I_T and I_R [assuming the switching model $dB/dt \propto (H - H_0)$], the ratio of switching rates at the outer and inner walls is much higher in the transmitter than in the receiver. For constant I_T and I_R , the rates of flux change $\dot{\phi}_T$ and $\dot{\phi}_R$ will be constant in time only as long as no material saturates. Clearly, the inner wall of the transmitter will saturate first. When this happens, $\dot{\phi}_T$ tends to decrease, since less material participates in the switching. However, this tendency is counteracted by an increase of I_T to bolster up $\dot{\phi}_T$ at the expense of $\dot{\phi}_R$ (i.e., I_R decreases), so that at all times the relation $\dot{\phi}_T = \dot{\phi}_R$ is maintained. Thus, the resulting I_T and I_R curves have the general form indicated in Fig. 23.

The main significance of the current relations indicated here is that during one transfer, the transmitter current is initially low but rises toward $I_A/2$ while transfer continues.

The ϕ^ Contribution to Flux Gain*

Flux gain G from one stage to the next may be defined as $\phi_R/\phi_{R'}$ where $\phi_{R'}$ is the flux received in the transmitter during the previous Advance pulse. If during the present Advance pulse, all available flux about the output aperture is switched but none is switched about the main aperture of the transmitter, then $\phi_T = \phi_{R'}$. Now $\phi_E = \phi_T - \phi_{\text{loss}}$, for a single-turn coupling loop (a special case of unity-turns-ratio operation) where ϕ_{loss} , the flux loss in the loop resistance R_L , is $\int I_L R_L dt$ volt-sec. Hence, if $\phi_T = \phi_{R'}$,

$$G = 1 - \frac{\phi_{\text{loss}}}{\phi_{R'}} .$$

Note that $G > 1$ is impossible here because of the subtractive loss term. This equation is characteristic of the operation of a conventional core-diode-type shift register, which requires $N_T > N_R$ in order to obtain $G > 1$.

In a MAD arrangement, ϕ_T can be more than $\phi_{R'}$ because of the possibility of flux switching not only locally about the output aperture, but also around the main aperture in a direction to increase the setting of the element. For illustration, assume that the Advance current is set in the center of its range, or $I_A = I_A^{av}$, so that $NI_A/2$ is below threshold F_2 as indicated in Fig. 24. Also

assume a level of input flux ϕ_R' as indicated in the figure. Then as I_T increases from its initial low level (see previous section) towards its steady-state value $I_A/2$, it soon enters a region in which flux may be switched about the main aperture. This flux is defined as ϕ^* . Thus $\phi_T = \phi_R' + \phi^*$ and the gain equation is modified to read

$$G = 1 + \frac{\phi^*}{\phi_R'} - \frac{\phi_{\text{loss}}}{\phi_R'}$$

It is clear from Fig. 24 that ϕ^* is a function of ϕ_R' having the form indicated in Fig. 25. ϕ^* is very small at low levels of received flux because the maximum value of NI_T is below threshold for switching around the main aperture. ϕ^* is also very small for large ϕ_R' because of saturation. It is important to note that because of the finite slope of the ϕ_T - F_T curve below threshold (even for the Clear state curve) that ϕ^* is everywhere >0 , and hence that $\phi^*/\phi_R' \rightarrow \infty$ as $\phi_R' \rightarrow 0$. Thus the contribution of the term ϕ^*/ϕ_R' to the gain equation has the form indicated in Fig. 25(b). The steeply rising portion of this curve for low values of ϕ_R' contributes to the lower unity-gain crossing of the Gain- ϕ_R' curve at $\phi_R' > 0$, resulting in $\phi_L > 0$, as would be expected. Thus, if over some interval, the ϕ_{loss} term is smaller in magnitude than the ϕ^* term (for $\phi_R' > \phi_L$), then a gain curve of the form indicated in Fig. 14 can be obtained, and the interval in question is just the interval $\phi_I < \phi_R' < \phi_U$ where $G > 1$.

Resistive Flux Loss

The resistive flux loss, $\int I_L R_L dt$, is just proportional to the cross-hatched area in Fig. 23. An accurate analysis of ϕ_{loss} is extremely difficult, since no accurate switching model is available for predicting the current shape $I_L(t)$ as a function of flux level, ϕ_R' .

In Sections II to IV, for purposes of comparing switching speeds for various circuit arrangements, it was assumed that I_A divides into I_T and I_R in such a way as to apply equal multiples of threshold mmf to transmitter and receiver. This model does result in the correct ordering of the different circuits in terms of speed, but it is very inadequate for predicting $\phi_{\text{loss}}/\phi_R'$ as a function of ϕ_R' for a given circuit. This fact should be quite obvious just from the complexity of measured switching characteristics as represented in Figs. 20 and 22.

True, one can definitely say that ϕ_{loss} increases with ϕ_R' , but not even this much can be said about the ratio $\phi_{\text{loss}}/\phi_R'$, except for very low values of ϕ_R' . In this latter case, as was indicated earlier in the discussion of the partial clipping action of loop inductance and resistance, $\phi_{\text{loss}}/\phi_R'$ will be essentially unity for sufficiently low flux levels.

In effect, then, all that can be said at present is that the loss term will decrease from near unity for very low ϕ_R' to a value Δ at some low $\phi_R' (= \phi_1)$ and remain below

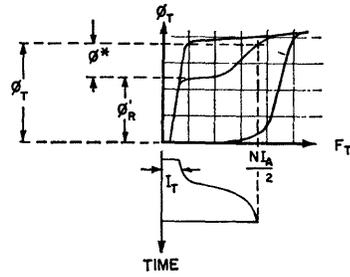


Fig. 24—Demonstration of ϕ^* flux gain.

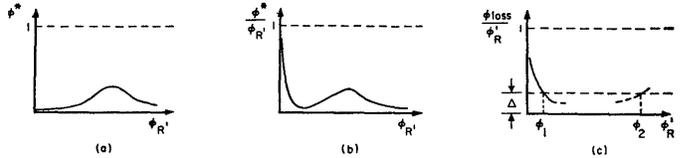


Fig. 25—Plot $\phi^*-\phi_R'$, $\phi^*/\phi_R'-\phi_R'$ and $\phi_{\text{loss}}/\phi_R'-\phi_R'$.

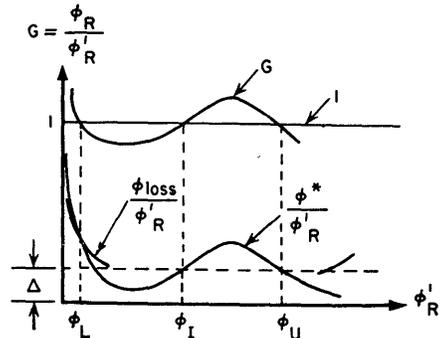


Fig. 26—Addition of gain equation components to form the gain curve.

Δ at least up to some high $\phi_R' (= \phi_2)$, as indicated in Fig. 25(c).

In Fig. 26, it is shown how the three terms of the gain equation

$$G = 1 + \frac{\phi^*}{\phi_R'} - \frac{\phi_{\text{loss}}}{\phi_R'}$$

may add to give the required form indicated in Fig. 14. The loss term is added as though equal to Δ in the indeterminate interval $\phi_1 < \phi_R' < \phi_2$.

Whatever the actual variation of the loss term in this interval, then, the qualitative nature of the gain curve will not be changed, rather only the locations of the unity gain points ϕ_I and ϕ_U .

Flux Boost

In order to get reasonable Advance current range, the short-short ϕ_T - F_T curves [Fig. 22(c)] should have the Clear state curve moved as far to the right as possible, and the higher level flux input curves moved as far to the left as possible. Experimental results have indicated that these conditions are generally met best by

materials exhibiting good peaking delay properties [Fig. 18(b)].

The range analysis of Sections II to IV does not apply well to the unity-turns ratio case since the advance current is more limited by the requirement of getting ϕ^* to make up for flux losses than by the assumptions of Section II.

In any case, the situation can be significantly improved by the circuit described here.

Between the time that the flux ϕ_R is received, and flux ϕ_T is transmitted to the next stage, the previous element is cleared. By having this pulse do double duty, as indicated in Fig. 27, the flux boost may be obtained (at Clear time) before transfer out of the element occurs. With N turns on the receiver, the Clear current is adjusted so that the receiver is brought up to the vicinity of threshold F_2 . Thus, if ϕ_R is low (as for a zero), then the boost pulse has essentially no effect. If ϕ_R is high (as for a one) but less than ϕ^M because of ϕ_{loss} during transmission, then the boost pulse will increase the set level of the element.

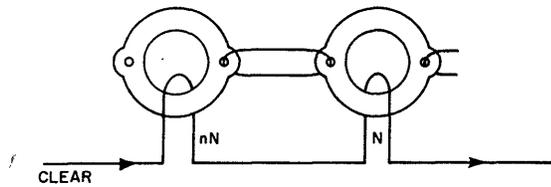


Fig. 27—Circuit for achieving flux boost.

The family of curves of Fig. 24 is taken with a winding about the output aperture. However, the reduced thresholds for partially set levels are characteristic of the state of the entire element (as previously described) and would also be observed in ϕ - F curves taken on the flux boost winding. Thus the effect of the flux boost current can be predicted from the ϕ_T - F_T curves.

The Clear winding has nN turns, which results in the transmitter being cleared with n thresholds of mmf. For good clearing, n should be greater than 2. By consulting Fig. 4, we see that with this new arrangement, two processes are going on simultaneously in the receiver. Clearing the transmitter causes a negative set current to switch flux locally about the input aperture of the receiver, and the mmf applied by the flux boost winding causes additional set flux to be switched about the main aperture.

In all previous circuits, the Clear pulse had unlimited range as long as it was above some minimum. Such is not the case in the present circuit, since the Clear current works against a threshold. However, although the Clear current range is decreased, the Advance range is significantly increased.

The flux boost method of making up flux losses has three main advantages compared to the ϕ^* method.

First, there are no coupling-loop losses associated with flux boost, whereas part of the available ϕ^* is always lost during transfer. In fact, with flux boost, it is possible for the receiver to be set fully before flux is transferred out of it; this condition is impossible to obtain for unity-turns-ratio operation without flux boost, because of losses. Second, the boost current has a fixed magnitude and duration independent of flux level, whereas the current I_T switching ϕ^* is greater (at least in integrated value) for low flux levels than for high levels. Third, the boost current can be adjusted in width and amplitude independently of the advance current, while the mmf that switches ϕ^* is tied to the advance current.

It is pertinent to note that with flux boost taking care of the flux-gain requirement, Advance range is more closely predicted by the relations of Sections III and IV. Furthermore, flux boost may be used advantageously in connection with all of the circuits derived in that section. This is so in particular for unity-turns-ratio operation, but advantage can be obtained even for circuits in which $N_T/N_R > 1$.

VIII. RESULTS AND CONCLUSION

The range and speed relations derived in Sections III and IV, although based on a very simple model, do properly predict comparative results for the various circuit arrangements. As pointed out in the discussion on flux boost, these relations do not apply well for unity-turns-ratio circuits if flux boost is not used. In this case, the Advance currents must be adjusted more to satisfy the flux gain requirements than the simple switching model used to derive these range and speed relations. Listed below is an example of the types of comparative results obtained with a coupling loop using $N_T = 6$, $N_R = 5$, for the circuits indicated.

Circuit of Fig.	Bias turns $N_{BT} = N_{BR}$	Range
5	—	15 per cent
10	1	30 per cent
11	2	50 per cent, using 50 per cent dc counter bias

Flux boost helps even where $N_T > N_R$. For example, when flux boost is used, the 15 per cent range obtained using no bias will increase to about 30 per cent. In flux boost circuits in general, the Clear and Advance currents may be independently adjusted to give either one a high range at the expense of the other. The range value given here, however, implies that the Clear and Advance currents are adjusted so that they both have the same range, namely 30 per cent.

Unity-turns-ratio coupling loops, *i.e.*, with $N_T = N_R$, have operated with the following typical results. With the bias circuit of Fig. 10, $N = 4$, $N_B = 1$, and flux boost, Clear and Advance ranges of greater than 40 per cent each are achieved. These results are obtained with 1- μ sec

Clear and Advance pulses driving a register of experimentally molded elements having the dimensions indicated in Fig. 19 and made with material having a long-pulse threshold of 0.7 oersted. Single-turn coupling-loop circuits with the same effective bias operate equally well.

The detailed analysis of these circuits is difficult not only because of the usual difficulties of dealing with the dynamic properties of highly nonlinear elements, but also because of the relatively complex geometries involved. It is clear that a good deal of the design of these circuits is necessarily based on intuition and empirical results. The circuits described here can be made to operate quite well, however, and the lack of analytical

tools is felt more in trying to decide how or when a particular arrangement is optimum. It is hoped that future efforts will result in the development of satisfactory switching models that will make the circuit design procedure routine.

The techniques presented here provide the potential for developing extremely reliable digital circuitry at least for the intermediate computer speed ranges of 0.1 mc to 1 mc clock (or bit) rates.

IX. ACKNOWLEDGMENT

The authors wish to acknowledge the very helpful suggestions and contributions of their colleague, Dr. Douglas Engelbart, to the material presented here.

A Twistor Matrix Memory for Semipermanent Information*

DUNCAN H. LOONEY†

INTRODUCTION

A NEW magnetic matrix memory has been developed for the storage of semipermanent digital information. The memory is designed for computers which require random access to stored information that is changed very infrequently. The information is stored in a pattern of permanent magnets arranged on a plastic board. The presence or absence of a permanent magnet is sensed nondestructively by a wire wrapped with a magnetic tape placed close to the permanent magnet. A stored word is read by a linear selection system using a biased core access switch.¹

The memory is fabricated in modules. A typical module is shown in Fig. 1. The photograph shows a 512-word memory consisting of 32×16 word locations. Each word location stores 26 bits of information. Any word location in the memory may be selected at random and the information read in a period of a few microseconds. The temperature range of operation of the memory is extremely wide.

The concept of storing information in an array of permanent magnets was advanced by the late S. Shackell. Mr. Shackell's work was interrupted by his untimely death and has not been previously reported in the

literature. With the development of the twistor,² John Janik, who was familiar with the Shackell scheme, suggested its use in such a system to reduce the size of the permanent magnets.

The operation of a store using the 512-word memory module is described in a companion paper.³ The store, which utilizes all solid-state circuitry, is compared to other systems using photographic or magnetic techniques which can be used for the storage of semipermanent information.

OPERATING PRINCIPLE

The information is stored in an array of small permanent magnets. The presence of the magnet is sensed by a wire wrapped with magnetic tape placed close to the magnets. A group of 26 wrapped wires are encapsulated in a plastic tape. The encapsulated wires are then enclosed in a set of copper solenoids as illustrated in Fig. 2. A particular solenoid corresponding to a stored word may be selected by activating one core of the biased core access switch. The bar magnets are arranged in a pattern on the surface of a thin plastic card. Each magnet is located at the intersection of a wrapped wire or twistor and a solenoid. The purpose of the permanent magnet is to inhibit locally the drive field of

* The work reported in this paper was done for the U. S. Dept. of Defense under Contract DA-30-069-ORD-1955.

† Bell Telephone Labs., Murray Hill, N. J.

¹ J. A. Rajchman, "A myriabit magnetic core matrix memory," *Proc. IRE*, vol. 41, pp. 1407-1421; October, 1953.

² A. H. Bobeck, "A new storage element suitable for large sized memory arrays—the twistor," *Bell Sys. Tech. J.*, vol. 36, pp. 1319-1340; November, 1957.

³ J. J. DeBuske, J. Janik, and B. H. Simons, "A card changeable nondestructive readout twistor store," this issue, pp. 41-46.

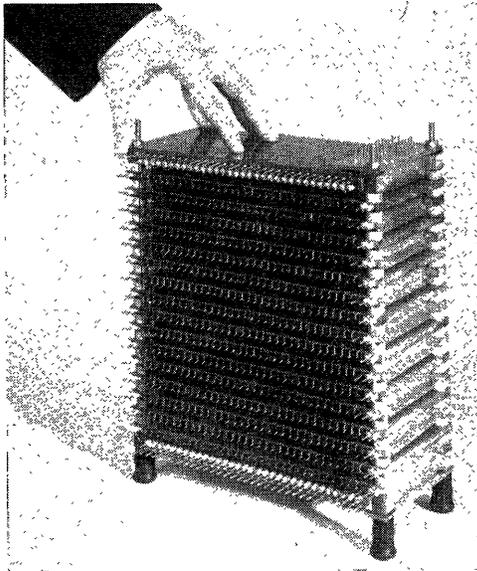


Fig. 1—A 512-word memory module. The unit is composed of 16 frames of 32 words. The cores of the access switch and the multi-turn windings are shown. The encapsulated twistor tape threads run through the module continuously.

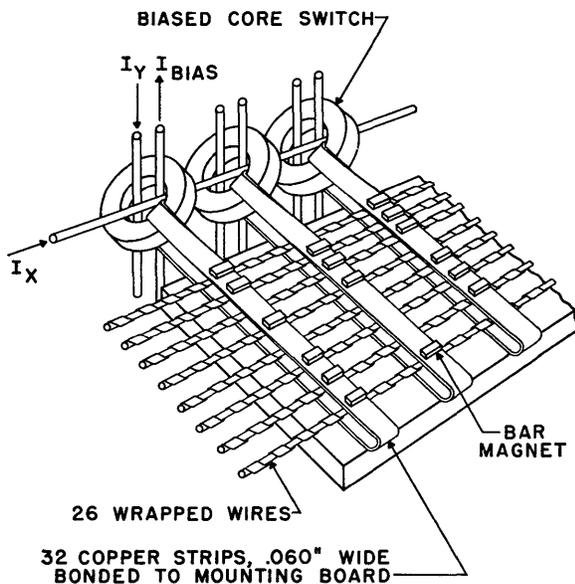


Fig. 2—A section of a memory frame. Three cores of the biased core switch and their word solenoids are shown. The absence of a bar magnet is sensed by the flux reversal of the wrapped wire when a drive current flows in the solenoid.

the solenoid. Thus, the permanent magnet prevents the switching of the wrapped wire located beneath the permanent magnet. The magnitude of the field in the solenoid can be high to achieve fast switching. The field of the bar magnet must, however, be sufficient to inhibit the switching of the wrapped wire. A plane of the module with a complete magnet board is shown in Fig. 3.

In the present design, the solenoids are 1/16 inch wide and spaced 3/16 inch apart. For a current drive of 1.8 amp in the solenoid a switching speed of about 1 μ sec

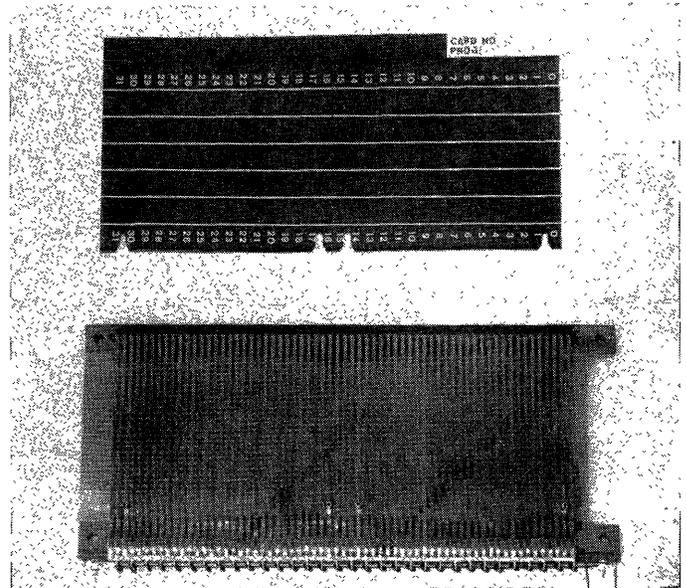
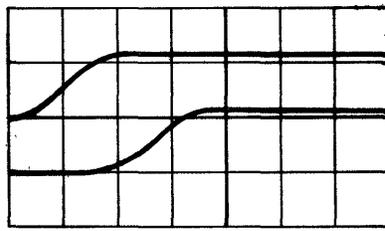


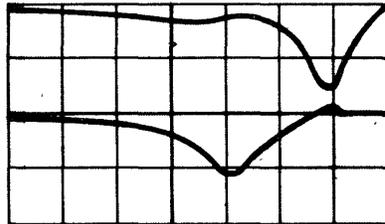
Fig. 3—A memory frame of 32 words and its magnet board. Every other copper strip is used as a solenoid and is attached to an access core. The guide pins of the frame and the corresponding holes of the magnet board are necessary to register the bar magnets at the intersections of the solenoids and the wrapped wires.

is obtained. Current pulses of 600 ma each are applied to four turn X and Y windings of the biased core switch. The bias on each core is 2.4 amp-turns. The X pulse is applied first since the X winding is parallel to twistor wires and results in a larger inductive signal. The sequence of operations is shown in Fig. 4. Time is measured from left to right. The two current pulses are shown in Fig. 4(a). When the applied pulses are removed, the bias current resets the core. The back voltages on the X winding through 32 cores and the Y winding through 16 cores are shown in Fig. 4(b). The core requires about 0.6 amp-turns to generate an output voltage which will drive 1.8 amp through the solenoid. The resultant output signals are shown in Fig. 4(c). Both one and zero output signals are shown by inserting and removing a magnet board. The one signals average 8 mv into 180 Ω while the zero signals are less than 2 mv. The signal-to-noise ratio is 5 to 1. Considering the time necessary to turn on the access switch, a 5- μ sec cycle time is easily achieved. Since there are no fundamental limitations on the magnitude of the drive or the amount of flux which must be reversed, shorter cycle periods are possible.

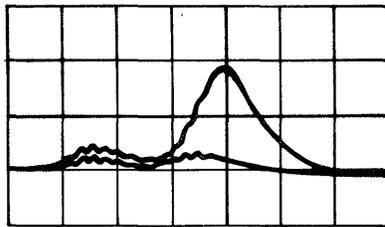
The general performance of the module is shown in Fig. 5. The two current pulses into the biased core switch are shown in Fig. 5(a). The one signals observed on one sensing wire from 32 word locations are shown superimposed in Fig. 5(b). The one signals observed on 26 sensing wires at one word location are shown in Fig. 5(c). The open circuit and matched load output signals are shown in Fig. 5(d) for the 512-word module. The resistive load is 22 Ω and is equal to the sensing wire resistance.



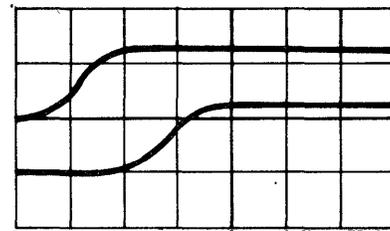
(a)



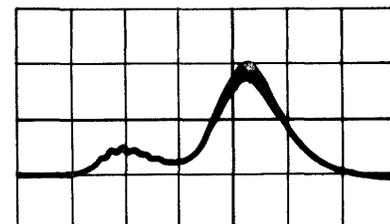
(b)



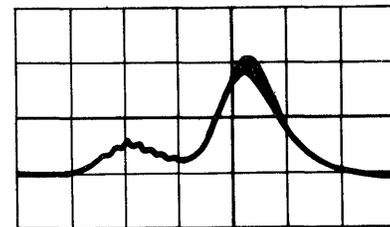
(c)



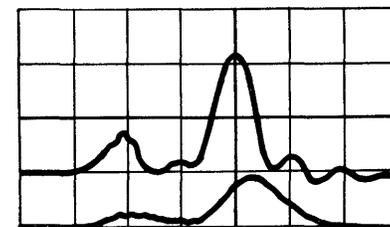
(a)



(b)



(c)



(d)

Fig. 4—Electrical characteristics of the module. Time is measured from left to right and each division is $0.5 \mu\text{sec}$. (a) Drive currents. The upper trace represents the X selection current, the lower trace the Y selection current. Each division is 500 ma . (b) Observed back voltages. The upper trace is the back EMF on the 4-turn winding through the 16 cores of the Y winding, the lower trace that of the 32 cores on the X winding. Each division is 5.0 volts . (c) One and zero output signals observed with the magnet board present and removed. The first low peaks are due to inductive pickup and shuttle. Each major division is 5 mv . The output load is 180Ω .

DESIGN OF THE PERMANENT MAGNET ARRAY

A number of factors must be considered in the design of the permanent magnet array. First, the array of magnets must be simple to manufacture. Since the magnets are the primary storage medium of the memory, they must be capable of retaining their magnetization under very severe conditions. Methods must be devised to register the permanent magnets accurately over the bit locations defined by the wrapped wires and the word solenoids. Finally, the spacing between the individual magnets must be chosen carefully, since one magnet should not disturb either the neighboring magnet or its sensing wire.

For illustrative purposes, the bar magnet will be considered as two magnetic charges $\pm m$ spaced a distance d apart. The magnetic field H at any distance r perpendicular to the center of the magnet is:

$$H = \frac{md}{[(d/2)^2 + r^2]^{3/2}} \quad (1)$$

and is directed parallel to the axis of the magnet. There are two limits to be considered. In the limit of $r \rightarrow 0$,

Fig. 5—Output signals of the module. Time is measured from left to right and each division is $0.5 \mu\text{sec}$. (a) Drive currents. The upper trace is the X selection current, the lower trace the Y current. Each major division is 500 ma . (b) 32 "one" signals observed on one sensing wire from 32 cores on one frame. Each major division is 5.0 mv . The load is 180Ω . (c) 26 "one" signals observed on the 26 sensing wires in one word location. Each major division is 5.0 mv . The output resistance is 180Ω . (d) The output signals for open circuit and matched output loads. The output resistance of the lower trace is 22Ω and is equal to the resistance of the sense wire. Each major division is 5.0 mv .

(1) reduces to:

$$H_0 = \frac{8m}{d^2} \quad (2)$$

The field which acts on a sensing wire placed near the magnet is, then, inversely proportional to the square of the length of the magnet. For values of $r \rightarrow \infty$, (1) reduces to:

$$H_\infty = \frac{md}{r^3} \quad (3)$$

Thus, for neighboring positions, the magnetic field is proportional to the length of the magnet.

Ideally, a permanent magnet should have a large effect on the sensing element just underneath it and no effect on any of the adjacent elements. The ratio of the field for r small to the field for r large should be very high. Using the two previous approximations, the ratio is:

$$\frac{H_0}{H_\infty} = \frac{8r^3}{d^3} \quad (4)$$

In order to reduce the interaction, the magnets should be made as small as possible. A number of other factors, however, prevent the magnet from being made extremely small. The first is that the length of wrapped wire underneath the bar magnet must be large enough to produce a detectable output signal. In addition, the demagnetizing factors associated both with the wrapped wire and with the permanent magnet must be considered. The demagnetizing field of the permanent magnet is inversely proportional to the square of the length d of the magnet. If d is reduced until the demagnetizing field is greater than the coercive force, the effective pole strength m will be reduced. In the case of the wire used as a sensing element, the demagnetizing field of the flux reversed must be less than the applied driving field.

Boards containing the permanent magnet arrays are prepared by etching sheets of Vicalloy I which have been bonded to plastic boards. Vicalloy tape can be obtained in strips about 4 inches wide and 2 mils thick. The Vicalloy is heat treated to produce a saturation magnetization of 5000 gauss and a coercive field of 200 oersteds. The individual magnets are etched using the standard photo resist etched wire technique. Master negatives are prepared such that the individual magnets may be removed by masking out their positions on the negative. Consequently, all information patterns may be prepared from one master negative. The flat form of the magnet simplifies its positioning over the bit location. The bar magnet used is $20 \times 60 \times 2$ mils. The direction of magnetization is parallel to the long dimension. The card containing the magnets is placed in registration by guide pins and is pressed firmly against the solenoids by springs. Consequently, the separation of the permanent magnet from the sensing wire is only a few mils. The two-pole approximation cannot be used to determine the true magnetic field for distances comparable to the bar magnet length. Experimentally, the magnetic field on the wire beneath the permanent magnet is about 20 oersteds. The field on the nearest neighbor is about 1 oersted.

The magnets were spaced unequally in the three dimensions of the present design. In order to reduce the inductance of the solenoid strip encompassing the sensing wires, the wires are spaced 100 mils apart which was considered the minimum distance to avoid lateral interactions. To prevent the permanent magnets of one word

from acting too strongly upon the sensing wires in the next word, the individual solenoids are separated $3/16$ inch. This distance should be kept small to minimize the length of line over which the output signal must travel before reaching the detecting amplifier. Finally, the individual frames are spaced about $1/2$ inch center to center. One quarter of an inch of this spacing is used for the solenoids, the sensing wires, and the supporting board. The remaining $1/4$ inch is taken up by the permanent magnet card and its spring assembly. Thus it is quite easy to slip the card in and out as is shown in Fig. 6.

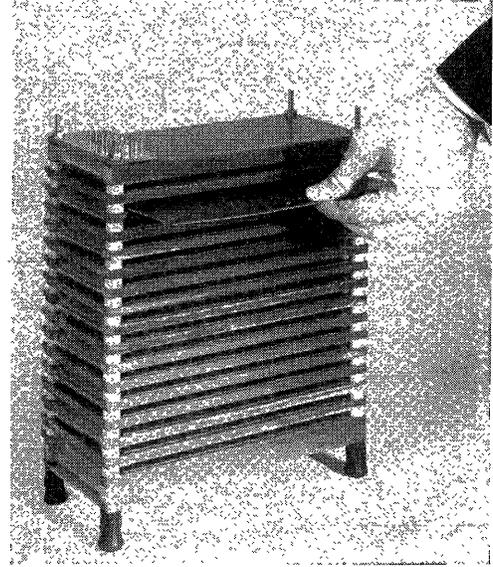


Fig. 6—Rear view of the 512-word memory module. The magnet board is being inserted. It is held against the memory frame by a spring.

THE WRAPPED WIRE USED FOR SENSING

The twistor wire is shown in Fig. 7. A three-mil copper wire has a magnetic tape wrapped around it at an angle of about 45° . The particular material used is 4-79 permalloy. The tape has a coercive force of about 3 oersteds and a cross section of 5×0.3 mils. The length of the wire per bit is determined by the width of the solenoid employed and is made the same width as the bar magnet for simplicity, *i.e.*, 60 mils.

The cross section of the tape must be adjusted to satisfy a number of conditions. First, the ratio of the bit length to tape thickness determines the demagnetizing field. It is desirable to keep the demagnetizing field small since it decreases the effective driving field. Also, the thickness of the tape should be kept below $1/2$ mil to insure that the eddy current losses are not excessive. The amount of material determines the size of the access core and no more material should be included than is necessary to provide a detectable signal. Finally, the wrapped wire consists of a copper conductor used for the transmission of information wrapped with a magnetic tape used for the detection of information at particular locations. Since the magnetic material acts as a loading on the transmission line, it is desirable to keep the

amount of magnetic material to a minimum. The sense wire is a small copper wire and has an appreciable resistance per unit length. It is desirable to use a large diameter wire to minimize the attenuation. Unfortunately, the output signal is determined by the number of times per bit length that the flattened tape wraps around the center conductor. For a given wrapping pitch and bit length, the number of wraps decreases as the diameter increases. Thus, the output signal would be reduced. A compromise must be made between the attenuation which produces a variation of output signal from near bit locations to far bit locations, and the amplitude of the output signals.

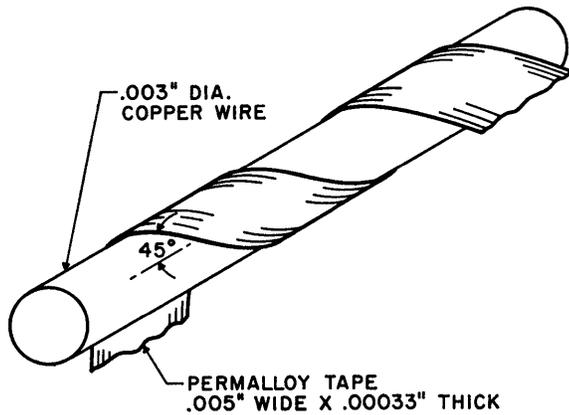


Fig. 7—Helical magnetization by wrapping. The permalloy tape is wrapped continuously around the 3-mil copper conductor.

The uniformity of the output signals is improved if the wire is magnetized in one direction. Consequently the wrapped wire is magnetized before the memory is placed in operation by the passage of a dc current down the wire. The permanent magnet field is directed to maintain the continuous magnetization of the wire. The drive field must switch the wrapped wire into an opposite state of magnetization. As a result, a demagnetizing field is created which opposes the drive field. When the core switch resets the bit, the demagnetization field is in the direction to aid the resetting of the bit. Thus, the wire will remain in a uniformly magnetized state.

THE BIASED CORE SWITCH

An equivalent circuit for the biased core switch with a constant current input is shown in Fig. 8. The net current drive is the number of ampere turns of the bias since each of the X and Y currents provide the same number of ampere turns as the bias. The core can be represented by two circuit elements. The first is a current sink of NI_0 which represents the fact that a given number ampere turns must be applied to the core before it begins to switch. The core, during the switching operation, acts as a resistance R_c ,⁴ which is proportional to

the total flux of the core divided by the product of the switching coefficient⁵ of the material in the core and the mean path length around the core. The core, in switching, generates a voltage which forces current into the load Z_L , which represents the solenoid and its sensing wires. In order to switch current effectively through the core, it is desirable that R_c be made large compared to Z_L . One possibility is to make the flux in the core large.

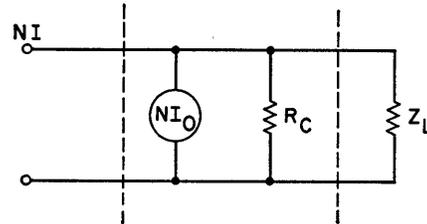


Fig. 8—An equivalent circuit for a biased core switch for a constant current input I . N is the ratio of input to output turns. The switch core is represented by the current sink NI_0 and the resistance R_c .

However, the total back voltage on a selection line is a function of the total inductance of the cores on the line. To reduce the inductance and hence the back voltage, it is desirable to keep the core cross section and thus the total flux small. The minimum flux must be sufficient to supply an output voltage to drive the required current into the load Z_L long enough to complete the sensing. The resistance of the solenoid, its inductance, and the flux, which must be switched in each of the wrapped wire elements contained in the solenoid, determine the required flux of the access core. Since the flux is equal to $\int_0^t v_0 dt$, a convenient flux unit is $mv \mu sec$. The flux required by the 26 wrapped wires is about $30 mv \mu sec$. The flux which must be supplied to drive the inductance with a current of 2 amp is about $50 mv \mu sec$ while the flux necessary to drive the current through the resistance of the solenoid for $1 \mu sec$ is $100 mv \mu sec$. A total of $180 mv \mu sec$ of flux must be supplied as a minimum by the access core. The cross section of the core is made sufficiently large to contain $300 mv \mu sec$ of available flux. Only about $200 mv \mu sec$ of the available flux is used.

The access core is made of ferrite containing cadmium as well as manganese and magnesium. The core has an extremely flat hysteresis loop, $Br/B_s \approx 0.93$, and a very low coercive field, $H_c \approx 0.15$ oersted. If a permalloy core were used, only a few wraps would be required to supply the $300 mv \mu sec$ of flux. Tape cores with a small number of wraps of $\frac{1}{4}$ or $\frac{1}{8}$ mil tape are not as square as the ferrite core. The tape core has higher dynamic resistance for a given flux than the ferrite core, but this factor is less important than the superior squareness and the lower cost of the ferrite core. In fact, the dynamic resistance of the ferrite core is so large that the current

⁴ E. A. Sands, "Behavior of rectangular hysteresis loop magnetic materials under current pulse conditions," *Proc. IRE*, vol. 40, pp. 1246-1250; October, 1952.

⁵ N. Menyuk and J. B. Goodenough, "Magnetic materials for digital computer components," *J. Appl. Phys.*, vol. 26, pp. 8-18; January, 1955.

regulation of the switch is extremely good. The biased core switch improves the rise time of the current delivered to the load compared to the rise time of the drive currents.

SUMMARY

A 512-word 26 bits per word module of a magnetic matrix memory has been developed for the storage of semipermanent information. The memory is capable of random addressing at high cycling speeds. The information is stored in an array of permanent magnets. The presence of a permanent magnet is sensed by a wire wrapped with magnetic tape placed adjacent to the permanent magnet. The magnetic materials used in the memory are permalloy tape, Vicalloy tape, and ferrite cores. As used all materials are relatively insensitive to any change in the ambient temperature and, consequently, the memory may be operated over a wide temperature range.

The electrical characteristics of the memory module can be compared to those of a ferrite core memory. By the use of four turns on the access core, the drive current into the magnetic switch becomes 600 ma. The back voltage, however, is low. The output signal at the word location is 8 mv. The transmission properties of the sense wire must be considered for large memories. The propagation time per bit is about 0.06 $\mu\text{sec/bit}$. Since the sense wire is resistive, the output signal may be attenuated as much as 1 db per 512-word module. Larger memories may be made by interconnecting several

modules, but the number of modules which may be connected to one sense amplifier is limited by the attenuation of the wrapped wire.

The present memory is an initial model which has been developed to demonstrate feasibility of the system as well as to meet certain operational requirements. Models will be made soon which have improved characteristics. In particular, the size of the memory may be reduced by a factor of two or more. The output signal from the sensed bit may be increased. The transmission line properties of the structure may be improved by making the conducting wire larger.

ACKNOWLEDGMENT

The semipermanent memory is the result of the effort of several people and the author is acting as the reporter for the group. J. Janik suggested the use of the permanent magnet and wrapped wire scheme. Dr. H. L. Stadler contributed the magnetic design, while A. J. Munn has been responsible for the mechanical design of the memory module. Test apparatus design and operation have been contributed by J. A. Ruff and J. L. Smith. The design of the permanent memory has been carried out in parallel with the development of a variable memory under the supervision of A. H. Bobeck. The permanent memory has benefited from the early design and the suggestions of this group. The present project was under the supervision of Dr. F. B. Humphrey. The author gratefully acknowledges his assistance in the preparation of the present paper.

A Card Changeable Nondestructive Readout Twistor Store*

J. J. DEBUSKE†, J. JANIK, JR.†, AND B. H. SIMONS†

INTRODUCTION

WITH the steady increase in the required speed and complexity of computing and data processing equipment necessary to handle today's problems, the role of the associated storage systems has been expanding. Generally, a rather large amount of permanent or semipermanent storage capacity is required to store information such as programs, constants, and tables. Besides having a large bit capacity at low bit cost, such stores must be fast, reliable, and flexible in

addressing. The twistor sensing an array of small permanent magnets meets these requirements admirably.

The "Twistor" as a memory element was conceived by Bobeck.¹ It may be used as either a memory or a sensing device. It is as a sensing device that it is used in the store described in this paper. The details of the magnetic structure are given in a companion paper.²

This memory matrix utilizes cards containing a space for a small magnet at each bit position. A magnet is

* The work reported in this paper was done for the U. S. Department of Defense under Contract DA-30-069-ORD-1955.

† Bell Telephone Labs., Inc., Whippany, N.J.

¹ A. H. Bobeck, "A new storage element suitable for large-sized memory arrays—the Twistor," *Bell Sys. Tech. J.*, vol 36, pp. 1319-1340; November, 1957.

² D. H. Looney, "A twistor matrix memory for semipermanent information," this issue, p. 36.

present if a zero is stored and absent if a one is stored. The model described in this paper is word-organized. One row across a card forms a word. These cards are placed over columns of twistor wires, one twistor wire for each bit which is used to sense the presence or absence of a magnet for the bits of the word being addressed.

By combining memory cells using the cards and twistor wire arrays with suitable access, timing, and readout circuitry, a store is obtained compatible with high-speed data systems. Furthermore, the information on the cards is permanent and easy to check. The stored information can be readily changed by removing the cards and replacing them with others.

THE 512-WORD, 26-BIT-PER-WORD, STORE

A block diagram of the store is shown on Fig. 1. Information is stored on 16 cards. Each card stores 32 26-bit words. Computer access to this stored information is via the timing and memory access circuitry. In order to sense the information stored in a particular word location, the computer must supply a "read start" signal and the binary coded address of the desired word. Upon receipt of the "read start" signal, the timing circuit generates the internal timing necessary to complete the operation. The information read out of the memory module is detected and amplified by the read detectors before being passed on to the computer at a logic level of -3.5 volts.

THE TWISTOR MEMORY MODULE

Each of the 512 words has a word access core and a word solenoid associated with it, as shown on Fig. 2. A group of 26 twistor wires, encapsulated in a plastic tape, threads the word solenoids. The region along each twistor wire, defined by the width of the associated solenoid, constitutes one bit of a word. A twistor wire consists of a 4-79 permalloy tape, of cross section 0.005 inch by 0.0003 inch wrapped around a 0.003-inch copper wire at an angle of about 45° . The 26 twistor wires are magnetized in the same direction, which is defined as the set state.

To simplify the description of the operation of the basic memory cell in Fig. 2, only a vertical winding is shown threading the cores. A current "I" of sufficient amplitude in the vertical wire threading a word core reverses the direction of the magnetic flux in the core. This change in flux induces a current in the solenoid which produces a magnetic field strong enough to reverse the direction of magnetization of the sections of the twistor wires within the solenoids.

A twistor bit having an associated magnet is prevented from switching its direction of magnetization by the presence of the external magnetic field of the magnet (bit 25), (Fig. 2). This field is in the same direction as the set state of magnetization of the twistor wire. The rate of change of flux in a twistor bit that is switched induces a voltage in the twistor wire. The passage of cur-

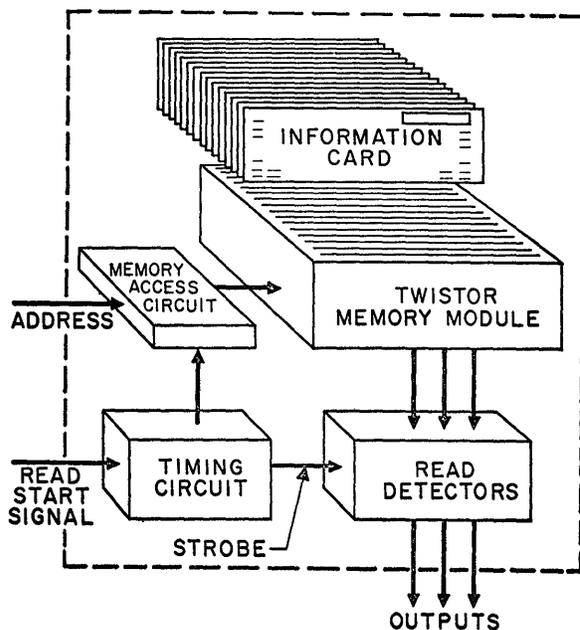


Fig. 1—Basic block diagram of the store.

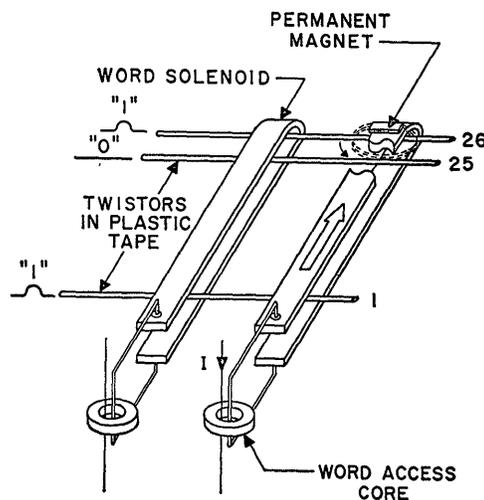


Fig. 2—Basic memory cell.

rent in the opposite direction through the word access core resets the core, which in turn restores the switched twistor bits to their set state. Therefore, in this application, the twistor elements are used to sense the presence or absence of permanent magnets. The presence of a permanent magnet represents a stored zero, and the absence of a magnet represents a stored one.

The 512 core-solenoid combinations are arranged and wired as a biased core switch³ in a 32 by 16 matrix. (See Fig. 3.) Note that the 26 tape-encapsulated twistor wires are in one continuous folded belt. A vertical drive winding threads each core of a particular column with four turns. A horizontal drive winding threads each core of a particular row with four turns. The vertical and

³ J. A. Rajchman, "A myriabit magnetic-core matrix memory," Proc. IRE, vol. 41, pp. 1407-1421; October, 1953.

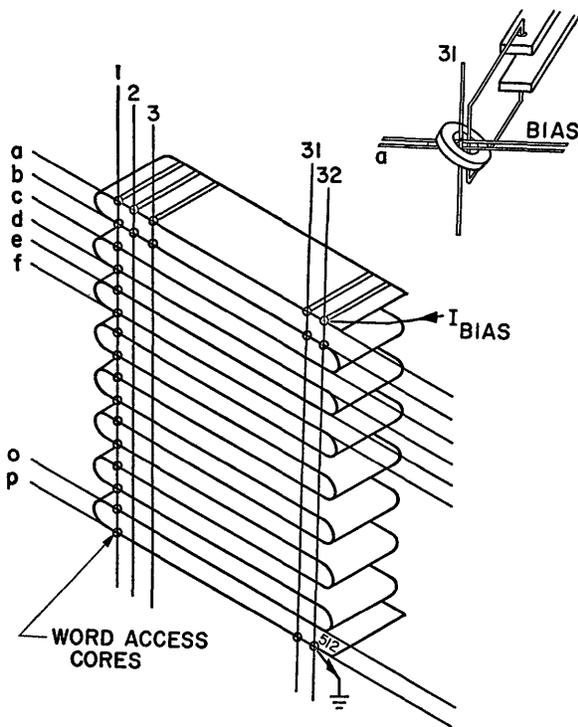


Fig. 3—Memory module biased core switch.

horizontal drive currents and windings produce aiding flux at a cross point in each core. A constant current bias winding threads each core of the matrix. The coincidence of currents in the horizontal and vertical drive windings of a word core overcomes the effect of the bias, and thus switches the core resulting in the readout of the information stored in that word location. The switched bits are restored to their former states by the action of the bias current which resets the word access core when the horizontal and vertical drives are removed. In order to select word 1, Fig. 3, it is necessary to supply pulse currents, of sufficient amplitude and duration, on horizontal drive winding "a" and vertical drive winding "1".

With horizontal and vertical drive currents of 0.6 ampere into 4 turns each, and a bias current of 2.25 amperes, the twistor output for a stored 1 is approximately 8 mv and about 2 mv for a stored zero. When a word access core is switched, the other 31 cores of the row and 15 cores of the column are shuttled by the drive currents. The effect of these shuttle currents and inductive pick-up noise is reduced by staggering the time of application of the drive currents and by strobing the read detector outputs.

In the present 512-word memory module, the transmission properties of the twistor circuit produce an output signal attenuation of as much as one db and a propagation time of about 35 m μ sec. In order not to penalize the signal-to-noise and timing of the words most remote from the read detectors, two modules of the present type may be read with a single set of read detectors.

THE APPLICATION OF THE MEMORY MODULE TO THE STORE

A diode, WECo type 1N2146, is added in each horizontal and vertical drive lead to block "sneak" paths. The leads are interconnected as indicated on Fig. 4. This arrangement permits the selection of one of the 16 horizontal drive leads by a 4 by 4 selection switch and the selection of one of the 32 vertical drive leads by an 8 by 4 selection switch. The selection switches and the decoders comprise the memory access circuits. (See Fig. 5.) Under the control of the binary address supplied by the computer, the decoders select the horizontal and vertical drivers required for access to a particular word. The selected drivers supply the necessary drive currents under the control of the timing circuit.

The outputs of the twistor wires are amplified by the read detectors to logic level (-3.5 volts) and passed on to the computer during the strobing interval. The strobe gates of the read detectors are controlled by the timing circuit.

When the timing circuit receives a "read start" signal from the computer, it generates the horizontal drive, vertical drive, and strobe signals, as shown in Fig. 6. Note that the vertical drive signal occurs approximately 0.5 μ sec after the horizontal drive signal. The selected drive currents flow in the word-access core matrix during the interval the drive signals are applied. The sole-noid current and the twistor output signal are shown on the figure for reference. The strobe gates of the read detectors are activated during the strobe interval of 0.2 μ sec commencing approximately 1.4 μ sec after the "read start" signal.

The function of the initial magnetizing circuit (Fig. 5) is to insure the uniform magnetization of the twistor wires in the same direction. This is accomplished by passing a 10- μ sec, 200-ma current pulse through the twistor wires, and must be done before the store is placed in operation or whenever the program cards are installed or replaced in the memory. This circuit is operated by a switch located on the control console of the computer.

DECODER

The decoding of the address is done in "double-rail" logic. To address the 512-word store, 9 binary address bits and their complements are required, these being divided into 4 groups. Three bits and their complements are used to select one of the 8 top vertical drivers (Fig. 5); 2 bits and their complements are used to select one of the 4 bottom vertical drivers; 2 bits and their complements are used to select 1 of the 4 right horizontal drivers; and 2 bits and their complements are used to select 1 of the 4 left horizontal drivers. The address bits from the computer are the outputs of the flip-flops of the address register and appear as steady ground or -3.5 volt signals. The address bits are assigned so that the first 32 words will appear in order from 1-32 on

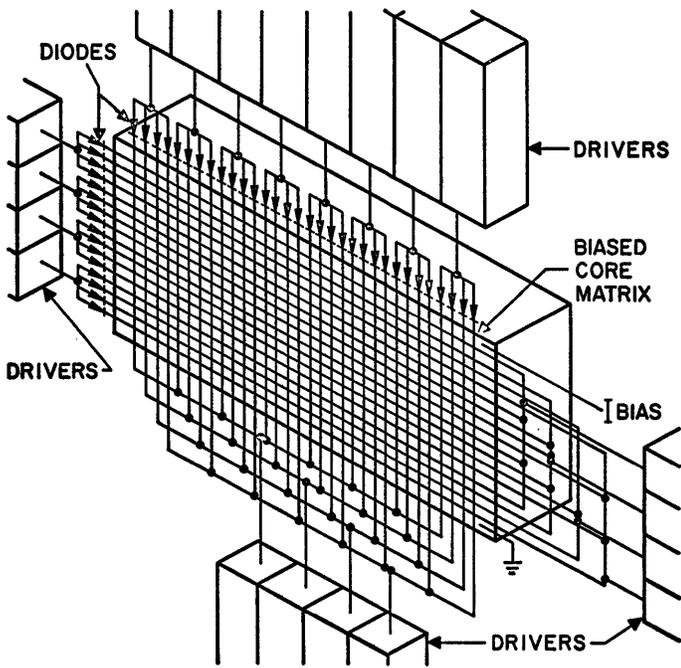


Fig. 4—Interconnection of drive leads.

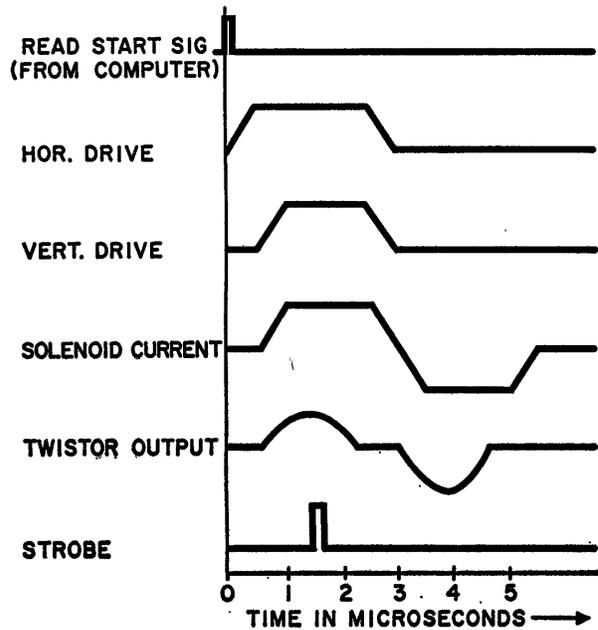


Fig. 6—Timing diagrams.

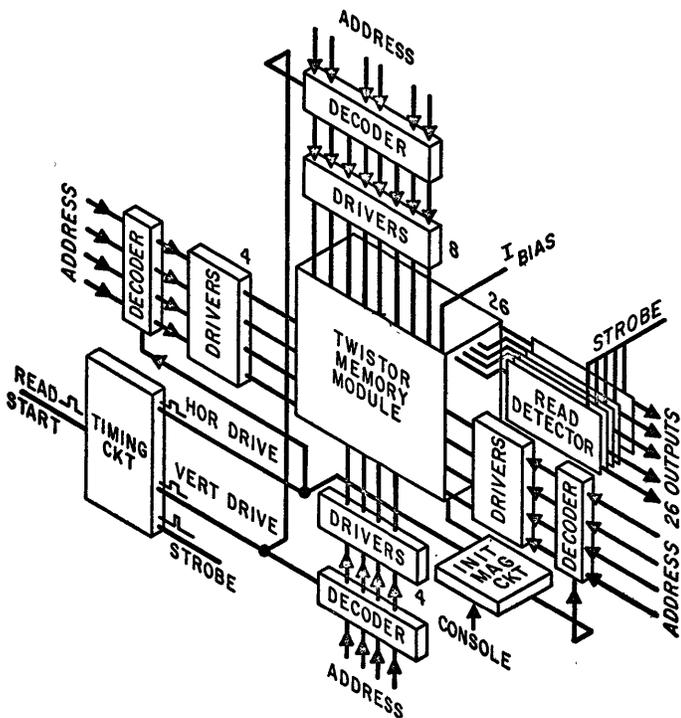


Fig. 5—Complete store block diagram.

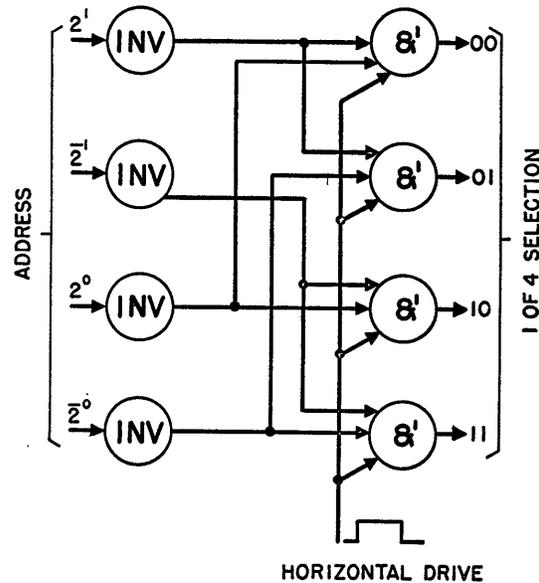


Fig. 7—Decoder logic diagram.

the first program card in the first slot, and the following words through 512 will appear consecutively in the adjacent slots.

One of the 2 horizontal decoders is illustrated in the logic diagram of Fig. 7. The inverters are provided for fan-out purposes. The 2 bits and their complements are combined with the horizontal drive signal (in "and" circuits) to produce a 1-out-of-4 selection. The selected lead remains at -3.5 volts for the duration of the hori-

zontal drive signal. Similar logic is used in the vertical for address decoding.

DRIVERS

The horizontal and vertical drive currents are each controlled by a pair of drivers capable of providing an 0.6-ampere pulse with a $0.5\text{-}\mu\text{sec}$ rise time. A vertical winding (4 turns shown as one wire) with drivers and cores is shown on Fig. 8. The bottom driver of the pair has a current monitoring resistor R8 and a feedback network composed of diode CR1 and resistors R7 and R8. The feedback network monitors, and thereby regulates, the drive current of 0.6 ampere. Transistor Q1 is a WECo type 2N560, and transistors Q2 and Q3 are WECo type 2N1072.

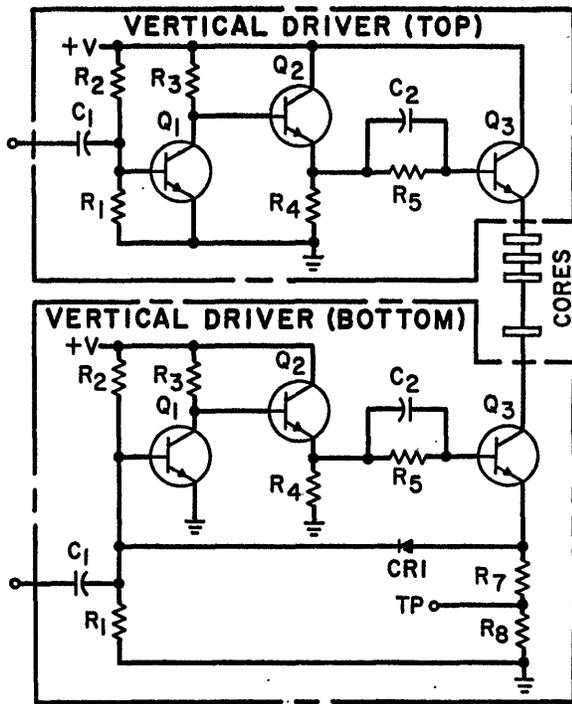


Fig. 8—Driver schematic.

READ DETECTOR

The read detector schematic is shown in Fig. 9. Transistors Q1 and Q2, WECO type 2N559, comprise a stabilized feedback linear amplifier⁴ whose gain is given to a good approximation by

$$\mu_{21} = 1 + R_3/R_4.$$

Transformer T1 is used to match the impedance of the twistor circuit to the input impedance of the amplifier stage. The third stage, transistor Q3, is normally held in a conducting condition by a current from the minus voltage supply through resistor R7. The signals at the collector of Q2 are of such a polarity as to tend to cut off the collector current of transistor Q3. A thresholding function is obtained because an output signal (one) from the twistor, amplified by Q1 and Q2, must overcome the base bias of Q3. The collector circuit of transistor Q3 is designed to be compatible with the connecting logic circuitry of the computer. Transistor Q4 is used to "strobe" the output of transistor Q3. This provides discrimination against signals coming from the memory except when the output signal is expected. Clearly, both a twistor output signal and the strobe signal must be present simultaneously or no output results. Both Q3 and Q4 are also 2N559's.

TIMING CIRCUIT

All of the timing for the store is generated within the store. The "read start" signal from the computer is the

⁴ F. D. Waldhauer, "Wide-band feedback amplifiers," IRE TRANS. ON CIRCUIT THEORY, vol. CT-4, pp. 178-190; September, 1957.

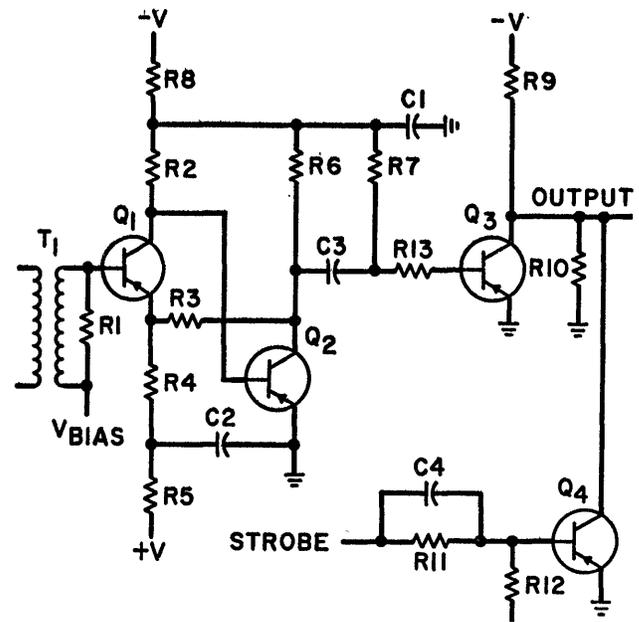


Fig. 9—Read detector schematic.

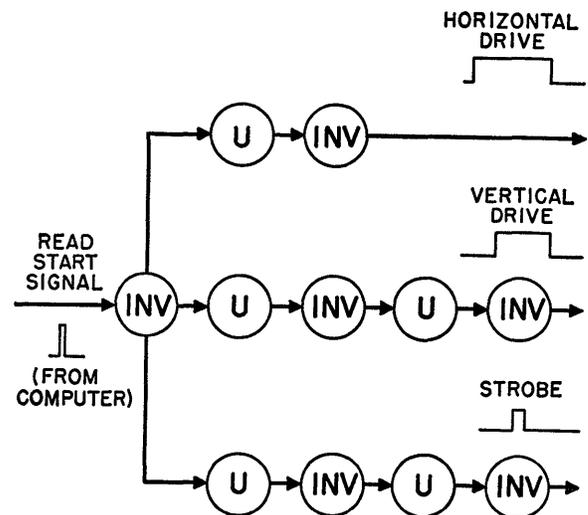


Fig. 10—Timing circuit logic diagram.

only externally generated signal and is used to initiate a series of univibrators (U). (See Fig. 10.) The univibrator is basically a 1-shot multivibrator whose pulse width is determined by an RC time constant. One univibrator is used to generate the 3.0- μ sec horizontal drive pulse (Fig. 6) Two univibrators are used to generate the vertical drive pulse, one univibrator to provide the 0.5- μ sec delay and the other to determine the pulse width. Similarly, 2 univibrators are used to generate the strobe signal.

512-WORD STORE (MECHANICAL CONSTRUCTION)

The store is assembled with its twistor memory module, decoders, drivers, timing circuit, and read detectors as one package. (See Fig. 11.) The weight of the store is 70 pounds and its over-all dimensions are 9 by 21 by 22 inches excluding power supplies.

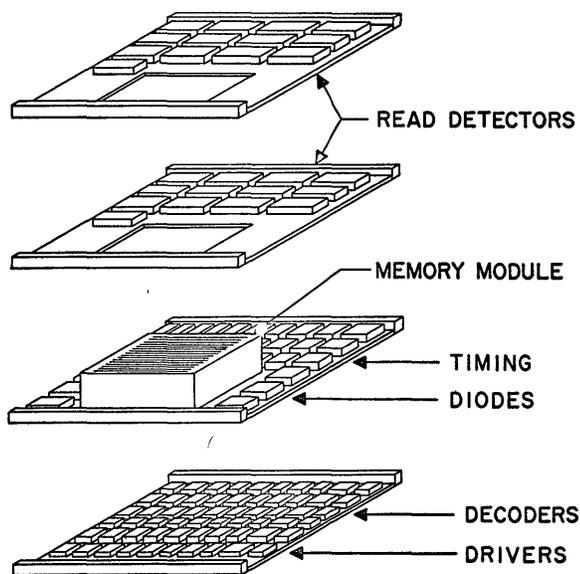


Fig. 11—512-word store (mechanical construction).

The memory module is located toward the front of the package for ease of card changing. The read detectors are located immediately to the rear of the memory to keep the low-level signal leads as short as possible. Two partial planes are used to mount the read detectors. Below these are mounted the memory module, timing circuit, and diode circuits. The bottom plane is reserved for the decoders and drive units.

The entire store can be disassembled by separating the individual planes. Connectors at the front and rear of each of the planes carry signals upward and downward between planes. Computer access is outward through rear connectors.

The store contains 200 2N559 diffused base switching transistors, 20 2N560 diffused silicon switching transistors, 40 2N1072 silicon, high current, switching transistors, and 48 1N2146 silicon diodes. Approximately 50 watts of power is required for the operation of the store. This power is supplied by 3 direct current supplies, -12 volts, +12 volts, and +32 volts.

PROGRAM CARD

The program card consists of 32 rows of 26 or less Vicalloy magnets per row. (See Fig. 12.) The magnets are 0.060 inch long, 0.020 inch wide, and 0.002 inch thick. Two types of cards are used because the twistor tape folds back and forth through the memory and the twistor wires are uniformly magnetized in one direction. Red cards are used in the even slots and are magnetized

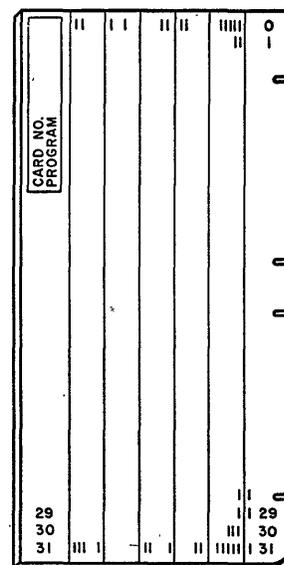


Fig. 12—Program card.

left to right; green cards are used in the odd slots and are magnetized right to left.

Each card has a tab on which the programmer marks the card number and the program symbol when the card is prepared. Lines are provided on the card separating the 26 bits of each word into five 5-bit columns and one 1-bit column to assist the programmer. Further, each word is numbered for his use.

SUMMARY

Information is stored on removable cards which can be prepared quickly by stenographic personnel if desired. Twistor elements are utilized to sense the information which is in the form of tiny permanent magnets. This store uses a basic 512-word twistor memory module and is packaged with all of its associated solid-state circuitry as one integrated and removable unit. It is operated on a 5- μ sec cycle time and requires approximately 50 watts of power. A 2048-word, 26-bit-per-word, store is under development; in principle it operates in the same manner as the 512-word store, except that four twistor memory modules are wired together.

ACKNOWLEDGMENT

This work has been done under the supervision and guidance of J. E. Corbin. The mechanical design has been accomplished by W. T. Drugan and members of his group, C. H. Williams, W. L. Richardson, and C. W. Mensch. The read detector was designed by D. C. Weller, and the organization of the access circuitry proposed by W. B. Gaunt.

Square-Loop Magnetic Logic Circuits

EDWARD P. STABLER†

INTRODUCTION

FERRITES, ferroelectric capacitors, and some ferromagnetic materials all possess bulk characteristics which are usually referred to as square-loop properties. It has been recognized for some time that these properties are particularly compatible with digital-device requirements. This paper attempts to categorize the various methods which are utilized in device synthesis. In addition, it introduces an equivalent circuit model of the devices to determine some of their attributes and limitations. The model is essentially a nonlinear resistive and reactive network whose parameters are determined by the internal state of the device and by its present state of excitation. For the sake of simplicity, the discussion will be limited to magnetic devices. Other authors have pointed out that a direct ferroelectric equivalent exists for multiple-aperture magnetic devices.¹

SIGNIFICANT PROPERTIES OF NETWORK ELEMENTS

The significant properties of an element of square-loop material are threshold, memory, and saturation. These properties are described more precisely below. Some rather gross assumptions are made; however, a more refined description does not seem warranted because of the increased analytical difficulty. The basic magnetic element, shown in Fig. 1, is a cylinder whose

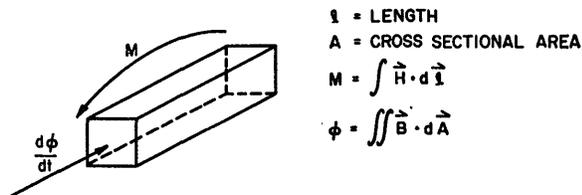


Fig. 1—Elementary magnetic element.

length is considerably larger than either of its other dimensions. The properties of the element are described in terms of the terminal magnetic variables. The cylinder is a two-terminal circuit element. The magnetomotive force M between the two terminals is a function of the flux ϕ passing through any cross section. In this approximation, the magnetic field is assumed to be constant over the length of the element.

† General Electric Co., Syracuse, N. Y.

¹ T. E. Bray and B. Silverman, "Shaping magnetic and dielectric hysteresis loops," *Proc. Special Tech. Conf. on Solid-State Dielectric and Magnetic Devices*, Catholic University of America, Washington, D. C.; April, 1957.

Threshold Property

For values of M below a threshold value M_0 , the terminal relationship is

$$\frac{d\phi}{dt} = K_1 \frac{dM}{dt}$$

However if M exceeds M_0 , switching may take place. The terminal variable relationship is

$$\frac{d\phi}{dt} = K_1 \frac{dM}{dt} + K_2(M - M_0).$$

The first term represents the reversible flux change and the second term represents the irreversible flux change during switching. The element has a negative threshold as well as a positive one. If M is negative and has a magnitude greater than M_0 , switching may also occur, hence

$$\frac{d\phi}{dt} = K_1 \frac{dM}{dt} + K_2(M + M_0).$$

Memory Property

The change in ϕ during switching is the sum of two terms. The first term, proportional to K_1 , is a reversible term. It represents the lossless linear flux change caused by an applied magnetomotive force. The second term, proportional to K_2 , is an irreversible change. A net change in ϕ proportional to K_2 will occur during switching. The implication is that ϕ has many stable values for zero applied magnetomotive force. The memory of the element is associated with this property. We can define an internal state S , which is related to the flux ϕ , in the element with zero applied field as

$$S = K_0\phi \quad \text{with} \quad M = 0.$$

Choose K_0 so that the magnitude of S never exceeds unity.

Saturation Property

The total irreversible flux change that can take place is limited by saturation. When the material is saturated by a positive drive, the terminal relationship is

$$\frac{d\phi}{dt} = K_1 \frac{dM}{dt}$$

for any applied magnetomotive force greater than $-M_0$. In negative saturation a similar expression is obtained:

$$\frac{d\phi}{dt} = K_1 \frac{dM}{dt}$$

for all M less than $+M_0$. Saturation is related to the irreversible flux changes which have taken place. We have chosen the constant K_0 so that

$$S = +1 \text{ for positive saturation}$$

and

$$S = -1 \text{ for negative saturation.}$$

Now we can summarize the complete set of terminal relationships.

1) We define $S = K_0(\phi - K_1 M)$, where we have subtracted the reversible flux contribution caused by an applied mmf.

$$2) \frac{d\phi}{dt} = K_1 \frac{dM}{dt} \quad \text{for } M < |M_0|,$$

$$3) \frac{d\phi}{dt} = K_1 \frac{dM}{dt} \quad \text{for } \begin{cases} S = 1 \\ M > -M_0 \end{cases}$$

$$4) \frac{d\phi}{dt} = K_1 \frac{dM}{dt} \quad \text{for } \begin{cases} S = -1 \\ M < M_0 \end{cases},$$

$$5) \frac{d\phi}{dt} = K_1 \frac{dM}{dt} + K_2(M - M_0) \quad \text{for } \begin{cases} |S| \neq 1 \\ M > M_0 \end{cases},$$

$$6) \frac{d\phi}{dt} = K_1 \frac{dM}{dt} + K_2(M + M_0) \quad \text{for } \begin{cases} |S| \neq 1 \\ M < -M_0 \end{cases}.$$

The constants K_1 , K_2 , K_0 , and M_0 depend on the bulk material and the dimensions of the element. M_0 is usually a function of S , but this fact will be neglected here.

A fourth property implicit in the above relationships is symmetry. If we conduct an experiment starting from state S , applying a drive $M(t)$, we observe the response $\phi(t)$. This experimental result implies that an experiment starting at $-S$, consisting of the application of $-M(t)$, will have the result $-\phi(t)$.

All of the terminal relationships listed are linear differential equations. The magnetic element is nonlinear because the applicable differential equation depends on the applied drive and the internal state. The next section introduces a nonlinear electrical circuit equivalent to the magnetic circuit element.

EQUIVALENT ELECTRICAL CIRCUIT

The equivalent electrical circuit will consist of linear electrical components and switches which are actuated at the turning points

$$|S| = 1$$

or

$$|M| = M_0.$$

We choose to relate

$$\frac{d\phi}{dt} \sim i \text{ electrical current.}$$

$$M \sim e \text{ voltage.}$$

The constraints on $d\phi/dt$ and M , imposed when several elements form a magnetic network, are exactly the constraints on i and e in electrical networks. In Fig. 2 we show three two-terminal electrical networks for which the relationship of the terminal variables e , i is identical to the relationship between M and $d\phi/dt$. The equivalent electrical network that should be used is a function of the internal state S and the applied voltage.

A capacitor appears in the equivalent circuit because we are forming the analog of the magnetic variable relationship. This relationship is different from that observed at the electrical terminals of a winding on the magnetic element.

If an electrical circuit, such as a drive winding or an output winding, is coupled to an elementary magnetic element, the relationship between the terminal variables M and $d\phi/dt$ is modified. Fig. 3(a) shows a magnetic element with a coupled electrical circuit which has been reduced to its Norton equivalent circuit. The previously derived circuit is appropriately modified by the addition of a series impedance numerically equal to $Y_e N^2$ and a series voltage source numerically equal to NI_e . Modification of the magnetic-network-element parameter relationships by means of coupling to electrical circuitry is useful in some synthesis problems.

When two or more magnetic elements are coupled by the same electrical circuit, an extension of the preceding technique is used to obtain an equivalent circuit. This procedure is not always desirable because there is no longer a close relationship between the graph of the derived equivalent circuit and the geometry of the magnetic device. In such cases, it is preferable to solve the network using a mixed (magnetic and electrical) set of independent variables.

PHYSICAL INTERPRETATION

It is reasonable to expect that the equivalent circuit parameters can be determined from the dimensions of the element and the properties of the bulk material. From the magnetic material properties, we obtain:

B_s = saturation flux density,

H_0 = threshold magnetic field,

S_w = switching constant,

μ = small signal permeability.

The magnetic element has a length l , and a cross-sectional area A . We solve for the circuit parameters:

$$K_0 = \frac{1}{B_s A}$$

$$K_2 = \frac{2B_s A S_w}{l}$$

$$K_1 = \frac{\mu A}{l}$$

$$M_0 = H_0 l.$$

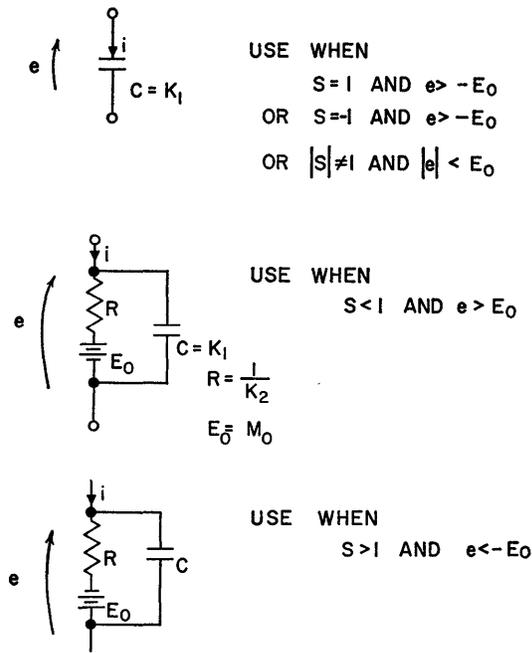


Fig. 2—Equivalent electrical circuits.

In practice, these values are reasonably accurate with the exception of the solution for K_1 . K_1 represents the reversible flux-change term and depends quite noticeably on dimensional relationships other than those mentioned, upon the electrical winding configuration and the internal state S . One contributing reason for this effect is that the small signal permeability of the materials used is often less than two orders of magnitude greater than that of air. It has been assumed that the flux entering or leaving the magnetic element through its walls was negligibly small compared to the amount of flux entering or leaving the ends of the element. Careful design is necessary if this assumption is to be a useful one.

Although the reactive nature of the equivalent circuit plays an important part in the analysis of a given device, it does not have the nonlinear property utilized in the digital-device synthesis. For this reason only the nonlinear resistive portion of the equivalent circuit will be used during the discussion of synthesis techniques. As a result, the network will consist of linear resistors, batteries, and switches which provide the nonlinear characteristic. In addition, an internal state of $S = \pm 1$ will be indicated by an arrowhead in the usual way. Elements in other internal states will have no arrowhead notation. Once a network has been synthesized in graph form, it is essential to perform a thorough analysis to fix the optimum geometric ratios and winding configurations. Fig. 4 shows the simplified equivalent circuit and notation.

LOGICAL FUNCTION SYNTHESIS

One method of synthesis may be called flux steering. In this type of operation, the device is put into an ag-

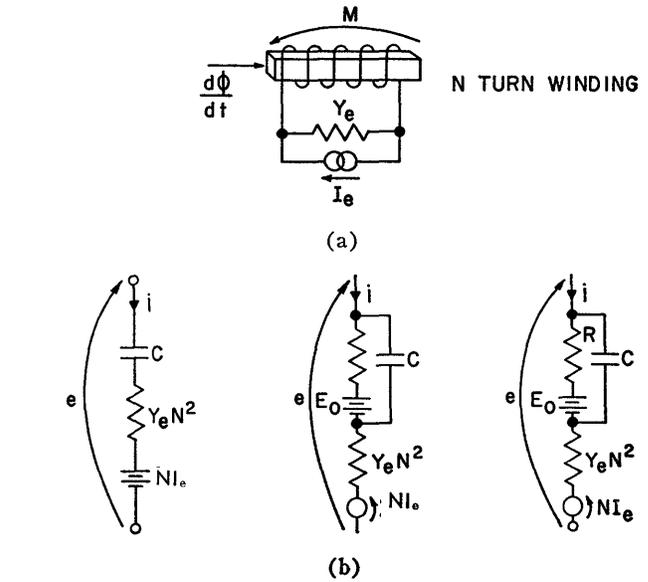


Fig. 3—Modified equivalent circuits for magnetic element coupled to an electrical circuit.

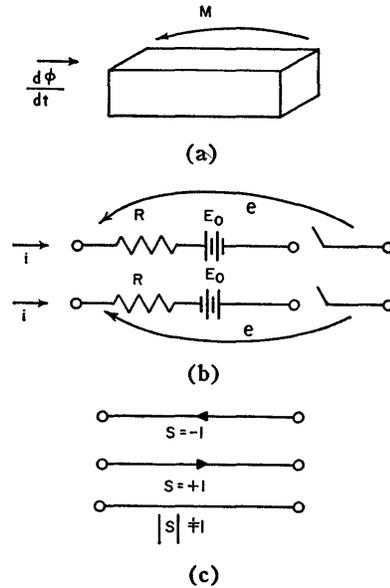


Fig. 4—(a) Elementary unit, (b) simplified equivalent circuits, (c) graphical notation.

gregate internal state dependent on the input binary variables. For each internal state, the output binary function is either equal to unity or equal to zero. When the internal state corresponds to a unity output, the device will respond to a drive D by having a large flux change all along a closed loop L called the output loop. The output function is zero when the response to the drive D is different from that previously described. Either the path along which switching takes place (along which $d\phi/dt$ is large) will differ from L , or there may be no switching at all. Any Boolean function may be written in canonical form as the conjunction of disjunctive polynomials or, alternatively, as the disjunction of conjunctive polynomials. We choose the former

representation to show a direct device synthesis. As an example, let

$$g = f(X_1, X_2, X_3, X_4)$$

$$g = (X_1 + X_2)(X_3 + X_4)(X_1 + X_3)$$

where g is a binary function of the four binary-input variables.

A device which will generate to this function is a three-input AND gate as shown graphically in Fig. 5. It is shown initially in its rest state prior to the reception of input drives. The closed path formed by l_1, l_2, l_3 , is the output loop L . The control legs l_{c1}, l_{c2} , and l_{c3} may be switched during the input period by drive windings controlled by the input variables X_1, X_2, X_3, X_4 placed on the control legs. Fig. 6 shows the approximate equivalent circuit of the device during an input period in which

$$X_1 = 1$$

$$X_2 = 1$$

$$X_3 = 1$$

$$X_4 = 0.$$

This input signal will cause all three elements forming the output loop L to switch. The device may be designed so that the control legs saturate, or reach $S = -1$, at the same time that the controlled legs l_1, l_2, l_3 reach $S = -1$. A subsequent drive tending to switch the output loop L will switch each element of the output loop. If the input state had been

$$X_1 = 1$$

$$X_2 = 1$$

$$X_3 = 0$$

$$X_4 = 0,$$

then leg l_2 would not have been switched during the input period. Consequently, l_2 would not switch during the output drive. There are a number of variations of this operation which tend to improve its characteristics. The example given, however, sufficiently illustrates the principle of operation of a class of gates called flux-steering gates. We restate this principle for emphasis.

In a flux-steering gate the internal states of a number of elements in an output loop L are controlled by a number of control elements. Switching takes place in every element in L during the output period if, and only if, the output binary function is equal to unity.

It is clear from the discussion that theoretically a single flux-steering gate can generate any combinational logical function.

Gates of this type have been constructed at the General Electric Electronics Laboratory. Reasonable power gain is available. Fig. 7 shows photographs of the output signal of a two-input flux-steering gate for unloaded and loaded operation.

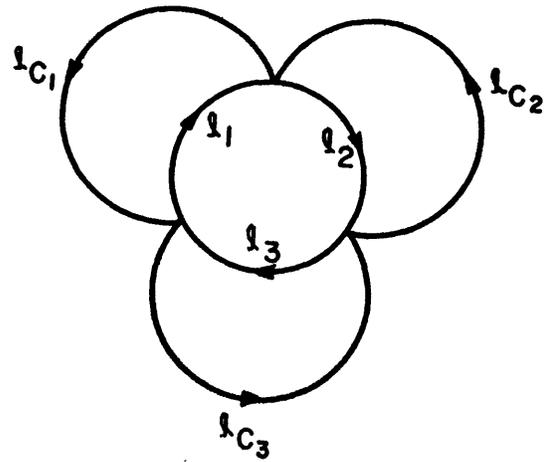


Fig. 5—Graph of three-input AND gate in initial stage.

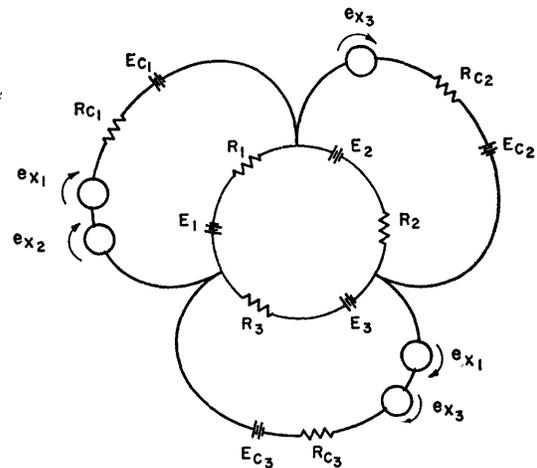


Fig. 6—Equivalent circuit of three-input AND gate during input period.

A second method of synthesis is named flux summation.² Again the function is written as the conjunction of a number of disjunctive polynomials. If the Boolean function is the conjunction of n disjunctive polynomials, the device will have n input elements, $n - 1$ shunt elements, and one output element. The device has a rest state which precedes any input time period. The geometry is such that switching takes place in the output element only when all the input elements are switched during the input period. As an example, we will synthesize the function

$$g = (X_1 + X_2)(X_3 + X_4)(X_1 + X_3).$$

A graph of the device in its rest state is shown in Fig. 8. Elements l_1, l_2 , and l_3 are input elements; elements l_4 and l_5 are shunt elements which may be combined if desired; and element l_6 is the output element. All elements have the same cross-sectional area. During the input period, elements l_1, l_2 , and l_3 may be driven and switched. The element l_6 has a high threshold because of its additional

² N. F. Lockhart, "Logic by ordered flux changes in multipath ferrite cores," 1958 IRE NATIONAL CONVENTION RECORD, pt. 4, pp. 268-278.

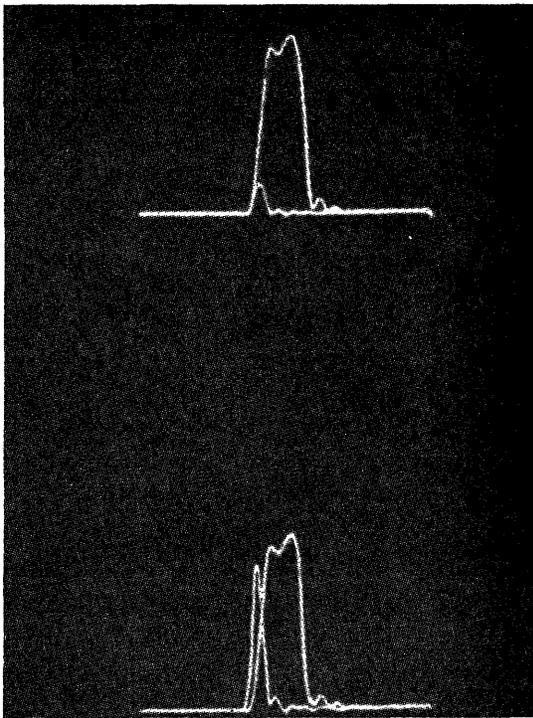


Fig. 7—Flux-steering gate output. Scale: 0.05 v per turn per division (vertical) and 0.2 μ sec per division (horizontal). Top: unloaded; bottom: loaded.

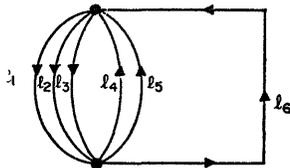


Fig. 8—Graph of flux-summation gate.

length. As a result, if two or less of the input elements are switched, the output element will not switch. The output element will switch only if all three of the input elements are switched. This results from the fact that the two shunt legs will saturate when any two of the input legs switch under the influence of input drives. If all three input elements are switched, the two shunt legs become saturated before the input elements saturate and cannot undergo further large flux changes. Switching in the output element ensues. The threshold M_0 of the output element is made considerably greater than that of the two shunt elements.

It is obvious that the device forms a three-input AND gate as shown. Multiple windings on the control elements perform the disjunction operation. On element 1, two windings are placed; one is excited whenever $X_1=1$, and the other is excited whenever $X_2=1$. The excitation provided by either winding is sufficient to switch element 1. The other control elements are wound in the same way. Various means are available for sensing the output element. The shunt elements may be combined into a single element or may be used to generate other functions of the input variables. This type of gate is only briefly discussed here but is covered thoroughly

by Lockhart.² A single gate of this type can generate theoretically any combinational logical function. The principle of this gate is restated for emphasis. In a flux-summation gate, switching of an output element is controlled by the presence of shunt elements in parallel with the output element. During the input period, the switching of input elements will cause switching in the shunt elements. Switching will occur in the output element only if the shunt elements saturate before the input elements saturate. Actually this description has been restricted in order to emphasize the principle.

A third type of logical-function synthesis is the relay analog method. It is related to the flux-steering gates but is sufficiently different to warrant separate treatment. The graphs of the relay analog unit in its two possible input states are shown in Fig. 9. The closed loop formed by l_0 , l_1 , and l_c is capable of storing information in the same way as a magnetic-memory core, by saturation of the three elements in a clockwise or counterclockwise direction. All three elements have the same cross-sectional area, and normally l_c has a higher threshold than either l_1 or l_0 . The closed loop is initially saturated in one direction or the other by means of an input variable X . Suppose $X=1$ corresponds to counterclockwise saturation. Now the element is read out by applying one mmf, M , as shown in Fig. 10. As a result of M , a flux change will take place in Z_i (the input lead) and in Z_1 (the unity output lead). Little or no irreversible flux change will take place in Z_0 (the zero-output lead). It is clear that these devices can be cascaded to form a wide variety of functions. A symmetric tree of three input variables is shown in Fig. 11, along with the relay equivalent. The output leads are Z_0 , Z_1 , Z_2 , Z_3 where the subscript refers to the number of input variables equal to unity. Fig. 11 shows the internal state of the device when $X_1=0$, $X_2=1$, $X_3=1$. Theoretically, devices of this type may be cascaded to generate any combinational function.

Relay analog elements have been built at the Electronics Laboratory. The ratio of the irreversible flux changes which take place in the two output paths, Z_1 and Z_0 , is about 15:1 for a single stage. Fig. 12 is a photograph of $d\phi/dt$ for the two paths.

We have described three different techniques of synthesizing a gate to generate a combinational logical function. The problems associated with the input and output circuitry have not been considered. There are a number of different types of circuitry which may be used to couple the magnetic gates.³⁻⁷

³ A. Wang, "Magnetic delay line storage," *PROC. IRE*, vol. 39, pp. 401-407; April, 1951.

⁴ R. D. Kodis, *et al.*, "Magnetic shift register using one core per bit," 1953 IRE CONVENTION RECORD, pt. 7, pp. 38-42.

⁵ V. Newhouse and N. Prywes, "High-speed shift registers using one core per bit," *IRE TRANS. ON ELECTRONIC COMPUTERS*, vol. EC-5, pp. 114-120; September, 1956.

⁶ D. Loev, *et al.*, "Magnetic core circuits for digital data-processing systems," *PROC. IRE*, vol. 44, pp. 154-162; February, 1956.

⁷ M. Karnaug, "Pulse-switching circuits using magnetic cores," *PROC. IRE*, vol. 43, pp. 570-584; May, 1955.

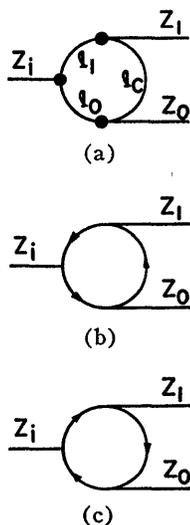


Fig. 9—(a) Relay analog unit, (b) relay analog unit storing a "1," (c) relay analog unit storing a "0".

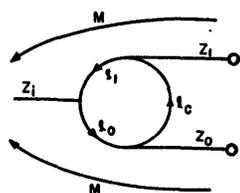


Fig. 10—Output period drives applied to relay analog unit.

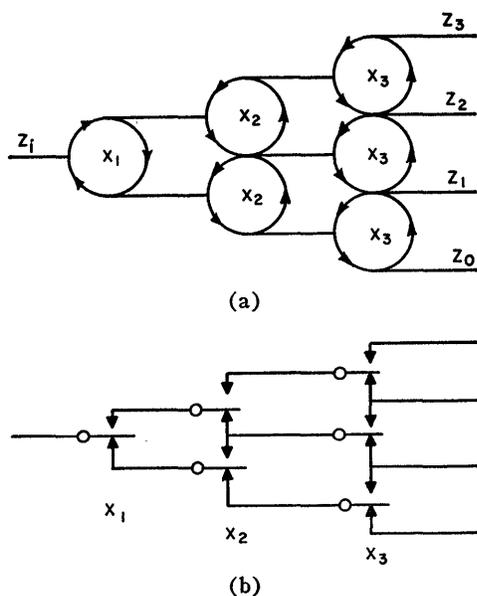


Fig. 11—(a) Relay analog symmetric tree, (b) relay symmetric tree.

In general, all the techniques successfully used in conventional-core logic may be used for multiple-aperture gates. In the next section, we discuss another technique of interconnection fundamentally different from core logic output circuitry and unique to multiple-path logic gates.

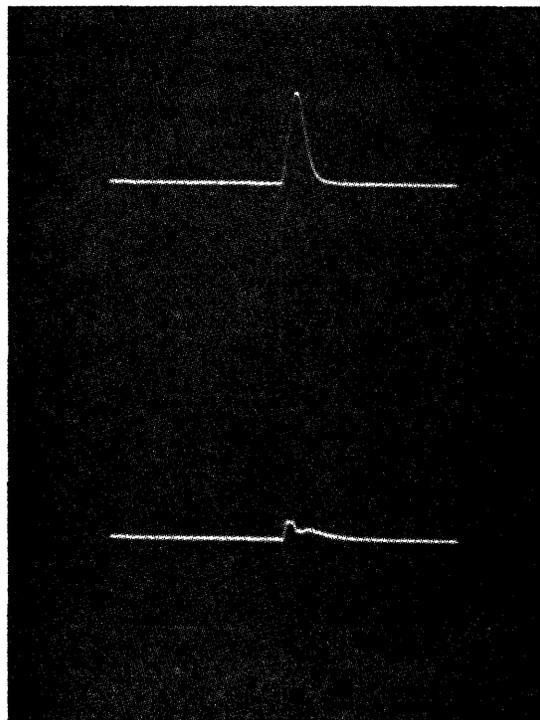


Fig. 12—Signal and noise responses of the relay analog unit. Time scale: 1 μ sec/cm.

MAGNETIC INTERCONNECTION

The conventional magnetic-core logic interconnection circuitry is used to force switching of the input leg of one gate when switching occurs in the output leg of another gate. Since both these events are merely flux changes, the possibility of coupling the two entirely within the magnetic medium seems promising. The magnetic-interconnection network should permit unambiguous signal propagation and should not adversely affect the operation of the units which it couples.

Fig. 13(a) shows a graph of a magnetic-network element with properties similar to an electrical diode. The approximate terminal characteristics are given in Fig. 13(b). Leg l_d is saturated as shown. Leg l_r has a much higher threshold than l_d , and a much higher cross-sectional area and saturation flux. The element is shown in its initial state. The diode element will be much more responsive to a positive-applied M than to a negative-applied M . Information can be transmitted from left to right, but not in the reverse direction. If the diode unit is used to transmit information, switching may take place in l_d . As a result, it will no longer have the desired diode characteristic and must be reset. A drive applied to l_r forces l_d back to its original saturation state. These diode units have the required properties for direct interconnection of two magnetic elements. The next paragraph demonstrates their utility by describing a digital-delay unit in which the storage locations are coupled through magnetic diodes. It is important to note that repeated transmission of information in a single direction will lead to saturation of l_r so that l_d cannot be

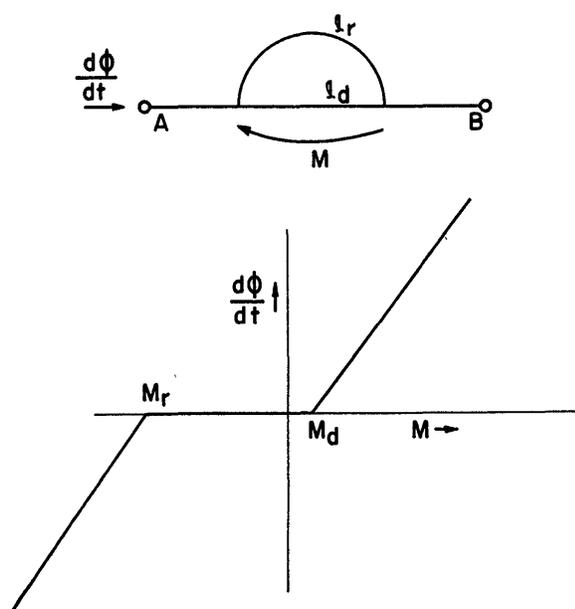


Fig. 13—Magnetic diode terminal characteristics.

reset. In any specific design, techniques can be found to prevent this occurrence. Also, other types of coupling networks can be used in which this problem does not arise. The diode described above is only intended to indicate an approach to the problem.

DIGITAL-DELAY COMPONENT

Fig. 14 illustrates a section of a digital-delay device which uses magnetic diodes to couple information-storage units. In order to advance the stored "1" to the right, two pulse drives occur in sequence. First, all odd-numbered vertical legs are driven upwards. The diodes prevent information flow to the left. Only the storage location storing a "1" will switch. For the state shown in Fig. 14, l_1 and l_2 will switch. The next pulse resets the diodes. The final state is shown in Fig. 14(c). The stored "1" has moved to the right. The next advance is caused by a pulse drive applied to all the even-numbered vertical legs, followed by a diode reset drive. We have described the digital-delay unit only briefly because we intend to illustrate the possibilities of the synthesis approach rather than to describe a practical device.

The magnetic-diode element may also be used to couple the output leg of a gate to an input leg of another gate. It is possible to construct fairly complicated logical machines in which the information is propagated entirely within the magnetic medium.

ALL-CORE NETWORKS

Networks built entirely of simple cores and copper windings exist which are equivalent to any of the multiple-aperture devices which have been described. A brief introduction to the subject is given here.

When magnetic elements are combined in a network to form a multiple-aperture device, nodal constraints are imposed on the flux levels in the elements. At a node

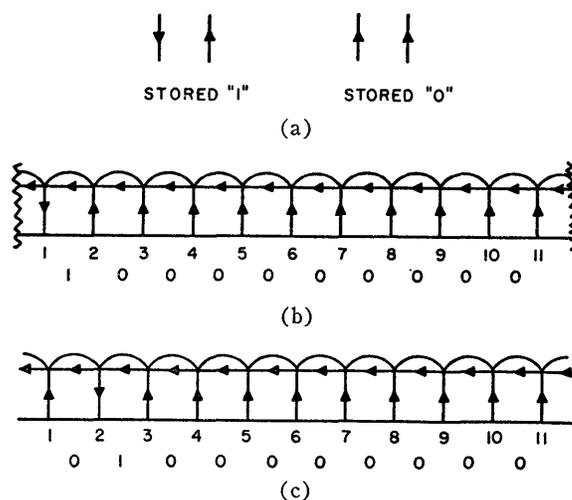


Fig. 14—Digital-delay element.

$$\sum \phi = 0$$

and

$$\sum \frac{d\phi}{dt} = 0.$$

If windings on several different cores are connected in series to form a short-circuited loop.

$$\sum e = 0$$

for the series loop. If there are no other impedances, all the voltages in the loop are induced voltages

$$e = N \frac{d\phi}{dt}.$$

The flux changes which take place in the cores coupled by the series loop obey the following equation

$$\sum N \frac{d\phi}{dt} = 0.$$

This constraint is similar to the nodal constraint of magnetic networks. Following this procedure, a core circuit equivalent to any multiple-aperture device can be found. The relative advantages of the two classes of circuitry will be discussed in a subsequent paper.

CONCLUSIONS

We have described an elementary magnetic element. This can be used as a building block for any multiple-aperture device. We have also presented a terminal variable relationship of the element, and introduced a nonlinear electrical equivalent circuit which may be used to analyze the operation of a multiple-aperture device. Three synthesis techniques have been given for obtaining a multiple-aperture device to generate any combinational logical function. Such devices may be coupled by conventional-core logic circuitry.

The possibility of coupling logical gates entirely within the magnetic medium has been discussed briefly.

A simple example of a magnetic "diode" unit was used to demonstrate that square loop materials can be used in interconnection circuitry. If this procedure is followed, many logical operations may be performed on the input formation while the information is kept continuously within the magnetic material.

All-core networks may be derived which are equivalent to the multiple-aperture devices. The networks which result have many advantages over their multihole counterparts. These networks will be discussed in a future publication.

FUTURE PROSPECTS

Coupling of logical gate devices within the magnetic medium has many attractions.

- 1) Compatibility—the physical input quantities are of the same magnetic nature as the physical outputs. Direct coupling permits scaling down physical dimensions to reduce the power consumption.
- 2) Resistance to noise—the magnetic elements are natural integrating devices. In a noisy environment they will not respond to large but short-lived noise impulses.
- 3) Reliability—the reliability of magnetic materials is seldom questioned. The circuits described here contain no electrical components except for a clock pulse source and drive windings.

Future effort should be devoted to developing simple, flexible magnetic-interconnection circuitry suitable for coupling the various types of logical gates.

Relative Merits of General and Special Purpose Computers for Information Retrieval

A. OPLER† AND N. BAIRD†

INTRODUCTION

INCREASING attention to automatic information processing is being given by all sections of our technology, commerce, and military operations. One of its more important aspects is the use of computing machines for storage and subsequent retrieval of information by request.

There have been two simultaneous patterns evolving in the last decade. One group has borrowed the equipment used for standard accounting and engineering calculations and has demonstrated the practicability of automatic information retrieval (IR) using such equipment. A second group has concentrated on designing special equipment for use solely for mechanized information retrieval.

We are at present in a transition period and it seems appropriate at this time to review the progress made using each of the two approaches and perhaps to introduce some constructive cross-fertilization. For the most part, this field, like many new and exciting disciplines, has produced much controversy, strong prejudices, and a tendency to place personal viewpoint above impartial analysis. Let us hope this "feudal" period is over.

INFORMATION RETRIEVAL ON GENERAL PURPOSE COMPUTERS

Excluding military systems, there have been approximately eighteen information retrieval systems pro-

grammed and debugged for general purpose (GP) computers. Some of these were written primarily for purposes of exploration and others have been actually made operational. From a review of this activity, we are now able to draw a number of generalizations regarding the performance of general purpose computers in the information retrieval area.

- 1) The high-speed computer has proven satisfactory for both exploration and operation. Intermediate speed machines have been satisfactory only for exploration, for operation of simple searching schemes or use with small collections.
- 2) The files of information to be stored were maintained on magnetic tape (one exception employed magnetic disk storage).
- 3) The full gamut of available machines has been used for information retrieval and it appears that no currently available logical design is markedly superior to any other. Information retrieval systems tend to take many forms. For each IR system formulation and for each machine design, there will be special programming techniques required.
- 4) A remarkable variety of document storage formats and retrieval schemes has emerged. With the general purpose computers, it appears that the searching system designer is relatively free to build using the classification system best suited to his needs. Indexing schemes as simple as Dewey Decimal and as complex as those required for

† Computer Usage Co., New York, N. Y.

describing the details of the kinetic processes involved in the synthesis of pharmaceuticals have been programmed with little real difficulty.

- 5) Most computer searching systems are designed to input the search requirements in formats suitable for use by personnel unfamiliar with computers and to produce outputs designed for a similar group.
- 6) Cost of development of such computer systems has been very high and the absence of a suitable information processing language compiler has been sorely felt.
- 7) Once a system is developed, the actual searching and file maintenance costs have proved to be moderate.
- 8) The most common technique is that of searching a magnetic tape file from beginning to end while seeking answers to more than one question (multiplexing). The searching speeds so obtained have proven better than first anticipated. Common speeds range from 1000–15,000 document interrogations per question per minute. As each new program is written, valuable experience is accumulating regarding the best programming and storage techniques to use.
- 9) Document collections thus far used have ranged from 1000–50,000 items. In every case, these collections have been of “more than average” value to the sponsoring organization. Typical systems search important collections of patents, developmental chemicals, and reports especially pertinent to the sponsor’s area of interest. No one has yet been willing to expend funds to index, code, and store the complete contents of newspapers, encyclopedias, or even technical journals.
- 10) In no case has the acquisition of a large computer solely for information retrieval been recommended or even suggested. Both the developmental and operational searching systems share machine time with other technical operations.

SPECIAL PURPOSE INFORMATION RETRIEVAL RELAY MACHINES

Compared to these rather imposing accomplishments, the work done to date by the special purpose retrieval machines has seemed less spectacular. During the last five years, the use of the GP *electronic* computer has been explored while the special purpose machines developed have been primarily *electromechanical*. Special purpose electronic machines are about to appear. It will probably be some time before we can obtain as much perspective here as we have on the general purpose electronic computers.

The first group of special purpose devices *select* from a collection of *separable* records (punched hole, optical film, and magnetic cards). In each of these devices, the file items are passed through a sensing device which analyzes the code stored on each card. The search criteria are established either by wiring patch boards, inserting special request cards, or by other means. The selected

cards may then be used to prepare printed lists, to control the selection of fuller documentary records or, in the case of film, to project and photocopy the original information.

Another electromechanical device in current use employs continuous reels of punched paper tape as input and still another uses punched card input and paper tape for intermediate storage. In the former case, the search requirements are established by patchboard wiring and in the latter, are established by machine option switches. In both devices, the successfully retrieved items are typed out by an attached, machine-controlled typewriter.

From the experience thus far gathered in the operation of these machines a number of generalizations may be made.

- 1) While there is a wide gamut of card-processing speeds, their general performance has been quite satisfactory. All operate on the site of the information processing activities rather than in a computation center.
- 2) The chief limitations have been in treating, a) problems where quantification is important (range of boiling points, per cent of each of several ingredients required), b) cases involving the conjunction of a number of disjunctive classes (e.g., a card is to be scanned to determine whether it contains both the code for three American states and the code for no French province), and c) problems involving detailed interrelationships such as sequence, connectivity, subject-object relationship. These limitations apply to some degree to each of the electromechanical devices but in some cases these limitations have been surmounted in ingenious fashion.
- 3) The cost of these machines, with the exception of the more powerful optical film and magnetic card devices, has been remarkably low compared to the high-speed electronic machines and their full-time use for information retrieval has been practical. The cost per search, as well as the capital investment, has been low.
- 4) The rate of search has been limited by card and paper tape handling speed. Again, excluding the optical and magnetic, the systems have ranged from 8–500 items per minute.
- 5) The separable unit record has proved to have a number of advantages and disadvantages. Among the former are the ability to select small subfiles for machine feed, the ease of file maintenance, and the easier access (in many cases) to original information (through film inserts, card drawings, etc.). The disadvantages center around the variability in information content of documents which cause either inefficient waste space on punched cards or introduce the difficulties of multiple card evaluation and manipulation. It has of course been possible to simulate the behavior of simple relay and medium electronic computers.

- 6) Experience with optical film and magnetic cards has been very limited. The specifications and demonstrations have been most impressive.

SPECIAL PURPOSE INFORMATION RETRIEVAL ELECTRONIC COMPUTERS

We are now entering the era of the special purpose high-speed electronic information retrieval computer. Only one or two published accounts of such machines have appeared, since they are still developmental and frequently related to military applications.

Such devices differ little from some currently announced data processing equipment and it appears that their usage will be justified only in the largest information processing establishments. These machines will look very much like a general purpose computer but trade unnecessary features (*e.g.*, floating point arithmetic) for hardware embodiments more needed for information retrieval (*e.g.*, a large number of independently operating comparison registers).

Another "special" unit is a special limited version of a standard data processor with certain features eliminated. The belief that this will materially reduce the cost of such a machine is *not* in accord with the experience of computing machine manufacturers.

Special computers are also required to handle special problems in information retrieval. One that is receiving a good deal of attention is the problem of the storage and retrieval of graphical, geometrical, and topological configurations. One special computer has already been designed for work in this area. Special military information retrieval problems have given rise to the design of special machines geared to these problems.

The following is a description of a nonexistent but typical large, special purpose computer. The external storage might consist of a large bank of high-density, high-speed magnetic tape units which would have the facility (under computer control) to advance or back up rapidly and then to search not for one "key" but for many logical combinations of keys. All tape units would be active simultaneously under independent control of a special tape manipulation and testing unit. As information which satisfies the rough screening conditions is brought from the tapes, it would be sent into a magnetic core unit where a further refining process is carried out. Information which successfully meets all the criteria for one of the many simultaneous searches to be conducted would be sent to the proper output area for both immediate visual display and for subsequent complete printing of full information on high-speed equipment.

The hypothetical computer just described represents only one direction in the development required. We assume that any good information retrieval system is sufficiently flexible to meet the logical requirements of the searching system and to search at sufficiently low unit cost. Beyond these, the two most critical factors are investment cost and the size of the document col-

lection that can be manipulated in some reasonable time. The typical relay card-selecting device represents relatively low investment but with correspondingly small collections manipulable in practical times. The typical search program running on available electronic computers represents a high investment cost with moderate sized collections searchable in practical times. The special purpose electronic computer, such as the hypothetical one described above, represents an attempt to obtain increased performance at an increased investment cost. Thus, we see that no major breakthroughs have occurred to date.

The challenge to the computer designer and to the system designer is the development of techniques for handling large collections on relatively inexpensive devices. It is hoped that the developments in computer components, computer logic, storage devices, and systems operation will lead to the development of improved devices. Hand-in-hand with such development must be the maturing of our understanding of the theoretical and practical bases for the retrieval of information.

BIBLIOGRAPHY

- [1] P. Bagley, "Electronic digital machines for high-speed information searching," Master's thesis, M.I.T., Cambridge, Mass.; 1951.
- [2] R. H. Bracken and H. A. Tillitt, "Information searching with a 701 calculator," *J. Assoc. Comp. Mach.*, vol. 4, p. 131; 1957.
- [3] S. R. Moyer, "Automatic search of library documents," *Computers and Automation*, vol. 6, p. 24; May, 1957.
- [4] J. J. O'Connor, Univac Applications Research Center Tech. Rept. No. 18; 1957.
- [5] A. Opler and N. Baird, *Proc. Internatl. Conf. on Sci. Information, Area IV*, p. 37; 1958.
- [6] B. K. Dennis, "Rapid retrieval of information," *Computers and Automation*, vol. 8, p. 8; October, 1958.
- [7] W. H. T. Davison and M. Gordon, "Sorting for chemical groups using Gordon-Kendall-Davison ciphers," *Amer. Documentation*, vol. 8, p. 202; 1957.
- [8] C. Mooers, Zator Co., Boston, Mass., Tech. Bull. No. 59; 1951.
- [9] C. Mooers, Zator Co., Boston, Mass., Tech. Bull. No. 64; 1951.
- [10] A. Opler and T. R. Norton, "A Manual for Programming Computers . . .," Dow Chemical Co., Midland, Mich.; 1956.
- [11] T. R. Norton and A. Opler, "A Manual for Coding Organic Compounds . . .," Dow Chemical Co., Midland, Mich.; 1956.
- [12] A. Opler, "A topological application of computing machines," *Proc. WJCC*, pp. 86-88; 1956.
- [13] A. Opler and T. R. Norton, *Chem. and Engrg. News*, vol. 34, p. 2812; 1956.
- [14] A. Opler, *Chem. and Engr. News*, vol. 35, p. 92; 1957.
- [15] A. Opler and N. Baird, paper presented before American Chemical Society, Div. of Chemical Literature; April, 1958.
- [16] R. A. Carpenter, *et al.*, "Correlation of structure and physical properties with utility of chemical compounds," paper presented before American Chemical Society, Div. of Chemical Literature; April, 1958.
- [17] W. H. Waldo and M. DeBacker, "Printing chemical structures electronically; encoded compounds searched generically with IBM 702," *Proc. Internatl. Conf. on Sci. Information, Area IV*, p. 49; 1958.
- [18] L. C. Ray and R. A. Kirsch, "Finding chemical records by digital computers," *Science*, vol. 126, p. 814; 1957.
- [19] H. R. Koller, E. Marden, and H. Pfeffer, "The HAYSTAQ system: past, present, and future," *Proc. Internatl. Conf. on Sci. Information, Area V*, p. 317; 1958.
- [20] J. J. Nolan, paper presented before American Chemical Society, Div. of Chemical Literature; April, 1958.
- [21] J. W. Perry and A. Kent, "Tools for Machine Literature Searching," Interscience Publishers, New York, N. Y.; 1958.
- [22] *Ibid.*, ch. 19.
- [23] *Digital Computing Newsletter*, vol. 10, no. 4, p. 4; October, 1958.
- [24] *Patent Office Res. and Dev. Repts.*, No. 13; November, 1958.

A Specialized Library Index Search Computer

B. KESSEL† AND A. DELUCIA‡

INTRODUCTION

THE NEED for mechanizing library information searches has become apparent during the past two decades. The phenomenal increase in the size and number of library establishments in conjunction with the requirement for greater speed in servicing information requests have been key contributory factors in focusing attention on this situation. A large variety of developmental and commercially available devices have been used for putting library card catalogs into forms more amenable to automatic searching. It was only a matter of time before many researchers became aware of the advantages of digital computers in mechanizing this operation. Government and commercial organizations, by writing new programs, were able to adapt those computers which were already available to them.

Early in 1958, a program was initiated by the Rome Air Development Center for the design and fabrication of an Index Searcher that was to be tailored specifically to the needs of library documentation. The Index Searcher was to contain only those logical functions that would be of value in library information searching. The Searcher was to be developed primarily as a research vehicle for use in studying various index and information retrieval approaches. The basic design was to include facilities to allow the Searcher to be used as a fully operational device. In April, 1958 a contract was awarded to the Computer Control Company with delivery of the Index Searcher to occur early in 1959.

Design concepts of the Index Searcher have evolved in part from a knowledge of the Minicard Selector.¹ However, the Index Searcher will be used in library situations where index data and graphics material are stored separately, rather than together as on Minicards.

The Index Searcher will search through large volumes of index data serially and print out the identification of those documents, reports, or graphic materials that satisfy the requirements of the search criteria. A consideration of the problems of library mechanization indicated that the following features should be included in the Searcher design: 1) high-speed searching of the index data; 2) the ability to reproduce all or part of the index library cheaply and quickly; 3) a minimum of delay in effecting the search beyond the setting up of the search criteria; 4) the capability for handling a wide variety of index and classification schemes; 5) ease of operation; 6) flexibility in permitting frequent updating of

the file; 7) search for more than one question at a time; and 8) a growth potential allowing for relatively efficient use of the Searcher either singly or in groups as the size of the library increased. The system design of the Searcher as a special purpose system, was to result in an information handling capability that could be matched in the general purpose computer field only by a considerably larger and more expensive machine.

GENERAL DESCRIPTION

The Index Searcher uses magnetic tape as the storage medium for index data. An index entry is made on the tape for each document, report, or other piece of physical material which is to be made available for rapid automatic searching. An entry might consist of anything from a title to a complete text, depending upon the storage and recovery system to be used. In typical applications the entry made for a given document will consist of a document number, title, author, date, and several descriptors that define the subject matter of the document.

The descriptors can be grouped into sets that are designated as phrases. The value of this feature can be illustrated by considering the indexing of a technical paper that describes a machine using transistorized logical circuits and a magnetic core shift register storage. Listing of just the four descriptors for transistorized, logic, magnetic cores, and shift registers could result in the false selection of this document during a search for magnetic core logical circuits. Use of phrase boundaries in the proper places assures that the descriptors will be properly associated with each other during the searches.

The standard machine word for the Searcher is 42 bits in length. This consists of seven alpha numeric characters of 6 bits each. The system also includes provision for handling double and triple length words so that it can accommodate clear text as well as coded index data.

A typical document index entry might consist of twenty machine words, making a total of 840 bits of information.

Search criteria are specified in terms of question words plus logical connectives to group the question words into question phrases and to group the question phrases into complete questions. The question words are stored in the internal memory of the Searcher. The memory has a capacity of 20 machine words.

The type of comparisons to be made between question words and the document index entry words is specified individually for each word by plugboard wiring. The specification can be for "equality," "less-than," "greater-than," or any combination of two of these types of comparison. For example, a question might specify that

† Computer Control Co., Inc., Framingham, Mass.

‡ Rome Air Dev. Center, Griffiss AFB, Rome, N. Y.

¹ J. W. Kuipers, A. W. Tyler, and W. L. Myers, "A Minicard system for documentary information," *American Documentation*, vol. 8, pp. 246-268; 1957.

a document is required on a particular subject that would be identified by descriptor equality comparisons; published after 1956, identified by a "greater-than" comparison; and with a security classification no higher than confidential, identified by an "equal-to or less-than" comparison.

The question words are grouped into question phrases by means of plugboard-connected logical circuits. Fifteen phrase elements are available for composing up to 15 different question phrases. Two or three phrase elements can be cascaded to make a larger phrase than can be handled in one logical element. Each phrase can use any desired combination of question words, in either assertion or negation form, as inputs, and each question word can be used in as many of the 15 different phrases as desired.

Complete search criteria questions are made up of question phrases by plugboard-connected select logical elements. The same flexibility exists here as in combining words into phrases. Ten question elements are provided, resulting in the ability of the Searcher to simultaneously search for documents meeting ten different search criteria.

Searching consists of scanning through the complete document index tape and comparing the contents of each index entry with the question words and logic stored in the Searcher memory and plugboard connections.

The result of a successful search is a print-out of the document numbers of those document index entries that have met the search criteria. An identifying number printed beside each document number shows which of the several search questions is answered by that document. A picture of the Index Searcher is in Fig. 1.

FUNCTIONS

The Index Searcher has six different modes of operation. Listed in the order in which they would be used, these are 1) Document Insert, 2) Regenerate Tape, 3) Question Insert, 4) Search, 5) Print, and 6) Edit. Each of these will be described after a brief reference to the system block diagram shown in Fig. 2.

The major blocks making up the Searcher, and their functions, are as follows:

- A) Magnetic Tape Unit—stores and scans document index data.
- B) Tape Buffer—serves as a buffer to and from the Magnetic Tape Unit and Flexowriter.
- C) Flexowriter—serves as a punched-paper-tape reader for document and question insertion, and as output printer for searching.
- D) Word Input Buffer—accumulates magnetic tape information frames to form complete machine words.
- E) Print-Out Buffers—store tape data which is to be printed out if the document is a desired one.
- F) Word Storage Buffer—stores complete magnetic tape index words for comparison with question words.

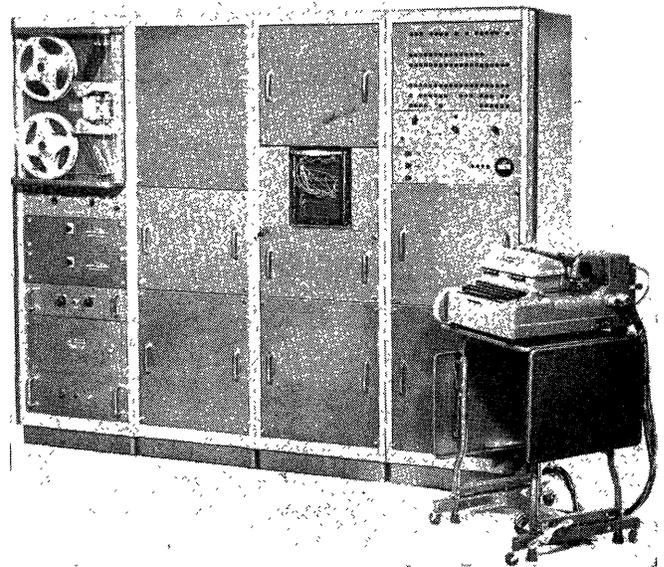


Fig. 1—Index searcher.

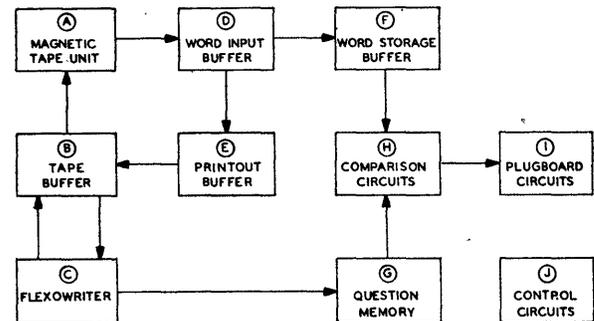


Fig. 2—Block diagram.

- G) Memory—stores question words.
- H) Comparison Circuits—compare tape index words with question words from memory.
- I) Plugboard Circuits—use results of word comparisons to make complete question comparisons.
- J) Control Circuits.

The actions of the Searcher in each of its operating modes follow.

Document Insert

This mode is used to place new document index entries into the Searcher's magnetic tape storage. Primarily it is a punched paper tape to magnetic tape conversion process. New index entries are submitted to the machine in the form of punched paper tapes. Paper tape frames are accumulated in the tape buffer until the buffer is full. Those contents of the buffer which represent complete index entries are transferred to the magnetic tape as one block of tape data. Any partial entry left in the buffer is then completed with the next paper tape information to arrive, and is followed by more documents. The process continues automatically to the end of the paper tape. Each magnetic tape block contains an integral number of complete index entries. The actual lengths of blocks on the tape are variable.

Regenerate Tape

This is a simple magnetic-tape to magnetic-tape routine, which allows the file tape to be duplicated as insurance against loss of file data through accidental damage to the tape on the Searcher. This requires an additional magnetic tape unit that is not part of the Searcher as originally built, although space has been left for it in the racks.

Question Insert

This mode transfers question words from punched paper tape to the Searcher Memory. This is accompanied by insertion of a question plugboard that specifies the nature of the comparison to be made for each word and the combination of the question words into phrases and complete questions. The plugboard connections can also specify one or two words per selected document to be printed out in addition to the document number. A simplified symbolic representation of the plugboard and its circuits is shown in Fig. 3.

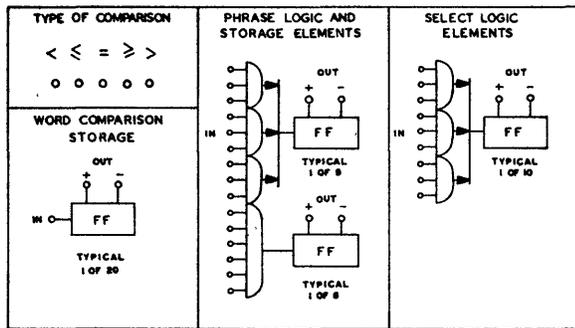


Fig. 3—Simplified representation of plugboard circuits.

Search

This is the primary operating mode of the Searcher. This mode performs the scanning of stored document index entries in search of those that meet specified question criteria. The tape can move in either the forward or reverse direction to accomplish this search. While the tape is scanned, tape frames are accumulated into complete machine words, and are compared with the twenty stored question words. A plugboard word storage element remembers a successful comparison with any of these words until an end-of-phrase designation occurs in the tape data. At that time the word storage outputs are sensed in the phrase element logic to determine whether any complete question phrase criteria have been satisfied. If so, a phrase storage element remembers this fact as scanning continues through the remainder of the tape index entry. When the end of tape data for the document is reached, sensing the outputs of the phrase storage elements determines whether or not a complete question criteria has been satisfied. During the scanning of the document entry, the Print-Out Buffer receives automatically the document number and two other plugboard-specified words. When a document answers a question, the Tape Buffer receives the contents of the Print-Out Buffer and a number identify-

ing the question answered by the document. This information is printed out on the Flexowriter. Further searching ordinarily continues during print-out. However, if a series of successive selections results in filling the Buffer faster than the maximum print-out rate, the tape automatically stops until adequate Buffer capacity is available for further document data. When the end of the recorded portion of the tape is reached, the tape unit automatically stops and positions itself ready for the next search in the opposite direction.

Print

This mode is used to print out entire document index entries rather than just the three words possible in a normal search. The machine operates in much the same manner as for normal searching until a selection is made. The tape must move in the forward direction. Selection of a tape index entry causes the tape to stop, reverse, reread the entire selected block into the Buffer, print out the complete selected entry, and then resume search.

Edit

This mode is used to delete unwanted index entries from the tape. Document entries to be deleted are specified by document number or other normal question criteria. Operation is similar to the Print mode up to the point of bringing into the Buffer the block of tape information containing the document to be deleted.

At this point the block is recorded in the same place it formerly occupied on the tape, but with blank characters in the position which had been occupied by the deleted entry.

PARAMETERS

The Searcher scans through magnetic tape document index entries at an effective rate of 218,000 bits per second, or about 5200 machine words per second, where each machine word contains 7 alpha numeric characters. For the typical document index entry length of 20 words mentioned earlier this amounts to 260 documents per second. While searching at this rate the machine seeks documents satisfying up to ten independent search criteria.

The Searcher uses 2400-foot rolls of one-inch magnetic tape for document index storage. One reel stores 57,600,000 bits, which is about 68,500 twenty-word documents. Uninterrupted search time for a complete reel is approximately 4.5 minutes.

The input-output rates of the Searcher are presently limited to the 10-character-per-second rate of the Flexowriter for document insertion, question insertion, and selection print-out.

Therefore, the document insertion rate, based on 20-word entries, is about 4 documents per minute, and the selection print-out rate, when printing out 3 words per selection, is approximately 25 documents per minute. A 20-word question insertion takes about 15 seconds. These rates can be substantially increased by use of high-speed paper tape reader units and high-speed printer or punch outputs.

Programmed Interpretation of Text as a Basis for Information-Retrieval Systems

L. DOYLE†

TWO conditions have made it almost inevitable that we have an information-retrieval project at the System Development Corporation (SDC).

First, our internal documentation. It has been estimated that we acquire 10,000 documents a year both from internal and external sources, not including books and periodicals. Internal distribution of these documents runs into millions of copies. We have not been able to afford to abstract and subject categorize more than 10 per cent of our 10,000 documents a year. With a good retrieval system we might be able to make many thousands more of our documents accessible by subject without increasing documentation expense.

Secondly, one of SDC's major activities is computer programming for air defense, and as a result of this we have charge of several large general-purpose digital computers.

Thus, we have both the motives and the equipment to study the computerization of the handling of documented information. And so, about six months ago a three-man project was created at SDC to do research and development in this area.

PROGRAMMED SELECTION OF DESCRIPTORS FROM TEXT

Previously we had considered using, as stop-gap solutions, some of the already existing well-known information-retrieval formulas, such as Uniterm, or marginal punched cards, or peek-a-boo, or something of this nature; but we thought, "Where are we going to get the people to read and categorize 10,000 technical documents a year?" We realized that we were already in a situation which was going to become increasingly common as time goes on—a setting where skilled man-hours are harder to find than time on a large computer.

At this time we switched to a machine-centered philosophy and began to explore the possibilities for performing the chores of documentation on digital computers and EAM equipment. The crucial difficulty in implementing such a philosophy is finding a way to use computer programs to interpret natural English text for the purpose of effectively subject-indexing documents, so that they can be retrieved precisely without any person's having read them for purposes of categorizing, picking descriptors, or encoding in any manner.

MECHANICAL INDEXING AS A PRELUDE

Our thinking about the interpretation and retrieval of natural text has changed greatly since we started.

For example, one of our realizations has been that it is premature to set oneself up for pure machine searching of natural text. Machine searching is superb if you know exactly how to describe what you are looking for and if you are sure that you know how to choose from among many possible searching strategies. I doubt if anyone is yet in this comfortable position with respect to machine searching of text. What is needed is a searching setup which is fast and convenient, while at the same time allowing the human mind itself, with its versatility and its powers of observation, to take part in the search. Our solution has been to employ mechanical indexing using an artificial language, which I shall describe later.

THE SYSTEM

We now have an experimental abstract searching system at SDC which was set into motion by our retrieval project. Through the use of it, we hope to find out the basic difficulties of natural text retrieval and to come up with new principles of language data processing, some of which may be useful for purposes other than information retrieval. It is also a research tool, through which properties of language and of collections of information may be subjected to analysis—for example, by making frequency counts. Later I shall describe an instance of the use of the system for research.

Fig. 1 shows the process we now use to encode abstracts. At the left is keypunching of the text, which in itself tends to undermine the purpose of natural text retrieval—however, we assume that input technology will develop to the point that keypunching will no longer be a barrier. For the present, key-punching limits our input to small chunks of text. We now work only with abstracts; however, we can handle any fragment of text of about abstract size (approximately 100 words).

As the text material arrives in computer storage (as 6-bit Hollerith), the text-compiler program translates the raw text into a more condensed form suitable for searching. There are two stages of this condensation.

First, selection of subject terms from the text. The text compiler has at its disposal a table of allowable subject words, and it searches text for these words. Secondly, all of the subject words or terms which the text compiler finds in an abstract are replaced by binary numbers which have been predesignated to represent the subject words when they are stored on tape. These binary number tags take up less than a third as much space in storage as would the subject words themselves.

† System Development Corp., Santa Monica, Calif.

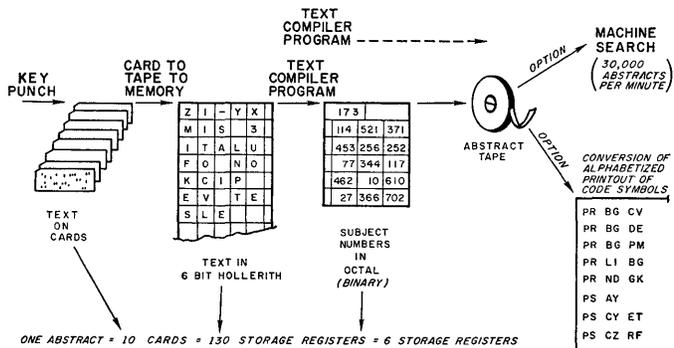


Fig. 1—Conversion of text to condensed code for searching.

By these means the text compiler will condense an abstract from about 130 storage registers of text down to about 5 or 6 registers of binary numbers. With this condensation one can represent 10,000 abstracts on about 5 per cent of the total length of a standard IBM tape reel. We now have a tape layout which contains the boiled-down essential information of many abstracts, and any time we wish to make reference to these abstracts, they can be read into computer memory at the rate of 500 per second. The text compiler can process 4 abstracts per second.

At this point we have our choice of two courses of action. We can either machine search the tape for particular combinations of subject words, or we can convert the entire tape to an alphabetical code, which can be printed out as an alphabetized format and searched by eye.

MACHINE SEARCHING

If one does decide to machine search this tape, it can be done in a very short time. With a large digital computer, such as IBM 709 or the AN/FSQ-7, the tape-stored abstract plots can be searched much faster than they can be read in. Since both these computers can execute instructions while in-out operations are going on, it is possible to conduct 100 simultaneous logical product searches during the 20 seconds it would take to read in 10,000 abstracts.

THE HUMAN MIND AS A SEARCHING INSTRUMENT

However, at this point in our development we value searching flexibility much more than we value speed or efficiency. Speaking of speed, it is still difficult to duplicate mechanically the feat of a person looking up a number in a telephone book. Let's consider this point. The Los Angeles Central Zone Directory has enough alphanumeric content to fill six IBM tape reels. But almost anyone is capable of finding an entry in this directory in from 10 to 30 seconds. This illustrates the power of familiarity with alphabetical order, which is a typical (and therefore unappreciated) human ability. If you have about seven RAMAC's you might just be able to exceed this common everyday performance by humans.

If you now imagine a whole room filled with telephone-directory-sized books, each of which contains en-

tries confined to a small portion of the alphabet, it is easy to see that it would take some person less than twice as long to find an entry in these many books as it would take to find something in just one book. Of course, for this to be possible, all the books have to be stacked in alphabetical order. We now have a very cheap searching mechanism, which no computer can equal in speed for such a large volume of material. There is, of course, the cost of publishing all these books to be considered, but when such things as microfilm scanners are available, the possibilities for exploiting human familiarity with alphabetical order are something to be seriously considered.

Another human skill of importance in searching is the ability to judge the meaning of a symbol by its context. It will take a great amount of programming and even more preliminary research before this skill can be challenged by electronic instruments.

From the standpoint of research and development, the most important thing about humans as searching instruments is that they are observant. They repeatedly notice things that they are not programmed to notice. At this stage, our progress strongly depends on the application of these powers of observation. Our intent is to use astute people to search small document collections in order to find out how computers should be programmed to search large collections. Our hope is that the state of the art of programming to produce highly condensed, information-rich formats for search by the human eye will improve at such a rate that pure machine searching of natural text may not for many years catch up in convenience or effectiveness with computer-assisted eye searching.

Our present alternative to machine searching, which is illustrated in Fig. 1, is the generation of an alphabetized printout consisting of two letter code words for subjects. So, instead of using the binary subject number tape as input to a searching program, we feed it to another program which converts all the binary numbers to two letter symbols, which we call bigrams. The conversion process is totally analogous to binary-to-decimal conversion, the only difference being that instead of subtracting powers of ten, we subtract powers of 26, the number of letters in the alphabet. These bigrams are punched out on regular IBM cards, after which EAM equipment can offset reproduce and alphabetically sort the cards prior to the printing out of a format.

Why do we use bigrams for our alphabetized printout instead of octal numbers, decimal numbers, or for that matter the original subject words themselves? One answer is that we get greater condensation. We can store all the subject words in one abstract on one IBM card. And we are now in a position to use EAM reproducing and sorting equipment to alphabetize on every subject word in every abstract. Fig. 2 shows how the reproducing is done on the contents of one abstract. One can see from this that if one has an abstract containing

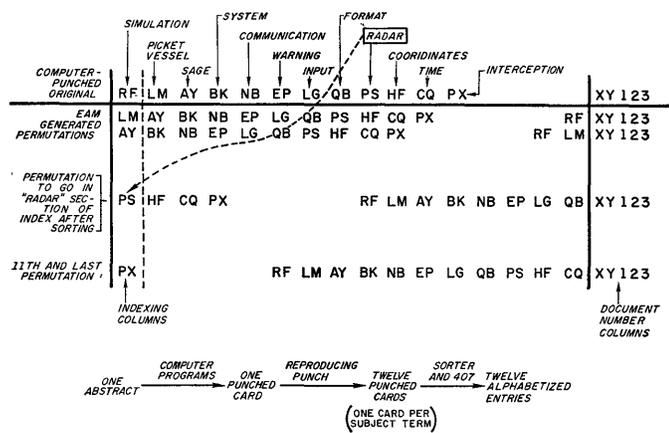


Fig. 2—Cyclic permutation of code symbols to produce index entries.

n subject words, then this EAM reproducing process will yield *n* cards, each beginning with a different subject word, and each constituting an entry in an index. Each entry carries, along with the indexing word itself, the remaining subject words in the abstract, which act as context for the indexing word.

Whenever I start explaining bigrams to people, their reactions convince me of the undying popularity of indexes which are in plain English. Nevertheless, I maintain that no spoken or written language was designed with alphabetized indexes in mind, and am convinced that special indexing languages, understood and used by librarians and other specially trained people, will play an important part in information retrieval as soon as the programs are available to translate from English to these languages.

A BETTER ARTIFICIAL LANGUAGE

Our present system at SDC is an adaptation from something that started out to be a pure machine searching system, and therefore it has some of the vestigial organs and imperfections of an evolved creature. One of these imperfections is that the bigrams are arbitrarily assigned. As long as we feel the necessity to use an artificial language, we might as well try to derive one which is tailor-made for indexing. What properties should such a language have?

- 1) All words should be short and uniform in length. This is the only property of the four I am going to discuss which our present bigram code actually has. Shortness and uniformity are desirable not only because of convenience in manipulating the language on EAM equipment, but also because more information is brought onto one page, which aids eye scanning.
- 2) No synonyms. The importance of not having synonyms is that it makes it possible for most searches to be satisfied by proceeding to one page or region of the index. Skipping around will be necessary only when one wishes to follow association trails to related topics.
- 3) Relationship between meaning and alphabetical order. This, I think, would be one of the handiest fea-

tures of any artificial language especially designed for indexing. It will greatly simplify the searching, it will make the artificial language easier to learn, and it will allow generic searching, the lack of which is probably one of the major disadvantages of natural text retrieval based too closely on the natural language itself.

4) Quantitative dependence of assignment of English meanings on the contents of the library. A document is retrieved *from* a library, but even so it is not obvious that the contents of the library should be as important a factor as the contents of the document itself in affecting the way the document is encoded as a search item. In our artificial language, the nature of its correspondence to English words ought to be governed by how much of *what* is in the library. Application of this principle is difficult because it must certainly involve a means of frequency-counting everything in the library.

But the benefits are many. One possible benefit is that the state will be approached where words of the artificial language will be equally used, which of course is an information theory ideal. It is important to apply this ideal if one is to get predictable effectiveness from a searching system. The equal use of artificial words tends to lead to equal retrieval precision for all documents and also has the effect of insuring that all parts of an index will be equally used. I have seen several automatically generated word indexes, and they all suffer from the presence of very large blocks of entries beginning with the same word. Their heterogeneous structure causes about half of the space to be taken up with blocks which are seldom used because they are so large. There are a great many arguments in favor of isotropic indexes, where entry blocks are similar in size regardless of the subject.

LIBRARY ANALYSIS

One of the general aims of our project is to develop means of analysis of very large collections of verbal information. We think of a collection of documents as having an inherent structure, which is not affected by physical rearrangement of the documents, a structure which, incidentally, we are fully able to probe only with the aid of fast computers. Fig. 3 shows how significant the structure of a document collection can be.

One question in my mind has been: Is it possible for a computer program to determine what is a subject word without having available any subject-word table or dictionary prepared by some human? In other words, do subject words have distribution characteristics within a library that a computer program can detect, thereby permitting distinction from nonsubject words? The data in this figure indicate that the answer is very probably "yes."

The dictionary which we now use as input to our text-compiler program contains no nonsubject words, but it does contain some borderline cases like the word "time," which many people would regard as too general a concept to be a good subject word. "ADC" on the

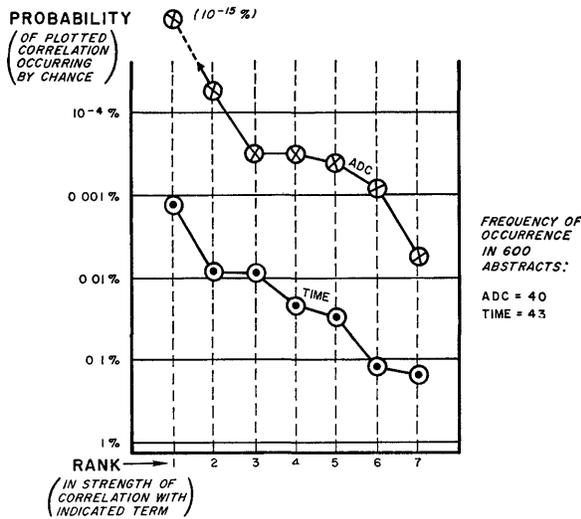


Fig. 3—Probability of highest observed correlations as a test for subject words.

other hand is a good solid subject word, standing for "Air Defense Command." Our alphabetized index enables us easily to select the words which correlate most highly with these two topics, and we can calculate and compare probabilities.

One can reason that a good subject word should have certain other words which co-exist with it in the same documents with a frequency much greater than expectable from chance distribution, but that nonsubject words, which are likely to be used by anybody writing about any subject, should not have high correlations.

To test out this notion I calculated the probabilities for the highest observed correlations occurring by

chance, both for "ADC" and for "time," given that all words in the library are randomly assigned to documents. Of course, the words are not randomly assigned, so I got some very low probabilities. As Fig. 3 shows, the ADC correlations are very much more improbable than those for "time."

Now these correlations will become weaker for any word, subject word or otherwise, as they are present in fewer documents. This means that in order to apply any sort of correlation test to select subject words, one has to make allowance for the frequency of the word. Unfortunately, the correlation test will fail altogether when a word is present in only 3 or 4 documents. One has to have a large enough sample. Also author biases in use of common words could conceivably cause many non-subject words to pass a correlation test. But, fortunately, as collections increase in size these effects should become less important, and it is in very large collections where this sort of methodology will be needed.

CONCLUSION

As a final comment: libraries and other collections of written information can be thought of as realms of nature, subject to scientific observation. Science brings valid simplicity to that which is apparently complicated, and it is hard to find anything more complicated than masses of ideas recorded on paper. And so I make explicit an idea which I hope has been implicit in this presentation—that general-purpose computers of today give us the opportunity to apply scientific method to uncover the principles of the nature and use of information in order that we may put to better use the vastly more powerful computers of tomorrow.

A Theory of Information Retrieval

CLINTON M. WALKER†

THE mathematical formula which best describes my conclusions from reading the literature on information retrieval (IR) is the following:

$$4 \text{ U OK } 4 \$ = \text{ET}$$

For you, better for dollar equality.

This states that a more economical approach for organizations interested in information retrieval might be collectively to support as an information retrieval center some nonprofit organization, such as SRI or SDC. Such

an organization could be a center for receiving and dissemination of up-to-the-minute retrieval literature of organizations concerned; could advise on the practicality of certain undertakings; and could perform experiments in the field of IR.

Aside from this one equation, formulation should proceed from basic principles. Perhaps the most basic of all principles is that meaning, rather than information alone, needs to be retrieved. Just how does one produce or obtain meaning? Take the example of a small child. All a child knows at first is himself. He gets acquainted with his hands and feet, and then with his near associates by relating them to himself. He gradu-

† Hughes Aircraft Co., Culver City, Calif.

ally learns to classify things in terms of roundness, which things he might call a ball; in terms of use, such as food. New things learned are related to things already known. For example, at an early age, any man might be classified as "daddy." Throughout his life, meaning is obtained by relating what is familiar to that which is unfamiliar.

We might christen this process the "Mew-Mew" theory of meaning. Each of us, as a "me" looks at something else as a "you," which we interpret in terms of the "me" or what is known, but which we might also take back into the relative "you" for objective evaluation.

This process of classification is an operational way to produce meaning. A language, in effect, classifies nouns; descriptions of "which," "what kind of," and "how many" have meaning when related to other nouns. What the nouns do—and how, when, and where they do it—has meaning when related to what other nouns might be doing.

So, an operational language is one in which classification takes place in familiar areas or domains. In these domains, dictionaries can be constructed of key nouns; definitions can include relationships to other key nouns within the area. For retrieval purposes, reference to a key noun could have a built-in potential reference to other key nouns, thus providing a built-in meaning potential. The prospects are exciting. But, before we develop the idea further, let us lay down some basic postulates.

We can set up a number series as a set R of objects called nouns, with the relationships defined by three operations denoted by Σ , Π , and f . Concomitant with this set is another set M whose members can be derived from certain operations on the set R . The symbol \rightarrow means "results in a relationship of," or "implies that"; the symbol "+" means "and"; the symbol "-" means "not"; "()" are used in the usual enclosure sense. An IR specific operation is one denoted by the symbols Σ , Π , or f . An IR nonspecific operation is any other operation in real or complex variable theory. We will assume that IR nonspecific operations will follow the manipulative rules of real and complex numbers for IR specific operations. For example, the operations are additively commutative.

$$\begin{aligned}(A_j B) + (C_j D) &= (C_f D) + (A_f B) \\ (A_\Sigma B) + (C_\Sigma D) &= (C_\Sigma D) + (A_\Sigma B) \\ (A_\Pi B) + (C_\Pi D) &= (C_\Pi D) + (A_\Pi B).\end{aligned}$$

The operations within the parentheses are IR specific; those between the parentheses are IR nonspecific. Additional postulates are required for defining the operation processes in an uncompleted operation. The following postulates and definitions are offered for consideration.

DEFINITION

The domain of A consists of all subcategories and subsequent subcategories under A .

$A_f B$ states that a word, A , which is classified in category B is put in a relationship such that A is in a hierarchy less than that of B , and that the domain of A includes not more than the domain of B . That is, A is part of B .

POSTULATE 1

$$A_f B_f C \rightarrow A_f C$$

states that a member of a subcategory is also a member of a category; for example, shoelace is a subcategory of shoe, which is a subcategory of clothing, which implies that shoelace is also a subcategory of clothing.

POSTULATE 2

$$A_f B \rightarrow B_f A$$

means that a category cannot be a member of a subcategory unless it is the only member. True, in ordinary language, sight can be thought of as a subcategory of sensing and perhaps sensing at the same time can be thought of as a subcategory of sight; but, for the convenience of constructing an unambiguous dictionary, we can exclude this possibility until such time as we find it absolutely required. Thus, we shall construct a dictionary in a domain with rigid hierarchical relationships among nouns. If later, we want to relax this requirement, we may find some interesting experiments available in the realm of "machine thought processes."

DEFINITION

$A_\Sigma B$ means that A is synonymous with B .

POSTULATE 3

$$(A_f C) + (A_\Sigma B) \rightarrow B_f C$$

means that synonyms within an area are necessarily members of the same category.

DEFINITION

$A_\Pi B$ means that A and B are related by means of the characteristics of some domain. We shall call these words "relatives."

POSTULATE 4

$$(A_\Pi B) + (A_f C) \rightarrow B_f C$$

means that relatives are subcategories of the same category.

DEFINITION

M is a set of elements of meaning derived by categorizing two or more elements of R at the same time.

POSTULATE 5

$$(B_\Sigma C) + A_f(B + C) \rightarrow (A_f B) + M = (A_f C) + M$$

means if B and C are synonymous, and A is a common subdivision of both of them, then the classification of A

into B and A into C simultaneously adds meaning to one of them, but it would be redundant to use both classifications.

POSTULATE 6

$$(A \Pi B) + (A + B) f C = (A f C) + (B f C) + M$$

means that when relatives A and B are, together, classified as members of C , they contain an element of meaning which is not present when they are separately so classified.

POSTULATE 7

$$(B \Pi C) + A f (B + C) \rightarrow A f B + A f C + M$$

means that to categorize a subdivision of two relatives, B and C , is to add meaning to both of them.

We have, by these postulated operations, created a language of classification—an operational linguistics which should be compatible with operational mathematics. Hopefully, a classification of nouns accessible by data processing equipment can relieve the information seeker of the trouble of searching the entire haystack for his needle of information and thread of meaning. Purposely sacrificed is the richness of redundant normal language in favor of the more important feature of exactness. Not only do we attempt to be more exact, but also to minimize ambiguity, to allow easy translation of concepts, to assure objective criteria of meaning, and to provide a basis of agreement in discrimination.

Postulate 1 tells us which words can be classified in a given domain. Postulate 2 prevents common words from being counted in esoteric categories, unless they are subsumed under those categories. Postulate 3 permits the counting of synonymous words in the same frequency tally. Postulate 4 permits the discovery of alternative paths for continued search. Postulate 5 permits singling out of the representative path among equivalent paths to be followed. Postulate 6 shows that two words are more significant if the context does classify them together. Postulate 7, finally, shows that paths which originally diverge become significant upon reconvergence. In all those postulates which have symbol M as an added element, significance is increased since M represents meaning and meaning is of prime importance in transference.

In any system of information retrieval, there are factors of cost, speed, and power. These three criteria can be used to determine, under a given circumstance,

which of several alternatives is to be preferred. In many instances, the major purpose of the retrieval system is to perform a rough scanning job for a literature searcher, to determine for him whether a particular document is worth further reading. Often the author can furnish, in addition to his name and topic, a list of his main ideas and purposes. He might even estimate a degree of correlation between the concepts embodied in his document and a list of key nouns in its general area.

To apply the power criterion, the machine or human doing the segregation of valuable from useless information can be simulated by a filter separating relevant message from total signal. Assuming homoscedasticity and linearity in the specified direction, a function

$$F = \Sigma(y - bx)^2$$

can be constructed, the parameter b minimized by least squares, and a Pearson correlation coefficient, r , obtained between simulated relevant message and simulated total message. The parameter, b , would represent the error term between, for example, time and amplitude. An autocorrelation can also be performed minimizing the error between message power and an amplitude-attenuation factor representing noise.

With power evaluated, we can set whatever boundaries we desire as to speed and cost and make our choice by linear programming.

An example of a low-cost, high-speed retrieval system with fair retrieval power is one based on the key nouns with which an author titles his document. Other key nouns are likely to be found in the same sentences as the title key nouns; therefore, the searcher, machine or human, can reduce the volume of the document to a desired degree of abstractness by selecting the frequency and location of the sentences containing these title nouns which he wishes to extract. A simple experiment was performed by the author, using as an abstract the first sentence containing a title noun in each major subdivision. Questions pertaining to the documents concerned were asked participants in the experiment, some of whom had read the author's abstract; some, the abstract of key words; and some, the entire document. Results of the scoring were roughly comparable for the three categories, for equal reading time. The experiment itself is not so important except as an illustration of the power of the use of key words. Properly categorized, the use of key nouns could become an effective means of speedy, powerful, and, in large volume, relatively inexpensive information retrieval.

The Role of USAF Research and Development in Information Retrieval and Machine Translation

ROBERT F. SAMSON†

INTRODUCTION

THE United States Air Force has numerous and varied types of data handling problems. This paper reviews some of the developmental approaches and contributions that the Air Force has made toward the solution of semantic-graphic information handling problems. Some of the interesting problems encountered in development of techniques and equipment in this field are presented.

BACKGROUND HISTORY OF ROME AIR DEVELOPMENT CENTER EFFORTS IN THE FIELD OF INFORMATION RETRIEVAL AND MECHANICAL TRANSLATION

In the past four years the Intelligence Laboratory of the Rome Air Development Center (RADC) has learned, through trying experiences, how to get the required movement in this data handling field. Understandably there are many approaches or philosophies, if you will, of how to develop the right synthesis of index, logic, hardware, etc., for any particular informational retrieval solution. The same can be said of mechanical translation (lexicon-logic and hardware). If we allow our minds to review these years and look at the situation as it was when we began our efforts, without today's vast knowledge of hindsight, I believe our approach would be quite similar to the one we took then. We would see the information retrieval problem growing at a staggering rate. The linguistic side was getting some attention, but the hardware, very little. The need and requirements for the Air Force were there and all that remained was to gather funds, select approaches, and secure contracts. I presume you recognize the humor of the previous sentence.

At that time the Air Force started to lend support to various projects already underway as well as to initiate entirely new work in this field. We knew the field needed much development effort, and involuntarily the Air Force took on the role of "catalyst" in information retrieval and mechanical translation developments. Note that I *am not saying* we were first with most; indeed not—we slipped into a "role" that was important to the Air Force and I believe it has done justice to the problem of both information retrieval and mechanical translation. You will note that I imply the existence of a common problem area in my use of the term "both information retrieval and mechanical translation." Indeed, with the exception of the problem of physically

handling documents and their contents, the Air Force Research and Development (R&D) program has been based on the premise that R&D effort in these two areas should be mutually cooperative. To illustrate this part before passing on: it gives a good return for effort expended because the two fields are interrelated, and advance in one usually means advance for the other. For example, if we were interested in information storage and retrieval alone, the Mechanical Translation (MT) field would be suffering for lack of a high-density storage that now seems quite practical. They "complement" one another from a development point of view, not only in hardware as mentioned but also, and perhaps more importantly, from the study of the rudiments of language.

SOME CONTRIBUTIONS BY RADC TO THE LARGE-SCALE INFORMATION RETRIEVAL PROBLEM

Several years ago RADC could not begin to say what type of development catalyst was needed. It could have been in the form of *heat* generated from "blowing off steam" about the "vast amount of data that must be handled" or it could have been in the form of a stimulating hardware development acting as a catalyst inserted into all the ingredients and by "stirring around to bring about enough agitation to get something done in the field." As mentioned in literature, the old cliché of bemoaning the fact that we are being overwhelmed with vast amounts of data and consequently develop only half-vast ideas, was not all correct, although I remember using the expression more than once. We accepted the approach of getting "something" underway and in so doing we became a doer in the field as well as the cause of the needed catalytic actions. From the start we realized we would have to accept the empirical approach; by this I mean a single superior approach was lacking. We accepted the empirical approach not in total ignorance, for we knew if one was to develop working tools, theoretical analysis alone would be of little help. In our search for new storage media in the field of information retrieval, we came across a high storage density, equipment technique which, when coupled with high read-out rates, could well be the answer to a *practical* and *economical* MT look-up or dictionary device. There was one storage medium known at that time that had possibilities of handling densities in the order of 10^6 bits per square inch; this was the work of King and Ridenour in the use of photographic emulsion on glass disks. The work that followed is now history—the disk photoscopic memory, handling 3×10^6 bits/square inch, was made feasible, providing us with an extremely valuable empir-

† Rome Air Dev. Center, Griffiss AFB, Rome, N. Y.

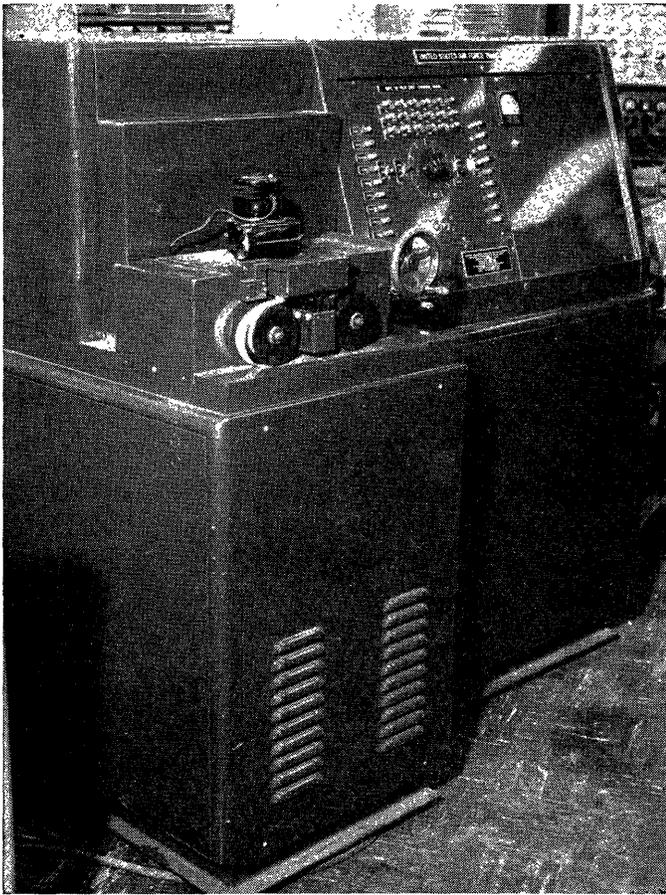


Fig. 1.

ical tool for further research in both information and retrieval *and* MT. This is an example to illustrate the point mentioned earlier, of getting better value in development investment when a development group has interrelated fields. While I am discussing the development of the photoscopic memory, I would also like to illustrate an interesting point. This is of particular interest because it illustrates a sometimes neglected point in developing an equipment that is dependent on a new technique. Referring to the photo disk memory, the equipment necessary to produce a disk was considerable, but of course necessary, if one was to get a high-density storage medium (see Figs. 1 and 2).

These two pieces of equipment by themselves do not represent all the necessary capability required to make a disk, but do show quite clearly the development involved in reaching a goal of practical and economical storage. The point here is that development in these two adjacent fields does not require only development of data handling equipment *per se*. It requires development of all those components that have anything to do with the creating of the media. Actually, the development breakthrough here in terms of what had to be done to produce the required density, was not the disk itself; although this is the end product, it was the precise components that allowed us to make this disk from the raw data on the tapes, thereby providing a facile method of

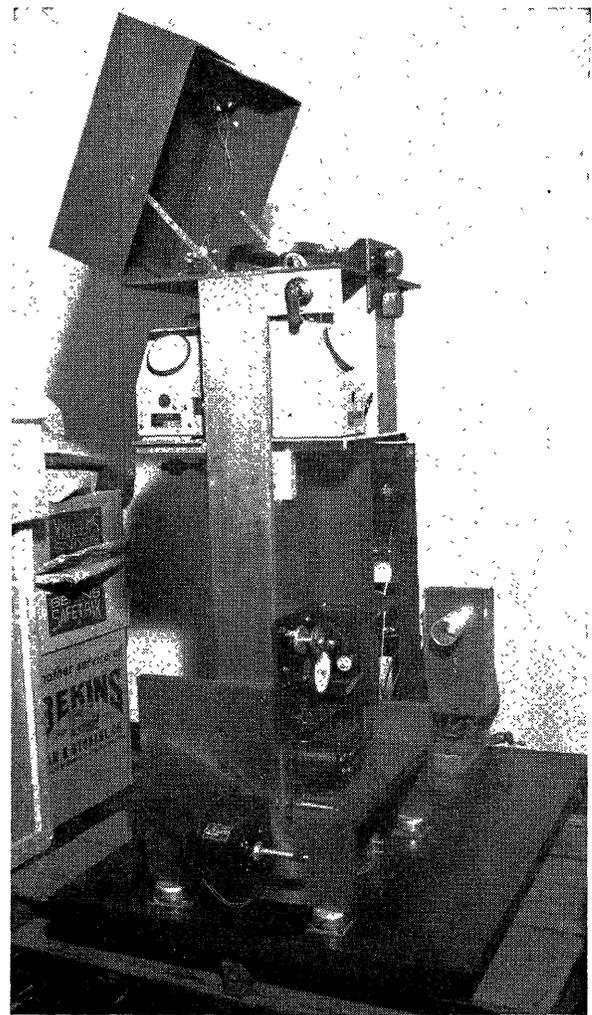


Fig. 2.

trying many types of stored data. This is not unusual in development programs of this type, but it is the unheralded side. In terms of engineering toil, it represents 70–75 per cent of the work and a substantial portion of the development dollars. Feasibility is a wonderful expression but a tricky term when it comes to development work. It was feasible to reduce a “bit” in terms of laboratory tests—the emulsion always had the resolution—putting the emulsion on optical flat glass had been done—reducing the bits to concentric tracks to disk to show feasibility for a digital store—all this then is “laboratory feasibility,” and the cost is quite insignificant when compared to the cost to reduce many millions of bits in a unit of time on a disk at the accuracy required. This had to be done precisely and accurately, and peripheral equipment had to be designed and constructed to make the photoscopic memory workable. The first set has been fabricated and improvements are now underway to perfect the disk-making equipment. The electronic logic used for reading in and out of the photoscopic memory comprise the “other half” of the development.

Now as to the empirical approach in information re-

trieval. Although it seems that a device would exist that allowed compact physical storage and efficient retrieval for large-scale document libraries, such was not the case several years ago. Today after some doing instead of speculating, several methods exist. I shall speak of three RADC equipment developments that cover conceptual voids in the field of storage and retrieval. These developments can be broken down into two general categories, both of which depend on environmental and operational conditions when selection is made.

Category 1—The separation of the index from the text. Defined, it is the index of the document removed from the document itself so that the index search is separate from the physical document. The physical document is retrieved by an identification number as a subsequent operation.

Category 2—The combination of the index with the text. The index will also produce the physical document when selection is made.

Under Category 1, at RADC we have the Magnacard Development and the Index Selector. Both these developments come under the heading of technical development, which means in reference to these particular equipments that we are developing "working" tools for experimental use at RADC. In the case of Magnacard, the storage medium is magnetic material deposited on segmented tape, 1×3-inch plastic cards. Engineers at RADC feel that Magnacard has excellent potential for files that require high-speed extraction of information and also where ease of updating and extensive file manipulation by categories is required.

The Document Data Index Set or the Index Search Computer for specialized library as reported in another paper at this conference by Ben Kessel of Computer Control Corporation, is an Index Searcher designed for library mechanization. It searches a large volume index data and prints out the identification of the document-graphic material, etc., that satisfies the search requirements. The Index Searcher uses continuous magnetic tape as the storage medium and the scan is serial in fashion.

Under Category 2, that is, index and document text stored together, we find the Minicard program. As one would suspect, this philosophy is based on usage with extremely large files; and, aside from its ability to perform random search, it of course reduces file space and bulk document handling problems considerably. We at RADC consider this as one of the outstanding examples in empirical development approaches, and state without reservation that this technical accomplishment is unsurpassed in the storage and retrieval field. Incidentally, this development has now reached a point where one can say, "It works." It is our sincere hope that large-scale empirical data will be obtained by its application that will give still further impetus to storage and retrieval development. Also at this point, it might be of interest to those in this field that achievements of this kind do not come easily, and I am sure designers

and engineers in this field realize fully that 4½ years is certainly a short period of time to develop an aggregate of ten complex equipments having many thousands of interrelated problems involving optics, emulsions, mechanisms, and electronics.

What does all this mean? It means we have mechanized library equipments that will simultaneously give improved operations and serve as tools by which we can experiment with various known library languages and in a relatively short time show the hidden problems in these index schemes themselves. It will also be quite natural to design the index around the logic and structure of the tool. We can prove the worth of indexes by constant evaluation while building a file.

I was asked to include in my paper all the work being done by RADC in the field of information retrieval and MT. In this respect I would like to mention that we are very much involved in the field of character recognition. This interest at first came about through the input problems associated with MT and subsequently considered for all input problems in data handling such as auto indexing, abstracting, etc. We have sponsored a development model which reads one English type font including numerals, both upper and lower case letters, space, and punctuation. We also are under way in developing a Cyrillic character reading machine which will give the MT field a tremendous boost in cutting down the transcription cost.

New York University has recently completed the first phase of a study for RADC on Russian printing matter. This study included such problems as the variety and frequency of Russian type faces and sizes in current use; the reflectance data of the printed type, the reflectance data of the Russian paper, the absorption and reflectance data on inks used in Russian printing, the predominant method of printing, and also the frequency of printing errors.

RADC is also doing other work in the MT field besides developing hardware. A contract with the University of Washington has brought forth a lexicon in the order of 500,000 words with Russian as the "source" language and English as the "target" language. These words will be used on the photo memory of the mechanical translator. RADC scientists are also aiding others in supporting the very interesting work of Dr. Oettinger at Harvard in linguistic work in producing scientific dictionaries automatically. We are also supporting the longer range efforts of the Cambridge Language Research Unit of Cambridge University. This research centers about the use of logical methods utilizing the thesaurus approach in obtaining a translation breakthrough in the multiple meaning problem. Here thesaurus¹ means "an organization of word usage in an ordering dependent on logical content (rather

¹ Report on the work of the Cambridge Language Research Unit for the National Science Foundation prepared by Gilbert W. King dated July, 1958.

than on alphabetic content as in a dictionary)." These two efforts are supported jointly with the National Science Foundation.

RADC scientists are also aiding in the support of the Research Group of the Center of Studies on Linguistic Activity and Cybernetics, University of Milan, Italy. This work is a continuation of the research studies performed on mental operation and semantic connections. The Research Group is pursuing the approach that man has fundamental order in his thinking process and that these are elements of a correlational net. Taking this correlational structure of thinking and mastering the semantic connections which link the input and output language within this structure, they believe, will be a solution to some of the more difficult problems in mechanical translation.²

We have a development that is completed and although it is classed in the field of information dissemination, we mention it here because it is used in association with storage and retrieval devices. We feel that dissemination exists as an important problem in the continuous flow of data in the field of data handling. This function can be automatized; the equipment referred to is the automatic disseminator jointly developed by RADC engineers and Magnavox Research Laboratory. The disseminator determines what groups are qualified to receive a given document and controls the production and addressing of copies so as to insure that the qualified groups get their copies quickly. The disseminator must determine on the basis of the subject and geographical area of coverage of a given document who is qualified to receive a copy of that document. The disseminator input, as used in one case by RADC, is the flexowriter tape that was used in the Minicard camera for control and code input. The information on the tape is compared to the stored requests in the disseminator as stored in a magnetic drum. The output is tape that contains control data for manufacturing duplicate Minicards based on a match in the disseminator.

² S. Ceccato, "Mechanical translation," *Automaz. e Automat.*, p. 1, April, 1958.

SOME REQUIREMENTS OF THE FUTURE

After this cursory review (and I hope some insight) into information retrieval and mechanical translation development efforts of the RADC, we come to a question of what lies ahead in these two fields. Before I go too far in this direction, I would like to mention that the Air Force has a cardinal interest in the national problem concerning technical information. As can be seen by our efforts, we are going through a "development era" which we feel will have a great influence on the national technical information picture. This is a natural feeling to come from a group that is engaged in developing techniques and hardware such as language research, print readers, automatic language translators, storage and retrieval devices, and disseminators. Equipment such as this will, out of necessity, play an important part in the national picture in both centralized information systems efforts or in decentralized efforts.

Being in the development field, one supposes we should have fine prediction qualities in the semantic-graphic data handling field. Frankly it boils down to studying the trends, following the curves and coming out with the statement that future equipment in these fields should have faster scanning rates, higher excess speeds, greater packing power, lower power requirements, lower cost, etc. However, anyone can make those predictions, but in speaking for a group which has a real invested interest in these fields, we feel the empirical exploitation of developed equipments should be aggressively pursued *and* that much more should be done in language research for both information retrieval and MT. We feel some effort is "coming about" in this field but many more "bold steps" must be undertaken. Mechanical translation by itself is a language problem, and, by its solution and future use, we only add more literature in the already heavily loaded field of storage and retrieval. Being engineers and scientists we tend perhaps as a group to shy away from the language research side of information retrieval and MT. However, we have slowly learned over the past few years that herein lies the ultimate solution to our immediate common problem.

Computing Educated Guesses

E. S. SPIEGELTHAL†

TO DISTINGUISH himself from the poor benighted man-in-the-street, the computer sophisticate is apt to refer to the beasts as "so-called giant brains" or as "lightning-fast idiots." He knows, as do we, the great gulf which separates the human brain from the general-purpose digital computer. Still, the exact dimensions of that gulf are quite unknown, and the desire to show that the hiatus between man and machine is smaller than many suspect impels both the adventuresome and the iconoclastic. The attempt, the successful attempt, to automate one area hitherto considered an exclusively human domain constitutes my topic today.

We are all familiar, if only by hearsay, with the troubles that can beset the best of computer programs if the input to the program is not thoroughly debugged. For a program designed to test the putative behavior of, say, a proposed steam turbine, where the input consists of a scant dozen or so parameters, input debugging is hardly a problem. The situation is quite different for a data-processing operation, particularly when the input is massive, as it usually is. Three alternatives, all unpleasant, present themselves to the supervisor of such a large-scale data-processing operation. He can build a wide variety of error-detecting features into his program, flagging all input errors for subsequent human correction, he can employ a host of human pre-editors to clean up the input, or he can hope that input errors are rare, and let it go at that.

Unhappily, there are many applications where errors are not rare, where the do-nothing solution is obviously frivolous and where, consequently, a sizeable group of humans is necessary, either as pre-editors or as on-line trouble-shooters. Nor is it always the case that the necessary human beings can be clerical types. Certain input debugging calls for sophisticated and knowledgeable practitioners. We are all hopeful—almost all, anyway—that keypunch machines and operators will sooner or later be superseded by character-reading devices and the like. There is no philosophical difficulty in conceiving of typed, printed, or handwritten characters being translated directly into computer language without any human intervention, provided, of course, that those characters were correctly typed, printed, or written to begin with. Suppose, however, that the source characters are incorrect. Consider the ingenuity expended in the Post Office just in recognizing all the variations of "Albuquerque." Our Russian colleagues are supposed to be far advanced in the domains of automatic translation and character-reading, but present their machines with

a first edition of "Cybernetics," with all its typographical errors, and horrible difficulties would ensue. Our choice, then, is clear. Either we admit that many important data-processing applications are impossible to automate completely, or we find a way to mechanize the human capacity for making educated guesses. We believe that, for some applications at least, we have found a way.

While the techniques we have developed were conceived with one particular application in mind, I shall describe them without reference to that application, successful as it was. The principal reason for taking this tack is to be able to present the basic, quite general, features of our method without being tripped up by the special form-fitting required by the actual problem. So, let us be general, and consider any language with which humans attempt to communicate with one another. These may be natural languages, like English or German, or artificial languages like Esperanto or certain telegraphic codes. There are all sorts of personal reasons for communication being difficult—ignorance, dogmatism, poor sentence structure, etc.; however, even if these factors did not exist, all sorts of nonhuman noise would beset would-be communicators. Information theory makes much of "redundancy" as an aid in error-detecting and error-correcting when a noisy channel is being used. Indeed, even humans who have never heard of information theory make continual, and skillful, use of redundancy in unscrambling all sorts of garbled communications, whether the trouble be cross-talk in a telephone conversation or missing letters in a crossword puzzle. Without attempting to build a model of the brain, replete with neural nets and such, let us see if we can single out the functions performed by human redundancy-exploiters. If these functions turn out to be performable without recourse to extrasensory perception or to the psychokinetic effect, our automation problem is essentially solved. There remain only the minor problems of collecting all the necessary data, carrying out a rather gruesome programming task and finding a computer fast enough and capacious enough to make our solution practicable. I shall return later to this question of practicability. At the moment, allow me to sketch the functions which, when suitably programmed, allow a general-purpose computer to simulate a redundancy-exploiting, error-detecting, and error-correcting human being.

Rather than jump into a completely general and abstract formulation, let me use a concrete illustration. Fig. 1 shows two familiar sights, a correctly prepared mailing envelope and, below it, a somewhat sloppier version of the same thing. We shall assume at first that a

† General Electric Co., Bethesda, Md.

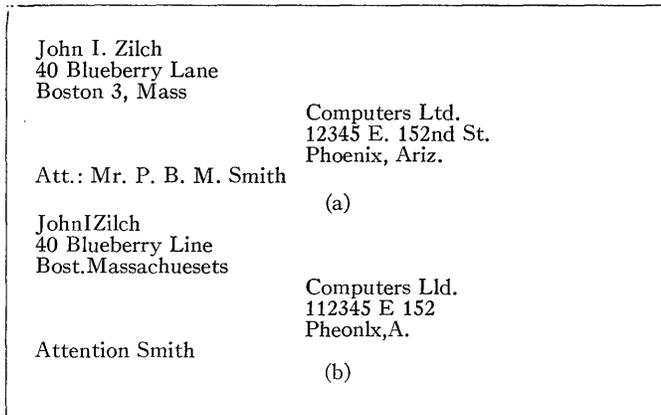


Fig. 1—Envelopes, (a) good, (b) bad.

“perfect” character-reading device has “read” the perfect envelope, and consider the functions which must be performed by a machine to “understand” the envelope, *i.e.*, to route it to the correct addressee by the desired means, *e.g.*, air mail or first class. We shall then consider a fallible character-reading device reading the lower, garbled, envelope, and see what can be done there. It should be emphasized that, in this application, we are not concerned with the essentially straightforward task of actually routing the envelope to its destination. Our job here is just to ascertain the information needed by the routing program.

Our machine must perform two separate functions on each “word” read from the envelope. A word here is any group of contiguous characters on one horizontal line, not containing any embedded blanks or commas. Given any such word, the machine must first ascertain the class of words represented by this word. In our example, the machine must determine that “Phoenix” is the addressee’s city, and that “I.” is an initial of the sender. This first function is called the “identification” of the word. The second function is that of “recognition.” Having established the class to which a word belongs, it is next necessary to determine which one of the class members the given word represents. In our first example, the recognition process is simple, almost trivial. “Phoenix” is matched against every element in a master list of cities and, lo and behold, it is found that “Phoenix” is “Phoenix.” A glance at the lower envelope on Fig. 1 will reassure you that the recognition problem is not always a trivial one.

The identification process is fairly easy for a Gestalt-perceiving, pattern-recognizing human who is himself accustomed to writing envelopes according to the standard format. The machine needs a little help in this direction. Fortunately, we can provide this help. On the one hand, we can program our machine to elicit the same data that our pattern-recognizing facility allows us to obtain. Clearly, our character-reader will be able to note, for each word, its relative position with respect to all other words on the envelope, and its position with respect to the envelope itself. For each word, then, we

start off with the knowledge of the line it is on, its position on the line (left end, right end, interior) and the words which flank it on either side. With a little extra programming effort we can determine the length, *i.e.*, the number of characters of each word, its character pattern (is it all alphabetic, all numeric, some sort of hybrid?) and, perhaps, the presence in the word of some salient feature, *e.g.*, the colon following “ATT.:". Indeed, we can usually determine quite easily much more information than we need for the identification of our words. Much more, that is to say, when we are dealing with a noiseless channel, and/or a communication format as simple and relatively invariable as the front of an envelope.

Of course, whether this information is adequate, overly complete, or inadequate depends on how we use it. At this point in the identification process, the machine must turn to its accumulated store of factual knowledge, a store which is compiled by a subsidiary program in advance of production running. This store consists of lists and tables of probabilities, and provides the data which, in conjunction with the specific information for each envelope, allow each word to be identified with a high probability of correctness. Our basic technique here is the use of Bayes Factors as instruments for weighing evidence. Fig. 2 gives the essentials of this technique.

For each class of words that can occur in the specific type of communication in question—mail envelopes, in our example—an *a priori* probability is given for the occurrence of a representative (or two, or n) of that class. This probability, like all the others we use in this process, is derived from frequency counts on sufficiently large samples of the data to be processed. Also for each class, we provide the probabilities that, for example, a specific representative of that class will have length 3, or 4, or 5, or that the class representative will be found at the beginning, or the end, of a line. In brief, for every piece of information we scan each envelope for, we have a corresponding set of probability distributions, one set for each class of expected words.

In the identification phase of our program, we consider one actual word at a time, testing that word against the hypotheses that it is a representative of expected class A, B, etc. Eq. (1) in Fig. 2 gives the skeleton of such a test. Here we are testing the hypothesis that the word “Smith” is a representative of the “zone-number” class. Our frequency counting is supposed to have informed us that the *a priori* probability that any word on our envelope is in the zone-number class is 0.017. We first test our hypothesis by using the empirically-determined fact that “Smith” has length 5. This gives us our second term on the right side of (1), *i.e.*, the Bayes Factor for the “length event.” The product of the Bayes Factor and the *a priori* probability is the *a posteriori* probability that “Smith” is a zone-number. Not very surprisingly, this is a small number. We now compare this number with two thresholds. If the *a*

$$P(H/E_1) = P(H) \cdot \frac{P(E_1/H)}{P(E_1)} = P(H) \cdot \frac{P(E_1/H)}{P(E_1/H)P(H) + P(E_1/\bar{H})P(\bar{H})} \quad (1)$$

$$= (0.017)(0.0001) = 0.0000017 \begin{cases} > 0.000001 = \text{Rejection Threshold} \\ < 0.9 = \text{Acceptance Threshold} \end{cases}$$

$$P(H/E_1 \cdot E_2) = P(H) \cdot \frac{P(E_1/H)P(E_2/H \cdot E_1)}{P(E_1 \cdot E_2)} \sim P(H) \cdot \frac{P(E_1/H)}{P(E_1)} \cdot \frac{P(E_2/H)}{P(E_2)} \quad (2)$$

$$= (0.0000017)(0.00001) < 0.000001 = \text{Rejection Threshold}$$

where

- H = hypothesis that "Smith" is a "zone-number"
- E_1 = the event that the length of "Smith" is "5"
- E_2 = the event that the pattern of "Smith" is "all alphabetic"

Fig. 2—Hypothesis testing.

posteriori probability exceeds the acceptance threshold, we accept the hypothesized identification and turn our attention to the next actual word; if the probability falls below the rejection threshold, we reject the hypothesis, and test the actual word against the next expected word class. Finally, if our probability falls between the two thresholds, we test the same hypothesis against the next event, using our *a posteriori* probability as the new *a priori* probability. In our example in Fig. 2 we have been generously low with our rejection threshold, so that it is necessary to go to (2) where we test the hypothesis, "Smith = zone-number," against the character pattern, and allow the low probability of a zone-number consisting exclusively of letters, to push our hypothesis into limbo. If we were scanning French addresses, with zone-numbers given in Roman numerals, the Bayes Factor in (2) would be very different.

After scrutinizing all the actual words on the envelope in this manner, we may find that certain words are still unidentified. In this case, we iterate through our process once again. However, certain features of the process will have changed. Suppose that we have identified two different zone-numbers in the first pass. Since we expect to find no further zone-numbers, we no longer test any of our undecided actual words against the hypothesis that they are zone-numbers. This not only reduces our processing time—it also changes the *a priori* probabilities of the remaining word classes, and affects the numbers entering into all the Bayes Factors. Another change in the second pass is that new evidence can be used to give rise to Bayes Factors. A word identified as a zone-number in the first pass provides strong evidence that the word to its left is a city name. Clearly, the topological relationships subsisting between words cannot be utilized until some words have been identified.

If successive identification passes still leave a residuum of unidentified actual words, as might happen if, for example, two or more words were run together, thus appearing to the machine as one word, there are subsidiary tricks that can be played. Due to time limitations, I shall have to leave these tricks to your imagination, and move on to the recognition phase.

In the simplest case, all actual words will have been correctly identified and, if the words are all correctly spelled and correctly ingested by our character-reader, recognition will consist of little more than finding the exact match in the proper list, a list determined by the identification of the word. It is possible to make even this simple process simpler or, at least, faster. To search a list of all the cities in the United States can be time-consuming, particularly if the list must be transferred from tape to core memory. However, if the corresponding state has previously been recognized, then a much reduced list of cities can be inputted and searched. Suppose further that the corresponding zone-number has been recognized as "25." Then we need consider only those cities in the given state which have at least 25 zones.

If we are bound to get a direct match whether we scan a big list or a little list, this process of list reduction is of secondary value only. It is when a direct match is not forthcoming that this technique assumes greater importance. In the absence of a direct match, we are constrained to use brute force techniques of a more or less sophisticated nature. If we are fortunate enough to reduce a list down to one entry, then we can avoid brute force completely. Failing this, we can expect two advantages to accrue to the use of a reduced list, in general. First, for the same elapsed time, we can employ more brute force techniques per list entry; second, we can at least hope that, by reducing our initial list, we will expunge spurious candidates to which our brute force techniques might give scores equal to, or even greater than, the score of the correct candidate. For example, Fig. 3 gives one horrible example, often quoted in this connection. Recognizing (a) as being either "New York" or "Newark" is an awful job. A non-brute-force technique, such as list reduction, which removes the false entry from consideration is a welcome way of cutting this Gordian knot.

Again for lack of time, I must give the actual brute-force techniques a very hasty treatment. Let me mention just two techniques of the many available. To match a word which has had two letters transposed [as

in (b) in Fig. 3], against the original word, we look for list entries with the same letter composition as our actual word, *i.e.*, entries with the same number of A's, B's, C's, etc. Scanning these for a single transposition is relatively easy.

A second technique is useful when a letter or two (or more) has been erroneously dropped from, or added to, a word. (c) in Fig. 3 is due to a stuttering typist who repeated the first letter of the word. Two words run together provide further examples of this kind of noise. What we try here is a direct match of our actual word with a proper subset of our list entries, and vice versa.

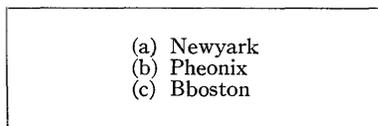


Fig. 3—Typical typographical errors.

If no amount of brute force seems to work, and certain words just cannot be recognized, we can either give up gracefully at this juncture or we can admit, even more gracefully, that one of our educated guesses might have been wrong. If we choose the latter alternative, we have the messy job of deciding whether we went haywire in the recognition phase, or all the way back in the identification phase. In either case, it is still necessary to find a likely spot for picking up the dropped stitch without causing the entire garment to unravel. Sometimes, indeed, we are left with the original ball of wool. These, however, are almost always the cases which stump human editors.

This ability to iterate back, and back, and back, can of course lead to excessive use of computer time. It does have its advantages though. It means that a bad guess is not an irrevocable misstep. It also means that various parameters, the identification acceptance and rejection thresholds, for example, are not nearly as critical as they would be in a once-through process. Since these are among the hardest parameters to estimate accurately, any diminution of their sensitivity is a positive gain.

At this point, I should like to restate our major techniques in somewhat folksier terms than "Bayes Factors" and "list reduction." In our identification phase, we attempt to use the constraints imposed by the format, mailing envelopes in our example, plus the constraints of the language itself, the length and character patterns of the expected word classes, to provide a rudimentary form of pattern recognition. We then use our *a*

priori knowledge of word statistics and interword relationships to find the most probable matching of the actual words to the expected word classes. Human beings presumably use rank orderings of hypotheses, modified by intuition, to perform such matchings. Since machines lack intuition and since we have not yet developed a calculus of rank orderings, we use the paraphernalia of Bayes Factors to accomplish the same task.

Without undue stretching of the terms, we might say that, in the identification phase, we exploit the syntactic constraints on the language we are processing, whereas in the recognition phase, by our use of the list reduction technique, we exploit the semantic constraints. We might subsume both types of constraint under that much-abused word, redundancy. As for a folksy term for our brute-force techniques, the most accurate that occurs to me is "knowledgeable cynicism." We expect errors to be made and, usually, we have some information as to the kinds and sources of error, as well as their frequencies of occurrence. If we know that typists frequently hit a key next to the one they should hit, we store the keyboard pattern in our program; if we know that our character-reader frequently confuses "o" with "c," that, too, goes into our dossier.

A final word now as to the applicability and practicability of our techniques. What with tape searches and Bayes Factor computations, processing time may, but need not always, be excessive. The preparation of all the lists required in the recognition phase is a painful task. With a relatively stagnant language, this list-making can be a one-shot ordeal; with a volatile language requiring frequent updating of the lists, the pain might be unbearable. What has been said about lists also applies to the preparation of the probability tables for the identification phase. A final pause-giving consideration is the amount of redundancy in the language to be processed, particularly when the processor cannot establish the language, which he sometimes can do. Our private feelings are that a language sufficiently low in redundancy to be unintelligible to a machine will also be unintelligible to a man. I won't press a point which trods so heavily on anthropocentric toes.

In summary, then, we feel our techniques can be useful in some massive data-processing applications, in automating post offices, in translating natural languages, where every second word in the source language has several correlates in the target language, and we know our techniques have worked at least once. I won't ask that you take this on faith, though I'd appreciate it if you would.

A Memory of 314 Million Bits Capacity with Fast and Direct Access—Its Systems and Economic Considerations

N. BISHOP† AND A. I. DUMEY‡

THE graphic arts laboratory of Time Inc. in Springdale, Conn., is a laboratory largely devoted to improving the arts of printing and paper-making. Of equal importance to a publisher of weekly news magazines is a large and well-serviced list of subscribers. This may serve to explain why there exists today in Springdale a completed and working engineering prototype of a large direct-access memory. This equipment was developed by the laboratories to prove the technical and economic feasibility of storing and processing large basic record files in memories providing direct access facilities to individual records. Since our program of systems study, equipment development, and economics study has indicated such feasibility, a review of our findings may well open new avenues of approach to the solution of present and expected future problems in the expanding field of automatic data processing.

The data-processing systems designer is constantly striving to strike an optimum balance between the costs of data storage, computer time, job requirements of input and output, and manpower. So fast is the art progressing that today's best balance for customer *A* may be quite different from that for customer *B* a year later, even though *A* and *B* present identical job requirements.

The concept of large data-storage capacity coupled with direct access to individual records has long intrigued the systems man. His interest is usually diminished when he starts to consider the cost of storing large basic files in such a manner with the presently available memories providing such direct-access facilities. Either the storage cost per bit is too high or the access time is too high to keep up with the daily work load. Serial storage provides high capacity and low cost per bit, but it does not provide for the often essential requirement of direct access to individual records. There appears to be a definite gap in the systems man's bag of tricks and hardware which needs filling by a direct-access memory of high capacity, reasonable access time, and reasonable cost per bit for storage.

In designing a high-capacity direct-access memory for use in very large basic record files, it is necessary to make as many records as possible accessible to each processing station, if low cost per bit of storage is to result. At the same time, access times must be low enough to handle traffic requirements. Excessive packing factors result in intolerable tolerance requirements

for manufacture and maintenance. Excessive travel distances from record to processing station result either in immoderate acceleration and torque requirements or excessive access times, if acceleration and torque requirements are kept within bounds. Too many records per access mechanism, with consequent increase in traffic per unit, can overburden the designer with access-speed requirements. The key to an approach which will meet systems and economic requirements is expressed in one word—moderation. A combination of moderate packing factors, moderate travel distances, moderate accelerations, and moderate average access times can result, and has resulted, in a fast and direct access memory with a record capacity sufficient for economical large basic file application. Let us now discuss its operational characteristics.

The significant characteristics of a direct access memory to a systems man are those which answer the questions: "How much data can I store?" "How many entries can I process per working day?" "What is the best available estimate of the cost per bit of storage plus provision for direct access?"

As far as the bit capacity of presently available direct-access equipment is concerned, the upper bound seems to be a few tens of millions; and, on the cost side, the lower bound seems to be some ten bits for the penny. Access times in such memories vary from a second down to a few milliseconds, the latter figure reflecting the need to cover high activity against the file, as in the case of short-term airline reservations.

The engineering prototype of our memory has a total information bit capacity of 314,500,000. This capacity is distributed in 262,144, (2¹⁸), separate storage locations within the memory, each location providing a message capacity of 1200 bits. The operation of the memory is under the control of three basic commands derived from peripheral equipment. The first command consists of an 18-bit parallel-fed address, followed by a seek record signal. The second is a process command calling for a read, write, or erase cycle. The third is a release-record command, which restores the memory to readiness for operation on the next series of commands.

The maximum time required to execute a seek record command is 3.87 seconds and the minimum is 0.61 second. Each process cycle, read, write, or erase, takes 0.45 second. On completion of the last process cycle, 0.6 second is required to restore the memory to a condition of readiness for the next series of commands.

These are the basic figures. Here are some examples

† Time Inc., Springdale Labs. Div., Springdale, Conn.
‡ Systems Consultant, Roslyn Heights, N. Y.

of how they would be applied to practical situations.

In an information-retrieval system, in which additions to the file tend to occur rarely, the over-all average working cycle is 3.14 seconds, which includes 0.45 second for one read cycle. Maximum and minimum figures would be 4.92 and 1.66 seconds, respectively.

In our own operation, about 95 per cent of the actions on the file call for five processing cycles per access. This results in an over-all working cycle of 6.72 seconds maximum, 4.94 seconds average, and 3.46 seconds minimum. These last figures imply that, under an even traffic flow, over 2 per cent of the storage locations can be consulted and processed in an eight-hour day.

Obviously, file arrangement, input form, and other considerations are important factors in arriving at a working average of access and processing cycle for any direct-access memory. The maximum, minimum, and average figures will permit a rule of thumb estimate of the applicability of this equipment to various problems.

In our own case, there were serious queueing considerations and addressing limitations, which made the actual speed of the device available as a safety factor for an actual file activity of about 1 per cent of the file per day.

Neither time nor circumstances permit a complete description of our engineering prototype at this time; however, some of its physical characteristics may be of general interest. Information bit rate in and out is 20,000 serial bits per second. Information input and output levels are adequate for computer communication purposes. Signal-to-noise ratio, measured as the ratio of peak signal voltage read from a recorded location to the peak noise voltage read from an erased location, exceeds 40 db. Semiconductor components are used throughout. The circuits within the memory cabinet provide all systems communication and control facilities except buffering. Power requirements are 550 watts average and 750 watts peak load on a 115-volt 60-cycle single-phase circuit. The prototype, which weighs 4800 pounds including its glass and steel enclosure, occupies a floor space of 51 inches by 62 inches and is 95½ inches high. It is designed to operate in average office conditions of temperature, humidity, and dust content.

This equipment is now set up for demonstration and cycled testing in the laboratory. It operates under the control of a device which was designed to run it in a test mode. The test equipment consists of a Flexowriter, 1200-bit shift register buffer, comparator and control circuits which also provide coding interlocks. Keyboard control, tape reader control, or a combination of the two is possible; and information may be stored, edited, justified, transmitted, and received. Special test tapes have been prepared which permit repetitive cycles of access and processing at rates, and in a manner, simulating hard usage in a system. The comparator and counter circuits associated therewith permit a wide variety of error checks. Input and output can be represented on hard copy, punched tape, or a combination of the two. Many test runs have clearly indicated the few

desirable design modifications for inclusion in a production unit.

In order to complete our economic studies on the application of a direct access approach to the handling of our subscription records, we had an independent manufacturer prepare a manufacturing cost estimate on the basis of duplicating the prototype memory in quantities necessary for such application. This estimate indicated a cost of storage of 2 cents per bit or less. Attractive as this figure of cost per bit for direct-access storage may be, the cost of storing billions of bits of data, such as a large publisher's subscriber records, represents a considerable investment for storage alone. However, the compensating factors are considerable. First, a potentially better grade of subscriber service is possible, based on faster action on subscriber requests; and second, a considerable saving can be made in costly computer decision-making time, as a result of limiting such computer time requirements to entries demanding decision.

The original specifications for our memory were based on the concept of storing all file data on each subscriber in one addressable location of 1200-bit capacity. As our studies progressed, it became apparent that it is not necessary to store all file data under direct access, since many of the data are not required for daily file maintenance, and can be stored to better systems operational advantage in serial access memories. Since we planned to store the name and address of subscribers on serial tape for operating address label printers, and the name and address always appear as input with each action against the file, it is unnecessary to store these data under direct access. Complete billing information can also be stored on tape, as it is not required for the daily maintenance job. It was decided to limit the storage per subscriber under direct access to the minimum required for decision-making in daily file maintenance plus the few characters necessary for minimizing duplication. The data retained are sufficient to provide systems communication with related files. Thus, it is now possible to store the decision-making data on two subscribers in one 1200-bit storage location. While traffic and overflow requirements have not permitted a 50 per cent reduction in the number of direct access units required in our system, they have enabled a 30 per cent reduction without impairing systems performance.

Efficient usage of direct-access storage capacity requires particular attention to the matter of distribution of entries within the file. This is especially true when input to the file is uncontrolled, as in a subscription service operation. An answer to efficient distribution within the file was found in an addressing procedure which translates a subscriber's name and address into a random and reproducible address within the file. Overflow capacity within each memory is reserved to handle new entries addressed to fully occupied storage locations.

An action against the file usually begins with a letter including the subscriber's name, address, and requested action. The file location for a subscriber, whether a new or an old customer, is derived from the most reliable

portions of the name, street address, city, and state. As stated above, to attain efficient usage of memory capacity, the data supplied by subscribers must be manipulated by a computer into addresses which distribute randomly over the field of available storage locations; and furthermore, identical subscriber data must always produce the same address. We have chosen a computing technique which, in brief, considers the subscriber-supplied data as a binary number, divides it by a prime number almost as large as the number of addressable locations, and uses the remainder as the address.¹⁻³ Such randomizing techniques lead to occasional duplication, and the system must be capable of distinguishing between such cases and of providing accessible storage for each. Entry data comparison with data stored at an occupied address in the memory provides this distinction. If the comparison indicates that the entry is in fact a new one, and if the location is fully occupied, the new entry is referred to a directory of unallocated storage locations specifically reserved for such overflow cases. The first such location address readout directs the access to this location where the new entry data are stored. This address is, at the same time, erased from the overflow directory and recorded in the original storage location. Subsequent access to the new entry is attained without reference to the directory, as the overflow address is now stored in the addressable location computed from subscriber-supplied data. The directory of available unallocated storage locations is kept current by re-entering storage locations allocated to overflow when they are emptied by subsequent file action. This approach to the attainment of economical distribution of entries in a direct-access file system appears to result in a favorable balance between the total number of addressable locations required and the multiple accesses required to reach entries stored in locations allocated to overflow. Other equally favorable solutions may result from specific choice of final systems componentry.

The principal output of a systems design for subscriber service is in the form of address labels printed on a weekly schedule. Conventional tape storage is the logical way to store the mailing list for efficient control of output printers. This requires weekly preparation of a new mailing tape, which is produced by collating a weekly change tape with the previous week's mailing tape. Computer requirements for this relatively simple collating operation are moderate, and our studies indicate that low-cost computers are adequate for this operation. A tape converter is included to translate the output of the file maintenance portion of the system into the tape format used by the collator. Decision-making computer requirements for file maintenance have been

minimized, since the daily work load is limited by direct access storage to an estimated average of 100,000 items per day against a total file of 12,000,000 entries, some nine to ten million of which are active.

We proposed to use a certain medium size computer for updating, because, although the list size is 12,000,000, the daily work load of this portion of the system is in the range of 100,000 items. In other words, instead of having to cope with about 400 items per second in order to run through the file in one day, the file maintenance equipment has to handle about four. In recapitulation, we can say that the more complex items, totalling about 100,000, make a day's work for a medium-size computer, while the 9,000,000 items of mailing information (only $\frac{3}{4}$ of the file is active) are handled by simpler tape collators.

Communication equipment was required to join the memories with the main computing element. Our plans called for a combination of a limited number of buffers, and a cross-bar switching arrangement, the speeds of each being consistent with the expected ratio of working cycles to access or rest times. Other details, too bound up with the specific use of the equipment to be of interest, will not be discussed. We did consider, as every user of new equipment must, the problems of conversion, including build-up of the file and parallel operation of the new and old system.

Our systems studies indicated that conventional techniques and readily available equipment could handle the systems functions of input, communication between computers and associated memories, control and switching, and output printing. We shall not dwell on these systems aspects other than to say that their combination with the processes described above resulted in a positive answer to the question, "Can direct access memories be applied with good economy to the daily maintenance of very large subscriber files?"

Our subscription service is a large-inventory, high-volume, data-handling problem with many complexities due to the nature of the weekly news magazine business. The amount of equipment necessary to the solution of this problem is determined by the job requirements and not by the operational characteristics of the direct-access memory we have described. There are other applications where the combination of such a memory with a small-scale computer will perform all necessary systems functions. Newly developed and working equipment opens consideration of new fields for electronic data processing, and it is our belief that this development will provoke such consideration.

We have shown that very favorable costs per bit can be achieved if traffic requirements are modest. Systems considerations have been noted. Testing experience has been given. Addressing and retrieval ideas have been set forth. We believe that a workable, large-scale, direct-access memory, with an access time of a few seconds, has a place in the roster of useful data-processing equipments, and we invite your attention to its possibilities.

¹ L. N. Korolev, "Coding and code compression," *J. Assoc. Comp. Mach.*, vol. 5, pp. 328-330; October, 1958.

² W. W. Peterson, "Addressing for random access storage," *IBM J. Res. Dev.*, vol. 1, pp. 130-146; April, 1957.

³ A. I. Dumey, "Indexing for rapid random access memory," *Computers and Automation*, vol. 5, pp. 6-9; December, 1956.

Information Retrieval on a High-Speed Computer

A. R. BARTON†, V. L. SCHATZ†, AND L. N. CAPLAN†

THIS discussion concerns a mechanized information retrieval system for the technical library of General Electric's Aircraft Gas Turbine Division. The system is confined to the technical reports and papers available in the Division's Library. Textbooks have not been included as of yet since they are carried on a Library of Congress system and did not readily fit into the manual scheme which was developed. This discussion will be in two parts. The first part will cover the information retrieval system prior to mechanization, and the second part will cover the mechanization of this system.

The technical library was established in 1953 using a uniterm or key word coordinate indexing system. To understand this system, we will follow the progress of a publication through the various steps of the system. First, as a publication is entered into the library, it is assigned a six-digit access number (see Fig. 1). This report is typical of the technical reports generated within the Division. Another example of reports in the library would be NACA technical reports. The next step in this system is the abstracting of the document. The abstracting is done by professional librarians and then posted to a card file as shown in Fig. 2. This card is controlled according to the previously assigned access number. The next step in the system consists of reviewing the title, abstract, and document to select the most descriptive words which will identify the document. These words are primarily nouns. They become the uniterms. In our hypothetical case, these words are shown on the right in Fig. 2, just as they appear in the system.

These uniterms, along with the appropriate access number, are then posted to the uniterm file (see Fig. 3). There are 100 numbers per side of card when full. Both sides are used, and in the case of general terms such as these, they are heavily posted so that several cards are required. Certainly this system appears cumbersome at this point, but it has the advantage that, in any given technical area, the number of uniterms tends to level off at a specific number after the system is developed. In our case, this number is something under 9000. The combined system is shown schematically in Fig. 4. I think now you can see how information is recalled from this system. The requestor discusses his problem with the librarian. They decide on the uniterms to search. The librarian then furnishes the appropriate uniterm cards to the requestor. Once again referring to Fig. 3, the problem facing him can be seen. He must cross coordinate the cards to find numbers which apply to all uniterms. In our case, we have used three uniterms. A

more typical case would be four to six uniterms but with less access numbers per term. In any case, if the requestor is persistent, he will come up with some of the matching numbers. The librarian can then go to the abstract file for the abstract cards. These are perused by the requestor who in turn selects from the abstracts those documents he desires to read in full. In the case of the three uniterms we have selected, each has over 1000 access numbers posted to it. It is easy to see the difficulty of cross coordinating these cards. This difficulty did little to promote the use of the technical library. The result was duplication of experiments, technical studies, etc., with the attendant delays in time and increases in cost.

Obviously, something better was needed. That "something better," we feel, was the program written for our IBM 704. This program is basically a mechanization of the manual system with very little effort to change the system itself. This program is in two parts. Part one is file updating and the cross coordinating of the master uniterm tape, or, referring to the manual system, the uniterm card file. Part two concerns the selection and printing of abstracts. (Part one can be run independently of part two if desired.)

Cross coordinating, at present, is done on a strict AND system. That is, an access number must appear under each uniterm used in the search. The possibility of an AND/OR system was considered and rejected. However, the AND/OR approach is being used on a modification of this program for a different type of information retrieval system. If no information is found, a modified list of uniterms can be developed by the requestor and the librarian, and another run made.

It was decided during the development of the program that one of the features of part one was to update new information into the file at the time of cross coordinating. This decision was based on the knowledge that in any information retrieval system there is a major problem of keeping the system updated. Thus, all uniterms with their associated access numbers were stored on the master uniterm tape in alphabetic order.

All search information and updating information can be read into the machine from either the card reader or tape. This information is processed to determine if it is in alphabetic order and if there is any updating information. If the input is not in alphabetic order, it will be sorted within the machine. If there is no updating, a new uniterm master tape will not be made up. Both of these decisions are made within the machine by interpreting the input information.

To utilize best the 32,000 locations of memory in the 704, it was decided to use long records of information.

† General Electric Co., Cincinnati, Ohio.

CLASSIFICATION
AGT-TECHNICAL INFORMATION SERIES

REPORT NO. M57AGT3471

10117

A STUDY OF COMPRESSOR VIBRATION UNDER
VARIED LOADING CONDITIONS

By

J. Doe

JET ENGINE DEPARTMENT
AIRCRAFT GAS TURBINE DIVISION
GENERAL ELECTRIC COMPANY
CINCINNATI 15, OHIO

CLASSIFICATION

Fig. 1—Title page of AGT Report.

Report No. M57AGT3471

A STUDY OF COMPRESSOR VIBRATION UNDER
VARIED LOADING CONDITIONS

AUTHOR: J. DOE 6/15/58

AIRFLOW
COMPRESSOR
LOAD
SPEED
TEMPERATURE
TURBOJET
UNBALANCE
VIBRATION

A STUDY OF COMPRESSOR VIBRATION WHERE THE
ROTOR WAS PURPOSELY UNBALANCED AND THEN RUN
UNDER VARIED CONDITIONS OF AIR FLOW, TEMPER-
ATURE AND SLEED FROM IDLE TO 100%. INCLUDES
DETAILED CHARTS, GRAPHS AND ANALYSIS

Fig. 2—Abstract card with uniterms.

Many of the uniterms, with only a few access numbers posted to them, are combined into one record with a total length of not over 7000 words. A total of 9000 locations is reserved for reading in these records. If any record exceeds 7000 words, the program will try to separate this record into two records. If the record exceeds 8995 words and cannot be broken down, the program will be halted. Because of this factor, no uniterm can have over 8995 access numbers posted to it. For a larger system, this could be easily modified so that there would be no limit to the number of access numbers posted to a uniterm. Another feature to help cut down on the length of records is whenever a new uniterm is

TURBOJET CONT'D

TURBOJET									
00010	00001	00102	01003	00024	00205	02006	00007	00308	04009
00030	00201	00122	01243	00104	00405	04006	00107	00318	04579
00170	00301	00132	01363	00404	00605	05496	00337	00358	04989
01340	00401	00142	01483	00654	00805	06996	00437	00478	05089
02240	00501	00152	02223	00764	00905	07346	00517	00598	06789
03450	00601	00162	03453	00844	01105	07466	00667	00688	07229
04680	00791	00172	04533	00874	01205	08796	00717	00778	08819
05890	00881	00182	05873	00894	01345	08946	00837	00858	09009
06770	00941	00192	06433	00924	01495	09176	10117	00918	09509
07650	01561	00192	07323	01224	01765	09196	10227	01018	10239

TOTAL ACCESS NOS. = 1249

COMPRESSOR CONT'D

COMPRESSOR									
00060	00111	00112	01013	00034	00205	02006	10117	00338	01029
00070	00121	00222	01113	00114	00215	04986	10227	00428	01339
00090	00151	00452	01123	00334	00335	05556	10327	00668	02439
00100	00331	00492	01443	00364	00575	07136	10557	00718	03389
00300	00541	00552	01563	00444	00775	08086	11127	00848	04659
00500	00721	00772	01773	00584	00795	09996	11247	00868	05579
00700	00831	00792	01783	00774	00885	01436	11417	00918	06119
00800	00971	00842	01883	00864	00975	01556	11607	00928	08889
00900	00981	00862	02323	00934	01125	01886	11887	01198	09239
00910	00991	00932	03773	00984	02455	01976	11997	01228	09999

TOTAL ACCESS NOS. = 1795

VIBRATION CONT'D

VIBRATION									
02320	01001	03002	00033	00244	01115	00006	00017	02228	02029
02490	01941	04442	00073	00264	01235	00226	00027	02568	02129
03330	02191	05892	00083	00294	01455	00316	00047	03118	03239
03570	03451	06992	00103	00444	01925	00336	00057	04448	03339
04680	04441	07112	00153	00494	01955	00466	00077	05238	03449
06060	05471	08222	00223	00504	02345	00666	00107	06868	03519
07850	07581	08352	00333	00884	03575	00716	00217	07778	03669
08230	08881	08882	00483	00914	04785	00886	00477	08828	04449
09110	08981	09912	00713	00124	05555	00896	00777	08918	07339
09360	09121	09962	00923	00224	09865	00966	10117	09998	09999

TOTAL ACCESS NOS. = 1029

Fig. 3—Uniterm wheel cards.

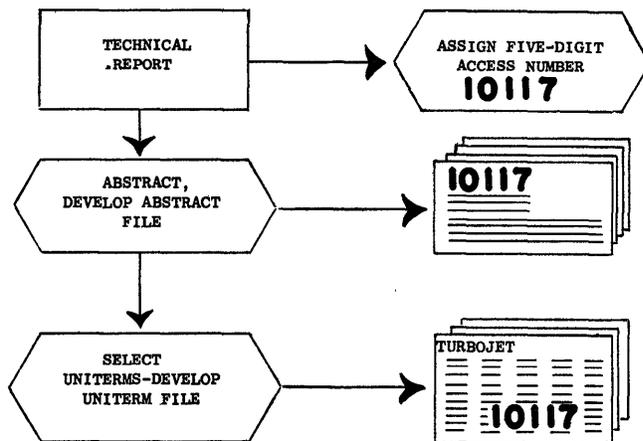


Fig. 4—Flow chart of manual system.

added, this uniterm will start a new record. This will also tend to break one record into two records because this uniterm may fall alphabetically between two uniterms that are in one record (see Fig. 5).

As the machine does the cross coordinating, each search is stored in variable length buckets in memory. The total length of these buckets is also 9000 words. If the amount of information stored in these buckets exceeds 9000 words, the uniterm that caused the overflow

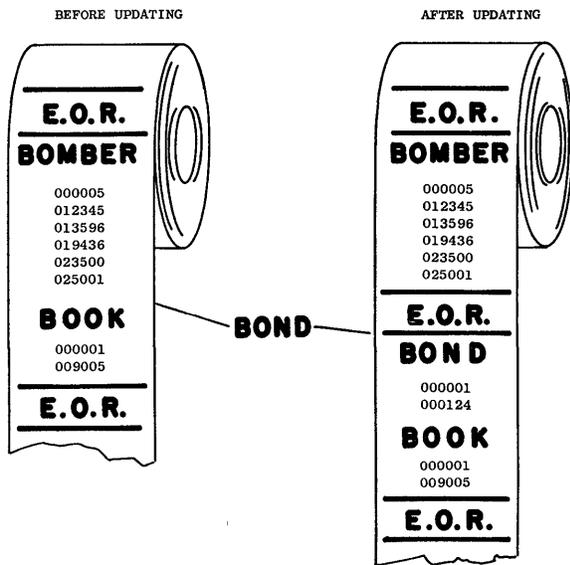


Fig. 5—Updating uniterm master tape.

is stored in a temporary area and the search continues. As more uniterms are read and cross coordinated, the length of the buckets is decreased, permitting the addition of another search at the end of the last bucket. When the master tape has been completely read, the program rewinds the tapes and makes a second pass on the tape using the uniterms from the temporary area. At this point, let me stress that a second pass is only made if there is an overflow on the first pass (usually over 50 searches). In most instances, this will not occur. If on the second pass all uniterms cannot be processed, the program will notify the operators that the search is too large and must be made smaller.

The present system contains about 35,000 abstracts with an average of ten uniterms each, or a total of over 350,000 numbers posted to the master uniterm tape.

Part one will handle 99 searches at once with no limit on the number of uniterms per search. It will also handle unlimited updating. A normal run is about thirty searches and updating of about 2000 access numbers into the general file. The time required is about two minutes for searching and four minutes for updating, or six minutes total. Tape assignments for part one are shown in Fig. 6.

Part two is the printing out of all abstracts which correspond to the access numbers discovered in the first part. One million abstracts have been allowed for in the program. Timing for part two is approximately four minutes per 10,000 abstracts. At the present time, 10,000 abstracts are placed on a master tape in numerical order by access number. A statistical study is now being conducted so that abstracts will be in order by frequency of use. This will significantly improve the timing for part two as we expand our system. Tape assignments for part two are shown in Fig. 7.

Originally it was thought that the program would, in

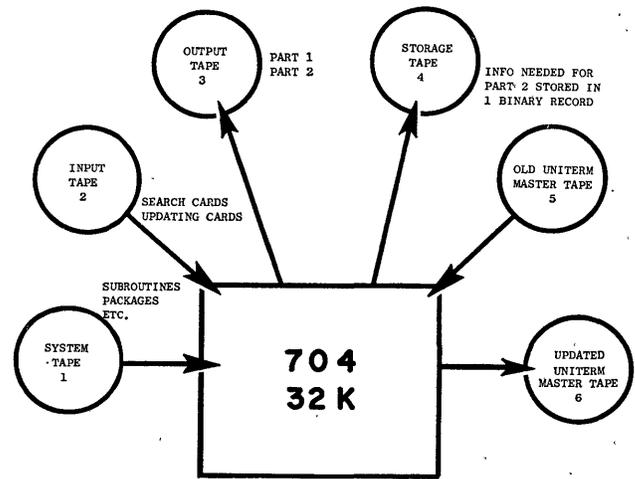


Fig. 6—Tape assignment part 1.

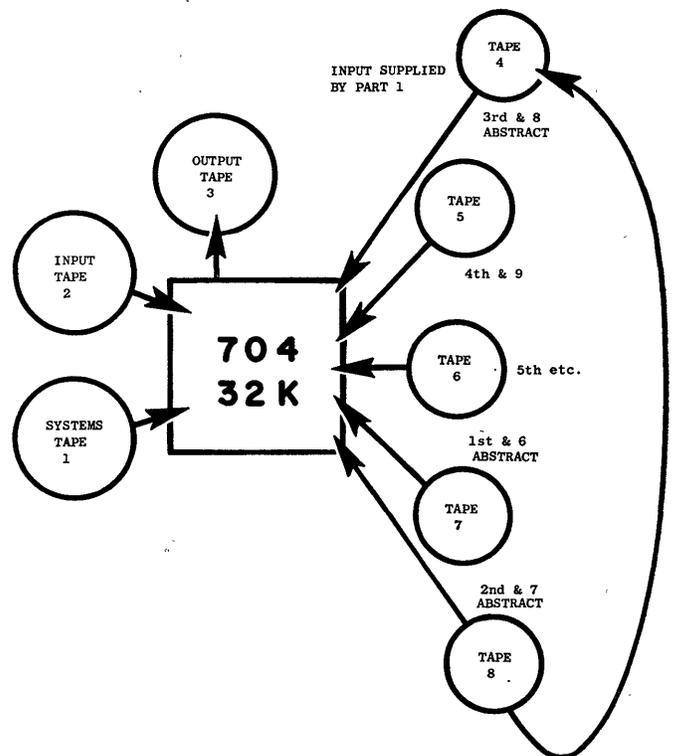


Fig. 7—Tape assignment part 2.

most cases, go right on into part two. Therefore, referring to Fig. 6, it can be seen that the only tapes available at this time are tapes 7 and 8. The first abstract master to be read is found on unit 7, the next on unit 8. While the program is searching tapes 7 and 8, the operator can mount other abstract master tapes on units 4, 5, and 6. After each master tape is searched, another abstract tape can be mounted. A continuous loop is set up selecting tapes 7, 8, 4, 5, 6 — 7, 8, 4, etc. The number of tapes to be read is determined by an input card. All access numbers found in part one are sorted into numerical order before starting part two.

At the beginning of each master tape is an indication of the range of access numbers of that tape. Each number that the program is looking for is compared against this range, and if it is not on this tape, an on-line comment will tell the operator we are finished with this tape. The program will then modify all addresses to pick up the next tape unit in sequence.

The output of this system is in two parts as shown in Fig. 8. At the top is a sample listing of those access numbers located by the system. Below is a set of single sheets of printing. Each sheet contains the abstract corresponding to one access number as shown in the column on the left. Each sheet is pre-addressed for direct mailing. If there is a security classification, it is shown. If for any reason it is considered necessary to suppress printing of security or proprietary information, this suppression is under the control of a sense switch. It is now possible for the requestor to review the abstracts, select those for which he would like to receive the original document, check the appropriate access number on the list on the left, and return this single sheet to the library.

I think that many of the advantages of this system are obvious, among which are speed, cost, designation of security classification, and the direct mailing feature. Advantages that are not obvious are:

- 1) Reduction in amount of human handling with the resultant errors.
- 2) All information is in narrative and is alphabetic. There is no coding.
- 3) Complete abstracts are readily available to the requestor.
- 4) The need for extensive manual files is eliminated.
- 5) No information ever need be removed from the system so no information can be lost.

This system has been in operation in General Electric since September, 1958. Some of the modifications that we have found to be advantageous through experience are:

- 1) In most cases it is desirable to stop the program after part one and examine the print-out before going on to part two.
- 2) It is advantageous to be able to supply to part two the ranges of access numbers wanted on each search.
- 3) An upper limit should be placed on the amount of abstracts to be printed if it is requested that the program continue on into part two without stopping after part one.
- 4) A method of combining several uniterms into one composite term on any individual search is extremely necessary.

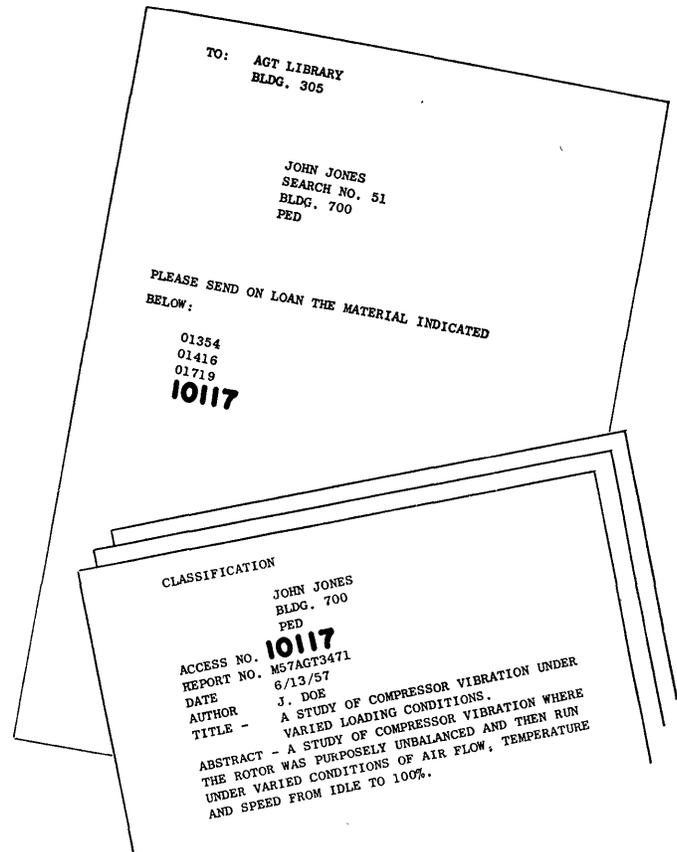


Fig. 8—704 output parts 1 and 2.

- 5) If any uniterm reduces the number of access numbers found to zero, this term is eliminated and the search continues as if this word was not given in this search.

These are only a few of the conclusions we have reached. Future experience, we are sure, will dictate many other additions or deletions to this program. Of course, we are looking forward to the time when with new equipment we will be able to search tapes simultaneously, thereby reducing our running time by a factor of approximately 10 to 1.

We are presently planning to expand this system to mechanize the records involved in checking material into and out of the library. We also plan to develop an automatic overdue notice system.

This information retrieval program has enjoyed wide acceptance in our plant. We have received requests to modify the program to process various types of personnel registers, engine test data files, specialized blueprint files, and various other types of information systems. In any place where the key word coordinate indexing system or some variation of it can be used, this program seems to be the answer to many of our problems.

The Next Twenty Years in Information Retrieval: Some Goals and Predictions*

CALVIN N. MOOERS†

A HISTORICAL PERSPECTIVE

ALTHOUGH information retrieval has lately become quite a fad, I intend in this paper to stand back and take an unhurried look at what is going on, and try to predict where this field must go and what it must do in the future. "Information retrieval" is a name that I had the pleasure of coining only eight years ago, at another computer conference.¹ The name has come a long way since then.

In thinking about a definition of information retrieval and in considering the future of this field, we must take an evolving view. At the present time, information retrieval is concerned with more than the mere finding and providing of documents; we are already concerned with the discovery and provision of information quite apart from its documentary form. To encompass future developments, as we shall see, even this broad view of information retrieval will have to be modified and extended.

When we speak of information retrieval, we are really thinking about the use of machines in information retrieval. The purpose of using machines here, as in other valid applications, is to give the machines some of the tasks connected with recorded information that are most burdensome and unsuited to performance by human beings. At all times, it is important to remember that it is the human customer who uses the information-retrieval system who must be served, and not the machine. It makes a difference who is served, and this little matter is sometimes forgotten in computer projects.

To get a historical perspective of the introduction of machine methods to information retrieval, let us look back over a bit of history. I think that it can be said that the introduction of machine methods has followed the realization of a need, backed by pressure and means to do something about the need. Thus, although quite powerful mechanical methods could have been de-

veloped by the technology of the Hellenistic Era for the Library of Alexandria, other methods of retrieval, presumably based upon human memory, and the making of lists, were apparently considered quite satisfactory. The simple, though powerful, mechanical technique that could have been used at Alexandria is the method of perforated stencils invented by Taylor in 1915, which has sometimes more recently been called "peek-a-boo."² The British inventor Soper in 1920 patented a device which was an improvement upon Taylor's perforated stencils, and Soper described the use of his mechanism for information retrieval employing some truly advanced conceptions.³

Much more attention, however, has been given to the development of devices for scanning and selecting upon film strips. This work was apparently spurred by the perfection of motion picture films and cameras. Goldberg in 1931 patented one of the earliest film-scanning and photographic-copying devices.⁴ Independently, Davis and Draeger during 1935, in the early days of the American Documentation Institute, in connection with their pioneering work in microfilm documentation, investigated the feasibility of a microfilm scanner using a decimal coding.⁵ Apparently stimulated by reports of this work, V. Bush and his students at M.I.T. in 1938-1939 built perhaps the first prototype machine along these lines, a microfilm scanner with each frame of text delineated by a single decimal code for the subject, and a photoflash copying method. However, they were unable to interest any commercial or governmental organization in the device, and wartime distractions intervened soon thereafter, so the project was dropped. Two more "rapid selectors" based upon these same general principles have been built,^{6,7} but for various reasons neither of them has operated in a fashion that is consid-

* This work has been supported in part by the U. S. Air Force Office of Sci. Res. through Contract No. AF 49(638)-376. All opinions are those of the author.

† Zator Co., Cambridge, Mass.

¹ C. N. Mooers, "The Theory of Digital Handling of Non-Numerical Information and its Implications to Machine Economics," Zator Co., Cambridge, Mass., Tech. Bull. No. 48, 1950; paper presented at the March, 1950 meeting of the Association for Computing Machinery at Rutgers University, New Brunswick, N. J.

C. N. Mooers, "Information retrieval viewed as temporal signaling," *Proc. Internat. Congr. of Mathematicians*, Harvard University, Cambridge, Mass., vol. 1, p. 572; August 30-September 6, 1950.

² H. Taylor, "Selective Device," U. S. Patent No. 1,165,465; December 28, 1915 (filed September 14, 1915).

³ H. E. Soper, "Means for Compiling Tabular and Statistical Data," U. S. Patent No. 1,351,692; August 31, 1920 (filed July 23, 1918).

⁴ E. Goldberg, "Statistical Machine," U. S. Patent No. 1,838,389; December 29, 1931 (filed April 5, 1928).

⁵ R. H. Draeger, "A Proposed Photoelectric Selecting Mechanism for the Sorting of Bibliographic Abstract Entries from 35 mm Film," Documentation Inst. of Science Service, Washington, D. C. (now American Documentation Inst.), Document No. 62; July 27, 1935.

⁶ R. Shaw, "Rapid selector," *J. Documentation*, vol. 5, pp. 164-171; December, 1949.

⁷ Anonymous, "Current Research and Development in Scientific Documentation," National Science Foundation, Washington, D. C., Rep. No. 3, p. 27; 1958.

ered generally acceptable, and neither is currently in actual use. At the present time, much attention is focused upon the Eastman Minicard machine, a cross between a rapid selector and a Hollerith punched-card machine.⁸

Card-sorting devices, such as those based upon the Hollerith card (the card used by IBM), as well as those based on other cards such as Perkins' marginally punched card, were recognized at an early date to be far too slow to cope with problems such as those of *Chemical Abstracts*. Within the past few years, there have been a number of instances of the use of electronic computing machines to perform information retrieval. As computing machines are presently designed, they are not matched to the job of information retrieval—they can do it, though not efficiently—and the situation of using a computing machine for this job is like using a bulldozer to crack peanuts. Oftentimes, if the information collection is small enough to allow the problem to fit upon the computer, there are easier methods to perform retrieval. If the collection is large, it does not have to be very large to tie up all the computer's memory capacity. It is clear that special computer-like devices will be called for if we are to perform efficient large-scale information retrieval.

Although we have been trying to build high-speed selecting machines for information retrieval over the past twenty years, since the date of Bush's machine, at the present time I do not think that it can honestly be said that we have done too well. We do not really have a machine which is an altogether happy answer to the problems of search and selection on collections ranging in size upwards from fifty or one hundred thousand items. The problem becomes even more unmanageable at the million point, since this size of collection requires reasonably high-speed processing and decision on a scanned record of something like 10^9 bits.

However, the hardware will be built—and is being built. But what about the classification terminology, the subject headings, the descriptors, and the like? One after another, various machine projects have foundered on this problem, especially those projects that have copied library classification decimal systems or made use in a detailed way of their indexing techniques. We should appreciate that new mechanisms deserve new methods, and that there is a consensus of opinion (although it is not unanimous) that the method of putting together independent idea-expressing terms and selecting upon their correlative occurrence constitutes the desired point of departure from the historic methods of the library.

A highly developed form of this point of view is the method of "descriptors," which was introduced and de-

veloped in theory in 1948–1950 in a number of papers in conjunction with a mechanical card selector.⁹ The descriptor method, which makes a great point of employing precisely defined terms composing a limited vocabulary, is a refinement of a number of earlier practices. The method was implicit in the work of Soper, it was toyed with and dropped by the librarian Bliss, and it was used in one fashion or another by a number of scientists and chemists with Perkins cards in the 1940's, e.g., by Bailey, Casey, and Cox.¹⁰ People seem to confuse descriptors with Uniterms. The latter might be described as a crude form of a descriptor system, originally making use of words lifted from titles and texts. The Uniterm approach, since it was introduced in 1951, seems however to be migrating both in concept and usage towards the descriptor methods, as is clear from many reports coming from projects where they claim to use Uniterms.

The problem of classification terminology or language symbols for machine retrieval is well toward a solution, even for complex and structured kinds of information. An example is the work on the coding of chemical structures for machine retrieval.^{11,12} However, it should be noted that considerable work on retrieval of structured information, especially for chemical compounds, has sometimes resulted in symbolism that is not completely suitable for machine use, as for example some of the methods considered by the International Union of Pure and Applied Chemistry.

THE PRESENT STATE OF AFFAIRS

Although we may soon have suitable machines for large-scale information retrieval and although the situation with respect to the language symbols of retrieval is in a reasonably satisfactory state (that is, ahead of the machines) we are not yet finished with our problems.

Presuming that we have a machine completely capable of dealing with a collection of one million—or even a hundred million—items, who will read these items or documents and assign the descriptors? Experience has shown that this is a difficult and time-consuming job. For example, in my experience in reading and coding patents, it takes me about fifteen minutes of reading, on the average, merely to figure out what the inventor is driving at. The Patent Office has some three million of such patents.

This is exactly the kind of burdensome job that should be turned over to the machine. In fact this problem is under active consideration and study in a number

⁹ C. N. Mooers, "Zatocoding and developments in information retrieval," *ASLIB Proc.*, vol. 8, pp. 3–22; February 1956. This paper summarizes these developments.

¹⁰ C. F. Bailey, R. S. Casey, and G. J. Cox, "Punched-cards techniques and applications," *J. Chem. Ed.*, vol. 23, pp. 495–499; 1946.

¹¹ C. N. Mooers, "Cipherring Chemical Formulas—The Zatopleg System," Zator Co., Cambridge, Mass., Tech. Bull. No. 59; 1951.

¹² L. C. Ray and R. A. Kirsch, "Finding chemical records by digital computers," *Science*, vol. 126, pp. 814–819; October 25, 1957.

⁸ A. W. Tyler, W. L. Myers, and J. W. Knipers, "The application of Kodak Minicard system to problems of documentation," *Amer. Documentation*, vol. 6, pp. 18–30; January, 1955.

of places. It is not an easy task to give to a machine. It contains a great many aspects that would seem to require the exercise of real "intelligence." Fortunately, we already have one remarkable accomplishment which shows that this seemingly intellectual job is not completely incompatible with mechanization. I speak of the work by Luhn on his "auto-abstractor."¹³ By the method of Luhn, the computer takes in the text of an article, statistically picks out the unusual words, and then chooses sentences containing these words to make up the auto-abstract. If this process were terminated at the point of picking out the words, we would have Uni-terms. If the words picked out in this fashion could be replaced by standardized words having approximately the same meaning, that is, if the synonyms could be eliminated, then we would have descriptors. It should be noted that this kind of treatment of synonyms, which has been going on in retrieval for some years, has lately been given the fashionable name of "the thesaurus method." In the interests of precision in terminology, I should like to point out that there are significant differences in Roget's concept of a thesaurus and the set of equivalence classes of terminology that are required for retrieval. Indeed, this is precisely why I introduced the new terminology "descriptor" some years ago, that is, to give a verbal handle for a group of new conceptual methods with language symbols.

Such a take-off on Luhn's method would not be the final answer, because as Luhn has set it up, the machine is operating in an essentially brainless fashion. To do better than merely picking up words on a statistical basis, we would have to build into the method the capability of handling the equivalence classes of words and phrases. This gets us into language translation. After the statistical approach has segregated words of high import from the text, we need to translate these words into the standardized descriptor terminology for further retrieval. However, even building up the equivalence classes of the terminology is a burdensome job, and this too should be turned over to the machine. Not only should the machine build up these equivalence classes, but it should be made to refine its performance with respect to using these terms and getting the descriptors, and it should even be made to learn how to improve its performance.

Fano and others have suggested the use of statistics on the way people come in and use the library collection in order to provide feedback to help a machine improve its performance.¹⁴ While the suggestion is in the right direction, I think that this kind of feedback would be a rather erratic source of information on equivalence

classes, because people might well borrow books by Jack London and Albert Einstein at the same time. While this difficulty can be overcome, there is a more severe problem. Any computation of the number of people entering a library and the books borrowed per day, compared with the size of the collection, shows, I think, that the rate of accumulation of such feedback information would be all too slow for the library machine to catch up to and get ahead of an expanding technology.

In this respect, it is my speculation that a more powerful source of educational material for a machine is already available, and it should be tapped. Despite the admitted limitations of such material, the subject entries, the decimal classification entries, and the other content typed on catalog cards contains a great deal of ready information that can be used in teaching a machine how to assign descriptors to documents. Other collections, besides those in the libraries, also often provide a ready source of classificatory information that should be tapped. For instance, in the Patent Office, in each case record of each application for patent, there is a great amount of specific reference to other related patents, and this information, along with the assigned class numbers, is readily available for machine digestion without further high-level human intellectual effort.

In order to do these things, we shall need a machine with some rudimentary kind of "intelligence;" or more accurately, we shall need an "inductive inference machine" in the sense used by Solomonoff.¹⁵ An inductive inference machine is one that can be shown a series of correctly worked out examples of problems, that can learn from these problems, and that can then go ahead on its own (probably with some supervision and corrective intervention) to solve other problems in the same class. While an inductive inference machine can be quite capable at a given class of jobs, it need not have "brains" or "intelligence" in the general sense.

As I mentioned before, putting the descriptors on the documents—that is, delineating the information in the text by symbols for retrieval—is a form of crude language translation. It is crude because the machine does not need to worry about grammar in the target language, since the grammar of descriptors is nonexistent, or at most, is rudimentary. As I see it, machine translations of this kind for the purposes of information retrieval will be an area of early pay-off for work in inductive inference machines.

If inductive inference machines can be built at all, then it certainly should be possible for us to feed them with subject headings and classification numbers on the one hand, and with the titles of book chapters and section headings on the other hand, in order to teach the machines how to do at least some rudimentary kind of job of library subject cataloging. With librarians at

¹³ H. P. Luhn, "The automatic creation of literature abstracts," *IBM J. Res. Dev.*, vol. 2, p. 159; April, 1958.

¹⁴ R. M. Fano, "Information theory and the retrieval of recorded information," in "Documentation in Action," J. H. Shera, A. Kent, and J. W. Perry, eds., Reinhold Publishing Corp., New York, N. Y., ch. 14-C, p. 241; 1956.

¹⁵ R. J. Solomonoff, "An inductive inference machine," 1957 IRE NATIONAL CONVENTION RECORD, pt. 2, pp. 56-62.

hand to provide suitable intervention or feedback, the machine's performance should improve, and by further work, even the categories or descriptors which are used can be improved by the machine. Thus I feel that we can expect in the next few years to see some interesting results along this line.

GOALS FOR THE NEAR FUTURE

There are a number of other applications of machines for purposes of information retrieval of a kind that have not yet been seriously undertaken, and others that have not yet been considered. In my discussion I shall bypass treating such useful and imminent tasks as the use of machines to store, transfer, and emit texts, so that at the time that you need to refer to a paper, even in an obscure journal, you can have a copy in hand within, say, twenty-four hours. Neither shall I consider the application of machines to the rationalization and automation of library ordering, receiving, listing, warehousing, and providing of documents. Neither shall I consider the application of machines to the integration of national and international library systems so that at any first-rate library, you will have at your command the catalogs of the major collections of the world. These are all coming—but it should be noted with respect to them that the problems of human cooperation ranging from person-to-person to nation-to-nation cooperation are more serious than some of the machine and technical problems involved.

The first of the rather unusual applications of machines to information retrieval that I want to talk about can be introduced as follows. When a customer comes to an information retrieval system, he comes in a state of ignorance. After all, he needs information. Thus, his problem of knowing how to specify pieces of information that are unknown to him is a severe one. For one thing, the vocabulary of the retrieval system, and the usages of the terms in the system, may be slightly different from the language that he is used to. For another thing, upon seeing some of the information emitted according to his own retrieval prescription, he may decide that an entirely different prescription should be used. In short, the customer definitely needs help in using a machine information retrieval system, and this help should be provided by the machine.

An indication of what kind of system needs to be provided, and how it can be done, is given by certain of the simple sorted-card retrieval systems. Some of the sorted-card systems do very well in this respect, others do not. It has been common practice in Zatocoding systems, which use a simple schedule of a few hundred descriptors, to employ a descriptor dictionary system having many cross-references from words in the ordinary technical usage to the appropriate descriptors.⁹ Thus the customer can find his way into the system starting out with his own terminology. After the customer is referred to a descriptor, he finds there a carefully drafted scope note explaining the range of meaning attached to the

particular descriptor. In another tabulation in the descriptor dictionary, the descriptors themselves are grouped or categorized into fifteen or twenty groups, and each group is headed by a question pertinent to the descriptors under it. Thus, under a question "Are geometrical shapes involved?" would be found descriptors such as "round," "square," "spherical," etc.

These simple card systems provide another source of assistance to the customer because they are able to emit cards within a minute or less from the time the retrieval search is begun. Thus if the search is headed into the wrong direction, the customer, upon looking at the cards or documents, will immediately detect this fact, and can reframe his request to the machine before any further searching is done. It is deplorable, but true, that many contemporary proposals for machine systems may be so slow in providing feedback that the feedback time is measured in hours or days, with the consequent waste of machine sorting time and accumulation of human frustration.

The problems of customer assistance are going to be severe with the large machine retrieval systems of the future, and these problems must be faced. The descriptor vocabularies are going to be large. Another possibility is that some of the machines will operate internally on vocabularies or machine code systems that are quite unacceptable for external communication to the human operators. There has already been some successful experimentation with symbol systems of this kind in coding chemicals.^{11,12} Such symbol systems work beautifully inside the machines, but people should not be forced to use them. For these reasons, in order to translate the customer's requests into forms suitable for the machine, machine assistance is going to be desirable. Holt and Turanski see this problem of processing the customer's request at the input to the machine as being very similar to the presently developing customer use of automatic programming for mathematical problems. The more advanced systems of automatic programming provide for a succession of stages of translation, with the symbolism at each stage moving further and further from the human word input to the abstract symbols and the detailed machine orders required for internal operation of the machine. In mathematical programming, the machine programs itself, and then carries out the program. In retrieval programming, the machine will form the proper machine prescription, and carry out the search. To my mind, there is an important difference. In retrieval, the machine should check back with the customer as it builds up the prescription in order to make sure that the search will be headed in the right direction; then it should search a sample of the collection and check again to make sure that the output being found is appropriate to the customer's needs. If we are to have larger and more complex machine retrieval systems, we must come to expect a great deal of back-and-forth man-machine communication during the formulation of a search, and as it is going on.

Quite another approach to handling the customer's input problem is advanced by Luhn, who suggests that the customer write a short essay detailing what he thinks is descriptive of the information he wants.¹⁶ The essay text would then presumably be handled in the fashion of the auto-abstract method (though Luhn is a little sketchy here on the details of his proposal), and the words selected from the short essay would be compared with words similarly selected from the document texts. When there is a sufficient degree of similarity, selection occurs. Although the Luhn proposal does put the load of translation of the customer's request upon the machine, it does not provide for customer guidance into the resources of the machine's selective language possibilities, or into the resources of the collection. Help in both of these directions would surely be of great assistance to a customer in extracting the maximum value from information in storage.

Another possibility is to use an inductive inference machine, because it is open to learning a great variety of tasks. It would be able to provide a generalized approach to the problem of customer assistance. But, however customer assistance is provided, I think it is safe to predict that we must build information retrieval systems with the planned capability to communicate back and forth with the customer so that he can better guide the machine in retrieving what will be useful to him.

RETRIEVAL VIEWED AS A PROCESS OF EDUCATION

If the machine aids the customer by guiding him in the use of the retrieval system, the machine is necessarily educating the customer. Let us take this viewpoint, and look upon a machine retrieval system as an educational tool. This viewpoint provides a number of new tangents to consider. We have seen how the customer can use some coaching by the machine in order to tap efficiently the information resources during the search process. But, as anyone knows who has had a large batch of documents sent his way, maybe the customer can also use some machine help in reading the mass of documents emitted from a retrieval system!

It is my prediction that some of the machine information retrieval systems of the future will go considerably beyond the tasks of mere retrieval or citing or providing document texts. I believe that some of them will also help the customer assimilate or read the output provided by the machine. This prediction is not at all fanciful, even though it is yet quite a way into the future. How far into the future it is we can only guess, or estimate by recalling that the Minicard follows a full twenty years after the first suggestions for a film selector, or that the widespread use of descriptors came about forty years after Taylor actually used something very much like them in information selection.

Machines can be very effective in teaching human beings. This is shown by the work of Skinner at Harvard where, in recent experiments, written modern languages and college mathematics have been set up on machine lessons.¹⁷ Essential to the process is rapid feedback, or communication between the machine and the human learner, so that the human knows immediately that he is on the right track, and so the machine can apply corrective action as soon as errors appear. Skinner's machines at present employ written materials prepared in advance by human beings, the machine performing on the basis of a fixed internal sequence of morsels of information of graded difficulty. However, machines need not be restricted to doing their teaching according to a preset sequence of lesson elements of this kind. In the same way that we are currently looking for techniques to allow machines to assign descriptors from texts, so can we contemplate the development of teaching procedures and machines whereby the machines by themselves will be able to pick out a graded sequence of information morsels from the documentary record retrieved and will then present them to the human customer.

Taking this view of a machine information center acting both as a retrieval device operating upon a store of information and as a teaching device for the human customer, we can see that the process of input request formulation and the process of giving out information will merge into a sustained communication back and forth between the customer and the machine. Of course, once the customer is on the track of documents containing information particularly pertinent to his interests, he will very likely desire to see the original text. This can be done, and a customer will have a choice of how much or how little of any particular actual document he wishes to read directly.

The range of future possibilities is even greater when these ideas are combined with the possibilities inherent in mechanical language translation devices. Of course, we should expect that future information centers will be able to provide translation from one ethnic language to another of the texts that the retrieval system provides. Let us look further. As is well known, one of the problems in machine language translation is to provide sentences in the target language in the required form—that is, to provide a smoothly running, colloquial translation. For example, in going from German to English, we must rescue the verbs from the end of the German sentence and put them up where they belong in the middle of the English sentence. Any machine capable of doing a high-grade language translation must be able to arrange and rearrange idea units and word units to make acceptable text out of them. This being the case, it is reasonable to predict that the information morsels that a teaching machine would put out could be given as in-

¹⁶ H. P. Luhn, "A statistical approach to mechanized encoding and searching of literary information," *IBM J. Res. Dev.*, vol. 1, p. 309; October, 1957.

¹⁷ B. F. Skinner, "Teaching machines," *Science*, vol. 128, pp. 969-977; October 24, 1958.

put to a machine technique patterned on the output half of a language translator. The resulting textual output would be in the nature of a written article having at least some degree of acceptable style.

This means that you could go to an information center, describe a certain kind of information needed, have the machine assist you in making your request more definite, and then order it: "Provide me with an 800-word article, not requiring more than an undergraduate chemistry background, on the deterioration of polymers by sunlight." After a short, but decent, interval, the machine would come forth with such an essay.

There is an important corollary to this notion of the machine central being able to provide essay articles upon request. We are all aware of the considerable duplication of information found in the technical literature. The same, or very similar, piece of information is repeated in one article after another. Many articles are summaries of other articles, or are derivative upon other articles, and provide little or nothing that is new. If the output from an information system does not need to be in the form of a graphic image of the original text, or a type-out of the text itself, then it is possible to consider the storage of new information only in the machine. A machine could store facts alone, and only new facts; it would not store text. By eliminating the dependence upon the original text, and avoiding the duplication of the same information written over and over, it might be possible to secure considerable increase in the machine's storage capabilities.

Yet there are problems of a kind that will occur to any thoughtful individual. I do not think we want to throw away the original record which we have already—the printed books and articles in our libraries. Neither am I sure that we want to give up entirely our system of printed publication. But, putting these problems aside, let us do some more speculating. It might be possible for the scientist in his laboratory to feed his raw (or nearly raw) results directly into a machine for computation, checking for acceptability, correlation with earlier facts, and ultimate storage. Thus, instead of a scientific archive existing almost solely on paper, as we now have, it is possible that a part of the archive in the future will be in machine form. The only way that such a machine archive would be tapped would be by having the machine write a summary or article upon specific request.

When we are thinking about information machines of this kind, I wish to stress that we should not think in terms of some big single machine central. This is important. It would be foolish and expensive to build up

a single central "bottleneck." If such machine central information systems as I describe will be at all possible, they will be important enough to be set up at a large number of installations, quite in the same way as we now are making use of a large number of electronic computer installations. There will be both large and small information machines. Some of these machines will be in intercommunication with each other, while others will operate in isolation. At various times, the machine memory from one or several of the machines can be played out onto tape, and the tape record, containing a vast amount of information, can be incorporated into the memory systems of many other information centrals.

If machines can store and correlate laboratory facts, and can communicate with laboratory workers, we shall have to expect that the machines will find gaps in the information as a part of the correlation, and they will point out to the laboratory workers the need for further experimentation in certain areas. How far we can expect this kind of active feedback to extend is hard to guess. The present work with pattern recognition will ultimately lead to a kind of a machine eye, and we already have machine hands for the handling of radioactive materials. An information central machine system, aided by such receptors and effectors, would become, in effect, a laboratory scientist.

At this point I would prefer to terminate my speculations on the excuse that we are now perhaps more than twenty years into the future, the limit that I set for myself in this paper.

In summary, I think that it can be said that mechanical information retrieval has started rather slowly; it has taken from about 1915 or 1920 until now to become as popular as it is. At the moment, except for certain highly integrated small retrieval systems, we are yet only dabbling in the subject. We do not now honestly have any appropriate large-scale machine for collections involving millions of items. We are only beginning to get a widespread recognition of the capabilities of suitable retrieval language systems, and there still remains the problem of getting machines with internal digital operations that are as suitable for retrieval and information work as the operations of addition and multiplication are suitable for mathematical work.

In any event, it is useful for us to know what some of our future targets are likely to be. With such knowledge, we will be in a better position to steer our activities in the present. This is the excuse for the predictions—which I take very seriously—that are contained in this paper.

Simulation of an Information Channel on the IBM 704 Computer*

E. G. NEWMAN† AND L. O. NIPPE†

INTRODUCTION

EXPRESSIONS for the probabilities of multiple error patterns in symbols consisting of N bits are needed to evaluate the effectiveness of applying error correcting codes to these information channels. For example, if a Hamming¹ single error correcting code is applied to the information, the probabilities of all errors greater than one should be established. This will indicate the number of times the single error correcting code has failed.

In addition, if the effects of channel asymmetry and regional error dependence on the probabilities of multiple errors are known, this information can be used to develop channel characteristics leading to the minimum number of errors.

Expressions for error probabilities have been obtained for a binary symmetric channel² and for an asymmetric channel [see (2) or (6) in the Appendix] when errors are assumed to be independent. Attempts to introduce error dependence were made by using relationships between stochastic processes and familiar network concepts.^{3,4} All results, however, for even relatively simple cases became too unwieldy to be useful. Therefore, a simulation program which gives approximate values for the desired error probabilities was developed.

Many practical information channels exist which exhibit both error dependence and asymmetry. An example of such a channel is a magnetic tape channel. A symbol of N bits can be recorded on N parallel tracks (Fig. 1). The error dependence is primarily the result of defects extending over certain regions of the tape which influence the correct transmission of successive bits in a particular track or in a number of adjacent tracks.

A single defect may produce errors on a single track only (Defect I, Fig. 1), or larger defects may cause errors in a number of adjacent tracks (Defect II, Fig. 1). These large defects produce multiple errors for all consecutive N -bit symbols which fall into this defective region. Multiple errors can also be produced by the simultaneous occurrence of different defects, each affecting the information in a particular bit of an N -bit symbol (Defects I and III, Fig. 1).

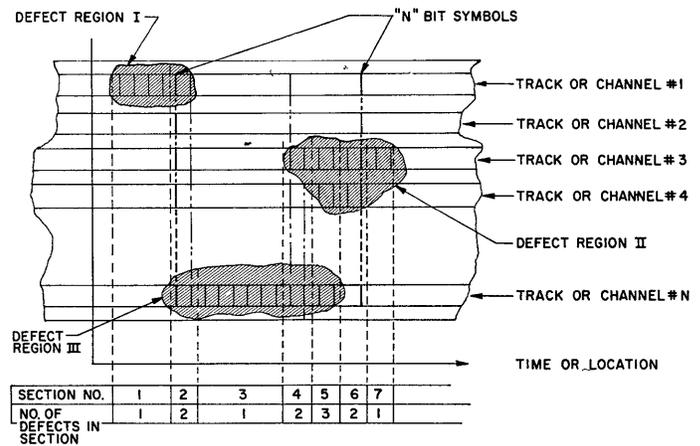


Fig. 1—Effects of regional defects on the generation of multiple errors in an N -bit symbol.

Depending on the type of recording used, certain types of defects may cause errors in the transmission of ones without affecting the transmission of zeros. Other types of defects or noise bursts affect the transmission of zeros only. Some defects can produce errors in both ones and zeros. These phenomena lead to channel asymmetry. The degree of this asymmetry is governed by the distribution of the various defect types.

For example, for a modified nonreturn to zero type of recording (Fig. 2), a one is recorded by changing the magnetization of the tape from one saturation level to the other. A zero leaves the magnetic tape at a previously established saturation level. As the read head senses only a change in the flux linking it, only ones produce signals.

Tape defects in the form of high spots in the surface of the magnetic coating or loose particles of oxide material produce a head-to-tape separation, which results in a loss of signal. This affects the transmission of ones only. The characteristic loss in signal amplitude is shown in Fig. 2. This phenomenon is not unlike the amplitude "fading" of radio signals and may, at high recording densities, produce a number of consecutive errors.

Other tape defects, such as pin holes in the magnetic coating, can produce errors in the transmission of zeros (Fig. 3). At the boundaries of the hole in the magnetic coating, a flux change will be sensed by the read head as it moves from a region free of magnetic particles to one where magnetized magnetic particles exist. This change of flux could be sensed as a one in place of a zero. In addition, amplifier noise can occasionally exceed a certain clipping level. If this noise peak occurs at the

* This work has been supported by the U. S. Dept. of Defense.

† IBM Product Dev. Lab., Poughkeepsie, N. Y.

¹ R. W. Hamming, "Error detecting and error correcting codes," *Bell Sys. Tech. J.*, vol. 29, pp. 147-160; April, 1950.

² L. Brillouin, "Science and Information Theory," Academic Press, Inc., New York, N. Y., ch. 6, pp. 62-70; 1956.

³ W. H. Huggins, "Signal flow graphs and random signals," *Proc. IRE*, vol. 45, pp. 74-86; January, 1957.

⁴ S. J. Mason, "Feedback theory—some properties of signal flow graphs," *Proc. IRE*, vol. 41, pp. 1144-1156; September, 1953.

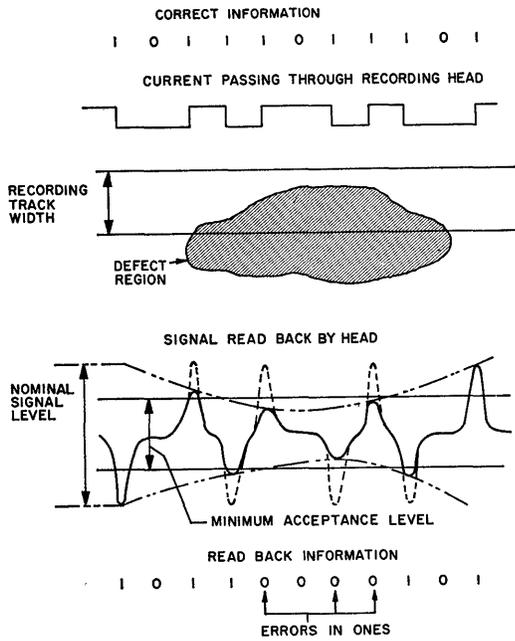


Fig. 2—Effect of tape defect causing head-to-tape separation on signal amplitude (modified nonreturn to zero recording).

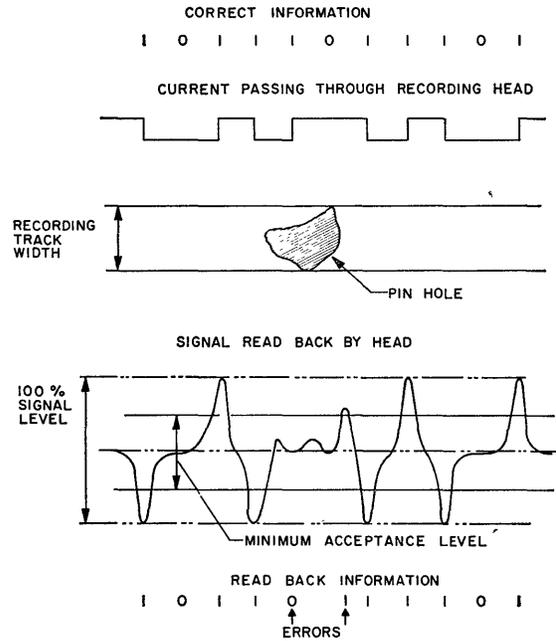


Fig. 3—Effect of defect in magnetic oxide (pin hole) on signal amplitude (modified nonreturn to zero recording).

time when a zero should be read, it may be sensed incorrectly as a one.

If the hole is large, ones recorded in this region will, of course, not be sensed. A hole in the magnetic oxide or a nonmagnetic particle in the magnetic coating can, therefore, produce errors in both ones and zeros.

These are some typical examples of the error-producing mechanisms which cause the complex characteristics of the magnetic tape channel.

CHANNEL SIMULATION

Because of the difficulty of obtaining analytic expressions for the probabilities of multiple error patterns in terms of the complex characteristics of a tape channel, simulation means had to be developed.

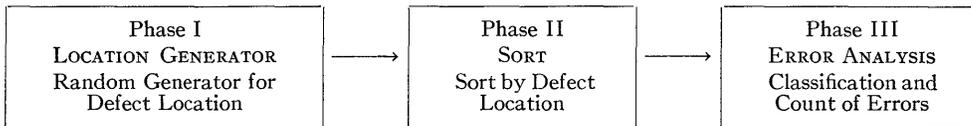
The simulation procedure can best be described by comparing it to the manufacturing of tape. The defects

single track or at the most a few adjacent tracks. A theoretical description of the channel characteristics can also be used. The program is completely flexible, that is, tape of any specifications can be “manufactured.”

Random information could now be “recorded” over the whole length of the hypothetical tape and errors introduced in accordance with the defect type. The program, however, concerns itself only with errors. Thus, information is placed only into the defective regions and the resulting errors are classified and counted. This results in a considerable saving of computer time.

THE 704 PROGRAM

The 704 Information Channel Simulator Program (Fig. 4) might best be considered by breaking the program into three major phases as indicated below:



in the tape, produced during the manufacturing process, can be considered to be distributed in a random manner over the length of the tape. The program simulates this process by assigning random locations to each of the defect regions, as they are read from a list of inputs. This list gives the number and class of all the various defects which must be placed on a particular length of tape. This defect listing can usually be prepared on the basis of error statistics obtained from tests involving a

Phase I might be called the “tape manufacturing” phase because it is responsible for the actual placement of defects on the hypothetical tape. All defects are assumed to be placed on the tape simultaneously and previous to the analysis phase.

Since the defect locations are generated randomly, it is next necessary to sort (Phase II) the defects by location so that they may later be processed in sequence.

Phase III performs the random generation of infor-

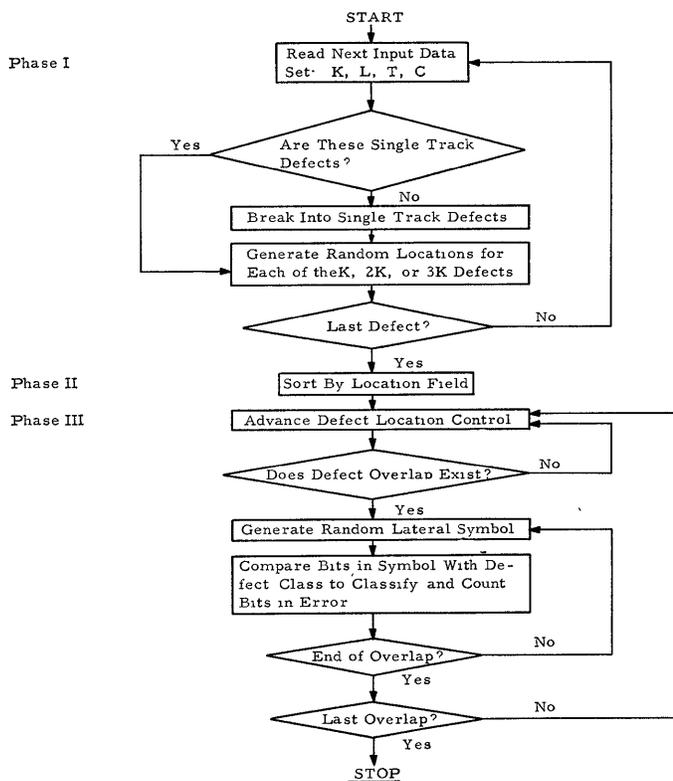


Fig. 4—General flow diagram.

mation to be written on the tape, the analysis of the influences of each defect on this information, and the classification and counting of resulting errors.

INPUT DATA

Before discussing any of the three phases of the program in detail, it is necessary to have an accurate picture of the input data, which are prepared on the basis of actual test data or from a theoretical description of the channel characteristics.

To formulate the input data for the program the following statistical information is needed:

- 1) The *a priori* probabilities that defects of a particular length will occur on a track. These defects can produce errors in ones or zeros or both on this track only.
- 2) The *a priori* probabilities that "centers" of large defects of a particular size will occur on a particular track. These defects can produce errors in ones or zeros or both in a region which encompasses a number of adjacent tracks.

For program use, the above information is converted into a set of values K , L , T , and C , where

- K = the number of defects per track of a certain type to be placed on the hypothetical tape,
- L = the length of the defect in bits,
- T = the track number of the defect center, and
- C = the defect class, and indicates whether the defect involves errors in ones, zeros, or both, and

whether one, two, or three adjacent tracks are involved. For the defects which produce errors in more than one track, the same defect length per track was assumed.

The above set of values is defined as an input data set. A different set may be used for each track.

Phase I—Location Generator

Each input data set read per track represents K defect areas. In order to assign a random longitudinal location X to each, a random number modulo M is generated. The pseudo random number generator program used here is PE RAND. It produces a 35-bit random number by multiplying two odd, 35-bit numbers, selected from a group of ten such numbers, to produce a 70-bit product. The center 35 bits of this product are used as the random result and may be divided by a previously specified number, M , to make the result modulo M . This generator has been thoroughly tested. The probability that any bit of the resulting random number is a one lies between 0.45 and 0.55. This was considered entirely satisfactory for this application.

This defect location just generated is stored along with the appropriate length, track, and class. Any given defect may involve one or more adjacent tracks as specified by its class. Arbitrarily, the program was written to handle defects extending over no more than three adjacent tracks. The program converts these adjacent track defects to individual track defects of equal length and assigns the same location to each.

Phase II—Sort

After a random location has been generated for each defect, these defects are sorted by location fields. There are two general sort programs available for the 704 through the SHARE organization. Either may be used, or the sorting may be done on another machine.

Phase III—Error Analysis

It can be seen from the flow diagram of Fig. 4 that there are three basic operations other than decision making that must be performed by this phase. They are

- 1) Comparison of defect locations,
- 2) Random generation of information in the form of N -bit symbols, and
- 3) Classification and counting of errors within the affected N -bit symbols.

As each defect is read from the sorted list, its location must be compared with the location of other defects to see if any of the defects overlap. When defects on different tracks overlap, multiple errors may result. When defects on the same track overlap, they are joined into one defect.

When the number of overlapping defects changes, a new "section" is formed (Fig. 1). The program keeps track of the number of overlaps involved at any instant.

The number of bits involved in these overlapping defects represents the maximum number of errors which can occur in the symbols in the overlap. For example, consider Section 2 in Fig. 1. This particular section is two bits long and involves Defects I and II. Thus, no more than four bits may be in error in this particular overlap, two bits in each of two N -bit symbols. The program provides three counters for double, triple, and higher order overlaps.

Next, the program must generate random information for each N -bit symbol in the overlap. Considering each bit individually, it will be in error if any one of the following three conditions exists:

- 1) The bit is a one and the defect produces errors in ones,
- 2) The bit is a zero and the defect produces errors in zeros,
- 3) The defect produces errors in ones and zeros.

A test for errors is now made on the basis of these three conditions. The previously described PE RAND program is used to generate the random information.

Each bit in the affected N -bit symbol is tested, and, if an error has occurred, the appropriate error counter is updated. The program provides counters for double and triple errors

MACHINE TIME ON THE IBM 704 COMPUTER

Table I indicates approximate machine times per phase for typical runs involving 3500 and 10,000 defect regions, respectively. It shows that the time required to run Phases I and III is linearly related to the number of defect regions involved, while the time for Phase II (Sort) increases more rapidly with the number of defects. This indicates that a larger number of runs involving fewer defects will give shorter over-all machine times.

TABLE I
704 MACHINE TIME IN MINUTES PER RUN

Number of Defect Regions	Phase I	Phase II	Phase III	Total
3500	3	4	4	11
10,000	9	22	12	43

RESULTS OF SIMULATION PROGRAM

The program was tested by using it to simulate a symmetric binary channel with no error dependence. For this channel, exact error probabilities can be computed [see (1) in the Appendix]. Results of these computations were compared with results of the simulation program. A few of these results are shown in Fig. 5 for N equal to 24, 12, and 8. This channel had an *a priori* error probability per track of $p = 2 \times 10^{-3}$. The average error for the double error probability, as compared with the theoretical results, varied in these examples from 0.76 to 1.77 per cent.

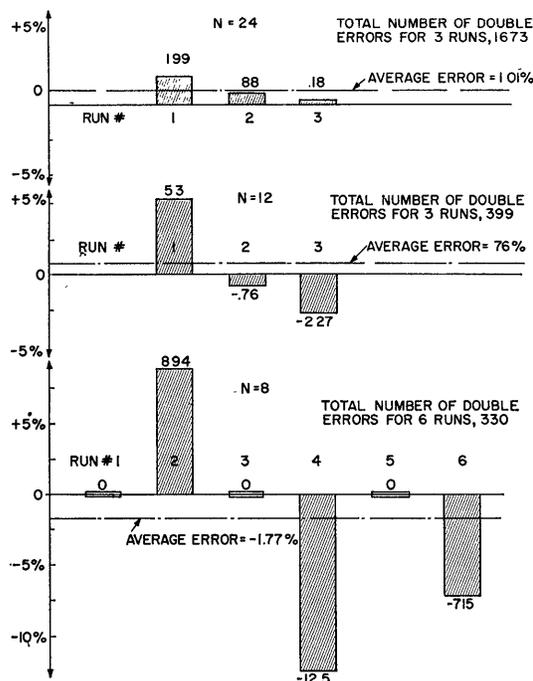


Fig. 5—Error in per cent for double error probability. Symmetric channel (defect length = 1); *a priori* error probability, $p = 2 \times 10^{-3}$.

As may be expected, the accuracy of the results depends on how many runs have been made. The number of runs required will be different for each problem and will depend on how long it takes to accumulate a significant count for the various error patterns of interest. The theoretical limit to the final accuracy of the results of the simulation program is set by the inherent inaccuracy of the pseudo random number generator and on the accuracy with which the mathematical model describes the actual information channel.

The simulation program has been designed for information channels with rather complex characteristics. An example of such a channel is a Z channel. In a Z channel, one of the binary symbols is always transmitted correctly; thus, it is a channel with the greatest possible degree of asymmetry. For the example selected, the probabilities of occurrence of defects of various lengths are shown in Fig. 6. This hypothetical distribution was developed to study the effect of channel asymmetry and error dependence on error probabilities. In this distribution, the defects contain as many single bits per track as in the previous example for the symmetric channel.

For this Z channel, the approximate double error probabilities for $N = 24, 12,$ and 8 are shown in Fig. 7. The results for this example indicate that good approximations for multiple error probabilities can be obtained by assuming that each long defect region is split up into as many separate regions one bit long, as there are bits in the original defect region.

The multiple error probabilities for this Z channel ($p_1 = 2 \times 10^{-3}$) can be computed, using (3) or (8) in the Appendix.

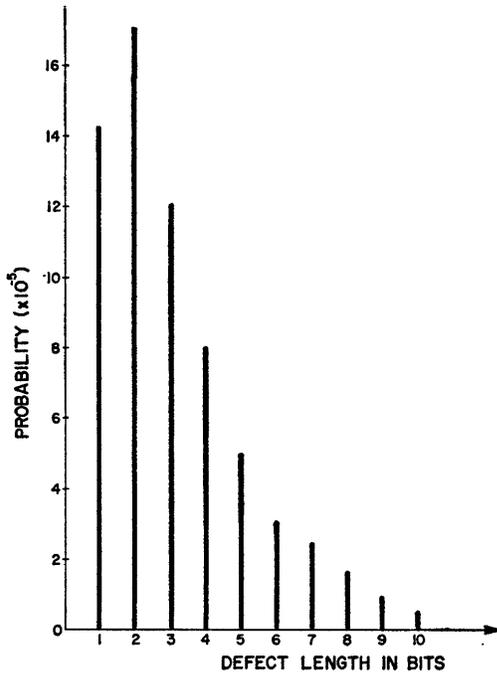


Fig. 6—Hypothetical probability of occurrence of defects of various lengths.

SUMMARY

The results of the simulation program indicate that good approximations to the probabilities of multiple-error patterns in symbols consisting of N bits can be obtained.

The computer time required for the simulation is quite reasonable and compares favorably with the costs for the testing of numerous complete versions of a system. The simulation procedure is, therefore, particularly useful during the early stages of development.

The program, because of its flexibility, also lends itself to purely theoretical investigations of error statistics.

APPENDIX

EXPRESSIONS FOR ERROR PROBABILITIES FOR AN ASYMMETRIC BINARY CHANNEL WITH NO ERROR DEPENDENCE

The binary asymmetric channel can be represented by Fig. 8 where

- q_0 = the probability of a transmitted zero being received as a zero, $q_0 = 1 - p_0$,
- q_1 = the probability of a transmitted one being received as a one, $q_1 = 1 - p_1$,
- p_0 = the probability of a transmitted zero being received as a one,
- p_1 = the probability of a transmitted one being received as a zero.

Let there be N bits in a character, of which r are ones and $(N - r)$ are zeros. The probability $p_N(z)$ or z errors in a character consisting of N bits can be expressed for a symmetric channel¹ where $p_1 = p_0$.

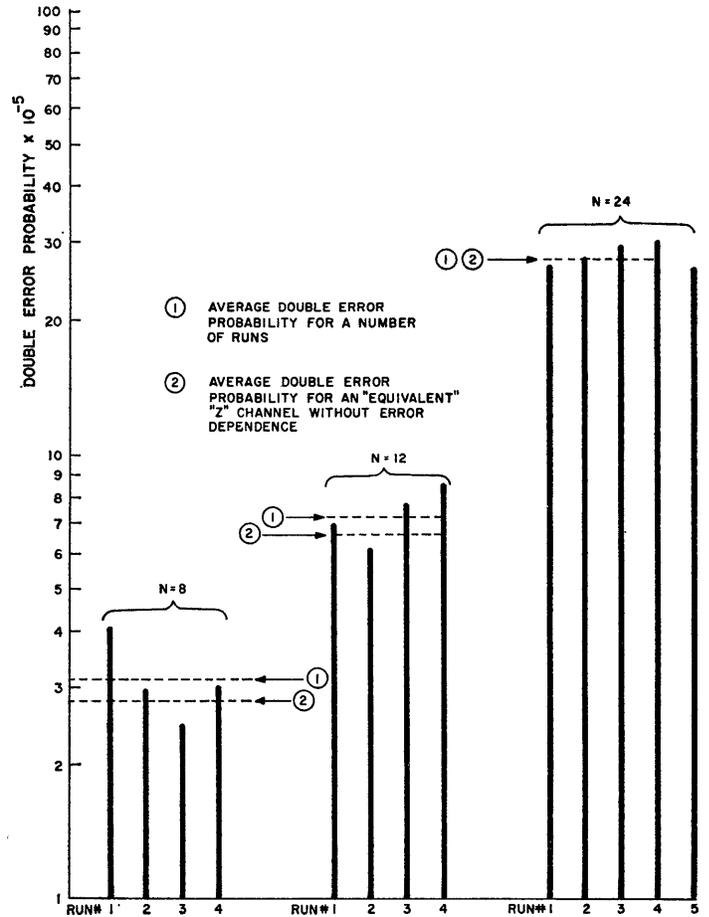


Fig. 7—Double error probability for a Z channel for error dependence shown in Fig. 5.

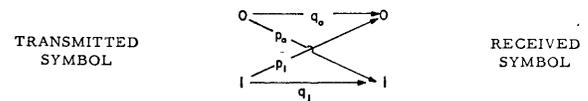


Fig. 8.

$$p_N'(z) = \binom{N}{z} p_1^z q_1^{(N-z)} \tag{1}$$

For an asymmetric channel, if all N -bit symbols are assumed to be transmitted with equal probability,

$$p_N''(z) = \frac{1}{2^N} \left[\sum_{r=0}^{r=N} \binom{N}{r} \left[\sum_{x=0}^{x=z} \left\{ \binom{r}{x} p_1^x q_1^{(r-x)} \right\} \cdot \left\{ \binom{N-r}{z-x} p_0^{(z-x)} q_0^{N-1-(z-x)} \right\} \right] \right] \tag{2}$$

and, for a Z channel, $p_0 = 0$ and $q_0 = 1$.

$$p_N'''(z) = \frac{1}{2^N} \left\{ \sum_{r=z}^{r=N} \binom{N}{r} \binom{r}{z} p_1^z q_1^{r-z} \right\} \tag{3}$$

where

$$\left(\frac{N}{x}\right) \equiv \frac{N!}{x!(N-x)!} \quad (4)$$

Simplifications can be made in (1)–(3), if systems of inherently high reliability are considered. That is, if

$$p_1 \ll 1, \quad p_0 \ll 1. \quad (5)$$

Expanding $(1-p)^k$ in terms of a power series and retaining only the first term, (1)–(3), respectively, become

$$p_N'(z) = \left(\frac{N}{z}\right) p_1^z \quad (6)$$

and, if $p_1 = \alpha p_0$,

$$p_N''(z) = \frac{1}{2^N} \left\{ \sum_{r=0}^{r=N} \binom{N}{r} \left[\sum_{x=0}^{x=z} \binom{r}{x} \left(\frac{N-r}{z-x}\right) \frac{p_1^z}{\alpha^{z-x}} \right] \right\} \quad (7)$$

$$p_N'''(z) = \frac{1}{2^N} \sum_{r=z}^{r=N} \binom{N}{r} \binom{r}{z} p_1^z. \quad (8)$$

ACKNOWLEDGMENT

The authors would like to thank the U. S. Department of Defense for permitting the publication of this paper. Valuable suggestions by C. L. Christiansen, R. J. Sippel, and P. J. Nelson contributed greatly to this paper.

A Compiler with an Analog-Oriented Input Language

M. L. STEIN[†], J. ROSE[‡], AND D. B. PARKER[‡]

INTRODUCTION

ANALOG computation is, for many problems, more convenient than digital computation but lacks the precision obtainable from a digital-computer solution. A compiler has been developed which, for problems involving differential equations expressible in the form

$$\dot{y}_i = f(y_1, y_2, \dots, y_n), \quad i = 1, 2, \dots, n \quad (1)$$

combines to a considerable extent the desirable attributes of both types of computation. The compiler achieves this by deducing from a description of an analog setup diagram the differential equations which the analog computer solves, and then compiling a program for solving the equations.

Two drawbacks are avoided. The differential equations represented by a diagram are deduced with no attempt to simulate the analog computer, and a sophisticated integration procedure is used. Therefore, an accurate and relatively efficient digital-computer program is obtained. Even though the solution is obtained from the differential equations, the identity of the output of each element on the diagram is not lost, so that the results may be visualized more easily as the response to the physical system being studied.

The integration procedure used in the final program is the Gill¹ version of the fourth-order Runge-Kutta

method. Since no starting procedure is required, and because the discontinuities introduced by nonlinear analog elements cause no difficulty, this method was adopted.

The compiler is not the first digital computer program with an input language related to differential analyzers; for example, there are the programs DIDAS² and DEPI.³ However, the program which this paper describes differs from other similar programs of which the authors are aware in one or more of the following respects:

- 1) The input language is closely related to an extensively used electronic analog computer.
- 2) The user need not provide his own input and output program. The compiler provides a complete program ready to run.
- 3) Since the final program produced is in machine language, it is efficient in terms of execution time as compared to an interpretive program.
- 4) Rather than simulating a differential analyzer, the compiler deduces from a setup diagram the differential equations represented by the diagram.

In producing a program to solve the differential equations expressed by an analog setup diagram, the compiler uses a technique of increasing popularity.⁴ This technique is the use of another processor as an inter-

[†] University of Minnesota, Minneapolis, Minn. The work of this author was supported in part by Convair-Astronautics.

[‡] Convair (Astronautics) Div., General Dynamics Corp., San Diego, Calif.

¹ S. Gill, "A process for the step-by-step integration of differential equations in automatic digital computing machines," *Proc. Cambridge Phil. Soc.*, vol. 47, pp. 96–108; June 3, 1950.

² G. R. Slayton, "DIDAS," presented at the Twelfth Natl. Meeting of the Assoc. for Computing Machinery; June, 1958.

³ F. H. Lesh and F. R. Curl, "DEPI, An Interpretative Digital-Computer Routine Simulating Differential-Analyzer Operations," Jet Propulsion Lab., California Inst. Tech., Pasadena, Calif., Memo. No. 20-141; March 22, 1957.

⁴ *Communications of the Assoc. for Computing Machinery*, vol. 1, no. 7, p. 5; July, 1958.

mediate step. In this case the intermediate processor is Fortran,⁵ an automatic coding system developed by IBM which accepts statements closely resembling the ordinary language of mathematics as input. The output of the compiler is in the input language of Fortran, and is translated by Fortran into machine language. Through the use of Fortran, the task of developing the compiler was greatly simplified.

The method used to deduce the differential equations represented by a setup diagram through analysis of its description is developed in a previous paper by two of the authors.⁶ Therefore the method will not be developed here. Instead it will be illustrated by example. A description of the preparation of problems for the compiler and a description of the compiler will be given. The application of the compiler will be illustrated with the solution of a simple problem, and in conclusion, experience in its use will be discussed.

THE ANALOG SETUP DIAGRAM DESCRIPTION

The conversion of an analog setup diagram to a digital-computer program involves several steps. The diagram is described. The description is processed by the computer and a Fortran program is produced. The program is compiled by Fortran into a machine language program ready to be run. All of these steps except the first are performed by an IBM 704. This first step, the description of the diagram, will now be discussed.

Analog Elements Recognized

In order to analyze the description of an analog setup diagram, the compiler must be able to recognize the more commonly used analog elements. Those elements available on the Electronics Associates PACE electronic analog computer were chosen as representative. These elements are listed in Table I. The diagram symbols for the elements are those most commonly used for PACE setup diagrams. The mathematical expression for the function of each element is chosen so that problems prepared for PACE computers can be converted to digital programs with little or no change in the setup diagram. For this reason, the scale factors associated with multipliers and dividers are included, and resolvers involve angles measured in volts where one volt equals two degrees.

Normally, scaling requirements for an analog computer are quite restrictive; but since the operations in the digital computer program are performed in floating-point form, all numbers, including parameters associated with elements, can vary over a much wider range in the digital computer. Therefore, diagrams to be processed by the compiler need not be scaled for an analog computer.

In addition to the use of amplifiers as summers and integrators, special one-amplifier circuits representing more complicated transfer functions are sometimes used on electronic analog computers in order to save analog elements. Such use of amplifiers is not permitted on diagrams whose description is to be processed by the compiler. One-amplifier circuits generating special transfer functions must be replaced with equivalent circuits using elements in Table I.

Two of the elements, element 13 and element 14, listed in Table I do not correspond to actual analog elements, but have been included for convenience and for the compilation of more efficient digital programs. Element 13, the Fortran statement, is included to allow the replacement, if desired, of feedback loops generating functions such as $x^{2/3}$, e^x , etc., by a Fortran statement utilizing a library subroutine in order to obtain a more efficient digital program. Element 14, the external input, is included to allow the combination of compiler-generated Fortran programs with other Fortran programs.

The Preparation of a Diagram Description

The compiler converts an analog-computer problem to a program by processing information supplied in a description of the analog setup diagram. Therefore, the preparation of an accurate diagram description by the user of the compiler is an essential step in the process of obtaining a correct digital program.

Before beginning the description of a diagram, however, it should be determined that the problem is of a type suitable for conversion. While the analog computer is capable of solving several types of problems, the compiler is restricted to problems involving differential equations expressible in the form (1). That is, no implicit relationships are allowed among derivatives or among other variables. The compiler detects implicit relationships by discovering the presence on the diagram of feedback loops without integrators, a procedure whose suitability for detecting implicitness is demonstrated in a previous paper.⁶ Therefore, the diagram should be examined for such loops before preparing the description.

Having determined that the problem is of a suitable type, the diagram must be examined for elements other than those listed in Table I. Circuits involving other elements must be replaced by equivalent circuits using acceptable elements. Circuits for obtaining powers, roots, arctangents, exponentials, and natural logarithms may be replaced, if desired, by Fortran statement elements.

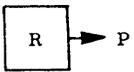
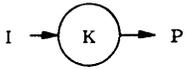
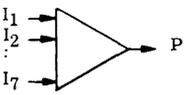
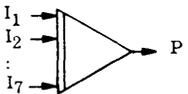
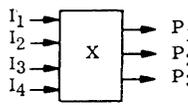
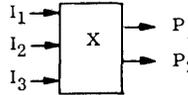
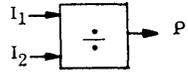
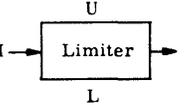
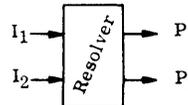
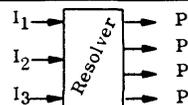
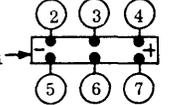
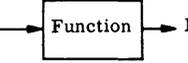
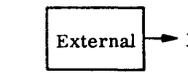
After it has been determined that the problem is of a suitable type and that all circuits are acceptable, the actual process of describing the diagram is clerical in nature. Consequently, the description may be done by an engineering aide, freeing the engineer who originated the problem for tasks making better use of his skills.

The first step in preparing a diagram description is to assign to each element on the diagram an arbitrary

⁵ J. W. Backus, et al., *Programmers Reference Manual for the Fortran Automatic Coding System for the IBM 704*; December, 1957.

⁶ M. L. Stein and J. Rose, "The Automatic Deduction of Differential Equations from Analog Setup Diagrams," *Mathematical Pre-Print Series*, Pre-Print No. 13, Convair-Astronautics, San Diego, Calif.

TABLE I
ELEMENTS RECOGNIZED BY THE COMPILER

	NAME	DIAGRAM SYMBOL	COMPILER SYMBOL	FUNCTION
1	Reference		RF	$P = R$, $R = \text{constant}$
2	Potentiometer		PT	$P = KI$ $K = \text{constant}$
3	Summer		SU	$P = -(a_1 I_1 + a_2 I_2 + \dots + a_7 I_7)$ $a_k = 1, 5, \text{ or } 10$
4	Integrator		IN	$P + C - \int_{t_0}^t (a_1 I_1 + a_2 I_2 + \dots + a_7 I_7) dt$ $a_k = 1, 5, \text{ or } 10$, $C = \text{constant}$
5	Servo Multiplier		MS	$P_1 = I_1 \times I_2 / 100$ $P_2 = I_1 \times I_3 / 100$ $P_3 = I_1 \times I_4 / 100$
6	Electronic Multiplier		ME	$P_1 = -I_1 \times I_2 / 100$ $P_2 = -I_1 \times I_3 / 100$
7	Divider		DI	$P = -100 (I_1 / I_2)$
8	Limiter		LM	$P = U, I \text{ or } L$ as $I \geq U$, $U > I > L$, or $I \leq L$ respectively
9	Rectangular to Polar Resolver		RP	$P_1 = \sqrt{I_1^2 + I_2^2}$ $P_2 = \arctan (I_2 / I_1)$
10	Polar to Rectangular Resolver		PR	$P_1 = I_2 \sin (I_1)$ $P_3 = I_3 \sin (I_1)$ $P_2 = I_2 \cos (I_1)$ $P_4 = I_3 \cos (I_1)$
11	Switch		SW	If $I_1 < 0$, ② and ③ ; ⑤ and ⑥ are connected. If $I_1 \geq 0$, ③ and ④ , ⑥ and ⑦ are connected.
12	Function Generator		FG	$P = F (I)$ as determined by linear interpolation in a table.
13	Fortran Statement		FS	$P = F (I_1, I_2, \dots, I_7)$ as expressed in a Fortran "arithmetic formula"
14	External Input		EI	P must be defined by another Fortran program.

number between 1 and 999. Using a suitable input form of the type shown in Fig. 1, the elements are described one at a time in any convenient order. An element's number is entered on the form. Then its type is designated as given in Table I—for example, *SU* for a summer. If the element has parameters associated with it, the values of the parameters are entered on the form. If the output of the element is to be printed, an *N* for normal output or a *C* for checkout output is written on the form. The inputs to the element are then listed by entering the numbers of the elements from which the inputs come. If the element is an integrator or a summer, the scale factors associated with the inputs are listed. As an example, Fig. 2 shows the complete entry for element 2, an integrator whose single input is equal to five times the first output of element 6, an electronic multiplier, and whose initial output is -45 .

Fig. 1—Input form for the compiler.

Each element is described in a fashion similar to that above on one line of the input form. When all elements have been described and the descriptions checked, the form is given to a key-punch operator, who punches each line on a card. These cards become the input to the compiler.

Fig. 2—Input form entry for one element.

THE TRANSFORMATION PROCESS

In order to transform a description of an analog setup diagram into a digital-computer program, the compiler, using a procedure developed in a previous paper,⁶ must deduce the differential equations which the diagram represents. Having deduced the equations, the compiler must then produce a Fortran program suitable for solving the equations. These two major phases of the transformation process will now be described and will be related to the complete compiler program.

First, the differential equation of a simple diagram will be deduced. In Fig. 3 we see the diagram for the equation

$$\dot{y} = R - Ky. \tag{2}$$

P1 represents the output of element 1, P2 the output of element 2, etc., and D2 represents the total input to element 2, an integrator. Examination of the types of elements 1 and 3 (see Table I) permits the equations

$$P1 = R \tag{3}$$

$$P3 = KP2 \tag{4}$$

to be written. Examination of element 2 discloses that it is an integrator. The equation for its input can be written as

$$D2 = + P1 + P3 \tag{5}$$

If (3) and (4) are substituted into (5), the equation

$$D2 = R + KP2 \tag{6}$$

is obtained. Recalling that

$$P2 = C - \int_{t_0}^t D2 dt,$$

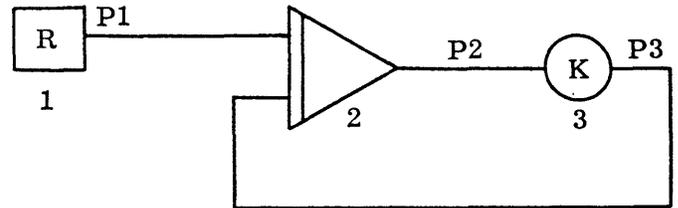


Fig. 3—Diagram for $\dot{y} = R - ky$.

as shown in Table I, (6) is seen to be equivalent to (2). Therefore, given values for *C* (the initial value of P2), *R*, and *K*, a solution for (2) can be obtained through use of a suitable numerical integration procedure.

Actually, (3)–(5) need not be combined if they are evaluated in the proper sequence, since all that is required to carry out the numerical integration is that it be possible to obtain a value for D2, given a value of *R*, *K*, and P2. If equations are not to be combined, deducing the differential equations from a diagram description consists of determining the proper sequence for evaluating each element and then producing equations for evaluating the output of each element with the exception of integrators, whose input must be evaluated as seen above. Leaving equations uncombined leads to a somewhat less efficient computer program. However, it has the advantage of preserving the identity of the output of every element, a useful property when programs are debugged, and also aids in visualizing the results as the response of the physical system being studied. Since a procedure for deducing differential equations is easier to implement, too, if equations are left uncombined, this method was chosen for the compiler.

If equations are not to be combined, the process of converting a description to Fortran input language can be divided into two distinct phases: 1) the determination of the proper sequence for evaluating integrator inputs and nonintegrator element outputs, and 2) the generation of suitable Fortran statements for obtaining

the value of these inputs or outputs as a function of the outputs of other elements. The former phase, sequencing, will now be considered.

Sequencing

In order to give a pictorial presentation of the sequencing process, a table giving the proper sequence for the evaluation of the elements of an abstract analog diagram will be constructed in a fashion similar to that which the compiler would employ.

An analog setup diagram is given in Fig. 4 for a first-order differential equation; that is, only one integrator appears. Since the sequencing process need distinguish only between elements which are integrators and those which are not, types of elements other than that of element 7, an integrator, are not specified. Given a description of the diagram, the first step in constructing a sequence table is to locate an integrator, in this case element 7, and enter its number in a sequence table. Examining the description of element 7, one of its inputs, element 3, is selected for consideration, the number of 3 is entered in the sequence table and its description is located. Proceeding to 3's input 5 and entering 5 in the table, the description of the portion of the diagram indicated by the heavy line in Fig. 5 has been examined, and the entries in the sequence table shown in Fig. 5 have been made. Since 5 has no inputs, element 3 must be reselected and its description examined for additional inputs which have not yet been selected. Element 3 has a second input, 1, which has not yet been examined. Proceeding in the manner similar to that above, 1, 6, and 4 are entered in the table as shown in Fig. 6. After 4 has been entered in the table, since 4 has no inputs, 1 is reselected, but both of its inputs have been examined. Therefore 3 is reselected, but both of its inputs have been examined also. This leads to the reselection of 7 and the examination of its second input, 8.

The tracing of the inputs to 8 will disclose an interesting feature of the sequencing process. This feature is that elements with split outputs such as element 1 cause duplication of entries in the sequence table. Examination of the description of 8 reintroduces 1 for selection. Examination of 1 reintroduces 6 and 4 as shown in Fig. 7. Since 4 has no input, and both inputs of 1 have been examined, 8 is reselected. Element 8's second input is 2, and 2's input is 9. Therefore, 2 and 9 are entered in the table. The reselection of 2 and the examination of its second input discloses that its second input is element 7, an integrator.

Inputs to elements which are the outputs of integrators are treated as a special case. The element number of an integrator is never entered in a sequence table except as an initial entry. The reason: no expression for evaluating the output of an integrator need be written. The value of the output of the integrator is supplied by the numerical integration. Since 2's second input is 7, an integrator, no table entry is made.

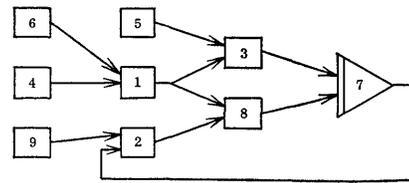


Fig. 4—An abstract analog setup diagram.

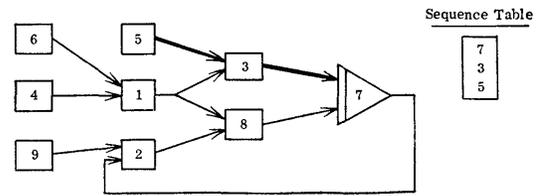


Fig. 5—Partially examined diagram description.

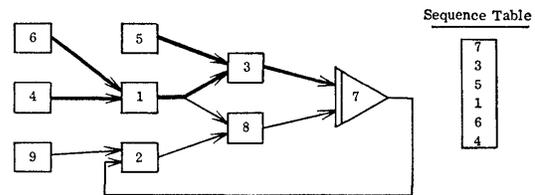


Fig. 6—Partially examined diagram description.

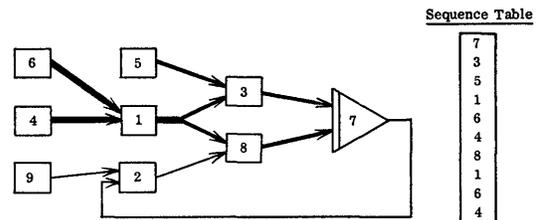


Fig. 7—Partially examined diagram description with reintroduction of 1, 6, 4.

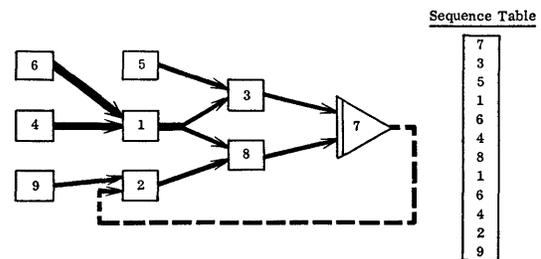


Fig. 8—The completed examination of the diagram description.

At this point, all inputs to every element of the diagram have been examined and the sequence table is complete. The compiler discovers that the examination of the diagram description is completed in the following fashion. Since both inputs to 2 have been examined, 8 is reselected for examination. Both inputs to 8 have also been examined. Consequently 7 is reselected, but both inputs to 7 have been examined and 7 is the initial entry in the sequence table. Thus, examination of the diagram description is completed as shown in Fig. 8.

While the examination of the diagram description is complete, the sequence table is not yet finished. Redundant element numbers must be eliminated with the lowest occurrence of a number retained as shown in Table II, and the table must be inverted and compressed as shown in Table III in order to obtain a proper sequence for evaluating the elements.

TABLE II
SEQUENCE TABLE WITH
REDUNDANCIES ELIMINATED

7
3
5
8
1
6
4
2
9

TABLE III
FINAL SEQUENCE TABLE
(TABLE II INVERTED)

9
2
4
6
1
8
5
3
7

The final sequence table, Table III, has been obtained. Comparison of Table III with the diagram of Fig. 8 shows that if the elements are evaluated in the order given by Table III, the output of each element 9 through 3 can be calculated and the input to element 7 obtained. Using a suitable numerical integration procedure, the output of 7 can be obtained. Hence, a proper sequence for solving the differential equation represented by the diagram has been determined.

The illustration above is for a single differential equation. If the diagram involves several differential equations, that is, several integrators, a slight elaboration of the procedure used above is required. An integrator is found and a sequence table is constructed as shown above. All elements in the final sequence table are defined as being *complete*. Another integrator is found, and the procedure above is repeated with the additional provisions that those inputs which come from *complete* elements give rise to no table entries, and that *complete* elements are not selected for examination of their inputs. The final sequence table obtained for the second integrator is added to the sequence table for the first. The process is continued until all integrators have been found and a sequence table has been obtained for the entire diagram.

A previous paper⁶ proves that the procedure demonstrated above does obtain in all cases the proper sequence for diagrams involving equations of form (1) and develops a method for detecting equations not of form (1). This method was not demonstrated above, but is included in the compiler.

Generation of Fortran Statements

The second phase of converting a setup diagram to Fortran input language, the generation of suitable

Fortran statements for evaluating elements, will now be considered.

In contrast to the sequencing phase, the generation of Fortran statements is lengthy but logically simple. The compiler begins with the first entry in the sequence table, and an element number is obtained. The description of the element is located, and its type is examined. On the basis of its type, a set of skeleton Fortran statements is selected, a set being provided for each type of element. The inputs to the element are obtained from its description and are inserted in the proper places in the skeleton statements. The statements are written on magnetic tape. Then the next entry in the sequence table is obtained and the process repeated. In Table IV we see examples of Fortran statements generated for each type of element.

In addition to generating statements for each element as described above, the compiler generates such other Fortran statements as are required to obtain a complete program.

The Complete Compiler Program

The complete compiler program including the sequencing and statement generation portions will now be described in terms of the functions which the program performs. These are as follows:

- 1) It tabulates the data supplied in the description of the analog setup diagram.
- 2) The differential equations represented by the diagram are deduced through determination of the proper sequence for evaluating elements.
- 3) Fortran statements providing a complete program are generated.
- 4) As the data of the diagram description is processed, it examines the data for errors.

These functions divide the compiler into four major segments—the input routine, the sequencer routine, the Fortran statement generator, and the error diagnostic routine.

The Input Routine: The input routine reads the description of the elements into the computer memory and forms various tables. These are a table of elements which have associated parameters, a table of the values assigned to these parameters, a table of elements which are integrators, and a table of those elements whose output is desired for printing.

The Sequencer Routine: The sequencer routine forms a table showing the order in which integrator inputs and nonintegrator element outputs must be evaluated.

The Fortran Statement Generator: The Fortran statement generator produces a Fortran program the parts of which are concerned with the following operations:

- 1) Assignment of memory locations for data;
- 2) Assignment of values to all parameters associated with elements including the initial values of integrator outputs;

TABLE IV
AN EXAMPLE OF FORTRAN STATEMENTS FOR EACH TYPE OF ELEMENT

Element	Statement Number	Statement
reference	00101	P101 = XR101
potentiometer	00002	P002 = XK002 * P001
summer	00016	P016 = -P001 - P002 - 5. * P0051 - 10. * P006
integrator	00004	P004 = +P003 + 10. * P011 + 5. * P0071
servo multiplier	00005	P0051 = P011 * P0071/100. P0052 = P011 * P001/100. P0053 = P011 * P003/100.
electronic multiplier	00007	P0071 = -P003 * P0051/100. P0072 = -P003 * P002/100.
divider	00008	P008 = -100. * (P0071/P019)
limiter	00095	IF (P094 - XU095) 02095, 01095, 01095 P095 = XU095 GO TO 05095 IF (P094 - XL095) 04095, 04095, 03095 P095 = P094 GO TO 05095 P095 = XL095 CONTINUE
	01095	
	02095	
	03095	
	04095 05095	
Fortran statement	00015	P015 = P009 ** 1.5 + EXPF (P012 - 17.6)
external input		no statement required
rectangular to polar resolver	00034	P0341 = SQRTF (P029 ** 2 + P028 ** 2) IF DIVIDE CHECK 01034, 01034 IF QUOTIENT OVERFLOW 02034, 02034 TANP2 = P028/P029 IF DIVIDE CHECK 04034, 03034 IF QUOTIENT OVERFLOW 04034, 07034 IF (P028) 05034, 20000, 06034 PN2 = -45. GO TO 08034 PN2 = +45. GO TO 08034 PN2 = ATANF (TANP2) * 28.6478897 IF (P029) 09034, 11034, 11034 P0342 = PN2 + 90. GO TO 12034 P0342 = PN2 CONTINUE
	01034	
	02034	
	03034	
	04034	
	05034	
	06034	
	07034	
	08034	
	09034	
11034 12034		
polar to rectangular resolver	00022	P0221 = P023 * SIN (.03490658 * P008) P0222 = P023 * COS (.03490658 * P008) P0223 = P025 * SIN (.03490658 * P008) P0224 = P025 * COS (.03490658 * P008)
switch	00100	IF (P011) 01100, 02100, 02100 P1003 = P098 P1006 = 0. GO TO 03100 P1003 = P099 P1006 = 0. CONTINUE
	01100	
	02100	
	03100	
function generator	00012	IF (P003 - V012(1)) 30000, 01012, 01012 DO 02012 K=1, M012 IF (P003 - V012 (K)) 03012, 02012, 02012 CONTINUE GO TO 30000 P012 = W012 (K) - (W012(K) - W012 (K-1)) * (V012(K) - P003) / (V012(K) - V012(K-1))
	01012	
	02012	
	03012	

- 3) Read-in of tables for function generators, of parameter changes, and of various constants associated with a particular run—the integration interval, the printout interval, the type of output, the cutoff time, and the number of parameter changes;
- 4) Numerical integration;
- 5) Evaluation of derivatives, that is, the computation of the outputs of all elements which are not integrators and computation of the inputs of integrators;
- 6) Finally, provision for printing output and testing for cutoff time.

The Error Diagnostic Routine: The detection of errors occurs in either the input routine or the sequencer. The function of the error diagnostic routine is to print a complete explanation of the type of error detected, the numbers of the elements involved, and other pertinent information to enable the user to locate and correct errors in the diagram or in its description. The error diagnostic routine must also make such changes in the diagram description as are required in order to allow the compiler to continue to the end of sequencing so that several errors may be detected in one compilation attempt.

AN EXAMPLE OF THE USE OF THE COMPILER

The solution of a simple problem will now be presented as an illustration of the use of the compiler. The authors are indebted to Robert Ferner of Convair-Astronautics for the preparation of the problem given below.

The problem to be solved involves the testing of a possible transfer function for translating an error in velocity normal to a reference trajectory of a missile into a turning rate to the missile's autopilot. The autopilot is approximated by the transfer function,

$$G = \frac{10}{S^2 + 4S + 10} \quad (7)$$

The acceleration of the missile is given by

$$A = \frac{Tg}{W - \int_0^t W dt} = \frac{100,000 \cdot 32.2}{50,000 - \int_0^t 500 dt}$$

The form of the transfer function to be tested is

$$\frac{W}{V_e} = \frac{K_1 + K_2 S}{S + K_3} \quad (8)$$

with K_1 , K_2 , and K_3 to be determined for stability and ability to correct V_e , the error in velocity. The rate input to the autopilot is limited in magnitude to 0.02.

A setup diagram for this problem is shown in Fig. 9. The elements on the diagram have been numbered arbitrarily in preparation for a description of the diagram.

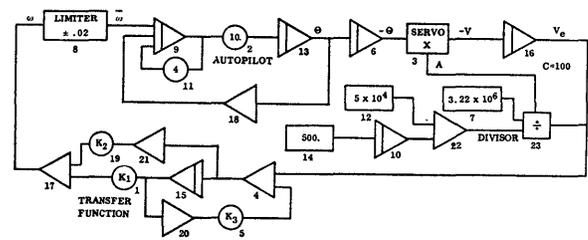


Fig. 9—Setup diagram for transfer function problem.

The description of the diagram is given in Fig. 10.

A list of cards keypunched from the description is shown in Fig. 11, and a list of the Fortran program generated by the compiler is given in Fig. 12. This program was processed by Fortran, and the resulting machine program was run. The results of a run with $K_1=0.003$, $K_2=0.0035$, and $K_3=0.5$ are shown in Fig. 13. Finally, a graph of the results as plotted by a card plotter is given in Fig. 14.

EXPERIENCE WITH THE COMPILER

To date, a "breadboard" version of the compiler has been in use. It differs from the final version in that the error diagnostic routine is not completed. The detection of an error in the diagram description ends the compilation attempt by giving a memory dump. Consequently, the "breadboard" compiler is not as convenient to use, inasmuch as several passes on the computer may be required if several errors are committed in describing the diagram. In spite of this shortcoming, the "breadboard" compiler has proven to be quite useful.

Nine problems have been done with the compiler to date; consequently, accurate statistics on its use are not available. However, orders of magnitude can be given. The analog diagrams for the problems ranged in size from approximately 20 elements to over 300 elements. The time required to prepare and check the diagram descriptions varied from one hour to about six hours for the largest problem. The computer time required to compile the program (including Fortran compilation) was from one-tenth of an hour to approximately one hour. The programs for problems which had also been run on the analog computer used approximately four times the amount of computer time required for an equivalent but less accurate analog run. These were small problems, however. Larger problems would give a much more unfavorable ratio.

As would be expected, the users of the compiler have been far more familiar with analog than digital computation. Nevertheless, it has been possible for them to originate problems for the digital computer and obtain solutions as accurate as those provided by hand-tailored programs. In general, this lack of familiarity with digital techniques has caused little difficulty since the compiler causes the digital computer to have many of the proper-

NUMBER	TYPE	R, K, C, U	L	PRINT	INPUT 1	INPUT 2	INPUT 3	INPUT 4	INPUT 5	INPUT 6	INPUT 7
1	P T	. 0 0 0 1		C	1 5						
2	P T	1 0		C	9						
3	M S			N	6	2 3					
4	S U			C	5	1 6					
5	P T	. 5		C	2 0						
6	I N	0 .		N	1 3						
7	R F	3 . 2 2 E 6		C							
8	L M	+ . 0 2	- . 0 2	N							
9	I N	0		C	8	1 8	1 1				
10	I N	0		C	1 4						
11	P T	4		C	9						
12	R F	5 E 4		C							
13	I N	0		N	2						
14	R F	5 0 0		C							
15	I N	0		C	4						
16	I N	1 0 0		N	3 1						
17	S U			N	1 9	1					
18	S U			C	1 3						
19	P T	0 0 2 0		C	2 1						
20	S U			C	1 5						
21	S U			C	4						
22	S U			C	1 2	1 0					
23	D I			N	7	2 2					

Fig. 10—Description of diagram for transfer function problem.

AUTOPILOT TRANSFER FUNCTION						
16IN	100.		N	31		
17SU			N	19	1	
8LM	+ .02	- .02	N	17		
13IN	0.		N	2		
6IN	0.		N	13		
23DI			N	7	22	
3MS			N	6	23	
1PT	.0001		C	15		
2PT	10.		C	9		
4SU			C	5	16	
5PT	.5		C	20		
7RF	3.22E6		C			
9IN	0.		C	8	18	11
10IN	0.		C	14		
11PT	4.		C	9		
12RF	5.E4		C			
14RF	500.		C			
15IN	0.		C	4		
18SU			C	13		
19PT	.0020		C	21		
20SU			C	15		
21SU			C	4		
22SU			C	12	10	

Fig. 11—List of cards punched from description. (The sequence has been altered to order problem output.)

```

AUTOPILOT TRANSFER FUNCTION
DIMENSION A(12), X(017), Y(007), Q(007)
EQUIVALENCE (X(001), T), (X(002), P016), (Y(002), D016),
1 (X(003), P013), (Y(003), D013), (X(004), P006), (Y(004), D006),
2 (X(005), P009), (Y(005), D009), (X(006), P010), (Y(006), D010),
3 (X(007), P015), (Y(007), D015)
EQUIVALENCE (X(008), XU008), (X(009), XL008),
1 (X(010), XK001), (X(011), XK002), (X(012), XK005), (X(013), XR007),
2 (X(014), XK011), (X(015), XR012), (X(016), XR014), (X(017), XK019)
20000 A(1) = .5
      A(2) = .29289322
      A(3) = 1.7071068
      A(4) = .16666667
      A(5) = 1.
      A(6) = .29289322
      A(7) = 1.7071068
      A(8) = .33333333
      A(9) = .5
      A(10) = .29289322
      A(11) = 1.7071068
      A(12) = .5
      X(001) = 0.
      X(002) = 100.
      X(003) = 0.
      X(004) = 0.
      X(005) = 0.
      X(006) = 0.
      X(007) = 0.
      X(008) = +.02
      X(009) = -.02
      X(010) = .0001
      X(011) = 10.
      X(012) = .5
      X(013) = 3.22E6
      X(014) = 4.
      X(015) = 5.E4
      X(016) = 500.
      X(017) = .0020
      READ 20001, NTO, MP, DT, CT, NC
20001 FORMAT (11, 18, F16.8, F15.8, 19)
      PRINT 20002
20002 FORMAT (49H1AUTOPILOT TRANSFER FUNCTION
1 55HO TYPE OUTPUT OUTPUT INTERVAL TIME INTERVAL)
      IF (NTO)20003, 20005, 20003
20003 PRINT 20004, MP, DT
20004 FORMAT (14H CHECKOUT, I15, F23.6)
      GO TO 20007
20005 PRINT 20006, MP, DT
20006 FORMAT (13H NORMAL, I16, F23.6)
20007 PRINT 20008, CT, NC
20008 FORMAT (33HO CUTOFF TIME NO. CHANGES/1H, F19.6, I10)
      IF (NC)20015, 20015, 20009
20009 PRINT 200010
20010 FORMAT (31HO I X(I))
      DO 20013 J=1, NC
20011 READ 20012, I, (X(I))
20012 FORMAT (I3, F14.8)
20013 PRINT 20014, I, (X(I))
20014 FORMAT (1H, I11, F23.8)
20015 ASSIGN 20022 TO JUMP1
      ASSIGN 20026 TO JUMP2
      ASSIGN 20016 TO JUMP3
      GO TO 20017
20016 ASSIGN 20018 TO JUMP1
      ASSIGN 20023 TO JUMP2
      ASSIGN 20031 TO JUMP3
      N = MP
20017 DO 20023 J=1, 4
      GO TO JUMP1, (20022, 20018)
20018 DO 20021 I=1,007
      IF (Y(I)) 20020, 20019, 20020
20019 R=0.
      GO TO 20021
20020 S = -DT * Y(I)
      R = A(J) * S - A(J+4) * Q(I)
      X(I) = X(I) + R
      Q(I) = Q(I) + 3. * R - A(J+8) * S
20021 CONTINUE
20022 Y(1) = -1.
00001 P001 = XK001 * P015
00002 P002 = XK002 * P009
00012 P012 = XR012
00022 P022 = -P012 - P010
00007 P007 = XR007
00023 P023 = -100. * P007/P022
00003 P0031 = P023 * P006/100.
00020 P020 = -P015
00005 P005 = XK005 * P020
00004 P004 = -P005 - P016
00006 D006 = +P013
00021 P021 = -P004
00019 P019 = XK019 * P021
00017 P017 = -P019 - P001
00008 IF (P017 - XU008)02008, 01008, 01008
01008 P008 = XU008
      GO TO 05008
02008 IF (P017 - XL008)04008, 04008, 03008
03008 P008 = P017
      GO TO 05008
04008 P008 = XL008
05008 CONTINUE
00011 P011 = XK011 * P009
00018 P018 = -P013
00009 D009 = +P008 + P018 + P011
00014 P014 = XR014
00010 D010 = +P014
00013 D013 = +P002
00015 D015 = +P004
00016 D016 = +P0031
      GO TO JUMP2, (20026, 20023)
20023 CONTINUE
20024 N = N - 1
      IF (N) 20025, 20025, 20031
20025 N = MP
20026 IF (NTO) 20027, 20029, 20027
20027 PRINT 20028, T, P016, P017, P008, P013, P006, P023,
1 P0031, P001, P002, P004, P005, P007, P009, P010, P011,
2 P012, P014, P015, P018, P019, P020, P021, P022
20028 FORMAT (1HO, F14.6/(1H, 4F14.6))
      GO TO 20030
20029 PRINT 20028, T, P016, P017, P008, P013, P006,
1 P023, P0031
20030 GO TO JUMP3, (20016, 20031)
20031 IF (X(1) - CT) 20017, 20000, 20000

```

Fig. 12—List of Fortran program generated by compiler.

AUTOPILOT TRANSFER FUNCTION			
TYPE OUTPUT NORMAL	OUTPUT	INTERVAL 40	TIME INTERVAL 0.025000
CUTOFF TIME	NO. CHANGES		
80.000000	2		
1	X(I)		
10	0.00030000		
17	0.00350000		
0.TIME			
100.000000 V_e	-0.350000 w	-0.020000 \dot{w}	0. θ
0. - θ	6439.9998784	0. - \dot{v}	
1.000000			
99.771433	-0.235164	-0.020000	-0.020674
0.011025	6505.050415	0.717208	
1.999999			
98.355980	-0.161977	-0.020000	-0.020226
0.032057	6571.428467	2.106574	
2.999998			
95.577519	-0.113329	-0.020000	-0.019941
0.052006	6639.174988	3.452772	
3.999997			
91.439002	-0.079243	-0.020000	-0.020008
0.071998	6708.333008	4.829841	
4.999996			
85.908216	-0.053577	-0.020000	-0.019999
0.092000	6778.946838	6.236650	
5.999995			
78.955898	-0.032604	-0.020000	-0.020000
0.112000	6851.063110	7.673180	
6.999994			
70.551993	-0.014055	-0.014055	-0.019236
0.131932	6924.730286	9.135929	
7.999993			
60.833487	0.002923	0.002923	-0.004096
0.144597	6999.998901	10.121800	

Fig. 13—A sample of output from the program.

ties of the analog computer as far as the user is concerned. However, the nonlinear elements, the switch and the limiter, have caused difficulties for the user in some cases. Either of these elements can introduce discontinuities in the functions being integrated. The response of an analog integrator to a discontinuity is nearly instantaneous; the response of the digital program is not. The numerical integration tends to smooth discontinuities, the amount of smoothing depending upon the integration interval. Then, too, in contrast to the analog computer, switching or the start of limiting can only occur at the beginning of or midway in an integration interval. Because of these effects, a certain amount of experimenting was sometimes necessary in choosing an integration interval.

When the compiler was undertaken, it was antici-

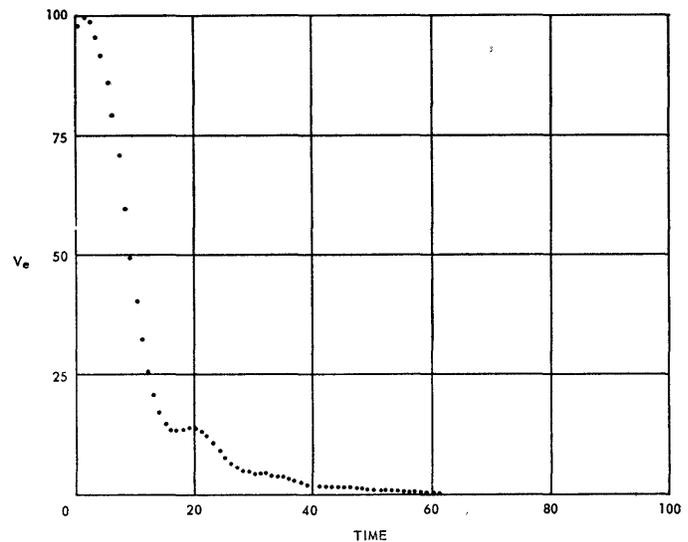


Fig. 14—A plot of the velocity error against time.

pated that it would have three uses—checking analog runs, obtaining greater accuracy for selected analog runs, or originating problems with no intention of making any actual analog runs. With the first two uses in mind, the elements which the compiler recognizes were made as similar to actual analog elements as possible so that existing analog diagrams could be converted with little or no change. In practice, the compiler has been used principally as a convenient way of originating problems for the digital computer. For these problems, the close duplication of analog idiosyncrasies has proven to be an inconvenience rather than an aid. Therefore, consideration is being given to producing another version of the compiler that recognizes elements more convenient for originating digital computer programs.

Experience has indicated that many problems are solved on an analog computer which cannot be transformed into programs by the compiler because of implicit relationships between derivatives or between the variables. Some of these problems contain implicit relationships involving nonlinear functions. However, this type of problem does not occur frequently enough to justify the effort required to develop a new compiler. Most of these problems, however, involve implicit relationships which can be eliminated by solving a system of simultaneous, linear, algebraic equations. Consideration is being given to the extension of the compiler to include this type of problem.

In conclusion, the authors feel that the compiler will be an extensively used program at Convair-Astronautics and will fully justify the modest expenditure of effort involved in its development.

Automatic Design of Logical Networks*

T. C. BARTEE†

THE frequent use of transcendental functions in digital computer programs has presented a basic problem in the design of real-time control systems. Since there are usually no single instructions to perform such functions as sine x , arc sine x , etc., two other techniques have been used.

The first technique involves storing a table of values for the function in rapid access memory. The advantage of this technique lies in the speed obtainable; however, in order to provide reasonable accuracy, the table must be of considerable length. Computations which involve physical measurements such as those made by one of the newer radars would require a table which would occupy a significant portion of most storage devices. An alternate technique involves storing a smaller table, and then interpolating between the values stored in this table. The principal advantage of table storage is then lessened, for the interpolation routine requires time and the function can no longer be generated as quickly.

The second technique involves the use of a programmed approximation routine. Most of the routines used are based on polynomial approximations which require several multiplications, and as a consequence, are time-consuming.

The utilization of digital computers in real-time control systems which must process large amounts of data at high speeds has made the problem more acute. Most programs for such systems require values for sines, cosines, etc., quite often, and the necessary computations must be made quickly and with considerable accuracy. The need therefore exists for a very fast and accurate method of generating trigonometric functions.

Lincoln Laboratory has recently prepared a set of flexible computer programs which automatically perform the design of logical networks which will perform such functions in a single step. These networks may then be connected into the arithmetic element of an internal-binary-operation digital computer, providing the machine with instructions which will yield the sine, arc sine, etc., of an angle stored in one of the registers of the arithmetic element. The design technique yields a logical network with a number of input and output lines, the input lines representing the independent variable and the output lines the dependent variable. The networks are completely digital in operation, utilizing high-speed transistor and diode logical circuitry.

Because of the magnitude of the design problem for large logical networks, conventional manual design tech-

niques could not be used. For instance, the complete set of expressions needed to describe a 12-input bit and 14-output bit sine network in developed normal form contains approximately 90,000 symbols. Normal design techniques involving manual calculations would prohibit either the generation or simplification of a set of expressions of this complexity. It was necessary, therefore, to design and prepare a set of digital computer programs which would automatically generate the desired set of expressions, simplify them, and then check the simplified equations. The series of programs written are general purpose in design and may be used to generate and simplify the logical equations describing any function with a unique value of the dependent variable for each value of the independent variable.

Only three basic logical relationships or operators are used in the equations. These are defined as logical addition, logical multiplication, and complementation. Fig. 1 illustrates a standard diode circuit for the logical addition relationship. The inputs to this diode "or" circuit consist of dc levels of -5 and $+5$ volts corresponding to the logical symbols 0 and 1, respectively. To the left of the figure is the block diagram symbol for the "or" circuit. The plus sign in the block is also used to indicate logical addition in the written expressions. This is, of course, the "inclusive or" function, sometimes referred to as "logical disjunction."

Fig. 2 illustrates a diode "and" circuit which performs the logical multiplication function, sometimes referred to as "logical conjunction." The block diagram symbol for the "and" operation is to the left of this figure.

Fig. 3 illustrates a two-level diode "and-to-or" circuit. There are six inputs to this circuit, each consisting of dc levels. The expression describing this circuit in Boolean algebra notation is written at the far right of the figure. This expression may be read a and b and c , or d and e and f or the product of a , b and c , plus the product of d , e and f . According to the notation used, the output will be equal to 1 only when a and b and c or d and e and f are equal to 1.

The third logical operation used is the complement function, sometimes referred to as inversion or negation. A transistor inverter designed for the Lincoln TX-2 computer is illustrated in Fig. 4. A prime sign ($'$) is used to indicate the "complement" or "not" function in the expressions. When flip-flops are used to provide the inputs to the networks, complemented as well as the uncomplemented values of the input variables are available.

At a given time each input line to a logical network will contain a signal representative of either a binary 0 or 1. The fundamental characteristic of the logical net-

* The work reported here was performed at Lincoln Lab., Mass. Inst. Tech., Lexington, Mass., with the joint support of the Army, Navy, and Air Force.

† Lincoln Lab., M.I.T., Lexington, Mass.

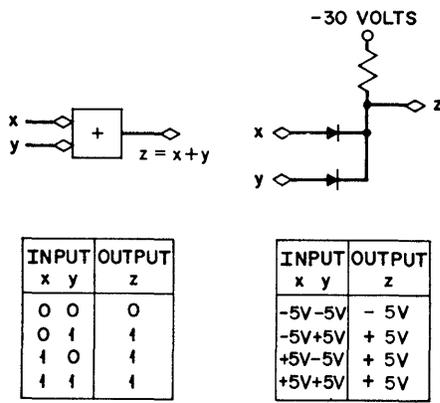


Fig. 1—The "or" circuit.

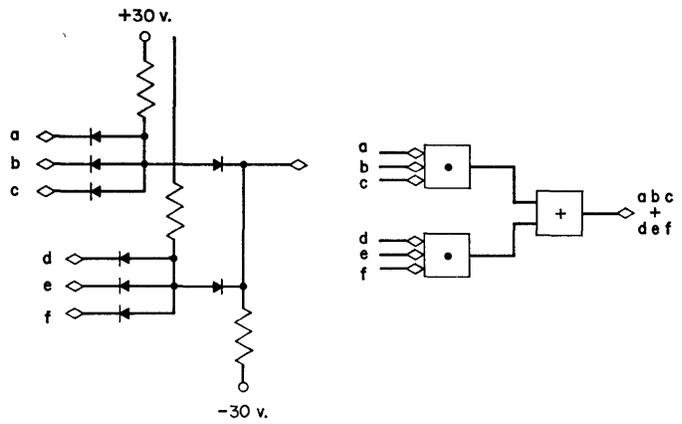


Fig. 3—Two-level "and-or" circuit.

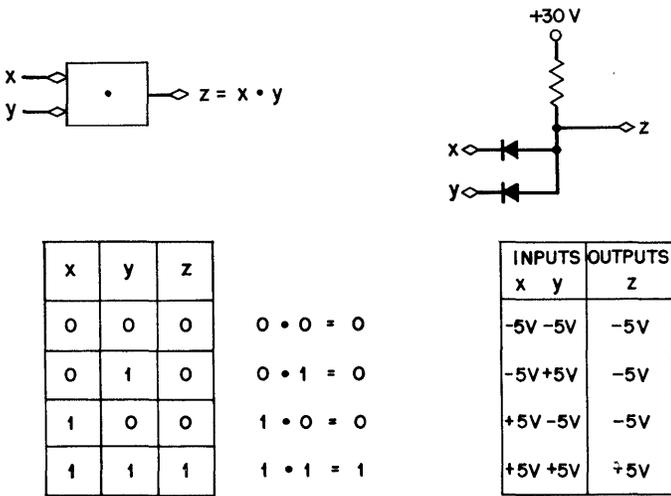


Fig. 2—Diode "and" circuit.

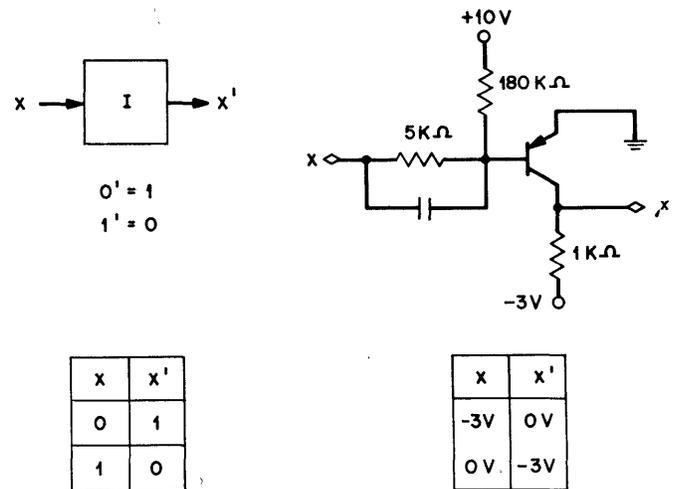


Fig. 4—Transistor inverter.

work is that for each combination of input values there will be a unique output value, represented by signals on the output lines. Since the output from a logical network of this sort is determined by the inputs to the logical network, it is possible to construct a logical network that will perform any function having the characteristic that for each state which the inputs assume, there will be a unique output. It may be seen that this does not limit the function performed to transcendental or straightforward mathematical relationships, but allows any input-to-output relationship.

When a logical network contains several input and output lines, it is possible to simplify the design problem by considering only one output line at a time. The signal on this particular line is determined by the values of the inputs to the logical network at that time, and may be specified by means of a Boolean function of the input variables. This expression will be equal to 1 when the output from this particular line of the network is 1 and to 0 when the output is 0. An expression that represents the operation of a single line of a logical network is called the transmission function for that line of the network. A fully developed transmission function, when written in the sum-of-products form, consists of a num-

ber of terms, each of which is a product of all of the input variables, certain of which may be complemented. This type of expression is referred to as a canonical expansion for the circuit transmission.

The first step in the design procedure for a network with several input and output lines consists of deriving the canonical expansion for each output line of the network. Table I shows the derivation of the canonical expansion for the transmission function for one output line bit of a small network which will yield the sine of x within the limits 0° to 90° . The leftmost column of the table lists the angle x in degrees, starting with 0° and increasing by increments of 6° to 90° . The next column contains the same set of angles coded as binary numbers, starting with 0000 and increasing to 1111 by steps of 0001, which increases the angle by 6° in each case. The values of sine x , expressed in binary form, are listed in the next column of the table. To the right of the column representing the sine values is a column listing the Boolean symbols for the input variables in product-term form. The variables are primed or unprimed depending on whether or not the respective input value is 0 or 1. The computer program generates the sine value for each input angle and then, examining one bit of the

TABLE I
CANONICAL EXPANSION FOR SIN x FROM 0° TO 90°

x (degrees)	x (binary form)				Sin x (binary form)				Product Terms
	a	b	c	d	(1)	(2)	(3)	(4)	
0	0	0	0	0	0	0	0	0	a' b' c' d'
12	0	0	1	0	0	0	1	1	a' b' c d'
18	0	0	1	1	0	1	0	0	a' b' c d
24	0	1	0	0	0	<u>1</u>	1	0	a' b c' d'
30	0	1	0	1	1	<u>0</u>	0	0	a' b c d
.
.
90	1	1	1	1	1	<u>1</u>	1	1	a b c d

$T(2) = a'b'cd + a'bc'd' + ab'c'd + ab'cd' + ab'cd + abc'd' + abc'd + abcd' + abcd$

TABLE II

DERIVATION OF PRIME IMPLICANTS
Theorem ($\phi\alpha + \phi\alpha' = \phi$)

First Cycle	Second Cycle	Third Cycle	First Cycle	Second Cycle	Third Cycle
a' b' c d	- b' c d		0 0 1 1	- 0 1 1	
a' b c' d'	- b c' d'		0 1 0 0	- 1 0 0	
a b' c' d	a b' - d	a - - d	1 0 0 1	1 0 - 1	1 - - 1
a b' c d'	a - c' d	a - c -	1 0 1 0	1 - 0 1	1 - 1 -
a b' c d	a b' c -	a b - -	1 0 1 1	1 0 1 -	1 1 - -
a b c' d'	a - c d'		1 1 0 0	1 - 1 0	
a b c' d	a - c d		1 1 0 1	1 - 1 1	
a b c d'	a b c' -		1 1 1 0	1 1 0 -	
a b c d	a b - d'		1 1 1 1	1 1 - 0	
	a b - d			1 1 - 1	
	a b c -			1 1 1 -	

sine values at a time, develops and stores the canonical expansion for each output line of the logical network. In effect, the computer proceeds down a column of output values until it finds a 1. It then stores the input values which generated the 1 output in the output table. Table I illustrates the derivation of the product terms for the second bit of the four output bits. Each 1 in the second-output-bit column is underlined, as is the product term for this particular output. The complete canonical expansion for the output line representing the second least significant output bit of this particular table is illustrated at the bottom of the figure. An expression is therefore generated for each output line of the logical network.

The minimization technique which has been programmed is based on work done by Quine^{1,2} of Harvard, and by McClusky³ during the preparation of his doctoral dissertation at M.I.T. The input to this particular program is the canonical expansion developed by the preceding program, and the output is a minimized equivalent sum-of-products expression.

¹ W. V. Quine, "The problem of simplifying truth functions," *Amer. Math. Monthly*, vol. 59, pp. 521-531; 1952.

² W. V. Quine, "A way to simplify truth functions," *Amer. Math. Monthly*, vol. 62, pp. 627-631; 1955.

³ E. J. McCluskey, Jr., "Minimization of Boolean functions," *Bell Sys. Tech. J.*, vol. 35, pp. 1417-1444; 1956.

The first step of the minimization procedure consists of matching each term of the canonical expansion with each of the other terms, and if the terms differ in only one variable, eliminating that variable. This matching procedure actually consists of the repeated application of the theorem ($\phi\alpha + \phi\alpha' = \phi$). Table II illustrates the first step of this matching procedure, using the product terms which comprise the expression developed in Table I. It is important that a given term be matched with all of the other terms and not dropped after the first match. For instance, $abc'd'$ in Table II must be matched with both $abc'd$ and $abcd'$. After the first set of matches, the resulting terms will each have one variable less than the original set of terms. Since, in Table II, the first set of terms consists of four variables each, the next col-

umn, containing the shortened terms, will consist of terms of three variables. In Table II the missing variables are indicated by dashes. If any term of the expression does not match with any other term, it is then a "prime implicant" term and will be a term of the final prime implicant expression.

After the original terms have all been compared, a second set of terms, each one variable shorter than the original terms, will have been derived. These terms are then matched with each other, using the same theorem $\phi\alpha + \phi\alpha' = \phi$. If the remaining variables are maintained in their original positions in the terms, and the eliminated variables indicated by means of a dash, as in Table II, the matching process may be made somewhat simpler to perform. In this case the terms are matched on the following basis: first, the dashes indicating the missing variable or variables must be in the same position (that is, $ab-d'$ and $ab-d$ may be matched but there is no possibility of matching $a-cd$ with $ab-d$) and second, the remaining variables must all be identical save one (that is, $ab-d'$ can be matched with $ab-d$, yielding $ab--$). This process is continued until no further matches can be made.

Third and further cycles of this process are continued, using the same rules, until a single pass through a cycle yields no matches. The remaining terms plus all the

terms that did not match during the process comprise the prime implicants.

The right half of Table II illustrates how the above matching process is performed in the computer. The terms of the original expression are stored in binary form, using positional notation to maintain the identity of the variables. A 1 is used to indicate an unprimed variable and a 0 to indicate a primed variable, so that $ab'cd$ is expressed as 1011. Since variables will be eliminated in this process, it is convenient to utilize another register of computer storage as a mask, and to alter the mask as variables are eliminated, while maintaining the variables in order. Each term is therefore represented by two registers, one containing the "values" of the variables and another the mask for the term. The mask is altered to indicate the eliminated variables. For instance, if two terms 1011 and 1010 and their masks are matched, the resulting term is 101- or $ab'c$. The first step in the matching process in the computer, therefore, is to compare the masks to see if they are identical. If the masks are identical, the terms are then matched and if they differ in only one variable, a new term is formed along with a new mask which indicates the missing variable or variables.

In order to shorten the matching problem, McCluskey has shown that the terms may first be sorted according to the number of 1's in each term. Table III illustrates the same set of terms after they have been sorted and arranged in tabular form. Each section of a column of the table contains terms with one more 1 than the terms in the preceding section of the table. It is necessary to match only the terms from one section of the table with the terms in the preceding and following sections of the table, for two terms which differ by more than one 1 cannot match: 1011 cannot match with any term containing only one 1, for instance 1000, for the two terms must, of necessity, vary in more than one variable. Further, if the terms from one section of the table are matched with those of the next section, the resulting shortened terms can be matched only with shortened terms formed by matches in the preceding and following sections of the table. This technique significantly reduces the number of matches which must be made and also materially lessens the amount of fast-access memory that is required at a given time. The number of terms formed by the matching process tends to increase considerably for large problems before finally decreasing. By storing only the sections of the table which are being matched in high-speed memory, and storing the rest of the terms on tape or drums, large problems may be handled more easily.

The set of terms derived in this manner, if collected in sum-of-products form, will form an expression equivalent to the original expression. An important characteristic of these terms is that none of them can be shortened by omitting a variable. Quine has shown that the shortest sum-of-products expression must consist of a subset of these terms. It may be shown that certain

TABLE III
ORDERED DERIVATION OF PRIME IMPLICANTS

First Cycle	Second Cycle	Third Cycle
0 1 0 0	- 1 0 0	1 - - 1
0 0 1 1	- 0 1 1	1 - 1 -
1 0 0 1	1 0 - 1	1 1 - -
1 0 1 0	1 - 0 1	
1 1 0 0	1 0 1 -	
1 0 1 1	1 - 1 0	
1 1 0 1	1 1 0 -	
1 1 1 0	1 1 - 0	
1 1 1 1	1 - 1 1	
	1 1 - 1	
	1 1 1 -	

Prime Implicants = (- 1 0 0 + - 0 1 1 + 1 - - 1 + 1 - 1 - + 1 1 - -)
or $(bc'd' + b'cd + ad + ac + ab)$

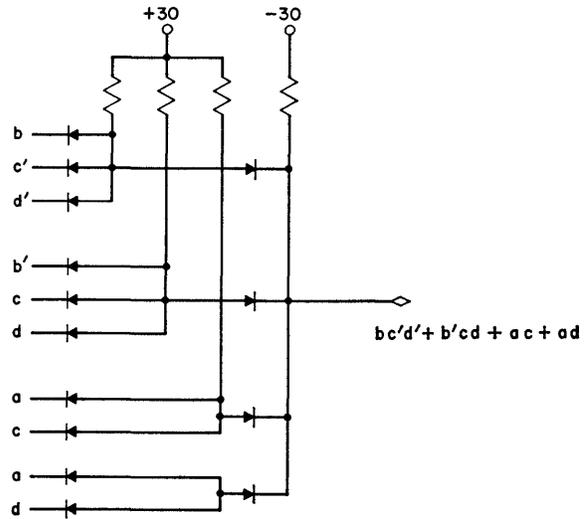


Fig. 5—Two-level diode circuit.

of these terms are common to every possible minimal subset, while the remaining terms of each subset may be chosen, subject to a set of constraints. The computer program has been designed to print automatically those prime implicant terms common to every minimum expression. The computer then prints a table of the remaining prime implicant terms along with the necessary constraint information. Choice of the remaining terms in the final expression is made from the table. This final choice has been left separate from the computer programs to permit flexibility in the design of the circuitry.

Fig. 5 illustrates a two-level diode circuit which performs the function developed in Table I. Fourteen diodes are required to construct this particular network. Since the original canonical expansion would have required 45 diodes, 31 diodes were saved by the minimization process.

The entire process was programmed for the Whirlwind computer located at the Massachusetts Institute of Technology. This is a general-purpose, stored-program computer with a word length of 16 bits. The entire set

of programs are slightly over 3000 orders in length. The program requires about 40 seconds to generate the canonical expansion for each output line of a 12 input-line by 14 output-line problem and from eight to ten minutes to minimize and print the final expression. About 25,000 registers are required to store partial results during the processing. As a result, drum storage is used during the minimization procedure. The procedure is now being programmed for an IBM 709 located at the laboratory, making possible the solution of larger problems and somewhat shortening the programs' running time due to the large core storage of the 709.

The design procedure described here appears very flexible. It can be used to perform automatically the logical design of circuitry which will perform any function which has a unique value of the dependent variable for each value of the independent variable.

To date, networks which yield sine, arc sine, and the square root of the input value have been constructed. The concept of programmed logic as an aid to computer design appears quite attractive for the design of future machines.

ACKNOWLEDGMENT

The computer programs described in this report were written by H. C. Peterson who has also contributed many helpful suggestions. The original problem of designing computer instructions which would yield transcendental operations arose during Lincoln Laboratories' ballistic missile early warning system studies and was suggested to the author by W. I. Wells. The author also wishes to thank V. A. Nedzel, R. W. Sittler, and J. F. Nolan for their helpful comments during the writing of this report.

The Role of Digital Computers in the Dynamic Optimization of Chemical Reactions

R. E. KALMAN[†] AND R. W. KOEPCKE[‡]

I. INTRODUCTION

ALONG with the increasing availability of high-speed, large-storage digital computers, there has been growing interest in their utilization for real-time control purposes. A typical problem in this connection and one of long-standing interest is the optimal static and dynamic operation of chemical reactors.^{1,2} To our knowledge, no digital computer is being used for this purpose, chiefly because of the many difficulties encountered in utilizing real-time machine computation in reactor control. These difficulties range from the unavailability or inadequacy of hardware (*i.e.*, transducers, measuring instruments, low-level analog-to-digital converters, etc.) to the lack of a well-established body of fundamental theoretical principles. Although a great deal is known about the basic concepts governing control systems,^{3,4} present methods cannot be readily applied to designing a program for a real-time digital con-

trol computer. This is because the existing design methods are applicable primarily to fairly small-scale systems, whereas the use of a digital computer (in fact the very attractiveness of computer control) arises primarily in connection with large-scale problems.

The role of the digital computer in real-time control consists essentially of "digesting" large amounts of information obtained from the primary measuring instruments and then calculating, as rapidly as possible, the control action to be taken on the basis of these measurements.

One purpose of this report is to provide a broad outline of a new approach to designing control systems for chemical processes which are to be built around a fast, general-purpose digital computer operating in real time. The specific engineering details of the computer will not be of any interest here; rather, we have concentrated on studying the types of computations the computer is to perform. To lend concreteness to the discussion, the chemical process under consideration will be a continuous-flow, stirred reactor. After the fundamental concepts have been established, the detailed analytic equations (in the linear case) leading to the dynamically optimal (and thus also statically optimal) design of the reaction control system are given in Section III. The equations of Section III represent a special case of the new design theory of linear control systems formulated

[†] Res. Inst. for Advanced Study, Baltimore 12, Md.

[‡] IBM Res. Center, Yorktown Heights, N. Y.

¹ T. J. Williams, "Chemical kinetics and the dynamics of chemical reactors," *Control Engrg.*, pp. 100-108; July, 1958.

² R. Aris and N. R. Amundson, "An analysis of chemical reactor stability and control," *Chem. Engrg. Sci.*, vol. 7, pp. 121-155; 1958.

³ J. G. Truxal, "Automatic Feedback Control System Synthesis," McGraw-Hill Book Co., Inc., New York, N. Y.; 1955.

⁴ J. R. Ragazzini and G. Franklin, "Sampled-Data Systems," McGraw-Hill Book Co., Inc., New York, N. Y.; 1958.

by the authors.^{5,6} The performance of the dynamically optimized control system is illustrated with the aid of a numerical example.

In Section IV the limitations of the linearity assumption or, rather, the additional steps necessary to attack realistic practical problems, are briefly discussed. It is impossible to give more than a rough sketch of these new methods in a short report; however, specific details, mathematical proofs, and discussion of engineering problems may be found in the literature.⁵⁻¹²

One of the mathematical tools used in the new approach has been called *dynamic programming* by its developer, Bellman.¹³ This is a new method for the solution of problems in the calculus of variations where *dynamic constraints* play the central role. It turns out that our new approach to the description of control-system dynamics leads to concepts which are also the "natural setting" for solving the optimization problem by dynamic programming. A second purpose of this report is to provide a better appreciation of the advantages as well as the limitations of dynamic programming, thereby promoting its use in the solution of engineering problems.

Perhaps the most outstanding advantage of the use of dynamic programming in our problem is that it reveals the intimate connection between the static and the dynamic optimization of the process. In other words, the problem of selecting the operating conditions of the process to obtain optimum yield or optimum product quality cannot be realistically divorced from the problem of providing effective regulation to maintain the process at these conditions. Although these matters are well known to workers skilled in the control art, they are often not clearly understood by others.

In addition to providing some practical means for the solution of reactor control problems, it is hoped that this report will help clarify a number of basic questions.

II. FUNDAMENTAL CONCEPTS

A. Description of Chemical Reactor

The continuous-flow, stirred-tank type of chemical reactor with which we shall be concerned here is shown in Fig. 1. The principal inputs to the reactor consist of liquid streams carrying the various raw materials. The volume-flow rates of the input streams in Fig. 1 are denoted by M_1, M_2, M_3 . Each stream carries one or more compounds, whose concentrations (measured in terms of moles/unit volume in Fig. 1) are denoted by U_1, \dots, U_5 . Other inputs to the reactor may include a catalyst stream (with flow rate M_4 in Fig. 1) and provisions for cooling or heating (with heat-flow rate M_5 in Fig. 1). The numbers X_1, \dots, X_n denote the concentrations of the various compounds inside the reactor (some of which come from the input streams and some of which are formed chemically inside the reactor); one of the X_i will denote the temperature of the material inside the reactor. Due to agitation, the concentrations of the various compounds as well as the temperature are assumed to be approximately the same at every point inside the reactor and in the output stream. In most cases, it is desirable to keep the amount of material in the reactor constant. This is achieved by means of a level controller which keeps the output stream (F_0 in Fig. 1) at all times approximately equal to $M_1 + \dots + M_4$.

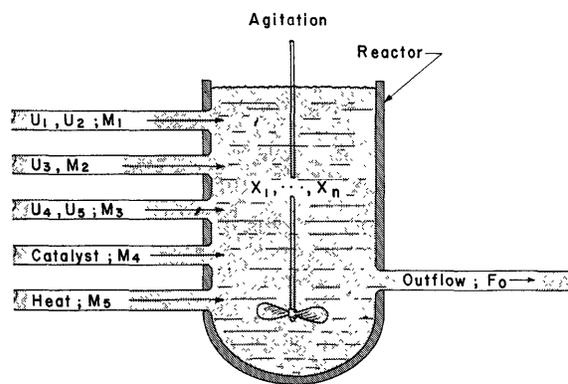


Fig. 1.

The object of the reactor is to produce a certain concentration of chemicals in the output streams. To accomplish this with the given types and concentrations of raw materials in the input streams, one can vary the flow rates M_1, \dots, M_5 . Since reactions take place more rapidly as the temperature increases, control can be exerted by changing the temperature in the reactor which, in turn, is achieved (subject to the dynamic lags of heat transfer to the reactor) by changing the heat-input flow-rate M_5 . The amount of catalyst present in the reactor also affects the reactions; the amount is controlled by changing M_4 (subject to a time constant = reactor volume/ F_0 if the amount of catalyst is not affected by the reaction). Similarly, some measure of con-

⁵ R. E. Kalman and R. W. Koepcke, "Optimal synthesis of linear sampling control systems using generalized performance indexes," *Trans. ASME*, vol. 80, pp. 1820-1826; 1958.

⁶ R. E. Kalman and R. W. Koepcke, "Dynamic optimization of linear control systems. I. Theory. II. Practical aspects and examples." (Scheduled for publication in the *IBM J. Res. Dev.*, vol. 4; 1960.)

⁷ R. E. Kalman and J. E. Bertram, "General synthesis procedure for computer control of single and multiloop linear systems," *Trans. AIEE*, vol. 77, pt. 2, pp. 602-609; 1958.

⁸ R. E. Kalman, "Optimal nonlinear compensation of saturating systems by intermittent action," 1957 IRE WESCON CONVENTION RECORD, pt. 4, pp. 130-135.

⁹ R. E. Kalman and J. E. Bertram, "A unified approach to the theory of sampling systems," *J. Franklin Inst.*, vol. 267, pp. 405-436; 1959.

¹⁰ R. E. Kalman, L. Lapidus, and E. Shapiro, "On the optimal control of dynamic chemical and petroleum processes." (Scheduled for publication in *Chem. Engrg. Progress*.)

¹¹ P. E. Sarachik, "Cross-coupled multi-dimensional feedback control systems," Ph.D. dissertation, Dept. of Elect. Engrg., Columbia University, New York, N. Y.; 1958.

¹² R. E. Kalman, "On the general theory of control systems," *Proc. Internat. Congr. on Automatic Control*, Moscow, U.S.S.R., Academic Press, New York, N. Y.; 1960.

¹³ R. E. Bellman, "Dynamic Programming," Princeton University, Press, Princeton, N. J.; 1957.

trol can be exerted by changing the flow rates M_1 , M_2 , M_3 ; the effect of these changes is complicated and depends on the reaction dynamics.

B. Statement of the Control Problem

The principal objectives in designing a reactor control system may be stated as follows:

Problem: Given the desired values X_1^d, \dots, X_n^d of the concentrations in the output stream at time t_0 , manipulate the control variables in such a manner as to bring rapidly the actual concentrations existing in the reactor at time t_0 as close as possible to the desired concentrations and then keep the actual concentrations constant at all times despite changes in the concentrations of the input streams, ambient temperature, etc. If, at time $t_1 > t_0$, the desired values of the concentrations are changed, the above process is repeated.

We now examine this problem in more detail. In doing so, we shall specify precisely what is to be meant by "as close as possible" and "rapidly."

C. Reaction Dynamics

Let us assume that p molecules of compound A and q molecules of compound B combine chemically to form a new compound C . If the concentrations X_A , X_B , X_C , of the various compounds are small, the rate of increase of the concentration of compound C is given by the well-known Arrhenius equation.^{1,2}

$$dX_C/dt = k_{AB}(T)X_A^pX_B^q. \quad (1)$$

In (1), the reaction rate coefficient is given by

$$k_{AB}(T) = \alpha_{AB} \exp(-E_{AB}/RT), \quad (2)$$

where α_{AB} is a constant, E_{AB} the activation energy of the reaction, T the absolute temperature, and R the gas constant. Moreover, the rate of decrease of the concentration of compounds A and B resulting from the reaction is equal to p resp. q times the right-hand side of (1).

In qualitative physical terms, the Arrhenius equation has the following interpretation. Consider a small volume with diameter equal to the effective range of intermolecular forces. If p molecules of A and q molecules of B have entered this small volume, a reaction takes place, but not otherwise. In a dilute solution, the probability of a molecule of some compound entering the small volume as a result of thermal agitation is proportional to the thermodynamic factor $\exp(-E_{AB}/RT)$ and the concentration of the compound, but independent of the concentration of the other compounds. The probabilities of independent events multiply, hence (1).

In general, the assumptions which lead to the particular form of (1) are not true, but the reaction rate is still a function of the temperature and concentrations. Thus, in general, one would replace (1) by

$$dX_C/dt = k_{AB}(X_A, X_B, X_C, T), \quad (3)$$

where k_{AB} is some scalar function of the four variables indicated.

It follows that the reaction shown in Fig. 1 can be described by the set of differential equations

$$dX_i/dt = f_i(X_1, \dots, X_n; M_1, \dots, M_l; U_1, \dots, U_k) \quad (4)$$

$$(i = 1, \dots, n; k, l, n = \text{integers}).$$

This is a good place, conceptually and in order to simplify the symbolism, to introduce vector-matrix notation. Thus, let X be a vector ($n \times 1$ matrix) with components X_1, \dots, X_n . Similarly, M and U are defined as a ($l \times 1$) and ($k \times 1$) matrix, respectively; f is a vector function of $k+l+n$ arguments with components f_1, \dots, f_n .

In terms of the new notation, (4) becomes

$$dX/dt = f(X, M, U). \quad (5)$$

The vector X is called the state of the reactor and the components of X are known as the state variables. The reason for this terminology is that if the reactor inputs $M(t)$ and $U(t)$ are specified for all time $t \geq t_0$, then the knowledge of $X(t_0)$ supplies the initial conditions from which the solutions of the differential equation (5) can be uniquely determined (subject to some mild mathematical restrictions) for all future values of time. Thus, the state is a fundamental mathematical concept for describing the reactor dynamics; it is also a physical concept. The temperature and various concentrations can be physically measured (at least in principle); thus the state at time t_0 may be regarded as the information necessary to determine the properties of the material inside the reactor at time t_0 .

The behavior of the reactor through time may be visualized as a succession of changes in state. This gives rise to the concept of the state-transition function. In fact, the function f in the differential equation (5) may be regarded as specifying the incremental state transitions taking place during the interval $(t, t+dt)$. For present purposes, it is more convenient to deal with finite-interval state transitions which are obtained by solving the differential equations. Anticipating the later discussion, let us note that for control purposes it is sufficient to sample the state of the process; *i.e.*, observe the state only at discrete instants in time, called sampling instants. Usually, the sampling instants are separated by equal intervals τ of time (τ is called the sampling period), *i.e.*, the sampling instants occur at times

$$t_0, t_0 + \tau, t_0 + 2\tau, \dots$$

Now suppose that τ is chosen to be so small that in the interval $(t_0, t_0 + \tau)$ the functions $M(t)$, $U(t)$ in (5) may be adequately approximated by the constants $M(t_0)$, $U(t_0)$. Then (5) can be readily integrated (if necessary, by numerical methods) and we get

$$X(t_0 + \tau) = \phi(\tau; X(t_0), M(t_0), U(t_0)), \quad (6)$$

where ϕ is a vector function with n components and $k+l+n$ arguments.

D. Static Optimization

Precisely what is meant by the phrase, "as close as possible to the desired concentrations" in the statement of the basic problem in Section II-B?

The states of the reactor may be represented as points in n -dimensional Euclidean space (the state variables being coordinates of the point) called the *state space*. Suppose we specify r components of the state vector as desired values, with the remaining $n-r$ components being arbitrary. This step may be regarded as essentially a management decision, relating to the question of how one should try to operate the reaction process. The set of states for which the operation of the reactor meets the management requirements is clearly an $(n-r)$ -dimensional hyperplane. If the state of the reactor at any instant of time does not lie in the hyperplane, we can measure the "badness" of that state by the distance of the state from the hyperplane of desired states (see Fig. 2). The definition of the distance function (technically, a pseudo-metric) is arbitrary and depends on a management estimate as to what types of deviations from the desired values are more harmful than others. One possible definition of the distance function is

$$\rho(X^d - X) = \left[\sum_{i=1}^r (X_i^d - X_i)^2 \right]^{1/2} \quad (r < n). \quad (7)$$

More generally, if Q is any positive semidefinite matrix, we can define ρ by the quadratic form

$$\rho(X^d - X) = (X^d - X)'Q(X^d - X), \quad (8)$$

where the prime denotes the transpose of the matrix.

By static optimization of the process we mean selecting a set of constant values M^0 of the control variables (subject to some magnitude constraints), so that at equilibrium the actual state lies as close as possible to the hyperplane of desired states. By definition, the equilibrium states X^* of the reactor are given by:

$$dX/dt = f(X^*, M, U) = 0, \quad (9)$$

M, U being constant vectors. Thus the statically optimal control vector M^0 and equilibrium state X^{*0} are determined by solving the minimization problem

$$\text{Min}_M \rho(X^d - X^*), \quad 0 \leq M_i \leq \mu_i. \quad (10)$$

To find the optimal control vector M^0 from (10), X^* has to be expressed as an explicit function of M from (9). This and the amplitude constraints on the control variables lead to great analytic difficulties when f is a nonlinear function. But even in cases where the static optimization problem can be solved, it does not provide a complete answer to the basic problem. This is because:

1) Static optimization does not provide a guide as to how the control variables should be manipulated to bring an arbitrary state as close as possible (in terms of the arbitrarily adopted distance function) to the desired state (dynamic optimization).

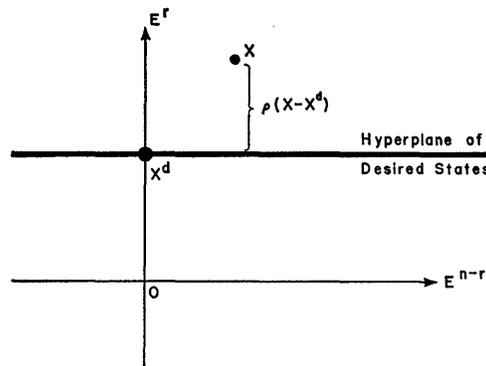


Fig. 2.

2) The equilibrium state closest to the hyperplane of desired states may not be stable.

3) The values of the control variables computed by static optimization will not remain optimal when some of the process parameters (concentrations in the input flows, ambient temperature, etc.) change. In other words, static optimization does not incorporate the important principle of feedback.

In the following it is shown that it is possible to combine both dynamic and static optimization in such a way that the principle of feedback is retained.

E. Dynamic Optimization

In our basic problem statement in Section II-B, the last remaining word to be defined precisely is "rapidly."

A performance index for the reaction under dynamic conditions may be defined as

$$\Phi[X(t_0)] = \int_{t_0}^{\infty} \rho[X^d - X(t)] \exp[\alpha(t - t_0)] dt, \quad (11)$$

where α is a real constant. We now agree that the phrase, "... to bring rapidly the actual concentration as close as possible to the desired concentrations ..." in the problem statement means that the control variables $M(t)$ are to be chosen, as functions of time, in such a way as to minimize the performance index (11) for any initial state $X(t_0)$. This is called dynamic optimization. Of course, the definition of Φ , in particular the value of α in (11), is arbitrary and depends on management estimates just as the definitions of X^d and ρ .

Static optimization is evidently a special case of dynamic optimization, as may be seen by setting $X(t_0) = X^{*0}$ in (11). In fact, it may happen that the dynamic optimization leads to the result that, instead of trying to maintain the control variables at constant (equilibrium) values, it is better to vary the control variables continuously, say, in a periodic fashion. In such a case, dynamic optimization will lead to a smaller value of $\Phi(X^{*0})$ than static optimization.

In order to perform dynamic optimization, we must find a particular vector function $M^0(t)$, defined for all $t \geq t_0$, among the set of all such functions (subject to amplitude constraints) for which the integral (11)

assumes its minimum or least upper bound. This is generally a very difficult problem in the calculus of variations and, for all practical purposes, cannot be solved by conventional analytic methods when the number of state variables is large.

So as not to be bothered by certain mathematical niceties, we shall assume from here on that the control variables have constant values over the sampling intervals τ (cf. Section II-C).

Thus instead of minimizing Φ with respect to all possible functions $M(t)$, the minimization is to be performed with respect to all possible sequences of constant vectors

$$M(t_0), M(t_0 + \tau), M(t_0 + 2\tau), \dots \quad (0 \leq M_i(t) \leq \mu_i). \quad (12)$$

Moreover, again for simplicity, the integral in (11) may be replaced by a sum:

$$\Phi[X(t_0)] = \sum_{k=1}^{\infty} \rho [X^d - X(t_0 + k\tau)] \lambda^k, \quad (13)$$

where $\lambda = \exp \alpha\tau$.

From (6) we see that there is a large number of possible state transformations $X(t_0) \rightarrow X(t_0 + \tau)$, depending on the choice of $M(t_0)$ (assuming that $U = \text{const}$). Similarly, the state transformation $X(t_0 + \tau) \rightarrow X(t_0 + 2\tau)$ depends on the choice of $M(t_0 + \tau)$ (see Fig. 3). Thus the minimization of Φ may be regarded as an infinite-step decision procedure. The optimal choice of $M(t_0 + k\tau)$ at the k th step in general depends both on the preceding and succeeding steps. Therefore, at first sight, it would appear that to obtain the optimal sequence of control vectors,

$$M^0(t_0), M^0(t_0 + \tau), M^0(t_0 + 2\tau), \dots \quad (14)$$

we must simultaneously minimize (13) with respect to all the terms of the sequence (12), which is an impossible job.

Fortunately, at this point we can achieve a decisive simplification by making use of the following intuitively obvious, but powerful, observation due to Bellman.¹³

Principle of Optimality: An optimal sequence of control variables (14) has the property that, whatever the initial state $X(t_0)$ and the initial choice $M^0(t_0)$ of control vector are, the remaining terms $M^0(t_0 + \tau), M^0(t_0 + 2\tau), \dots$ of (14) must constitute an optimal sequence with regard to the state $X(t_0 + \tau)$ resulting from the choice of $M^0(t_0)$.

Using the principle of optimality, we can obtain various expressions for the theoretical study and practical determination of the optimal control sequence (14). (Methods derived from the principle of optimality are known by the generic name of dynamic programming.)

We first observe that (13) can be written in the form:

$$\begin{aligned} \Phi[X(t_0)] &= \lambda \rho [X^d - X(t_0 + \tau)] \\ &+ \sum_{k=2}^{\infty} \rho [X^d - X(t_0 + k\tau)] \lambda^k \\ &= \lambda \{ \rho [X^d - X(t_0 + \tau)] + \Phi[X(t_0 + \tau)] \}. \end{aligned} \quad (15)$$

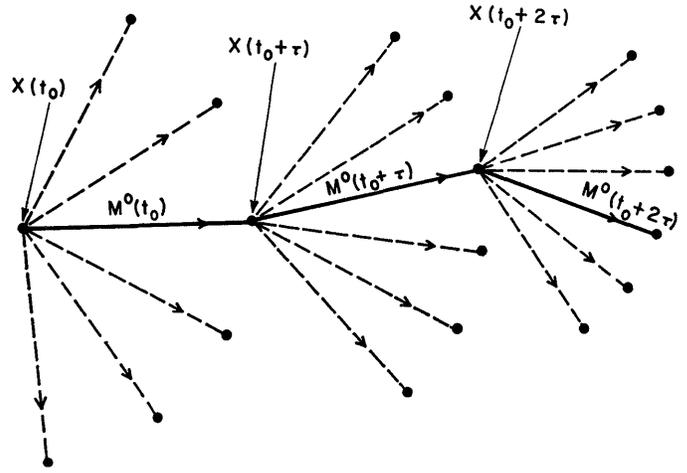


Fig. 3.

Now let $\Phi^0(X(t_0))$ be the value of the performance index when the optimal sequence (14) is used. Substituting (15) and invoking the principle of optimality, we obtain a functional equation (13) for Φ^0 ; *i.e.*,

$$\begin{aligned} \Phi^0[X(t_0)] &= \text{Min}_{M(t_0), M(t_0 + \tau), \dots} \Phi[X(t_0)] \\ &= \text{Min}_{M(t_0)} \lambda \{ \rho [X^d - X(t_0 + \tau)] + \Phi^0[X(t_0 + \tau)] \}. \end{aligned} \quad (16)$$

Note carefully that the right-hand side of (16) is a function of $M(t_0)$ and $X(t_0)$ through (6). The solution of the functional equation (16) determines $M^0(t_0)$ as some function of $X(t_0)$ which may be denoted by

$$M^0(t_0) = h(X(t_0)). \quad (17)$$

Now at time $t_0 + \tau$ we are confronted by the same decision problem as at time t_0 . This shows that $\Phi^0[X(t_0 + \tau)]$ also satisfies the functional equation (16), and therefore $M^0(t_0 + \tau)$ is the same function of the state at time $t_0 + \tau$ as $M^0(t_0)$ was at time t_0 . Thus we have arrived at the following result.

If the performance of a dynamic system governed by (6) is optimal in the sense that the performance index (13) is a minimum, then the sequence of optimal control variables is obtained by observing the state of the system at times

$$t_0, t_0 + \tau, t_0 + 2\tau, \dots$$

and computing the optimal control variables at each sampling instant by means of the formula

$$M^0(t_0 + k\tau) = h(X(t_0 + k\tau)), \quad (18)$$

h being determined by solving (16).

Eq. (18) shows that the feedback principle can be included in the framework of dynamic optimization. This means that the entire future evolution of the dynamic system, including the values of the optimal control variables at each sampling instant, could be predicted in principle by means of (6) and (18). However, because of the inaccurate knowledge of the state-transition function, unknown disturbances acting on the process, and random effects such as turbulence, etc., the prediction

based on (6) will be less and less correct as the prediction interval increases. By re-measuring the state of the system at the sampling instants [assuming that τ has been chosen small enough so that the one-step prediction based on (6) is sufficiently accurate], the prediction errors are corrected so that the control variables assume very nearly their optimal values at all times. This is the conventional use of feedback. As is well known, feedback will also tend to minimize the sensitivity of h to variations in ϕ .

To solve the functional equation (16), it is often convenient to use an iterative procedure. To derive the iteration scheme, we replace the performance index (13) by

$$\mathcal{P}_N[X(t_0)] = \sum_{k=1}^N \rho[X^d - X(t_0 + k\tau)]\lambda^k. \quad (19)$$

In other words, the original infinite-step decision process is converted into a finite-step decision process. Proceeding exactly as in the derivation of (16), we find that the successive optimal performance indexes \mathcal{P}_N^0 are connected by the recurrence relations:

$$\left. \begin{aligned} \mathcal{P}_1^0[X(t_0)] &= \text{Min}_{M(t_0)} \lambda \rho[X^d - X(t_0 + \tau)] \\ \mathcal{P}_{N+1}^0[X(t_0)] &= \text{Min}_{M(t_0)} \lambda \{ \rho[X^d - X(t_0 + \tau)] + \mathcal{P}_N^0[X(t_0 + \tau)] \} \end{aligned} \right\} 0 \leq M_i(t_0) \leq \mu_i. \quad (20)$$

In each stage of the iteration (20), the optimal control signal $M^0(t_0)$ is determined as some function h_N of $X(t_0)$. As $N \rightarrow \infty$, it can be shown under various restrictions^{6,13} that \mathcal{P}_N^0 converges to \mathcal{P}^0 , and h_N converges to h .

III. DYNAMIC PROGRAMMING IN THE LINEAR CASE

The ease or difficulty of carrying out iterations (20) is determined largely by the complexity of the dynamics of the reaction and by the limits imposed on the control variables. To illustrate these computations concretely, we consider now the very special (but practically important) linear case where

- 1) The reaction dynamics are governed by an ordinary linear differential equation with constant coefficients
- 2) There are no amplitude constraints on the control variables.

Linear differential equations arise when the dynamic equations (6) are linearized about some equilibrium state X^* and the corresponding values of the control variables M^* . If we let

$$X = X^* + x, \quad M = M^* + m, \quad \text{and} \quad X^d = X^* + x^d, \quad (21)$$

and, if the deviations x, m from the equilibrium values are sufficiently small, (6) leads to the linear differential equation with constant coefficients

$$dx/dt = Fx + Dm, \quad (22)$$

where F is a constant $n \times n$ matrix and D is a constant $n \times l$ matrix. The elements of these matrices are de-

termined from (6) by means of the formulas (assuming $U = \text{const}$)

$$\begin{aligned} F_{ij} &= \partial f_i / \partial X_j |_{X=X^*, M=M^*} \dots \quad (i, j = 1, \dots, n) \\ D_{ij} &= \partial f_i / \partial M_j |_{X=X^*, M=M^*} \\ &\quad (i = 1, \dots, n; j = 1, \dots, l). \end{aligned} \quad (23)$$

As is well known,¹⁴ the solution of the differential equation (22) has the form:

$$x(t) = \Phi(t - t_0) x(t_0) + \int_{t_0}^t \Phi(t - \tau) Dm(\tau) d\tau \quad (24)$$

for any t, t_0 . The matrix $\Phi(\tau)$ is called the *transition matrix* of the system (22) and is given by

$$\Phi(\tau) = \exp F\tau = \sum_{k=0}^{\infty} F^k \tau^k / k! \quad (25)$$

The Taylor series is a convenient way of calculating numerical values of $\Phi(\tau)$ when there are a large number of state variables and when a digital computer is available. There are also analytic ways of computing $\Phi(\tau)$.^{5,7}

When $m(t)$ is constant during the intervals between sampling instants, (24) takes the simpler form

$$x(t_0 + \tau) = \Phi(\tau) x(t_0) + \Delta(t)m(t_0), \quad (26)$$

where

$$\Delta(\tau) = \int_0^\tau \Phi(\tau - \sigma) D d\sigma. \quad (27)$$

Eq. (26) is the explicit form of (6) in the linear case.

We now give a formal derivation of the explicit equations for accomplishing the iterations indicated by (20). The various formal steps of the derivation can be justified under mild mathematical restrictions.⁶

If ρ is given by (8) and \mathcal{P} by (13), it can be shown by induction that the optimal performance index may be written in the form

$$\begin{aligned} \mathcal{P}_N^0[x(t_0)] &= x'(t_0) P_N x(t_0) - 2x'(t_0) R_N x^d \\ &\quad + x^d S_N x^d \quad (N \geq 0) \end{aligned} \quad (28)$$

P_N, R_N, S_N being $n \times n, n \times l$, and $l \times l$ matrices, respectively, and

$$P_0 = R_0 = S_0 = 0.$$

For simplicity, we now drop the arguments of $\Phi(\tau)$ and $\Delta(\tau)$. Using (20) and (26), we calculate the deriva-

¹⁴ E. A. Coddington and N. Levinson, "Theory of Ordinary Differential Equations," McGraw-Hill Book Co., Inc., New York, N. Y., ch. 3; 1955.

tive of the scalar $\Phi_{N+1}[x(t_0)]$ with respect to the vector $m(t_0)$. This is a vector (with components $\partial\Phi_{N+1}/\partial m_i(t_0)$), which is given by:

$$\partial\Phi_{N+1}/\partial m(t_0) = -2[\Delta'(R_N + Q)x^d - \Delta'(P_N + Q)(\Phi x(t_0) + \Delta m(t_0))] \quad (N \geq 0). \quad (29)$$

Now $\Phi_{N+1}[x(t_0)]$ is evidently a quadratic function of each of the incremental control variables $m_i(t_0)$. It follows that Φ_{N+1} has a single extremal value [which may be a minimum or a maximum, depending on the value of $x(t_0)$] at that value of $m(t_0)$ which makes the right-hand side of (29) zero. It can be shown⁶ that the extremal value is a minimum for every $x(t_0)$. Hence, $m(t_0)$ is found by setting (29) equal to zero, which yields the following expressions for $m^0(t_0)$ and the matrices defining Φ_{N+1} :

$$m^0(t_0) = -A_N x(t_0) + B_N x^d, \quad (30)$$

where

$$\left. \begin{aligned} A_N &= [\Delta'(P_N + Q)\Delta]^{-1}(P_N + Q)\Phi \\ B_N &= [\Delta'(P_N + Q)\Delta]^{-1}(R_N + Q) \end{aligned} \right\}. \quad (31)$$

With some further calculations, using (30) and (31) we find that:

$$P_{N+1} = \lambda(\Phi - \Delta A_N)'(P_N + Q)\Phi; \quad (32a)$$

$$R_{N+1} = \lambda(\Phi - \Delta A_N)'(R_N + Q); \quad (32b)$$

and

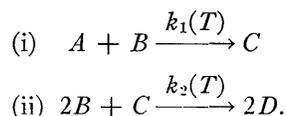
$$S_{N+1} = \lambda(S_N + Q - B_N'\Delta'(R_N + Q)). \quad (32c)$$

The iterations indicated by (32) can be readily performed on a digital computer. Note that S_N need not be computed if only the optimal control vectors are of interest. In the limit $N \rightarrow \infty$, all quantities in (32), except S_{N+1} , may be shown to converge under certain restrictions on F , D , and λ .⁶

We get by inspection of (30) the important result: *In the linear case, the optimal control variables are linear functions of the actual and desired states of the reactor.*

Since the control variables are linear functions of the state variables, it follows that under closed-loop control the reactor is a linear dynamic system. It can be shown⁶ that the only possible type of limiting behavior in such systems as $t \rightarrow \infty$ is for the state $X(t)$ to converge to an equilibrium state X^* . Since dynamic optimization includes static optimization, it follows at once that: *In the linear case, the states of a dynamically optimized system tend asymptotically to the same equilibrium state X^* which is obtained under static optimization.*

Example: As a numerical illustration of the results obtained by the use of dynamic programming in the linear case, let us consider the following hypothetical reactions:



The objective is to convert raw materials A and B by means of reaction (i) into C obtaining as much quantity of C as possible. The optimization of the process is complicated by the undesired side reaction (ii) which produces the contamination product D . Under steady-state conditions, the resulting concentrations in the outflow as a function of the "hold-up" time (reactor volume/outflow rate) will have the qualitative shape shown in Fig. 4.

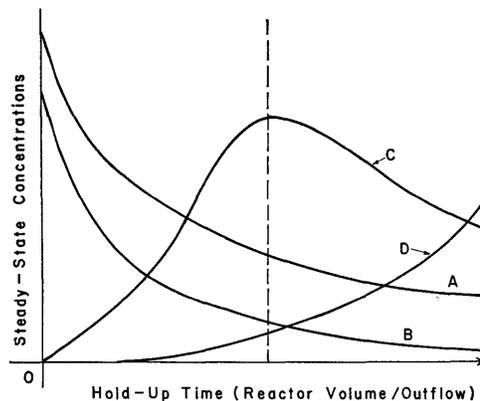


Fig. 4.

We now derive the analytical form of the dynamic equations of the reactor using the assumption that the Arrhenius equation (1) holds. Denoting the concentrations of A, \dots, D by X_1, \dots, X_4 , and the flow rates of A and B by M_1, M_2 we find, using conservation of mass, that:

$$\begin{aligned} dX_1/dt &= -k_1(T)X_1X_2 + (M_1/V)U_1 \\ &\quad - [(M_1 + M_2)/V]X_1; \\ dX_2/dt &= -k_1(T)X_1X_2 - 2k_2(T)X_2^2X_3 + (M_2/V)U_2 \\ &\quad - [(M_1 + M_2)/V]X_2; \\ dX_3/dt &= k_1(T)X_1X_2 - k_2(T)X_2^2X_3 \\ &\quad - [(M_1 + M_2)/V]X_3; \end{aligned} \quad (33)$$

and

$$dX_4/dt = 2k_1(T)X_2^2X_3 - [(M_1 + M_2)/V]X_4.$$

Let T_1 and T_2 denote the temperatures of the input flows M_1, M_2 ; let T_c be the average cooling water temperature inside the cooling coils of the reactor; and let h be the corresponding average heat transfer coefficient per unit cooling water flow. Furthermore, let H_1 be the heat generated per molecule of the first reaction; H_2 the heat generated per molecule of the second reaction; ρ the average density of the material in the reactor; and c the average heat capacity of the material. Denoting the temperature in the reactor by X_5 and the cooling-water flow rate by M_5 , conservation of energy yields

$$\begin{aligned} dX_5/dt &= k_1(T)X_1X_2H_1 + k_2(T)X_2^2X_3H_2 \\ &\quad + (M_1/V\rho c)(T_1 - X_5) + (M_2/V\rho c)(T_2 - X_5) \\ &\quad + (h/V\rho c)M_5(T_c - X_5). \end{aligned} \quad (34)$$

At equilibrium, the variables entering (33) and (34) are assumed to have the values shown in Table I. Note that the first reaction is assumed to be exothermic, and the second is assumed to be endothermic.

Using these values, (23) yields the following numerical values for the matrices describing the dynamics of the reactor in the vicinity of equilibrium:

$$F = \begin{bmatrix} -0.325 & -0.5625 & 0 & 0 & -0.200 \\ -0.225 & -0.8125 & -0.014286 & 0 & -0.368 \\ 0.225 & 0.4875 & -0.107143 & 0 & 0.116 \\ 0 & 0.1500 & 0.014286 & -0.1 & 0.168 \\ 0.450 & 0.7500 & -0.035714 & 0 & -0.060 \end{bmatrix} \quad (35)$$

Assuming that only the flow rates M_1 and M_3 can be changed to effect control, we get:

$$D = \begin{bmatrix} 55 & 0 & 0 \\ -4 & 0 & 0 \\ -21 & 0 & 0 \\ -3 & 0 & 0 \\ -2 & 0 & 35.5 \end{bmatrix} \quad (36)$$

Using a sampling period $\tau=1$, the transition matrix for the linear system can be obtained using the Taylor series (25).

$$\Phi(\tau) = \begin{bmatrix} 0.7407 & -0.3581 & 0.00498 & 0 & -0.0909 \\ -0.1793 & 0.4095 & -0.00448 & 0 & -0.2197 \\ 0.1566 & 0.2895 & 0.89510 & 0 & 0.0264 \\ 0.0151 & 0.1373 & 0.00946 & 0.9048 & 0.1288 \\ 0.3004 & 0.3872 & -0.03460 & 0 & 0.8172 \end{bmatrix} \quad (37)$$

$$\Delta(\tau) = \begin{bmatrix} 48.216 & 0 & -2.169 \\ -7.695 & 0 & -4.682 \\ -15.744 & 0 & 0.9124 \\ -3.089 & 0 & 2.5132 \\ 7.056 & 0 & 32.822 \end{bmatrix} \quad (38)$$

The performance index is defined as:

$$\mathcal{P} = \sum_{k=1}^{\infty} [x_3^d - x_3(t_0 + k)]^2 + [x_4^d - x_4(t_0 + k)]^2 \quad (39)$$

The optimal control variables for this performance index are given by the following functions of the desired and actual state of the reactor:

$$m = \begin{bmatrix} 0 & 0 & -0.0680 & 0.0217 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0788 & 0.3932 & 0 \end{bmatrix} x^d + \begin{bmatrix} 0.0102 & 0.0164 & 0.0605 & -0.0197 & -0.0012 \\ 0 & 0 & 0 & 0 & 0 \\ 0.0064 & -0.0327 & 0.0668 & -0.3537 & -0.0504 \end{bmatrix} x \quad (40)$$

TABLE I
VALUES OF REACTOR CONSTANTS

U_1	U_2	M_1^*	M_2^*	M_3^*	X_1^*	X_2^*	X_3^*	X_4^*	X_5^*
65	59	0.05	0.05	0.1	10	4	21	3	120
$k_1(X_5^*)$		$k_2(X_5^*)$		$\partial k_1(X_5^*)/\partial X_5$		$k_2(X_5^*)/\partial X_5$			
0.05625		0.00044643		0.005		0.00025			
T_1	T_2	T_c	H_1	H_2	V	$h/V\rho c$	ρc		
100	100	49	2	-5	1	0.5	10		

To improve the operation of the reactor, it is desirable to increase the yield of C and cut down the yield of D . Therefore, the desired state of the reactor may be defined as:

$$x_3^d = 5 \text{ and } x_4^d = -1. \quad (41)$$

If the reactor starts out at the old equilibrium state ($x_1 = x_2 = \dots = x_5 = m_1 = \dots = m_3 = 0$) at time $t_0 = 0$, the behavior of the state and control variables as a function of time will be as shown in Fig. 5. It is evident from Fig. 5 that it is possible to achieve almost exactly the new desired state and that the new equilibrium state can be reached rather quickly. It should be noted, however, that the results are valid only if the linearized approximation of the dynamics is valid.

IV. LIMITATIONS OF THE LINEARITY ASSUMPTION

There are a large number of problems which must be considered before fully automatic dynamic optimization of chemical reactions can take place.

1) If state variables are not physically measurable, they must be generated artificially in order to be able to compute the optimal values of the control variables.

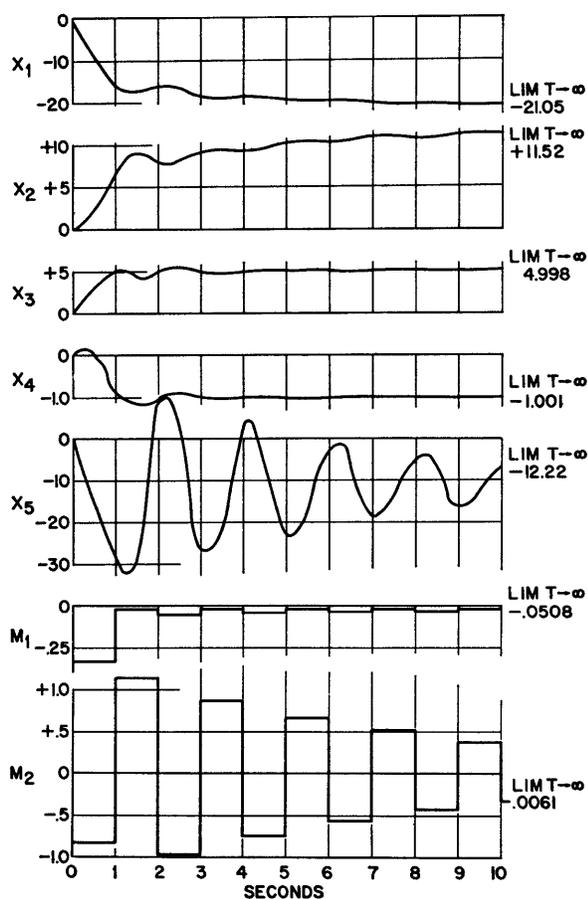


Fig. 5.

This calls for simulating some of the reaction dynamics as an integral part of the control system. This, too, can be done by means of a digital computer. (Most of the analog-type control instruments used at present may be thought of as performing essentially this function.)

2) The dynamic programming equations can be solved, practically speaking, only in the linear case. In reality, of course, the reaction dynamics are nonlinear. Moreover, they may change with time due to uncontrollable or unknown effects. There are essentially two possibilities of attacking these problems.

a) The reaction dynamics are linearized over a certain region in state space. The reaction is then optimized on a linear basis, computing the dynamic programming equations in real time. If, as a result of this optimization, the state moves into another region of the state space, another set of linearized equations is obtained to describe the dynamics in the new region. These equations are then used to obtain a new dynamic optimization, etc. This method of attack is closely related to the problem of designing adaptive or self-optimizing systems¹⁵ about which little is known at present. The chief difficulty is

the rapid and accurate determination of the linear dynamics in the presence of measurement noise.

b) The dynamic optimization is solved directly by purely numerical methods. The chief difficulty encountered here is the experimental measurement and representation of the reaction dynamics in a nonlinear form. Very little is known about this problem at present.

c) The control variables cannot be chosen freely but must lie within certain prescribed ranges; in other words, the control variables "saturate." The problem of designing a control system where the dynamic equations of the control object are linear but where the control variables saturate has an extensive literature usually under the subject heading of "Optimal Relay Servo Problem." At present this problem is solved only in the case where 1) the dynamic equations are of the second order and 2) there is only one control variable.¹⁶ Using the point of view of this paper, a rigorous method was recently obtained (which is not subject to the above restrictions, 1 and 2)⁸ for the computation of the optimal control variables; however, this method is very inefficient. When the control object has nonlinear dynamics, no method of computing the optimal control variables is known.

Despite these obstacles, much progress can be expected from the utilization of the "state" method of describing reaction dynamics combined with dynamic optimization as presented in this paper. These new ideas will probably be most helpful in attempting to control (by means of real-time digital computation) dynamic systems which have many state variables.

LIST OF PRINCIPAL QUANTITIES

Sections II-A and II-B

U ; U_i = vector denoting concentrations in input streams; its components.

M ; M_i = control vector; control variables.

l = number of control variables.

T = temperature.

X ; X_i = state vector; state variables (concentrations and temperature inside reactor).

n = number of state variables.

t ; t_0 = time; initial time.

Section II-C

f ; f_i = infinitesimal state transition function; its components.

τ = sampling period.

ϕ ; ϕ_i = (finite-interval) state transition function; its components.

¹⁵ R. E. Kalman, "Design of a self-optimizing system," *Trans. ASME*, vol. 80, pp. 468-478; 1958.

¹⁶ R. E. Kalman, "Analysis and design principles of second and higher-order saturating servomechanisms," App. 2, *Trans. AIEE*, vol. 24, pt. 2, pp. 294-310; 1955.

Section II-D

- ρ = distance function in state space (pseudo-metric).
 $'$ = transpose of the matrix.
 Q = positive semidefinite matrix.
 $()^*$ = equilibrium values.
 $()^0$ = optimal values.

Section II-E

- \mathcal{P} = performance index.
 $h; h_i$ = optimal control function.

Section III.

- $x; x_i$ = incremental state vector; incremental state variables.
 $m; m_i$ = incremental control vector; incremental control variables.
 F = infinitesimal transition matrix in linear case.
 D = matrix denoting instantaneous effect of control variables in linear case.
 $\Phi(\tau)$ = (finite-interval) transition matrix in linear case.
 $\Delta(\tau)$ = matrix denoting effect of control variables in linear case (finite-interval).
 A_N, B_N, P_N, R_N, S_N constant matrices.

Simulation of Human Problem-Solving

W. G. BOURICIUS† AND J. M. KELLER†

SIMULATING human problem-solving on a digital computer looks deceptively simple. All one must do is program computers to solve problems in such a manner that the computer employs the identical strategies and tactics that humans do. This will probably prove to be as simple in theory and as hard in actual practice as was the development of reliable digital computers. One of the purposes of this paper is to describe a few of the pitfalls that seem to lie in the path of anyone trying to program machines to "think."

The first pitfall lies in the choice of an experimental problem. Naturally enough the problem chosen should be of the appropriate degree of difficulty, not so difficult that it cannot be done, and not so trivial that nothing is learned. It should also involve symbology and manipulations capable of being handled by digital computers. At this stage of problem consideration, a devious form of reasoning begins to operate. Usually the people engaged in this type of research will have had a thorough grounding in conventional problem-solving on computers. Consequently, they are conversant with the full range of capabilities of computers and have an appreciation of their great speed, reliability, etc. They also know what kinds of manipulations computers do well, and conversely, what kinds of things computers do in a clumsy fashion. All of this hard-earned knowledge and sophistication will tend to lead them astray when the time arrives to choose a problem. They will try to make use of this knowledge and hence choose a problem that will probably involve the simulation of humans solving problems with the aid of computers rather than the

simulation of humans solving problems with only paper and pencil. Consequently, the characteristics of present-day computers may confine and constrict the area of research much more than is desirable or requisite. What is liable to happen, and what did happen to us, is that the experimental problem chosen will develop into one of large size and scope. If this always happens, then those human manipulative abilities that are presently clumsy and time-consuming on computers will never get programmed, simulated, or investigated. Fortunately for us, the two experimental problems we chose were of such a nature that they could be easily miniaturized, and this was done as soon as the desirability became apparent.

The second pitfall which must be avoided is the assumption that one knows in detail how one thinks. This delusion is brought about by the following happenstance. People customarily think at various levels of abstraction, and only rarely descend to the abstraction level of computer language. In fact, it seems that a large share of thinking is carried on by the equivalent of "subroutines" which normally operate on the subconscious level. It requires a good deal of introspection over a long period of time in order to dredge up these subroutines and simulate them. We believe people assume that they know the logical steps they pursue when solving problems, primarily because of the fact that when two humans communicate, they do not need to descend to the lower levels of abstraction in order to explain to each other in a perfectly satisfactory way how they themselves solved a particular problem. The fact that they are likely to have very similar "subroutines" is obvious and also very pertinent.

† IBM Res. Center, Yorktown Heights, N. Y.

The third pitfall consists of the following fact: any problem chosen as an experimental vehicle is likely to produce interesting results if the program is successful. We consider these results to be byproducts of the general problem of studying the methodology of problem-solving, though they do serve as a test for success of the methods employed. As these byproducts accumulate, one is increasingly tempted to spend a disproportionate amount of time on obtaining useful and/or interesting byproducts. This diverts the researchers and they make little progress along the lines originally intended.

The class of problems we chose was the following: given a set of elements, and a criterion of compatibility between any two of its elements, find a subset of mutually compatible elements satisfying given constraints. This covers a large class of problems to which no satisfactory analytic solutions are known. An example would be: given the set of all airplanes near an airport, find the largest subset of those which are on compatible (*i.e.*, noncollision) courses.

The first experimental problem we chose consisted of this: a word list was given together with a table of synonyms for each word. The test of compatibility between any two words on the list was provided by a speech-recognition machine which was not as discriminating as the human ear. The problem was to find a word list the same length as the original list, but with synonyms substituted wherever necessary so that all the words on the resultant list could be unambiguously identified by the speech-recognition machine. The difficulty encountered in substituting the synonyms is that each substitution may cause added incompatibility relationships. The final list of mutually compatible words would be a practicable working vocabulary for voice control of, say, an air defense center.

We decided that the following three heuristic methods would probably be used by humans. These are:

Method 1—test the first word against all the following words. Wherever two words are incompatible, substitute synonyms for the second word until a compatible synonym is found, then proceed. Repeat with the second word of the list. After the last two words are tested, reiterate. Eventually a word list meeting the compatibility criteria will be found, or else the tables of synonyms will be exhausted.

Method 2—this method is basically the same as method 1, with this added sophistication: whenever a compatible synonym is substituted, it is immediately tested further for compatibility with all words in the list occurring before the particular two words concerned.

Method 3—determine which words are incompatible with the largest number of other words in the list and substitute synonyms for these highly incompatible words first. Then reiterate.

Two submethods also suggest themselves as processes to apply prior to trying the above methods. These are:

Submethod 1—sort the words on the first phoneme.

Submethod 2—try a batch procedure: divide the original list into two or three parts, apply one of the three main methods to each in turn, then put the batches back together before trying the final manipulation.

All of these methods were programmed and put together in a master program which determined the sequence of the application of each of the methods and submethods. Quite naturally, this sequence was: first, the “quick and dirty” method 1; then, the more sophisticated method 2; and last, the more complicated and most powerful method 3. At the start, each of the methods was allotted a certain amount of time, and if no compatible word list was produced within the allotted time, then the master program switched to the next method. This procedure was modified so that the time was extended if an analysis showed that the method had a good chance of successfully producing a compatible word list. To determine which submethod to employ first, a random choice was made, and lack of success automatically switched control to the other submethod.

To simulate these methods we employed a random number generator to generate 18-bit pseudo words, each consisting of three 6-bit pseudo phonemes. Inasmuch as the English language contains approximately 43 phonemes, 6-bit pseudo phonemes can reasonably be expected to represent adequately human phonemes. The human ear was considered capable of distinguishing between any two 6-bit pseudo phonemes that differed in one or more bit positions. The hypothetical speech-recognition machine, being not so discriminating, was considered capable of differentiating between any two 6-bit pseudo phonemes if and only if they differed in two or more bit positions. Mathematically stated, two pseudo phonemes P_i and P_j are considered compatible whenever

$$W(P_i + P_j) > 1$$

where the weight function, W , merely counts the number of ones in the argument, and the operation $+$ between the two pseudo phonemes is bit-by-bit addition modulo two, which is equivalent to exclusive OR.

Our experimental byproduct results are given in Table I, where the “word-list length” is defined as being the length of word lists that have a 50 per cent chance of being satisfactorily manipulated.

The second experimental problem consisted of finding the largest mutually compatible set of 12-bit numbers satisfying the following criterion of compatibility:

$$W(A_i + A_j) \geq 5.$$

This set will have the characteristics that double error detection and correction is possible when employing it as an information transmission code.¹

¹R. W. Hamming, *Bell Sys. Tech. J.*, vol. 29, pp. 147-160; April, 1950.

TABLE I
EFFECTIVENESS OF SEVERAL METHODS OF FINDING COMPATIBLE
WORD LISTS

Method	Word-List Length	Average Computing Time in Minutes
1 alone	325	0.65
1+sub 1	300	0.74
1+sub 2	325	0.78
2 alone	550	3.70
2+sub 1	525	2.55
2+sub 2	550	2.95
3 alone	740	3.82
3+sub 1	740	3.94
3+sub 2	740	4.50
Master Program	750 ±	Dependent on Word List Length

While in general the resulting code is nonconstructive, in this paper we confine ourselves to a subgoal, namely, the results obtained when constraining the set to be a constructive group code. With this constraint, the original compatibility criterion of $W(A_i + .A_j) \geq 2E + 1$, where E is the number of errors to be detected and corrected, can be reduced to the following set of constraints on binary numbers whose lengths are those of the check bits only; the so-called parity check matrix:²

$$\begin{aligned}
 W(C_i) &> 2E - 1 \\
 W(C_i + .C_j) &> 2E - 2 \\
 W(C_i + .C_j + .C_k) &> 2E - 3 \\
 &\dots \dots \dots \\
 W(C_i + .C_j + .C_k + \dots C_n) &> 0.
 \end{aligned}$$

The size of the set of C 's determines the maximum number of information bits that can be handled by check bits whose formulas for construction are given by the columns of the C 's. These constraints were programmed and the size of the matrix determined by exhaustion.

The experimental byproducts are given in Table II. Having satisfied your curiosity regarding the possibility of obtaining useful and/or interesting results from these methods, we now wish to discuss those aspects that are pertinent to the simulation on computing machines of human problem-solving strategies and techniques.

In the beginning we had what we thought were very good ideas of how to proceed. We had solved all of the problems on a medium level of abstraction. We soon learned, however, that an abstraction is nearly always a refuge from accurate knowledge. Explaining to each other how to reach a particular goal did not teach us how to program the simulation of the manipulations in-

TABLE II
DOUBLE ERROR-CORRECTION CODE LENGTHS

Number of Check Bits	Number of Information Bits
6	2
7	4
8	≥ 9
9	≥ 13
10	≥ 16

involved until we descended the abstraction ladder to the level of the symbolic coding process we were employing. For example in submethod 1, sorting the words according to the size of the leading 6-bit phoneme is useless. For detecting 1-bit differences it is a useful procedure, but for detecting 2-bit differences, one must order the words in an order that bears a useful relationship to the compatibility criterion. In our first problem we found this relationship in the weight of the phonemes, and accordingly arranged the words in order of increasing weight of the first phoneme. The result was a quicker finding and weeding out of incompatible words in isolated cases.

A more subtle difficulty arose in the second of our problems. Consider the following five beginning members of a larger set satisfying the constraints previously described:

- 1) 11 11 00 00
- 2) 00 11 11 00
- 3) 00 00 11 11
- 4) 11 00 00 11
- 5) 10 10 10 10.

If element number 5 was tried and discarded because it was found not to yield a sufficiently large set of C 's, then the number 01 01 01 01 need not be tried because it will yield similar unsatisfactory results. When doing this by hand on a piece of paper, one can immediately detect the equivalence between the two trial numbers for the fifth element. The criterion for this equivalence is that the two sets of five numbers containing different fifth elements are identical upon column permutation. It took us literally days to describe this simple relationship to the 704 computer whereas it takes only a minute to describe it to you. One of the reasons, of course, is that this concept involves two-dimensional visualization at which humans are particularly good and 704's are particularly inept. As a result, we wasted a lot of time trying to circumvent the transposition of the matrix elements in order to avoid using an excessive amount of machine time. A second-level difficulty arose when it was realized that all of the possible equivalences were not discovered by the original code, which did not contain column transposition. Consider, for example, the situation when the fifth member is 10 10 11 00. Then the number 10 10 00 11 is equivalent, but an involved

² D. Slepian, *Bell Sys. Tech. J.*, vol. 35, pp. 203-234; June, 1956.

column permutation had to be programmed in order to detect such an equivalence.

In conclusion, we would like to give our motivation in trying to simulate human behavior. At the present time, there are many problems whose nature is such that machines do not handle them successfully even though they are regularly solved by humans.³ Therefore, it seemed desirable for us to study a "working model" in action, to determine its component functions and to analyze each function to find out what part it played in the solution of these difficult problems. Our main goal is to be able to solve these kinds of problems, partly by new and different programs, and partly by

³ A. Newell and H. A. Simon, "Current Developments in Complex Information Processing," RAND Rept.; May 1, 1956.

H. L. Gelernter and N. Rochester, "Intelligent behavior in problem-solving machines," *IBM J. Res. Dev.*, vol. 2, pp. 336-345; October, 1958.

new and different machine-instruction sets. We do not insist that the machines do all the work, because we are convinced that a combination of a human and a machine working in tandem will always have superior problem-solving powers than either working alone. To complement each other, better communication is required between humans and machines, and this implies communication on higher abstract levels than the ones now employed. In order for this to be possible the machines must have either wired-in or programmed "subroutines" to interpret the directions and/or suggestions given them by their human colleagues. These "subroutines" should overlap those of humans in order to be useful. We feel that this goal will be furthered by continued research along the lines described in this paper.

The authors would like to express their sincere thanks to N. Rochester for suggesting the problems and outlining the heuristic programming methods utilized.

The Role of the University in Computers, Data Processing, and Related Fields*

LOUIS FEIN†

INTRODUCTION

SINCE the Fall of 1956, the author has been studying the genesis and operation of university programs in the fields of computers, data processing, operations research, and other relatively new and apparently closely related fields. The specific purposes were:

- 1) To study and evaluate the organization, curriculum, research program, computing equipment, financing, and facilities of universities in the United States having computer and/or data processing and/or related programs.
- 2) To identify those fields of study (some already accepted and identified as disciplines as well as those not yet so designated) that are unambiguously part of the computer and data processing fields and those closely related fields that might legitimately be part of a university program.
- 3) To appraise the role of the universities in these fields and to determine what universities might do to build distinguished programs in these fields.

During the course of the study it became clear that a university program in any field is an important function of the role of the university in society itself. The identification of this role thus became a fourth separately identifiable purpose of the study.

Source information was obtained by formal interviews and informal sessions with university administrators, directors of computing centers, faculty members, students, industrial representatives, and other interested persons. Places of interview were universities, scientific meetings, social gatherings, industrial plants, and research institutes. Important information was obtained from a few publications.

A questionnaire intended for mailing was considered and dropped because it became clear very early that only personal interviews could bring out the important facts and opinions. Hence, the conclusions and recommendations are not always derived from a mass of accumulated data. They do reflect the author's experience in the computer and data processing field, information about what universities are doing and especially what they are not doing, and the influence of those individuals interviewed who have experienced and reflected seriously on the same problems as those considered in this study.

* Part of this work was undertaken when the author was a consultant to Stanford University.

† Consultant, Palo Alto, Calif.

STUDY AND EVALUATION

Modern U. S. Culture

In response to the increasingly large and complex problems faced by society, our culture has changed in several important respects. For their solution, these new problems require new techniques and sometimes refinement of already developed techniques. The requirements for training people in new fields (and in older ones) and for research, design, and development are difficult for us to cope with. In fact, we have, as a nation, sought solutions to our problems in training, education, research, design, and development, from whatever institutions or individuals would make themselves available. On this account, traditional roles of institutions in our society are changing, and in many cases transformations are almost complete.

In the past, industry, government, and the university were each roughly assigned specific fields and subject matter in which they were active almost exclusively. Research and instruction in the "intellectual disciplines" were the province of the university, and design and development belonged mainly to industry and government. Today, one can sometimes scarcely identify teachers, researchers, designers, and development people as members of a university, industry, or government, for each is involved to a considerable extent with all three. Nor can they be identified by their (overlapping) fields of interests. For example, an information theorist may just as well be found at Bell Telephone Laboratories as at the Massachusetts Institute of Technology.

Specifically, the milieu of our society is characterized by a demand that more phases of our traditional culture be subjected to a critique guided by rationalistic and scientific rules of evidence. Thus,

Information-Data—gathering, storage, searching, classification, cataloguing, retrieval, encoding, decoding, interpretation, sampling, filtering, analyzing, checking;

Models—mathematical, Boolean algebraic, Post algebraic, logical, stochastic, computational, statistical;

simulators, computers, automation, language translation, switching theory, information theory, coding theory, cybernetics, decision theory, statistics, operations research, econometrics, psychometrics, management science, linear programming, game theory, automata theory, artificial intelligence, self-organizing machines, adaptive mechanisms, neural psychology, learning machines, and numerical analyses

are examples of topics—some old, some new—that are having a terrific impact on and general acceptance by the industrial, government, and university community.

This acceptance is due in no small measure to the spectacular successes enjoyed by some of these fields when applied to the solutions of problems in many of

the crash programs that characterize the present generation. The following excerpt from the April, 1956 issue of the *Journal of the Institute of Management Science* characterizes a phase of the milieu:

Primarily, what are needed are methods for grappling with multi-variable situations as they occur in the world rather than the single-variable methods of classic laboratory science. Hopefully, it should be possible to start with qualitative or rough quantitative approaches and gradually develop tools of greater precision. This suggests the use of mathematical models in both their qualitative and quantitative aspects. Techniques grounded in mathematics, notably including linear programming, search theory, and game theory (particularly non-zero sum multi-person games) have outstanding potential contributions. Marketing budgets involve commitments on too large a scale to permit executives to continue to accept historical patterns in a management world in which rational explanations of alternative strategies and planned optimizing are becoming standard operating procedures.

Industry, business, and government have varied requirements for trained people to do the work requiring talents in these fields. Need for investigating these fields from the scholar's viewpoint has also been recognized although support for this endeavor lags. The problems of coping with practical problems in and with these fields, training professionals, training scholars, and doing research, have been handled in a variety of ways by industry, government, and the university when indeed they have tried to handle the problems at all. On the whole, each of these three important segments of the community has been unprepared by virtue of its traditional organization, philosophy, responsibilities, and procedures to incorporate and cope with these new situations—situations whose fundamental implications usually require a modification of their organizations, philosophies, responsibilities, and procedures.

Thus, the government has set up RAND Corporation(s) for research and project work; it has set up and supported institutes at universities for research and project work; it has indirectly supported development, if not exploratory work, in these fields as byproducts of government contracts to industry.

Industry and business have set up separate departments charged with over-all responsibility of applying these techniques to company operation, management, design, manufacturing, sales, etc. Graduate-level schools for instruction of professionals in these fields are being run by industry itself.

The scholars and practitioners in these new fields are uncertain both as to the nature and structure of the fields and their relation to each other. As would be expected, new societies and magazines devoted to these fields have sprung up. The following quotes (from *Transactions of the Institute for Management Sciences*) bespeaks the growing pains of the community of practitioners and scholars in these youthful fields:

The writer has a strong feeling that those who worked so hard and successfully to organize *TIMS* were on the right track. However, it appears that they were led more by broad feelings than by logical reasoning. They knew (in the sense of having faith) that there was a need for something not yet in existence but they failed to develop a clear and distinct picture of what they intended to accomplish.

Further, the "Statement of Editorial Policy" from the September, 1957 issue of the *Journal of Information and Control*,

The theories of communication, computers, and automatic control were initially of interest primarily to mathematicians and engineers. The papers relating to these fields have therefore appeared in a wide variety of engineering and mathematical journals. Some of the results of this work have been applied by scholars in such areas as linguistics, psychology, statistics, physics, genetics, neurophysiology, and even philosophy. Such applications have led to results which are of interest in their own areas, but which in some instances have also suggested models and raised questions of fundamental interest to the theories being applied. Unfortunately, they appear in many entirely different sets of professional publications, which few readers of engineering and mathematical journals see.

It is the purpose of this new journal, *Information and Control*, to publish papers which make significant contributions to the theories of communication, computers, and automatic control and also papers which present experimental evidence or theoretical results bearing on the use of ideas from such theories in any field to which the ideas are relevant. Papers will be published, for example, on such topics as:

Theory of communication

Theory of automata

Theory of automatic control systems

Description and analysis of language and other natural information sources

Communications and control systems which may include links within or between organisms

Information aspects of physics and of the theory of observation and measurement

Organization, processing and retrieval of data.

Any statement of policy for a new enterprise is, at best, an educated guess, and this one is no exception. It is, of course, subject to gradual change without formal notice, as the *Journal* takes shape under the diverse pressures of authors, editorial board, editors, and readers.

THE ROLE OF THE UNIVERSITY

The universities, as institutions, are having a hard time learning to cope with their new role in society in general and in particular learning how to effectively incorporate these new fields into the academic structure.

Policy

Probably 150 universities and colleges are engaged in some kind of activity in the fields of our concern. The purpose of the schools that first got involved was "to get their feet wet" in fields that seemed to have appeal. Usually, one man who had a strong interest in this or that field would give a course in whatever department he happened to be in. Several institutions have embarked on a program of building large-scale digital computers. Some were successful. Some are now being built. Later, as the pressures for information in these fields grew, courses were added, some equipment was obtained, centers and institutes were established. Only a few universities have made a determined effort to select a field of interest, set up a policy and goal, and implement it. Most were feeling their way. Most are now feeling their way.

The most important impact on university programs in these areas has been the educational program of IBM. IBM has a manpower problem now; they know it will be severe in ten years. Their problem is two-fold. They need professionally trained people to help sell their product. They want their customers to have pro-

fessionally trained people to use their product properly. IBM has "presented" 650's to over 50 universities by now under the condition (among others) that a couple of courses in data processing and numerical analysis be given. A 709 has been "given" to U.C.L.A. for "emphasis on the study of business management problems." M.I.T. has been "given" a 704 for "research and education of students in computing techniques." The University of California has a retread 701. Sperry Rand, Burroughs, Bendix, Royal McBee, and a handful of other computer manufacturers have also "contributed" computers to universities (see the Appendix).

It is fair to say that, in many cases, to the extent that a university computer activity has a purpose at all, it has been made for them by IBM. It is true that for many universities, this is good. Otherwise, they may never have developed a program at all. Nevertheless, there are no distinguished academic centers of computers, data processing, and related fields, and I believe that this is so because not enough attention has yet been given to the development of an *integrated* program and policy in response to the needs and conditions of the whole community rather than as a supplement to computers obtained at a "bargain."

The scramble to get in on a "free" 650 computer from IBM is a disgrace in some cases. Course titles and contents have been created on the spur of the moment to fit the IBM requirements. Faculty have been assigned on the basis of their not having a full teaching load—more evidence of what is done without a clear, well-defined policy and program.

Organization

The university entity most popular as the center of activity for computers, data processing, operations research, industrial engineering, mathematics, business, etc., is the computing service center. Sometimes an interdepartmental committee is in charge of the service and of a few courses. Sometimes, all courses are given in one department without any necessary relation between course content and department. Some schools run one-week seminars. Others run symposia. In many cases, institutes, not altogether part of the academic structure, are involved.

Faculty

The range and variety of faculty participating in university computer and data processing programs compares favorably to the range and variety of the programs themselves. Everyone from novices to people with ten years of experience is participating. Part-time instructors from industry and government have been used. Interest without regard to experience and competence is sometimes the main requisite of the teacher in many instances. With the exception of some instruction in programming, courses designed for faculty training hardly exist.

There are several reasons for this unfortunate situation. First, the fields are so new and the need so great, that not enough experienced and competent people have developed to handle this need. Secondly, many people who could most adequately fill the need are being more highly paid for their services by industry or by the government. Thirdly, competent researchers ordinarily have better facilities and equipment for work, which, in many industries and government, is identical to what he would be doing at a university.

Curriculum

A few selected examples of topics in computers, data processing, and related fields covered in the present curriculum of universities are:

- Use of electronic data processing
- Logical design of computers
- Computer electronics
- Electronic digital computers—digital computer circuitry
- Theory of and operation of computing machines
- Statistics in business forecasting
- Dynamic programming
- Numerical mathematical analysis
- Numerical mathematical analysis laboratory
- Matrix analysis
- Matrix analysis laboratory
- Numerical solution of differential equations
- Numerical solution of differential equations laboratory
- Principles of digital computers
- Programming for digital computers
- Business and industrial analysis
- Statistical methods—regression
- Probability models
- Linear programming
- Game theory
- Monte Carlo techniques
- Data processing
- Systems and analysis
- Information theory
- Switching and computing circuits
- Theory of coding
- Information storage and retrieval
- Documentation and classification.

Such topics are covered not only in normal courses, but in special courses, seminars, symposia, and lectures.

It may be seen that even this incomplete list covers aspects in the design and design models for computers, programming, and applications of interest to people in the business schools, engineering, the physical and biological sciences, applied mathematics, statistics, industrial engineering, logic, etc. These courses reflect an appreciation of the ability of a computer to handle new computational models. They reflect an appreciation of the theoretical techniques of computer design.

What is impressive is the lack of courses reflecting appreciation of the computer as an *aid to routine mental*

effort, a theory of computers, a theory of programming, a theory of applications. This in turn is probably a reflection of the youth of the fields. Such theories have not yet been born. When they are created and developed, courses will undoubtedly follow.

Few, if any, of the university curricula are integrated. Even those that do have an aspect of integration—as the one at the Engineering School at the University of Pennsylvania, or the one at the Business School at U.C.L.A.—concentrate on the interests of people in particular disciplines interested in computers or data processing as tools for their own use exclusively. This is not to say that this is bad. But it is narrower than a university program might be.

Research—Subject Matter

The range and variety of potential research activity in the design, programming, and utilization of computers, and in closely related fields is enormous.

Theses and research papers have been written on topics in computer logic, automatic coding, switching theory, coding theory, neurophysiology, inventory control, production control, office automation, machine translation, organization, classification and retrieval of knowledge, and solutions of problems having computational models in almost every quantitative discipline.

Here, as in the curriculum, the fields of computer theory, application theory, model theory do not yet appear to have been successfully attacked.

Equipment Facilities

Most of the equipment (see the Appendix) now present in universities in the United States has been selected on the basis of a single criterion, the cut-rate price, in spite of the fact that the selection committee would sometimes have preferred other equipment on the basis of technical and operational criteria. The effect of this on the quantity and quality of the research output cannot be measured. However, it is clear that the basis for selection of equipment ought to be the maximum benefit to the university community. It is difficult to see how any of the universities, now selling computing time to sponsored contracts (in order to pay the bill, to be sure) and having little time (usually the second shift) for legitimate academic research and instructional pursuits, will gain an outstanding reputation for solving university research problems with computational models. They *will* become known for providing bargains for service to sponsored (usually government) research. The main objection to this sort of thing is that the benefit to the university research staff derived from the activity is fortuitous and incidental rather than the planned, aggressive activity it ought to be and must be for maximum benefit to the proper university function.

Some universities are still building their own computing equipment. There seems to be little reason to attempt this now, even at the present prices of commercially available computers.

SOME OBSERVATIONS FOR A UNIVERSITY PROGRAM

Probably the most important reasons for the universities' inability to cope adequately with their new role in instruction and research in these fields, are their failure to identify clearly the role *they* want to play, and their failure to try to fit the new fields to their role. This section includes some observations that should be aids in the conscious selection by universities of their particular roles and of how to implement these selected roles in these new fields.

The Function of the University

The legitimate function of a university in any society has been the subject of ancient and continuing controversy. The controversy has revolved around the questions loosely described as training vs education; practical vs theoretical subject matter; routine vs non-routine activities; scholarly vs professional endeavor; and the various shades of grey in between. A. N. Whitehead stated and elaborated on one position at the dedication of the Harvard Business School in 1927,

This article will only deal with the most general principles, though the special problems of the various departments in any university are, of course, innumerable. But generalities require illustration, and for this purpose I choose the business school of a university. This choice is dictated by the fact that business schools represent one of the *newer developments* . . . of university activity. . . .

There is a certain novelty in the provision of such a school of training, on this scale of magnitude, in one of the few leading universities of the world. It marks the culmination of a movement which for many years past has introduced *analogous departments* throughout American universities. This is a *new fact in the university world*; and it alone would justify some general reflections upon the *purpose of a university education*, and upon the proved importance of that purpose for the welfare of the social organism.

The novelty of business schools must not be exaggerated. *At no time have universities been restricted to pure abstract learning*. . . . There is, however, this novelty: the curriculum suitable for a business school, and the various modes of activity of such a school, *are still in the experimental stage*. . . .

These reflections upon the general functions of a university can be at once translated in terms of the particular functions of a business school. We need not flinch from the assertion that the main function of such a school is to produce men with a greater zest for business. . . . Business requires a sufficient conception of the role of applied science in modern society. *It requires that discipline of character which can say "yes" and "no" to other men, not by reason of blind obstinacy, but with firmness derived from a conscious evaluation of relevant alternatives*. . . .

Whitehead spoke well. If we substitute for the field of business, computers, data processing, and closely related fields (let us call them the "computer sciences"), then universities may well select as their role:

- 1) Training of professionals in these fields of "computer sciences" of interest and competence in the university.
- 2) Training scholars in these fields.
- 3) Doing exploratory research in these fields.
- 4) Developing the subject fields into new disciplines.

Disciplines and the Computer Sciences

Many fields are now providing models for many other disciplines. Thus, work under the province of operations

research, or game theory or decision theory or management science, or linear programming, or econometrics or statistics is being applied to business problems like market analysis, inventory control, long-range planning etc. In many instances, a computer is used to process data, solve equations, do analyses, even make decisions based on criteria it has been given.

Switching theory, coding theory, information theory, Boolean algebra provide models not only for design and programming and applications of computers, but also of analogous fields like neurophysiology.

The computer thus provides a significant link among various established disciplines as well as those fields of endeavor of intense present interest. Computers are related to other fields in one or more of three ways:

- 1) Workers in these fields use the computer as a mechanical or a mental aid.
- 2) These fields provide models useful in the design, programming, and applications of computers.
- 3) Analogies exist in the internal structure and organization of a computer with structures and organizations in other fields.

It seems plausible to designate the fields mentioned above and those enumerated in the Introduction, as the "computer sciences" since they are related to each other in one or more of the three ways enumerated above.

We must expect that some of these fields will coalesce and develop into disciplines on their own. These will then almost certainly be universally accepted as the legitimate province of the university scholar. Others may not turn out to be disciplines and will gradually be abandoned by universities. But not knowing now which field(s) will meet which fate, the university scholar must presently be interested in all of them.

The term "discipline" has been tossed around loosely, so perhaps we had better try to define it.

Although a set of agreed-to criteria do not exist by which one can determine whether or not a field rates as a discipline, there are several characteristics of accepted established "disciplines" that may be referred to when one has the problem, as we do, of deciding whether or not a given field is potentially a discipline.

Established disciplines, like mathematics, have the following characteristics:

- 1) The terminology has been established, a glossary of terms exists.
- 2) Workers in the field do nonroutine intellectual work.
- 3) The field has sometimes been axiomatized.
- 4) The field is open, *i.e.*, problems are self-regenerating.
- 5) There is an established body of literature, textbooks, sometimes treatises—even handbooks—and professional journals.
- 6) University courses, sometimes departments, and indeed schools are devoted to the field.

Most aspects of computers, data processing, and the related fields discussed in this study now meet these specifications or may be meeting them in the next ten years. It is clearly the job of university people to help create the axiomatization, theory, terminology, curriculum, etc. Computer science is *not* an isolated field. It is interdisciplinary. It is analogous to a library, or mathematics, where library science or mathematics are disciplines in themselves as well as providing service tools to other disciplines. Thus, future university courses of instruction and research should be provided in the unit devoted to the disciplines of interest and in units using these disciplines as tools.

The Computer

Too much emphasis has been placed on the computer equipment in university programs that include fields in the computer sciences. Curriculum and research programs have been designed as supplements to the computing equipment. The reverse should be the case. Computers should be supplements to a well-organized and integrated university program in the computer sciences! An important supplement, to be sure, but a supplement. Indeed, an excellent integrated program in some selected fields of the computer sciences should be possible without any computing equipment at all!

Universities and Their Competition

Universities must recognize and design around the fact that both industry and government have already undertaken and will continue to participate in the university's traditional role of instruction and research in the computer sciences. This implies competition with government and industry for personnel, equipment, and financing. The university structures and policies built over the years to cope with traditional situations may not be adequate to cope with the present situation.

A RECOMMENDED UNIVERSITY PROGRAM

The following is a detailed integrated program designed around the observations made in the previous section.

Organization

A graduate school, called the Graduate School of Computer Sciences, should be formally created. A policy committee, including a dean and his academic and service departments heads, should be appointed. Its primary function will be to set up and implement policy and budget on the function of the school, the curriculum for an integrated program of instruction and research, its relation to other schools and departments of the university, and other pertinent points, including financing. A "five-year plan" should be formulated.

The dean of the school should report to the president. His responsible administrators will be scholars first and foremost. Their professional fields of interest will be in the fields of interest of the school, although they will

have an appreciation and respect for the problems, philosophy, and ideals of scholars in fields other than those of their own interest. They will be enthusiastic, competent, and experienced.

The school administrators will not be promoters. The problems of budget, financing, equipment support, and public relations—all of which normally require the talents and time of a promoter type—will, however, be assigned to professionals in these fields. They will report to the administration and will be observers in policy meetings. Theirs will be a service activity and not an academic one.

None of the academic administrators will be required to show a profit on his activity. All activities with an objective of making money will be organized under the service activities associated with, but not an essential part of, the academic unit.

The declared policy of the school concerning its fields of interest should be that it is interested in instruction and extension of knowledge in all fields that provide either quantitative models, or techniques for solutions of problems in quantitative models for phenomena of interest to the scholar. The theoretical and experimental fields enumerated earlier in this report are examples of these fields.

Since it is recognized that some of these fields are disciplines in themselves as well as tools for use by other fields, the Schools of Business, Industrial Engineering, Electrical Engineering, Mathematics, Physics, Psychology, etc., should be encouraged, when they find it desirable, to give courses and even to do research in fields normally covered by the new graduate school. But it is the responsibility of the new school to program an integrated activity for the whole university.

The declared and announced policy of the school should be that its function is fourfold: 1) To train professional scientists; 2) to train scholars; 3) to do exploratory research; and 4) to develop the new disciplines.

These are only a few policies recommended for adoption. It cannot be emphasized too strongly, however, that the significant recommendations here are not so much in the detailed recommendations enumerated above, but in the *recommendations for deliberate policy making and policy review of these matters.*

Departmental Structure

The departmental structure of a new school must of necessity be arbitrary. One structure for consideration would consist of four departments as follows:

Department 1—Computer Department: This department would include groups concerned with computer equipment and service, faculty instruction in programming and computational models, model making, automatic programming, logic, computer organization, computer mathematics, computational models, computer theory, component and circuit research (hardware) if not covered in other departments, systems research (hardware) if not covered in other departments, etc.

Instruction and research in these fields will fall in this department. The computer equipment used in a service bureau or for instruction or research will be under the cognizance of the Computer Department head.

Computation center: The purpose of the "computation and data-processing center" is to provide the tool for solutions of problems that have been cast into computational models by members of the university community.

The head of this center, who should be a scholar primarily, will be responsible for organizing the center operation for maximum efficiency. This will involve educating the general faculty, providing programming and operating assistance, promoting research and development of automatic programming and handling techniques, as well as assisting in the creation of computational models. By whatever means the equipment, facilities, and personnel are financed, the head of the computing center should *not* be obligated to show an operating profit. If profit is a prerequisite, then he should by all means go into business for himself. Functionally, the computer center should be organized into groups individually responsible for: 1) faculty education; 2) programming, coding, and operation assistance; 3) model construction assistance; 4) research in automatic coding, handling, etc.; and 5) a planning, scheduling, and monitoring activity.

The future demand for the products and services made available by a computing center depends, of course, upon the willingness and ability of the research faculty at the university to use them properly, as well as upon the utility and applicability of computational models in their work. It is true now that there is hardly a field in the physical, mathematical, biological, or social sciences that has not used computational models for research purposes. Future researchers in these fields will be using computational models more and more. A computing center will fill an important need on the university campus. It may be said, "To the extent that a researcher does not use such facilities when they are available, applicable, and potentially useful, to this extent a researcher is not doing a competent job."

Department 2—Operations Research: This department will cover instructional and research activities in operations research, linear programming, dynamic programming, game theory, queueing theory, and decision theory.

Department 3—Information and Communication: This department will cover instruction and research activities in information theory, switching theory, coding theory, automata theory, artificial intelligence, learning, language translation, and theory of simulation.

Department 4—Systems: This department will cover instruction and research activities in management science, econometrics, systems theory, information classification, indexing and retrieval, model theory, self-organizing systems, and adaptive mechanisms.

Curriculum

The curriculum will contain courses, seminars, lectures, etc., that constitute an *integrated* program for students pursuing an advanced degree. What constitutes an integrated program will be determined as part of school policy. Special courses, seminars, or symposia may be given as the need arises. The curriculum will be expanded or modified in accordance with periodic scheduled evaluations by curriculum personnel. Extensive special studies of what other universities and industry and government are doing will make appropriate information available to the curriculum evaluators.

The curriculum will also reflect the needs of other departments and schools within the university.

Faculty

One of the reasons for recommending an independent entity such as a graduate school—rather than a department in an already existing school or interdepartmental committee—is to allow a freedom of choice in policy that can respond to the practicalities of the situations faced by the university today, without the constraints imposed by an existing structure designed to cope with situations that no longer exist. Thus, the department heads, if not others, may command perhaps \$20,000 a year for their services in today's competitive market. If this is indeed so, then the school should plan to put itself in a position to obtain these people and to pay such prices.

Each course, lecture, and symposium will be conducted by faculty members who are competent to teach the course, give the lecture, or conduct the symposium. The faculty will be interested, enthusiastic, and competent, and preferably experienced. They will be paid a salary commensurate with their experience and ability on a scale more in keeping with present industrial scales, not present university scales.

No faculty member will be assigned to instruct because his teaching schedule isn't full, or because he is a competent researcher who "ought" to teach, or because he is enthusiastic but inexperienced in the field.

Research and Related Activities

Research by graduate students or the faculty cannot be a planned affair, of course. One would expect to see research activity on selected topics of interest and within the capabilities of the school.

However, there are related activities that may be arbitrarily categorized as research having to do with the development of new disciplines and the incorporation of information developed by research into the curriculum. A well-established discipline usually has a well-established terminology and a glossary, axiomatization, and text books, treatises, handbooks, journals, and organizations. Thus one should expect to see the research and instruction faculty engaged in establishing terminology, axiomatizing a field, writing or editing

books, journals, and other such material, and even helping to organize professional societies.

Manual Aids

Manual aids, such as desk computers, computresses, bookkeeping machines, punched card machines, and the like, have been extremely valuable to university scholars and students. The ability of modern high-speed equipment means that the quality and quantity of university output should be improved to the extent that this kind of manual aid is made available and used efficiently.

It may be here noted that the quantity of the research output should increase not only because of the high speed and versatility of the electronic aid, but also because more of the problems, tasks, and the models created in the various disciplines become tractable with the use of the new aid. The quality should improve for a similar reason. There exist models of various phenomena in almost all disciplines whose problems have not been soluble with present techniques.

In many cases simplifying assumptions are made and the model is revamped in order to make the problems soluble. In these cases, the *wrong* problems have been solved quite accurately. Present high-speed equipment makes possible the accurate solution of more of the *right* problems.

The availability of high-speed computing systems and efficient operating crews has a much greater impact on the university output than, for instance, the availability of batteries of desk calculators and operators. The curriculum, research program, and even faculty education are affected. Scholars and students will have to learn enough about the available devices to use them well with their present models. This knowledge should serve to motivate them to the creation of new machine-soluble models. In fact, a general theory of the construction of techniques of machine-soluble model construction for any discipline seems clearly to be a job for the university scholar.

This is an unexpected but inescapable development growing out of the computer as a manual tool!

In addition, those techniques peculiar to the design and programming of high-speed equipment should be developed, if they can serve to make it a more efficient aid for manual routine work and especially if it results in an aid that could help handle more models. As a basis for this endeavor, there will undoubtedly be required at least a *competent theory of computing machines*, a *competent theory of models*, and a *competent theory relating the two*.

Mental Aids

Equipment for aid in routine mental tasks has fewer precedents than equipment as an aid for routine manual tasks. Computers are now being used to check results of some mental efforts. For example, they are being used to check the logical design of other computers. The ability to use equipment as an aid to do routine mental work, and thus make it literally an adjunct of one's self in creating new disciplines or developing existing ones, may turn out to be a discipline in itself. To be sure, this is rather vague, but this in no way weakens the conviction that it is a legitimate interest of the university. The university scholar must also be interested in the peculiar programming and design techniques of an equipment that will make the equipment most useful as an aid to routine mental work.

Financing and Student Body

Specific recommendations for sources of funds are not made here. The strong conviction exists that the present growing demand for this kind of activity by industry and government is so great that students and money should be knocking on the door where such a program is available.

Students will come from government, industry, and undergraduate schools. They will be interested in later applying their new knowledge to industrial or business pursuits; others will wish to be educated to become researchers and scholars in selected fields. How many of these aspirants attend the university will be a measure of the appeal, if not the distinctiveness, achieved by the university in these fields.

APPENDIX

SUMMARY OF COMMERCIALY AVAILABLE COMPUTERS INSTALLED AT UNIVERSITIES USED FOR UNIVER- SITY RESEARCH AND INSTRUCTION

	Number Installed	Estimated Number to Be Installed in 1959
IBM 650	65	No estimate
709	1	No estimate
704	2	No estimate
701	1	No estimate
Sperry Rand 1101	1	5 large-scale machines
1103	2	(models undetermined)
1105	2	
Univac 1	4	
Burroughs 220	1	2
205	7	6
Bendix G-15	12	12-24
Royal McBee LGP-30	13	20
NCR 102D	3	—

The RCA 501 Assembly System

H. BROMBERG†, T. M. HUREWITZ†, AND K. KOZARSKY†

CURRENT techniques in automatic coding attempt to shift the user's tasks away from the computer and closer to his application. Sacrificing coding details in this way, it is believed that monumental savings will result both in computer acceptability and utilization since everyone is now able to describe his own programs. Universal acceptance of problem oriented languages has, for several reasons, not yet followed. One influence is that the generated object programs reflect the adroitness of the executive routine and the remoteness of the input language.

In a recent count, there appears to be well over one hundred automatic coding systems produced for twenty or more different computers. This reflects the recognition of the disparity that exists between the methods of problem preparation and actual problem solution.

By most methods of classification, these hundred-odd automatic codes range more or less continuously between extremes. They vary considerably in complexity, extent of problem area of useful application, and in range of intended user.

One categorization used to differentiate these automatic codes is by the sophistication of the input language—particularly whether this language is “problem oriented” or “machine oriented.” However, it must be admitted that if a ranking of these automatic codes is made according to efficiency of the object program, the list would tend to be in nearly inverse order to that obtained by ordering on the level of the input language. It should also be noted that evaluations of the academic aspects of these automatic codes are often greatly at variance with the judgments of the occasionally unfortunate users of these routines.

This is not to say that object program efficiency is the only value criterion of an automatic system. Frequently for short programs or where the capacity of the data-processing equipment greatly exceeds the required performance, it is almost irrelevant. But, in instances where object program efficiency is significant, alternative coding procedures are desirable.

It is conceded that the Problem Oriented Language deservedly has greater prestige than the Machine Oriented Language and greater theoretical interest (at least from a philosophic or linguistic point of view). Nevertheless, the current mechanization of these languages and the distribution of computer expenses dictate demands for both types. It is recognized that direct, facile communication between the layman and his computer as well as the advantage of interhuman communication of the problem definition are obtainable from a Problem

Oriented Language. However, there are also needs for programs handling tasks near the limits of the equipments' capabilities as well as for infrequently changing, very highly repetitive, data-processing routines.

One of the often expressed goals in automatic coding is the development of complete problem oriented languages entirely independent of any computer. To produce any “most efficient” coding in this circumstance means that, among other things, psychological inferences as to the intentions of the writer are to be made by the automatic code. Furthermore, the apparent trend in machine design toward many simultaneous asynchronous operations, multiprogramming and the like, increase the problems associated with producing efficient machine programs from a problem oriented language. It is hardly unreasonable for a user of the new potentially powerful systems to request a coding scheme capable of using these complex, expensive features.

It appears likely then, in the near future at least, that some problem oriented languages will be augmented by some prosaic statements, directly or indirectly computer-related, which will permit attainment of a more “most efficient” machine code. Similarly, machine oriented languages may also yield to this trend and incorporate some features, within their inherent limitations, which tend to be associated with problem language codes.

All this may justifiably be construed as motivation for the automatic routines offered with the RCA 501. The first of these, a machine oriented automatic code, the RCA 501 Automatic Assembly System, is described in this paper.

The Assembly System provides for: relative addressing of instructions and data; symbolic references for constants and data; macro-instructions and subroutines; variable addresses; and descriptor verbs.

SOME MACHINE CHARACTERISTICS

It is appropriate, as background for what follows, to describe briefly some of those features of the 501 Computer (Fig. 1) which have influenced the design of the Automatic Assembly System. (Incidentally, this computer has been in operation in Camden since April, 1958.)

The 501 Computer has a magnetic core storage with a capacity of 16,000 characters, which is increasable in steps of the same to a maximum of 262,000 characters. Each character, consisting of six information bits and one parity bit, is addressable, although four characters are retrieved in a single memory access. Binary addressing of the memory is provided, requiring 18 bits or three characters per address.

† RCA, Camden, N. J.

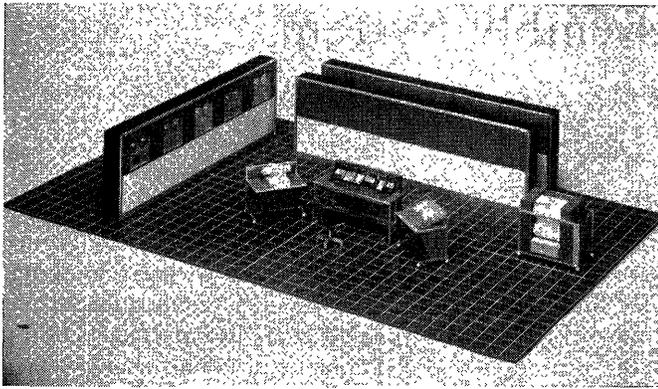


Fig. 1—The 501 computer.

The instruction complement consists of 49 two-address instructions (see Fig. 2). Each instruction consists of:

- One character for the operation symbol
- Three characters for the *A* address
- One character for the selection of address modifiers
- Three characters for the *B* address,

for a total of eight characters or sixteen octal digits.

There are several control registers, each of which stores 3 characters, or one address (see Fig. 3). The *A* and *B* registers are used to store the *A* and *B* addresses of operands during their execution. The *P* or program register stores the memory address of the next instruction in sequence. The *T* register stores a memory address which is made use of by certain instructions which require three addresses during their execution. All of these registers are addressable and therefore directly accessible to the program.

Seven address modifiers are available. Four of these are standard memory locations and three are the *A*, *T*, and *P* registers. Use of the *P* register as an address modifier permits the writing of self-relative machine coding which may be operated, without modification, in any part of the memory.

The RCA 501 exploits certain control symbols (Fig. 4) in the data. The Start and End Message control symbols define a message on tape, and in the memory also act as control symbols for certain instructions. The Item Separator control symbol is not used as such on tape, but is used in the memory to control certain operations. These control symbols permit variable item lengths and variable message lengths both on tape and in the memory. The entire message may be variable, dependent upon the number and size of the individual items. The instruction complement includes both symbol controlled and address controlled operations.

The 501 includes provision for simultaneous read-write, read-compute and write-compute. This is accomplished by designating magnetic tape instructions as "potentially simultaneous" and establishing a program controlled gate between the normal and simultaneous modes of operation. Thus, the programmer can

Operation Code	<i>A</i> Address	Address Modifier	<i>B</i> Address
<i>O</i>	<i>AAA</i>	<i>N</i>	<i>BBB</i>

Fig. 2—A description of one instruction.

<i>P</i>	<i>A & B</i>	<i>T</i>
Stores Address of Next Instruction	Store Address of Operands During Execution	Third Memory Address

Fig. 3—Control registers.

Start Message and End Message (())	Item Separator (·)
Define Message on Tape and Control Operations in Memory	Controls Operations in Memory
Typical Message (·12564·John-Doe·8934·7)	

Fig. 4—Control symbols.

permit completely automatic switching of tape instructions to the simultaneous mode or he may optionally bracket off portions of the program where such switching is inconvenient.

As for the 501 Assembly System, there are two programmer-prepared inputs. To guide the Assembly System in generating a running program from the pseudo-code, the user provides the system with a description of the data files which the program is designed to process. A portion of a data sheet is illustrated in Fig. 5.

Certain auxiliary computations are performed on these data sheets and the results printed out for the programmer's information—such as average message lengths, approximate tape passage time, and weighted average.

DATA ADDRESSING

Completely variable length data, on the one hand, yields economies in tape storage and effective file passage time; on the other hand, it presents certain problems with respect to symbolically addressing data items in the memory. These problems are handled in several different ways:

- 1) Those items whose lengths are fixed relative to the beginning of the message may, of course, be directly addressed by the data name designated in the data sheets.
- 2) The variable length items of a message may be transferred to a working storage area of memory where space is allocated for the maximum possible size of each variable length item. A single pseudo-instruction performs this function. From this point on, the variable length item may be directly

Item No.	Sub Item	Abbreviation	Description	FAA	JY	Sign	No. Char.		% Use	Wtd. Avg.
							Max.	Avg.		
1		Date	File Label	X	R		6	6	100	
	A	Month					2	2		
	B	Day					2	2		
	C	Year					2	2		

Fig. 5—The automatic code data sheet.

Instr. No.	Comments	OP	A Address	B Address	T Address	CSG	IF	GO TO	IF	GO TO	IF	GO TO

Fig. 6—The automatic code program sheet.

Instr. No.	Comments	OP	A Address	B Address	T Address	CSG	IF	GO TO	IF	GO TO	IF	GO TO
PRO 1		LRP	POLCY									
	(Macro-Instruction)	TEST	POLCY				EF	PRO40+1	ED	PDQ8+10		
		SC	DATE	DATE	"122558"		+	PRO23+5	-	PHI 15	O	N
PRO 20	"Extra Beneficiary"	DEFK	KBEN									
		DA	RATE	TAX	WRATE							
		ADV		+V4								

Fig. 7—Assembly pseudocode entries.

addressed by the data name preceded with a *W* (representing working storage).

- It is possible to locate the address of any item in a message by using an instruction which scans a message searching for and counting the control symbols defining items. This instruction leaves the address of the item in an address modifier.

The second input, whose format is shown in Fig. 6, is the pseudocode written on the program sheets.

This is seen to be an expansion of the machine code format which normally includes the operation field and the two addresses *A* and *B*. This is augmented on this sheet by the *T* or third address which for several machine instructions requires presetting. Furthermore, there are provisions for 3 "IF-GO TO" statements providing for conditional or unconditional transfers of control.

The inclusion of these IF-GO TO statements as an

optional part of every pseudoinstruction line has two primary motivations. First, it accommodates as a single pseudocode statement the function "Compare and Jump" which has a relatively high frequency in data-processing problems. Second, about $\frac{1}{3}$ of the 501 instructions automatically set a register to 1 of 3 states depending on conditions encountered during the operation of the instructions. Branching instructions may then be used to select different paths depending on the setting of the 3 state register and are easily designated in the IF-GO TO columns.

A single character entry in the CSG column generates an instruction to open or close the simultaneous gate, controlling the phasing of simultaneous tape operations.

VARIABLE INSTRUCTION GENERATION

An interesting feature of the Assembler is the handling of the normal complement of 501 machine instruc-

tions. Each of these instructions, when used in the expanded pseudocode format, assumes the identity of a macro-instruction. Up to five, two address machine instructions may be generated by employing a single machine operation code with appropriate entries along the pseudocode line. As an example, the 501 instruction, Decimal Subtract, may accomplish not only a three address subtraction, but also a simultaneous gate operation, and transfers of control dependent upon the sign of the difference.

SYMBOLIC AND RELATIVE ADDRESSING

Included in the automatic system are such features as mnemonic operation codes, symbolic and relative addressing including the use of both alphanumeric and octal literals for constants, and acceptability of machine code should it be desired. Fig. 7 shows examples of pseudocode entries.

The Instruction number is composed of characters designating the page and line number of the instruction.

When addressing an instruction, reference is made to that instruction whose elements appear in the Operation, *A*, and *B* fields of the program sheet. Since, however, one single pseudoinstruction may account for as many as five machine instructions, it is desirable to address those. Therefore, a stipulated suffix to an instruction number will allow a reference to any generated instruction and to any field or desired character within one of these instructions.

Relative addressing of these symbolics enables the programmer to refer to the *N*th pseudoinstruction following or preceding a given symbolic. The program will be ordered by page number in alphabetic sequence and within pages by line number before any processing is undertaken. Accordingly, to accomplish an insertion, one need only assign appropriately sequential labels to the desired instructions and the program will place them in the proper positions.

It is not necessary, however, to label every instruction. Relative addressing allows reference to be made to unlabeled instructions in the program. One might usually expect to be labeled the first of a sequence of instructions performing a logical function and those to which frequent reference is made by other instructions, but this is left solely to the discretion of the programmer.

DESCRIPTOR VERBS

Descriptor verbs constitute an important part of the automatic code. These verbs contribute, in general, only to the description of the program and do not become a directly converted active part of the machine code.

These special verbs perform a variety of functions such as the definition of program segments, overlaying memory regions, reserving areas of memory, extracting the machine address corresponding to any symbolic name, defining constants and variables and providing for insertions, deletions, and corrections in pseudocode.

These verbs are executed during assembly and are deleted from the final program.

VARIABLE ADDRESSING

Another programming aid incorporated within the system is the variable address feature. A variable address allows the specification of addresses or constants to be symbolically named and to be defined later in the program. A variable may be substituted for any other machine or symbolic address in any instruction. This feature, for example, permits tagging, as a variable, the address of an instruction not yet written. It is only necessary then, at a subsequently convenient moment, to employ the Descriptor Verb, "Define V," to supply the actual address of this variable for every place it was used.

It is also possible to use variable addresses in addition to any machine or symbolic address. This is accomplished by placing the variable address in the same column as the one to which the variable is to be applied and in the directly succeeding line. A plus or minus prefix will then specify addition or subtraction of the variable address. A variable to be added or subtracted will not be applied until the variable is converted to an actual machine address. The use of variable addresses, then, allows for symbolically designated modification of the program at the actual or machine code level.

LITERALS

Literals, or constants whose address and name are identical, are used in the assembler. Two types of literals are provided, alphanumeric literals for operations with data, and octal literals for operation with instructions which, it has been noted, are binary coded.

A literal is normally carried along with the segment in which it appears. However, a terminal character of the literal may be used to specify that the literal be stored in a common constant pool available to all segments of a program. A terminal character may also be used to designate and to differentiate among duplicated copies of the same literal in the program. Here, too, these duplicate literals may be associated with the segment or with the common pool of constants.

Alternatively, of course, constants may be defined by a "Define Constant" Descriptor Verb and assigned an arbitrary symbolic address. Terminal characters on these constants perform the same functions with these constants as those just described.

MACRO-INSTRUCTIONS

The macro-instructions included with the Assembler create 2 address symbolic coding which is spliced directly into the main body of coding in place of the macro-instruction pseudocode call-line. A single macro-instruction will generate all of the instructions required to perform some task which would normally require the writing of a sequence of machine instructions.

Parameters, which the macro-instruction uses, are specified at the pseudocode call-line by the program-

mer. No restrictions exist as to number or size of these parameters. If a macro-instruction is to be generative, it contains one other part aside from the main body of stored coding. This part decides, from an interrogation of call-line parameters, which particular set of macro-instruction coding is to be included in the main routine.

SUBROUTINES

The assembly system provides for an expandable library of subroutines to be available to the programmer. These subroutines generate assembly language pseudocode and as such may use all the assembly features such as macro-instructions, descriptor verbs, and so forth. Subroutines may be open or closed and generative or fixed.

Parameters for subroutines are specified at the pseudocode calling line. For open subroutines, parameters are incorporated during the operation of the assembler. These parameters may merely be substituted in the subroutine as in the case of a fixed routine or may be subject to considerable testing and manipulation as occurs with a generative subroutine.

Closed subroutines may either incorporate parameters during assembly or use parameters generated by the running machine program. In this case the parameters are located relative to the subroutine call-line.

The design of the system is open to the extent that any useful number of macro-instructions and subroutines may be added.

PROVISIONS FOR PROGRAM MODIFICATION

The Assembly System offers two main listings for program up-dating. First, listings are given of the object machine code and the Assembly language pseudocode. Second, is a list of all symbolic addresses and those in-

structions referring to them. In addition, the Assembly System generates an information block preceding each object program. This block, which contains all program stops, breakpoint switches, and tape addresses is available for input to a service routine which will modify any corresponding entries within the object program.

There are two types of error indicators used by the Assembler. One causes the Assembly System to print the source of trouble and stop immediately. The other and major class consists of on-line printed statements indicating the type and location of errors. In this case the Assembly System continues its functions ignoring the "guilty" statements until all such indicators have been found. This permits the user to specify corrective measures for all errors at one time.

In summary then, the 501 Assembly System lies in an intermediate category. On the one hand, it is definitely machine oriented, amplifying the 501 instruction complement and requiring a knowledge of the 501. However, it also provides for a flexibility of order statements, not confined to the 2 address machine order code. A variable number of machine instructions are generated dependent upon the number and types of entries made on each pseudocode line. Both macro-instructions and subroutines may be of the generative type and since the library is open-ended, may be augmented whenever necessary.

In short, the RCA 501 Assembly System is a programmer's aide, enabling him to make maximum use of machine capabilities with a minimum of clerical effort.

ACKNOWLEDGMENT

The authors acknowledge the extensive contributions of M. J. Sendrow, who participated in the planning and creation of the RCA 501 Assembly System.

A Program to Draw Multilevel Flow Charts

LOIS M. HAIBT†

INTRODUCTION

THE preparation of a program for a digital computer is not complete when a list of instructions has been written. It still must be determined that the instructions do the required job, and if necessary the instructions must be changed until they do. Also a description of the program should be written for others who may want to understand the program. A useful tool for the last purpose is a graphical outline of the program—a flow chart.

Flow charts serve two important purposes: making a program clear to someone who wishes to know about it, and aiding the programmer himself to check that the program as written does the required job. A flow chart drawn by the programmer would serve for the first purpose, but drawing one is often a tedious job which may or may not be done well. For the second purpose, it is important to have the flow charts show accurately what the program does rather than what the programmer might expect it to do. Consequently, it was decided to write a program, the Flowcharter, for the IBM 704 to produce flow charts automatically from a list of in-

† IBM Res. Center, Yorktown Heights, N. Y.

structions. Another reason for the project was to get further insight into the characteristics of computer programs.

Since programs in many programming languages and even for different machines differ mainly in superficial aspects such as names and numbers for various operations, names and types of registers, it was decided to have the Flowcharter do the main part of its work on a common, machine-independent language and to have a set of preprocessors, each of which would translate one programming language into this common internal language.

We also felt it was desirable not to attempt to show the whole program as one chart, which for a moderate size program would either present a confusion of detail or be too general to serve the purpose. In order to provide both a good general picture of the program or of any part of it, and a more detailed description of a smaller piece of program, the Flowcharter produces a series of flow charts on a number of levels of detail; each part of a chart is shown in more detail on a succeeding chart. How to determine the makeup of the charts was one of the most difficult problems encountered in planning the Flowcharter.

Another feature of a flow chart is a description of the procedure represented by each box. The Flowcharter provides a summary of the machine input-output done in the box and a summary of the computation done in the box, listing the quantities computed and those used in the computation of each of them.

DESCRIPTION OF THE FLOWCHARTER

The Flowcharter is composed of four main parts: the preprocessors, the flow analysis, the computation summary, and the output program.

The preprocessors each do a simple translation from the external instructions to the internal language. For most machines, an instruction may represent several different processes done in the machine, such as fetching from memory, storing the memory, and instruction sequencing. These operations are each described separately in the internal language. One external instruction is translated by the preprocessor into a suitable list of these operations.

Many of the problems which arose in designing a Flowcharter were in the section which determines what is to be shown on each chart. It is very easy for the programmer to mark off his program into logical parts, but to determine these from the program itself is quite difficult in most programming languages. We have worked out a set of techniques which we feel will do quite well for most programs and will be acceptable in other cases. We have also provided facilities for the programmer to specify how he would like the breakdown done on various levels if he does not like the choices made by the Flowcharter. The techniques used depend mainly on analysis of the flow properties of the program but provision is also made in the Flowcharter for using the data to help in the analysis. The Flowcharter is written in

such a way that various techniques and combinations of techniques can be tested to see what results they give.

This flow analysis is done by iteratively forming regions from groups of subregions. The smallest subregions are individual instructions. In general, each region will be represented by one flow chart and each box drawn on the chart will represent a subregion of that region. However, when it is reasonable, two or more regions, each consisting of only two or three subregions, will be shown on one flow chart. This is done to keep the output moderately compact. Also, those regions which are formed directly from instructions are not shown as flow charts but are given as a list of the instructions in the region, with a reference to the page on which this region is shown in context. (The Appendix shows an example of this.)

The techniques used for region formation are of two kinds, combination and division. A combination technique is one which starts with individual instructions and, by repeated applications, combines them into larger and larger regions. A division technique is one which starts with the whole program and divides it into smaller parts. Each of these parts is in turn divided until each part consists of not more than six or seven of the regions formed by the techniques of the first type. Each technique is represented by a subroutine.

Each combination subroutine searches for a particular configuration of flow in the program. Three such subroutines are: STRING, DIAMND, and TEST, which look for "strings," "diamonds," and "test sets."

A "string" (see Fig. 1) is an ordered set of regions satisfying the condition that every region, except the first, has an entry only from the preceding region and each, except the last, has an exit only to the next one.

A "diamond" (see Fig. 2) is a set of regions containing a first region *F*, a last region *L*, and some intermediate blocks. Each intermediate block must not have any predecessor other than *F* nor any successor other than *L*. All successors of *F* and predecessors of *L* must be in the "diamond."

A "test set" is a set of regions which together make up a compound test. A set of regions forms a "test set" if each region ends with a test of the same special register. Also, every region except the first may have only one predecessor which must also be in the set. Finally, only the special register tested may be changed by the instructions in any of the regions except, possibly, the first one. For example, consider the 704 SAP instructions:

CLA	ALPHA
TZE	ISZERO
SUB	ONE
TZE	ISONE
SUB	ONE
TZE	ISTWO
SUB	ONE
TZE	ISTHRE

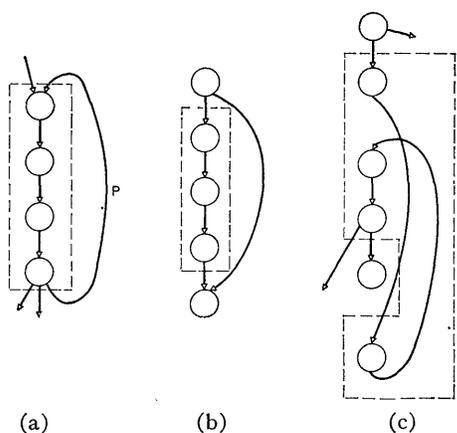


Fig. 1—In each case the dotted lines enclose a “string.” (Circles represent regions formed earlier and solid lines represent paths of flow in the direction of the arrow.)

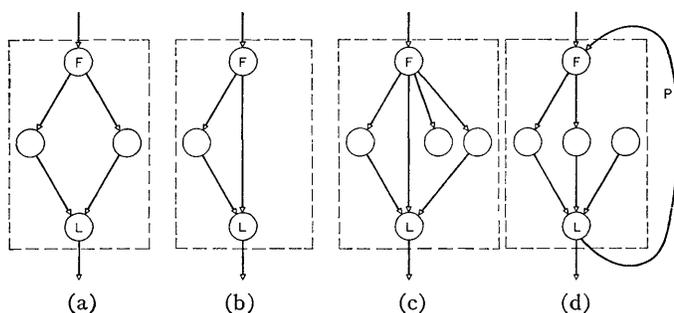


Fig. 2—In each case the dotted lines enclose a “diamond.” (Circles represent regions formed earlier and solid lines represent paths of flow in the direction of the arrow.)

The pair CLA, TZE, and each pair SUB, TZE make up a region found by STRING; then these four regions will be combined by TEST.

It should be pointed out that the first two configurations, “strings” and “diamonds,” are sufficient to describe most programs. Iterative loops do not have to be taken care of separately; when the program within the loop is combined into a region, the return path of the loop is also included in the region. For example, in Fig. 1(a) and Fig. 2(d), the return path, *P*, although not a part of the string or diamond, is a link between the subregions forming the region and is therefore included in the flow chart of that region. The example used to show the output of the Flowcharter is a program for which only STRING and DIAMND are needed.

The division subroutines attempt to discover particular configurations “in the large.” Two such subroutines are UNWRAP and SPLIT which look for loops and easily separable parts of the program. The division subroutines are not allowed to separate the regions already built up by the combination routines.

UNWRAP determines if the program is essentially one large loop; that is, it has an entry block E_1 , which has only one successor S , an exit block E_2 , which has only one predecessor P , and there is a path from P to S . In this case, the region representing the program is made up of three subregions: E_1 , E_2 , and the subregion including everything else. The last now becomes the “program” to be divided further.

SPLIT looks for the situation where the program is composed of several essentially distinct parts, each of which has only one entry point and one exit point for paths to or from other parts. Each such part is one subregion and is divided further if necessary.

At present, STRING, DIAMND, and TEST are used repeatedly until none of them can do any further combining. If there are no more than six regions left, these are combined to make the region representing the entire program. If there are more than six left, UNWRAP and SPLIT are used repeatedly until they have either divided the entire program into the regions left by the combination routines or can not divide it any further. In the latter case, at present, arbitrary divisions are made until the program is so divided.

This method should be adequate for most programs; however, the Flowcharter is written in such a way that routines can be added and other methods tried easily.

In planning the computation analysis, the major problem encountered was that of determining when cells or registers were used only as temporary or erasable storage. In order to keep the amount of information down to a readable size, we wanted to list only the cells actively used in the region. We started with the idea of labeling a quantity computed but not used as “output,” and those computed and then used, “tentative outputs” to indicate that they might be erasable cells. A “tentative output” was carried forward until an exit from the program or a use of the same quantity was encountered. If there was such a use, the “tentative output” became a real output—if not, it was considered erasable and would not appear further on the flow charts for that part of the program. Since a “tentative output” had to be carried forward on all possible paths but changed to a real output only on those paths on which a use was encountered, the bookkeeping necessary became unmanageable when the flow of the program was complicated.

If the computation is traced backward rather than forward, the procedure becomes much simpler. If a quantity is needed at one point of a program, it must be available along every possible path backwards from that point until some point is encountered where the quantity is computed or until an entrance to the program is encountered. In the latter case, this quantity must be available at that entrance to the program. With each region shown on a flow chart, all the quantities computed in that region are listed except those erasable cells which are used only within the region. For each quantity computed, there is given a list of quantities which are required at the entrances to the region and which enter into the computation of this item whether directly or indirectly.

The last part of the Flowcharter arranges and prints the results of the other sections. The appearance of the final flow charts will be one of the most important features to anyone using the Flowcharter and will be as much like hand-drawn flow charts as possible. Each page will show one region composed of as many as six or seven

smaller regions. Each of the boxes will indicate the entry and exit locations of the subregion it represents, the number of the page that has a detailed flow chart of the subregion, the location of any exits and entrances in the middle of the region and the exit conditions for any exit. Each box will be outlined by asterisks and all transfer paths will be represented by lines to the appropriate points. If the lines go outside the region, the name and page number of the instruction at the other end will be given at the top or bottom of the page for entrances or exits, respectively. The boxes themselves will be arranged in two dimensions to show the flow in the region as clearly as possible, using horizontal as well as vertical displacements of the boxes. On the right-hand side of the page will be listed further information in three columns: the names and numbers of any input-output units used, and which memory cells were involved in each case; a list of those quantities for which values must be available at the entrance to the region; and the computation summaries mentioned above. Provision is also made for reproducing comments, box titles, page titles, and other comments given by the programmer. (See the Appendix for an example of the output.)

STATUS OF THE PROGRAM

On February first, the program described here was nearly complete and checked out with a preprocessor for 704 SAP language. The parts not yet finished were UNWRAP, SPLIT, the drawing of the boxes and lines in the output program, and provision for some of the specifications by the programmer. In each case, much or all of the planning has been done.

As soon as these are complete, it is planned to write 705 and FORTRAN preprocessors and at least one region forming subroutine which considers mainly the data used by the program, and has much less emphasis on the flow properties than the routines described here. Also planned is some experimentation with various methods of region formation.

ACKNOWLEDGMENT

The author wishes to acknowledge the contributions of Alex Bernstein, who collaborated on the initial phases of the project, and of James Lagona, who wrote certain of the subroutines. The author also would like to thank colleagues in the Programming Research Department for helpful discussions and advice.

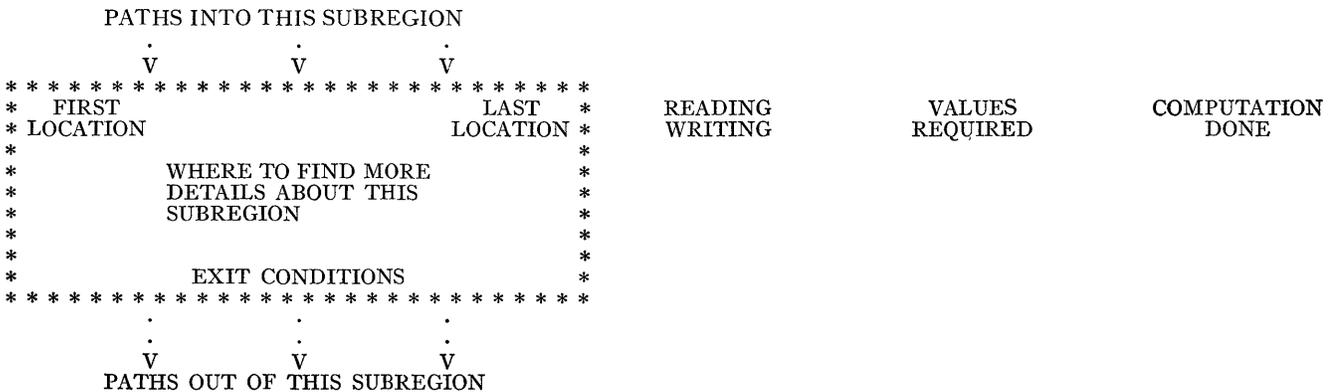
APPENDIX

The source program flow charted here is:

```

C   A PROGRAM TO MULTIPLY TWO MATRICES AND SUBSTITUTE PLUS ZERO FOR
C   EACH ZERO ELEMENT, PLUS ONE FOR EACH POSITIVE ELEMENT, AND MINUS
C   ONE FOR EACH NEGATIVE ELEMENT.
      10 READ 200 ((M(I, J), I=1, 3), J=1, 4), ((N(J, K), J=1, 4), K=1, 5)
      20 DO 140 I=1, 3
      30 DO 130 K=1, 5
      40 L(I, K)=0
      50 DO 60 J=1, 4
      60 L(I, K)=L(I, K)+M(I, J) * N(J, K)
      70 IF (L(I, K) 120, 100, 80
      80 L(I, K)=+1
      90 GO TO 130
     100 L(I, K)=0
     110 GO TO 130
     120 L(I, K)=-1
     130 CONTINUE
     140 CONTINUE
     150 PRINT 200 ((L(I, K), I=1, 3), K=1, 5)
     160 STOP
     200 FORMAT (15I4)
    
```

EXPLANATION OF CONTENTS OF BOXES IN THE FLOW CHARTS



PAGE 1

A PROGRAM TO MULTIPLY TWO MATRICES AND SUBSTITUTE PLUS ZERO FOR EACH ZERO ELEMENT, PLUS ONE FOR EACH POSITIVE ELEMENT, AND MINUS ONE FOR EACH NEGATIVE ELEMENT.

ENTRANCE TO PROGRAM	READING WRITING	VALUES REQUIRED	COMPUTATION DONE
V			
<pre> ***** * 10 ***** 20 * * P.3 * * UNCOND * ***** </pre>	<pre> READ CARDS . . . M(I, J) N(J, K) </pre>	<pre> M(I, J) . . . CARDS N(J, K) . . . CARDS I . . . +1 </pre>	
V			
<pre> ***** * 30 ***** 30 * * P.3 * * UNCOND * ***** </pre>			<pre> K . . . +1 </pre>
V			
<pre> ***** * 40 ***** 130 * * P.2 * * K IS TO 5 * * GREATER LESS, = * ***** </pre>		<pre> I K M(I, J) N(J, K) </pre>	<pre> K . . . K +1 L(I, K) . . . +0 -1 +1 </pre>
V			
<pre> ***** * 140 ***** 140 * * P.3 * * I IS TO 3 * * GREATER LESS, = * ***** </pre>		<pre> I </pre>	<pre> I . . . I +1 </pre>
V			
<pre> ***** * 150 ***** 160 * * P.3 * * STOP * ***** </pre>	<pre> PRINT . . . L(I, K) </pre>	<pre> L(I, K) </pre>	
V			
EXIT FROM PROGRAM			

PAGE 3

INSTRUCTIONS

FOR CONTEXT
SEE PAGE

C	A PROGRAM TO MULTIPLY TWO MATRICES AND SUBSTITUTE PLUS ZERO FOR	
C	EACH ZERO ELEMENT, PLUS ONE FOR EACH POSITIVE ELEMENT, AND MINUS	
C	ONE FOR EACH NEGATIVE ELEMENT.	
	10 READ 200 ((M(I, J), I=1, 3), J=1, 4), ((N(J, K), J=1, 4), K=1, 5)	1
	20 DO 140 I=1, 3	
	30 DO 130 K=1, 5	1
	40 L(I, K)=0	2
	50 DO 60 J=1, 4	
	60 L(I, K)=L(I, K) M(I, J) * N(J, K)	2
	(END OF DO AT 50)	
	70 IF (L(I, K) 12, 10, 80	2
	80 L(I, K) = +1	2
	90 GO TO 130	
	100 L(I, K) = 0	2
	110 GO TO 130	
	120 L(I, K) = -1	2
	130 CONTINUE	2
	(END OF DO AT 30)	
	140 CONTINUE	1
	(END OF DO AT 20)	
	150 PRINT 200 (L(I, K), I=1, 3), K=1, 5)	1
	160 STOP	

A Compiler Capable of Learning

RICHARD F. ARNOLD†

INTRODUCTION

WE WOULD like to consider a new approach to the general problem of programming computers. To date, the methods of handling programming problems can be roughly classified into two families, each of which have certain characteristic advantages and disadvantages which seem to complement those of the other.

The first group, developed from the subroutine philosophy, includes all interpretive schemes, as for example the "Bell Labs Interpretive System" for the IBM 650. The advantages of interpretive routines are that they are very versatile in the languages they can interpret and are comparatively easy to write. It is a fairly simple matter to write an interpretive routine to simulate another computer and thus achieve program compatibility between different machines. The crippling drawback is the excessive time needed to execute routines inter-

pretively. Higher order interpretive schemes increase executions time exponentially.

The second group consists of compilers and assembly programs. They are characterized by the fact that, unlike interpretive routines, they produce object programs which may be executed in reasonable amounts of time. Compilers, however, are difficult to write. "Fortran," for example, took twenty-five man years to write. A second difficulty of compilers such as "Fortran," is that although they are becoming more and more versatile, they still fail to express certain types of operations, and it has become necessary to make it possible to adapt the compiler so that the "Fortran" language may be temporarily left and programming done in a language closer to the initial machine language. Of course, this is a desirable feature for a compiler to have, but it does not solve the initial problem for which it was created, namely, to avoid machine languages completely. A further disadvantage is that as a compiler system becomes adapted for use on more than one computer, many of the "coding tricks" will have to be avoided. This may be desirable from the point of view of the compiler writer, but

† Michigan State University, East Lansing, Mich. This research was supported in part by a grant from the Natl. Sci. Found. and taken from a thesis written under the direction of G. P. Weeg.

optimum programs will not be written. This will be particularly true as newer generations of computers are built which may have less similarity to present ones than we expect.

Also, as new computers come into existence, it is likely that it will become increasingly difficult for even the trained programmer to use the machines efficiently. Only time can tell whether or not compilers and their improved offspring are the answer for the future.

However, we would like to offer a different approach to the problem which, although it will probably not prove to be a better solution for the present, may indeed offer a much better line of attack for the future. Let us review first what is desired. We would like a scheme that would accept a program, either in the pseudo code of a compiler or in a different machine language, and would produce an *object program* for a given machine. We would like this program to take full advantage of all the special features of the machine and to be coded in a fashion such that running time and program length are at a minimum. We would also like to avoid too much work in adapting this scheme to a new machine. Since we may not know ourselves just how best to program a given machine, we would like it if the scheme itself could find the optimum procedures and apply them. Finally—and this is the one requirement which will be most difficult to meet—the scheme must produce a program in a reasonable length of time and not require a tremendous amount of memory space.

With the exception of the last requirement, we believe that the following scheme is capable of satisfying all these requirements, and that perhaps this one too may be met. The techniques used have almost all been suggested in other contexts.¹ Freidberg's programs² also have some organizational resemblances to the compiler described.

A COMPILER USING A RANDOM PROGRAM GENERATOR

The Compiler

Let us consider a new kind of compiler. It is similar in function to other compilers in that it accepts a program in language A and produces an equivalent program in a language B. Since the requirement of our languages is that each order must specify exactly an operation, these languages define computers. Therefore, we may sometimes speak of computers A and B. The compiler operates as follows. It first generates a program in language B at random, as will be described below. Although the program must be of specified length, it may be any conceivable combination of orders. It then proceeds to determine whether this candidate program is equivalent to the given program in language A. The method of determining the acceptability of the program involves the

use of two interpretive routines, A and B, which are capable of executing the orders in the two languages. The subject program and candidate program are then both executed using identical data to ascertain whether the candidate program B is capable of producing identically the same output. If it is determined that they are equivalent, then this program will be punched out and the translation will be complete. If not, a new candidate program is produced and tested in a similar fashion. The process is repeated until an acceptable program is produced.

The Random Program Generator

The random program generator will have the following characteristics. Provided with the length desired, the first order of the program will be chosen at random from all possible orders that might appear there, so that the probabilities associated with the choosing of all orders are identical. The second order will be chosen in the same manner, and then each successive order in the program, until it is complete. This becomes the candidate program. This method could generate any conceivable program of a given number of orders. However, the probability that a particular program is generated is exactly the same for all. The program generator will utilize a random-number generating routine, and the range of the random numbers will be partitioned in such a way that by assigning equal intervals to each order and selecting that order which corresponds to the interval containing the random number, a program may be generated in the desired fashion. It is recognized that the computer can generate only "pseudo" random numbers and that this will introduce difficulties. It is not too much to expect that the numbers be distributed rectangularly. Of more importance, however, is the sequence that the generation of these numbers may take. It will be necessary that the random numbers not appear in a sequence so that the corresponding programs do not contain certain sequences of orders. We must continue to watch for this possibility even if the compiler works, since it may be producing only certain kinds of programs.

The Acceptability Criterion

The Interpretive Routine Simulation of Computers A and B: The simulation of the A computer will be straightforward. All one really needs to know is what the output is for a given input. However, we shall provide the interpretive routine that executes the A program with one additional feature, an "address stop." That is, the programmer, or in this case perhaps some other part of the compiler, will be able to specify, before transferring control to the A interpreter, a particular location which, if appearing in the control counter will cause the routine to jump to some previously designated location outside the interpretive routine itself. Any order which would otherwise cause the computer to stop would be treated in a similar fashion.

¹ "Automata Studies," *Ann. Math. Studies*, no. 34, pp. 215-277; 1956. Note last 4 articles by Ashby, Mackay, and Uttley.

² R. M. Freidberg, "A learning machine: part I," *IBM J. Res. Dev.*, vol. 2, no. 1, pp. 2-16; January, 1958.

The B computer will be simulated in a like manner with the address stop feature. The B interpreter will also be equipped with a "clock" that will keep track of the running time of the program it is executing as if it were being executed directly by a real computer B. Thus, it will be possible to discriminate between two acceptable programs which have different running times.

Also, by placing a limit on the length of time a program may run on interpreter B, it will have a means of discovering and stopping endless looping. The "clock" would be checked before each order was executed to see if the maximum time permitted had been exceeded.

Our final specification will be that these interpretive routines be in some standard form as it is hoped that to adapt the compiler to any other two languages, it will only be necessary to put in the interpretive routines for those languages and to provide the random program generator with a list of the orders comprising the object language.

The Use of Randomly Generated Data in Acceptability Tests: A first definition of equivalent programs only required identical outputs given identical inputs. It would be possible of course to require that for two programs to be considered equivalent, they must use the same algorithm. However, given a program, it is not possible to recover the algorithm used, and the only alternatives are either that the routine itself be treated as the algorithm or that we consider only results. Acceptance is therefore determined by a statistical criterion and the probability of correct answers on a production run may be made arbitrarily close to 1 by increasing the number of randomly selected sets of input data for which the candidate program must successfully compute the correct answers. It will, however, only be necessary to use one set of data until a candidate program succeeds in producing the correct output for that one. It will also be necessary that the range of the data variables be specified along with the subject routine.

The procedure will be to examine the output and compare it with that of computer A at the completion of the running of a program. If the output is different, the candidate program will be rejected as it would also be if a specified time on the "clock" were exceeded or if the computer "hung up." If the output is found to be identical to that of computer A, then a new set of data would be generated and run through computer A with the subject program to determine the correct output; and computer B would be given the same data and the candidate program executed again. When an acceptable program is found, it may either be punched out on cards or tape or else retained and a search made for a better program in terms of number of words and running time.

We would be fortunate indeed if the compiler, as it has been described, could produce a program of moderate size in the life of the machine, much less within the hour or so maximum that might be allowable. Nevertheless, aside from this difficulty, this compiler would do

everything else that was desired. If the compiler were allowed to run long enough so that it could choose among the best of many acceptable programs, it would not only tailor the program to the machine involved but also in fact find many new "coding tricks" that a programmer might never stumble upon. How then can the expected time involved in a translation be cut down? Two methods will be discussed. The possibility of breaking the subject program into parts and translating one part at a time will first be considered. Then we shall consider how the program generating part of the compiler may be replaced by a unit which will gradually modify the probabilities associated with the generation of programs in a manner such that they will produce acceptable programs more frequently.

Sectioning the Subject Program

Criteria for Sectioning: If it is desired to break the subject program into sections, it will be necessary that for a given section there be a criterion for the acceptability of that section. Also, it would be required that this criterion be such that if all the sections were correct, the entire program would also be correct. If a method of sectioning can be found and these requirements met, the compiler could then handle one section at a time in the same way it previously handled the whole program, and such a procedure might be much quicker.

The acceptance or rejection of the program described in the previous section was based on the fact that both computers had input and output devices and, furthermore, that these input and output devices were enough alike in the form in which they handled the data to make comparison easy. However, when considering a section of a program, we can no longer define correctness in these terms because it is unlikely that input and output operations even occur in that section. The input and output devices were treated as equivalent parts of the computer. However, other parts may also be treated as equivalent, and this will make possible the development of adequate criteria for the testing of sections. Since it will be the procedure to test sections by executing them, it will be necessary to specify a set of states, one of which computer B should be in at the termination of the execution of a section. Since this set must be determined by the state of computer A at the same stage, there must be a one-to-many mapping of the states of computer A onto B. This is obtained by defining equivalent parts. One natural equivalence is that the contents of the two memories should be in some sense analogous. It might be said that the state of A is equivalent to the state of B if their memories contain identical information. If it were required that every part of computer A have an equivalent part in B, then the normal operations of B would have to be abandoned and B made to mimic A's every move. This indeed would be undesirable since, for example, if it were required that B be in an equivalent state to A after every order, then even if it were a decimal machine, B would have to find a way in

which it could do operations such as forming logical products, which are extremely awkward in any system other than a binary one. Therefore, equivalent parts will only be defined when the equivalence is natural. Initially, this will mean that the memories, control counters, and input and output devices must be made to correspond. If only equivalent parts may be examined, it is necessary that only those parts of the computer may contain information relevant to the program. This then requires that our sections be constructed so that if the state of all other parts of computer A were altered at the time when tests for equivalent content were being made, there would be no interference with the correct operation of the program. This requirement will be used in sectioning the subject program. The subject program will be "cut" between those orders when the state of all nonequivalent parts of computer A may be altered without affecting the running of the program. The procedure will be that all possible "cuts" be made, the states of those nonequivalent parts altered on the basis of a random number, and the program continued. Those cases where no errors are introduced will then be recorded and the program divided at those points.

The comparison of the inputs and outputs, if any, and the control counters is fairly straightforward. The addresses specified in the control counter would have to be compared by ascertaining whether or not the sections referred to by the two addresses are equivalent.

The comparison of the two memories is more difficult. Regarding memory itself, it is clear that locations which have in them numbers that are computed functions of the data should be compared for equality. Orders which have been modified by a section must somehow be examined to see if they will perform correctly in their modified form. Also, the contents of other locations may have information which depends for its form on the order code of the particular computer, for example, "binary switches" and "dummy" orders. It is necessary then to establish the correctness of such locations by linking them with other sections. Thus we can conclude that if these sections are acceptably executed throughout the entire program, then the location in question must be acceptable. A similar procedure is used for sections whose function is to modify instructions. For each section this entails making a list of sections whose acceptance must come prior to the acceptance of that section. In some cases it may be necessary to work backwards from sections which only operate on data and may be checked immediately through several different sections, the acceptance of each one being a necessary requisite for the acceptance of a prior one.

Admittedly, the above discussion is not complete and there may be situations arising which we have not considered. Nevertheless, it is felt that the method itself is powerful and can probably be adapted to new difficulties as they arise.

Expected Running Times: The purpose of sectioning the program is, of course, that it is desirable not to

throw out a program completely when only a part of it is incorrect. However, though it might seem obvious that the expected running time would be reduced, this does not follow just because the program is being compiled in sections, as will be seen.

Consider first the probability that an entire program generated by the random program generator is exactly a given program. If the program contains l orders, each of which might be any of k different orders, this probability will be $1/k^l$. The expected number of trials until it is constructed will be k^l trials. Now, if the program is sectioned into g parts of lengths m_1, m_2, \dots, m_g , then the probability of a particular candidate section being a particular one is $1/k^{m_i}$, the expected number of trials is k^{m_i} , and the expected number of trials for the entire program is

$$\sum_{i=1}^g k^{m_i}.$$

However, this is not the situation. In fact, there are many acceptable programs. If there are c acceptable programs of length l , then the expected number of trials to hit one of them will be k^l/c , and if the program is divided into g sections with lengths m_1, m_2, \dots, m_g and there are c_i acceptable ways of writing the i th section, then the expected number of trials to translate the entire program will be

$$\sum_{i=1}^g k^{m_i}/c_i.$$

If

$$\prod_{i=1}^g c_i = c,$$

then it could be concluded that the expected number of trials for the procedure using sectioning is much less than the other one provided the minimum expected number of runs for any section is greater than $1/g^{g-1}$. For $g > 1$, $1 < (1/g^{g-1}) \leq 2$ and the expected number of runs for any section will never become that low. However, by requiring the program to achieve certain criteria at many points during its execution, many programs that would satisfy the criteria of identical outputs will be rejected. Therefore, in general

$$\prod_{i=1}^g c_i < c.$$

There is no *a priori* method then of saying conclusively that sectioning will lower the expected number of runs, although it seems that it would except in unusual circumstances. However, it might at some point be possible to pool some of the sections and reduce expected translation time, while at the same time increasing the probability of producing a better program because of the fewer restrictions.

THE ADAPTION OF LEARNING MODELS TO
THE COMPILER

The Alteration of the Random Program Generator to Permit Learning

So far, the random program generator has selected each of the k possible alternatives at each step with equal probabilities. It is entirely possible to allow the orders to be selected with other probabilities. Given a set $\{P\}$ of probability vectors $P = (p_1, p_2, \dots, p_k)$ such that

$$\sum_{i=1}^k p_i = 1$$

where p_i is to be associated with order O_i , the selection of an order may be made in the following manner. Given a pseudo random number R such that $0 \leq R \leq 1$, and a vector P , then O_j will be selected when

$$\sum_{i=1}^{j-1} p_i \leq R < \sum_{i=1}^j p_i$$

If R has a rectangular distribution as it should have, then O_j will be selected with probability p_j . A specific selection of an order will constitute a *response*, and the set of probability numbers (p_1, p_2, \dots, p_k) will be abbreviated P .

Intuitively, it is felt that some P 's will be more satisfactory than others, and a method must be found to arrive at some "best" one. First, some measure of performance is needed. The measure might include not only how often a particular P produces an acceptable program, but also whether the program is concise and has a short running time. However, it will be more convenient at present only to consider a $U(P) = Pr$. (A candidate program using P is acceptable.) It would then be desirable to find the maximum of this function where P ranges over the set $\{P\}$ and use the corresponding P . At present there is no information about this function. A guess might be made that it is continuous but has several modes. It will be assumed, however, that it has but one, the hope being that if any of them are discovered, a fairly satisfactory selection of orders will ensue.

One method that might be used would be to modify the p_i 's whenever an acceptable program is achieved, or if sections are being dealt with, an acceptable section. Given that a particular order O_e were in the acceptable section, the current set of p_i 's would be replaced by

$$\left[\left(p_1 - \frac{p_1}{d} \right), \left(p_2 - \frac{p_2}{d} \right), \dots, \left(p_e + \frac{(1 - p_e)}{d} \right), \dots, \left(p_k - \frac{p_k}{d} \right) \right]$$

where d is a parameter that would reflect the magnitude of the desired change. This procedure would be repeated for each order in the acceptable section. The expected

value of P can be shown to approach in expectation the value at which $U(P)$ has its mode. It is desirable to let d be a function of the number of modifications already made, so that as a modal value is approached, the variance about it will be decreased.

This procedure might be called learning because it permits an increment in performance to occur as a result of the "experience" of the compiler.

The Stimulus Situation

Consider now the situation of the response selector at a given point in its selection of orders for a candidate section. Previously, the P 's used for the selection of each order in the program have been identical. However, if P_y is the set of probability vectors that might be associated with the selection of a particular response y , it would seem that $\max U(P_y)$ will occur at different values of P for different values of y . In order to devise a method for obtaining these different values of P , there must first be a method of classifying the y 's. It may seem at first that y could be classified according to its location alone in the program, but this would not help a great deal. This seems to have been one of the difficulties of Freidberg's programs. It is clear that most of the deviation of P_y from P may be explained in terms of y 's relation to the orders around it in the program. For example, P_y should certainly depend on the selection that has already been made for $(y-1)$. Since $\max P_y$ is conditional upon the value of $(y-1)$, a set of k P 's may be used, one for each value of $(y-1)$. In general, not only will $\max P$ be dependent on $(y-1)$ but also on $(y-2)$, $(y-3)$, and in fact on many other variables that might not even be defined in the candidate program.

Other variables that should be considered are the particular orders that make up the subject section being translated and the contents of certain locations in both the interpretive routines used. While the relationship of the subject orders might be obvious, that of the interpretive routines is probably not. The interpretive routines do contain variables in the sense that $(y-1)$ is a variable, for example, the "left-right counter" necessary in an interpretive routine simulating Princeton-type machines having two orders stored in one location in memory. In fact, it might be suspected that since a human being can know all there is to know of a computer's importance to programming by examining an interpretive routine simulating it, indeed a great deal might be gained by considering the relationship between $\max P_y$ and some of these variables. The reason for the concern with the dependence of all these variables with $\max P_y$ is that prior to selecting the value of y , the particular values that these other variables have taken will be known; and if the relationship is also recognized, a better P_y may be used. Certain definitions follow naturally. If $X = (x_1, x_2, \dots, x_n)$ is a set of variables whose values may be determined prior to the selection of order y and $S = (x_{1i}, x_{2i}, \dots, x_{ni})$, a set of particular values for X , then $\max P_{y,S}$ will mean that value of

$\{P\}$ for which $U(P_{y \cdot s}) = P_r$ (a response chosen using $P_{y \cdot s}$ is acceptable given $X = S$) is maximized. S is called the stimulus vector, and x_1, x_2, \dots, x_n , stimulus variables.

The Selection of the Response

Considering again the selection of the value of y , it is desired to find $\max U(P_{y \cdot s})$. If the best $P_{y \cdot s}$ were determined independently for each possible S and if $d=1$, that is, if p_c becomes 1 when c is the correct response for the specified S , and if all possible stimulus variables were included, the scheme would have these characteristics. It would classify completely all the correct responses corresponding to the given stimulus vectors. But the same situation, that is, S , would never be encountered twice unless the scheme were again asked to translate the same program. Therefore, it would never be any better than a scheme which took no cognizance of the stimulus situation. Of course, this method is impossible because of the space requirements. Restricting the number of stimulus variables might be one way out. However, if one such variable could take on the average 40 values, the p 's expressed to five binary digits and $k=40$, the 512,000,000 words of 40 bits necessary to store the p 's for just five variables would be prohibitive.

A Linear Modal

Clearly the trouble lies at least partly in allowing the $\max P_{y \cdot s}$ to be freely determined for every S . While it is true that in general the effect of one stimulus variable x_j on $\max P_{y \cdot s}$ will depend upon the values of the other stimulus variables, it is not necessarily the case that the effect of x_j on $\max P_{y \cdot s}$ will change for every change in the value of any stimulus variable.

A procedure that would take advantage of this fact might be developed along the lines of linear regression theory. The modal assumes that $\max P_{y \cdot s}$ is a linear combination of $\max P_{y \cdot x_{j \cdot i}}$ where $P_{y \cdot x_{j \cdot i}}$ represents the i th probability vector associated with the stimulus variable x_j . Given then an $S = (x_{1i}, x_{2i}, \dots, x_{ni})$, the value $P_{y \cdot s}$ to be used in the selection of y would be

$$P_{y \cdot s} = b_1(\max P_{y \cdot x_{1i}}) + b_2(\max P_{y \cdot x_{2i}}) + \dots + b_n(\max P_{y \cdot x_{ni}}).$$

The $\max P_{y \cdot x_{j \cdot i}}$'s would be estimated in the standard way; $P_{y \cdot x_{j \cdot i}}$ would be modified to increase p_c when O_c was an acceptable order occurring at y when x_j had value i .

b_j should be increased as all the individual probabilities within $\{P_{y \cdot x_j}\}$ tend away from $\max P$ ($\max P$ taken in the sense of the last section). This could be accomplished by letting b_j increase in increments according to a sample taken from a random variable highly correlated with

$$\sum_{i=1}^k (P_{y \cdot x_{j \cdot i \cdot h}} - P_h)^2$$

where $P_{y \cdot x_{j \cdot i \cdot h}}$ is the h th component of $\max P_{y \cdot x_{j \cdot i}}$ and P_h is the h th component of $\max P$. Likewise, b_j should be decreased as x_j shows increasing correlation with the other variables of X .

It is suggested that a measure of how much b_j should be lowered because of its correlation with other variables might be some function of the similarity of the particular $P_{y \cdot x_{j \cdot i}}$ to the other $n-1$ vectors determined by the particular S .

A method for the selection and rejection of stimulus variables would follow naturally. At periodic intervals that x_j whose b_j were lowest would be discarded, and a new one selected at random from the remaining variables not currently represented in the stimulus variable set.

The model may be made even more flexible by permitting variables which are the products of two or more stimulus variables.

A dubious future would seem to be ahead for the model described above if required directly to "learn" how to translate programs for an 1103A to 704 language. The unfortunate situation is, however, that almost nothing is known of the joint distributions of what have been called the stimulus and response variables in such a translating endeavor, and therefore our judgment of any model, prior to its incorporation in a translator program, must be intuitive.

AN EXAMPLE

For the purpose of exhibiting some of the general characteristics of the type of translator proposed, a program has been written that does in fact compile complete programs for an imaginary one-address machine given those for an imaginary three-address machine. To be sure the two machines used are neither of the size nor complexity of current machines, nor are they as different from each other as current machines. Nevertheless they are sufficiently powerful to be able to compute transcendental functions, invert small matrices, etc.

Our experience with this compiler is limited and a full report on its performance is not yet ready; however, our results to date have proved both educational and encouraging.

Our estimations of the initial expected time of translation was of the order of one hour per instruction of the subject program. After gaining experience, the translator should reduce this to about 90 seconds per subject instruction. In the first programs translated, the translator did considerably better than anticipated. Upon examining the programs produced, the reason became evident. We had overlooked a large class of acceptable programs, namely, those in which numbers were left in the arithmetic register of the one-address machine at the termination of a section; those same numbers could be of use in the succeeding section. The translator had promptly taken advantage of this.

The principal difficulty encountered so far has come with conditional transfers of control. The trouble lies

in that the acceptance of a sequence containing a conditional transfer is contingent on its correct operation at several points in the program. Therefore, a large portion of the program must be executed before a candidate unit may be rejected or accepted. This implies that for very long programs the probability of success for a particular candidate unit containing a conditional transfer is going to have to be much higher than would be necessary for a purely arithmetic unit. This same difficulty will arise in the situation of units whose function is to modify instructions.

The translations produced have been far from optimal, primarily because the length of unit translated at one time was one subject order. If two at a time had been taken, the initial expected running times would have been much too great. However, the present translator may now be modified to use its experience on one-at-a-time translation in translating two at a time, and the initial expected running times will be reasonable. This suggests that eventually the translator should choose the size of the unit it attempts to translate according to the subject orders involved. Thus the translator would reduce the size of the unit attempted if it

were recognized as one with too high an expected translation time, and increase the size as it gained experience.

CONCLUSION

It was suggested at the beginning of this paper that the type of compiler described might not be of immediate use. In the author's opinion, this is not because our machines are too small and too slow, although certainly larger random access memories would be helpful. The cause seems to lie more in our ignorance of machine design and programming, which amounts to essentially the same thing. Any increases in speed are quickly absorbed in the realm of combinatorics.

It is hoped that the ideas expressed here will have some value in finding the solution to this large class of problems, which includes not only compilers but human-language translation, game playing, and other problems where the initial complexities suggest a solution that is self-improving. Development of information processing theory will certainly make these modified British Museum techniques obsolete. Nevertheless, they may be of value in the interim and may be able to contribute to the more rapid development of that theory.

Special-Purpose, Electronic Data Systems—The Solution to Industrial and Commercial Automation

WILLIAM V. CROWLEY†

INTRODUCTION

THE concept that any "standard" electronic data processing or computing system should be adapted solely through programming to suit a particular business or industry is no longer supportable, except as an interim step for testing the planning and design of the data system through simulation. Nor does the answer to making "standard" electronic data systems more easily adaptable to various business applications lie in the area of so-called automatic programming. This is not to imply that this endeavor has not or will not be valuable and useful.

Its main contribution, however, will be to force data definition discipline in business systems so that more powerful electronic data processing equipment, with more pertinent and functional instructions and commands, will be built.

A careful study and comparison of the current data systems of many different types of business enterprises

† Information Systems Dept., Ramo-Wooldridge Corp., Los Angeles, Calif.

will show considerable similarity in the general structure and media of the data systems, such as files, action documents, information documents, reports, etc., but will disclose numerous differences in the composition and arrangement of the data itself, as well as management's philosophy of and concern with data handling. Some of these differences are degree of variability of the length of the data items; relative occurrence of alphabetic and numeric information; language idioms or usage peculiar to the business or industry; attitudes and practices concerning the coding of information; government regulations; accounting customs, etc. Because of these existing variable factors and the social and economic resistance to absolute standardization, industrial and commercial data systems require equipment designed, not adapted, for their particular situation.

NEGATIVE INFLUENCES

Let us review some of the important factors which tend to inhibit this inevitable trend toward the building of electronic data systems tailored to a particular in-

dustry, type of business, public utility, or governmental organization.

Perhaps the most important single deterrent to more rapid development and use of "specialized" general purpose electronic data systems is the lack of realization, understanding, and acceptance by top management officials of the ultimate advantages of a tailored, controlled data system in efficiency; compression of traditional time cycles; provision of real cost reductions; and detection and elimination of errors. For example, most manufacturing executives are willing to invest in and work out financing for an expensive machine tool to improve performance and output. Many are still reluctant to invest the money and effort associated with implementing a large electronic data system. As the middle management of today moves into the top jobs the pace of electronicization of data flows will increase markedly.

The second most influential inhibiting factor is that the business machine manufacturing industry is largely anchored to the past both philosophically and economically. Conceptually, the idea of integrating the informational aspects of business enterprises, and the design of techniques to accomplish this, did not come primarily from machine manufacturers, but from business systems analysts and imaginative managers who first understood the data flow requirements of business and then sought to learn of the capability of existing electronic machines. Many of the first electronic data systems were oversold through use of traditional marketing approaches. The state of the art of design and manufacture of electronic machines has advanced rapidly in the last four years. The art of applying the machines effectively has also advanced, but not as rapidly. Some commercial enterprises such as insurance companies have advanced markedly in data systems understanding. Others, such as manufacturing concerns, have not progressed as much.

Another important reason for a lag in advancing toward the use of special electronic data systems for general purpose use in a particular industry is the matter of communication and agreement between the equipment designers and manufacturers; business systems analysts; equipment programmers and operators; and business operating and staff managements. The differing objectives of these groups and shortage of managers who understand all these points of view have impeded the faster introduction and use of electronic data systems in business. The designers and manufacturing personnel are faced with keeping costs down, and as a result important equipment functions may be left out. Similarly, a clever equipment design feature may contribute nothing but higher cost to the user. Competent business systems analysts who really understand the objectives of the enterprise and the data processing requirements are considered staff and are often not placed sufficiently high in the organization to enforce their improvements. There is a continual job of "selling" both to top management and operating management. The programming

staff is always seeking to ease its function, and if a very precise statement of the problem is not presented, interpretations will be made which may weaken or change the desired end result. Top management wants quick results, whereas the relative efficiency of the existing system, and the specified objectives, and the inherently complicated planning and implementation procedure require a considerable investment in time, personnel, and money. Since a justifiably efficient EDP system implies functional integration, operating management may be reluctant to accept the inevitable re-centralization of authority.

Many attempts are being made to reconcile these points of view through EDP equipment users' organizations, industry committees, conventions such as this one, and university and special courses which attempt to explain and present the various points of view adequately.

FAVORABLE INFLUENCES

There are several distinguishable factors and developments which promote and support the concept of "specialized" general purpose electronic data systems.

Perhaps the most important positive indication of this trend is the existence and development of several such specialized electronic data systems and projects in different organizations. Some of the better known examples are electronic equipment to maintain the status of airline reservations; equipment tailored to banking operations; equipment suitable primarily for the control of process industry operations; equipment especially designed for hospital operations; and government special systems for capturing, assimilating, and presenting defense warning information for field army operations, and data needed in U. S. Air Force logistics.

Increasing costs of comparable standard systems and progress in the state of the art make it possible to obtain a specialized electronic data system at not too much greater cost than the ultimate cost of implementing a "standard" adapted system. Furthermore, the buyer is assured of a well operating system because of specially negotiated contractual terms. The increasing costs of standard systems are associated with increasing marketing costs imposed by competition, the need for more highly trained people to harness the more powerful equipment, and higher development and manufacturing costs.

Another significant indication of this trend is found in the actions of certain well-known accounting and business machine manufacturers to concentrate on a specific industrial or commercial business.

BENEFITS OF THE TREND

There are many advantages which will accrue from the extension of this predicted trend, toward the use of specialized general purpose electronic data systems.

For the user this development will mean that the buyer and user are assured of getting a data system that

will work. Because of individually negotiated contracts, there will be no vague paragraphs in contracts about training of personnel, library and subroutines to be furnished, assistance in system analysis and programming, availability of various techniques of adaptation, and the attempt to force the user's system into an artificial equipment usage schedule which is not related to the specific situation. Under present terms the user will get a more or less good job done depending on the strength or weakness of the local equipment manufacturers' office.

For the equipment manufacturers the most advantage will accrue to those who accept this trend first. It will also mean some changes in marketing and operating methods. The highly skilled sales team approach to selling will be required. Thus many manufacturers will be placed at a temporary disadvantage depending on their present situation. Ultimately, manufacturers will benefit in that, although they may have to apply more selling effort and customer assistance, there is a greater chance that they will be paid more fairly and adequately for their efforts. The relationship of manufacturers and buyers will be clarified, and there will be less chance for dissatisfied users—a condition which is harmful to the broad concept of electronic data systems and the entire industry.

For the state of the art, advancement will accelerate because, with the closer user-manufacturer relationship the buyer will be willing to pay for advanced equipment built especially for him.

For personnel trained and experienced in electronic data system techniques and an understanding of the business data requirements of the various commercial, industrial, and governmental enterprises, there will be expanded opportunities for employment and innovative work.

For the economy as a whole, we can look for continued increased productivity per individual, and the ultimate, virtual elimination of the wasteful errors of carelessness, misinterpretations and the varying application of logical rules which plague the modern business enterprise in its daily operations.

RESEARCH AND DEVELOPMENT REQUIRED

Before generalized special electronic data systems reach the anticipated rate of installation more pure and developmental research must be made in several areas.

The first field of development has to do with further scientific investigation into the information requirements of the various levels of management in the several types of business enterprises. In many industries the creation, maintenance, and dissemination of data has become an end in itself. De-integration and compartmentalization of larger business organizations has given custody and control of certain portions of vital operational data to superficial or minor organizational units which are their only source of power. Often these data are in the form of reference files, where the chief

file clerk is a force to be reckoned with in the existing administrative mechanism. Other artificial administrative data terminals may be illustrated by a coding section where data are coded for facilitation of processing and summarization. Still another artificial administrative data terminal is found in the specialty of cost estimating. Usually the data required to estimate any particular cost reside in or are generated by:

- 1) Vendors' files outside the company
- 2) The engineering section
- 3) Material control and purchasing section
- 4) Production planning section
- 5) Accounting department records
- 6) Payroll department
- 7) Written or verbal policies of profit margins and markups.

These data are also required for many other purposes. Why then should not one central company file of necessary business data be maintained for presentation to the functional area officials as required? Under this concept management can truly manage, by exercising direct control over that vital body of company operational intelligence.

A second area of investigation should be into the nature of various functional areas themselves. For example, such obvious questions as: What is production control? What information is used in making production control decisions? Where does it originate? Where does it go? How is it used? etc. For every type of business a study must be made by function, noting any peculiarities of a particular function in the specific business. The peculiarities themselves must then be analyzed to determine the significance of the differences. After logical conclusions are reached a foresighted, enlightened management is required to implement the findings.

These investigations can best be performed by competent independent organizations such as the business schools, research organizations, consultants, and industry committees.

EQUIPMENT DESIGN

Concurrent with applications, research should be continuing in equipment design. It should be universally accepted among business data processing specialists that information files are the centers or center from which data to form decision patterns must come. Therefore, the handling of file maintenance, file reference, and data organization should be the primary area of research. The other significant problem is the accurate, rapid capture of raw data as they occur. In addition to the physical devices needed to capture the data there is often the problem of transmitting the data to some central location. This problem is more or less simple depending on the distance involved and the format of the data.

Most of this is and should be carried on by the various manufacturers. There is an economic cost to this of

course, and unless some of the research is done by endowed organizations, the immediate costs are likely to be high.

There must also be a closer liaison between the digital data processing engineers and the communications engineers. As electronic data systems become more responsive, communicating and transmitting devices will be needed to connect the data processing center with the various segments of the system. Terminal data transfer and translation problems must be solved to permit the ultimate automation of data manipulation that is logically feasible.

CONCLUSION

The demise of the medium and large-scale general purpose electronic data processor or computer for business purposes is in sight. A sufficient number of indus-

trial and commercial procedural analysts are now able to specify their data system requirements with cognizance of the speed and ability of electronic devices so as to build what is needed—not use just what is available. Small general purpose computers and large capacity computers for scientific calculation will continue in long usage.

Many large companies with special electronic data handling problems have found the traditional large manufacturers of business machines unwilling to do more than tie together existing standard lines of equipment. Often unwilling to entrust the smaller electronic manufacturers with their problems, several companies have embarked on their projects of tailor-made electronic systems.

I predict that this trend will continue until, or unless, some better-known companies enter the field.

The Residue Number System

HARVEY L. GARNER†

INTRODUCTION

IN THIS PAPER we develop and investigate the properties of a novel system, called the residue code or residue number system. The residue number system is of particular interest because the arithmetic operations of addition and multiplication may be executed in the same time as required for an addition operation. The main difficulty of the residue code relative to arithmetic operations is the determination of the relative magnitude of two numbers expressed in the residue code. The residue code is probably of little utility for general-purpose computation, but the code has many characteristics which recommend its use for special-purpose computations.

The residue code is most easily developed in terms of linear congruences. A brief discussion of the pertinent properties of congruences is presented in the next section.

CONGRUENCES

The congruence relationship is expressed as

$$A \equiv \alpha \pmod{b}$$

which is read, A is congruent to α modulo b . The congruence states that

$$A = \alpha + bt$$

is valid for some value of t , where A , α , b , and t are integers, α is called the residue, and b the base or modulus of the number A .

As examples of congruences, consider

$$10 \equiv 7 \pmod{3}$$

$$10 \equiv 4 \pmod{3}$$

$$10 \equiv 1 \pmod{3}.$$

In these examples the integers 7, 4, and 1 form a residue class of $10 \pmod{3}$. Of particular importance is the least positive residue of the class which in this example is one. The least positive residue is that residue for which $0 \leq \alpha \leq b$.¹

Consider the following set of congruences:
Given

$$\begin{aligned} A_1 &\equiv \alpha_1 \pmod{b} \\ &\vdots \\ &\vdots \\ A_n &\equiv \alpha_n \pmod{b}. \end{aligned}$$

Then

- 1) Congruences with respect to the same modulus may be added and the result is a valid congruence.

$$\sum_{i=1}^n A_i \equiv \left(\sum_{i=1}^n \alpha_i \right) \pmod{b}.$$

¹ The equality sign may exist on only one side of the expression.

† University of Michigan, Ann Arbor, Mich.

It follows that terms may be transferred from one side of a congruence to the other by a change of sign and also that congruences may be subtracted and the result is a valid congruence.

- 2) Congruences with respect to the same modulus may be multiplied and the result is a valid congruence.

$$\prod_{i=1}^n A_i \equiv \left(\prod_{i=1}^n \alpha_i \right) \pmod{b}.$$

It follows that both sides of the congruence may be raised to the same power or multiplied by a constant and the result is a valid congruence.

- 3) Congruences are transitive. If $A \equiv B$ and $B \equiv C$, then $A \equiv C$.
- 4) A valid congruence relationship is obtained if the number, the residue, and the modulus are divided by a common factor.
- 5) A valid congruence relationship is obtained if the number and the residue are divided by some common factor relatively prime to the modulus.

The material of this section has presented briefly, without proof, the pertinent concepts of congruences. Additional material on the subject may be found in any standard text on number theory.²

DEVELOPMENT OF THE RESIDUE CODE

A residue code associated with a particular natural number is formed from the least positive residues of the particular number with respect to different bases. The first requirement for an efficient residue number system is that the bases of the different digits of the representation must be relatively prime. If a pair of bases are not relatively prime, the effect is the introduction of redundancy. The following example will illustrate this fact. Contrast the residues associated with bases of magnitude 2 and 6 against the residues associated with bases of magnitude 3 and 4. In the first case, the bases are not relatively prime while in the second case the bases are relatively prime. The residues associated with the bases of magnitude 2 and 6 are unique for only 6 states while the residues associated with the bases of magnitude 3 and 4 provide a unique residue representation for 12 states. This is further clarified by Table I.

An example of a residue number system is presented in Table II. The number system shown in Table II uses the prime bases 2, 3, 5, and 7. The number system, therefore, contains 210 states. The 210 states may correspond to the positive integers 0 to 209. Table II shows the residue number representation corresponding to the positive integers 0 to 29. Additional integers of the number system may be found by congruence operations. Let $a, b, c,$ and d be the digits associated with the bases 2, 3, 5, and 7, respectively. The following congruences

TABLE I
REDUNDANCY OF A NONRELATIVELY PRIMED BASE REPRESENTATION

Least Postive Residue				
Number	Mod 2	Mod 6	Mod 3	Mod 4
0	0	0	0	0
1	1	1	1	1
2	0	2	2	2
3	1	3	0	3
4	0	4	1	0
5	1	5	2	1
6	0	0	0	2
7	1	1	1	3
8	0	2	2	0
9	1	3	0	1
10	0	4	1	2
11	1	5	2	3
12	0	0	0	0
13	1	1	1	1
14	0	2	2	2

TABLE II
NATURAL NUMBERS AND CORRESPONDING RESIDUE NUMBERS

Natural Numbers	2357	Natural Numbers	2357	Natural Numbers	2357
0	0000	10	0103	20	0206
1	1111	11	1214	21	1010
2	0222	12	0025	22	0121
3	1033	13	1136	23	1232
4	0144	14	0240	24	0043
5	1205	15	1001	25	1104
6	0016	16	0112	26	0215
7	1120	17	1223	27	1026
8	0231	18	0034	28	0130
9	1042	19	1145	29	1241

TABLE III
NUMBER OF STATES AND DIGITS ASSOCIATED WITH A RESIDUE REPRESENTATION

i	p_i	$\sum_{i=1}^n p_i$	$\prod_{i=1}^n p_i$	p_i bits	$\sum p_i$ bits
1	2	2	2	1	1
2	3	5	6	2	3
3	5	10	30	3	6
4	7	17	210	3	9
5	11	28	2,310	4	13
6	13	41	30,030	4	17
7	17	58	510,510	5	22
8	19	77	9,699,690	5	27
9	23	100	223,092,670	5	32

define $a, b, c,$ and d for the residue representation of the number N :

$$\begin{aligned} N &\equiv a \pmod{2} \\ N &\equiv b \pmod{3} \\ N &\equiv c \pmod{5} \\ N &\equiv d \pmod{7}. \end{aligned}$$

The residue number system is readily extended to include more states. For example, if a base 11 is added to the representation, it is then possible to represent 2310 states. Table III shows the product and sum of the first nine consecutive primes greater than or equal to 2.

²G. H. Hardy and E. M. Wright, "An Introduction to the Theory of Numbers," Oxford University Press, London, Eng.; 1956.

The product of the primes indicates the number of states of the number system, while the sum of the primes is a measure of the size of the representation in terms of digits. Table III also includes the number of bits required to represent each prime base in the binary number system.

RESIDUE ADDITION AND MULTIPLICATION

The residue number representation consists of several digits and is assumed to be in one-to-one correspondence with some positive integers of the real number system. The digits of the residue representation are the least positive residues of these real positive integers with respect to the different moduli which form the bases of the residue representation. It follows as a direct consequence of the structure of the residue number system and the properties of linear congruences that operations of addition and multiplication are valid in the residue number system subject to one proviso: the residue system must possess a number of states sufficient to represent the generated sum or product. If the residue number system does not have a sufficient number of states to represent the sums and the products generated by a particular finite set of real integers, then the residue system will overflow and more than one sum or product of the real number system may correspond to one residue representation. For a residue number with a sufficient number of states, an isomorphic relation exists with respect to the operations of addition and multiplication in the residue system and a finite system of real positive integers.

Each digit of the residue number system is obtained with respect to a different base or modulus. It follows, therefore, that the rules of arithmetic associated with each digit will be different. For example, the addition and multiplication of the digits associated with moduli 2 and 3 follow rules specified in Table IV. No carry tables are necessary since the residue number system does not have a carry mechanism. Addition of two residue representations is effected by the modulo addition of corresponding digits of the two representations. Corresponding digits must have the same base or modulus. Modulo addition of digits which have different bases is not defined. Multiplication in the residue system is effected by obtaining the modulo product of corresponding digits. The operations of addition and multiplication of two residue numbers are indicated by the following notation:

$$S = A \oplus B$$

$$p = A \odot B.$$

Consider a residue number representation with bases 2, 3, 5, and 7. We assume an isomorphic relation between the residue number system and the real positive numbers 0 to 209. An isomorphic relation then exists for the operations of multiplication and addition only if the product or sum is less than 210. The following examples

TABLE IV
MOD 2 AND MOD 3 SUMS AND PRODUCTS

$\begin{array}{r l} \oplus & 0 \ 1 \\ \hline 0 & 0 \ 1 \\ \hline 1 & 1 \ 0 \\ \hline \text{sum mod 2} & \end{array}$	$\begin{array}{r l} \oplus & 0 \ 1 \ 2 \\ \hline 0 & 0 \ 1 \ 2 \\ \hline 1 & 1 \ 2 \ 0 \\ \hline 2 & 2 \ 0 \ 1 \\ \hline \text{sum mod 3} & \end{array}$	$\begin{array}{r l} \odot & 0 \ 1 \ 2 \\ \hline 0 & 0 \ 0 \ 0 \\ \hline 1 & 0 \ 1 \ 2 \\ \hline 2 & 0 \ 2 \ 1 \\ \hline \text{product mod 3} & \end{array}$
--	--	---

employing residue numbers illustrate the addition and multiplication operations and the presence of an isomorphism or the lack of isomorphism in the case of overflow. Residue numbers will be distinguished by the use of parentheses.

$$\begin{aligned} 29 + 27 &= S = 56 \\ 29 &\leftrightarrow (1 \ 2 \ 4 \ 1) \\ 27 &\leftrightarrow (1 \ 0 \ 2 \ 6) \\ 56 &\leftrightarrow (0 \ 2 \ 1 \ 0) \\ &\oplus \begin{array}{l} (1 \ 2 \ 4 \ 1) \\ (1 \ 0 \ 2 \ 6) \\ \hline (0 \ 2 \ 1 \ 0) \end{array} \end{aligned}$$

The following operations are considered in performing the addition of the two residue representations:

$$\begin{aligned} 1 + 1 &\equiv 0 \pmod{2} \\ 2 + 0 &\equiv 2 \pmod{3} \\ 4 + 2 &\equiv 1 \pmod{5} \\ 1 + 6 &\equiv 0 \pmod{7}. \end{aligned}$$

Consider the addition of two numbers which produce a sum greater than 209.

$$\begin{aligned} S &= 100 + 200 \\ &\oplus \begin{array}{l} (0 \ 1 \ 0 \ 2) \\ (0 \ 2 \ 0 \ 4) \\ \hline (0 \ 0 \ 0 \ 6) \end{array} \end{aligned}$$

The residue representation (0 0 0 6) corresponds to the real positive number 90. In this particular example, the sum has overflowed the residue representation. The resulting sum is the correct sum modulo 210.

$$300 \equiv 90 \pmod{210}.$$

Finite real number systems and residue number systems have the same overflow characteristics. The sum which remains after the overflow is the correct sum with respect to a modulus numerically equal to the number of states in the finite number system.

The following is presented as an example of the process of residue multiplication:

$$\begin{array}{r}
 p = 10 \times 17 = 170 \\
 10 \leftrightarrow (0 \ 1 \ 0 \ 3) \\
 17 \leftrightarrow (1 \ 2 \ 2 \ 3) \\
 170 \leftrightarrow (0 \ 2 \ 0 \ 2)
 \end{array}
 \quad \odot \quad
 \begin{array}{r}
 (0 \ 1 \ 0 \ 3) \\
 (1 \ 2 \ 2 \ 3) \\
 \hline
 (0 \ 2 \ 0 \ 2)
 \end{array}$$

The process of multiplication involved consideration of the following relations for each digit:

$$\begin{aligned}
 1 \times 0 &\equiv 0 \pmod{2} \\
 1 \times 2 &\equiv 2 \pmod{3} \\
 0 \times 2 &\equiv 0 \pmod{5} \\
 3 \times 3 &\equiv 2 \pmod{7}.
 \end{aligned}$$

An overflow resulting from a multiplication is no different from the overflow resulting from an addition. Consider the product obtained from the residue multiplication of the numbers 10 and 100. The result in the modulo 210 number system is 160, since

$$1000 \equiv 160 \pmod{210}.$$

SUBTRACTION AND THE REPRESENTATION OF NEGATIVE NUMBERS

The process of subtraction is obtainable in the residue number system by employing a complement representation consisting of the additive inverses of the positive residue representation. The additive inverse always exists, since each of the elements of the residue representation is an element of a field. There is no basic problem associated with the subtraction operation. There is, however, a problem associated with the representation of negative numbers. In particular, some mechanism must be included in the number system which will permit the representation of positive and negative numbers. This problem is discussed here and in the following section.

The additive inverse of a residue number is defined by the following:

$$a \oplus a' = 0.$$

The formula may be considered to apply to a digit of the residue system or equally well to the whole residue representation. Consider the following examples with reference to the modulo 210 residue number system:

$$a = (1 \ 2 \ 4 \ 1)$$

then

$$a' = (1 \ 1 \ 1 \ 6),$$

since

$$\begin{array}{r}
 (1 \ 2 \ 4 \ 1) \\
 \oplus (1 \ 1 \ 1 \ 6) \\
 \hline
 (0 \ 0 \ 0 \ 0)
 \end{array}$$

The following examples have been chosen to illustrate the subtraction process and to some extent the difficulties associated with the sign of the difference:

$$D = A \ominus B = A \oplus B'.$$

We consider first the case where the magnitude of A is greater than B .

$$\text{Let } A = 200 \quad B = 100.$$

In residue representation,

$$B' = (0 \ 2 \ 0 \ 5)$$

and

$$\begin{array}{r}
 (0 \ 2 \ 0 \ 4) \\
 \oplus (0 \ 2 \ 0 \ 5) \\
 \hline
 (0 \ 1 \ 0 \ 2)
 \end{array}$$

The residue representation of the difference corresponds to positive 100 in the real number domain. We consider next the case where the magnitude of B is greater than the magnitude of A .

$$A' = (0 \ 1 \ 0 \ 3)$$

then

$$D = A' \oplus B$$

and

$$\begin{array}{r}
 (0 \ 1 \ 0 \ 3) \\
 \oplus (0 \ 1 \ 0 \ 2) \\
 \hline
 (0 \ 2 \ 0 \ 5)
 \end{array}$$

The difference (0 2 0 5) is the additive inverse of (0 1 0 2). Unless additional information is supplied, the correct interpretation of the representation (0 2 0 5) is in doubt. (0 2 0 5) may correspond to either +110 or -100.

The difficulties associated with whether a residue representation corresponds to a positive or negative integer can be partially removed by the division of the residue number range into two parts. This is exactly the scheme that is employed to obtain a machine representation of positive and negative natural numbers. For the system of natural numbers, two different machine representations of the negative numbers may be obtained and are commonly designated the radix complement representation of negative numbers and the diminished radix complement representation of negative numbers.

The complement representation for a residue code is defined in terms of the additive inverse. Thus, the representation of negative A is A' where $A \oplus A' = 0$, and the range of A is restricted to approximately one half of the total possible range of the residue representation. This can be illustrated by consideration of a specific residue code. This residue representation employing bases of magnitude 2, 3, 5, and 7, is divided into two parts. The residue representations corresponding

to the natural numbers 0 to 104 are considered positive. The residue representations corresponding to the natural numbers 105 to 209 are considered inverse representations and associated with the negative integers from -1 to -105 . The range of this particular number system is from -105 to $+104$. The arithmetic rules pertaining to sign and overflow conventions for this particular number system are the same rules normally associated with radix complement arithmetic.

The complement representation does eliminate in principle any ambiguity concerning the sign of the result of an arithmetic operation. However, there is a practical difficulty. The determination of the sign associated with a particular residue representation requires the establishment of the magnitude of the representation relative to the magnitude which separates the positive and negative representations. The determination of relative magnitude for a residue representation is discussed in the next section, where it is shown that the determination of relative magnitude is not a simple problem.

CONVERSION FROM A RESIDUE CODE TO A NORMAL NUMBER REPRESENTATION

It is frequently desirable to determine the natural number associated with a particular residue representation. The need for this conversion occurs frequently in investigation of the properties of the residue system. The residue representation is constructed in such a manner that magnitude is not readily obtainable. The presence of digit weights in the normal polynomial type number representation greatly facilitates the determination of magnitude. However, it is possible to assign a weight to each digit of the residue representation in such a manner that the modulo m sum of the digit-weight products is the real natural number in a consistently weighted representation. m is the product of all the bases employed in the residue representation. The conversion technique is known as the "Chinese Remainder Theorem." The material which follows describes the conversion technique but omits the proof. A simple and straightforward proof is found in Dickson.³ The proof does not refer specifically to residue number systems, but rather to a system of linear congruences. If so regarded, a system of congruences defines a component of a residue number system.

Consider a residue number system with bases $m \cdots m_t$. The corresponding digits are labeled $a_1 \cdots a_t$. The following equations define the conversion process:

$$a_1 A_1 \frac{m}{m_1} + \cdots + a_t A_t \frac{m}{m_t} \equiv S \pmod{m}$$

where

$$A_i \frac{m}{m_i} \equiv 1 \pmod{m_i}$$

and

$$m = \prod_{j=1}^t m_j.$$

The conversion formula for a particular residue number system is now obtained.

$$m_1 = 2 \quad m_2 = 3 \quad m_3 = 5 \quad m_4 = 7$$

$$105 A_1 \equiv 1 \pmod{2} \quad \text{so } A_1 = 1$$

$$70 A_2 \equiv 1 \pmod{3} \quad \text{so } A_2 = 1$$

$$42 A_3 \equiv 1 \pmod{5}$$

$$2 A_3 \equiv 1 \pmod{5} \quad \text{so } A_3 = 3$$

$$30 A_4 \equiv 1 \pmod{7}$$

$$2 A_4 \equiv 1 \pmod{7} \quad \text{so } A_4 = 4$$

$$105 a_1 + 70 a_2 + 126 a_3 + 120 a_4 \equiv S \pmod{210}.$$

The conversion formula is now used to determine the natural number corresponding to the residue representation (1 2 0 4).

$$105 (1) + 70 (2) + 126 (0) + 120 (4) = 725$$

$$725 \equiv S \pmod{210}$$

$$S = 95.$$

The conversion process described above requires conventional multiplication and modulo addition.

Other conversion techniques exist. In particular it is possible by means of a deductive process to determine the magnitude of a particular residue representation. This requires both a knowledge of the nature of the residue system and the natural number representation associated with at least one residue representation.

Due to the deductive nature of the process, it is more suitable for human computation than for machine computation. The process is explained using the residue number of the previous example (1 2 0 4). The knowledge of the residue representation for unity which is (1 1 1 1) is assumed. Consider the effect of changing the second digit from one to two. The change adds the product $m_1 m_3 m_4 = 70$ to the number, since 70 is congruent 1, modulo 3. The resulting residue representation (1 2 1 1) corresponds to 71. The effect of changing the third digit is to change the magnitude by some multiple of the product $m_1 m_2 m_4 = 42$. The correct change in magnitude is $42x$ where $42x \equiv 4 \pmod{5}$; so $42x = 84$ and the residue representation (1 2 0 1) corresponds to 155. The fourth digit is modified by the addition of a three. The effect of this change is determined by $30x \equiv 3 \pmod{7}$. The magnitude change is 150. The sum of 150 and 155 modulo 210 yields the correct result 95, in correspondence with (1 2 0 4).

³L. E. Dickson, "Modern Elementary Theory of Numbers," University of Chicago Press, Chicago, Ill., p. 16; 1939.

Sign determination for the residue code is dependent on the determination of a greater than or less than relationship. A possible method might involve the conversion techniques described previously. Such a scheme would involve the standard comparison techniques associated with the determination of the relative magnitude of two numbers represented in a weighted representation. An alternate conversion procedure yields a conversion from the residue code to a nonconsistently based polynomial number representation by means of residue arithmetic. Consider a residue code consisting of t digits. The t digits of the residue code are associated with t congruence relationships as follows:

$$S \equiv a_i \pmod{m_i} \quad 1 \leq i \leq t.$$

S is the magnitude of the number expressed in normal representation. It is also possible to express the number S as

$$S = a_i + A_i m_i.$$

A_i is the integer part of the quotient of S divided by m_i . In regard to a greater or less than relationship, the determination of A_i divides the range of the residue representation into m/m_i parts. We proceed to calculate A_i from the set of t equations given above. Let

$$S = a_t + A_t m_t \quad A_t < \frac{m}{m_t}.$$

This equation is then used to replace S in the remaining $t-1$ equations, yielding $t-1$ equations of the form

$$A_t m_t \equiv (a_i + a_i') \pmod{m_i} \quad 1 \leq i \leq t-1$$

or

$$\begin{aligned} A_t &\equiv (a_i + a_i')/m_i^i \pmod{m_i} \\ A_t &\equiv d_i^i \pmod{m_i} \end{aligned}$$

where $/m_i^i$ is the multiplicative inverse of m_i with respect to base m_i . The multiplicative inverse is defined as⁴

$$x_i/x_i^i \equiv 1 \pmod{m_i}.$$

d_i^i is the least positive residue of $(a_i + a_i')/m_i$ with respect to base i .

a_i' is the additive inverse of a_i .

Let A_t be expressed as

$$A_t = d_i^{t-1} + A_{t-1} m_{t-1} \quad A_{t-1} < \frac{m}{m_t m_{t-1}}.$$

If this expression is substituted for A_t a set of $t-2$ equations remain. The equations are of the form

$$\begin{aligned} A_{t-1} &\equiv [d_i^i + (d_i^{t-1})']/m_{t-1} \pmod{m_i} \quad 1 \leq i \leq t-2 \\ A_{t-1} &\equiv d_i^i \pmod{m_i}. \end{aligned}$$

⁴ The existence of the multiplicative inverse requires that x_i and m_i be relatively prime.

The system of equations shown below is generated by repetition of the above substitution process until no equations remain.

$$\begin{aligned} S &= a_t + A_t m_t \\ A_t &= d_i^{t-1} + A_{t-1} m_{t-1} \\ A_{t-1} &= d_i^{t-2} + A_{t-2} m_{t-2} \\ &\vdots \\ A_3 &= d_3^2 + A_2 m_2 \\ A_2 &\equiv d_2^1 \pmod{m_1} \end{aligned}$$

The equations are combined to yield

$$\begin{aligned} S &= a_t + m_t \{ d_i^{t-1} + m_{t-1} [d_{t-1}^{t-2} + m_{t-2} (d_{t-2}^{t-3} + \dots) \\ &= a_t + m_t d_i^{t-1} + m_t m_{t-1} d_{t-1}^{t-2} + m_t m_{t-1} m_{t-2} d_{t-2}^{t-3} + \dots \\ &\quad + \frac{m}{m_1} d_2^1 \end{aligned}$$

where

$$\begin{aligned} A_t &< m_t \\ d_{t-n}^{t-n-1} &< m_{t-n-1}. \end{aligned}$$

Therefore, S is never equal to or greater than m and d_2^1 divides the range into m_1 parts, d_3^2 divides each of the m_1 parts into m_2 parts, d_4^3 divides each of the m_2 parts into m_3 parts, etc.

The determination of the less than or greater than relationship consists of the successive comparison of the d_{t-n}^{t-n-1} constants corresponding to two residue representations. Let the representations be designated E and F . The first step of the greater than or less than determination is the comparison of $d_2^1(E)$ and $d_2^1(F)$. If the two constants are different the process may be terminated and the larger number is associated with the larger constant. If the constants are equal in value, the comparison process must consider the pair of constants $d_3^2(E)$ and $d_3^2(F)$. The process is continued in this manner until a set of nonidentical constants is found. If all of the d constants are identical, a final comparison is made on the basis of the pair of t th digits of the two residue representations.

The formulas which define the greater than, less than process may be applied recursively to obtain a formula for a greater number of digits. The process has been extended to five variables and the results are shown in Fig. 1.

Admittedly, the process required to obtain a greater than or less than relationship leaves much to be desired. One presumed advantage of the residue number system was the absence of a carry process. The greater or less than process is essentially sequential and is in many ways similar to the carry process of ordinary arithmetic. The ultimate usefulness of the residue code for general-purpose computation appears very much dependent on the development of simple techniques for the determina-

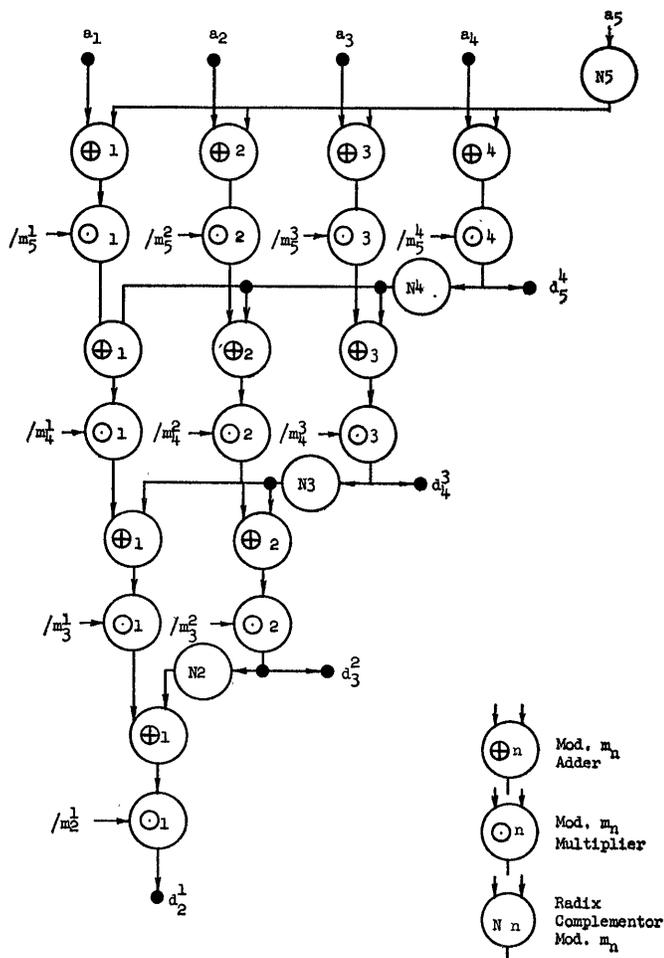


Fig. 1—Logic for the determination of the greater or less than relationship.

tion of the relative magnitude of two residue code digits.

DIVISION

The division process for residue codes is complicated by two factors. The first is the absence of a multiplicative inverse for the zero element. The second difficulty is the fact that residue division and the normal division process are in one to one correspondence only when the resulting quotient is an integer value. We shall consider first the problem of residue division of the elements of a single field and shall consider later the elements of several fields considered as a residue code. The division process represented in equation form as

$$\frac{a}{b} = q$$

implies

$$a = bq.$$

The difference between normal arithmetic and residue arithmetic is that in residue arithmetic the product bq need not necessarily be equal to a ; only the congruence of a and bq is required.

$$bq \equiv a \pmod{m_n}$$

Multiplication by the multiplicative inverse of b designated $/b$ obtains

$$q \equiv a/b \pmod{m_n}.$$

The correct interpretation of q in the above equation is that the number a is obtained by forming the modulo sum consisting of b, q representations. The sum is carried out in a closed and finite modulo number system of base m_n . Thus, q corresponds to the quotient only when the quotient has an integer value. Examples may be obtained from the consideration of a modulo 5 number system:

$$\frac{4}{2} = q$$

$$2q \equiv 4 \pmod{5}$$

$$q \equiv 2 \pmod{5}$$

$$\frac{4}{3} = q$$

$$3q \equiv 4 \pmod{5}$$

$$q \equiv 3 \pmod{5}$$

note $3 \times 3 \equiv 4 \pmod{5}$

$$\frac{3}{4} = q$$

$$4q \equiv 3 \pmod{5}.$$

$$q \equiv 2 \pmod{5}.$$

In the above examples, q corresponds to the quotient only in the first example.

The residue code representation of a number consists of many digits, $A = (a_1, a_2, \dots, a_n)$. Each digit of the representation is associated with a different prime base. The number system is a modulo m system where

$$m = \prod_{i=1}^n m_i.$$

The division of two numbers in residue code may be expressed by a system of congruences. The solution $Q = (q_1, q_2, \dots, q_n)$ must satisfy all the congruence relationships of the system. A zero digit in the divisor $B = (b_1, b_2, \dots, b_n)$ means that B and m are not relatively prime hence the multiplicative inverse of B does not exist.

$$QB \not\equiv A \pmod{m}.$$

For the special case in which $b_i = 0$ and $a_i = 0$, a valid congruence relationship of the form

$$\frac{QB}{m_i} \equiv \frac{A}{m_i} \pmod{\frac{m}{m_i}}$$

is obtainable.

The process of residue division has certain interesting properties and quite possibly has applications in respect to special problems. Unfortunately, the residue division process is not a substitute for normal division. It appears that the only way in which division can be effected in the residue code is by the utilization of techniques similar to those employed for division in a consistently weighted number system. The division process then requires trial and error subtraction or addition and the greater than or less than relationship. The division algorithm could also include trial multiplication, since in the residue system addition and multiplication require the same period of time.

CONCLUSIONS

The material of this paper forms a preliminary investigation of the applicability of residue number systems to the arithmetic operations of digital computers. The residue system has been found attractive in terms of the operations of multiplication and addition. It is possible to realize practical logical circuitry to yield the product in the same operation time as for the sum, since the product is not obtained by the usual procedure of repetitive addition. The main disadvantages of the residue number system are associated with the necessity of determining absolute magnitude. Thus, the division process, the detection of an overflow, and the determination of the correct sign of a subtraction operation are processes which at this stage of the investigation seem to involve considerable complexity. Nevertheless, many

special-purpose applications are certainly well-suited to the residue code. In particular, there exists a class of control problems characterized by the absence of the need for division and the existence of a well-defined range for the variables, and also by the fact that the sign of the variables is known. For the problems of this class, the use of the residue code should result in a reduction of the over-all computation period and should yield a computer with a higher bandwidth than obtainable with the conventional number system.

The ultimate usefulness of the residue code will probably be determined largely by the success of the circuit designer in perfecting circuitry ideally suited for residue code operations.

The material of this paper is essentially Chapter 5 of the author's doctoral dissertation.⁵ At the time of the completion of the dissertation, the author was unaware of the work of M. Valach⁶ and A. Svoboda^{7,8} in Czechoslovakia. Additional literature⁶⁻⁸ was obtained from recent visitors to the Soviet Union. The author wishes to take this opportunity to acknowledge the work of Valach and Svoboda.

⁵ H. L. Garner, "Error Checking and the Structure of Binary Addition," Ph.D. dissertation, University of Michigan, Ann Arbor, pp. 105-140; 1958.

⁶ M. Valach, "Vznik kodu a ciselne soustavy zbytkovych trid," *Stroje Na Zpracovani Informaci*, Sbornik III; 1955.

⁷ A. Svoboda and M. Valach, "Operatorove obvody," *Stroje Na Zpracovani Informaci*, Sbornik III; 1955.

⁸ A. Svoboda, "Rational numerical system of residual classes," *Stroje Na Zpracovani Informaci*, Sbornik V; 1957.

System Evaluation and Instrumentation for Military Special-Purpose Digital Computer Systems

A. J. STRASSMAN† AND L. H. KURKJIAN†

INTRODUCTION

TESTING and instrumentation are essential prerequisites for the completion and operation of any new system. A system can be defined as a number of components that are amalgamated or integrated together to perform a desired operation. Throughout this paper a "component" is considered to be a complete functional part of a data processing system such as an arithmetic unit or a buffer. To ascertain if a component in the system is going to perform correctly its specific function, it is sometimes necessary for the implementa-

tion of tests to be more complex than the component undergoing the testing. This becomes apparent when the component is a part of a large system and has many inputs and outputs.

To prove the system feasibility or operation of the components it is necessary to do either of two things: 1) duplicate and maintain an entire system and use it as one master test fixture to evaluate each functional component; or 2) provide individual test facilities for the evaluation of each of the functional components. The second approach requires the design of simulation equipment to provide the necessary inputs (control signals and data) to check out completely the operation of each

† Hughes Aircraft Co., Fullerton, Calif.

individual component. It is believed that this approach offers the greatest advantages for large special-purpose digital computer systems.

SYSTEM ORGANIZATION DETERMINES WORK ORGANIZATION

It is necessary to provide the proper work organization for the evaluation of these computer systems. A differentiation can be made between small and large systems and the work organization can be adjusted accordingly. Although the basic philosophy of test remains the same, the details evolved for the testing or evaluation of a small system will be different from that evolved for a large system. Since the flip-flop is a basic element in many digital computers, the number of flip-flops can be used as an indication of the size and complexity of the system. For the purpose of this paper, in which the evaluation and instrumentation of a large system is described, a "large" system is defined arbitrarily as one that contains more than 500 flip-flops.

In the case of a small system, all the work can be handled by a small group which will perform the necessary tasks from system design to final evaluation. Fig. 1(a) shows the work flow for the "small system" organization. The work usually begins with a proposal outlining the new system. This is followed by system design, logical design, circuit design, testing, evaluation, and delivery. It is admitted that to be able to perform all the tasks included, the technical personnel associated with a small system must have a broader background than those required for the evaluation of a large system.

The actual limitations of time, complexity of large systems, and efficiency of utilization of personnel, necessitates the use of specialists in each specific work area to perform all the necessary tasks to complete the large system. Specialization is indicated by the fact that the system design is done by a group of systems engineers whose function is to define the necessary components needed to implement the system and their interrelationships. Logical design and circuit design are two functions that are performed in an analogous manner by logicians and circuit engineers who are specialists in their respective realms. The test and evaluation of the system is also handled in a specialized manner. Each component is assigned to a circuit or unit engineer whose responsibility is to 1) design the logical circuitry of the component from the Boolean equations, 2) design the test fixture, 3) write the necessary procedurals, and 4) test the component when it is fabricated.

Since a large system requires many individuals to complete these tasks, the question of communication between groups of specialists becomes a problem. This is an important item especially when the decision of one group vitally affects the work of another group or groups. To maintain a continuous flow of data in both directions which enables each group to perform their tasks more efficiently, information feedback loops are provided in the form of signature control and written

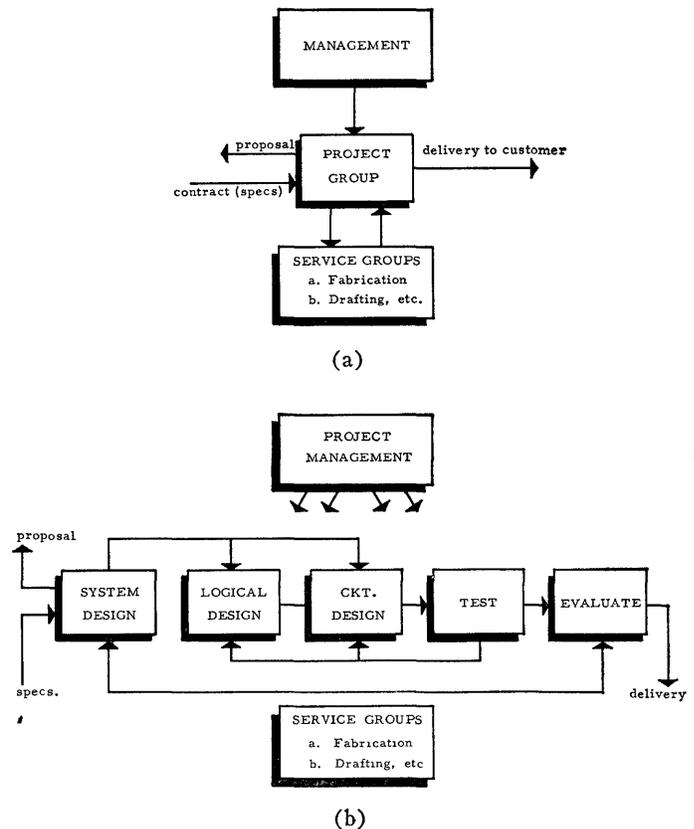


Fig. 1—Work organization for typical small (a) and large (b) systems.

reports. Fig. 1(b) shows the interrelationship of all groups from system design to delivery.

The philosophies contained within this paper led to the basic planning considerations for the test and evaluation of a large special-purpose military data processing system, parts of which will be described in later paragraphs. This data processing system contains approximately 1500 tubes, 2500 transistors, 250,000 diodes, and 3500 flip-flops. Each flip-flop in the system has four transistors, making a total of 16,500 transistors in the entire system. This qualifies the described system to be classified as a large system.

TYPES OF TESTS TO BE PERFORMED FOR EVALUATION

The basic parts of any large system can be broken down into five categories which are listed in their order of complexity: 1) elements, 2) units, 3) components, 4) subsystems, and 5) systems. If these basic parts of the system are evaluated and tested in order of complexity, the sequential building of testing integrity gives us the understandable advantage of solving small problems first before becoming involved with the intricacies and troubles inherent in any large system. The first type of tests to be performed therefore would be element tests.

Element Tests

The basic computing elements are usually flip-flops, logical followers, drivers, shift registers, diode cards,

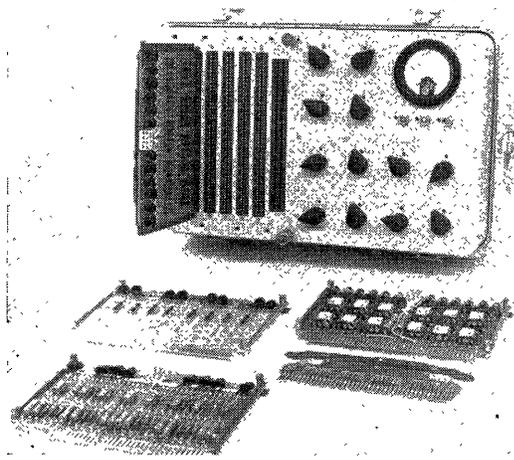


Fig. 2—Standard elements and standard element test fixture.

and pulse regenerators, etc. These items are referred to as “standard elements.” The functional requirements of each of these standard elements determine the design of the test fixture necessary for evaluation. The test fixture for a flip-flop contains the necessary steering circuitry which makes the flip-flop a modulo 2 counter. When more than one flip-flop is built on a standard card, the fixture can be expanded to make the flip-flops count in any prescribed manner. Shift registers, followers, and drivers most commonly are evaluated by inserting specific computer word patterns at the input and observing the appropriate outputs. Simple sequential relay circuits are used to step through the forward and reverse characteristics of diodes on standard diode cards. Typical standard elements of a digital computer system along with a standard element test fixture are shown in Fig. 2.

The electronic implementation of the digital computer logic that is represented in Boolean notation is formed from the standard diode card and specific resistor networks on the matrix card assembly. The wiring of the resistors to the diodes on each individual matrix card determines its logical function. The logical function of each card can be statically evaluated by a “matrix card tester” which simulates each input term to the card. The output of each gate is monitored as logical true and false levels are placed at the inputs to the gate. A meter is used to indicate to the operator the result of the simulation of any of the logical terms under test. Fig. 3 is a photograph of a matrix assembly and a matrix card tester.

Unit Tests

Each module of the system under discussion is called a unit and contains up to twenty-one element cards. A unit module is shown in Fig. 4. The combinations of, and connections between, the elements in the unit provide a portion of an over-all computing function that is to be performed by the system. Evaluation of units is difficult because units are incomplete functional items

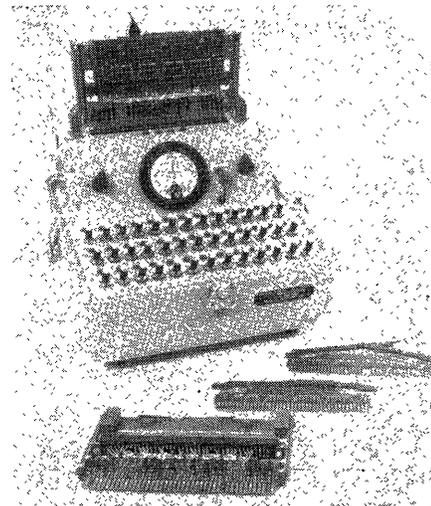


Fig. 3—Matrix assembly and matrix tester.

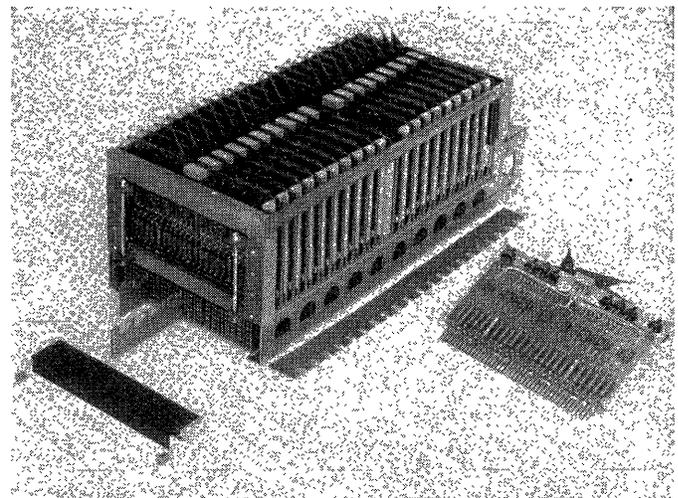


Fig. 4—Unit module.

and therefore the amount of simulation becomes large and complex. However, it is considered that this step in the system evaluation is critical. It is therefore necessary to ascertain that each unit has been tested to the maximum. This obligates us to perform the most exhaustive tests possible on the unit level within the framework of the computer. Provisions to accomplish this can only be done by generating ideal simulated signals that the unit would expect in system operation. This type of simulation has been achieved by the design of equipment referred to as the “unit tester.” The unit tester provides combinations of static and dynamic signals to the unit under evaluation. All system timing signals, synchronizing signals, and data inputs are generated in the unit tester. Each connection in and out of the unit under test is available on a patch panel on the unit tester. The choice of four signals is available at each point. The point may be 1) connected to a logical true signal, 2) connected to a logical false signal, 3) connected to a special function (sync, timing, data, etc.), or 4) unconnected if it is an output of the unit that is to

be observed. The unit test insures that the interelement wiring and the input-output wiring of the unit is correct and at the same time provides a semidynamic test to the various element configurations. Many of the logical functions can be completely evaluated during this phase of test. The unit test can be easily modified for production testing of each module by simple automation techniques. In Fig. 5 a unit is shown under test with the unit tester.

Component Tests

A system component is a unit or group of units that has been defined in the system to perform a particular computing or data processing function. Examples of typical components are the arithmetic unit, the various buffers, the computer controls, and the buffer controls. It is at this level that the simulation of external signals is very important, as the completeness of testing at the component level determines the ease with which it is possible to test and evaluate the entire system. The component test provides for the testing of all of the logic contained within the integrated units by means of external simulated signals. These external signals are developed by a special test fixture that is unique for each component. A component consisting of eight units mounted on its test fixture is illustrated in Fig. 6. The test fixture is designed to simulate the complete system to the component. This test is basically dynamic and as a consequence logical errors can be discovered during this phase of evaluation. The simulation equipment consists of the appropriate switches, function generators, and timing and synchronizing signals that the component would operate from if it were in the system. Procedurals are written which outline the detailed steps necessary to evaluate the component function as specified by the system design. This test actually proves or disproves the component logic with the test fixture as the system simulator. Both the test fixture and the procedurals are designed and written by the cognizant circuit or unit engineer who is charged with the responsibility of this component and has by necessity a complete grasp of the functional operation of this component. Typical examples of system components and an idea of their complexity are given in the following paragraphs.

1) The coordinate extrapolator updates coordinates on the basis of velocity stored in the memory. This component requires twelve flip-flops, eight logical followers, and 180 diodes. The logic written in Boolean notation consisted of three typewritten pages and the test procedural was nine pages long. The control and addition logic in this component were evaluated by means of a component test fixture which simulated the system input coordinate data, velocity, and time by means of variable word generators and counters.

2) The computer control is a special-purpose wired program computer that controls information from and to three arithmetic units. Its outputs include control

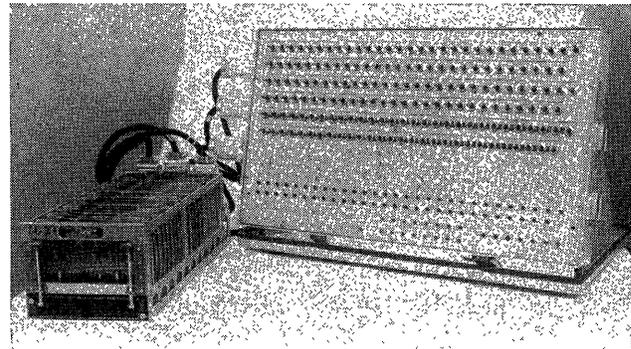


Fig. 5—Unit tester.

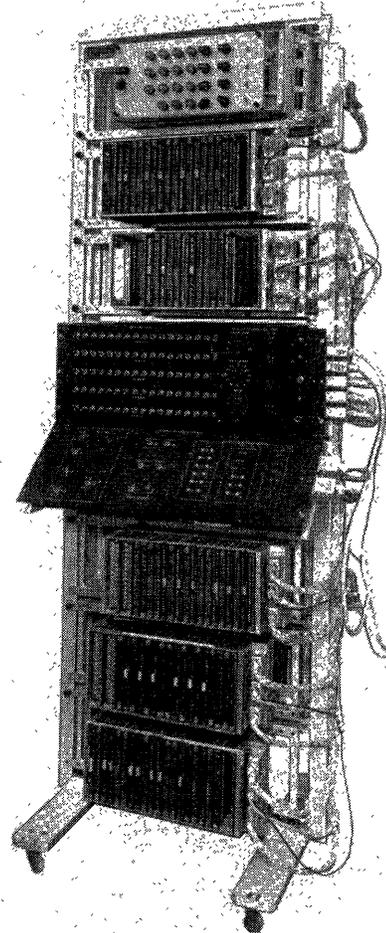


Fig. 6—Component test fixture.

signals and generation of appropriate constants needed during the various steps in the wired program. The component was implemented with 15 units which contained 148 flip-flops, 384 logical followers, and 6350 gating diodes. The logical equations in Boolean notation comprised 26 typewritten pages, and the test procedural was 114 pages long. The control signals and the terms of the constant generators were checked by a test fixture which simulated the essential control signals required to cause the computer to perform each program step.

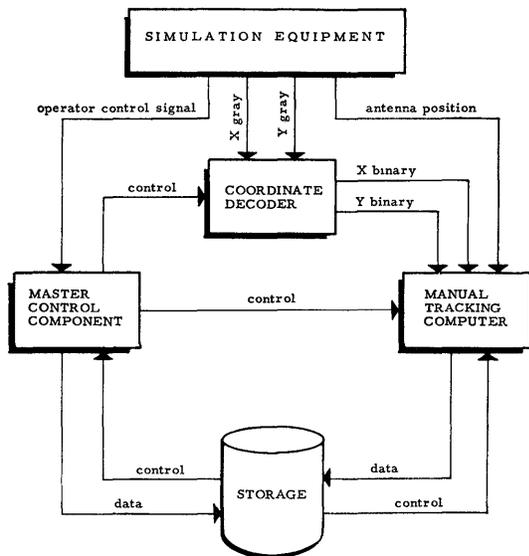


Fig. 7—Typical subsystem block diagram.

Subsystem Test

After the component has been completely evaluated, the next step for system completion is to integrate the components together into the various subsystems as determined by a logical sequential build-up. Fig. 7 demonstrates an integration of one subsystem consisting of four components. Simulation equipment needed in this phase is less than during component test. The example shows control and decoder components that have the facility for entering data into a special-purpose computer which steps through a wired program cycle and stores information of a magnetic drum. Parts of this information are used in the control component during system operation. This makes the sub-system a small closed loop within the system. Logical tie-in and timing errors can be found and solved during this part of system completion. Simulation equipment for subsystem test usually consists of inhibiting signals that affect the closed loop operation and generate all those other signals which are necessary to make the loop operate. In the example shown, X and Y coordinate data in Gray Code, simple operator control buttons, and radar antenna position signals were the only signals needed to be simulated. Parts of existing component test fixtures can be used during subsystem test as they contain the necessary simulation equipment.

System Test and Evaluation

This phase is the culmination of all the test and evaluation effort that has been performed previously. All the elements, units, and components have been proved to perform within the framework of the several subsystems and now it is necessary to prove complete system operation. This is done in two phases. Since the final military installation is a vehicle that has limited working space, a laboratory mock-up is provided. This mock-up as shown in Fig. 8 simulates the trailer installation as

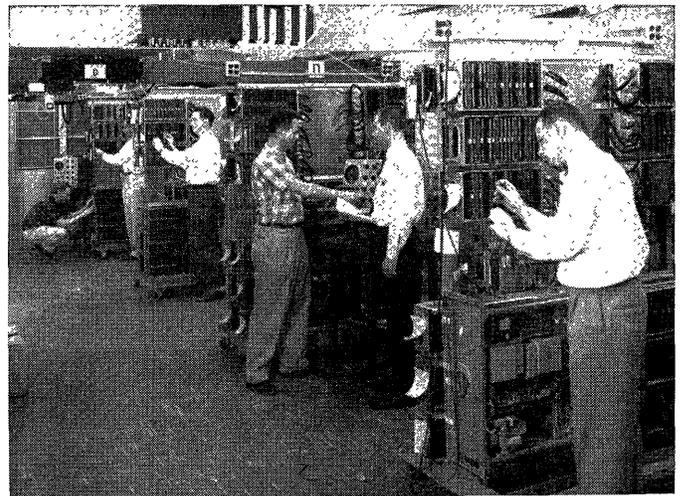


Fig. 8—System mock-up fixtures.

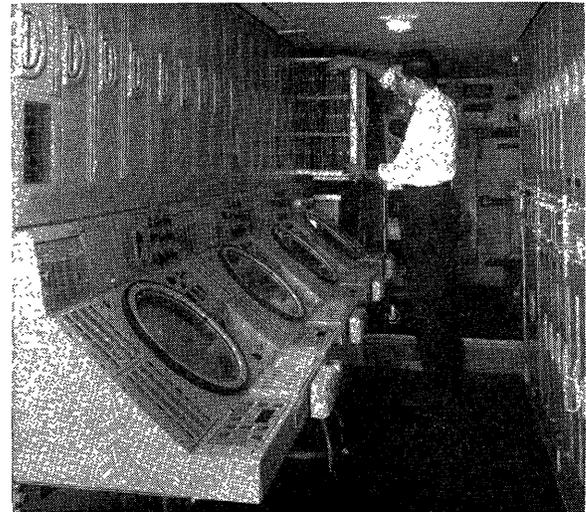


Fig. 9—Final system installation.

closely as possible, yet provides ample working space for many more of the engineering personnel so that much system testing and trouble shooting can be carried on simultaneously in many areas of the system. An additional advantage gained by this 2-step operation is provided by the ability to modify the final wiring installation as required as problems are encountered in the mock-up test phase. All system errors will be discovered in the mock-up phase and corrections can be made to the equipment before installation into the vehicle. Upon completion of the tests in the mock-up area, the equipment is then transferred to the vehicle and the complete system can be integrated with a minimum of personnel due to the fact that the system has been completely tested and all system errors removed. In fact, the only difficulties to be encountered are wiring errors caused by human inefficiencies. The vehicle interior working area is shown in Fig. 9.

In the process of the design of the laboratory test equipment, many of the simulation concepts evolved are readily useful and on occasion can be incorporated

into the system as self-test features. This equipment can be utilized in the initial system evaluation as well as later during normal test modes of system operation. Since the end user of this equipment will be military personnel, many self-test and automatic indicator devices were incorporated to decrease the training requirements for operation and maintenance of this equipment. Another requirement that is very often specified for military equipment is that of providing operation for 23 out of 24 hours. This fact dictates the requirement for having a very rapid means of performing operational and preventive maintenance checks by semiskilled personnel.

Military requirements include a controlled complete system test to prove that the system meets the initial specification. A comprehensive system test plan is most often written by the system engineers to test all the functions of the system. During this test only external system inputs must be simulated; following the test, the system is ready for operational use and field evaluation.

SYSTEM TEST PERSONNEL TRAINING

The previously mentioned steps in providing for sequential testing of all components up to the complete integration for system test and evaluation allow certain personnel to acquire gradually system knowledge necessary to perform efficiently and rapidly the complex task of testing such a large system. It is obvious that no one individual, no matter how magnificently endowed with mental powers, can be expected to understand all necessary details of such a complex system containing equipments involving such diverse fields as displays, conversion, and data processing. A plan was evolved for certain specialists to become facile in the over-all system concepts, yet utilize the certain portions of their specialty to a large extent as possible. This control is achieved in the following manner: in the initial design phases, each component or allied group of components is assigned to a cognizant circuit engineer whose responsibility during this phase is to design the logical circuitry for the component, or in the case of the display subsystem, to implement the original specifications. Once these data have been released for equipping and packaging, this same engineer proceeds to design and build the necessary unique test equipment for component evaluation. As part of this task the engineer also writes the procedurals to be followed in the testing of the component. This is the first step in causing the engineer to investigate external requirements of the component assigned to him. As the component is integrated into a subsystem area, it is necessary for the engineer to become more familiar with the input-output requirement of the adjacent components, so that, while he remains a specialist

for his assigned component, his system knowledge must perforce increase because of interdependency of the components in the subsystem. Since all components have been completely tested, there need be only one engineer now assigned to each subsystem. The remaining component engineers, however, are available for consulting as needed. When all the components are finally integrated as a system, there remain but a few engineers necessary for systems testing, each with a broad knowledge, rather than many component engineers with limited specialized knowledge. Final installation can be completed more efficiently with a minimum of personnel.

CONCLUSIONS

It is apparent that any complex system can be tested and evaluated by a step-by-step instrumentation. Providing the necessary special-purpose instrumentation has proved to be more rapid and economical than the accumulation of general-purpose testing devices. In the testing of special-purpose computer components within a system, there are many instances in which general-purpose instrumentation devices would not suffice, no matter how much and how varied the instruments could be interconnected. In each of the stages of the system integration, particular classes of errors and failures can be uncovered. During element tests, electronic part failures and mechanical errors are discovered and corrected. After element testing, each element is considered operative and the troubles found in unit tests cannot be attributed to the elements. During unit testing, logical and timing design errors can be uncovered and intra-unit connections are ascertained to be correct. At the completion of the unit test, each unit is considered to be completely operative. Therefore, during the component test phase, any difficulties discovered cannot be attributed to the unit, but rather to logical tie-in errors between units and inter-unit wiring. Similarly, the problems within the subsystem test are related to only those difficulties encountered in integrating more than one component because of the completeness of the component evaluation. System testing is merely an extension of the previous statements, but now referring to problems encountered in integrating subsystems. The sequential building of test complexity gives us the advantage of solving small problems first before becoming involved with the intricacies and troubles inherent in any large system integration.

Finally, the experience of the personnel involved in the test build-up enables a better understanding of the system operation, thereby decreasing the time required to integrate a large system made up of many discrete and special components.

Automatic Failure Recovery in a Digital Data-Processing System

R. H. DOYLE,[†] R. A. MEYER,[†] AND R. P. PEDOWITZ[†]

INTRODUCTION

PERFECT reliability in digital computers has not yet been achieved by simply designing ruggedness into the equipment components. Nevertheless, it is essential for a computer to perform dependably under all conditions. In certain computer applications, errors resulting in unscheduled maintenance delays can be tolerated, but only at the cost of expensive computer time. In some special military and civil applications, such as the SAGE system and air-traffic control systems, poor equipment reliability can be disastrous since input information not processed when the system is inoperative can become obsolete during the time required for manual recovery.

Although it is virtually impossible to guarantee that failures will never occur, it is possible to maintain high over-all reliability of the system by immediately recovering from these failures with a negligible loss of operational time.

The FIX program was designed to effect automatic recovery from failures either by

- 1) reinitiating the operation that failed,
- 2) preventing the operational program from processing incorrect data, or
- 3) determining the effect that a particular failure would have on a word of information and then modifying the information to compensate for this failure.

The error-detection circuitry of the computer is relied upon to indicate the existence of an error in computer operations. When an error is detected by this equipment, the FIX program will be operated in an attempt to diagnose the failure and to compensate for it.

Although FIX was specifically designed to work with the Air Defense Program of the SAGE Computer, the technique employed may be modified for other operational or production systems.

Several other methods for maintaining system reliability have already been developed. Some of these methods will be briefly outlined in the preliminary section of this paper, followed by a detailed description of the structure and operation of the FIX program.

RELIABILITY TECHNIQUES

In a complex computer system, component quality standards are necessary but cannot in themselves insure

complete reliability. To approach the goal of high reliability, a more sophisticated viewpoint has been taken in designing both the equipment and the computer programs.

In the SAGE system, for example, the complete central computer has been duplexed, and the two computers alternately performed the operational program and a standby program on a 24-hour schedule. Special alarm circuits provide for alerting the standby computer when the active computer breaks down so that the standby machine will prepare to assume the active role. A portion of the standby-computer time is devoted to attempting to predict potential failure conditions before they occur. This technique, known as "marginal checking," consists in operating and testing various circuits while an abnormal voltage is supplied to them. In this simulated aging of the equipment, the potential failure spots are anticipated.

Modern computing equipment is usually designed with built-in circuitry¹ that will automatically detect the majority of errors that occur during system operation. Many operational programs are written to take advantage of this circuitry by including alarm-interrogation routines which will automatically repeat any operation that generated an alarm.

Error-checking routines have also been incorporated directly into operational programs.² In programs where it is necessary to store blocks of information on auxiliary storage drums or tapes before reusing it, the accuracy of the transferred information may be checked by comparing the arithmetic sum of the block before it is stored to a similar sum obtained after the block is brought back from storage. If the two check sums are not equal, the reliability of the information block cannot be depended upon and the program should be rerun. If the program is of considerable length, this task may be shortened by periodically saving the environment of the program as it operates. This will provide a convenient recovery point should it be necessary to regenerate a particular block of information.

Elaborate equipment and coding systems, such as the Hamming Code,³ can provide for automatic self-correction of errors and for detection of multiple errors. This

¹ C. J. Swift, "Machine features for a more automatic system for digital computers," *J. Assoc. Comp. Mach.*, vol. 4, p. 172; April, 1957.

² J. H. Brown, J. W. Carr, III, L. Boyd, and J. R. McReynolds, "Prevention of propagation of machine errors in long problems," *J. Assoc. Comp. Mach.*, vol. 3, p. 348; October, 1956.

³ R. W. Hamming, "Error detecting and error correcting codes," *Bell Sys. Tech. J.*, vol. 29, p. 60; 1950.

[†] IBM Corp., Kingston, N. Y.

is accomplished by dividing the information to be checked into groups of bits and by parity-checking each group. The groups of bits are chosen in a manner such that an error in any bit in the entire word will generate alarm indications for a unique combination of these groups. Conversely, incorrect parity counts for any combination of these groups will uniquely identify the erroneous bit in the word. Since the incorrect bit can be identified, circuitry can be provided to correct the error. This ingenious coding system achieves excellent results, but only at considerable expense. Channel capacity of the equipment must be increased to provide for enough checking bits to represent a number equal to the total number of information bits plus the checking bits.

Although FIX incorporates some of the techniques described above, the distinguishing feature of this program is that it achieves automatic failure recovery by means of programming techniques after an error has been detected by machine circuitry. While variations of the FIX concept will be necessary for other operational systems, depending upon the error-detection circuitry of the computer and upon the form of the operational program, this paper will serve to illustrate the general principles of the FIX technique.

Errors in a computer can occur either during the actual processing of data, such as sorting, collating, arithmetic computations, etc., or during the transfer of information between the central computer and the various auxiliary drum storage units. Since the Air Defense Program requires a large storage area, it is stored on auxiliary drums, and a considerable number of information transfers continually occur during normal operations as the various subprograms and their data tables are brought into core memory to be operated. It is extremely important that these transfers be performed correctly; hence a large portion of this paper discusses the technique of monitoring and correcting errors in such transfers.

Errors incurred during either central computer or transfer operations may be either transient or "solid" in nature. Errors which are due to high stresses of voltage, temperature, shock, etc., and which have a low probability of recurring, will be referred to as "transient errors." Those errors which are a result of a persistent equipment malfunction and which can continually be expected to reappear whenever the submarginal area of equipment is used, will be referred to as "solid errors." The FIX program has achieved a high degree of success in automatically recovering from most of the classes of errors described above.

PROGRAM DESIGN

Storage requirements for the present version of the FIX program are 50 core-memory registers and 5000 auxiliary-drum-storage registers. This represents approximately 3 per cent of the total storage available in the SAGE computer. During any alarm condition, the short FIX routine that is permanently stored in core

memory will save a portion of the operational program in order to provide a working area for the main section of FIX. This same routine will then read the appropriate diagnostic FIX routine into core memory.

Fig. 1 is a flow chart of the logical structure of the FIX program. This structure may be analyzed in terms of four functions:

- 1) Alarm monitoring and control
- 2) Diagnosis
- 3) Logging
- 4) Recovery.

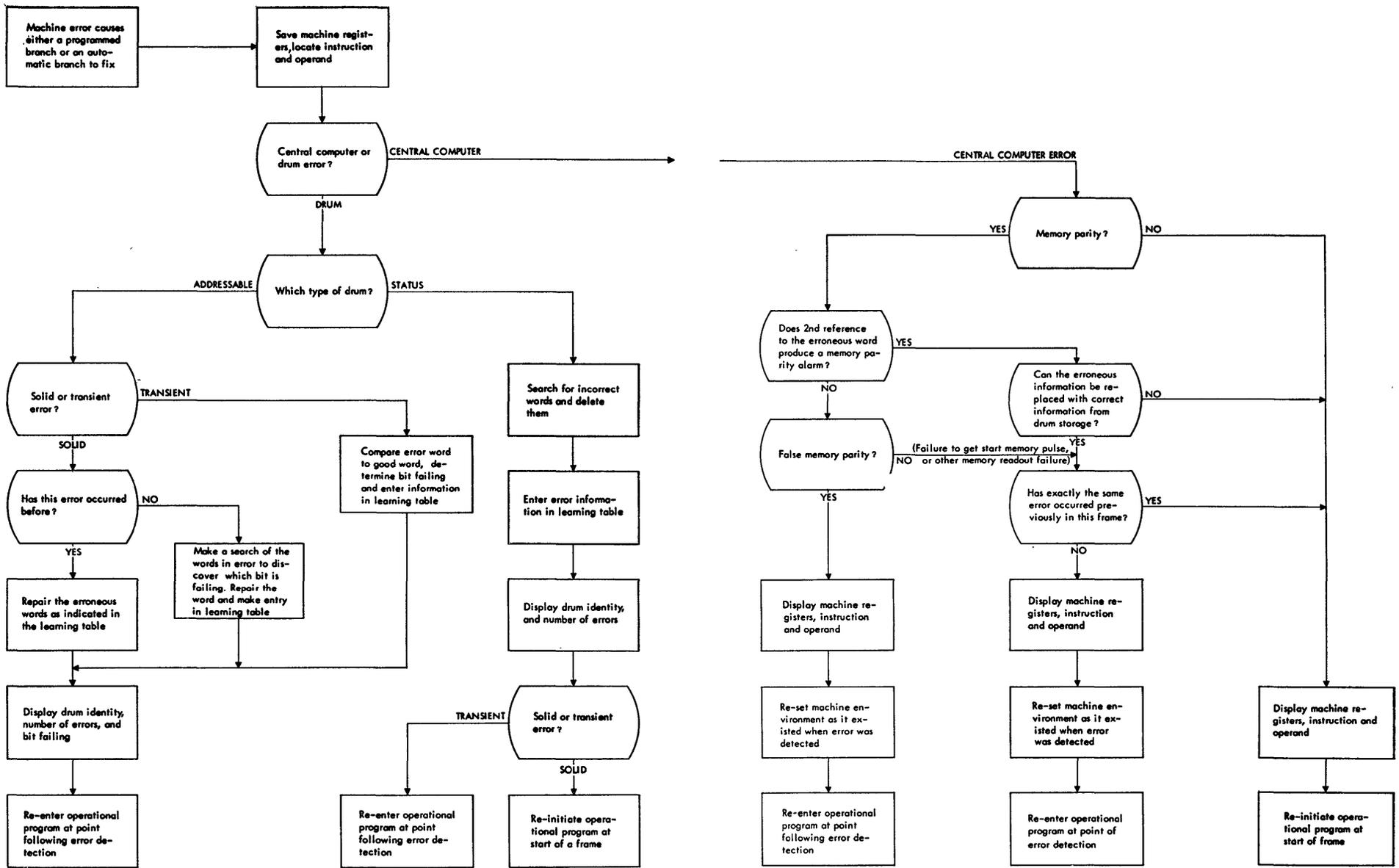
These functions are closely related and, although the above list represents the over-all time sequence of the operations to be performed, there will be considerable overlapping in the detailed structure. Since the design of the FIX program is a function of the makeup of the operational program and of the system to be monitored, some of the features of the SAGE system, including the error-detection equipment of the computer and the structure of the Air Defense Program, will be discussed during the analysis of the FIX program.

ALARM MONITORING AND CONTROL

The operation of the FIX program is greatly dependent upon the means by which FIX can be notified of an error occurring in the monitored system. In the SAGE computer, this is provided for by error-checking and alarm-control circuitry.

Self-checking is performed by the use of parity-code generation and checking circuits that determine if the correct number of bits in a binary word have been transferred from register to register during the normal data-processing operation. This is accomplished by increasing channel capacity to allow for one redundancy bit to be contained in the information transferred. As each instruction or data word is stored in the computer, it passes through a buffer register, which counts the number of "one" bits in the word. The parity bit associated with each word will be set to a "one" or a "zero" to give an odd number of "one" bits in the word, including the parity bit.

The parity bit will then be stored with the rest of the word. When this instruction or data word is referred to by the program, a parity-check count is again performed in the buffer register as the word is brought out of storage. If the total parity count is not still odd at this time, the word is presumed to be incorrect and a parity alarm will be generated. If no error is detected, the operation will continue and a new parity assignment will be performed prior to storing the word after it has been operated upon. The parity circuitry is used to check the correctness of all data transfers that occur in the system. It should be noted that a major shortcoming of the parity-checking system is that if two bits in the word are altered as the result of some failure, the odd parity count will not be disturbed and the error will not be detected by the parity circuitry. Such an error might remain unnoticed, in which case the final result would



(a)

(b)

Fig. 1—Flow chart of the logical structure of the FIX program.

be incorrect, or it might result in other error indications which could be detected.

Automatic detection of other abnormal conditions in the SAGE computer is also provided by circuitry. Sometimes because of circuit failure or an undetected parity error, or because of a peculiar set of environmental circumstances unanticipated by the program designer, the computer can begin a nonterminating cycle of meaningless operations, commonly referred to as an "illegal loop." Similarly, the computer might begin an inactive period during which it does nothing but wait for some anticipated event. If for some reason the event can never occur, the computer will remain in this inactive condition indefinitely. Special circuitry designed to impose time limits on such conditions can, upon sensing an illegal delay in computer operations, terminate the condition and by means of an inactivity alarm indicate to the computer that the delay existed. The inactivity alarm will be activated if a pulse is not generated by the program at regular intervals or if too many of these pulses are generated within a given time period, usually about eight seconds. The programmer must, therefore, insert the pulse-generating instruction at regular intervals throughout his program if he intends to use this circuitry. If the program operates normally, the pulses will be generated at regular intervals. If the program is "illegally" delayed in a routine, or if it continuously loops through a few instructions, either too few or too many signals will be generated, and the inactivity alarm activated.

Finally, in certain instances an error in a series of arithmetic operations may result in the attempted development of a sum or quotient which has increased in size beyond the physical limits imposed by the register capacity of the computer. This condition, too, can be sensed by machine circuitry in the SAGE computer and indicated by means of an overflow alarm.

The programmer can choose various modes of operation by using switch settings when planning the reactions of the SAGE computer to these alarm conditions. These options can be set to have the computer automatically

- 1) stop on alarms,
- 2) branch on alarms, or
- 3) continue on alarms.

Under option 3 the program retains the ability to interrogate the alarms at some convenient time before taking any automatic action.

The mode of operation used by the FIX program was determined by the nature of the errors that would be encountered. Certain types of computer malfunctions demand immediate transfer of control to the FIX program. For example, if there is a parity alarm when the computer refers to its internal memory for a new instruction step or operand, further operational steps would be useless and might even destroy information. An inactivity alarm, too, will cause an immediate trans-

fer, since to continue in this case means to continue the abnormal function. The overflow alarm can also cause an automatic branch to FIX, but this feature is designed so that the alarm may be suppressed in the operational program when it is known that overflows may occur during normal operation.

An automatic transfer of control to FIX is effected by setting the core memory parity, inactivity, and overflow alarm switches in the "active" position and the stop-branch switch in the "branch" position.

Transfers of data between core memory and magnetic drums may be monitored in another manner. Erroneous information that might be included in such a transfer cannot adversely affect the computer until used. Therefore, a drum parity alarm need not cause an immediate branch to the FIX program. Instead, the drum-parity-alarm switch is set to continue on alarm. At the conclusion of every block transfer, FIX checks the drum-parity-alarm indicator by means of a program instruction. Since the Air Defense Program was designed so that all transfers are controlled in one section of the program, the insertion of one interrogation instruction is the only modification of the operational program necessary to enable FIX to perform its entire monitoring function. If the alarm indicator is sensed inactive, there is no change in the normal sequence of events in the operational program. If at the end of a block transfer of data into core memory the appropriate alarm indicator is tested and found to be active, a programmed branch to the drum recovery section of FIX is effected.

DIAGNOSIS

At this point FIX will attempt to perform all diagnostic work necessary for recovery. Where time permits, FIX will also perform several diagnostic operations that are desirable for corrective maintenance studies. In all cases the results will be saved for logging and, depending on the circumstances, they may also be displayed immediately.

In the event of any type of alarm, initial FIX action would save the contents of all the computer registers, such as the accumulator, index registers, buffer register, program counter, etc., as they existed at the time of the error. This information is used as an aid to diagnosis, as part of the record of the error for maintenance purposes, and also to enable FIX to restore the environment of the Air Defense Program prior to effecting a recovery. The type of error, such as would be indicated by a drum parity alarm or memory parity alarm, will be determined by considering the mode of entry to the FIX program and by sensing the various alarm indicators.

If there has been a memory parity alarm, FIX will refer to the program counter setting as it was at the time of the error and will locate the incorrect information that was being operated upon at that time.

When the erroneous information is referred to a second time, a second memory parity alarm may or may

not be generated. Considering the case where a second parity alarm is not generated, FIX will continue its diagnosis by comparing this instruction or operand to the word that was parity-checked in the buffer register at the time of the error. If the contents of the buffer register match either the instruction or data word, FIX concludes that there was a false parity error, *i.e.*, an error in the parity-checking circuitry itself, and that the operation was in fact completed correctly.

If upon comparing the buffer register to the memory register FIX finds that the buffer register was completely zero at the time of the error, this would indicate that the alarm was probably due to a failure to get a start memory pulse and that no operation had begun when the alarm was generated.

Finally, a condition may arise where the buffer register is neither all zero nor equal to the instruction or data word in memory that supposedly generated the alarm. This would indicate a memory readout failure and an incorrectly completed operation. If a second parity alarm is generated on the second reference to the instruction or operand in core memory, the error is considered genuine, and, once again, the operation could not have been completed correctly.

The results of the investigation of each memory parity error are included in a record for maintenance purposes and will also serve as a guide to proper recovery action. No diagnostic action is taken in the event of an inactivity or overflow alarm other than saving the contents of the computer registers, and recording the type of alarm and the alarm exit location for the maintenance records.

Errors incurred during the transfer of information from the magnetic drums to the central data-processing unit are treated according to the class of drum involved.

Input status drums represent the supply of new information to the computer from an external source, such as a radar site. Under ordinary circumstances input data cannot be stored on a status drum by a program, nor is it possible to transfer the same information from a status drum to the central computer more than once. Consequently, a status drum is not normally available for complete testing and diagnosing by FIX without undue delay of the operational program.

Recovery from status-drum errors will vary according to whether the failure was transient or solid. Therefore, when a status-drum error is detected, FIX will examine the block of transferred information in core memory and, on the basis of the number of errors found, classify the failure as transient or solid. An erroneous status-drum transfer is classified as a solid failure if the number of errors contained in that transfer is more than five (an arbitrary figure). Further diagnosis for recovery and maintenance consists in determining the identity of the failing drum-input channels and the total number and frequency of similar errors.

Addressable drums serve as an auxiliary information-storage area. All data transfers to and from addressable

drums are performed under program control. Addressable drums are therefore readily available for diagnosing by the FIX program.

Upon noting an erroneous transfer of this type, FIX will search the block of transferred information in core memory for the information in error. The original version of data transferred incorrectly will remain unchanged on the drum until it is deliberately replaced with new data. For this reason, when an incorrect word is found, FIX will locate the original information on the drum and repeat the transfer of the incorrect word to determine whether the error was transient or solid. If the second attempt succeeds, a correct version of the word is now properly transferred, and by a comparison of the correct and incorrect information, the exact cause of the failure may be determined and saved for logging. This process is repeated if a second word in the transfer is in error.

The timing requirements of the operational program will not permit the luxury of individually treating more than two such words solely for maintenance purposes. If more than two words in a given transfer are found to have been in error, the remainder of the erroneous transfer is repeated at once, sacrificing additional diagnostic information for increased speed in recovery.

If any one of the recovery transfers is not successful on the second attempt, the error is considered to be of a solid nature. Further diagnosis is necessary, but recovery can still be achieved. Test data of known structure may be transferred over the same channels and checked by return transfer. Since the failure is solid, these transfers will also fail, but this time the exact nature of the failure can be determined. FIX maintains a history of results obtained in this way in a Learning Table. This table is used by FIX to compensate for future errors and also serves as a guide for corrective maintenance.

Fig. 2(a) represents an abbreviated word, consisting of five bits plus a parity bit, which FIX has determined had been incorrectly transferred into core memory from an addressable drum. The total number of "one" bits, including the parity bit, must be odd in order to be correct. The parity alarm which identified this word for FIX was a result of a check which indicated only that an even number of "one" bits was transferred in this word. Therefore, further diagnosis is necessary to determine which bit has been modified in transit.

Fig. 2(b) illustrates the standard method of testing the transfer channels to and from an addressable drum. By using a pattern of all "ones" and then of all "zeros," the channels may be tested for evidence of bit modification. This method, however, precludes the possibility that a unique pattern of bits in a word contributed to the failure of one particular channel. Investigation has established that failures may sometimes be uniquely associated with one word pattern and not with another. In the critical circumstances under which FIX is activated, it is felt that the extreme importance of being accurate has justified using a more detailed testing pro-

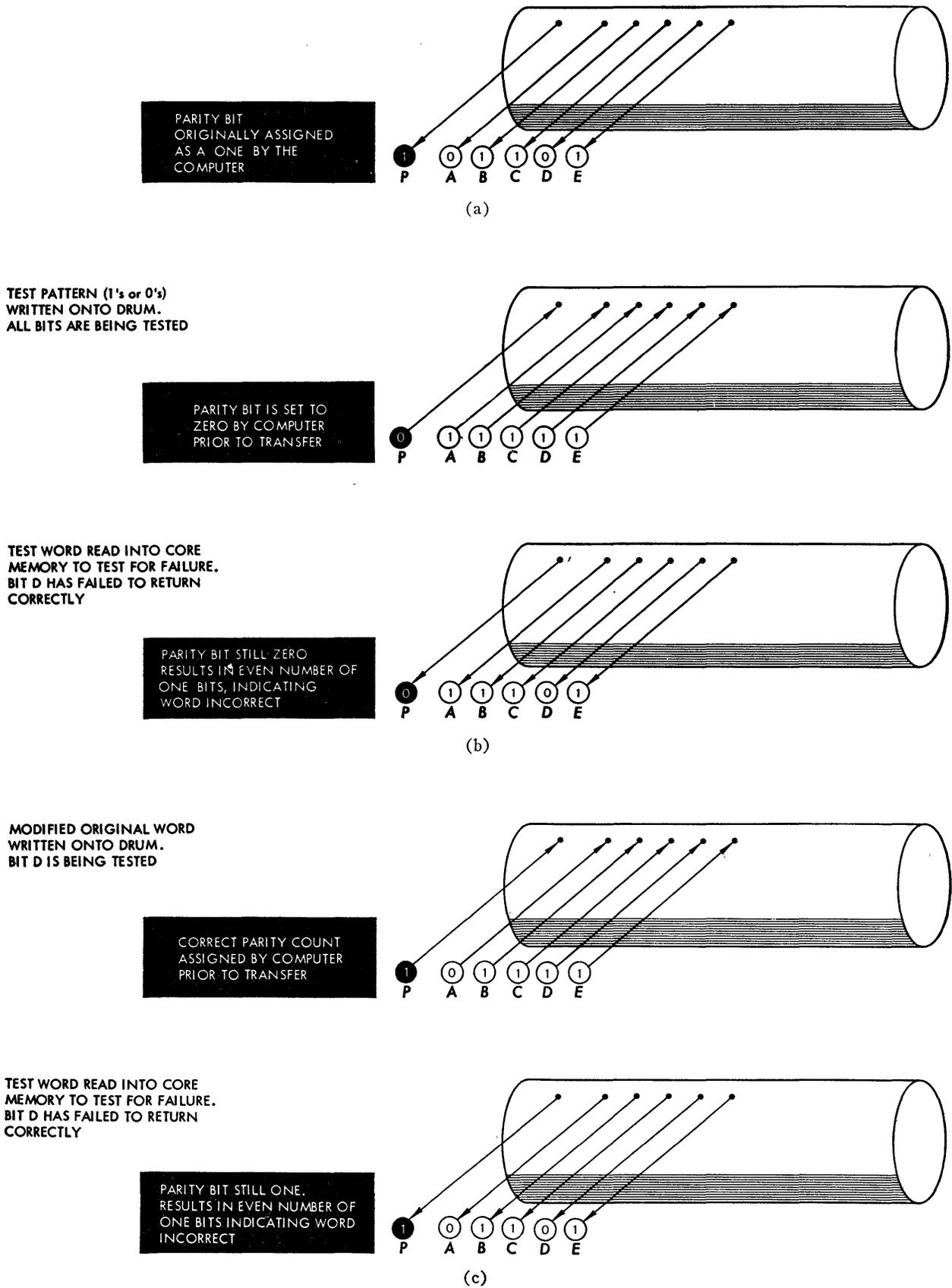


Fig. 2—(a) Even parity count indicates word incorrectly transferred but does not indicate which bit failed. (b) Standard test pattern technique for checking transfer channels comparing “before” and “after” words discloses the discrepancy. (c) Fix technique using original word [see 2(a) above] to check one channel at a time. The bit discrepancy is double checked by “before” and “after” comparison.

cedure than the common test pattern. FIX checks the transfer channels by using the original pattern of bits in the incorrect word as nearly as possible.

When a solid failure is detected, FIX first checks the Learning Table for a history of solid errors on this particular drum. If such records exist, FIX checks each transfer channel indicated by the Learning Table as having failed before. Fig. 2(c) illustrates a channel being tested in this manner. A bit which was suspected of having been lost in transit is changed to a "one" in core memory, and the entire word is then transferred to and from the drum to test this channel. If this bit fails to return as it was sent out—in this case as a "one"—this discrepancy is recorded. The Learning Table search will usually take no more than about 50 msec. If only one bit is found to be erroneous in this word at the completion of the Learning Table test, the results will be used to effect a recovery. If the Learning Table examination is not fruitful, recovery may still be achieved by further diagnosis.

In this event the next step would be for FIX to conduct a complete examination of the word. This is basically the same as the Learning Table test, except that the entire word is tested for evidence of failure instead of only those channels indicated by the Table. Each bit in succession is complemented, transferred to and from the drum along with the rest of the word, tested for evidence of modification in transit, and restored to its original state.

Any bit that fails to return in the same form as it was sent out is recorded. The bit-by-bit findings are accumulated until the end of the examination, which takes about one second. If the complete examination discloses that only one bit has failed in this word, the results will be used to effect a recovery.

LOGGING

All information gathered by the FIX program will be either printed on the teletype monitor, displayed immediately, or recorded in the Learning Table. The recovery that will follow is meant to improve the system reliability, not to shield equipment failures. If failures were not logged when recovery was achieved, the equipment could deteriorate with age until, without warning, catastrophic failure occurred.

During any alarm condition where automatic transfer occurs, FIX saves the current contents of all the computer registers for logging. The various alarm indicators will be tested to determine which type of alarm occurred. This information, together with the identity of the operational routine interrupted by the alarm, the data that was being processed at that time, and the details of the error as diagnosed will be logged on the teletype printer immediately after the error occurs. A record of the number of such errors will also be maintained on a display.

Status- and addressable-drum errors are internally recorded in the Learning Table and are also displayed as they occur. Each time a status-drum error occurs, the drum field in error and its input channel are recorded and displayed together with the total number of such errors recorded up to this time. The record and display for addressable-drum errors includes the drum field, failing transfer channel and the nature of the failure, *i.e.*, whether the erroneous bits were "ones" or "zeros," whether the failures were solid or transient, and the total number of such errors.

Enough pertinent information concerning each failure incident is logged to permit maintenance study teams to attempt to duplicate the trouble and keep detailed statistics on the reliability of the circuits in the system. Maintenance personnel can resolve machine difficulties only if this kind of logging is done. As experience is gained, the difficulties will recur less frequently, since equipment and program design improvements will be suggested by the statistics.

RECOVERY

The function of the recovery sections of FIX is to perform all operations necessary to restore control of the computer to the operational program. The choice of the method depends on the following factors:

- 1) The nature of the interrupting malfunction
- 2) The results of the diagnosis
- 3) The time spent in detecting and diagnosing
- 4) The number and frequency of this type of failure incident.

Some failure incidents in the central computer may be rectified by restoring the contents of the computer registers and internal memory to their original values, as saved by the alarm monitoring and control sections of FIX, and then transferring computer control back to the operational program at the point of interruption. This recovery method is most efficient, requiring up to about 30 msec, and is used, wherever possible, in the case of memory parity errors.

If diagnosis indicates that the operation was completed correctly, as in the case of a false parity error, the environment of the Air Defense Program will be restored, and recovery will be effected by reinitiating operations with the next program step following the one that operated at the time of the error. The exception to this would be if the interrupted instruction involved a transfer operation, in which case program control would be returned to the same transfer instruction.

If the instruction was never completed correctly, as in the case of a failure to get a start memory pulse or a memory readout failure, recovery will be attempted by reinitiating the Air Defense Program with the instruction that originally failed.

When an error is diagnosed as a genuine memory-parity error, the erroneous word must be corrected in core memory before the operational program can be resumed. At the present time FIX cannot automatically regenerate a correct version of a faulty word in core memory. However, since all of the program instructions and most of the data that are used by the Air Defense Program are stored on magnetic drums, FIX will attempt to locate a correct copy of a word in auxiliary storage and substitute it for an erroneous word in core memory. Whenever this is possible, recovery will then be achieved by reinitiating the Air Defense Program with the correct version of the instruction that originally failed.

If any of these methods are not successful in achieving recovery, *i.e.*, if the identical failure is immediately encountered, or if these methods are not feasible, as in the case of inactivity or overflow errors, an alternate method of recovery is available.

The Air Defense Program consists of a series of subprograms, each of which operates in its turn upon the latest input data fed to it. The entire process is an iterative one. After the last program has been completed, the first program will be called on to repeat its function upon the latest available data.

When an operation cannot be resumed at the point of interruption, recovery can often be achieved by reinitiating the Air Defense Program at the beginning of a cycle or frame of operations so that completely new input data may be processed. The startover procedure takes approximately five seconds. Thus, an abnormal condition which was the result of a transient failure will not degrade the performance of the system. A more serious or solid failure in the central computer will of necessity cause repeated restarts of the operational program. It is left to the discretion of the operator to impose limits on the number or frequency of the attempts to recover in this manner.

FIX does not distinguish between central computer errors that occur while the Air Defense Program is operating and those that occur during FIX operation. If a second error is encountered while one error is being corrected, the later error will take precedence. FIX will attempt to correct a central computer error within itself in exactly the same manner as an error occurring within the Air Defense Program. The original error, however, will then be disregarded, recovery of the Air Defense Program being achieved via startover. Of course a serious error in a vital section of FIX will preclude automatic recovery, and manual intervention will be necessary.

Recovery from transient status-drum errors is achieved in another manner. In the event of a status-drum failure, the testing procedure is limited by the fact that there is no practical way to make an experimental transfer between the central computer and a

status drum without a prolonged interruption of the Air Defense Program. Since FIX cannot determine exactly what failed in this type of transfer, it is not possible to estimate what the data should have been.

The solution is to render the erroneous data harmless by eliminating them from core memory, adjusting the Air Defense Program's records to compensate for the decrease in the amount of input information to be processed, and then returning to the logical operation in the main program that would normally follow the status-drum transfer. The momentary loss of some input data to the computer from a radar site, for example, will have no more effect on the Air Defense Program than would the slight interruption of radar fixes that are expected to occur during normal operation. Such temporary losses, or "miss fixes," as they are commonly called, are not unusual and the Air Defense Program provides for this by extrapolating or filling in for missing radar fixes when they occur. In this manner an accurate plot of the velocity of a hostile ship can easily be maintained despite the fact that a few positional fixes are missing, if the available fixes are dependably correct. A much smaller number of incorrect fixes can destroy the accuracy of a course plot if no means is provided to prevent these fixes from being included in the plotting computations and therefore the elimination of incorrect input data is much more desirable than treating such information as valid.

FIX cannot allow the situation to continue where a large number of consecutive errors reduces the flow of information to the Air Defense Program below an acceptable minimum. An excessive number of errors in one status-drum transfer will require that recovery be attempted by reinitiating the Air Defense Program at the beginning of a new frame of operations so that new input data can be called for and processed.

A dynamic display of all facts which are pertinent to this type of error is up-dated each time an error occurs. By observing the display, maintenance men may be able to determine the input channel from which most of the errors are coming. They may thus be able to eliminate or reduce the quantity of status-drum errors by substituting a spare input channel without interrupting the operational program.

Recovery from addressable-drum failures may be achieved by modifying a bit in the incorrect word or words in core memory according to the nature of the failure. Transient errors are corrected during the diagnosing operations.

In a solid failure during the transfer of a block of information to core memory from an addressable drum, many words may be expected to have transferred incorrectly. In an operational period, time does not permit that FIX be allowed to diagnose each word before correcting it. When FIX is satisfied that the Learning Table test or the complete examination has disclosed

the failing transfer line for one word, it will use this information to correct this word and the remaining words in error in the same transfer. In Fig. 3(a), let us say that Words 1-10 were incorrect in the block of transferred information shown. Suppose that a complete test of Word 1 indicated that channel A had failed, and that the bit in position A should have been a "one." After correcting Word 1 in core memory as shown in Fig. 3(b), FIX would continue to examine the remaining incorrect words. If a check of bit position A in each incorrect word indicates that it was possible that the identical error occurred in each incorrect word, these words would also be corrected in the same manner as Word 1. [See Fig. 3(b), Words 2-7.]

In an actual transfer, several thousand words might be involved. If a solid failure occurred, the number of words in error could be expected to be quite large. Consequently, if the error initially found in the complete test did not apply to all of the incorrect words in a transfer, it is felt that this fact would soon be obvious. Continued examination of the remainder of the incorrect words should reveal at least some words which could not have failed in the same manner. In Fig. 3(a), Words 8 and 10 now contain a "one" in bit position A. Therefore, that bit could not have dropped during the original transfer. This would indicate that another channel had failed in transferring these words and might also have failed during the transfer of any of the previously "corrected" words. This inconsistency would invalidate the "corrections" made earlier to these transferred data. In such a situation, recovery is effected by restarting the operational program at the beginning of a frame so that new input data may be processed. If more than one channel is found to have failed, this information will be immediately logged prior to initiating a startover of the Air Defense Program. The maintenance men may then determine whether or not it will be possible to permit the Air Defense Program to continue to process new input information.

The main section of the FIX program is also stored on an addressable drum. If an error is encountered in reading FIX into core memory, the diagnostic and repair sections cannot be used to correct this error. Instead, the permanent FIX routine in core memory will initiate a startover of the Air Defense Program. If this technique is unsuccessful for recovering from an error, manual intervention will be necessary.

RESULTS

After a six-month study of failures during 87 missions of the Air Defense Program of the SAGE computer, it was determined that the causes for failures were distributed as follows:

Drum parity (status and addressable)	53 per cent
Memory parity	10 per cent
Inactivity	21 per cent
Miscellaneous (power failures, stopped by operator, etc.)	16 per cent

PARITY BIT	INFORMATION BITS					
	A	B	C	D	E	
1	1	1	0	1	0	10
1	0	0	0	0	1	9
0	0	0	1	0	0	
1	1	0	0	1	1	8
0	0	0	1	0	1	7
1	0	0	0	0	0	
0	0	1	1	1	1	6
0	0	1	0	1	1	5
1	0	0	1	1	1	4
1	0	0	1	1	0	
1	0	1	0	1	1	3
1	0	1	1	1	0	2
0	0	1	0	1	0	
1	0	1	1	0	1	1
<hr/>						
0	0	0	0	1	0	
1	0	1	1	0	0	

(a)

PARITY BIT	INFORMATION BITS					
	A	B	C	D	E	
1	1	1	0	1	0	10
1	0	0	0	0	1	9
0	0	0	1	0	0	
1	1	0	0	1	1	8
0	1	0	1	0	1	7
1	0	0	0	0	0	
0	1	1	1	1	1	6
0	1	1	0	0	1	5
1	1	0	1	1	1	4
1	0	0	1	1	0	
1	1	1	0	1	1	3
1	1	1	1	1	0	2
0	0	1	0	0	0	
1	1	1	1	1	0	1
<hr/>						
0	0	0	0	1	0	
1	0	1	1	0	0	

(b)

Fig. 3—(a) A portion of a block of data transferred into core memory from an addressable drum. Words 1-10 were incorrectly transferred. (Note the even parity count in each erroneous word). An examination of Word 1 indicated that bit A had dropped in transit. This bit would be corrected in Word 1 and all other incorrect words which could have failed in the same manner. (b) Bit position A has been corrected in Words 1-7. The inconsistency in Word 8 prevented further attempts at recovery in this transfer.

On the basis of these results, FIX theoretically is capable of automatic recovery from 84 per cent of all failures occurring in the period studied.

Shortly after this study FIX was employed for an extended number of evaluation missions on the SAGE computer.

During this large trial period of computer operation, FIX provided automatic recovery from more than 92 per cent of the failures. Of the remaining errors, only about 2 per cent require unscheduled maintenance, the

TABLE I
SUMMARY OF FIX ACTION FOR EACH TYPE OF ERROR

Type of Error	Recovery Procedure
False memory parity: within Air Defense Program and/or FIX.	Reinitiate program with next instruction following the one that operated at the time of the alarm.
Fail to get start memory pulse, or other memory readout failure: in Air Defense Program and/or FIX.	Reinitiate program by repeating instruction that operated at the time of the alarm.
Genuine memory parity: in Air Defense Program and/or FIX.	Replace incorrect word with good copy from auxiliary storage drum and reinitiate program by repeating instruction that operated at the time of the alarm.
Inactivity, overflow, genuine memory parity that cannot be corrected, solid memory parity: within Air Defense Program and/or FIX. Solid status-drum failure (an excessive number of errors in one transfer) in Air Defense Program only. Drum parities while bringing in FIX: FIX only.	Reinitiate Air Defense Program at the beginning of a new frame of operation. FIX imposes no limits on the number of "startovers" that may be initiated in attempting to recover. The operator must determine if an excessive number of restarts is cause for manual intervention.
Transient status-drum parity: in Air Defense Program only.	Eliminate erroneous information from core memory, adjust records of Air Defense Program and continue operations at point following transfer.
Transient addressable-drum parity: in Air Defense Program only.	Repeat transfer. Re-enter Air Defense Program at point following transfer operation.
Solid addressable-drum parity: in Air Defense Program only.	Correct erroneous words in core memory. Restart Air Defense Program at point following transfer. If diagnosis is inconclusive for purposes of correcting error, restart the Air Defense Program at the beginning of a new frame of operation.
Any catastrophic error from which FIX does not successfully recover.	Manual intervention.
Errors which do not result in memory parity, drum parity, inactivity or overflow alarms.	None; FIX not activated except by the alarm circuitry of the computer.

other 6 per cent being due to operator and program errors.

It appears that the inclusion of FIX in the tests increased computer efficiency. In addition to the improvement in system performance achieved with FIX, a long-term gain should be realized in basic equipment reliability and useful operational time because of the increased accuracy of the maintenance information supplied by FIX.

In summary, the advantages offered by FIX are these: Recovery from most errors is accomplished automatically, almost immediately, and accurately, thus assuring the correct results with a negligible amount of lost operational time. Table I is a summary of FIX action for each type of error.

The Learning Table record permits more immediate correction of some errors the second time they occur.

Solid errors in addressable-drum transfers, for example, have been virtually eliminated as a source of reduced computer efficiency.

The logging feature of FIX affords a detailed record of all errors exactly as they occurred as an improved aid to corrective maintenance.

Although the advantages and results obtained so far have been limited to one specific application in the SAGE system, it is felt that variations of the FIX concept can be successfully applied to other operational and production programs written for a data-processing system.

ACKNOWLEDGMENT

We wish to acknowledge the cooperation of those who assisted with the first trials of the FIX program, which were made at Lincoln Laboratory, Lexington, Mass.

A High-Speed Data Translator for Computer Simulation of Speech and Television Devices

E. E. DAVID, JR.†, M. V. MATHEWS†, AND H. S. McDONALD†

INTRODUCTION

THIS PAPER describes a data translator which, when used with a digital computer, permits simulation of speech- and television-processing devices.

The usual function of such devices is to realize efficient codings of information-bearing signals thereby economizing on transmission requirements. The merit of any particular coding must be based upon human judgment. Such subjective reactions can seldom be predicted by analytical means. For example, the evaluation of speech-transmission systems relies heavily upon quality and intelligibility tests using human observers. Thus, instrumentation of operating models to generate test speech samples invariably becomes a part of any such evaluation. In addition to transmission codings, signal-processing devices have been used to investigate automatic recognition of abstractions such as phonetic elements in speech or geometrical figures. Such recognition functions typically require even more extensive instrumentation.

Models for studying coding and recognition schemes currently being considered are complicated, involve great logical complexity, and require large flexible memories. The cost of construction is high not only in money, but in the time and energy of the researcher. Several years are often involved in adequately testing relatively simple notions, and systematic evaluation of some well-founded proposals has been delayed more than 20 years. Clearly, the need for extensive instrumentation is a bottleneck limiting creativity in this area.

It is now possible for most laboratory purposes to eliminate the construction of much complicated equipment by means of simulation with general-purpose digital computers. A computer can be made to act the part of any specified device for the duration of an experiment and requires only a changed program to transform its action to simulate a new device. Not only are savings in money and time realized, but many other equally important advantages can be achieved. For instance, the range of ideas which can be considered is greatly expanded, the simulated device and the data can be precisely controlled, and device parameters can be easily and flexibly modified.

Simulation of speech and television signal processing devices requires an input and output translator capable of delivering such signals into the computer and recover-

ing them from the computer without appreciable degradation. We have described in a previous paper¹ one attempt to satisfy this requirement. A block diagram of that translator is shown in Fig. 1 and we shall review its operation and limitations before describing a new high-speed version. The analog input is filtered to a band limit of W cps, and then sampled $2W$ times per second. An analog-digital converter codes each sample into 11 binary digits which appear in parallel at its output. Ten of these digits are monitored by the recording and playback electronics whose function it is to arrange the digits in the proper format and record them on a magnetic tape medium. The initiation, cessation, and timing of these events are assured by the control unit. This translator is bilateral, recording tapes for computer input, and decoding tapes written by the computer. Thus the processed computer output can be displayed audibly or visually for subjective evaluation.

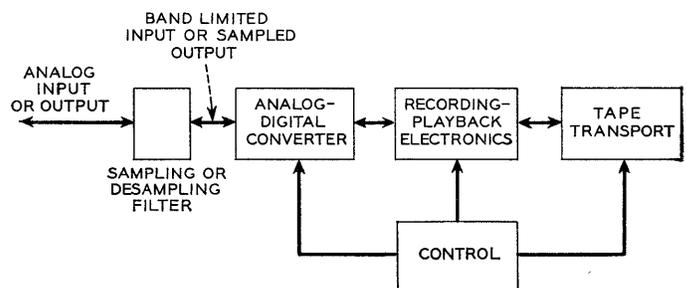


Fig. 1—Original data translator.

The operating parameters of the translator are set by the tape format, which of course is in turn determined by the computer input requirements. Our translator is designed to match the IBM 704 computer which calls for a seven-track nonreturn-to-zero recording with digits recorded simultaneously in each track. There need be 200 such seven-digit characters per inch on the tape. The translator operates in two modes. In mode I, each input sample is represented as a six-digit binary number (the remaining four digits are discarded) and is recorded in one tape character (only six places are available in each character since the seventh place is reserved for a parity check digit). In mode II, each input sample is represented as a 10-digit binary number and

¹ E. E. David, Jr., M. V. Mathews, and H. S. McDonald, "Description and results of experiments with speech using digital computer simulation," 1958 WESCON CONVENTION RECORD, pt. 7, pp. 3-10.

is recorded in two tape characters. The five most significant digits, identified as such by a zero in the sixth place, occupy the first character, while the five least significant digits, identified by a one in the sixth place, occupy the second character.

In either mode I or mode II, the character density on the tape together with the linear tape speed fixes the input sampling rate. For instance at a tape speed of 50 inches/second, $50 \times 200 = 10,000$ characters per second are recorded. In mode II, this figure corresponds to 5000 input samples/second. Three tape speeds are available on the tape transport, and the corresponding input sampling rates are summarized in Table I. These figures show both an upper and lower limit on the sampling rate. Experience with speech and television simulation experiments have shown that these limits are inconvenient constraints² on the analog-signal bandwidth. For instance, unless the analog signal is subjected to a time-scale transformation before its introduction into the translator (such a transformation can be accomplished by recording and reproducing the analog signal at different tape speeds), a 5000-cps speech wave can be represented to only six-digit accuracy. Even time-scale transformations do not alleviate these limits greatly, since record-reproduce speed transformations using analog recorders introduce their own signal degradations and thereby have restricted application.

TABLE I

Tape Speed Inches/Second	Mode I Samples/Second	Mode II Samples/Second
12.5	2500	1250
25	5000	2500
50	10,000	5000

The synchrony of the sampling and recording processes in this translator not only places an upper and lower bound on the available sampling rate, but in addition restricts that rate to a number of discrete values. This limitation has likewise proved to be an inconvenience limiting the flexibility of the translator. Two further shortcomings result from this same basic cause.

First, for ease of programming signal-processing experiments, it is desirable to have the digital data divided into several fixed-length blocks called "records," each separated by a gap in which nothing is recorded. The present translator has no provision for so punctuating the tape with "record gaps." Recording must proceed continuously during the presence of an analog input if it is to be sampled without interruption. Such a procedure often results in long recordings which are inconvenient for the computer to handle.

The second shortcoming is of even greater importance and arises from irregularities in character spacing on

output tapes written by the computer. These irregularities are introduced by the low-inertia, fast start-stop tape transports associated with the computer. When such tapes are reproduced through the translator, these fluctuations appear as "flutter" and "wow" in the analog output. Though this effect has been minimized by careful selection of tape transports, it remains the most severe limit on the fidelity of reproduction.

To recapitulate, all of the limitations imposed by the translator, namely: 1) upper and lower bounds on input sampling rate, 2) only discrete sampling rates available, 3) data blocks on digital tape of arbitrary size, and 4) analog output having flutter and wow, result from the inherent synchrony between the input-output sampling rate and the recording-reproducing rate on the digital tape. The remainder of this paper describes the design and construction of a new translator which avoids these problems by divorcing the two rates.

SYSTEM DESIGN OF DATA TRANSLATOR

The limitations in the existing translator have been overcome by incorporating a buffer storage between the analog-digital converter and the digital tape recorder. During recording, the buffer stores the samples of the input while the tape recorder is inserting record gaps. Thus convenient fixed-length records may be produced without interruption of input sampling. During playback, the buffer has sufficient capacity to continue delivering characters (to the digital-to-analog converter) during record gaps in the digital tape, thereby assuring an uninterrupted flow of output samples. In addition, the buffer can smooth the jitter in the flow of samples, producing a uniformly timed sequence to the digital-to-analog converter.

A buffer of practical size can hold only a small amount of data compared to the amount of data passing through it in most applications. Consequently, to prevent completely emptying or filling the buffer, the average rate of recording samples must be equal to the average rate of sampling the input. (This discussion will be limited to the recording process in most cases. Extension of these considerations to the playback process is usually obvious.) Matching the average rates while maintaining a uniform character spacing on the digital tape requires controlling the average tape speed. This control could be accomplished either by servo control of the speed or by using a fast start-stop tape mechanism and varying the idle period. The latter procedure was chosen because it results in a much simpler mechanism.

A block diagram of the buffered system during recording is shown in Fig. 2. The system is constructed to allow any sampling rate from zero to the maximum allowable, to produce constant-length records, and to automatically control the start-stop cycle of the tape recorder so as to prevent the buffer overflowing. The analog signal is sampled, digitized, and the digits transmitted into the buffer each time an external synchronizing signal is applied. A preset reversible counter keeps

² R. E. Graham and J. L. Kelly, Jr., "A computer simulation chain for research on picture coding," 1958 WESCON CONVENTION RECORD, pt. 4, pp. 41-46.

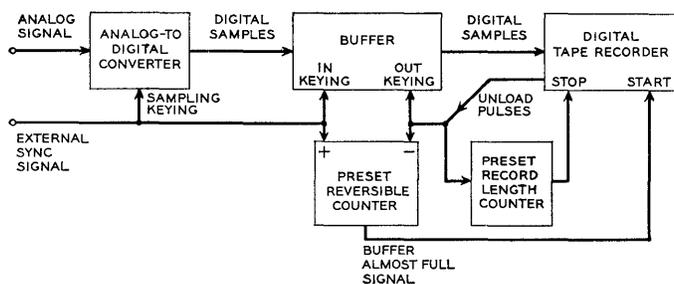


Fig. 2—Buffered recording system.

track of the contents of the buffer, and when sufficient samples almost to fill it have been accumulated, the counter emits a signal which starts the tape recorder. For each character to be recorded, the recorder produces an unload pulse which causes the buffer to deliver one character's worth of digits. The unload pulse also counts down on the reversible counter and up on a record-length counter. The latter, after a preset number of characters have been recorded, stops the tape recorder, which then waits until the next "almost-full" signal is received to start recording the next record. Buffer loading continues during the recording cycle. This procedure requires the buffer to interleave loading and unloading operations in any sequence, and possibly to load and unload simultaneously. A modified commercial unit described in the next section meets these requirements.

During its ON cycle, the recorder operates at a constant speed, thus simplifying the recording electronics over that required for a variable speed machine, and increasing the reliability. If the number of characters in a record is set to be less than the size of the buffer, and input sampling rate is less than the rate at which the buffer can be emptied, then the input sampling rate is entirely independent of the recording rate. Indeed, asynchronous sampling can be used equally well.

The maximum sampling rate depends on the recording rate, the record length, and the minimum idle time of the tape transport. The relation for the maximum is

$$\text{maximum input rate} = \frac{\text{recording rate}}{1 + \frac{\text{minimum idle time}}{\text{record writing time}}}$$

where the minimum idle time is the minimum time to produce one interrecord gap and the record writing time is the time to write the characters in one record. For the equipment described below, the recording rate is 30,000 characters per second and the minimum idle time 4.7 milliseconds. Thus for a record length of 1000 characters, the maximum input rate is very close to 26,000 characters per second (26,000 mode I samples per second or 13,000 mode II samples per second).

During playback the digital-tape machine supplies characters to the buffer and the digital samples from the buffer are converted to analog output samples at the

command of an external synchronizing signal. The operation sequence is such that one record of characters is put into the buffer from the tape initially. Thereafter, each time the buffer becomes almost empty, as indicated by a preset reversible counter, another record is put into the buffer. The tape transport is started by the "almost-empty" signal, and stops at the end of each record of data. All starting and stopping is thus done in the record gaps, and hence no data are read while the tape is accelerating or decelerating. This control prevents the buffer from becoming empty and divorces the rate of delivering output samples from the rate at which data are coming from the tape.

The buffered recording system thus overcomes the principal limitations of jitter, inflexible sampling rate, and record length which harassed the operation of the original data translator. In addition, by using a higher-speed transport, a substantially faster recording rate is achieved. The details of the new translator are described in the next section.

DATA TRANSLATOR DESCRIPTION

A photograph of the new high-speed data translator to perform the task of recording speech and television signals on digital tape is shown in Fig. 3. The unit is contained in three racks. The rack at the left houses the tape transport and read-write electronics. The center rack contains the control circuits and the buffer storage unit, while the rack on the right contains the analog-to-digital and digital-to-analog converters.

Many of the components of the new high-speed system are commercially available stock items. The tape transport is an Ampex FR-300 Instrumentation Tape Transport with a tape speed of 150 inches/second. It was found necessary to construct special reading translators to attain the desired reliability. The recorder produces all features of the IBM tape format including lateral parity bit, longitudinal parity character, and end-of-file marks and gaps. During playback a clock generator which forms a logical "or" of all the seven tracks to produce a character synchronizing signal is required to provide timing. The IBM format of using six data bits plus an odd parity sum makes this unit necessary since it is possible to have a character with only one "one" in it and that "one" can occur in any of the seven tracks.

The buffer storage unit is a Telemeter Magnetics 1092-BU-7R buffer which is capable of storing 1092 seven-bit characters. The buffer is capable of either loading or unloading one seven-bit character in 10 μ sec but it must not be loaded and unloaded simultaneously. In this application, input data must be loaded during writing on the tape, so that buffer is augmented by the addition of a seven-bit storage register and a guard circuit. In the composite unit, loading of the buffer is unchanged except it is delayed for 10 μ sec by the guard circuit. This circuit establishes a zone which starts 10 μ sec before and ends 10 μ sec after the actual loading of

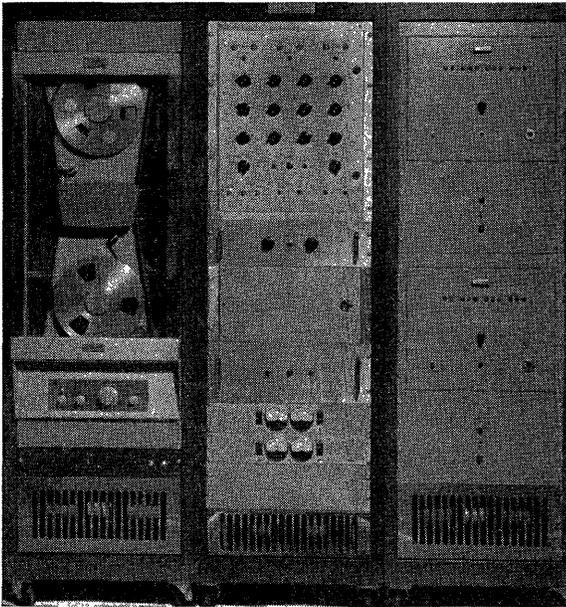


Fig. 3—The new high-speed data translator.

the data. If an unload is attempted during the time of this zone, the actual unloading is delayed until the end of the zone. In this manner, the actual loading and unloading of the magnetic cores can never occur within $10 \mu\text{sec}$ of each other. As a result, unloading is no longer uniform in time and may be delayed up to $20 \mu\text{sec}$ depending upon the loading conditions. An additional storage register removes the $20 \mu\text{sec}$ uncertainty in unloading by holding the output data from the magnetic core unit until the subsequent unload operation is attempted. The augmented buffer unit, comprising the core storage, the guard circuit, and the seven-bit storage register can now be loaded and unloaded simultaneously. However, the changes have reduced the maximum rate of cycling of the buffer from 50 kc to 33 kc and have introduced a delay of one unload-sample-time in the output.

The analog-to-digital and digital-to-analog conversions are performed by two Datrac B-611 converters manufactured by Epsco, Inc. These units are equipped with a sample-and-hold feature and can sample and convert data at any rate up to the tape writing rate. Eleven binary digits are available to form two tape characters which specify the input signal to 0.05 per cent. The FR-300 allows simultaneous recording and playback. During recording, the tape is monitored by the playback heads and the resultant pulses are converted to analog samples by the digital-to-analog converter. Because the buffer is occupied during recording, the output samples are not buffered and appear only when the tape is in motion, but such a signal is adequate for monitoring with an oscilloscope.

The control circuits are composed of about 130 solid-state logic units which perform the functions of data sorting, data counting, and control to integrate the tape transport, the buffer, and the converters. This unit, along with the packaging of the other components, was designed and constructed to our specifications by the D. G. C. Hare Co. of New Canaan, Conn. A great deal of care was taken to prevent uncertainties from creeping into the time of input sampling, the output sampling rate and duration of the output samples. During playback, the sample duration is controlled by the time between positive and negative zero crossings of the external synchronizing signal. The operator supplies this signal so presumably the duration can be made as precise as desired. Barring any digital errors, the only signal distortion is the quantizing noise, which is less than 0.05 per cent and timing uncertainties which are functions only of externally supplied synchronizing signals.

There are several functions which are manually executed by means of front panel controls. The digital tape can be punctuated by an end-of-file mark by means of "Write End of File" control. Any residue of data remaining in the buffer at the end of a recording can be dumped on the tape by the "End of Data" button. During playback the unit will read out data until an end of file is sensed, at which time it will stop and disconnect the external synchronizing signal as if the stop button had been depressed. A reset control clears the buffer and all of the logic. Front panel indication of buffer full, buffer empty, and parity error are provided.

EFFECTIVENESS OF NEW TRANSLATOR

The principal advantage of the new translator is its high speed and accuracy. This unit is capable of recording data in computer format on tape moving at 150 inches/second.

The unit has a channel capacity of 150,000 binary pulses per second. These pulses are used to represent analog signals to either 11- or 6-bit accuracy at sampling rates up to 12.5 and 25 kc, respectively.

The rate of sampling input data is independent of digital tape speed and data density on the digital tape. The rate of output samples from digital tapes is also independent of these factors. It is possible, for instance, to record a speech wave sampled 10,000 times a second and play it back at two samples per second into a pen oscillograph.

There are no discontinuities in gathering data, yet the translator produces a digital tape with convenient punctuation gaps at frequent intervals. This punctuation is removed automatically when reproducing the signal. There is no wow or flutter in the output due to fluctuations in the tape motion.

Some Experiments in Machine Learning

HOWARD CAMPAIGNE†

EVER since the development of automatic sequence computers it has been possible for the machine to modify its own instructions, and this ability is the greatest single faculty in the complex that tempts the term "giant brain." Friedberg¹ demonstrated a new technique in the modification of instructions; he allowed the machine to make alterations at "random," and lent direction to the maneuver by monitoring the result. This technique is far from being a feasible way to program a computer, for it took several hundred thousand errors before the first successful trial, and this was for one of the simplest tasks he could imagine. A simple principle of probabilities shows that a task compounded of two tasks of this same complexity would take several hundred thousand times as long, perhaps a million computer hours. It is the object of this study to examine techniques for abbreviating this process.

The work reported here is not complete, nor is it likely to be for several years. The field is immense, the search proceeds slowly, and there are few clues as to where to look. This paper is, therefore, in the nature of a preliminary report.

The memory of a computer can be pictured as a piece of scratch paper, ruled into numbered cells, on which notes can be written or overwritten. I set aside a portion of this memory in which the computer is supposed, somehow, to write a program. Some of this dedicated space is for data and some for instructions. I programmed a generator of random numbers, and instructed the computer to use these numbers to write a program (and later to modify a program already written). The method of using the random numbers is such that all addresses generated refer to the data, and all instructions generated are from an approved list. After a little thought one sees that by putting sufficient limitations on what the computer is allowed to write the result will be a foregone conclusion. The object is to design an experiment which leaves the computer relatively free from limitations but which will still lead to a meaningful result. Friedberg did this.

One way for the machine to write the instructions is to have it write random bits into a form word, the form and the dedicated spaces being selected to be coherent.

Now a simple task is envisioned and a monitor routine written to test whether the randomly generated program (which Friedberg christened HERMAN) has accomplished the task. If it has it is tried again until the probability is high that the task is being done correctly.

If at any step the task is not done then a change is made before another trial. It is clear that once a routine which can do the task has been arrived at no further changes will be made. The procedure can be pictured as a random walk. The number of possible programs is finite; in one of my experiments it was 2^{96} . The rule for going from one trial to another can be chosen in various ways, but in the same experiment there were just 64 alternatives at each step. The number of routines which satisfy the test must be, judging from my results, on the order of $2^{96}/2^{12} = 2^{84}$. For some tasks and for some repertoires of instructions it can be estimated directly.

AN EXAMPLE OF A MACHINE WRITTEN PROGRAM

Operation	Operand	Address	Data	
QJP	27 00	0447	0440	13351604
COQ	24 00	0443	0441	77777777
LDQ	22 00	0453	0442	0
STQ	23 00	0457	0443	57312317
COQ	24 00	0452	0444	0
QJP	27 00	0452	0445	77777777
LDQ	22 00	0445	0446	77777777
QJP	27 00	0457	0447	77777777
STQ	23 00	0444	0450	0
LDQ	22 00	0444	0451	0
STQ	23 00	0450	0452	0
QJP	27 00	0454	0453	77777777
COQ	24 00	0445	0454	54040476
STQ	23 00	0442	0455	77777777
COQ	24 00	0447	0456	77777777
LDQ	22 00	0451	0457	0

In designing these experiments one has a tremendous number of choices. There is the repertory of instructions from which the machine chooses to make up its routine. The instructions of this repertory need not be selected with equal probability; some can be used more often than others. There is the size of the area of the memory dedicated to the random routine. There is the task to be performed. There is the time allowed the routine to make its trial. And there are a multitude of other variations, some of which will be mentioned later.

I first used the simplest repertory to be found capable of performing the Sheffer "stroke" function. It is:

LOAD from y into the accumulator,

STORE at y from the accumulator,

COMPLEMENT the accumulator,

JUMP to y if the accumulator is positive.

This differs drastically from that used by Friedberg.

The area of the memory devoted to the random routine must be in two parts, one for data and the other for instructions; otherwise the machine would try to execute data and come to an intolerable halt. Therefore the address y of the JUMP order must be interpreted differently from those of the LOAD and STORE orders.

A problem related to that which leads to the separation of the two dedicated areas is that endless loops are highly probable and intolerable. This problem can be

† American University, Washington, D. C.

¹ R. M. Friedberg, "A learning machine," pt. I, *IBM J. Res. and Dev.*, vol. 2, p. 2; January, 1958.

solved by the routine which interprets the jump addresses. At the occasion of a JUMP a calculation is made about the time of running; if the time is excessive then the trial is terminated and called a failure. If the time is acceptable then the address to which the jump is made is interpreted.

In an earlier experiment I used a clock to time the routines and adjusted the jump addresses with a B-box. The use of a clock made checking and debugging very difficult and eventually forced me to the later method.

I have used several sizes of dedicated spaces and plan to try still more, but most of the trials have been with sixteen lines of coding. Friedberg used sixty-four lines. Clearly one must allow enough lines to permit coding the task. More than enough slows the "learning" process. To allow just enough prejudices the event and is not "fair." A felicitous solution to this quandary would be to find a way to expedite the "learning" so that very large areas of memory can be dedicated, thus guaranteeing that the answer has been supplied by the machine. It is the object of this study to find such a solution.

The time allowed to execute the routine can be varied. Its absolute minimum is that for three instructions. The maximum could be quite large if the routine were retraced several times. Oddly enough the time allowed on each trial has only a small effect on the number of trials before learning. This may be because when the machine has extra time it uses that time to destroy what it may already have accomplished. If so then two effects tend to neutralize each other; the freedom of more time tends to lengthen the learning period, and the greater number of potential solutions tends to shorten it (Table I).

TABLE I
THE EFFECT OF TIME ON LEARNING

Time	Median number of trials	Number of experiments
14	4400	31
10	5800	47
7	6800	30
4	15,500	13

These experiments differed only in the time allowed each trial. The time is measured by the number of program steps possible. Each experiment was run until the machine had demonstrated that it succeeded (one hundred consecutive successes) and then the number of trials was recorded. In this set of experiments about 38 per cent of the trials appeared by accident to be successes. Thus the number of different programs tried was about 62 per cent of the total number of trials. As the time was shortened it became more difficult to find a successful routine.

The median has been quoted here to avoid a bias which might affect the average. The range of the number of trials is large, and there might be some prejudice against the longest runs, such as stopping them to check for faults.

The task selected was to transfer a word from one place to another. This was chosen because of its simplicity. An easier task was to copy a word at a prescribed spot from one of several places (redundant input). A harder task would be to copy a word at several places (redundant output). With redundant input (the key word written at two accessible places) the median number of trials was 2600, compared with 5800 in non-redundant experiments.

The procedure for the machine making changes in the routine offers many opportunities to be different. In one series of trials the routine was rewritten completely after each failure. In another only one instruction was rewritten, the instructions being taken in turn in successive tests. In another series just one instruction was rewritten, this time chosen by an elaborate procedure involving "success numbers." The success numbers were accounts, one for each instruction of the routine, which were increased when a success or what appeared to be a success happened, and decreased when a failure occurred. This procedure was roughly the same as that of Friedberg's, although not quite as elaborate.

A comparison of the results of these alternative methods were that "learning" occurred most rapidly with the first method, about twenty-five hundred errors before a success. It was less rapid with the second, about six thousand trials, and slowest with the last, over one hundred thousand trials. The "learning" is an abrupt process, a "flash of insight." The procedure with success numbers resembles that used in other experiments with self-improving programs, such as those used by Oettinger.² It seems to be inappropriate here since a line of coding is not itself a unit. If the program were organized into units to which success numbers could be appropriately applied then it would begin to appear that the answer to our problem was being built into our approach. Success numbers will have an important place in the ultimate "learning" machines, but some sort of self-improving without them will also be required.

A SAMPLE OF EXPERIMENTS ARRANGED IN ORDER

No. of Trials	Apparently Right	No. of Trials	Apparently Right
20	20	576	273
38	32	592	280
54	37	598	303
68	39	612	317
72	46	614	297
76	40	618	290
76	47	680	331
88	53	800	385
156	77	996	509
160	89	1090	526
162	88	1100	526
234	131	1246	595
222	124	1324	629
276	130	1412	688
294	153	1420	678
330	160	1528	753
338	173	1616	782
352	160	2016	990
386	181	2872	1366
521	249	3198	1522

² A. G. Oettinger, "Programming a digital computer to learn," *Phil. Mag.*, vol. 43, pp. 1243-1263; December, 1952.

TABLE II
SYNOPSIS OF EXPERIMENTS

Type of change		Length	Time	Task		Competing twins?		Median no. of trials before learning	Notes
All	One at a time			Transfer word	Choose an exit	Yes	No		
	X	2	2		X		X	548	
X		14	10		X		X	960	
	X	14	10		X		X	1530	Reversed exit
X		16	12	X			X	1567	Input redundant
	X	16	12	X		X		1775	
	X	8	4	X		X		1902	
X		16	10		X		X	2097	
	X	16	10		X	X		2210	
	X	8	4	X		X		2280	
X		16	10		X		X	2608	Reversed exit
	X	16	10		X		X	2683	
X		16	10		X		X	2998	Reversed exit
	X	16	10		X		X	3125	Input redundant
	X	16	10		X		X	3610	Input redundant
X		16	10		X		X	4200	
	X	16	12	X			X	4300	
	X	16	10		X		X	4401	
X		16	10		X		X	4406	
	X	8	4	X			X	4988	
	X	16	8	X		X		5177	
	X	8	4	X			X	5280	
	X	16	10		X	X		6700	
	X	16	7		X		X	6800	
	X	16	10		X	X		7273	
	X	16	10		X		X	8600	
	X	16	8		X	X		9266	

In these experiments the minimum possible space, two orders, was allowed for the program. The learning was judged complete when 20 successive right answers were given. Thus there is a chance of something less than one in a million that a given experiment did not succeed, despite appearances. Notice the wide range, the slowest taking 160 times as long as the fastest (Table II).

These experiments throw some light on this kind of "learning," and particularly on which versions learn fastest. The process is analogous to evolution, since a routine survives only if it meets the challenge of its environment. This analogy suggests some further experiments where each trial is built on two successful trials, perhaps by taking lines of coding from each, much as genes are taken from chromosomes. I have hopes that routines can be built in this manner to meet complex environments and that the number of trials will be only the sum of the number for each individual requirement rather than the product.

In the experiments described above once a task has been "learned" there is no further improvement. It would be desirable for the routine to improve its per-

formance even after it had demonstrated acceptable skill. This can only be done by some sort of flexible criterion of satisfaction, and by some way of keeping progress already made. I have tried to do this by twin learning routines. The twins compete to see which will first learn the task. When one of them has learned the other continues to try, but now it must complete the task more quickly than its sister. In this way some improvement takes place. It is not quicker than the alternative of insisting on a high standard *ab initio*.

Other techniques need to be tested. One of these is to use as components not lines of coding but subroutines. In this way the average coherence of the trials should be raised. Another is some way of accumulating successes and then using them cooperatively to meet more and more complex environments. This resembles biological history, where evolution has produced increasingly complex organisms which become more and more effective in dealing with their environment.

If someone could invent a technique which produced programs as effective as organisms in a time which is electronic rather than biological it would be a revolution in programming.

Some Communication Aspects of Character-Sensing Systems

CLYDE C. HEASLY, JR.†

INTRODUCTION

A LARGE number of bad jokes have been made about the "real characters" encountered in the reading machine business. If the jokes are not too funny, they serve nonetheless to highlight a significant fact. Character-sensing machines must read real characters if the machines are to be useful outside of the laboratory.

Practical character-sensing machines have been in use for a number of years. Enough experience has now been accumulated to warrant an analysis of the environment in which they must operate. Specifically, this will involve an understanding of the nature of real characters, the factors which determine their nature, the techniques which may be employed to improve these factors, and some of the considerations which must be judged in determining which techniques are appropriate to a given system.

A CHARACTER-SENSING SYSTEM MODEL

Fig. 1 shows a character-sensing machine enmeshed in the conventional model of a communication system. Messages are sent along this communication system by selecting a series of symbol shapes and transferring the shapes in inked form to a document. The document is then moved by various handling processes to the point where the message is to be received. There the document is scanned, the shapes converted to signals and the signals decoded.

As the model is considered in greater detail, the encoder is found to degrade the symbol shapes by variation in strength of impression and in amount of inking, and by inking noise. On typical business equipment, these factors may result in line-width variations of five to one, missing portions in light impressions, blotted portions of heavy characters, random additional interference on either light or dark characters, and a pronounced ribbon pattern.

The handling process further degrades the symbol by superimposing interference. In the worst cases this may be offset endorsement stamp ink, or in less drastic cases, a mild smudging of the printed characters. Finally, the sensing mechanism and quantizing apparatus cooperate to produce a signal representative of the character shape. Whether the decoding of this signal will yield the original intended characters depends on how carefully information has been conserved all along the channel.

† Intelligent Machines Res. Corp., Alexandria, Va.

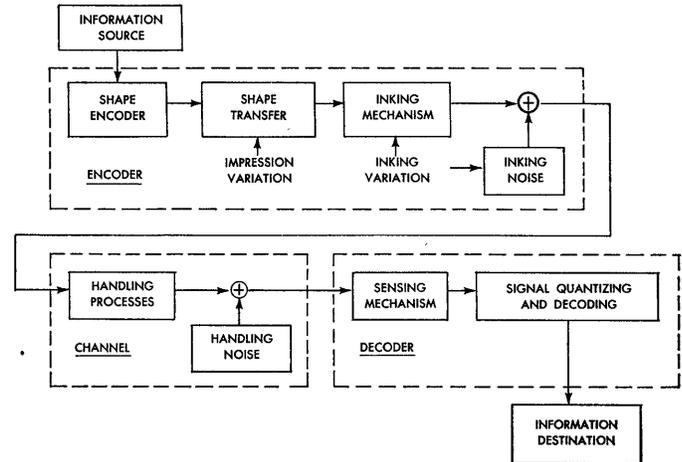


Fig. 1—The character-sensing machine is the decoder of a communication system.

THE ENCODER—PRACTICAL DATA PRINTERS

Selection of a typeface which complements the reading scheme or combats typical interference is one of the least expensive changes. Apart from tooling costs, type bars or wheels are inexpensive. The symbols must be similar for the conventional characters which they represent and should conform to the limitations imposed by the printing techniques to be used. Within these constraints, however, there is ample room for the designer to increase the amount of information which represents each character.¹

Regardless of the type shapes used, it is intended that the shape be completely black and that everywhere else the document be white. It is the job of the printing process to approach this intended ideal as closely as possible. Direct printing from inked type and lithography border on the ideal. Unfortunately, such processes can only be used for fixed or serial data which are applied before the document is in field use.

Variable data are frequently printed by key-driven business equipment, typewriters, adding machines, cash registers, and the like. Generally these equipments print with ribbons and metal type. When the recorder is built from scratch, it is desirable to use controlled impression and direct-transfer ribbons or the newer ink-saturated rubber type. Another class of data recorder is the identification imprinter which prints from embossed plates by means of ribbons, ink-saturated rollers, or carbon papers.

¹ C. C. Heasley, Jr., "Selfcheck—a new common language," presented at ACM Symp., Los Angeles, Calif.; May 8, 1958. Copies available at Intelligent Machines Res. Corp., Alexandria, Va.

However, the data recorder is the most numerous element in many character-sensing communications. The ability to use existing business equipment with only minor modifications often provides the economic impetus for a character-sensing system. Thus the system designer may find himself limited to changing typeface and perhaps use of an improved ribbon.

A third source of variable data is tabulators and various high-speed printers which record the output of data processing systems. These printers for the most part use ribbons and, because of their high speed, a direct transfer ribbon becomes a very costly nuisance.

The preponderance of ribbon printers and low-cost recording devices virtually forces system designers to examine their output and to find ways to recover the information which they record. The principal characteristic of ribbon printing is the dot structure imposed by the ribbon itself. The size of the dots varies with ribbon inking, strength of impression, and chance alignment between type and ribbon. Under light impression and light inking, some dots may fail to print. Under normal inking, the dots blend together to form lines of appreciable width with irregular boundaries. Heavy inking and pressure may result in considerable bleeding of ink well beyond the intended boundaries of the character shape. In nearly all cases, unintended spots occur in the nominally white areas of the character, their number and darkness being dependent on inking and impression.

These dots of varying size and intensity are not individually reliable. However, they may be aggregated by statistical operations to yield reliable bits which define the strokes of the characters. The printing resolution of the dots, usually 8 to 10 dots from top to bottom of a character, may be statistically "quantized" to a reliable vertical resolution of five bits. This is equivalent to stating that a five-bit code might be printed reliably in the vertical space occupied by a character symbol. As will be shown, this resolution is about the minimum which can be used to print symbols which resemble conventional character shapes. Fig. 2 shows enlarged views of typical characters under three different impressions.

THE COMMUNICATION CHANNEL—HANDLING NOISE AND DISCRIMINATING INK

Handling noise, surprisingly, can often be controlled by fairly simple methods. For example, on bankchecks the largest sources of noise are the handwritten figures, signature, etc., and wet endorsing ink offset from the back of one check to the front of the next. Intelligent format design can isolate handwritten data from the reading field. Endorsing stamp offset can be controlled by restricting the variety of ink colors to those which can be discriminated by color analysis scanning.

Regardless of the printing method, the choice of printing medium offers interesting possibilities. Conventionally, the sole requirement of inks is good optical contrast. However, where the primary source of noise is in the handling rather than in the printing process, it



Fig. 2—Character line widths and noise vary greatly with inking and impression.

is possible to use inks which have more than one sensible property. The most widely known is of magnetizable ink which has optical properties for human recognition and magnetic properties for machine sensing. A second approach which has likewise been used in the banking problem is the use of glossy inks. Here varnish agents cause a glossy character surface which can be sensed in combination with direct black to discriminate against all but the extremes of handling noise. An even more powerful approach can be achieved by use of a combination fluorescent ink which appears black under normal illumination, but which fluoresces under special illumination. Here again, combination sensing techniques may be used to overcome interference due to handling.

It is worth noting that selection of a printing medium is in no way helpful in combating the noise resulting from the printing process itself. For example, the carbons used in credit identification invoice forms are the chief source of noise in that system. Efforts to reduce this noise by improved carbon have been very rewarding, but no advantage would be obtained by using magnetic or fluorescent material, since the noise would have the same property as the characters.

SENSING METHODS

Perhaps the most important system choice and certainly the most interesting from an analytical point of view is the choice of a sensing method. The two choices worth serious consideration are amplitude scanning and two-dimensional scanning.

For quite a while these scanning schemes were closely associated with the medium in which the characters were sensed. Magnetic character sensing has been associated almost exclusively with amplitude scanning, while optical character sensing has been based on two-dimensional scanning. However, two-dimensional scanners have recently been developed for magnetic characters, and an amplitude scan system has been announced for optical character sensing.

The principal difference between the two schemes is the way in which they respond to the two-dimensional information in a character. Two-dimensional scanning covers each point of the area, usually in a scanning raster similar to television. Amplitude scanning compresses information in one direction into an amplitude signal. For example, in magnetic sensing, amplitude scanning is accomplished by passing the characters under a vertical gap. The scan signal responds to the total amount of magnetic ink passing under the gap. As reading progresses, the read head produces an output function related to the horizontal distribution of magnetic material along the character. Although it responds to the vertical amount, it cannot respond to the vertical distribution of the contributing elements. This results in a loss of information.

Superficially considered, the amount of information lost by amplitude scanning may seem trivial. However, even casual analysis will reveal that the signal recovered from scanning a conventional "2" is virtually identical with that of a conventional "5." Similarly, the difference between conventional "4" and "9" or "0" and "8" are alarmingly small. In practice these pairs may be rendered different by changing the horizontal scale of one or by changing widths of vertical strokes. This is a perfectly valid solution to the problem, but these examples serve to underscore that shape differences easily detected in a two-dimensional scan are obscured by amplitude scanning. It is of some interest to gain a more exact knowledge of information contained in characters and of the ways in which it is preserved or lost.

ANALYSIS OF SCAN INFORMATION

The maximum amount of information which can be used to convey a set of character symbols may be said to be the number of bits which can be resolved by scanning and quantizing the "printing cell" occupied by one character. It will be shown presently that the value of this resolution will depend on the scanning method, but for the moment let resolution be considered independently of scanning method. This may be thought of as the channel capacity in bits per symbol. While channel capacity could be used as the independent variable, it is both more convenient and more meaningful to relate results to an independent vertical resolution which may be defined as the number of bits which can be resolved along a vertical line extending from the top to the bottom of the cell.

The manner in which amplitude scanning reduces information can be easily understood from an example based on a vertical resolution of "4," although better resolutions are required if the constraint that the symbols resemble conventional characters is to be obeyed.

With a vertical resolution of 4, 16 different inputs, X_{ij} may occur. The amplitude scanner can respond to such inputs by providing 5 different outputs, y_j . Fig. 3 is a matrix which shows the joint probabilities P_{ij} relating the 16 inputs and the 5 outputs.

		FIVE POSSIBLE OUTPUTS, y_j					
		0	1	2	3	4	
SIXTEEN POSSIBLE INPUTS X_i	0000	$\frac{1}{16}$	→ $\frac{1}{16}$
	0001	.	$\frac{1}{16}$.	.	.	→ $\frac{1}{16}$
	0010	.	$\frac{1}{16}$.	.	.	→ $\frac{1}{16}$
	0011	.	.	$\frac{1}{16}$.	.	→ $\frac{1}{16}$
	0100	.	$\frac{1}{16}$.	.	.	→ $\frac{1}{16}$
	0101	.	.	$\frac{1}{16}$.	.	→ $\frac{1}{16}$
	0110	.	.	$\frac{1}{16}$.	.	→ $\frac{1}{16}$
	0111	.	.	.	$\frac{1}{16}$.	→ $\frac{1}{16}$
	1000	.	$\frac{1}{16}$.	.	.	→ $\frac{1}{16}$
	1001	.	.	$\frac{1}{16}$.	.	→ $\frac{1}{16}$
	1010	.	.	$\frac{1}{16}$.	.	→ $\frac{1}{16}$
	1011	.	.	.	$\frac{1}{16}$.	→ $\frac{1}{16}$
	1100	.	.	$\frac{1}{16}$.	.	→ $\frac{1}{16}$
	1101	.	.	.	$\frac{1}{16}$.	→ $\frac{1}{16}$
	1110	.	.	.	$\frac{1}{16}$.	→ $\frac{1}{16}$
	1111	$\frac{1}{16}$	→ $\frac{1}{16}$
		↓	↓	↓	↓	↓	
		$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	
		OUTPUT PROBABILITIES, q_j					
		JOINT PROBABILITY, P_{ij} MATRIX					
		AMPLITUDE SCAN, VERTICAL RESOLUTION 4					

Fig. 3—Probability matrix shows how amplitude scanning response loses information.

The information content $H(X)$ of the input (which for the present is assumed to be equiprobable) can be seen to be 4 bits.

$$\begin{aligned} H(X) &= - \sum P_i \log_2 P_i \\ &= - 16(1/16 \log_2 1/16) = 4. \end{aligned}$$

The probabilities, Q_j , of the five scanner outputs are found by summing the elements of each row of the matrix. From these in turn the output information $H(Y)$ is found.

$$\begin{aligned} H(Y) &= - \sum Q_j \log_2 Q_j \\ &= - (2/16 \log 1/16 + 8/16 \log 4/16 + 6/16 \log 6/16) \\ &= 2.03 \text{ bits.} \end{aligned}$$

Thus, for every vertical column of 4 bits in the input channel, an amplitude scanner captures 2.03 bits and loses 1.97 bits. Regardless of the number columns in the cell, nearly half of the input information is destroyed.

The graph of Fig. 4 shows how the efficiency of this type of scan decreases as the resolution increases. The actual channel capacity (input information), conditional information (lost information) and transinformation (output information) are shown for a character cell having aspect ratio 5 to 4 (vertical to horizontal).

CHARACTER INFORMATION

Whether or not this information lost is meaningful, of course, depends on the extent to which the input symbols utilize the channel capacity. Fig. 5 shows a set of symbols based on a vertical resolution of 5 with 5 to 4 aspect ratio. It can be seen that this is about the poorest resolution which can be used and still obtain ten symbols which resemble the numeric digits. It can also

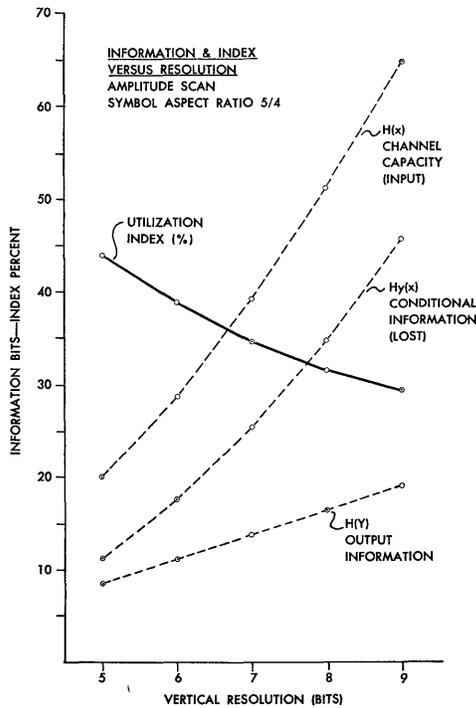


Fig. 4—Efficiency (utilization index) of amplitude scanning decreases as resolution increases.

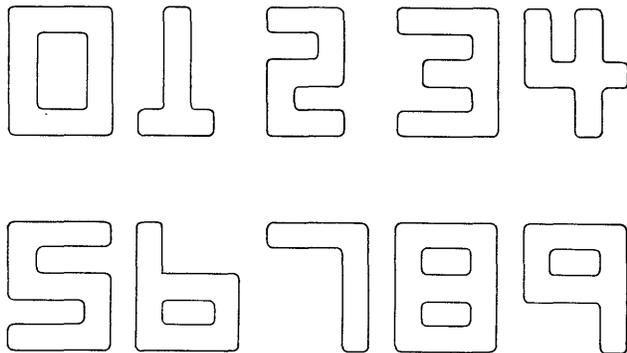


Fig. 5—Exemplary numeric font based on vertical resolution of 5 uses 15.4 bits (out of possible 20).

be seen that variations in style are quite restricted. This font has been deliberately designed to circumvent the limitations of amplitude scanning by changing the horizontal scale of the "2" to differentiate it from the "5."

The channel capacity is 20 bits. The type font utilizes 15.4 bits, of which 8.18 bits are recovered by the amplitude scan.

VERTICAL RESOLUTION AND QUANTIZING

It may not be readily apparent that the vertical resolution as defined does not have the same value for a two-dimensional scan as for an amplitude scan. This difference arises from the methods whereby quantizing may be used to effect resolution.

A two-dimensional scan system offers the possibility of quantizing in two steps. The first step involves examining each elemental area of the image to determine whether it is intended to be black or white. The second

step involves performing a statistic on the results of the first step to determine whether a particular small area is intended to be black or white. It is in this sense that the resolution has been defined, the small areas which can be resolved in a printing cell being the bits which comprise channel capacity.

An amplitude scan must perform the equivalent quantizing in a single step and must quantize to a larger number of levels, in the 5 to 4 case just shown, an amplitude scan must have the ability to quantize to six output levels in order to resolve the five vertical bits.

The significance of this relationship between quantizing and resolution is more apparent in the case of nonuniform impression. Fig. 6 shows the two examples of the character 5, one printed light and one dark. It is at once obvious that a two-dimensional scanning system could easily resolve either input character with a vertical resolution of 5 bits. The transmitted amplitude-scan information is shown below in the two examples. If the two amplitude-scan signals are normalized so that maximum responses are equal, quantizing to six levels will produce the two different results shown. In order for printing to have resolution of 5 bits for amplitude scanning, vertical line widths of horizontal strokes have to be constant within plus or minus one-sixth of the nominal line width. It can be shown that this is equivalent to a vertical resolution of twenty-five bits for a two-dimensional scan.

Thus it is seen that, if amplitude scanning is used, the freedom from handling interference which magnetizable inks and magnetic sensing offer, is purchased at the cost of considerable loss of recovered information. This may be compensated by greatly improved resolution. Either the printing cell must be greatly increased in area or the printing must be held to very close tolerances. Since character area is at a premium in many applications or fixed in advance by the printing mechanism, close printing tolerance is apt to be the only answer.

QUANTIZING AND DECODING

In most practical character sensing decoding equipment, it is difficult to draw a fine line between the quantizing mechanism and the decoding mechanism, since these system elements often work in close cooperation. It is probably true that the real sophistication in character-sensing machines is in the quantizing mechanism. It is in improved quantizing methods that information may be preserved and engineering considerations of simplicity, reliability, and cost may be most profitably pursued.

Once the information in a character has been sensed and quantized, the decoding of the resulting signal is fairly straightforward. Choice of logic schemes is not as complicated as might be supposed. The success of one decoding scheme as compared to another will depend primarily on the efficiency with which it utilizes the redundancy of the quantized signal to promote accurate decoding.

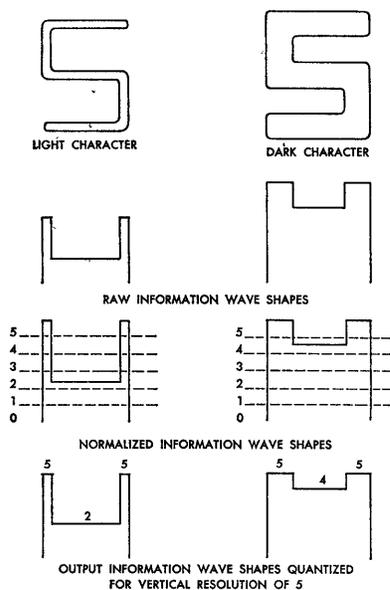


Fig. 6—Amplitude scanning is greatly affected by variations in impression.

Thus the character is seen to be a code which is superior to other conventional codes only in the excess of information it embodies and in the ease with which decoding can be performed by human beings who have learned to read. It might be remarked that a fair portion of the information present in real characters has to be used in combating the low fidelity and noise introduced by conventional printing methods. These considerations aside, there is little choice between bar or dot codes and conventional characters so long as area and printing resolution are identical.

Once the nature of the character is identified, it becomes quite apparent that any decoding scheme will give better performance if it has more information as its input.

SUMMARY

The attempt here has been to consider the entire communication system of which the character-sensing equipment is but one functional part and to identify both the factors which reduce information transmitted by the system and the techniques which may be used to insure that the message is finally received ungarbled.

If there is any single conclusion to be drawn, it is perhaps that the best way to achieve economical character sensing is to consider the whole communication system in which it is embedded and to improve the information transmitted by every element of the system insofar as economical and system requirements permit.

Beyond this the points at which the most improvement per dollar could be obtained might be re-emphasized.

Since the printers are frequently the most numerous elements in the system, the extent of change is usually limited. Changes of typeface are inexpensive and may be quite effective in improving the amount of information per character. Improvements which enhance printing resolution are the most rewarding. A simple increase in the character cell area is as effective as improving the printing tolerance, but frequently neither improvement can be made without printer redesign. Reduction in printer interference is particularly worthwhile, especially if special inks are needed to combat handling interference.

The best solution to handling interference is to control the source of interference. If interference cannot be eliminated by document design and improved handling procedures, it may be possible to restrict the kinds of interference to those which can be discriminated by special sensing techniques. If this is not possible, then use of special inks which may be detected outside of the interference medium is useful.

The scanning employed should recover as much of the character information as possible. From an information point of view, it may prove to be better to use scanning which conveys all of the character information even though degraded by interference, than to scan in a medium which is free of interference with a scanning technique which degrades the character information. This last conclusion is particularly true when close printing tolerance cannot be maintained, when alphabetic or alphanumeric information must be transmitted, when the typeface cannot be controlled in advance, or when serious interferences are inherent in the printing process.

With respect to quantizing, considerable care is warranted to insure that the information is quantized rather than some other variable which is only casually related to the information.

In this regard it is well to note all *a priori* information about the symbols and the printing process. Full utilization of this information in the quantizing is the only way to insure maximum information at the decoding stage.

Finally, it might be said that system design of character-sensing communication systems is primarily a matter of selecting that set of system elements consistent with the problem requirements which will convey the maximum amount of information. It is hoped that this survey of system elements and their characteristics has been helpful to that end.

An Approach to Computers That Perceive, Learn, and Reason*

PETER H. GREENE†

THE purpose of this paper is to mention some of the problems which I believe will be central ones in the effort to design machines that can in some sense be said to perceive, think, learn, make reasonable inferences, or learn to perform acts of skill. First I wish to discuss a general conceptual outlook which has important consequences for our approach to the design of such machines. I shall show how this outlook calls for modifying or supplementing some current approaches; however, specific techniques for dealing adequately with the problems raised remain to be found. The second part of the paper will point to specific techniques which now exist for dealing with the problem of reasonable inference and inductive judgment, but which to my knowledge have not been considered in connection with computers and control mechanisms.

Polya [34] gives certain heuristic principles whereby we tentatively arrive at judgments of the likelihood of propositions. Yet he categorically states that there can never be a machine which can perform such plausible inferences. Why is this? I suggest that one reason is merely that he did not anticipate progress which has occurred since his book in the analysis of such inferences. I shall discuss this progress later in this paper.

Another reason for Polya's pessimism is more fundamental. Present-day machines cannot really be said to contain the meaning of the propositions they handle. Every proposition is reduced to a hole or an electric charge or a magnetic field at some point of space. Thus the propositions may be sorted and combined but their meaning resides in the mind of the person who looks at a list which says that a hole in this place means that proposition. Pattern recognition, too, depends in all but a few experimental computers upon preset responses which do not take meaning into account. Thus it has become fashionable to talk in terms which exclude meaning. We recognize the next letter or word, so it is said, because we know the transition probabilities. This is only a partial solution; it ignores the outstanding fact that we are able to use relevance and meaning as guides, rather than probabilities alone. The first topic will concern a general conceptual outlook which is essential if computers are ever to become capable of having this too.

Most of our perception is too rich to be described in what is called a discursive code, a one-dimensional sort of language that can be written on a typewriter or spoken or put on magnetic tape. Thus we shall need to make use of non-discursive symbols (or presentational symbols in Susanne Langer's terminology) which are rich enough to embody complex patterns in their entirety as presented to the computer. Such recognition, pattern by pattern instead of element by element, has been discussed before—for instance, in connection with Uttley's conditional probability machine [35]. However, the basic mechanism for the recognition of patterns in all these studies is that of abstraction, which is supposed to account for the recognition of complex patterns and general ideas. This makes use of association, the formation of a linkage between representations of similar patterns.

That abstraction as generally defined is totally inadequate as a fundamental principle has been pointed out convincingly by many authors, who unanimously arrive at that judgment from diverse starting points. For their arguments I refer you to the bibliography on judgment and concepts. A first inadequacy of abstraction as the fundamental mechanism of concept formation involves the person's or computer's way of approaching the matter to be examined. If I hold up my hand with one finger extended, what makes you look in the direction of my finger instead of the direction of my shoulder, or indeed look in any direction whatever? The answer is that the meaningfulness of the gesture depends upon a background of conventions which determine one's way of approach and which must precede any alleged abstractions. Thus more is needed than abstraction [12].

The second fundamental difficulty is that abstraction is said to proceed by the formation of groups of things which are similar in regard to the abstracted concept. But the formation of these classes presupposes some notion of the concept just to determine which things are similar in regard to it. It presupposes at least a certain "point of view" from which the elements can be designated like or unlike. The development of the concept then involves a further articulation of this general point of view [3].

The third fundamental difficulty is that abstraction works by forgetfulness. Cassirer points out that this separates us more and more from the pattern and produces something superficial. He contrasts this with the generality reached, say, when a mathematician generalizes. For instance, it is a generalization to say that cir-

* This research was supported by the U. S. Air Force through the Air Force Office of Sci. Res. of the Air Res. and Dev. Command under Contract No. AF 49(638)-414. Reproduction in whole or in part is permitted for any purpose of the U. S. Government.

† Committee on Mathematical Biology, University of Chicago, Chicago, Ill.

cles and ovals are all instances of the same concept, namely ellipses of various eccentricities; but this generalization, rather than neglecting the particular features, embodies a universal rule for the connection and ordering of the particulars. In general, the essential feature of the general concept is a generating relation which gives a principle of ordering and dependence. We have seen that this is presupposed by the ability to form classes of similar elements. It is also presupposed by the ability to make reasonable judgments about a whole group of forms without running through the particular examples—often an infinite set—individually. To do this we need some representation of a generating rule of construction [3].

Thus the first task which precedes abstraction is not the comparison of representations but the formation of impressions into representations adequate to sustain abstraction. We tend to think that abstraction alone is sufficient because this first task has for the most part already been performed in infancy and in the construction of our traditional language. We therefore presuppose that the raw materials for our judgments are not disconnected particularities, but are already given to us as a connected manifold. The essence of these arguments was first arrived at by Kant in 1781 [2], [5]–[7], [9], [10].

The consequences for computer design are many. First, we must find a way of building up meaning, which, though it may construct hierarchies of patterns in a way called for by many authors, must use essentially richer means of construction than the means called for by those authors. That is to say, we need more than the principles of association and abstraction, although these will be important ingredients. We shall see later that abstraction itself, in its valid aspect, requires more than association. Second, the construction of the unitary rule of connection of a manifold—the “first universal” of the 19th century logician Lotze—requires a principle of construction which supplements the relation of class inclusion which is appropriate to the formation of the “second universal”—the class of similar objects. It is this latter relation alone which has been studied in any detail by people interested in computers. Third, since the truly general concept contains structure which includes the generative principle of the system in which it stands, it follows, as Kant and Cassirer indicate, that no content can be posited without thereby positing a complex of other contents. In other words, true generality embodies the judgments of causal relations which we are seeking to embody in computers [2], [5], [6].

Fourth, we must not regard the linguistic concepts of the computer as copies of a definite world of facts; if so, we are doing the machine's job of drawing the significant outlines of meaningful units. Rather, the computer, before it compares experiences, must in Cassirer's words concentrate the experiences and distill them down to one point. This point is the unit which is at present

handled in the ingenious ways known to computer designers, but here we see that an involved process must go into its formation. Since the particular outlines constructed depend upon generative principles, it would be expected that any conceptual product of such a computer would be constructed out of manifolds of related precursors formed around individual generative principles built into the machine. This appears to be the case in human thinking [1], [13], [17], [19]–[21].

Fifth is the need to supplement those notions of the relations between a language and the world that have so far been worked out, for example, Carnap's treatment of mathematical semantics. In treatments like this a sentence is true if and only if the fact to which it refers happens to be the case. Thus they assume an extralinguistic world of facts open to observation. In our problem, on the contrary, observation is possible only through symbolic forms contained in our language; hence we may say that for us, or for the machine which we envisage, the world of observables comes into existence along with the language. The same drawback is apparent in the analysis of meaning given by Morris, which is very frequently cited in connection with our problem. The first link between information theory and meaning, namely the notion of semantic information of Carnap and Bar-Hillel, suffers from the plight of logical atomism, the doctrine that complex ideas may be analyzed into one of the 2^N possible conjunctions of N atomic facts or their negations. Please bear in mind that I am not making a single complaint against any of the methods I have cited. These methods, together with all existing proposals for logical networks, must be supplemented by something which produces the meaningful symbolic forms which are the raw material for these methods and networks. At present it is the human operator, not the machine, which produces this raw material [11], [29]–[31].

I shall conclude this conceptual part of the paper with a few remarks on what we would mean by subjective action and thinking in a machine. The question is very involved and I am not competent to provide an adequate answer. I shall therefore name only a few properties which we can all agree on as necessary, and which will lead to the discussion of specific techniques which are now available.

We may all agree that one feature that we seek is the presentational nature of the internal symbolism of the machine, involving the ability to perceive patterns too complex to be communicated verbally and the ability to acquire skills which we are not able to program discursively for the machine. Since the internal workings of the machine would proceed by the manipulation of non-discursive symbols, while the only known means of communication with the machine would be via discursive symbols, we could find out how the machine was arriving at its conclusions only by a long process of inference, and we could modify its basic patterns only through a gradual process of “training.” The patterns

would have enough internal structure so that new information could be obtained by operations based upon content, rather than the external operations of sorting and propositional logic which present computers use. The symbolic operations, rather than the visible receptor structure, would be the entities closely related to subjective meaning. For instance, we should not want to call the pattern on a television screen or the visual cortex subjective. These are patterns which are causal transformations of external objects. The construction of subjective meaning from these raw materials must proceed in the ways which I have discussed in the first part of this paper, and in fact does proceed in that way in human beings; the fundamental units of perception being dynamic, unstable visual effects which are combined into a coherent image, and which may become dissociated in pathological states. Feeling is more closely related to these processes of combination than to nerve pathways, as we observe when we feel the texture of an object with another object such as a pencil held in our hand and tend to localize the meaningful feeling in the end of the pencil rather than in our hand.

By consulting the references provided on psychology, it will be possible to learn how all the concepts I have discussed are embodied in examples of what I believe to be the only plausible and nontrivial psychological results and hypotheses which actually deal with the mechanisms for the synthesis of meaning. Anyone interested in the problem of intelligent machines should consult these references to see how one might go about building meanings on a basis of non-discursive symbols [13]–[24].

Next I wish to discuss that aspect of subjective meaning that involves our feeling that the meaning should in some sense be self-contained—that the machine read its own dials and in some sense understand them the way we would. In this way the machine's consciousness would be a collection of internal feedbacks which had some special feature. What is this feature? I think one aspect may be seen by means of an example. Goodman [32] uses a tick-tack-toe diagram to represent isomorphically the pattern of hostilities existing among four gorillas in a zoo. The four lines stand for the four gorillas, and the four intersections stand for the four relations of hostility which happen to obtain. To us intersecting lines seem unlike gorillas so we say that the full meaning is not self-contained. On the other hand, our mental gorillas seem just like physical gorillas so we say that we contain the meaning. Since physical gorillas are known to us only by means of mental gorillas, of course they seem similar. For our present purposes we might say that the machine contains the meaning of what it perceives in case its internal dials register symbols whose complexity compares favorably with the complexity of our own ideas.

This notion suggests a topic which I think is worth investigating. Reichenbach [26] remarks that it seems to be a general property of semantical systems that pre-

cise concepts in the object language may arise from vaguer concepts in the metalanguage (the language which talks about the object language). Can we find languages embodying the complexity of our experience which are based upon a hierarchy of meta- and meta-metalanguages and so on down to a language so simple that it can be adequately embodied in the electronic units (or the physiological units) available to us? This possibility would be consistent with the psychological views which I have mentioned.

The last aspect of subjective meaning that I wish to discuss is the one which will allow us to proceed from the *terra incognita* of the synthesis of symbolic forms to more welcome regions in which valuable resources have already been discovered. This is our well-known ability to become acquainted with causal relations and with the potentialities of things and actions, and to make appropriate generalizations and inductive inferences. This part of my paper involves the recent progress mentioned in connection with Polya's pessimism regarding intelligent machines. This progress involves methods which were devised mainly with reference to meaning and inference in the methodology of science. However, we can use them to handle the content of our science, which involves just these problems of inference. Perhaps it will be best to list a substantial number of specific logical tools so that you may get a general idea of the scope of these methods and turn to the reference for details of any that sound interesting [25]–[28].

The basic fact to remember is that all these methods deal with the weighing and balancing of judgments against a background of relevant material which in general is not explicitly expressed, and the resulting rules of logical manipulation differ considerably from the customary symbolic logic of isolated propositions. Among other differences, the validity of a statement will depend not only on what is presented but on how it is presented, with logically equivalent statements often not having equal validity. Validity will involve matters of past linguistic usage and thus the whole previous history of how the world has been described and anticipated by use of the language. Thus these techniques deal realistically with common sense judgments in their full complexity of context and suggest that inductive judgments in machines may be a real possibility in the not too distant future.

To begin the list we may consider a problem treated by Goodman [25]. This is the problem of *contrary-to-fact conditional* statements, or *counterfactual conditionals*. These are statements such as "If I had struck the match it would have lighted," which we regard as true, and "If I had struck the match it would not have lighted," which we regard as false. Since we did not strike the match, and the match may not exist at present, these conditionals (which are thus contrary-to-fact) may not be directly verified, and the question arises as to what criterion we use to establish the truth of one and the

falsity of the other. The ability to judge what consequences *would* result from certain circumstances or actions which are not actually present is an essential part of intelligent behavior. It turns out that the problems to be met and criteria to be formulated in judging counterfactuals suggest specific tasks that would have to be performed by components of intelligent machines.

The first problem is to define the set of relevant conditions which are necessary for the truth of the counterfactual (oxygen, dryness, etc., in our example). We cannot include all true statements because many of these conflict with the antecedent of the conditional, since by definition it is a contrary-to-fact conditional. Here is another blow at simple association as an adequate basis for relevance. We need more structure. It turns out to be a difficult problem to specify in advance what type of information must be considered. We shall come to an application of this specification later.

The next consideration is that not all conditionals may be used counterfactually, but only certain ones which express laws as opposed to mere facts. The structure which distinguishes laws from facts turns out to have potential applications to structures in machines, and is moreover intimately connected with the rule that it is not the truth of laws but their inductive verification that makes them laws. Reichenbach in particular gives rules for the structure of laws that may be translatable into rules of structure and function in inductive computers. The problem of counterfactuals leads Goodman to the problem of establishing dispositions—the potentialities of things to behave in certain ways under certain circumstances—on the basis of behavior which has been actually manifested. The ability to embody dispositions is another important feature of intelligence. Now the class of objects having a given disposition (*e.g.*, flexible) is broader than the class of objects having the manifest property (in this case, is flexed). The problem of extending the smaller class to the larger, which Goodman calls *projection*, leads him to a consideration of induction. The main problem here is that of choosing among the infinite variety of generalizations consistent with given evidence, and this is another important feature of intelligence. Goodman and Reichenbach give rules for the resolution of conflicts between generalizations which seem as though they could be embodied in a conditional probability machine like that of Uttley, or else in models of the nervous system proposed by Hebb and others. The exact form of a proposition is important to its mode of confirmation since it is possible for the same evidence to confirm one proposition yet be irrelevant to the truth of a logically equivalent proposition [25] (“paradox of the ravens”).

Reichenbach [26], [27] concentrates upon the formal properties of implication in ordinary and counterfactual conditionals and in laws which make them seem reasonable or unreasonable. It is his great achievement to have formulated rules concerning the detailed logical structure of propositions which amazingly allow statements

that seem reasonable to common sense and exclude most statements that seem unreasonable, despite the fact that the difference is one which seems most subtle and elusive and which depends upon the exact form in which the statement is expressed in addition to its extensional truth value. I shall select certain topics which will be applicable to computer design. His rules are generally such that they could probably be programmed for computers with no more trouble than any other program now employed.

Reichenbach first discusses the rules for a form of implication (which he terms *connective*) which expresses necessary connections, and thus differs from the usual form of implication of symbolic logic (or *adjunctive* implication), which merely expresses the fact that certain conditions are adjoined. To know the truth or falsity of the adjunctive implication “if A then B” one must first ascertain the truth or falsity of “A” and of “B” and call the conditional true in case these truth values stand in a certain relation to each other, whether or not there is any natural connection between A and B. The connective implication, on the other hand, asserts a connection between A and B; it asserts that if “A” were true then “B” would be true, and we can decide whether such a connection exists without knowing the truth or falsity of “A” and “B.” Connective implications are involved in most reasonable inferences.

Using his notion of connective implications, Reichenbach is able to define a class of *nomological* statements, which correspond to laws of nature and logic, and a narrower class, the *admissible* statements, which correspond to ordinary reasonable statements. He distinguishes three orders of truth, analytic, synthetic nomological, and factual, and is able to arrive at a characterization of reasonable statements based solely upon the orders of certain parts of the statements. Then he states a variety of rules for the transformation and combination of reasonable statements which differ substantially from the ordinary rules of symbolic logic. He proposes modified rules to cover what he terms *semi-adjunctive* implications, which make assertions about individual events with no claim that the same result will obtain upon repetition.

With these rules, also conceivably adaptable to computers without too much trouble, we come to our last group of possible applications to intelligent computers.

First we consider the problem of structuralization raised by the question of what are the relevant conditions. Practically everybody who has considered the problem of learning in computer or person considers the basic operation to be a certain kind of association whose inner structure is not known or else assumed to be simple. Sellars shows that if it is the case that the operation of entailment is a simple connection, then unreasonable results can be avoided only if the entities considered in the judgments are classified into four types. The system must be able to distinguish among and deal with expressions for kinds of things, kinds of circumstances,

something done to a thing, and something it does in return. The descriptive function " x is F at time t " obeys different logical rules depending upon which category is represented by " F ." Even the phrase "at time t " means something different in each case. On the other hand we can simplify the entities and put all the complexity into the notion of entailment, which must then contain specifiable logical complexities. These considerations are of greatest importance in deciding upon basic operations adequate to produce reasonable judgments.

Second, another form of counterfactual, the *counterfactual of noninterference*, states "'A' is true. Even if 'B' were also true 'A' would still be true." Although these are more complicated grammatically than the ordinary counterfactual, Reichenbach shows that the criteria for their validity require only notions of conditional probability. Hence they could be built into Uttley's machine with only minor modifications.

Third, Reichenbach shows that for many purposes we may use the ordinary adjunctive operations instead of the more complicated connective ones and not arrive at unreasonable consequences provided that we do not admit negations of lawful statements. When in this usage we wish to deny a connective implication, we must do so by saying "Do not use!" in the metalanguage. This is another sense in which negations may be absent in addition to the frequently remarked absence of negations of presentational symbols.

Fourth, Reichenbach discusses the important case in which some result follows lawfully from a factual statement. Although the result is a fact, and hence not nomological or lawful, it is, however, nomologically derivable from a fact, and it is termed *nomological relative to the fact*. In the interesting case we use an elliptic form of speech which omits mention of the law. If this can be validly done, the relative nomological statements are called *separable*. These account for the important kind of reasoning which uses implications for counterfactuals even though they do not directly represent laws of nature. The rules of manipulation of these statements predict correctly that what we regard as reasonable logical operations may vary with the extent of our knowledge. Since in practice we never draw the boundaries of the system under consideration so wide that all relevant information is explicitly included, the separable statements assume the greatest importance.

Fifth, Reichenbach employs the term *proper implication* to refer to a class of separable implications all taken relative to some convenient reference set of background information. A majority of our implications will be of this form. According to Reichenbach, the rules for the application of connective logical operations to these statements are so complicated that in general, in order to construct derivations one must use the simpler but less appropriate adjunctive calculus and then check the results to see whether they are reasonable. He concludes that the class of reasonable statements is thus not complete, in the sense that in order to construct derivative

relations between members of this class we have to go beyond the class. It would in this sense appear that not only on the lowest levels of the synthesis of meaningful symbols, but at the highest propositional levels, too, reasonable statements may be produced only by a process of rejection of a multitude of lawfully produced but unreasonable statements.

In summary, we have seen that present-day computers deal primarily with external relations among concepts which are given in a form that does not represent their "inner structure." In order for a machine to discover new patterns and to make inductive generalizations, it appears that the following notions must be taken into account.

The primary challenge to be met is that of forming impressions into logical elements. This involves Kant's notion of the synthesis of a connected manifold of impressions—that is, the individual element must contain the schemata for its connections with other elements. No conceptual product can exist in isolation from all other conceptual products; rather, each concept must contain a partial representation of many other concepts. Thus, as Kant first noted, spatial and causal relations are not derived from experience; experience is derived from them—they are prerequisites for experience of certain types. The consequence for computer design is that one must not seek to build a machine that perceives and conceptualizes and then inserts into its concepts independently achieved schemata of spatial and causal relations. On the contrary, in order to perceive at all it must perceive according to schemata, and the spatial and causal schemata must be an inherent part of the percepts and concepts themselves.

It appears that any activity of a computer rich enough in structure to deserve the term concept must be expressed in a language which is given meaning by a hierarchy of metalanguages whose initial members are simple enough to be organized rather directly around structures in the computer of a technological level which is perhaps within our present command.

Much of the activity in the earlier levels of this hierarchy would then consist in operations upon non-discursive symbols, and in their development from a state in which the basic computer structures are represented by a multitude of particular presentational symbols to a progressively articulated state amenable to the rules of discursive logic. In short, it seems that if a computer is to "think," its concepts will have to undergo the process of development that Freud [13] termed the primary process.

Finally, the logic of inferences and inductive generalizations made against a background of tacit premises and relevant information is quite different from ordinary propositional logic, but some of its rules are already known in a form which may be used at the present time in programming computers. A computer will probably have to arrive at reasonable inferences by selection from a larger set of inferences which are not all reasonable.

I hope that this selection of ideas from a mass of technical, logical, and philosophical investigations, and from certain psychological ideas not generally considered in connection with computers, may help provide a program for computer research, and that bringing this program to your attention may be of some use in adding to the present capabilities of computers and control mechanisms.

BIBLIOGRAPHY

Judgments and Concepts—Analytical Studies

- [1] E. Cassirer, "Language and Myth" (1925). Translation by S. K. Langer, Dover Publications, Inc., New York, N. Y.; 1946. See pp. 8-14, 23-28 (function of language).
- [2] E. Cassirer, "The Philosophy of Symbolic Forms, Vol. I. Language" (1923). "Vol. III. The Phenomenology of Knowledge" (1929). Translation by R. Manheim, Yale University Press, New Haven, Conn.; 1953, 1957. See vol. 1, pp. 13-15, 27, 50-52, 97-98, 101-102, 104-105, 278-283; vol. 3, pp. 13-15, 61, 64-65, 107-108, 114-117, 123-130, 139-141, 142-143, 150-154, 160-161, 164, 167, 169, 172-173, 176-179, 191, 202-203, 270-272, 287-288, 308-313.
- [3] E. Cassirer, "Substance and Function" (1910). Translation by W. C. Swabey and M. C. Swabey, Dover Publications, Inc., New York, N. Y.; 1953. See chap. 1 (abstraction); pt. 2, chap. 5 (induction and generalization).
- [4] P. Geach, "Mental Acts: Their Content and Their Objects." Routledge and Kegan Paul, London, Eng.; 1957. See chaps. 6-11 (abstraction).
- [5] I. Kant, "Critique of Pure Reason," 2nd ed. (1787). Translation by N. Kemp Smith, Macmillan Co., New York, N. Y. and London, Eng.; 1956. See pp. (B) 104-105, 122-124, 129-131, 136-138, 176-181 (connected manifold).
- [6] I. Kant, "Prolegomena to Every Future Metaphysics That May Be Presented as a Science," (1783). Selections, translation by C. J. Friedrich. From "The Philosophy of Kant," C. J. Friedrich, ed. Modern Library Inc., New York, N. Y.; 1949. See secs. 17, 20, 26-28, 30-31, 34 (the conditions of experience). (See sec. 3 of Introduction by Friedrich for Kant's terminology.) A lucid summary of Kant's methods.
- [7] S. Körner, "Kant," Penguin Publications, Harmondsworth, Middlesex, Eng., and Baltimore, Md.; 1955. See chap. 3, secs. 2, 3, 5; chap. 4, secs. 1, 6.
- [8] S. K. Langer, "Philosophy in a New Key," Mentor Press, New York, N. Y.; 1942. See pp. 75-80 (presentational symbols).
- [9] H. Lotze, "Logic" (1874). Translation by B. Bosanquet, 2nd ed., Clarendon Press, Oxford, Eng.; 1888. See book 1, chap. 1 (concepts).
- [10] H. Lotze, "Outlines of Logic" (1885). Translation by G. T. Ladd, Ginn and Co., Boston, Mass.; 1887. See pt. 1, chap. 1 (concepts).
- [11] J. O. Urmson, "Philosophical Analysis: Its Development Between the Two World Wars," Clarendon Press, Oxford, Eng.; 1958. See chaps. 2, 5 (logical atomism), 9, 10 (defects of logical atomism).
- [12] L. Wittgenstein, "Philosophical Investigations," Blackwell & Co., Oxford, Eng.; 1953. See secs. 72-74 (what is common to particulars), 81-86 (rules and conventions).

Thinking Mechanisms—Psychological Studies

- [13] S. Freud, "The Interpretation of Dreams" (1900-1930). Translation by J. Strachey, Basic Books Publishing Co., New York, N. Y.; 1954. See pp. 277-278, 279-281, 305-308, 310-314, 330, 339-340, 488-493 (representation of thoughts in non-discursive images), 536-542, 565-567, 593-597, 598-603, 610-611, 616-617 (genesis of thoughts and images).
- [14] J. Piaget, "The Construction of Reality in the Child" (1937). Translation by M. Cook, Basic Books Publishing Co., New York, N. Y.; 1954. See pp. vii-ix, 86-96, 209-214, 308-319, 350-364.
- [15] J. Piaget, "The Origins of Intelligence in Children" (1936). Translation by M. Cook, International University Press, New

York, N. Y.; 1952. See pp. v-vii, 32-35, 42-45, 122, 125-133, 137-143, 147-148, 150-152, 153-156, 188-189, 192-195, 210-211, 225-230, 236-240, 247-248, 253, 264, 294-305, 311-314, 322-327, 343-344, 351, 353-355, 357-419 (detailed study of the stages from automatic actions to manipulation of mental representations).

- [16] J. Piaget, "Principal factors determining intellectual evolution from childhood to adult life" (1937), in D. Rapaport, "Organization and Pathology of Thought," see ch. 6 of [18] below.
- [17] D. Rapaport, "The psychoanalytic theory of thinking" (1950), and "The conceptual model of psychoanalysis" (1951), in "Psychoanalytic Psychiatry and Psychology," R. P. Knight and C. R. Friedman, eds., International University Press, New York, N. Y.; 1954. See pp. 259-273, 221-247 (condensed summary of the views on thinking referred to in text).
- [18] D. Rapaport, ed., "Organization and Pathology of Thought: Selected Sources," Columbia University Press, New York, N. Y., 1951.
- [19] D. Rapaport, "Toward a theory of thinking," in D. Rapaport, "Organization and Pathology of Thought," see pt. 7 of [18].
- [20] P. Schilder, "Mind: Perception and Thought in Their Constructive Aspects," Columbia University Press, New York, N. Y.; 1942. See chaps. 1-4 (basic units of perception), 17-18 (memory and thinking).
- [21] P. Schilder, "On the development of thoughts" (1920), and "Studies concerning the psychology and symptomatology of general paresis" (1930), in D. Rapaport, "Organization and Pathology of Thought," see ch. 24, 25 of [18] (stages in the synthesis of a thought).
- [22] W. H. Thorpe, "Learning and Instinct in Animals," Methuen & Co., Ltd., London, Eng.; 1956. See chap. 2.
- [23] N. Tinbergen, "The hierarchical organization of nervous mechanisms underlying instinctive behavior," in "Physiological mechanisms of animal behaviour," *Symp. Soc. Exptl. Biol.*, vol. 4, pp. 304-312; 1950.
- [24] N. Tinbergen, "The Study of Instinct," Clarendon Press, Oxford, Eng.; 1951. See ch. 5 (hierarchy of stages of synthesis of an action).

Reasonable Inferences

- [25] N. Goodman, "Fact, Fiction and Forecast," Harvard University Press, Cambridge, Mass.; 1951.
- [26] H. Reichenbach, "Elements of Symbolic Logic," The Macmillan Company, New York, N. Y.; 1947.
- [27] H. Reichenbach, "Nomological Statements and Admissible Operations," North-Holland Publishing Co., Amsterdam, The Netherlands; 1954.
- [28] W. Sellars, "Counterfactuals, dispositions, and the causal modalities," in "Minnesota Studies in the Philosophy of Science," vol. 2, "Concepts, Theories, and the Mind-Body Problem," H. Feigl, M. Scriven, and G. Maxwell, eds., University of Minnesota Press, Minneapolis, Minn., pp. 225-308; 1958.

Semantic Analysis

- [29] R. Carnap, "Meaning and Necessity: A Study of Semantics and Modal Logic," University of Chicago Press, Chicago, Ill., 2nd ed.; 1956.
- [30] R. Carnap and Y. Bar-Hillel, "An outline of a theory of semantic information," Res. Lab., M.I.T., Tech. Rep. No. 247; 1953.
- [31] C. Morris, "Signs, Language and Behavior," Prentice-Hall Inc., New York, N. Y.; 1946.

Miscellaneous

- [32] N. Goodman, "The Structure of Appearance," Harvard University Press, Cambridge, Mass.; 1951. See p. 24 and passim.
- [33] D. O. Hebb, "The Organization of Behavior," John Wiley & Sons, Inc., New York, N. Y.; 1949.
- [34] G. Polya, "Mathematics and Plausible Reasoning, Vol. II. Patterns of Plausible Inference," Princeton University Press, Princeton, N. J.; 1954.
- [35] A. M. Uttley, "The Conditional Probability of Signals in the Nervous System," Radar Res. Establishment, Malvern, Worcestershire, Eng., Memo. No. 1109; 1955.
- [36] A. M. Uttley, "Conditional probability machines and conditioned reflexes," in "Automata Studies," C. E. Shannon and J. McCarthy, eds., Princeton University Press, Princeton, N. J., *Annals of Mathematics Study*, No. 34, pp. 253-275; 1956.

Automatic Data Processing in the Tactical Field Army

A. B. CRAWFORD†

THE use of digital computer techniques has become mandatory in the whole realm of ballistic missile design and guidance. And we are rapidly approaching the era in which computer techniques similarly might well become a necessity in many phases of ground combat. The feasibility of applying these techniques in a tactical warfare environment has been established, and the Department of the Army has a high priority program underway to place a prototype Automatic Data Processing System (ADPS) into its tactical organization within the next four years. This paper will explain the design objectives of this system and the progress to date in the applications area, and describe the extension of several techniques required to make the system function.

The technical design objectives may be summarized in about three key phrases—integrated, common-user service center, employing general-purpose hardware. The system in employment has been likened to the widespread tactical communication system, that is, a network of switching centers operated by specialists but tying together all users or customers, largely through the use of common-user trunks and switchboards. More specific design guidelines include the following:

- 1) Common-user facilities with a minimum number of single-user processors;
- 2) Integrated data files;
- 3) Simple foolproof input devices;
- 4) Interrogation-immediate reply;
- 5) GP, building-block hardware;
- 6) Automatic operation;
- 7) Flexible output.

The tactical ADPS will consist of several elements. (See Fig. 1.) Simple, foolproof input devices will be located at the source of data input. Connecting the inputs to the data-processing center and centers with each other is the Army's area communication system. According to the work load and applications being processed, each center will employ one or more general-purpose computers. Finally, a whole range of output devices is available according to the type of output desired, or rather required, by the commander to facilitate his making a sound decision—volatile but rapid video-display graphical or map-overlay form, or hard-copy reports.

† Automatic Data Processing Dept., U. S. Army Electronic Proving Ground, Fort Huachuca, Ariz.

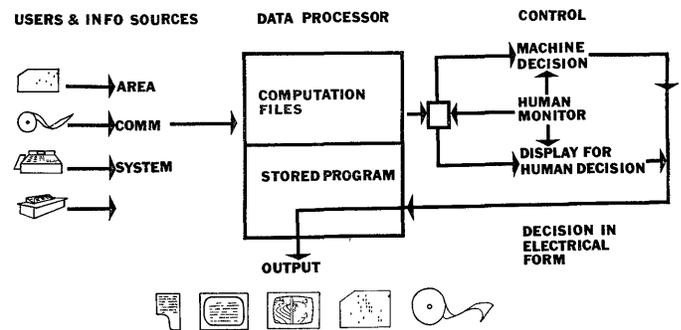


Fig. 1—The elements of the tactical ADP System are located in such a fashion as to minimize the burden on the users of System operation.

Functionally, the system is portrayed as an integration of subsystems according to the military staff function mechanized. The Army's traditional categorization is the well-known G-1, 2, 3, and 4 or Admin., Intelligence, Operations, and Logistics. In each of these categories, however, the uses of information are similar—that is, immediate reaction, longer range planning, and historical preparations. Our system design must be tailored to meet the rigorous specifications of high speed for the immediate reaction use as well as the capacity for large amounts of file maintenance and processing.

Looking at the over-all system configuration, one sees a far-flung complex of data processors and data transmission links, the magnitude of which is presently unique. As is well known, the tactical Army's command structure runs from the Company headquarters to Battle Group, Division, Corps, Army, etc. It appears desirable—and in fact a necessity—to install a data-processing capability within each headquarters from Battle Group back. The physical size and processing abilities will of course increase as we proceed toward the rear of the Combat zone. In fact, at Corps and Army more than one computer will be linked together to make up that data-processing center.

Now, everything described to this point might be called long-range planning. The title of the system discussed is ARMYDATA, and the general time-frame is beyond the immediate future. Nevertheless, we are guided in the immediate systems developments by these objectives. The remainder of this paper will describe the Army's project to attain a prototype ADP System as an intermediate step toward reaching the ultimate.

Slightly less than two years ago, a major project was launched within the Army to place into operation as soon as possible a system which could support the ever-

increasing military requirements for mobility and effectiveness in combat. Because of the long lead-time inherent in the R & D cycle for complex digital computers, the decision was made to parallel the development of the system's hardware and the system's concepts and applications. (The full description of the so-called FIELDATA family of equipment has been left to the companion paper prepared by Captain Luebbert of the Signal R & D Laboratories.)

At the time the decision was made to proceed on the hardware, a plan of action was laid out whereby the entire Army's methods for information handling in the field were to be scrutinized in great detail. The ultimate objective was a detailed description of each user's data-processing requirement with the more immediate by-product of streamlining the manual methods.

Those persons associated with business data-processing systems are familiar with the peculiarity of the systems problems involved in mechanizing integrated business procedures of any magnitude, and here I use the word "system" to mean "procedural" structure. We propose to integrate some 50 or more separate procedures (for as many customers) from concepts developed initially in an isolated fashion. That is, the logistics applications are being derived through systems analysis by the logistics specialists, the combat intelligence applications by G-2 specialists, and so on. To take full advantage of the great power of automatic data processing, we must mold or integrate these individual requirements into a realistic yet imaginative working system which meets the needs of the diverse staff requiring its services. The types of applications to be satisfied again can be broken down into the traditional areas of Operations, Intelligence, Logistics, and Admin. A different categorization according to data-processing techniques might be Combat Control (random access), Combat Support (batched processing), and Combat Computation.

Approximately twenty separate study groups are analyzing some seventy separate potential ADP applications. To assure that these studies will be more directly usable, an instructional booklet has been distributed setting forth a very rigid report format and content. The fact is that the area specialists are not experienced in the techniques of systems analysis, so we designed the booklet more or less to lead the study group "by the hand" through the steps of an ADP application study or systems analysis. Further, through contracts and the use of our own systems analysts, we are providing technical assistance to the study groups, particularly as they reach the phase of proposing an automated procedure for their problem.

Upon completion the individual study reports are subject to a technical review to determine feasibility, completeness, and the amount of dependence on other applications. Analysts return to the agency which con-

ducted the study to fill in details and to eliminate ambiguities. The object of the next step is to define the application to the detail required for programming and to derive certain workload data to be used in the over-all system design.

About this time, too, the preparation of digital models of the system and its parts is to begin. This, I believe, implies the underlying approach to our whole problem: simulation! We shall be utilizing all means and all levels of digital simulation techniques. First of all, interpretive simulation must be employed to permit the preparation and checking out of the computer programs prior to the delivery of the militarized computers. A successful simulation of the MOBIDIC on the IBM 709 has been completed, and so far we have put through successfully a few math subroutines, an intelligence filing and retrieval experiment, a combat surveillance target acquisition simulation, and a limited payroll problem. To extend this technique we are attempting now to develop a simulator generator which will enable us to experiment inexpensively when altering machine parameters or order codes. (This latter is an attempt to lay the groundwork for the later definition of the advanced system characteristics and computer designs.)

Before the detailed design of the prototype system is launched, a subset of the available application areas will be selected for implementation. Only then can we say specifically how comprehensive the FIELDATA system will be in covering tactical procedures. Since it is recognized that the hardware components are experimental in nature, in order to meet a tight schedule without a crash program the FIELDATA system will itself be experimental in nature.

Parallel to the detailed programming of each application will be the use of a digital model of the system information flow to permit a prediction of the needed data rates, alternate routing, and potential bottlenecks. A model is being used now using the 709 which simulates a general army area communications complex. Probability distributions and Monte Carlo techniques are employed throughout, from the preparation of a message entering the system to human switchboard operator actions and to message processing as it progresses toward its destination. A statistical analysis routine produces model-run data reduction and also recommends a sample size according to our level of confidence desired.

To explain the next type of simulation, first I shall outline the organization of the ADPS Test Facilities at the Proving Ground. Computer test facility is around a large-scale computer center, specifically the IBM 709. Herein all of the simulation work will be conducted, and herein will be the means for conducting controlled environmental tests for predicting the validity of our proposed computer procedures. The field-test facilities

are for the purpose implied by the name and will naturally be employed to try to prove out the simulation and model results. Even during these field-test phases we are to utilize simulation techniques. In this context the 709 will be linked to that part of the system being subjected to evaluation in a field operational environment; a single thread employment of equipment will be supplemented by simulation of the remainder of the system. (See Fig. 2.) That is, the computer will provide the data sink to introduce input into the system and to absorb output from these echelons actually being operated.

The Computer Center became operational in February, 1959. The larger part of the application studies are completed, and detailed analysis has begun on several of them with demonstration runs already made on at least two. The major hardware items of the prototype system are on order, with the first to be delivered this coming Fall. Combining this progress with that reported by Captain Luebbert on hardware, transmission, and programming aids we remain confident that our objectives can be attained.

In summary, this paper has reported on an ambitious and futuristic program undertaken by the Signal Corps to provide the Army with a vast tactical Automatic

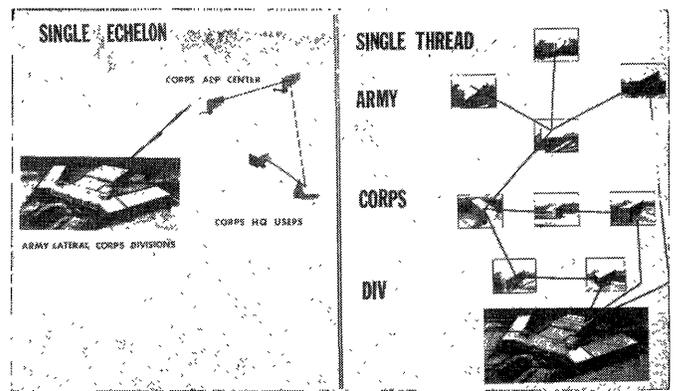


Fig. 2—The IBM 709 will serve as a source and destination of system input and output by simulating the missing echelons during field tests.

Data Processing System. The proposed system in prototype form is to be operational by 1963 and will incorporate the very latest developments in digital techniques, *i.e.*, new miniaturized general-purpose data-processing devices, computer-to-computer communications, and automatic programming. The research efforts in this project, and certain standards derived, are bound to have an effect on and contribute to related commercial data-processing activities.

Data Transmission Equipment Concepts for FIELDATA

W. F. LUEBBERT†

FIELDATA is an integrated family of data processing and data transmission equipment being developed for Army use. A unique feature of this family is the almost complete disappearance of conventional distinctions between communications and data processing. This paper deals primarily with the concepts and techniques developed to create this evolutionary merger emphasizing the ways in which conventional communications concepts have been adapted to achieve a high degree of interoperability with computers and other data processing equipment, and an extraordinary degree of flexibility and adaptability of application.

In order to explain and illustrate the FIELDATA concepts, this paper makes extensive use of specific examples of design decisions, particularly those dealing with common features such as codes, voltage and impedance levels, data rates, etc. Among the equipments

of the FIELDATA family developed in accordance with these concepts and common standards are the following, all of which are scheduled for completion prior to the end of 1960: the MOBIDIC computer (Sylvania), the BASICPAC and LOGICPAC computers (Philco), the AN/TSQ-35 19,200 bit/second data transmission equipment (Bendix-Pacific), the AN/TSQ-33 2400 bit/second data transmission equipment (Collins), the AN/TSQ-32 1200 bit/second data transmission equipment (Stelma), the DATA COORDINATOR, a facilities coordination and control equipment for an integrated communications and data processing system (IBM), and a host of miscellaneous equipments such as magnetic tape transports (Ampex), a flexowriter-like electric typewriter (Smith-Corona), high-speed printers (Anderson-Nichols), security equipment (Collins), etc.

The fundamental capabilities of data processing equipment can be described as the ability to transform

† U. S. Army Signal Res. and Dev. Lab., Fort Monmouth, N. J.

information into more desirable or useful forms. Using the same viewpoint, the fundamental capability of conventional communications equipment can be described as the ability to transfer information to a more desirable or useful location. Of course, from an abstract logical viewpoint a transfer is merely one special kind of transformation, one in which only physical location is changed. Combining these descriptions, one is led to the concept of a generalized data handling system capable of performing generalized transformations, of which conventional one-location data processing would be one special case and conventional no-processing data transmission would be another special case. In such a system there would be no fundamental distinctions between data processing and data transmission, but only distinctions of convenience based upon application, use, and design emphasis.

Is such a concept a reasonable one, and does it have any practical utility? The answer to the first part of this question will be examined in detail in this paper; the answer to the second part will be determined by the success in actual use of the equipment and system concepts it generates, the concepts explained in this paper which are receiving initial implementation in the FIELDATA family of equipments.

An examination of the information flow and manipulation in typical data processing and data transmission equipments shows almost immediately that there is a very considerable mixture of functions going on in both types of equipment. A very considerable amount of the activity going on inside any computer or data processor consists of simple transfers of information from one part of the processor to another. Within data processing equipment a major part of the activity is concerned with the generation, manipulation, and other processing of information used for control, supervisory, and error reduction purposes. In many cases nearly identical operations go on in both data processing and data transmission equipments, with the differences, if any, being matters of design emphasis based upon application and use.

Many practical cases of this similarity are immediately obvious, particularly in the area of devices used for data entry and output. For example, computers frequently use paper tape readers similar to those used in teletypewriter transmitter distributors, and paper tape punches that could easily be used in teletypewriter reperforators. Similarly the idea that kinds of input-output devices such as card readers and punches and magnetic tape transports which are widely used for data processing can effectively be adapted for use of communications lines is also being exploited in equipments such as the IBM transceiver, the Collins Kinetape, etc.

FIELDATA emphasizes this kind of exploitation to the extreme, particularly encouraging it by assuring that interconnection of semiautonomous equipment modules be made in accord with common standards without distinction whether these equipments are conceived primarily for computer-associated or transmis-

sion-associated functions. This makes it possible for the same data terminals to operate not only with data processing-type inputs and outputs such as computers, paper tape, magnetic tape, and IBM cards; but also for it to operate with real time weapons system data and with telegraphic data.

Control circuitry is so devised that pure binary as well as alphanumeric (*alphabetic-numeric*) data may be handled. Since any digital code, be it Baudot (teletypewriter) code, Holerith (IBM card) code, or any of a wide variety of computer codes may be represented in binary bit-by-bit form, the FIELDATA devices have the potential of transmitting or handling any type of digital data. Thus, they could be used with digitized voice, digitized facsimile, or other types of digitized analog signals.

The use of common standards, codes, and standard data rates makes possible the kind of data transmission equipment concept shown in Fig. 1. This concept leads naturally to a division of the subassemblies of data transmission equipment into three kinds:

- 1) *Input-output transducers* are devices for converting information from some human or machine usable form such as paper tape, magnetic tape, punched cards, analog electrical voltages, strokes on a keyboard, etc., into digital form.
- 2) *Transmission transducers* are devices for converting data in digital form into appropriate signals for transmission over radio, wire or other kinds of propagation media.
- 3) *Embolical¹ equipment*, normally inserted between input-output transducers and transmission transducers, is used primarily to perform control and supervisory functions, error detection and/or correction, buffering, and/or speed conversion, code conversion, or encryption necessary for proper system operation of the data transmission equipment. The functions of embolical equipment are information processing functions. Inputs and outputs will both be digital in form, although supplementary analog information may also be available particularly in some kinds of error control schemes. A general-purpose computer is potentially a very powerful and flexible type of embolical equipment, but the necessary functions can often be performed much more economically by specialized equipment.

This division of subassemblies may be a physical division into separate items of equipment, into semi-autonomous parts of a single equipment (either in a separate box or in a single box) or it may be merely conceptual with no physical implementation.

¹ Embolical is a coined term from the Greek *embolesimos* meaning to put between or insert. This word is also used in medical, astronomic, and ecclesiastic literature to describe other specific kinds of intercalations.

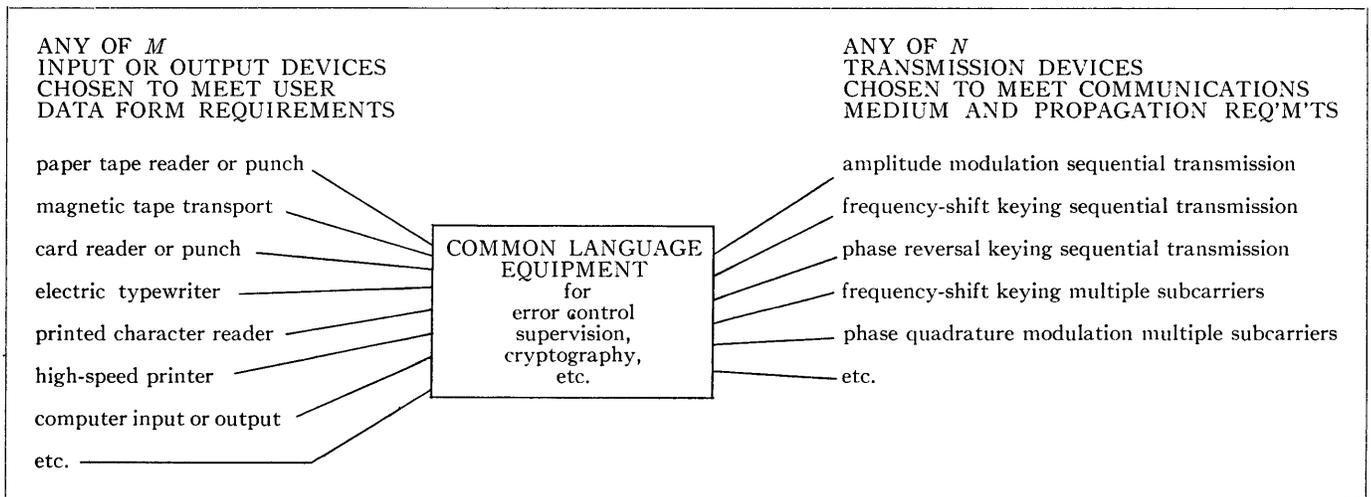


Fig. 1—FIELDATA equipment modularity concept.

Obviously maximum flexibility and adaptability can be achieved if any input-output transducer can operate with any transmission transducer. This permits one to tailor equipment to meet the requirements of a particular situation. It permits one to choose the in-out device suitable for the user's most convenient data form (paper tape, magnetic tape, IBM cards, etc.) independently of the nature of the transmission facility available. One can then choose the transmission transducer on the basis of the transmission medium used (VHF radio relay, HF radio, loaded cable, carrier transmission on wire, etc.) independently of the user's data form. Then join the two together taking advantage of common standards of interconnection or intercommunication between modules to create a well-tailored combination.

The question then arises, "Why embolic equipment?" Certainly by proper choice of common interconnection or intercommunications characteristics one can minimize requirements for code conversion, buffering, speed conversion, etc. Ideally one should be able to join the input-output transducer to the transmission transducer without embolic equipment, so why have it? The answer is that there are three important functions in a data terminal which seem convenient to separate from both in-out transducers and transmission transducers: 1) communications supervision, 2) error control, 3) cryptographic security.

Supervisory functions are conveniently separated from the input-output transducers and the transmission transducers because supervisory requirements may be strongly influenced by both. Error control requirements may also be strongly influenced by both the in-out data and the transmission situation, and may even in some cases not be desired at all. Thus it is best treated as a module which can be tailored to the paired requirements of in-out and transmission or completely omitted. Cryptographic security is conveniently a separate module so that it may be omitted in those cases where security is not required or desired.

In order to discuss supervisory activities conveniently it is desirable to make a distinction between two kinds of information which flow through a communications system:

- 1) *Primary information* is that which a user wishes transferred to another location, that is, the information to be communicated.
- 2) *Secondary information* is added to the primary information either by the originator or by equipment of the communications system which is used to perform functions of supervision, routing, error control and related activities necessary or desirable to permit the primary data to be effectively communicated. This information is used by communications equipment and personnel and is normally of no use or interest to the ultimate recipient of the primary information.

A basic requirement for maximum flexibility and adaptability is that the user, who enters data at an input device and receives data at an output device, need not be required to exercise judgment or knowledge, to perform special activities, to use or to interpret secondary information. This, in turn, makes it desirable to isolate secondary information from the input-output devices, which are the user's point of contact with the communications system. A convenient way of establishing and enforcing this isolation is the creation of a distinct embolic module which generates, receives, interprets, and/or acts upon secondary information and passes on action requirements derived from this secondary information over local control lines to its associated input-output transducer and transmission transducer. It is important to note that such equipment need not have any ability to interpret or act upon primary data, and thus its operation can be made completely independent of the coding used for primary data so long as there is a unique method of distinguishing primary from secondary data.

In the FIELDATA family of equipments the segregation of primary and secondary data is accomplished in a particularly simple way by providing separate transmission symbol "alphabets" for primary and secondary data. The basic unit of data in FIELDATA codes is a six-bit character. For primary data this may be merely a six-bit block with no specific meaning assigned to specific bits within it, and no restrictions on the permissible code combinations which may be transmitted. If this information is alphanumeric in form so that specific bit meanings must be assigned for electric typewriters, printers and similar input-output transducers to operate, then the FIELDATA alphanumeric code given in Appendix I is used. A similar but distinctly different six-bit FIELDATA supervisory data code has also been created and is outlined in Appendix II. Message heading and addresses, dialing information, multiplexing information, signals indicating start and end of message blocks, and control signals are all created using this alphabet.

In many situations, for example inside computers, it is desirable to both the six-bit primary data alphabet and the six-bit FIELDATA supervisory code alphabet with implicit differentiation between them similar to the implicit differentiation between data and instructions in the computer. However, other situations arise where it is desirable to provide an explicit identifying tag to specify which alphabet is being used.

A basic tagging method has been adopted for use on interconnecting cables employed for joining equipment modules. A seventh or tag bit is added to each six-bit symbol: a binary "one" if the symbol is primary data, and a binary "zero" if the symbol is from the FIELDATA supervisory alphabet. In the interconnecting cables an eighth bit in the form of an odd parity bit is also added to provide error protective redundancy. This basic eight-bit form creates the appearance of an eight-bit code. It has widespread use, but cases do exist where other alphabet tagging techniques and other redundancy is preferable. For example, serious complications would arise if this basic eight-bit form were used for the actual punching of FIELDATA information on paper tape. This occurs because control difficulties occur with a paper tape punch-reader system when primary data symbols are allowed to use either the blank tape condition (no channels punched) or the deleted tape condition (all channels punched). Since the use of the basic eight-bit form inevitably results in one or the other of these symbols being a primary data symbol depending upon whether a hole is interpreted as a binary "zero" or as a binary "one," a slightly different tagging and parity scheme must be adopted. This, of course, results in another eight-bit form which could be interpreted as an eight-bit code. Other tagging and redundancy schemes could lead to other apparent codes. Thus, there is no such thing as a *unique* eight-bit FIELDATA code, although it may be convenient to think of the basic eight-bit form as an eight-bit code.

In some situations transmission errors may create important reasons for using other tagging and error protective redundancy schemes. Errors which cause loss or change in supervisory information such as dialing information, message headings, start and end of message indications, and so forth, may completely disrupt the proper functioning of a communications system. Thus they may require protective redundancy many times more powerful than the simple parity used in the basic eight-bit form. However, compared to these secondary data, primary data may be capable of tolerating considerably higher error rates. For example, if the data are English text the inherent redundancy of the language may permit significant corruption without loss of intelligibility. Numerical data may consist of successive observations of a physical phenomenon in such a form that inconsistent data may be deleted and ignored, or they may be protected by numerical checks similar to those used by accounting systems to detect bookkeeping errors. In a situation where error rates were severe it might not be desirable to apply as powerful error control to these less demanding primary data as to secondary data; thus one might desire to use methods of differentiating between primary and secondary data which are more resistive to corruption by errors than the simple tags used in the basic eight-bit form together with different error control schemes for primary and secondary data.

This kind of differentiation illustrates a fundamental assumption about error control which is part of the FIELDATA concepts. Specific error control requirements should be determined by the nature of the data and their use. As mentioned above there may be cases where primary data can tolerate frequent errors with little loss of usefulness, but there are other cases, such as the transmission of computer programs, where very low error rates are required. In some cases if errors above the desired maximum rate occur and are detected, the erroneous information may be deleted without significant harm; in other cases they must be corrected. In some cases where correction is required it may be deferred and handled by a service message; in other cases it must be corrected before the data are released, and so forth.

Notice that all these requirements are determined by the use of the data, and that these demands remain the same regardless of the transmission path and types of transmission equipment through which the data may be required to flow. However, the occurrence of errors is anything but independent of transmission factors. Although errors do occur in input-output devices and embolic equipment, by far the most variable and difficult to control errors normally occur in the transmission portion of a data link. These errors are often quite variable in frequency and interrelated in occurrence. They are quite strongly dependent upon the nature of the propagation medium and the kinds of disturbances to which it is subject. For example, an HF radio link sub-

ject to severe fading and multipath disturbances could hardly be expected to have the same error problems as a wire and cable link subject to intracable crosstalk and impulse-type switching noise. Furthermore, for a given propagation medium and kind of noise and disturbances, the frequency and interrelationships of errors are strongly dependent upon the characteristics of the transmission transducers used. For example, if amplitude modulation is used one would expect different errors than if frequency shift keying were used; if sequential transmission of short bauds on a single subcarrier is used one would expect different error problems than if parallel transmission of long bauds on multiple subcarriers is used; and if sampling or nonintegrating types of detection are used one would expect different error problems than if full integrating detection schemes were used.

The number, variability, and difficulty of measurement and analysis of the various factors which contribute to the frequency and interrelationships of transmission errors or particular circuits is staggering. At the present time the state of the art is such that only crude estimates of frequency of error can be made for typical equipments when exposed to disturbances other than Gaussian noise, and that practically nothing can be estimated in advance about the interrelationships of errors under practical conditions of impulse noise, crosstalk, propagation variations, etc. This is particularly unfortunate because the effectiveness of the various digital error control schemes available is strongly dependent upon the interrelationships of errors. For example, a simple parity check is capable of detecting single errors but not double errors. If errors occur randomly, double errors will seldom occur and this very simple check will be quite powerful. Thus if the bit error rate is 10^{-4} , a simple parity check will reduce the undetected error rate to 10^{-8} . On the other hand, if errors tend to be clustered a parity check will be rather ineffective. Thus if the conditional probability of a second error immediately following the first is 0.5 and the bit error rate is 10^{-4} , then a parity check will reduce the undetected error rate to only 0.5×10^{-4} . Given knowledge of the interrelations, checks can be designed which give high protection with a minimum amount of checking equipment. Unfortunately this knowledge is usually unavailable.

If the most important sources of errors are in a data communications link, is it proper to incorporate the major error control features of the link into the transmission transducers? The FIELDATA concepts answer this question with a resounding NO! Why? The key reason is that while the *occurrence* of errors and the raw error rate and characteristics are determined primarily by transmission factors, the error *requirements* are determined by the use of the data. The means and techniques of error control appropriate to a particular situation obviously depend upon the *interaction* of these factors. However, it is a fundamental modularity principle of

FIELDATA that the characteristics of transmission transducers should as nearly as possible be independent of the details of user input-output characteristics and data employment.

If one were to incorporate the error control features into the transmission transducers and one desired to provide p classes of controlled error service to users with q modulator/demodulator assemblies, then one would require p times q types of complete transmission transducers. The obvious answer is to separate modularly, making the error control module an item of embolic equipment. This allows one advantage of similarities in requirements and raw error characteristics among the different situations to reduce the variety of equipment to be constructed.

In view of the present difficulties in predetermining the specific error characteristics of transmission transducers prior to construction and test, it also permits construction of new transmission transducers and their use with existing error control embolic equipments until the specific error characteristics of the transducer can be measured and new error control embolic equipment designed if necessary to meet user requirements with the measured transmission error characteristics.

In addition to these practical advantages of placing error control responsibilities in embolic equipment modules rather than in transmission transducer modules, there are conceptual advantages associated with maintaining the simplest possible information flow patterns and division of activities among the three basic kinds of assemblies.

In general transmission, transducers pay no attention to the information content of the digital information they convert to modulated transmission form, neither knowing nor caring whether the data are primary or secondary, whether they are redundant or irredundant, or what code or codes they use. In contrast to this, embolic equipments normally act as information processing devices. Thus, in supervising a transmission link embolic supervisory equipments act on sensory information received from transmission transducers and in-out transducers and generate, process, and interpret secondary supervisory information to control the over-all operation of the communications link. The error control problem is exactly parallel. Acting on information about user requirements from the in-out transducer side, and information and sensory information about transmission errors from the transmission transducer side, error control equipment is required to generate, process, and interpret error control information using it to control (often via supervisory operations) the over-all operation of the communications link in such a way as to control its errors. Thus it is obvious that from the information processing viewpoint the performance of error control as an embolic function has a close parallel to other embolic functions and is distinctly different from the functions otherwise performed by transmission transducers.

Some of the FIELDATA equipments being developed in implementation of these concepts are listed in Table I.

The transmission transducers of the FIELDATA family are limited in number. However, the choice of the 75×2^n pattern of data rates permits widespread augmentation by minor modification of existing or developmental teletypewriter multiplexed transmission equipments. In addition, future expansion is simplified by the fact that future equipments will be able to utilize the same embolic and input-output equipments, and will thus be cheaper to develop than data transmission equipments which require development of embolic and/or input-output equipment as part of the same package. Expected future expansion will place greater emphasis on transducers for radio circuits.

TABLE I

Transmission Transducers		
AN/TSQ-32	600-1200 bit/second	Stelma Corp.
AN/TSQ-33	600-2400 bit/second	Collins Radio
AN/TSQ-35	19,200 bit/second	Bendix-Pacific
General Purpose Computers		
MOBIDIC		Sylvania
BASICPAC/LOGICPAC		Philco
INFORMER/DATA COORDINATOR		IBM
In-Out Transducers		
Electric Typewriter		Smith-Corona
Paper Tape Reader		Smith-Corona
Paper Tape Punch		Smith-Corona
High-Speed Printer		Anderson Nichols
Paper Tape Transport		Ampex Corp.
Magnetic Tape Transport		Ampex Corp.
Tacden		Aeronutronics Systems
Special Embolic Equipment		
CV-689	Cryptosecurity Adaptor	Collins Radio
CV-690	Control Equipment	Collins Radio
CV-691	Data Concentrator	Collins Radio

The AN/TSQ-32 is a transmission transducer capable of accepting data from a standard FIELDATA connection and of transmitting it serially as frequency-shift modulation of a single subcarrier over a standard voice channel. Transmission rates are 1200 bits/second (1500 words per minute) over good quality circuits, and 600 bits/second over poorer circuits.

The AN/TSQ-33 is a transmission transducer capable of accepting data from a standard FIELDATA connection and of transmitting it as 8 channels as synchronous phase quadrature modulation of four subcarriers over a standard voice channel. Transmission rates are 2400 bits/second (3000 words per minute) over good quality circuits, with 1200 and 600 bits/second available for use over poorer circuits. This equipment is essentially a militarized, miniaturized, FIELDATA compatible version of the Kineplex TE-206.

The AN/TSQ-35 is a transmission transducer capable of accepting data from a standard FIELDATA connection and of transmitting it by amplitude modulation of

8 subcarriers orthogonally spaced in frequency and time. The transmission baseband is the 48-kc band between 12 and 60 kc used for military and commercial cable and carrier circuits. Representative circuits are the Army spiral-four and associated AN/TCC-7, 8, and 11 equipments; and commercial types N and K carrier equipments. In addition to point-to-point full duplex operation, this equipment has special features for multiple station "common net" round-robin type operation.

In addition to these transmission transducers specifically designed for FIELDATA a wide variety of existing military and commercial teletypewriter transmission equipments can be used for FIELDATA transmission with only minor modification. Examples of such equipment are the AN/FGC-54 capable of transmission and diversity reception of FIELDATA information at 2400 bits/second over long-haul 3-kc radio circuits using 32 channels each operating at 75 bits/second. Another example is the AN/FGC-29 potentially capable of transmission and diversity reception FIELDATA information at 1200 bits/second using 16 channels each operating at 75 bits/second. Yet another example is the AN/TCC-30 potentially capable of transmitting 1200 bits/second using 16 channels operating at 75 bits/second.

The FIELDATA computers form the most complete and well-balanced portion of the FIELDATA family.

The FIELDATA computers may act as either input-output transducers or as embolic equipments for data transmission purposes, having the ability to accept, process, and emit either primary or secondary data. All are designed for direct operation with data transmission equipment. In their employment the MOBIDIC, which was the first machine to have its characteristics frozen, is the least capable of transmission of machine data; it has serious restrictions on its use of the supervisory code functions. The most flexible in its employment of data transmission will be the DATA COORDINATOR, a newer equipment which will have the capability of terminating a large number of data transmission circuits simultaneously and which will have a number of special capabilities and console positions to facilitate its use as a facilities coordination processor for an integrated communications/data processing system.

All the FIELDATA computers are general-purpose processors of modular design and great flexibility. Their relative computation speeds are shown on the bar chart. All are designed for field use with operation and maintenance simple enough for field personnel. The largest, MOBIDIC, mounts in a semitrailer van, while the others mount in shelters which can be carried on a truck. It is interesting to note that the minimum assembly which forms a fully operational stored program computer of the BASICPAC/LOGICPAC type or the INFORMER/DATA COORDINATOR type weighs in either case less than 150 pounds subdividable into 50-pound or smaller packages. However, these central processors are normally used for data processing purposes

in vehicular assemblies which far exceed these weights because of auxiliary equipments such as multiple tape transports, special real-time input-output units, additional magnetic core memory modules, data transmission equipment, input-output converters, etc. For example, a vehicular BASICPAC would augment the central processor with four magnetic tape transports, a high-speed paper tape reader and punch, an AN/TSQ-33 transmission transducer, and other auxiliary equipment. Fig. 2 indicates the relative computational capabilities of the various FIELDATA processors.

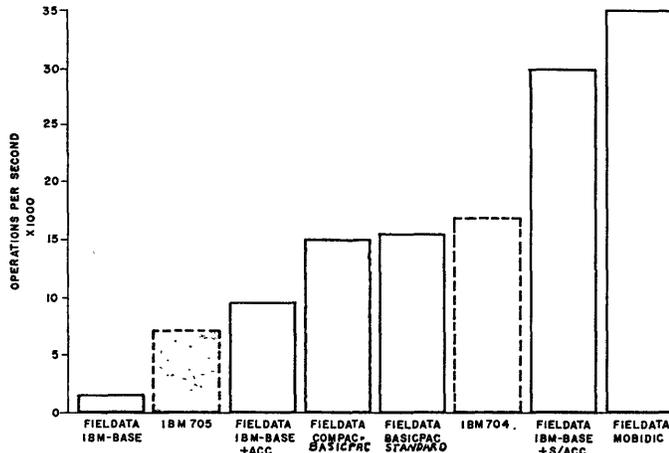


Fig. 2—Speeds of computation, FIELDATA processors.

The input-output transducers of the FIELDATA family serve double duty as computer input-output devices and as data transmission input-output devices. The group under current development constitutes a minimum group of general usage items, a number of which are in only partially militarized form. This minimum group will be augmented by future field teletypewriter equipments which will utilize FIELDATA code, and by advanced equipments now under study to provide specialized input-output capabilities. Since the items are mostly quite conventional, detailed descriptions are omitted here in order to save space.

The specialized embolic equipments in FIELDATA provide cryptographic security, interconnection of input-output and transmission transducers, and tie-in of FIELDATA circuits to existing teletypewriter circuits.

Three major items of special embolic equipment are being developed for FIELDATA. The CV-689 is a special cryptosecurity adaptor which permits an existing type of security equipment to be inserted just before the transmission transducer in any FIELDATA data transmission assembly, thus providing cryptographic security.

The CV-690 is a device which provides supervisory control, error control, and synchronizing buffer facilities for connecting paper tape or magnetic tape units to the AN/TSQ-32 or 33. Although future plans call for similar special militarized embolic equipments for other kinds of input-output equipment such as card equip-

ment none is now under development. However, rather minor modifications of commercial Collins Kinocard equipment will permit nontactical employment of AN/TSQ-33 equipment for card transmission, and at least some versions of the MOBIDIC will include card equipment which, through the computer, can reach transmission facilities.

Although FIELDATA concepts make provision for a wide variety of error control and supervisory control systems only the particular system used in the CV-690 will actually be used initially except for computer-to-computer transmission, since other embolic devices using different error control or supervisory control schemes are now under development. It is expected that when experiments determine the actual frequency and interrelationship of errors for particular transmission transducers, more effective schemes will be devised. However, the very simple and easy to implement two-dimensional (interlaced) parity error detection scheme followed by request-back or rerun-type error correction used in the CV-690 is particularly easy to implement, and is expected to suffice for initial testing.

The CV-691 Data Concentrator is a device designed to bridge the gap between existing large-scale 60- and 100-wpm teletypewriter facilities and FIELDATA equipment. Although all FIELDATA processors have normal provision for a paper tape reader which accepts teletypewriter tape as well as one for FIELDATA tape, there exists a significant need, especially at locations where FIELDATA processors might not be available, to accept multiple channels of teletypewriter information and convert it into FIELDATA form to take advantage of FIELDATA transmission transducers and error control, permit recording on FIELDATA magnetic or paper tape, permit printing on FIELDATA high-speed printers, simplify entry into FIELDATA computers, etc. The CV-691 accepts up to 25 (or 50) teletypewriter inputs, stores the information in a buffer core memory (made up of the same memory planes as used in MOBIDIC), assembles it into message blocks, converts it into FIELDATA form, applies the same error and supervisory control as the CV-690, and emits the data in FIELDATA form at rates up to 2400 bits/second (3000 words per minute). The receive side performs the inverse functions.

It is expected that as more and more of the voice and other analog communications systems convert to pulse code modulation and other digital forms that additional types of FIELDATA embolic equipment will be required to perform the error control, supervisory functions, and buffering/synchronization necessary to tie input-output transducers to their digital bit streams.

The FIELDATA family is an attempt to create an integrated family of data transmission equipments to meet Army needs. Though lacking many of the features and equipments of an ideal family of data transmission equipments, it will make available in experimental quantities by the end of 1960 the first integrated family

of equipments for experimental establishment of a truly integrated communications/data processing system. It will be the first system in which data processing and communications equipment both utilize the same input-output and storage devices, the same voltages, impedance levels, codes, and other common interconnection characteristics, and in which the equipments are so designed that in many cases the only way to determine whether a device is used for communications or data processing is to look at its specific application in the system.

APPENDIX I

FIELDATA ALPHANUMERIC CODE

A key step in achieving the required compatibility between the various elements of an automatic data system is the adoption of a common "language" for the storage and transmission of data throughout the system. The basic 6-bit alphanumeric code for use in this family shall consist of 2 indicator bits and 4 detail bits. The pattern of character assignment for the code is as follows with the 2 indicator bits determining the choice of column and the 4 detail bits determining choice of row in Table II.

APPENDIX II

FIELDATA SUPERVISORY CODE

The FIELDATA supervisory code is used for message headings, dialing, multiplex identification, supervisory control, and other activities associated with secondary data. This code is similar to the FIELDATA alphanumeric code used for primary data. It also consists of 2 indicator bits and 4 detail bits. The pattern of control assignment is as follows with the 2 indicator bits determining the choice of column and the 4 detail bits determining the choice of row in Table III. When it is not necessary to provide alphabetic supervisory information only the latter two columns are used. In this case when the basic 8-bit FIELDATA form is used, an OR of the first indicator bit and the tag bit will provide clocking for the 96 legitimate characters of the 8-bit form.

TABLE II

	00 (Upper and Lower Case)	01 (Upper and Lower Case)	10 Upper Case	11 Lower Case
0000	Master Space	»)	0
0001	Upper Case	~	-	1
0010	Lower Case	×	+	2
0011	Tab.	◇	<	3
0100	Car. Ret.	<	=	4
0101	Space	△	>	5
0110	A		-	6
0111	B		\$	7
1000	C		*	8
1001	D		(9
1010	E		U	,
1011	F		V	;
1100	G		W	/
1101	H		X	.
1110	I		Y	'
1111	J		Z	Special <input type="checkbox"/>
			Stop ⊕	Back Space

TABLE III

	00	01	10	11
0000	BLANK/IDLE	k	Dial 0	Ready to Transmit
0001	Control Upper Case	l	Dial 1	Ready to Receive
0010	Control Lower Case	m	Dial 2	Not Ready to Receive
0011	Control Tab.	n	Dial 3	End of Blockette
0100	Control Carriage Ret.	o	Dial 4	End of Block
0101	Control Space	p	Dial 5	End of File
0110	a	q	Dial 6	End of Control Block
0111	b	r	Dial 7	Acknowledge Receipt
1000	c	s	Dial 8	Repeat Block
1001	d	t	Dial 9	Spare
1010	e	u	Start of Control Block	Interpret Sign
1011	f	v	Start of Block	Non-Interpret Sign
1100	g	w	Spare	Control Word Follows
1101	h	x	Spare	S.A.C.
1110	i	y	Spare	Special Character
1111	j	z	Spare	Delete

A High-Accuracy, Real-Time Digital Computer for Use in Continuous Control Systems*

W. J. MILAN-KAMSKI†

IT HAS become evident during the last few years that the accuracy requirements of analog computers have become too difficult to be easily satisfied. The rising pressure to achieve better computational accuracy has led to significant improvements in the computational techniques used in analog computers. These new improvements have made it possible to achieve a high degree of precision so that a 0.1 per cent accuracy has gradually become a realistic figure in many analog machines.

However, present-day analog computer technology is completely helpless if accuracy requirements approach the magnitude of 1 part per million, or 0.0001 per cent. The only available computers which can achieve this degree of accuracy are obviously digital computers.

Many attempts have been made to design digital computers so that they might be used as direct replacements for analog computers. However, a rather unexpected difficulty has arisen. Digital computers, which have received a great deal of publicity as being the fastest computational tools, are extremely slow when compared to analog computers. Since the comparison is made between digital and analog computers, the operation of the digital computer must be such as to satisfy the bandwidth requirements of the analog computer. By this equivalence, the bandwidth of a digital computer can be defined as the bandwidth of an equivalent analog computer.

There are three distinct approaches in solving the problem of designing high-accuracy, real-time digital computers. All three of these approaches are directed toward building high-accuracy digital computers which can replace analog computers in applications where accuracy requirements exceed present capabilities of these machines.

At least one approach has come from engineers whose experience and background have been chiefly in the field of analog computers. Their basic approach was to replace various analog computer elements by equivalent digital operational blocks. For example, an integrator which consists of a motor with appropriate velocity control can be replaced by a reversible counter; a potentiometric multiplier can be replaced by a digital element which is called a rate multiplier, and so on.

Since the operation of a computer of this type is incremental, its design approach led to the development of a family of computers called incremental digital computers.

The second approach was to translate the problem into a differential equation and then to solve the differential equations by integration. Since the solution of differential equations is done using finite increments, the family of digital differential analyzers is closely related to the family of incremental computers. The output function of incremental computers and of the digital differential analyzers is determined by the increment of the input function and by the internal state of the machine. These computers, therefore, can be regarded as deterministic transducers with infinite memory.

The third family of real-time digital computers is represented by machines which go through a complete computational cycle every time a new input sample is taken. These computers normally adopt computational techniques which have been developed in programming general-purpose digital computers.

These machines normally have short memories or, in many cases, no memory at all. Their output is always uniquely determined by the input.

The latter group of computers is particularly suited to applications in which a number of problems must be solved simultaneously and concurrently. It is achieved usually by interleaving several programs.

The computational speed of digital computers is usually defined as the number of additions or multiplications which the computer can perform within a certain period of time. This computational speed is extremely high when compared to the computational speed of a desk calculator. In real-time computation, however, the speed of operation is defined as the ability of the computer to generate output functions, which vary rapidly with time. Not only must the output function contain large values of higher order derivatives, but also must not be delayed by the finite computational time of the computer. The transfer function of real-time computers is often complicated and usually contains trigonometric functions. If a high degree of accuracy is desired, the word length required may be as large as 30 binary digits or more.

It is possible to show that a high-accuracy machine has a limited ability to generate output functions which contain large values of output function derivatives. The computational time increases very rapidly as the word length increases.

* Presently being developed under a subcontract from the Military Products Dept. of Detroit Controls, in Norwood, Mass., for the U. S. Navy.

† Epsco, Inc., Boston, Mass.

The design of a real-time digital computer is usually based on an input-output accuracy specification and on the bandwidth requirements. For a digital computer, the bandwidth requirement can usually be expressed in terms of the amplitudes of output function derivatives. Maximum possible values of the derivatives can normally be determined by analyzing the geometry and the dynamic character of the output function.

The first trial in the determination of the maximum permissible computational time can be accomplished by first calculating the greatest possible velocity of the output function, and then by selecting a computational time such that the change of the output function within the computational time will not be greater than a maximum permissible error.

Errors due to quantization, truncation, round-off, function approximation, etc., must be considered separately as additional system errors. In certain problems, the computational time calculated from the investigation of the maximum output velocity may be extremely short. Extremely short computational times can be realized with incremental computers. However, internal rates of several megacycles are necessary in order to construct incremental machines which have equivalent bandwidths equal to the bandwidth of analog computers and accuracy of 1 part in 10,000.

In many applications, long computer memory is undesirable as, for example, in all real-time control and stabilization computers. Computer response to step inputs in target tracking applications must be excellent. Errors must be self-correcting, and the accuracy of the computer must be independent of the accuracy of previous computations.

These requirements cannot be readily satisfied by purely incremental computers. The selection of a certain type of real-time computer should be based on the specific requirements of each problem.

The best results can be achieved if the design of real-time computers is specially tailored to each problem. The specification for a real-time computer is usually determined by accuracy requirements and the characteristics of the time function to be controlled.

There are usually several other factors which are normally well specified; for example, the weight and size of the computer and the type of hardware to be used. These requirements, combined with the environmental specification, usually determine the maximum practical internal rate of the machine.

Several design parameters must be considered to determine the optimum combination of computer accuracy, internal speed of operation, approximations used, sampling rate, and the time of computation.

The maximum permissible computational time can be determined by analyzing the nature of the output function. The output function can always be expressed in terms of a Taylor series. The actual mathematical manipulation can be quite involved. It may also be difficult to determine the maximum possible values of

all the derivatives of the output function. However, if the motion of a physical object is considered, it is usually sufficient to analyze only the first two or three derivatives in order to describe adequately the output function. Rapid changes in acceleration are very rare, and, therefore, higher order terms of the expansion can be disregarded.

The Taylor expansion can be regarded as a polynomial in t . It is possible then to substitute a polynomial for the output function. The period of time in which the polynomial substitution is valid can be determined by calculating the difference between the polynomial approximation and the output function. The difference must be less than the maximum permissible error. The higher the order of the polynomial used, the longer the period of time over which the substitution is valid. The computational time can then be determined by the time it takes the output function to diverge by a certain predetermined amount from the polynomial approximation.

The minimum sampling rate and the maximum computation time can then be determined for each order of the polynomial used as the output approximation. Computation times are progressively greater as the order of the polynomial increases.

The determination of the coefficients of the polynomial require the determination of the appropriate derivatives of the output function. The polynomial coefficients can be calculated on the basis of several samples computed at given time intervals. Using Newton's backward interpolation formula, it is possible to determine the coefficients of the polynomial by simply calculating the differences on the basis of several samples of the output function. (See Fig. 1.)

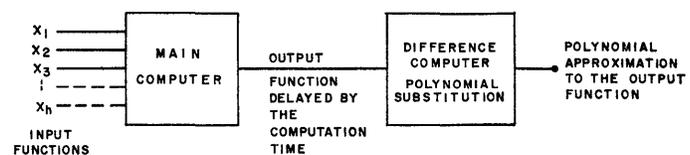


Fig. 1—Computer block diagram. Output function is compensated for the computational delay by means of a polynomial substitution.

The computation of function differences involves subtraction. Since random errors are not correlated, they are not subject to cancellation. In systems in which random and bias errors are of the same magnitude, a second order polynomial is probably the highest order which can be practically used. The computation of the terms of the polynomial makes it necessary to memorize the results of several computations. In other words, it is impossible to construct a computer which uses polynomial approximation and has no memory. However, the memory is relatively short. If a second order polynomial is used, the computer memory is equal to only three computation cycles.

The use of a polynomial approximation to the output function offers an added advantage which may be important in certain applications. The output function can be generated in steps which are smaller than the maximum permissible errors. The need for this form of output may arise if a high performance servo is controlled by the output of the computer. It can be seen from Fig. 2 that the actual output of the computer consists of a sequence of polynomial segments and that there is a discontinuous jump from a polynomial to the polynomial whose terms have just been calculated. This discontinuity can be made as small as desired. The reduction of the output function steps, however, can be achieved only at the expense of the computational time. It is possible, then, to trade computation speed for accuracy and vice versa.

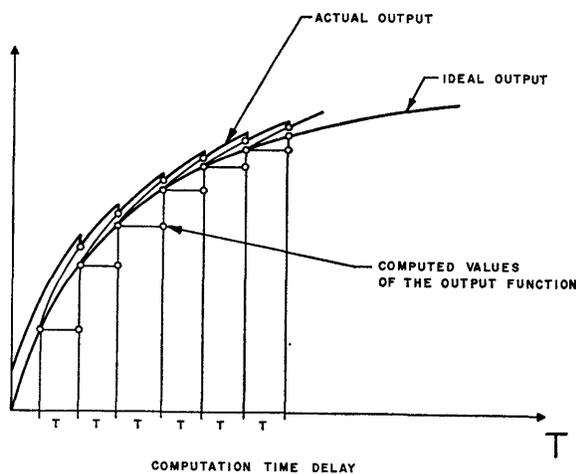


Fig. 2—Computer output function. Output function is approximated by polynomial substitution.

In between the computational times, the output function is not directly controlled by the input functions. However, the nature of the output function is such that it cannot possibly diverge from the approximated value by more than a certain predetermined value. This maximum deviation can be calculated by taking the terms of the Taylor expansion of the output function which do not appear in the polynomial approximation.

Once the sampling rate and the order of the polynomial approximation of the output function is determined, it is possible to determine the bandwidth of the computer. The bandwidth can be calculated by evaluating the accuracy of the computer as a function of the output function frequencies.

The frequency of the output function is postulated, and the rms value of the errors due to the polynomial approximation is calculated. For every frequency, a certain value of the rms error can be determined. The bandwidth of the computer can then be defined as the maximum frequency at which the rms error is still within the permissible limits.

In all real-time control and stabilization computers, it is always necessary to compute some trigonometric functions. There are many ingenious schemes of computing these functions by using the incremental techniques. All these techniques, however, suffer from the limitation of having infinite or very long memories. In the Epsco STARDAC Computer, the trigonometric functions are calculated using the Tchebycheff polynomials. Sine and cosine functions are usually needed simultaneously. In the Epsco STARDAC Computer they are calculated concurrently by using the powers of the argument and the multiplying the result by appropriate Tchebycheff coefficients. A very high degree of accuracy can be realized if the Tchebycheff polynomial is used within an interval of 0° to 90° . Simple logic is used to accommodate arguments outside of this range.

In this high-accuracy, real-time system, error analysis is probably the most important phase of the system design. All possible sources of accuracy-limiting factors must be carefully analyzed.

In the applications in which the computational time cannot be disregarded, a polynomial substitution for the output function is used to offset errors due to the computation time. The polynomial substitution can be only approximate and consequently an error is introduced. Truncation and round-off errors can be determined by analyzing the number of significant digits lost in the computations. Errors introduced by the substitution of Tchebycheff polynomials for the trigonometric functions can be determined.

Output errors due to the errors present in the input functions must be carefully analyzed since these errors determine the maximum realizable accuracy of the system.

The accuracy of the input function has a profound effect on the decisions which must be made in the design of the computer. If the computer is designed correctly, the errors it introduces are normally smaller than the output errors caused by the errors in the input functions. However, the propagation of the input errors through the computer must be carefully analyzed since some of them can be amplified in the computer more than others. The input function errors can be divided into two categories, bias and random.

Bias errors can be defined as those whose magnitude is consistent. In other words, the magnitude of an error can be predicted with a certain accuracy on the basis of the errors present in several previous measurements. On the other hand, random errors can be defined as unpredictable. The random error in any sample has a probability which is independent of the errors present in the previous samples.

The propagation of these errors through the computer can be traced easily by using appropriate partial derivatives. This error analysis is well known to those who have designed fire control computers. However, the relative magnitude of bias and random errors in real-time digital computers is normally different from the

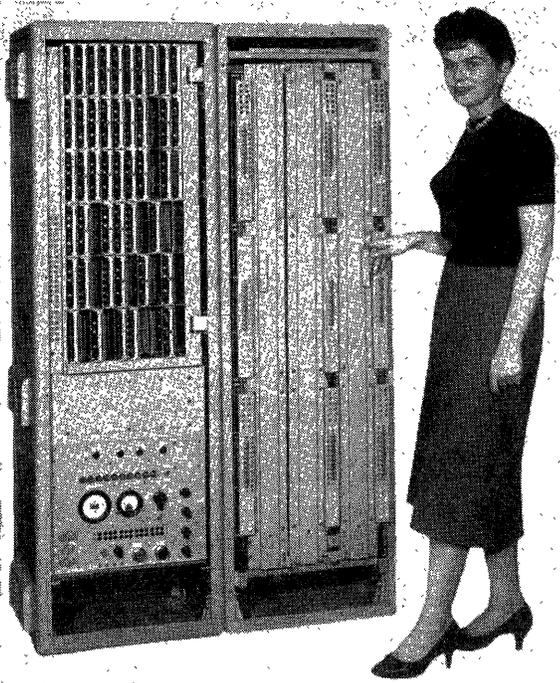


Fig. 3—Computer with covers in place.

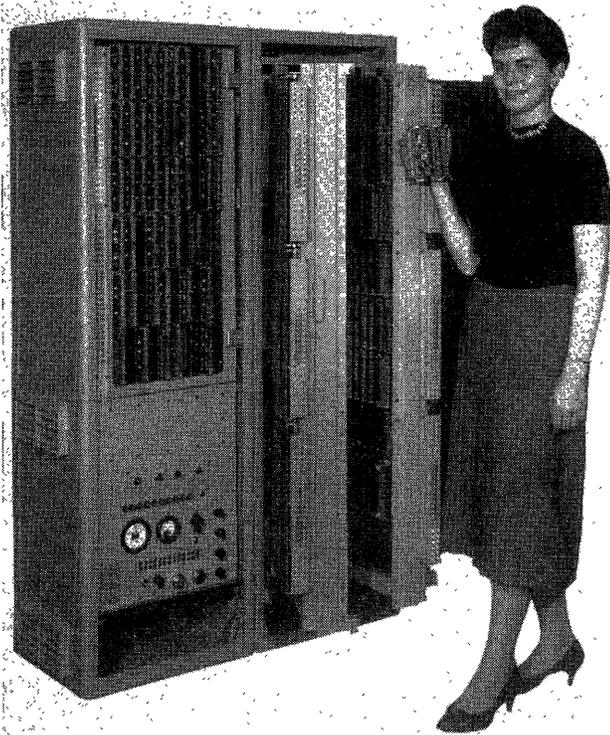


Fig. 4—Computer with covers removed, showing access for servicing.

relative magnitude of bias and random errors in, for example, radar returns.

In real-time control computers, input random errors are usually small and they are very often introduced only by the input quantization. The quantization random error has a rectangular probability distribution

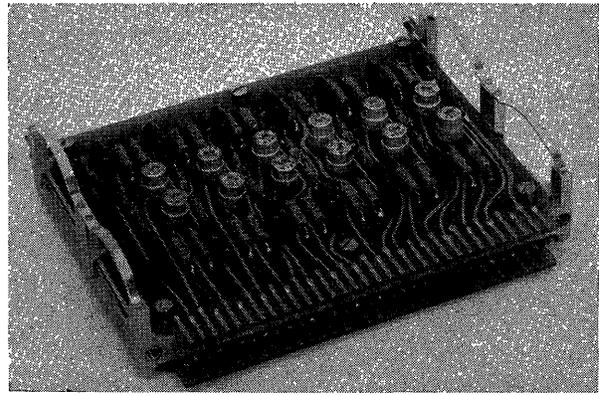


Fig. 5—Digital computer module, assembled.

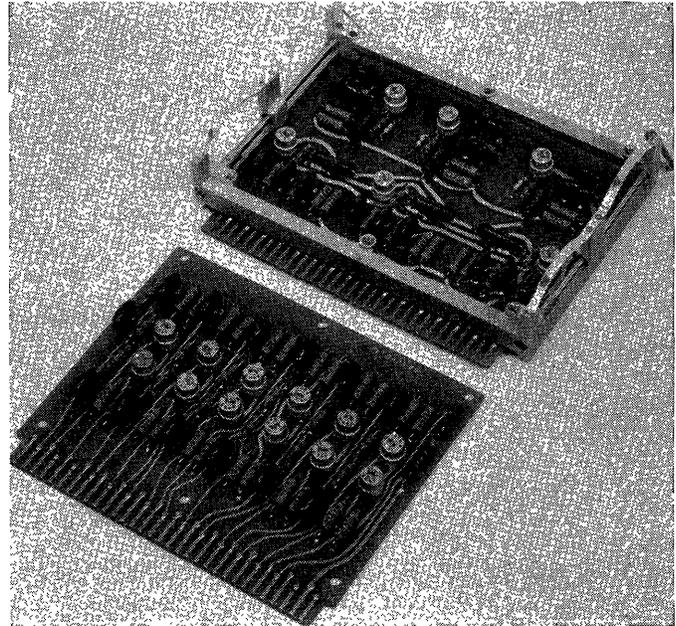


Fig. 6—Digital computer module, disassembled.

with a maximum possible error equal to one half of the least significant digit.

Various methods can be used in order to minimize the effect of random errors on the output function. Input random errors are particularly harmful if differences are employed in the computation of the polynomial which is used as the approximation to the output function. For example, if a second order polynomial is used, the third difference is calculated and is used to smooth out the output function. This compensation is valid only if the noise level is such that the third difference of the output function is much smaller than the measured third difference due to random input errors. This method, however, leads to relatively complicated equations. It is often possible to obtain a significant improvement by simply reducing the quantization errors. This is obvious since bias errors are not amplified as much in the computation of differences as are random errors.

Accuracy analysis would not be complete without a description of the selection of the control equations. In

real-time, digital, control computers, accuracy can be greatly limited if a large number of mathematical operations must be carried out in order to compute the output function. Long computations are undesirable for two reasons. Large numbers of computations are time-consuming; and also, in each arithmetic addition as much as one half of the least significant digit may be lost. It is then necessary to know exactly what is the largest possible number of operations which might be necessary under the worst possible combination of input variables. The number of computations, sometimes, is very difficult to predict. This is particularly true if the computer function involves division and if the denominator, under certain conditions, approaches zero.

Unfortunately, this condition arises often in all problems in which spherical geometry is involved; this happens, for example, if it is necessary to compute an angle whose tangent is determined by a ratio of two expressions which, in turn, are determined by some other trigonometric functions. The angle itself is uniquely determined for the whole interval from 0° to 360° ; however, the tangent is discontinuous at 90° and 270° .

In the STARDAC Computer, this problem was solved by the use of an iterative routine, which made it possible to compute the argument even if the tangent of the angle approached infinity.

As mentioned before, the STARDAC Computer has a built-in sine-cosine function generator. First a number is substituted for the value of the argument and the computer calculates the sine and the cosine. Then the sine of the argument is multiplied by the denominator and the cosine of the argument is multiplied by the numerator. In the second step of the computation, a comparison is made between the two products. The difference is then added directly to the number which was substituted for the argument. Then the cycle is repeated.

Mathematical justification for this operation is almost self-evident if the numerator of the fraction is represented as $\sin A$ and the denominator as $\cos A$. The term which is added to the argument can be expressed as

$$\Delta = \sin \theta K \cos A - \cos \theta K \sin A,$$

but θ was selected at random and was not equal to A . So the equation can be rewritten as:

$$\Delta = \sin (A + \Delta\theta)K \cos A - \cos(A + \Delta\theta)K \sin A$$

or

$$\Delta = K \sin \Delta\theta.$$

For small $\Delta\theta$, the value of Δ is equal to $K\Delta\theta$. The function converges rapidly if the value of the coefficient K is close to unity, and in a few iterations the error becomes negligible even for systems which require extremely high accuracy. The program is simple. No ambiguities arise and the arithmetic operations contain only multiplications, additions, and complementing. All these operations are particularly easy if performed in straight binary code.

The packaging techniques used in the construction of the STARDAC Computer can best be presented by referring to Figs. 3-6. Fig. 3 illustrates the computer complete with power supplies and input-output equipment. Fig. 4 shows the computer with covers removed and the frames pulled out for servicing. Figs. 5 and 6 show typical modules used in the computer.

It is felt at Epsco that a family of real-time computers such as described in this paper will find broad application in the field of high-accuracy real-time control systems such as stabilization computers, fire control computers, navigation computers, autopilots, etc.

A computer whose design is based on the approach outlined in this paper can offer an ideal solution to the problem of maintaining extremely high internal accuracy. It is believed that the need for these computers will grow together with the need for miniaturized, general-purpose computers. It is felt that this new type of computer will soon establish itself as a member of the family of computers together with the stored program, general-purpose machines and analog computers.

The Man-Computer Team in a Space Ecology

J. STROUD† AND J. McLEOD‡

IN OUR coming adventures in the conquest of space we have more problems than the well-publicized one involved in merely getting off the earth; once in space, man has the problem of survival in an extremely hostile environment. We are adapted to life on this earth, not in space nor on any other planet of which we have knowledge. In our first frantic efforts to find solutions to the many problems attendant on this, we might reflect that once upon a time man was not adapted to live on the surface of this earth either. In order to survive on land, the many specialized cells which cooperate to make up modern man found it necessary to bring their environment with them when they crept ashore. These cells now live and replicate in a miniature sea in which the temperature and even the salinity are controlled at about the same values as those of the ocean in which they once lived.

So it seems that the solution to the problem of survival in space is obvious: To stay alive man must again take a reasonable facsimile of his accustomed environment with him.

We have found the answer!—Or have we? The only way we can think of implementing our solution would be to arrange an “air lift” to man’s abode in space, and thus keep him supplied with his every need. The trouble with this is the high cost of the “lift.” Current American freight rates from earth to a minimum satellite orbit, the staging area for our journey into space, happen to be about \$30,000 a pound. And present indications are that it will be a long time before the rates are reduced by a factor of ten—and a lot longer before they can be considered “reasonable.” So when we consider that a man eats and breathes his own weight in food and oxygen every two months, and that he will certainly need other supplies, we find that our first solution just isn’t feasible.

Stated another way, we are forced to conclude that to survive and replicate himself in space, man must be equipped *before he leaves his staging area* to “make do” with what he finds in space, namely, abundant radiant energy from the sun and some very raw, raw material from an occasional asteroid.

Certainly there is sufficient energy; more than a kilowatt per square meter at the distance of the earth from the sun. This is an amount which, if applied to a man’s hide, would be sufficient to run him if he were only an efficient transducer. It would also be enough if applied

to the hide of the space ship, or better, an extension of it, to propel it about the solar system from the inside of the orbit of Mercury to the orbit of Jupiter.

Of course, as is sometimes said of the impossible, solving the problem of converting asteroid chips to support life in space may take a bit longer. But with energy and sufficient know-how, it can be done. Theoretically man can maintain himself in a space-ecology indefinitely. In fact, we can say with some certainty that someday precisely this process will take place, and that the human population of the solar system may very well increase a millionfold in a couple of thousand years, with 99.9 per cent of this population living in comfort and even luxury in little artificial worlds in space.

Most of the knowledge which allows us to live on this earth is currently locked up in our own genes and those of the plants and animals which do the real work of our planetary ecology, while we exist as the prime parasites. To re-educate the necessary genes to live in space would be at best a slow evolutionary process, which we have neither the knowledge nor the time to accomplish. We must find a fast, a revolutionary, way to furnish man with the necessary know-how.

Admittedly, to satisfy our reactionary genes will take a lot of know-how. It is beyond the scope of this paper—that is to say, it is beyond the combined abilities of the two authors—to say just how much a man or a colony must know to survive in a space ecology.

Or, much more to the point, considering that the cost of a first-class (and only class) one-way ticket just to the staging area for a space mission is about half a megabuck per person, the question becomes: How *few* people can collectively know enough to survive in space? So far as we know no one knows, or is making a serious effort to find out. This is either a sad commentary on the direction of our space effort or a happy one on our security system—if anything about security can be happy. However, if it is true that we are not making an all-out effort to answer this question, to point out that it is already frighteningly late to begin such a fundamental task is to belabor the obvious. Suffice it to say that we must get on with the task as though our very existence depends on it, *as well it might*.

But no matter what amount of know-how is found to be required to exist in space, it is the burden of this paper that computers are the only means by which the necessary knowledge can be made available to men out there at freight rates we can afford.

Having stated the problem and suggested a broad solution, we leave the details to the fertile imaginations

† Naval Electronics Lab., San Diego, Calif.

‡ Convair-Astronautics, San Diego, Calif.

of our bright young colleagues while examining in more detail one important aspect of the problem: the man-computer relationship which will be required.

First let us take a brief look at man. His internal memory is so unreliable that he must depend on external aids, usually written records. These must be 50 to 75 per cent redundant to be understood, and they cannot contain more than 100 bits of information per square inch and still be read. Microfilming can increase the density of information but it cannot improve man's reading rate beyond the five bit per second human limitation.

Moreover, there are very severe limitations as to what man can stand physically and emotionally. His sensitivity to his environment, already mentioned, may be reflected in a quite understandable anxiety concerning his own well-being under one set of conditions and boredom under another. Neither is conducive to reliable performance.

Computers, however, can read non-redundant material with densities as great as a billion bits per square inch, and they can read at rates of the order of a million bits per second. That they can do literature searches and prepare abstracts is hardly news, but it is also true that they can perform feats of symbolic logic and deductive "reasoning." Moreover, they don't get dangerously upset under some conditions and negligently bored under others.

For these and other reasons, computers are being widely used on earth to augment man's efforts in science and industry, even when the salary and overhead of the individuals being augmented may not exceed \$30,000 a year.

Certainly then, all will agree that when man is sent into space where the cost of his per diem and transportation is nothing short of fabulous, he must have computer support.

Note that we have used the words "augment" and "support." Even in our enthusiasm for some of the superior capabilities of computers we do not suggest that man will continue to allow them to explore space without him, although it is evident that several generations of computers will have acquired quite a wide experience in space operations before man ventures forth.

Eventually man must go into space; if for no other reason, "because it is there!" However, for a sufficiently great number of people to go in one colony to collectively know enough to survive is not practical with any earth-orbit ferry which we can expect to have in the foreseeable future.

For these reasons the man-computer team proposed here for a space ecology does not include a large number of human specialists, but rather a few humans of unusually broad background who will only have to be able to ask the right questions and do any inductive reasoning which might be required. The computers will select and supply all detailed information needed, make necessary computations, and make all decisions which can be reached by symbolic logic or deductive processes.

We must recognize, however, that to use computers in space effectively as a source of most of the know-how required for man to thrive will require improving the art more than a little, and mostly in the area of establishing man-computer rapport.

At present, many people are inclined to refer all too freely to computers as "high-speed" morons. It is just possible that this is because the esoteric art called programming is *so* esoteric that it is practically a cult, headed by the Senior Programmer as High Priest. If computers do, in fact, sometimes behave like morons, perhaps we should ask if anyone has taught them to do otherwise. We think nothing of spending twenty-odd years programming a two-legged computer. And the amount of effort that has gone into developing the material required is measured in man-centuries.

In contrast, we have had high-speed electronic computers for less than twenty years—only a few computer-generations. We take, at most, a few man-months to "educate" them, and then say, in effect, "You do exactly what I say in exactly the way you are instructed to or I'll brain-wash you!" If we are to get along well with our computers, we are going to have to give a great deal more thought and effort to their education than we have to date. At present, we are paying some pretty fancy prices for a very inferior brand of education for some rather bright hardware because we got started on the wrong track. Instead of instructing them by rote, we should teach our computer to be actively curious, to attempt to find a few answers of its own.

Such a radical reorientation of our approach to computer education is going to be quite painful to some. As the old man said, "If you want to train a dog, ya gotta be smarter than the dog be!" If the same can be said of computers, many "trainers" will be immediately disqualified. Do you know what makes *you* curious? Do you know how *you* distinguish sense from nonsense? Neither do we, precisely. But we had better find out—and teach our computers—if we are to survive in space or anywhere else!

The RCA 501 High-Speed Printers— The Story of a Product Design

C. ECKEL† AND D. FLECHTNER†

THE 501 High-Speed Printer is an output device of the RCA 501 Electronic Data Processing System. This system is fully transistorized. It is expandable in both the area of the high-speed memory and the input-output devices. In the specific case of the High-Speed Printer, this expandability takes the form of optional use of the printer mechanism either as an on-line device or an off-line device.

Initially, the 501 product plan was to design and produce a minimum cost EDP system. Therefore, the first printer specifications called for on-line operation with the printer being driven directly by the computer. This accomplished two things. It held the cost of printer electronics to a minimum, but still allowed the system to have a high-speed output capability.

Subsequent product planning developed the need for system expandability, that is, a system which could be enlarged at the user's convenience if the work load increased. To meet this requirement, a program was also started to design buffer electronics to allow the printer to be run directly from magnetic tape—off-line.

Fig. 1 shows a scale model of a basic RCA 501 system. Information is entered through the paper tape reader at 1000 characters per second. Printed output is available from either the monitor printer or the on-line printer. Fig. 2 shows an expanded system with provision for punched card input and output and additional magnetic tape storage. The High-Speed Printer is now an off-line device with buffer electronics permitting it to operate from magnetic tape.

The specifications set up for this printer were that it should be capable of printing at least 600 lines per minute with 120 columns per line. It should be capable of producing at least an original and three carbon copies, offset masters and ditto masters, and contain the necessary logic to be applicable in all types of format printing. Not an integral part of the design specifications, but perhaps most important over-all, were the criteria that the printer should be low enough in manufacturing cost, but high enough in performance so that these factors alone could ward off early obsolescence. A corollary to this was the fact that the length of design cycle must be held to a minimum and the design cost should be reasonable.

The mechanism selected for the printer was of the "flying wheel" variety. Basic techniques employed in printers of this type are well known, therefore we shall

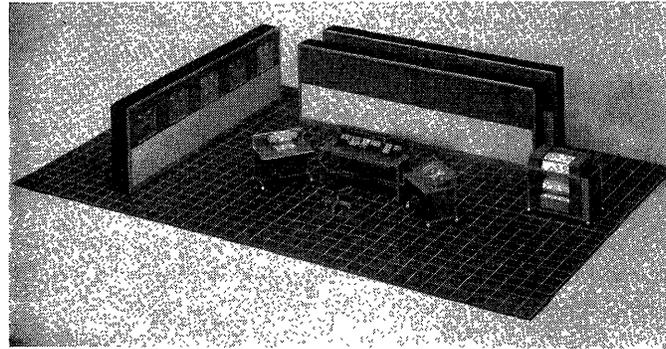


Fig. 1—Basic 501 system.

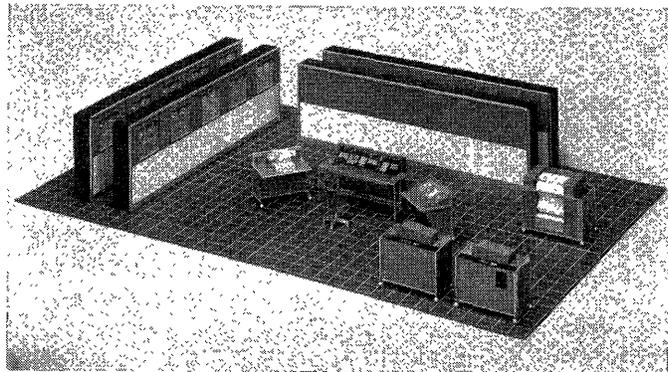


Fig. 2—Expanded 501 system.

only dwell on the functional aspects that illustrate the product development.

Fig. 3 shows a flow diagram for the on-line printer. Information to be printed is stored in the high-speed memory of the computer and the format is controlled by a program in the computer program control. The memory contents are scanned and a line is printed with each revolution of the print wheel cylinder. Synchronization of the memory and the character identity coming into position to be printed is accomplished by a photoelectric code disk assembly, mounted on the same shaft as the print cylinder. The coded bits emerging from this disk are mechanically phased with respect to the character they represent on the print cylinder. This allows sufficient time for a particular character to be compared for its occurrence in the computer's memory and if it exists, a bit is placed in the shift register corresponding to the proper print column. A clock pulse from the computer then causes printing of this character identity. The process is continued until the computer memory has been examined for all 51 possible printing characters. The computer next generates a sig-

† Electronic Data Processing Div., RCA, Camden, N. J.

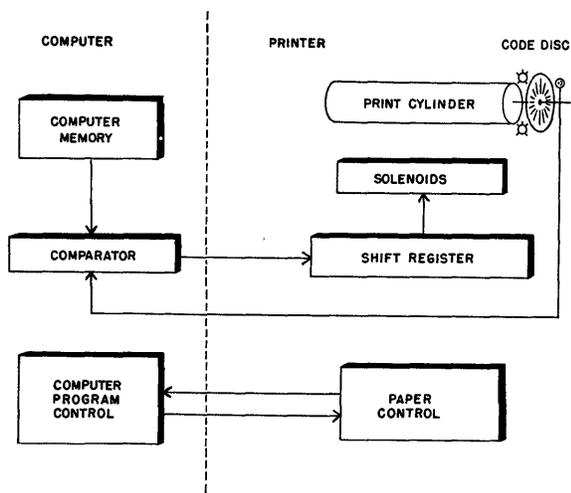


Fig. 3—On-line flow diagram.

nal indicating the amount the paper should be moved, and upon receipt of a return signal indicating that the paper has been moved, the entire process is repeated.

Fig. 4 illustrates the off-line operation employing suitable buffering logic. The buffer unit is designed to accept one line of information at a time, from magnetic tape, store it temporarily, and then print it out. The line is stored in a core memory, the input to which consists of a coincidence between character identity and column location. The memory is clocked out by the photoelectric code disk assembly as each character identity comes into print position.

Printing is normally accomplished in an asynchronous manner. That is, provision is made to determine when all character identities to be printed on a line have been printed. Upon receipt of this signal, another line of information is immediately read in from magnetic tape as the paper is shifted. In this manner, basic printing speeds may exceed 600 lines per minute reaching as high as 900 for numeric printing. The logical circuitry in this area also serves as an accuracy check on the number of characters printed vs the number of characters which should have been printed.

In order to control the printed format, several features have been incorporated. First, by means of a plug-board, incoming information may be tabulated to any of 24 predetermined positions; this same feature may also be used to delete information which is not wanted. It is also possible to effect multiple printing of the same data on one line, again by use of the plugboard.

Fig. 5 shows the printer mechanism, which is used for either the on-line or off-line operation.

Now that the printer has been described, the following discussion will outline some of the factors which influenced the product design.

The need for economical high-speed printers for computer output has persisted. Both the electronic and electromechanical printers were considered at RCA. From an economic and state-of-the-art point of view, the electromechanical seemed more promising. Rotary

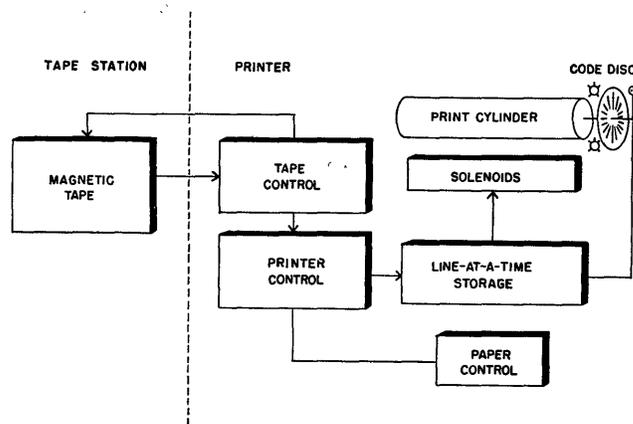


Fig. 4—Off-line flow diagram.



Fig. 5—Picture of printer.

wheel printers (Fig. 6) looked to us to be the best compromise as far as simplicity of mechanism and high printing speed are concerned. The earlier equipment was designed using mechanical printers of this type. Since we already had experience with this type of printing mechanism (certain problems were known to be problems) the new product development for the RCA 501 system consisted of refinements and improvements in the techniques. We already knew how to make good print wheels, and how to be consistent with the solenoid fabrication, and were familiar with the many other necessary techniques.

An area that we felt needed some investigation was that of high-speed paper shifting. At the time the project was initiated, we had a development design of an electromechanical detent spring clutch which gave promise of very high-speed paper shifting. We found, however, that a magnetic clutch, suitable for paper shifting, though not quite as fast, was already commercially available, and so it was adopted.

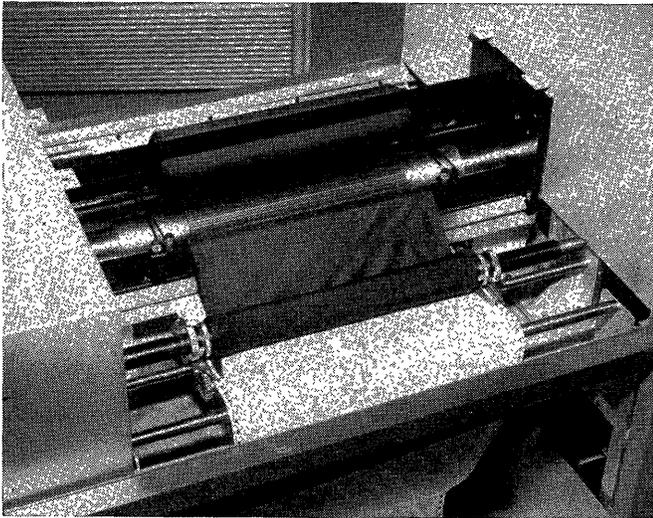


Fig. 6—Print wheel area (ribbon removed).

In order to appreciate the problems involved in fabrication of the print solenoids let us examine for a moment the general concept of print quality. Admittedly, this is a subjective type of thing and depends primarily on the ability and the resolving power of the human eye.

We have found, for example, that it is possible to detect a vertical misalignment of about 0.005 inch between adjacent characters in a line of print without much difficulty. In the conventional typewriting or typesetting, where similar misalignments occur more frequently in a horizontal direction, the effect is not generally displeasing. People are more familiar with this type of printed copy, and they usually accept it without notice. However, they detect any vertical misalignment quickly and question its reason for existing.

Fig. 7 shows the word "link" in which the *i* is 0.005 inch above a line and *n* is 0.005 inch below the line with *l* and *k* on a line. Here, also, you can see the irregular horizontal spacing in the top word, a design requirement of wheel printers. We readily accept this difference in spacing because of the differing widths of letters.

Obviously, vertical misalignment is a normal consequence of rotary wheel printers which must be minimized. The primary way in which this is done is to assure that all 120 solenoids are as near alike as possible both electrically and mechanically.

From an ease of manufacturing and of maintenance standpoint, we would have liked to have placed the print solenoids in two banks, one on each side of the print wheel center line, that is, 60 per side, so that all the solenoids would be similar. However, the problem of getting enough energy into and out of a solenoid, which is only 2/10 of an inch thick, without crosstalk, resulted in a design compromise of two banks of solenoids on each side. We found that with the two-bank design, we could assemble and pot the solenoids in groups of five. Fig. 8 is a representation of the solenoid area of the printer.

The potted assembly of five solenoids is machined in a single operation so that the tolerances can be held.

LINK
LINK

Fig. 7—Normal printing and 0.005-inch misalignment above and below a line.

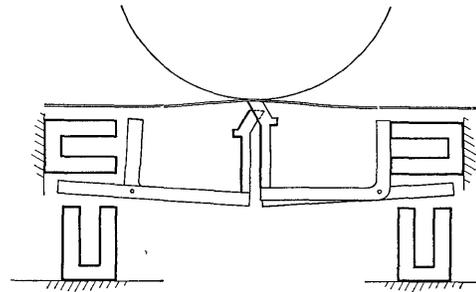


Fig. 8—Print hammer action.

This technique is similar to that used in fabricating digital magnetic recording heads. By this means, we are able to make a good solenoid economically with the correct tolerances built in, thus eliminating later assembly adjustment. Fig. 9 shows the solenoid area with a view of an individual unit.

Another area of investigation centered in the code disk. This assembly has the function of establishing the angular position of character on the print drum and translating the character into coded notation. The code disk has perforations corresponding to the 7-bit code used in the RCA 501 system. Fig. 10 shows the code disk area.

Here we were able to affect manufacturing economies by photo etching through a plate to obtain the coding.

The logic of the on-line and off-line printers is implemented with circuit boards of standard configuration. Most are the same board types used elsewhere in the system (Fig. 11).

The use of standard plug-in packages, plug-ins that are also used in the computer and the rest of the units of the system, also helps to keep manufacturing and service costs down.

Design for simplification and ease of field maintenance meant that the logic should be straightforward and easy to understand. All necessary adjustments should be in convenient locations and the mechanism should be designed in modules which are easily replaceable, such as the print drum, ribbon drive, and paper shift assembly. In the on-line case, a small maintenance panel simulates the computer so that the unit can be serviced independently without tying up the rest of the system.

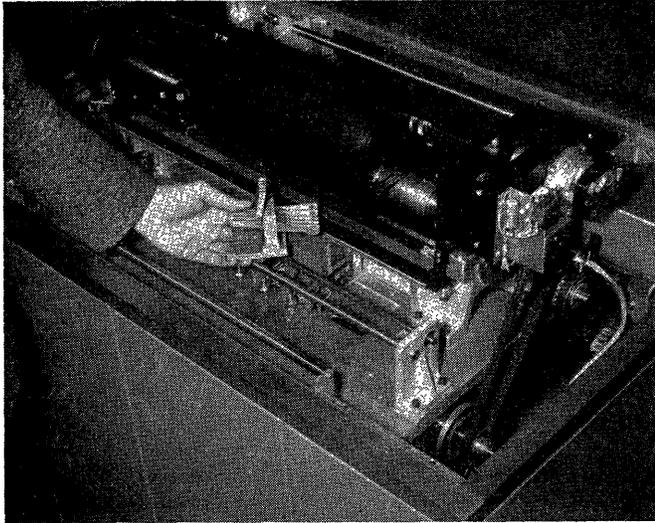


Fig. 9—Solenoid area.

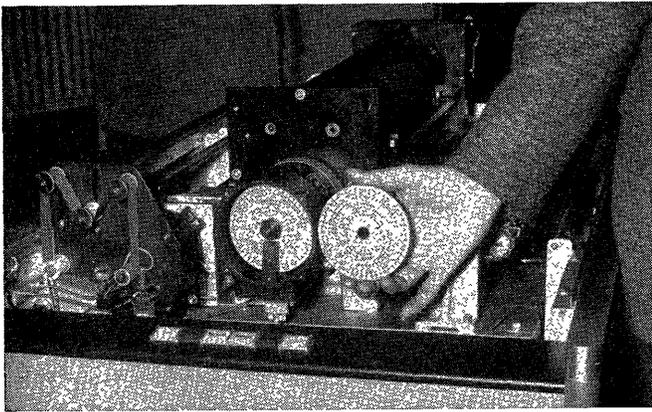


Fig. 10—Code disk.

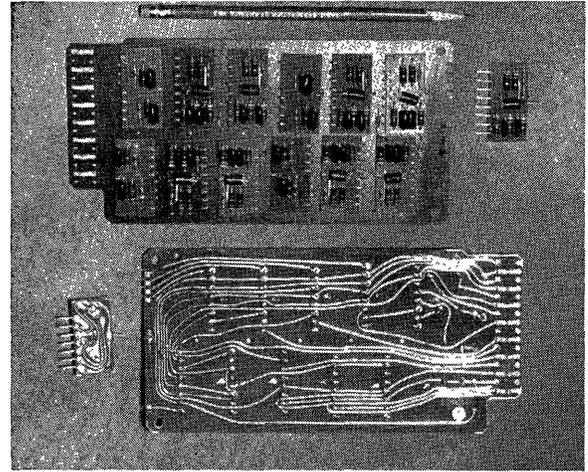


Fig. 11—Typical plug-in transistor circuit board.

CONCLUSION

In the computer field, the major problem encountered in product design is time. The technology is advancing at a rate which constantly makes new products obsolete in the design stage. The product design team must carefully weigh the technological advances which can be incorporated in a design against the need for production release so that the device can be made ready for sale. To insure that the product design remains saleable, the following three items are basic. First, the design should be functionally good. Second, it should be reliable, and third, it should be reasonably priced. When these characteristics are achieved in a product design, regardless of technological advance, the product will not become obsolete—it will remain marketable.

A Digital Computer for Industrial Process Analysis and Control

EDWARD L. BRAUN†

INTRODUCTION

AMONG the more important reasons advanced for the relatively unexploited use of digital computers in industrial process control systems are

- 1) a lack of knowledge concerning process dynamics,
- 2) inadequate development of computers engineered for and suited to process control applications, and
- 3) inadequate reliability of current digital computers.

† Genesys Corp., Los Angeles, Calif.

Our purpose here is to describe a computer which has been designed specifically for industrial process control applications. It promises to satisfy reliability requirements, and can be of great utility even in the absence of complete information on the dynamics of a process. This type of machine can be used for either one or both of the following major functions: It can be used to advantage in the quantitative determination of the effects of different controllable parameters on process performance and also as a process optimization control computer.

We shall not consider here the dynamics of a particular process nor attempt to present a quantitative picture of the benefits to be realized from the use of new instrumentation and digital control computers in various industrial processes. These are the subjects of recent and continuing studies by a number of organizations. These studies indicate that the utilization of machines which allow control to be based on process dynamics as well as steady-state considerations may offer, in particular areas, one or more of the following advantages:

- 1) A reduction in capital investment for new process plants by the substitution of small responsive equipment and control systems for some of the mass and storage capacity on which many plants presently rely for stability and self-regulation
- 2) A reduction in expenditures for raw materials, heating, cooling, catalysts, etc., as a result of more precise control
- 3) Improved productivity
- 4) Improved quality control
- 5) Realizable, effective control for new processes necessitated by technological progress, and which must function under conditions beyond the present limits of controlled process variables.

ANALYSIS OF A PROCESS

The effectiveness of computer control of an industrial process is dependent to a large degree on the data available concerning the effects of controllable parameters on pertinent characteristics of the process. However, this does not necessarily imply that the exact dynamics of a process must be known in precise analytical terms before a process can be controlled. The fact is that control can be and is effected with only a qualitative knowledge of system behavior, coupled with the use of feedback methods.

The objective of a process analysis is the determination of the relationships between the major process parameters and the location of optimal operating regions for the process in terms of these variables. Once the optimum operating regions have been determined, a decision can be made in regard to which variables need be controlled and in what manner in order to maintain optimization of one or more characteristics of the process in the presence of disturbances that may arise.

Let us consider now a relatively simple procedure for analyzing the effects of various parameters on system performance which can be incorporated into the special purpose computer to be described. The procedure consists essentially of specifying initially allowable variations in a number of variables and then programming these changes in order to obtain data on their effects on the various characteristics of the process. The information entered into the computer prior to an investigation consists of the process parameters that are to be varied, the size of the incremental steps of the variation (variable over a limited range), and the upper and lower limits within which the variations must be kept in order

to prevent upset of the process. Where two or more parameters are to be varied, the program will cause all combinations of these parameter values (within the imposed limits) to be impressed upon the system. The only other quantity that need be entered is the stabilization time required for the process to stabilize after each programmed change. This delay is usually referred to as dead time or process lag. It allows for the occurrence of two events. First, it permits a process variable to reach the steady-state value called for by a command in the program. Also, it provides time for the over-all process to adjust to this value.

The values of the various quantities whose effects on the process are to be investigated are determined by suitable sensing devices (whose outputs usually are in the form of dc voltages) which are coupled to the computer via an analog-to-digital conversion system. A convenient way of effecting the programmed incremental changes in the process variables is simply to generate commands which cause the set points of the various controllers in the system to be altered. Just prior to the initiation of each new "change" command, the computer causes the current values of all input and output variables of interest to be read out, either in printed form or graphically.

A useful procedure that can be incorporated into the program is to cause a reversal in the sign of the subsequent incremental commands whenever either the upper or lower limits of a particular variable are reached. This produces automatic cycling between preset limits of a given variable. A convenient method of programming changes in a second variable under investigation would be to cause it to be changed by a single increment each time the first variable reached a limit and reversed. A third variable would be advanced by one increment each time the second reversed, and this type of procedure could be used for as large a number of variables as desired.

Specifically, the analysis procedure outlined takes place as follows. First, the variables of interest are programmed for relatively large incremental changes within a limited range around their nominal set points, the latter being determined either theoretically, from simulation experiments, or operator experience. For purposes of visualization, assume that the effects of two input variables upon a particular output variable are being investigated. For each pair of values of the input variables there will be a corresponding value of the output variable. In general there will be a set of values of the input variables which produce the same value for the output and define what may be referred to as process output contour lines. Examination of these contour lines in a particular area will indicate in what direction within the plane to proceed in order to find better values for the output variable. Once the direction has been determined, the computer investigates the new area in like manner. When the new area to be investigated becomes relatively small, *i.e.*, convergence to a solution is approached, the size of the programmed increments

may be diminished. Once the optimum setting for a pair of variables has been determined (for fixed values of other process variables), it will be found generally that the effect of a third variable is to cause both a shift and change in size of the optimum output variable contour line. Fig. 1 illustrates the variation in the contour lines of the V_1 , V_2 plane as a function of a third variable, V_3 . The values associated with each contour line are given by k_i .

The same type of procedure may be used not only in an experimental effort to gain information about a particular process but also in using a computer to control the process. In this case, small adjustments are made in the controlled variables until a set of values is obtained that produces an optimum output. Whenever a deviation from the optimum occurs, the computer initiates a search for a new set of values of the controlled variables that will produce an optimum output. Thus, by experiment and successive approximations, an optimum solution can be produced even in the absence of complete quantitative knowledge of the process dynamics. Useful data are obtained not only on the relationship between specified input and output variables but also on the accuracy of control that would be required for an allowable change in a given output variable.

COMPUTERS FOR PROCESS CONTROL APPLICATIONS

For optimal control of an industrial process, it is usually necessary to maintain close control over a large number of process variables in a way which takes into consideration not only the effects of the individual inputs on certain specified outputs but also the relative effects of these input quantities. The control system should have the capacity (in the event that a particular input cannot be made optimum) to generate a compensating change in one or more other variables of the system. Also, it is desirable that the computer be capable of optimizing different process outputs in accordance with the current economics influencing the relative desirability of producing different output products, *i.e.*, the fluctuations of supply and demand.

Once the computer has, by the processes indicated or similar ones, produced data on how a given process may best be controlled, it may subsequently be used to control that process. As a process controller, it is desirable that it have the capability to

- 1) monitor, store, and log process data,
- 2) determine the values of the controlled variables that will optimize the output,
- 3) actuate controllers, and
- 4) check the system and itself to detect malfunctions in either.

Before proceeding to the description of a digital computer useful for analysis and control of a typical industrial control process like fractionation or distillation, it is desirable to review the general characteristics and capabilities of analog computers and the two major types of digital computers—namely, the arithmetic or

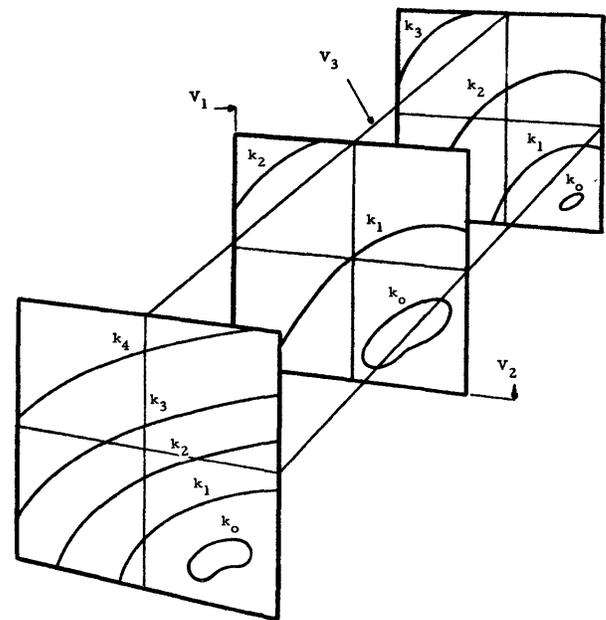


Fig. 1—Planar maps showing the variation of an output variable as a function of three input variables.

integral transfer type of machine and the incremental transfer machine.

The relative merits of analog and digital computers for process control applications have already been considered in the literature. The conclusions reached from these comparisons are that while an analog computer may be adequate in certain cases, it does not in general have adequate capabilities to suit it for more sophisticated control systems. It is limited in its ability to perform operations like the multiplication or division of variables, the generation of functions of several variables, data correlation, extrapolation, etc. It does not have the capacity for logical operations, nor does it provide adequate data storage facilities. It is not well suited for complicated correlation or data processing. Often, it may not be adequate even for relatively simple computations if there are a large number of them, or if non-linear functions are involved. In addition, a digital machine offers greater flexibility in the sense of relative ease of modification of control functions and also in that it provides a number of facilities in addition to the computations required for control, such as data storage (including the storage of calibration data), logging operations, alarm generation, etc. Finally, the analog computer is more prone to faulty operation from marginally operating components and does not offer the self-checking feature of the digital machine.

The relative merits of integral transfer and incremental transfer machines may be summarized as follows.

The integral transfer machine has excellent data storage and processing facilities with a large measure of operational flexibility. The cost of this is the price of a large main store and a large number of arithmetic and control circuits. The upkeep is also high because the complexity of programming and preventive maintenance

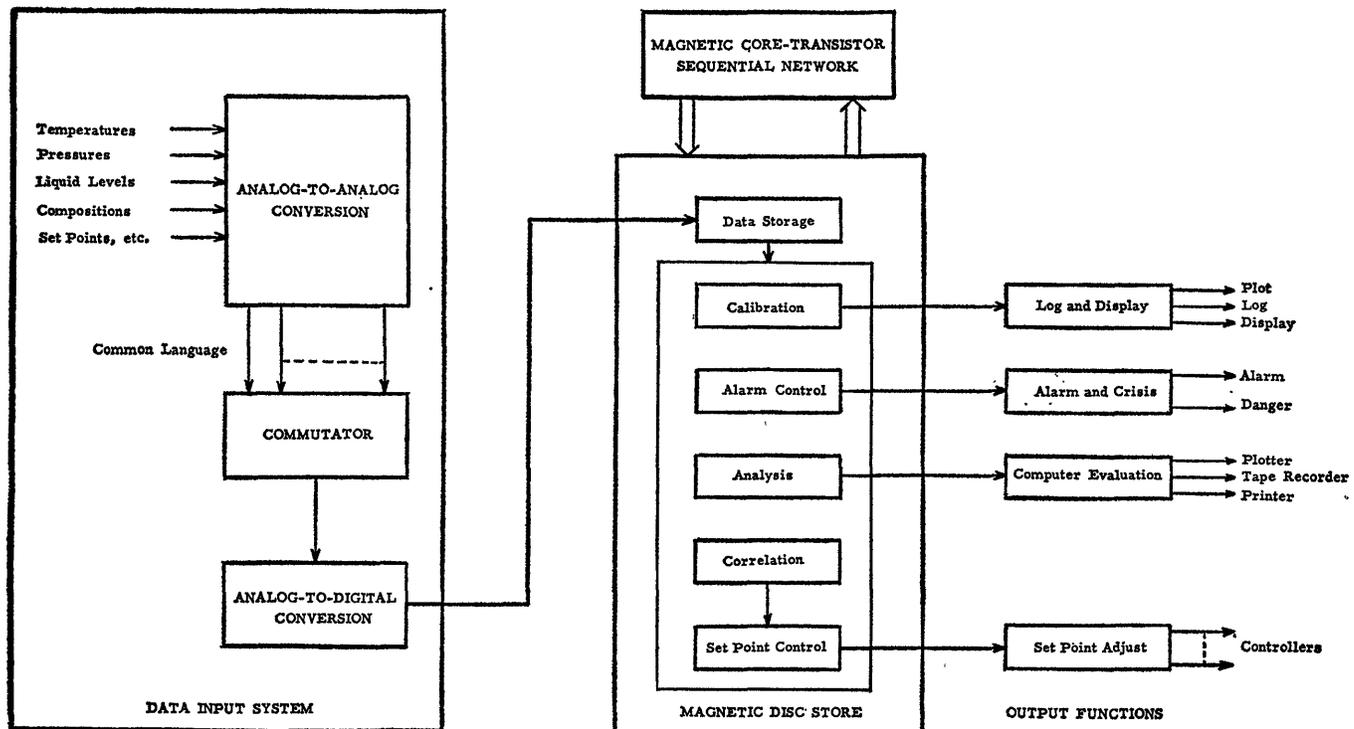


Fig. 2—Functional diagram of industrial process analysis and control computer.

nance procedures demands skilled programming and operating personnel. However, beyond a certain size, additional capabilities and flexibility can be added at little cost. Maintenance of this type of machine is facilitated by its capacity to provide elaborate checks on itself and the control system in which it is incorporated. The integral transfer machine is superior to the incremental machine (even one with variable selectable slewing rates) in slewing time, and therefore better able to implement decisions. However, it is relatively inefficient in computations on continuous variables and capable only of a moderate computation frequency.

Because of its efficiency in computing with continuous variables, an incremental machine with relatively few elements can provide good capacity and a high computation frequency. Its relatively small size gives it considerably better reliability with respect to failures, though its reliability with respect to malfunctions is comparable to that of the integral transfer machine. Also, its maintenance is often complicated because of some of the logical devices used in its design. In its basic form it is relatively poor in respect to slewing time. It is not well suited for problems of logical analysis and is lacking in certain data processing capabilities.

THE INDUSTRIAL PROCESS ANALYSIS AND CONTROL COMPUTER

It is apparent that both integral and incremental computation techniques offer advantages. A basic premise of the machine described herein is that it is good economics to minimize the number of active storage and switching elements by the use of incremental computa-

tion and stored logic wherever possible. This results in a speed of operation that is relatively slow but adequate for process control applications. A functional diagram of a machine of this type now being built is shown in Fig. 2. As indicated, all the functions of the computer are achieved through extensive use of a magnetic disk store in conjunction with a small magnetic core-transistor sequential network. The functions for which the machine was specifically designed include

- 1) storage of data from process instruments, instrument calibration data, safe limits of variables, computation constants, etc.,
- 2) data processing and computational capabilities like integration, function generation, data correlation, smoothing and prediction, solving of differential equations, etc., and
- 3) generation of signals for adjustment of controllers, generation of alarms, etc.

Briefly, these functions are accomplished as follows. Storage of the values of all variables for a specified period, say an hour, at a sampling rate of one or two points per minute per variable is accomplished by use of a delay line of several thousand bits capacity. Thus, whenever a variable exceeds prescribed limits, the recent history of the process is available to aid in determining the cause. These data can also aid in operator supervision of the process, since system trends can be checked by read-out of the same variables at different times as a plot.

Analytical or empirical calibration data for selected variables are stored on a group of channels referred to as

linearizing channels. To alleviate the problem of instrument drift, measurement by each instrument of a known quantity can be transmitted to the computer at specified intervals. From these data, new calibration constants may be generated and stored as required.

A relatively simple alarm control facility is obtainable by the allocation of certain channels for the storage of upper and lower safe operating limits of certain variables. These data are continuously compared with the data in the long delay line. Exceeding a limit causes annunciators and other alarm indicators to be actuated. Off-limit data can also be logged or plotted whenever alarm conditions are indicated. Computational and logical facilities can be provided to produce anticipatory indications of trouble, *e.g.*, searches can be made for dangerous trends and the simultaneous occurrence of events that imply trouble.

The analyzer is used principally to investigate overall system behavior. It can serve as a simulator, computing the solution to equations describing the behavior of the process and its controllers. It can be used to study the dynamic effects of variations in the choice of controllers, and their set points, utilizing sampled data from the system itself.

The function of the correlator is to aid in the determination of the effects of different controllable parameters on the behavior of the system in order to ascertain what variables to control and to compute optimum set points for their controllers.

Set point control is accomplished with the aid of channels in which the set points and control parameter settings believed desirable are stored. These data are continuously compared with sampled data from the process to determine current settings for the controllers. Also, the settings of the controllers are adjusted to maintain output optimization in accordance with data from the correlator. Any combination of proportional, rate, and integral control can be provided.

A most important requirement of an on-line process controller is a long mean time between failures—of the order of six months. Satisfaction of this requirement too is facilitated by a design that minimizes the number of active storage and switching components. Various devices employed to reduce the component count while at the same time maintaining specified capabilities are

- 1) simulation of active storage and switching elements by passive storage elements, use of the disk

store not only for the function of data storage, but also for arithmetic and logical transformations and control functions;

- 2) multiplexing of stored information and time sharing of active components,
- 3) the utilization of both incremental and integral transfer techniques,
- 4) an organizational structure of the stored data which minimizes the amount of control circuitry required for manipulation of the data, and
- 5) specific logical configurations that capitalize on the particular requirements of the control applications in question.

Active elements are used principally to direct data from the disk to the controllers and to output equipment, to modify data in any given channel, and to control the flow of data between channels.

A high degree of reliability is also promoted by using circuits operable over a wide range of parameters, by underrating circuit components, and by eliminating components considered inherently unreliable. Fail-safe operation is achieved by placing the computer in parallel with the process, *i.e.*, so that it controls only the set points of the controllers.

Over-all system performance depends not only on techniques for minimizing the probability of occurrence of a malfunction but also on rapid detection of a malfunction when it occurs.

To serve this end, the entire disk system is periodically and automatically given a test problem, and there is also provision for the insertion of special diagnostic programs by the system operator to allow any detected error to be traced to the process, including controllers, or to the computer.

The type of computer system described can serve in a test and evaluation phase as a process analyzer to determine the feasibility of computer control and as a simulator to investigate the effects of different types of control schemes. It can also function as a fixed-program process-optimization control computer. Though a single machine can be provided with both capabilities, the economics of a particular situation may be such as to justify the use of separate computers, one a flexible analytic computer, and the other a relatively inflexible fixed program computer for control optimization. Each of these machines would, of course, be less expensive than the machine with both capabilities.

The Burroughs 220 High-Speed Printer System

F. W. BAUER† AND P. D. KING†

INTRODUCTION

IN the past few years, computer manufacturers have been developing and delivering data processors with faster and faster processing speeds. Less attention, however, has been paid to getting printed information from the data processor. As a result, most data processing systems are input/output bound.

The speed of the printing device has not been the only restriction; expensive and time-consuming central processor runs were necessary to perform the editing and formatting of the information for the printed page.

In order to bring about a system balance and relieve the central processor of unnecessary data manipulation, a high-speed printing system with complete off-line editing was needed.

To define the problem more closely, we studied such applications as: the printing of insurance premium notices and declaration forms; wholesale drug and grocery billing, oil company and public utility billing; and the preparation of bank statements, stock transfers, and dividend checks. To handle these applications, a printer system must not only be fast but also—considering customer relations—capable of producing a neat legible document.

SPECIFICATIONS

The high-speed, large-volume, complex-editing application, then, was the general framework which defined the boundaries of our system planning.

First of all, the new printer system must be a high-speed device, capable of producing the printing requirements of a majority of applications. The necessity for system speed established the need for independent printer control of format and editing, since high-speed output which depended on extensive editing by the data processor could not be considered true speed at all.

Independent operation and speed requirements indicated that magnetic tape was the logical choice for communication between the central processor and printer system. By using magnetic tape as a buffer, printing as far as the central processor is concerned, is at magnetic tape speeds; the central processor need never wait for the printing device to accomplish its task.

To provide further versatility it was deemed advisable to allow the printer system to work directly with the central processor for those applications requiring on-line operation.

As a final requirement we decided we would not want to prepare special print tapes for off-line operation or require central processor time for editing when on-line.

In either case our obvious aim was to reduce the amount of data manipulated within the central processor. Printing from master tapes or records was a must.

This, in brief, is the application framework within which we established the performance specifications of our new printer system. The term "printer system" is significant. In this case, terminology was dictated by a desire to describe accurately the operation of the units, independent of direct processor control.

GENERAL DESCRIPTION

The Burroughs 220 High-Speed Printer System (Fig. 1), is a transistorized, buffered, on-line/off-line subsystem with versatile editing capabilities controlled by a plugboard. The system, which includes a Printer Control Unit and a Printing Unit, is designed to operate on-line with the Burroughs 220 Data Processor, or off-line with one or two standard Burroughs 220 Magnetic Tape Storage Units.

Control Unit

The Printer Control Unit (Fig. 2), houses an 1100-digit, random-access core storage used as a buffer, which accommodates up to 100 computer words (10 digits plus sign). The Control Unit also contains the system's control circuitry, a 120-character print register, a 120-position bit register, the transistor power supply, and the plugboard.

Printing Unit

The Printing Unit contains a drum-type, high-speed printer (Fig. 3), having 120 print positions, with 10 characters to the inch, and a total of 51 characters per print position. Of these, 15 are special characters, including CR, OD, DR, + and - symbols.

The Printing Unit (Fig. 4), also contains paper motion controls and the power required to drive the printing mechanism.

Vertical line spacing is fixed at six lines to the inch. Printing can be positioned any place on a 16-inch form. The printer can accommodate a maximum form width of 20 inches for centered printing. The maximum form length is 22 inches. The printer can print an original and five carbon copies.

Complete control of paper skipping (Fig. 5) is accomplished by means of a seven-channel punched paper tape loop, $\frac{7}{8}$ inch wide and photoelectrically sensed. The paper tape loop provides six predetermined paper skip positions and a carriage exit position, which is used primarily for page overflow or for logical decisions. Thus, jumping to and printing header information on the next form is easily programmed.

† ElectroData Div., Burroughs Corp., Pasadena, Calif.

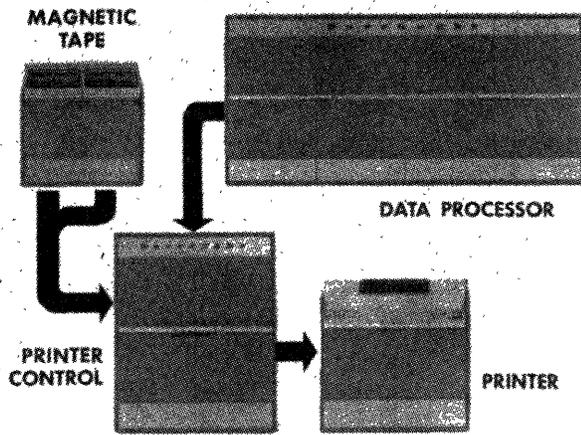


Fig. 1—Burroughs 220 High-Speed Printer System.

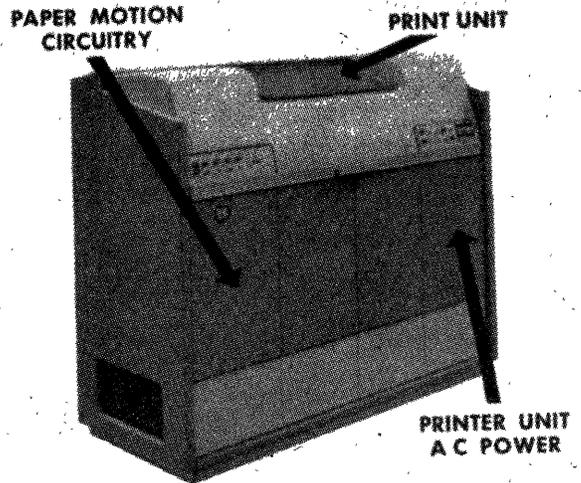


Fig. 4—Printer unit.

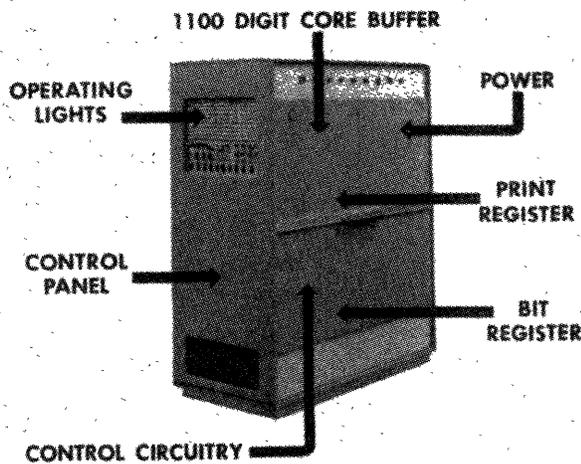


Fig. 2—Control unit.

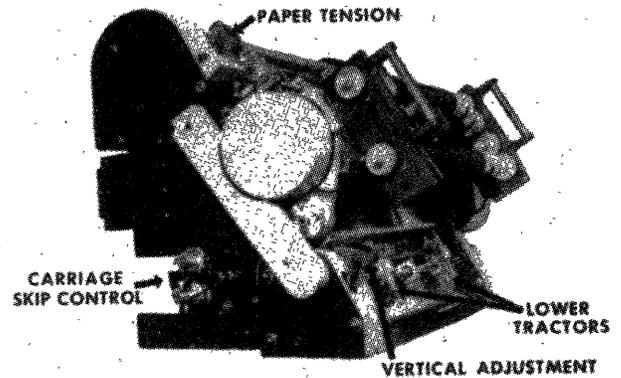


Fig. 5—Printing unit, side view.

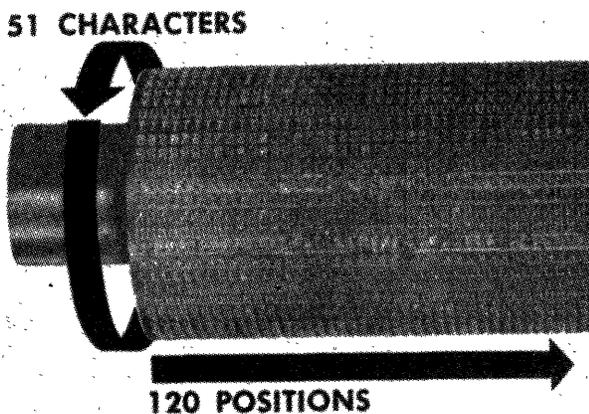


Fig. 3—Print drum.

Under plugboard control, page-skipping and single- or double-line spacing before and after print allows easy accommodation of preprinted forms. Paper moves at the rate of 25 inches per second, with 9 msec start-stop time for a single-line space time of 16 msec.

The printer may be operated at print drum speeds of 750, 900, 1500, and 1800 rpm. The effective printing rate is dependent upon the information to be printed. For alphanumeric information, the effective printing

rates are 624, 720, 1068, and 1225 lines per minute. If only numeric information is to be printed the effective printing rates automatically become 750, 900, 1500, and 1225 lines per minute.

These higher numeric print speeds are accomplished by detecting in the control unit during the loading of the print register that only numeric information is to be printed. (See Fig. 6.) When this condition exists the control unit terminates the print cycle at the end of the numeric section of the print drum rather than requiring a full drum revolution. The time required to traverse the remainder of the print drum, the alphabetic section, is used by the printer system in spacing paper and reloading the print register.

It can be seen that at the top drum speed of 1800 rpm the printing rate is only 1225 lines per minute for numeric information, the same as for alphanumeric information. This is because the time required to load the print register or space the paper is greater than the time required to traverse the alpha section when the print drum is rotating at 1800 rpm.

Fig. 7 is a front view of the print unit showing horizontal paper positioning controls, vertical paper positioning controls, paper tension controls, the ribbon mechanism, and the upper and lower set of form tractors. Lateral positioning of printing and form size ad-

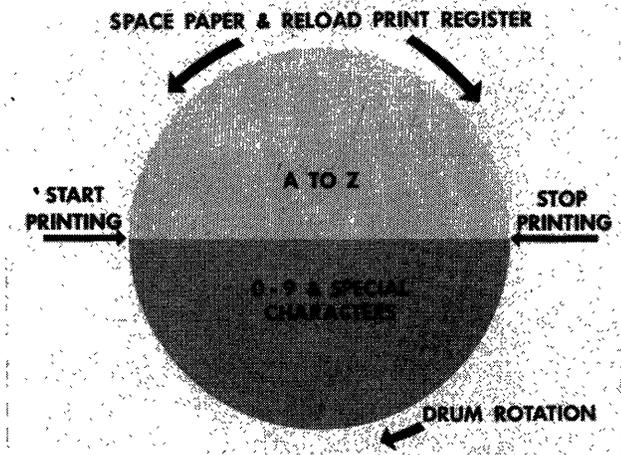


Fig. 6—Character arrangement on print drum.

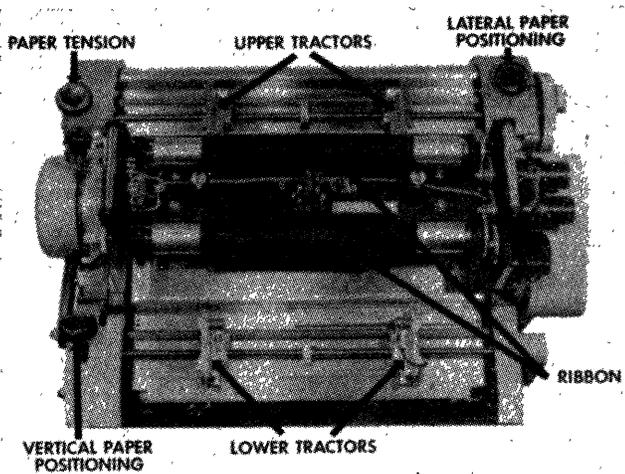


Fig. 7—Printing unit, front view.

justment are accomplished with the control in the upper right of the picture.

Controls are available for moving all tractors to the right or left in synchronism for positioning of printing on the page. In addition, controls are available for moving either the right or the left set of tractors individually for accommodating form width. Paper tension between the upper and lower sets of tractors is controlled by the upper left control knob. By using two sets of tractors, the upper and the lower, and positive mechanical detenting, paper creep is eliminated. Fine vertical adjustment within a line is accomplished with the lower left control knob.

OPERATION

With some of the details out of the way, we can now push a few buttons and see how the Burroughs 220 High-Speed Printer System works. The internal operation of the system is divided into three basic cycles (Fig. 8):

1) Load cycle—During the load cycle, the core buffer is loaded from magnetic tape or from the Data Processor.

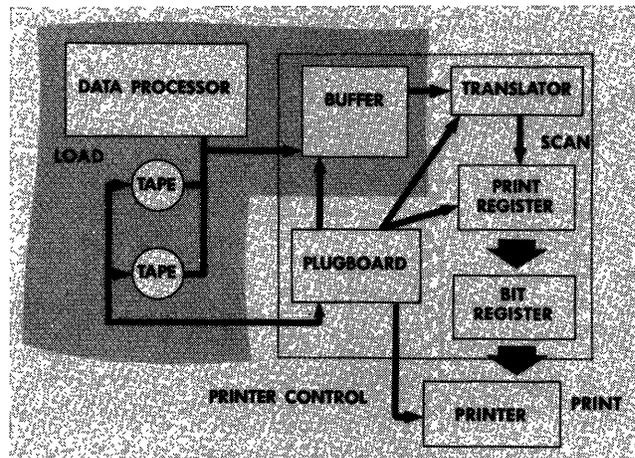


Fig. 8—System flow diagram.

2) Scan cycle—The scan cycle is basically the transferring of selected information from the buffer to the print register. During the scan cycle, all editing, formatting and selection is accomplished. The scan cycle is normally completed during paper spacing time.

3) Print cycle—During the print cycle the line of information is transferred from the print register to the paper. Since the core buffer is not used during the print cycle, the load and print cycles can occur simultaneously.

Load

When used off-line, the load cycle is initiated from the plugboard. Information is read from one of two magnetic tape units. The block length on tape is variable from 10 to 100 words and can be variable within a run. The number of blocks read per load cycle is selected from the plugboard, the only restriction being that no more than 100 words can be loaded into the buffer at any one time. If the capacity of the buffer is exceeded, a buffer overflow alarm is available on the plugboard for automatic corrective action if needed. Selection of the storage area to which the information will be sent is plugboard controlled. As a result, it is possible to have in the buffer at one time any combination of information from 2 tape units and the data processor.

When used on-line, loading of the buffer is controlled by two commands from the data processor. The first command is used to determine if the printer system is ready to accept information. The second command loads the buffer with a record from 1 to 100 words in length. Once loaded, the data processor is freed.

Scan

The scan cycle is so named because information to be printed is not transmitted by plugboard wires but rather by internal channels. The scan cycle includes the transfer of selected buffer information through a translator to the print register for subsequent printing. All editing of information is accomplished during this phase of the printer operation. The plugboard wiring is used

for decision making, selection of starting positions of fields, special character insertion, editing, and formatting functions.

During scan, the starting position of a field is selected; the field then reads out sequentially to the print register, with no restriction on field length, and continues until ordered to a new starting location. After a change of address has been ordered the scan continues sequentially from the new address. The addressing of the buffer is controlled by a counter called the character address counter, which changes the location when set to a new value. Fig. 9 illustrates the operation during the transfer of one digit from the buffer to the print register. The character address counter selects the digit to be transferred, out of 1100 possible settings. For each one of these settings there is an exit hub on the board. These exit hubs are the source of control pulses, to cause address selection, formatting and control. Other functions which can be initiated by these pulses include zero suppression and the insertion of blanks, commas, decimal points or dollar signs. A feature which helps reduce the amount of information to be manipulated is the character emission of all 51 characters for printing of fixed information—the date, for example.

When information is read out of the buffer, the digit value is available on the plugboard and can be used for controlling and testing purposes. Digits so used need not be printed; for instance, decisions whether to print a record or not can be based on the digital value of a key.

The scan cycle is automatically terminated when the print register has been filled with 120 characters, and a print cycle is automatically started. At this time an end scan pulse is available for initiating a buffer load if desired.

Print

At the time the print cycle starts, the print register is filled with a 120-character line. There is a counter synchronized with the rotation of the print drum which tells at any instant the next character on the drum in position to print. By comparing each position of the print register with this counter the positions to be printed are determined. At the start of the print cycle this comparison process begins immediately (Fig. 10).

To illustrate, if the counter were at a value corresponding to *R* all *R*'s in the print register would be printed first. When the print drum has rotated through one position and the counter advanced to *S*, all the *S*'s will be printed and so on. The print cycle will be completed in this case when the print drum has rotated back to *R*. The printing actually occurs by timed firing of print hammers. The 120-position bit register contains the "yes" or "no" of whether the print hammer associated with a particular print position will fire at a specified character time. Loading the bit register is accomplished by means of the print register comparisons just mentioned.

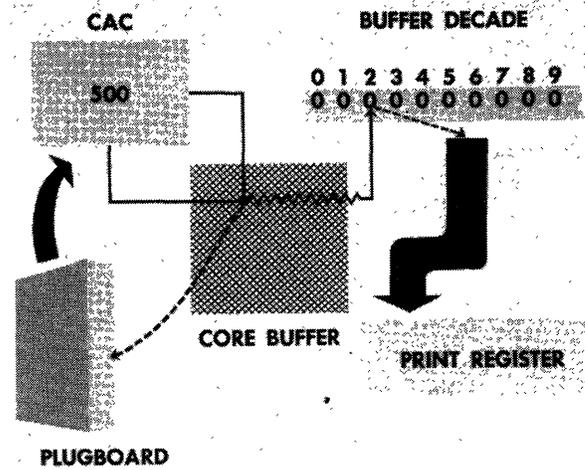


Fig. 9—Buffer readout.

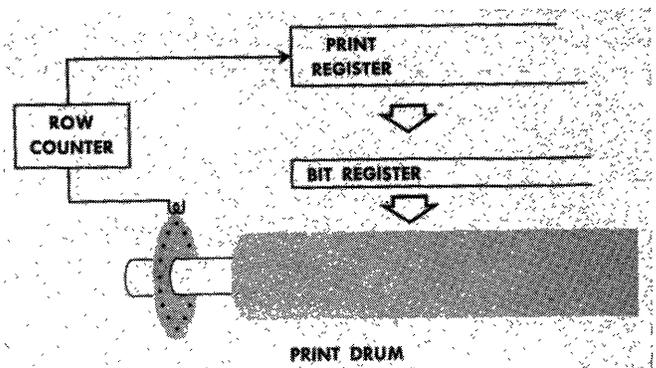


Fig. 10—Print cycle.

If spacing is wired on the plugboard to occur after printing, paper spacing will start immediately after the print cycle. A start scan pulse is available on the board to start a new scan cycle, assuming no load cycle is taking place. If a load cycle is not yet completed, the start scan impulse will be held up until it is. However, a load cycle will normally be completed during a print cycle, except at the highest drum speeds.

SPECIAL FEATURES

This, in essence, is the Burroughs 220 High-Speed Printer System. We will look next at some of the special features of this system. Fig. 11 lists the functions available to complete the editing ability and enable printing from master tapes; these are initiated by character address pulses.

The special features of the Burroughs 220 High-Speed Printer System are described in the following paragraphs (Fig. 12).

Wiring by Exception

The control panel need only be wired when information is to be printed out in a different sequence than that contained in the buffer. In this case, the character address pulse of the last digit of a field to be transferred is wired to address the starting position of the next field.

1. Zero suppress
2. Check protecting asterisk insertion
3. Comma and decimal point insertion
4. Insertion of blanks
5. Delete
6. Special sign translation to +, -, CR, OD, DR

Fig. 11—Special functions.

1. Wiring by exception
2. Field selection
3. Exception or selective printing
4. Field interrogation
5. Multiple line printing
6. Retention of fixed information
7. Relative addressing

Fig. 12—Special features.

Thus, field selection is accomplished. Since it is not necessary to provide plugboard wires for the transmission of all positions of a field the actual number of plugboard wires is reduced to a minimum.

Exception or Selective Printing

Transferring the digit value of the contents of a buffer location to the plugboard during transfer of information to the print register allows recognition of keys within the information. Thus, logical electronic elements cause or prevent printing. For example, when printing paychecks, the plugboard can be wired to print checks for only those employees with earnings, ignoring all employees with no earnings.

Field Interrogation

Logical decisions can also be made by comparing buffer information to preset information in 10 rotary switches. For example, in a statement preparation application, the date would be set in the switches. Only those accounts scheduled for that date would have a statement printed. All other accounts would be skipped.

Multiple Line Printing per Buffer Load

Because the buffer is actually a storage device, any number of lines can be printed from one buffer load. Repeat printing of any information is possible. Complete documents can be prepared with one reading of the tape record.

Retention of Fixed Information

The ability to select the starting address during load enables the retention of information in the buffer, for example, page heading information. This feature is particularly useful when multipage documents are to be printed.

Relative Addressing

(See Fig. 13.) This is a form of indexing register which enables grouped records to be printed in the same format with wiring for only the first of the records. The

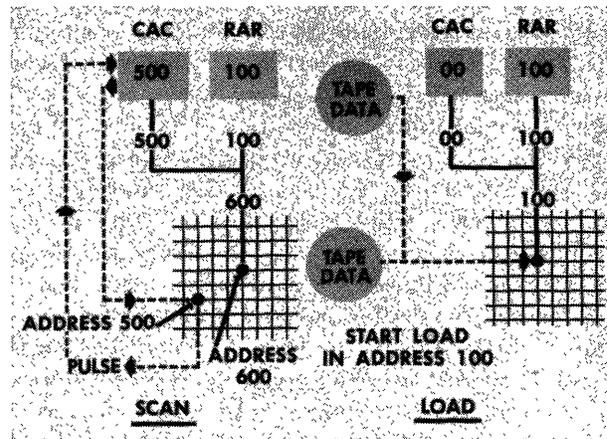


Fig. 13—Relative addressing.

character address pulses are always governed by the character address counter. However, the information is available from the buffer position determined by the sum of the setting of the character address counter and the relative address register. Thus, by changing the value of the relative address register, different information can be read out of the buffer with the same value of the character address counter. This feature allows side-by-side printing without duplication of wiring.

It is the relative address register which determines the starting address during a load cycle. The character address counter is always set to zero at the start of a load cycle. As during the scan cycle, the sum of the character address counter and relative address register accesses the buffer.

CHECKING

Complete checking of all information transfer and programmer control of error conditions are provided. Fig. 14 shows the checking points and methods within the system.

During load, each digit is checked for parity and invalid combinations. In addition, a check is made on the number of digits in the tape record. If any error is detected, the tape is automatically reread and an error signal emitted from the board. If an error persists after two retries, the system automatically stops, unless programmed to ignore this stop or to take other remedial action.

Parity is checked during the transfer from the buffer to the print register through the translator. Parity and invalid characters are checked again in the translator, and errors are indicated on the plugboard.

During printing, another parity check is performed with errors again indicated on the plugboard. In addition to parity checking during printing, a synchronization check is made which insures that the print position was fired at the correct time for a given character. This print check feature (Fig. 15) works as follows.

A reluctance emitter in the printer counts a row counter. This counter determines the character to be read out. A home pulse from a second reluctance emitter is

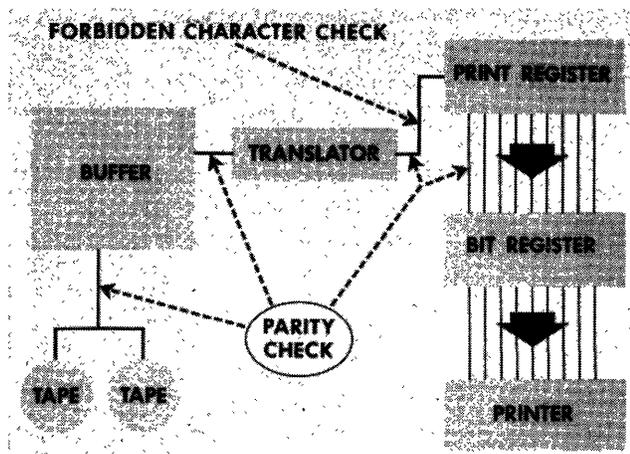


Fig. 14—System checking.

compared with the row counter when the latter's value is at zero. If the two are out of synchronization, the system automatically halts.

This is the only checking feature which cannot be ignored; the print check alarm is not available on the plugboard.

All error checking alarms that are available on the plugboard can be used to cause retries or brute-force operation which can be flagged on the printed page. Thus, in every case but one, the programmer, not the machine, decides whether an operation is to be halted or not. Oftentimes, the programmer will want to wire automatic restart procedures on the plugboard.

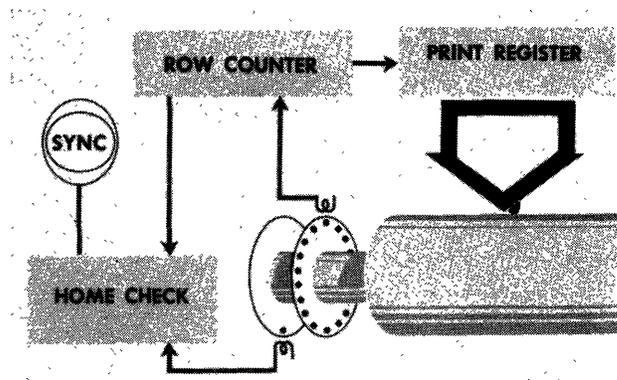


Fig. 15—Synchronization check.

Because of the versatility of the Burroughs 220 High-Speed Printer System, all operations must be programmed or wired. The operator effectively provides his own logical operations by wiring.

CONCLUSION

To summarize, the Burroughs 220 High-Speed Printer System offers a maximum of editing versatility with minimum plugboard wiring. More important, it allows swift, simple but complete rearrangement of buffer information—and eliminates the necessity for complex and time-consuming data shifting within the computer or the preparation of special print tapes.

Because of this flexibility and power, the printing problems of a wide range of applications can be solved with ease.

The ACRE Computer—A Digital Computer for a Missile Checkout System

RICHARD I. TANAKA†

INTRODUCTION

THE effectiveness of a missile system is directly dependent upon the proper assembly and subsequent reliability of its various subsystems. A supporting checkout system which enables rapid, consistent, and thorough testing of subsystems is an essential item in insuring the over-all operational success of a complex missile.

This paper describes a digital computer which is used as the central controller in an automatic checkout system. The system itself is called ACRE, for Automatic Checkout and Readiness Equipment; the computer is

referred to as the ACRE computer. The ACRE computer is, essentially, a general-purpose, stored-program digital computer; particular capabilities, however, have been emphasized to enable efficient operation of the checkout processes.

The computer and associated system are required to perform functions which can conveniently be grouped as follows:

- 1) Monitor key quantities which indicate the existence of conditions hazardous to the missile or to associated personnel.
- 2) Perform detailed checkout on a newly manufactured missile system to inspect for proper operation or to diagnose possible causes of malfunction.

† Lockheed Missiles and Space Div., Palo Alto, Calif.

- 3) Perform tests on a standby missile system at periodic intervals, to verify tactical readiness. Again, if a malfunction is detected, a diagnosis is required.
- 4) Execute, rapidly, the test sequence required prior to firing. For possibly marginal systems, a quantitative measure of operational success probability is desirable.

It is possible, of course, to meet the above requirements by utilizing manual methods or special-purpose test devices. There are obvious disadvantages, however, to both of these procedures. The described digital computer enables efficient operation in all of the modes described above and affords the following salient advantages.

Adaptability

The system can easily adapt to changes in the missile, in the test procedures, or in the criteria used for evaluating results. If required, the system can be used to test differing kinds of missiles.

Reproducibility of Test

Once a test sequence has been established, there are no unwanted variations in the test procedure thereafter. The operational advantages of a special-purpose checkout device are obtained.

Versatility

In the event that particular test results indicate a system malfunction, the computer program enables a wide variety of alternate procedures to be followed. Some factors for determining alternatives are:

- 1) the importance of the malfunctioning subsystem;
- 2) the extent of the malfunction;
- 3) the particular test mode in progress (*i.e.*, a system in the process of standby check offers alternatives not available for a system which is being checked in preparation for firing).

Possible reactions to an indicated system malfunction might include a retest against less stringent test limits, halt of the test sequence, or automatic switching to standby system components.

Automatic Sequencing

By using a stored program which includes the procedures to be followed if a malfunction is detected, the test sequences become completely automatic in their execution. This feature is particularly valuable during countdown; the test times are accurately known, and human operator errors (which could easily occur during the stress of tactical countdown) are eliminated.

Self-Diagnosis Possibilities

As a corollary to the above, the computer also can be programmed to execute routines intended to monitor the operation of the checkout system itself. The rou-

tines can be commanded to occur at periodic intervals, or may be the first step following an indicated system malfunction (to verify that the missile system, not the checkout system, is at fault).

GENERAL SYSTEM OPERATION

Fig. 1 illustrates the functional requirements for a general checkout system. The following operations are required.

Select and Control Test Stimulus

If a signal source is required as input to a subsystem or system, the appropriate signal generators are selected and adjusted to provide the proper levels.

Select Input to Missile

The test stimulus is then channeled to the proper input terminal of the system under test.

Obtain and Convert Results

The resulting output from the system (the test value) is converted, if necessary, from analog to digital form.

Perform Comparison

The test value is compared against previously established test limits.

Evaluation of Test

If the test value falls within the test limits, a "Go" result is obtained; if the test value falls outside the allowable limits, a "No-Go" result is obtained. These two possibilities determine the choice between two paths: a successful test or "Go" result causes the normal test sequence to continue; a "No-Go" result causes execution of evaluation modes as previously described.

Documentation

For later diagnostic purposes, it is extremely important that a detailed and accurate record be kept. Furthermore, the documentation process should be as completely automatic as possible.

Display

For supervisory purposes, a display panel of suitable information (test number, results, etc.) is required.

GENERAL CHARACTERISTICS OF THE ACRE COMPUTER

The ACRE computer is designed to meet the functional requirements outlined above; further, since the checkout system must operate under tactical as well as laboratory conditions, requirements of reliability, maintainability and environmental suitability are also involved. In a very general sense, the machine satisfies the latter requirements with conceptually straightforward logical organization and carefully designed, conservatively operated circuits.

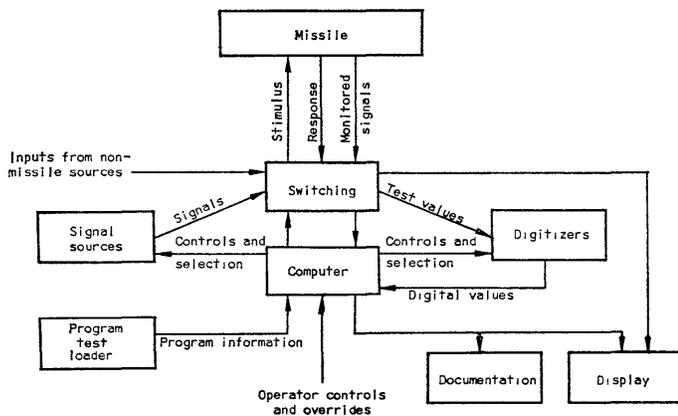


Fig. 1—Generalized checkout system.

Physical Characteristics

The specifications on physical volume enable an overall assembly oriented toward accessibility and ease of maintenance. The computer circuits, on plug-in etched card assemblies 6×6 inches in size, are arranged in horizontal chassis in groups of 25. The chassis are wired into assemblies termed "pages"; two pages, forming a "book," comprise the computer.

The front of a page is shown in Fig. 2; the interconnection wiring is on the back of each page.

Fig. 3 illustrates the flip-flop card, which contains three transistorized flip-flop circuits. Diode logic gates are fabricated on similar cards.

The computer requires approximately 40 flip-flops, and operates at 100-kc clock rate. Physical dimensions are $20 \times 20 \times 56$ inches high.

To aid in the maintenance of the computer, a special maintenance panel is attached, which includes: 1) a tester for the various circuit cards; 2) a memory simulator which allows static test of the computer with the magnetic memory disconnected; 3) controls which allow single clock operation and manual setting of flip-flops; 4) indicators attached to the individual flip-flops; and 5) voltage level switches for marginal checking.

Depending upon external system requirements (*i.e.*, tie-in with missile subsystems, various converters, etc.), as many as 30 additional flip-flops may be required. This quantity varies directly with the desired system performance requirements.

Memory

The ACRE computer utilizes a rotating magnetic memory with an addressable capacity of 3904 words, 24 bits in length. These are organized into 60 channels, each with 64 words, and four faster access channels of 16 words each.

The memory also contains a nonaddressable 64-word channel which is used for automatic storage and output tape buffering for the later described documentation capability. Further, the clock pulses, sector reference information, and the various one-word registers are supplied by the magnetic memory.

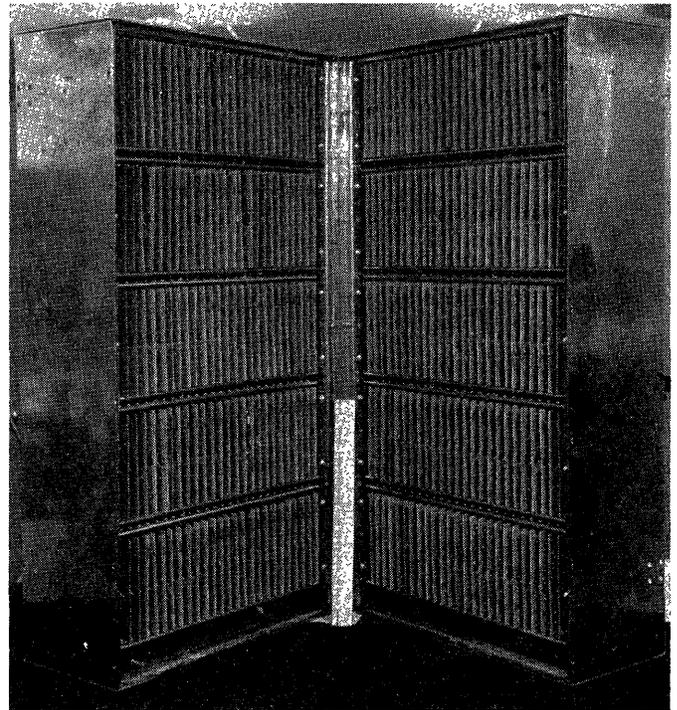


Fig. 2—ACRE book: front view.

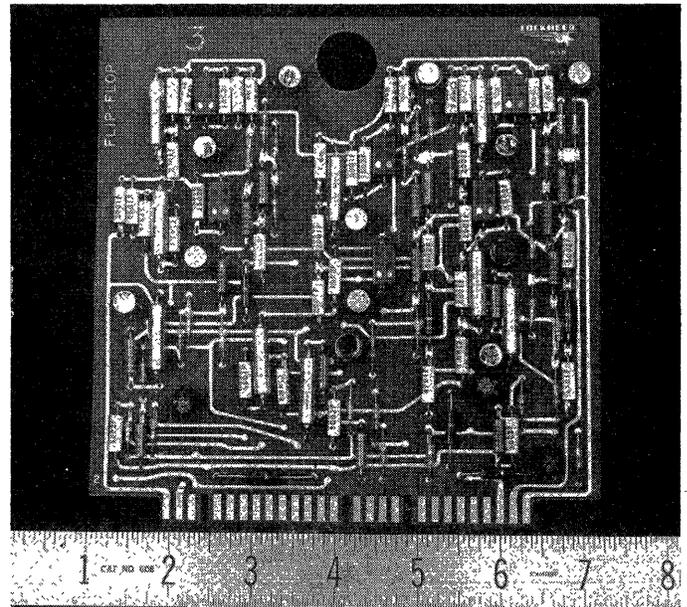


Fig. 3—ACRE flip-flop board.

The six 24-bit registers on the memory are: 1) the *A* register, the main arithmetic register in the machine; 2) the *D* register, an auxiliary arithmetic register; 3) the *B* register (Index register), used both for address modification and for tallying purposes; 4) the Order Counter, which specifies the address location of the next order to be executed; 5) the Address register, which stores the address portion of an order obtained from memory; and 6) the Documentation or *X* register, which receives information either from selected arithmetic registers or from an external keyboard, for eventual storage on magnetic tape.

Logical Organization

An Order Counter is used to specify the normal program sequence; the order structure is single address. An order word contains a 5-bit command code, a 12-bit address (designated as m), and single bits each for parity, B -reference, documentation, and spacer.

Numerical information is represented as sign and magnitude (fractional binary), with single bits for parity and spacer.

The machine has a total of 25 commands; these can be categorized loosely as 5 arithmetic commands, 9 internal transfer commands, 6 control commands (jump to location m for various conditions), and 5 special commands which relate directly to the checkout requirements. (The latter 5 are included in the later description of special commands.)

SPECIAL COMPUTER FEATURES

Miscellaneous features, which define some of the capabilities of the ACRE computer, are described below.

Parity

To detect malfunctions in the transfer and retention of information, word contents undergo automatic parity check during transfer. If a word is modified by arithmetic processes, a new parity bit is generated and inserted in the word. Parity error causes the computer to halt; the operator is notified by a suitable alarm indicator.

Switching Matrix

The ACRE system incorporates a switching matrix which is directly controllable by the computer. At present, a relay switching matrix is used, since the speed of the relay network is sufficient to meet existing system requirements. (Also, relays enable convenient handling of a wide range and class of test variables without elaborate preprocessing.) If desired, a solid-state switching network can easily be substituted.

B Register

As previously mentioned, an Index register is provided for automatic address modification and for tally uses. A one-bit in the B -reference position of an order word causes the contents of B to be added to the address portion of the order prior to execution.

For tally purposes, commands which increase or decrease the contents of B are provided. The decrease B command allows branching on the sign of B . The logic allows this command to be used either for branching each time until the final traversal of a loop, or for the opposite case of not branching until the final traversal.

Because of the similarity of many of the test sequences, the B register enables a significant saving in storage required. Since, for tactical operation, the program and all associated parameters are stored permanently on the memory, the inclusion of a B register (at a

cost of two flip-flops, associated logic, one read and one write amplifier) contributes much more than programming convenience alone.

Documentation

The documentation feature enables automatic recording of all information pertinent to the checkout processes. Normally, the information to be stored on the output tape is specified by the program and hence is documented automatically; however, to enable the operator to insert additional information, input from a keyboard can be documented during intervals when the computer is idling.

During computation, a one-bit in the documentation code position of an order word causes information appropriate to the command to be documented (two examples: the sum for an addition command; the word transferred, for any of the various transfer commands).

This information progresses from the one-word X register into the 64-word special storage channel. When the channel is filled, the entire channel contents are transferred automatically to an output magnetic tape.

The output tape recorder uses magnetic tape 1 inch wide, on 8-inch reels, each with a total capacity of 5 million words of storage (each word 24 bits long). Cross-channel and longitudinal parity are automatically generated and recorded.

Display

The progress of each test sequence is indicated by a panel of display lights. The display lights are addressable through the switching matrix; the operator receives a direct indication of the sequence in progress, and of test results obtained.

INPUT-OUTPUT

Program Test Loader

The magnetic memory is filled by the Program Test Loader, a demountable input device using seven-channel, punched paper tape. The Loader uses channel parity and cross-channel parity to check information pickup; as a further safeguard, the memory contents are automatically read back and verified.

For laboratory use, the Loader can be used as a normal input reader.

For tactical operation (where the test sequences presumably have been generated, tested, and then stored in their entirety in the memory), the Loader is disconnected. Disconnecting the Loader insures that no inadvertent modification of memory contents can occur. The stored routines are so designed that the operator is required only to monitor the results, or, in a few instances, to initiate sequences by push button control.

External Controls

The operator is provided with a minimal selection of controls. These include Start, Stop, the documentation

keyboard, and Test Selectors. The latter are individual buttons which set the Order Counter to preselected configurations, enabling convenient manual selection of particular tests.

For maintenance purposes, controls affecting physical conditions, *e.g.*, single clock, marginal checking voltages, drum simulator levels, flip-flop set and reset switches, are provided. Further, a Function Switch enables selection among three modes of operation: Normal Operation, Single-Order Execute, and Breakpoint Operation (the last-named causes halt after execution of each command accompanied by a breakpoint code-bit).

Output Display

A numerical display device can selectively display the contents of the Order Counter, the *A* register, or the Documentation Register. The Order Counter display consists of four octal digits; the other two registers appear as sign and four decimal digits.

As previously mentioned, test indicators, primarily controlled by the switch matrix, are provided. These indicate, for example, status of various missiles, test in progress, subsystem under test, test results, etc.

DISCUSSION OF COMMANDS

The commands available in the ACRE computer have been mentioned in the description of logical organization. Seven of the 25 commands directly related to the checkout process are discussed in detail below.

Compare (cpr)

The contents of the *A* register are compared with the contents of *D*; if *A* is greater than *D*, the computer will transfer control to the order found in memory position *m*. Both registers remain unchanged. The comparison process is basic to the operation of the checkout system, and is required frequently. Hence, although the comparison, and transfer of control, can be performed by a subroutine, it is convenient to have the process available as a command.

Switch (swc)

Twenty bits in position *m* are transferred into the Switching Address Register. At the end of the transfer process, a signal to initiate switching is generated. The switching matrix proceeds to establish the connections specified by the 20 bits; the computer is free to proceed with the program. An interlock is provided, so that if switching is still in progress when a new switch operation is commanded, the computer will idle until the previous operation has been completed.

Select (slt)

The four least significant digits in the address code *m* are used to set the four flip-flops of the Select Register. The register contents then determine which item of external equipment is to be affected by subsequent commands.

Conditional Adjust Test Equipment (caj)

An item of test equipment, designated by the Select Register, receives the 12 bits of the address *m*. The bits are made available only after any previously commanded switching operation has been completed.

The selected equipment could be a signal generator whose output is adjusted by the 12 bits; it also could be an analog-digital converter, whose scale adjustments or whose turn-on signal is derived from the 12 bits. The switch interlock insures that switch connections pertinent to the operation of the selected equipment have been made before the equipment is activated.

Unconditional Adjust Test Equipment (uaj)

This command is similar to "caj" above, except that no interlocks with the switch operation are provided. This command is used when the desired connections are known to be made. The unconditional aspect insures that a simultaneous switching operation related to another test cannot inhibit an operation initiated by this command.

Bring Test Equipment Output (bte)

The output from an item of test equipment, designated by the Select Register, is read into *D* and also into position *m*. This command is used, for example, to obtain the output of an analog-digital converter for subsequent comparison against programmed limits.

Halt (hlt)

The Halt command causes the computer to idle. Simultaneously with the Halt, the Select Register receives the four least digits of the address code *m*. The machine will start automatically when one of the 16 signal sources, selected by the contents of the Select Register, turns on. This allows the checkout system to wait for the establishment of various conditions in the missile before proceeding with a test.

CONCLUSIONS

The ACRE system has demonstrated the feasibility of assigning to a digital device of the stored program class all of the central control requirements for a missile checkout system. The advantages inherent to a stored program have contributed significantly to the derivation and application of test sequences at all levels of missile operation, from manufacturing to field readiness to prelaunch countdown. The tests, in turn, are of paramount importance in insuring the highest possible degree of successful missile operation.

ACKNOWLEDGMENT

The writer wishes to acknowledge contributions by B. D. Leitner, P. W. Cheney, and L. D. Healy to the computer logical design. Circuit design and computer assembly were the responsibility of various members of the Lockheed Computer Research Department.

IBM 7070 Data Processing System

J. SVIGALS†

IN 1953, International Business Machines Corporation (IBM) introduced its first production model of a large-scale computer—the IBM 701 Data Processing Machine. Since that time, the company has engineered and delivered a number of major data processing systems, each designed to meet certain business and scientific requirements.

These systems have included the 305 Ramac, 650 Basic, 650 Tape, 650 Ramac, 702, 704, 705, 705 III, and 709. In addition to these computer systems, the completely transistorized 608 Calculator has been in productive operation since 1957.

Each of these systems has extended and refined the growing body of practical knowledge of both the versatility and limitations of data processing equipment. The result has been an increase in the total productive time of the machine installations and more efficient use of these machine hours.

More important, the application of these systems to actual working problems has produced thousands of specially trained personnel in the business world as well as within IBM itself.

The new IBM 7070 Data Processing System is a product of the ideas and expressed needs of this experienced group. It is designed to serve as a partner of the man-machine-methods team of modern business.

The purpose of this paper is to describe the general organization of the IBM 7070. This description will include a discussion of the special features of the system and its physical and electrical characteristics.

GENERAL ORGANIZATION

The IBM 7070 Data Processing System combines advanced engineering design, based on high-speed, solid-state components, with an equally modern machine organization. The system is capable of efficient solution of both commercial and scientific applications. The IBM 7070 Data Processing System spans a wide range of capacities and features. These include punch card, magnetic tape, and magnetic tape/Ramac configurations.

The punch card IBM 7070 system consists of the central computer, one to three punch card readers, and one to three card punches or printers. The card readers operate at a speed of 500 cards a minute; the card punches operate at a speed of 250 cards a minute; and the on-line printers operate at a speed of 150 lines per minute. In the card system configuration, all card input and output devices can operate completely simultaneously with each other and with internal computation.

To achieve a tape system configuration of the IBM 7070 Data Processing System, two tape channels are

added to the computer configuration. Each tape channel is capable of operation simultaneously with the second tape channel and internal computer operation. Each tape channel may have from one to six magnetic tape units connected. These may be any combination of regular-speed (41,667 characters per second) or high-speed (62,500 characters per second) magnetic tape units. This allows for a total of 12 magnetic tape units, any two of which will operate simultaneously with computation.

A tape Ramac IBM 7070 configuration is obtained by the addition of one to four Ramac file units to the system. The Ramac units are interconnected so that they may be operated through either tape channel. This allows completely simultaneous operation of two Ramac read and/or write operations in a manner equivalent to that of the magnetic tape units. Each Ramac file provides three access arms. Experience with the IBM 650 Ramac systems indicates that access to these files can be achieved effectively in zero time. This is accomplished by seeking ahead for the next record while a previous record is processed.

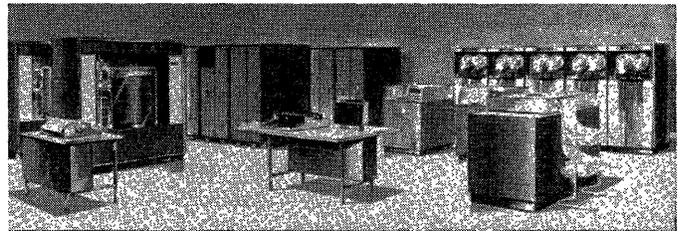


Fig. 1—The IBM 7070 Data Processing System.

A typical IBM 7070 system is shown in Fig. 1. Included are the following units:

Console: This is a separate unit which includes the console typewriter and a small operator's panel. The console unit is designed to simplify and expedite the operator's task and to insure maximum productive machine time. The typewriter is the principal operator's tool. It replaces many of the indicator lights and control switches of previous data-processing machines. Operator error is minimized by the computer's ability to audit operator commands through a stored program and by a printed record of all data entered and emitted through the console typewriter.

Magnetic Tape Units: The magnetic tape units are seen to the right, rear of the picture. Two types of units are available. The 729 II reads or writes tapes at a rate of 41,667 characters per second. The Model IV reads or writes tapes at a rate of 62,500 characters per second.

Card Reader: Immediately in front of and to the left of the magnetic tape units is a card reader. This unit

† Regional System Dept., IBM Corp., Los Angeles, Calif.

operates at a rate of 500 cards per minute with format control by means of a control panel mounted on the reader. Data from a full 80-column punched card may be transferred into the computer simultaneously with internal computer operations. The card reader is equipped with a front attended tray feeding hopper and stacker. As many as three card readers can be utilized for card input. Selected cards may be offset in the stacker.

Card Punch: To the right of the card reader is the card punch. This unit operates at a punching speed of 250 cards per minute with format control by control panel wiring. Front attended hopper and stacker are used. Selected cards may be offset in the stacker. As many as three card punches can be utilized for card output.

Printer: The printer is located to the right front of the picture. This unit operates at a speed of 150 lines per minute with format control provided by the control panel. The printed line output consists of a span of 120 characters spaced ten to the inch.

Central Processing Unit: This unit is seen to the left rear. It contains most of the system electronics and consists of the following elements:

- 1) Arithmetic registers and core memory,
- 2) Indexing hardware,
- 3) Space for optional floating decimal arithmetic (with automatic double precision operation),
- 4) Magnetic core memory of five or ten thousand words,
- 5) Two data channels, code translators, data registers and controls for magnetic tape and Ramac storage units,
- 6) Buffers and controls for card input, card output, and the printers.

Disk Storage Units: To the left of the picture are the disk storage units. These units consist of a stack of large disks which magnetically hold up to 12,000,000 digits each. Information is read from and written onto these disks by an access mechanism containing a magnetic recording head. This mechanism moves rapidly to any disk in the file. Three access mechanisms for each storage unit are provided to minimize access time by overlapping the "seek" operation. Up to four disk storage units can be utilized by the 7070 system, providing a total storage capacity of up to 48,000,000 digits of rapid, random access memory. These units are attached to the system by the same two channels that connect the magnetic tape units to the system. Each of these channels is linked to a disk storage unit by program control, and allows any combination of simultaneous read/write/compute.

Manual Inquiry Station: The manual inquiry stations are shown immediately in front of the Ramac disk storage units. These units permit fast interrogation of data stored in the computer core storage, in the disk storage unit, or on magnetic tape. The station consists of a

		Bit Code				
		0	1	2	3	6
Decimal Digit Value	1	■	■	□	□	□
	2	■	□	■	□	□
	3	■	□	□	■	□
	4	□	■	□	■	□
	5	□	□	■	□	□
	6	■	□	□	□	■
	7	□	■	□	□	■
	8	□	□	■	□	■
	9	□	□	□	■	■
	0	□	■	□	□	□

Fig. 2—Two-out-of-five code.

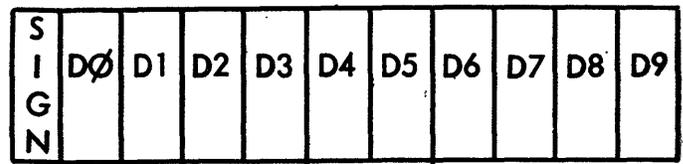


Fig. 3—Word format.

special typewriter equipped with a solenoid-driven keyboard and transmitting controls. A 16-channel punched mylar tape provides format control. Up to 10 manual inquiry stations can be attached to the system, through two buffers. The stations can be connected to the system by cable up to 2500 feet from the central processing unit.

MACHINE CHARACTERISTICS

Bit-Code Structure

Information is represented by a two-out-of-five code. The total number of possible combinations is 10, one for each numerical digit. As shown in Fig. 2 the bit positions are designated 0, 1, 2, 3, and 6. Each of the digits 1 to 9 is made up of two bits whose sum equals the number in question. Zero is designated by the one-two combination. Alphabetic information is represented by a dual-digit code.

Data Storage

A word in the machine code is composed of 55 bits. Fig. 3 shows a word which consists of 10 digits plus sign, or five alphabetic characters with an alphabetic sign. In each case, automatic recognition of the sign position indicates to the computer whether the information is alphanumeric or numeric and all operations are performed automatically according to the sign.

Validity Checking

All information transfers to and from storage within the 7070 computer are tested to insure that each digit has two "one" bits, no more and no less, for each five-

bit position. Because the checking is for an exact number of "one" bits for every digit, this type of coding structure affords a complete and consistent self-checking of data flow.

Data Transmission

An important feature of the 7070 is parallel transmission of data to and from core storage. An entire word, including sign, is moved all at once. A channel for parallel bit transmission consists of 55 lines, one for each bit in each of the 10 digits and the sign position. This enables a word in core storage to be moved in 6 μ sec to or from core storage. Data are transmitted between the core registers within the computer at a 4- μ sec rate. All information is transmitted in parallel with the exception of data transmitted to and from the magnetic tape units. Parallel transmission is represented by the parallel lines in Fig. 4. The accumulators all have serial paths connecting them to the core adder. There also is a serial data path to and from the input/output synchronizers.

Arithmetic Operations

The arithmetic unit of the IBM 7070 contains three accumulators, each with a capacity of 10 digits and sign. In addition to the accumulators, the auxiliary register and the arithmetic register are each capable of containing ten digits and sign. The interconnection of these units is shown in Fig. 4.

Addition

An amount to be added to an accumulator is brought to the arithmetic register. The number in the accumulator is sent to the auxiliary register. The content of these two registers is added, one digit at a time, and as the result is developed by the adder, it is brought to the arithmetic register. At the conclusion of the operation, the result is sent from the arithmetic register to the designated accumulator.

Arithmetic Timing

The duration of an arithmetic operation is determined by the size, in digits, of the factors. In an add instruction, for example, it is determined by the size of the field or by the significant digits in the accumulator, whichever is greater. Arithmetic operations take only the amount of time needed to perform the actual computation; there is no time wasted in accumulating a full ten-digit number for each arithmetic operation. The addition times for fixed point operation are shown in Table I. These figures include access time for the instruction and an operand.

TABLE I

Length of Operand	Execution Time
1, 2, or 3 digits	48 μ sec
4, 5, or 6 digits	60 μ sec
7, 8, 9, or 10 digits	72 μ sec

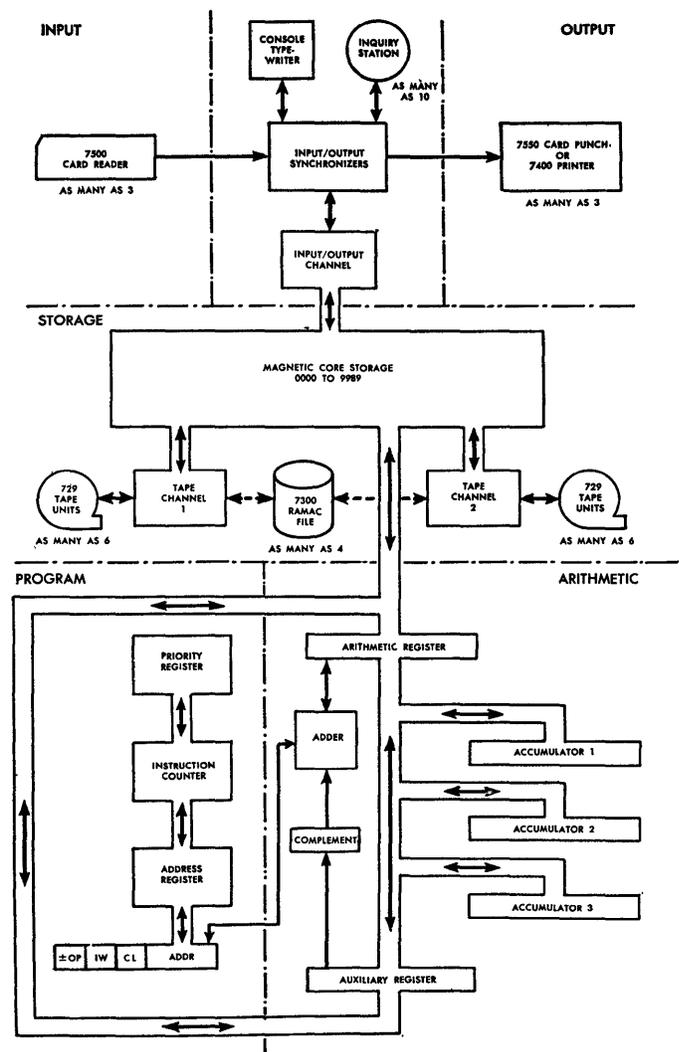


Fig. 4—IBM 7070 data-flow schematic.

Instruction Format

Each instruction in the program consists of 10 digits and sign. The digit positions are numbered 0 through 9 (left to right) as shown in Fig. 5. For most operations these digits are utilized as follows:

Sign and positions 0-1:	operation code
Positions 2-3:	indexing word
Positions 4-5:	control digits
Positions 6-9:	address

Operation Code: The sign and first two digit positions provide for a maximum of 200 different operation codes of which 120 are currently used. In addition, some of the operation codes have multiple functions. These are accomplished through different values which are placed in digit positions four and five to further define the operation. For example, in a card operation, position four specifies a particular input or output unit. Position five defines the specific card operation such as read, punch, or print.

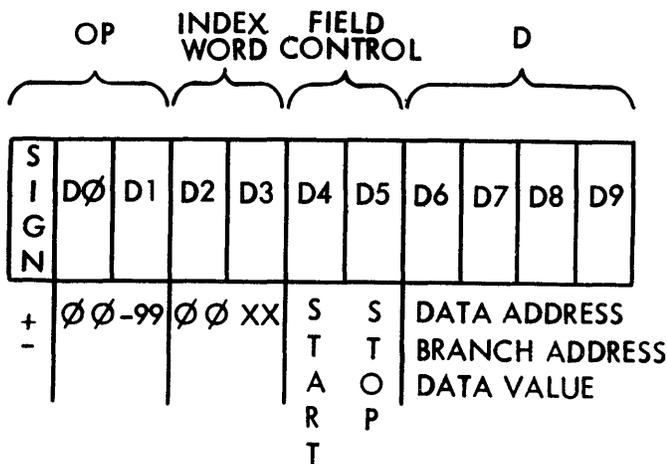


Fig. 5—Instruction format.

Index Word: Positions 2 and 3 of the instruction specify the indexing word to be used. There are 99 index words in the magnetic core storage, each of which contains a 10-digit number with sign. They are stored in memory locations 0001–0099. The indexing word portion of a program step determines which of these 99 index registers will be applied; 00 means no indexing.

Control: In all of the arithmetic instructions, any portion of a word can be processed as easily as the entire word. Positions 4 and 5 of an instruction determine the part of the word that will be used. The digit in position 4 denotes the left end of the field. The digit in position 5 specifies the right end of the field. This is called “field definition.” The field definition feature means that several fields with like signs can be stored in a single word with no inconvenience to the programmer in processing an individual field. It should be noted that in all arithmetic operations using field definition, information extracted from a word is shifted automatically to the right when placed into the designated accumulator. In storage operations, a number of digits equal to that specified by the field definition digits is extracted automatically from the right-hand portion of the accumulator, shifted left to the designated location of the word, and then inserted in the word without disturbing the remaining digits. Field definition operations occur without any additional execution time.

Address: The address portion of an instruction, positions 6–9, may refer to the storage location of the data or the location of the next instruction.

Indexing: As stated previously, memory locations 0001–0099 also may be used as index registers. When a word is used as an index register, the digits have the following value:

Sign position: This specifies whether the actual value of the indexing portion shall be added to or subtracted from an address during an index operation.

Digits 0–1: Not used.

Digits 2, 3, 4, 5: This is the indexing portion of the index word. Together with the sign of the index

COMPARISON CHART 729 TAPE UNITS		
	Model II	Model IV
Coding	Even number out of 7 (C, B, A, 8, 4, 2, 1)	Even number out of 7 (C, B, A, 8, 4, 2, 1)
Automatic validity check from tape on writing operation?	Yes	Yes
Character density	200–555 characters per inch	200–555 characters per inch
Passing speed	75 inches per second	approx. 112.5 inches per second
Character rate (maximum)	41,667 alphanumeric characters per second	approx. 62,500 alphanumeric characters per second

Fig. 6—Magnetic tape characteristics.

word, this portion is added to the data address of the instruction. All instructions are indexable.

Digits 6, 7, 8, 9: This is the fixed portion of the index word and may be used by the programmer to store constants, decrements, increments, or limits.

Magnetic Tape Characteristics: The IBM 7070 Data Processing System provides two types of magnetic tape units. These are the IBM 727 Model II or the 729 Model IV. Data from magnetic tape to the system are read into magnetic-core storage, and a record is written on tape from the core unit. Any group of locations in core storage can be used for these purposes. The tape control unit provides two channels, each connecting as many as six tape units with the main system. The total of 12 magnetic tape units can be used in any combination. The comparative characteristics of the two tape unit models are shown in Fig. 6. The primary difference between them is the greater density of bit storage and passing speed of Model IV. Both units provide high-speed rewind at a maximum speed of 500 inches per second.

Successive records on tape are separated by a blank space called the interrecord gap. Every tape-read instruction causes an entire record to be read: the reading is stopped by the interrecord gap. The size of a tape record has no limitation except the capacity of core storage used in transferring the information to or from the tape. When reading from the tape, even this restriction is not rigid if data from only a portion of the tape record are needed. The program defines the number of words and locations that can be read into core storage. Any information from the tape in excess of that amount is not accepted by the machine.

The coding structure used on magnetic tape is a seven-bit alphanumeric code. This code is identical to that used by other IBM data-processing tape systems. Every character must have an even number of bits, and all are tested for this in every tape read or write operation. In addition, there is a horizontal check of each record. As the record is written on tape, a horizontal check char-

acter is created to make the number of bits even in each of the seven channels. The check character itself must have an even number of "one" bits to make the over-all total of bits even.

The IBM 729 tape units have an additional checking feature on tape-writing operations. The reading head reads the tape just after it is written. This function is illustrated in Fig. 7. This check is performed automatically and provides immediate verification of the actual tape record.

NEW MACHINE CHARACTERISTICS

Record Definition Words

An important new concept introduced by the IBM 7070 Data-Processing System is that of record definition words. These words define the first and last addresses of a block of data stored in core storage. One of these words is placed in storage for each block. The words may be used singularly or in tables of record definition words, as shown in Fig. 8. In each case, the last word in the table is signified by a minus sign. Record definition words are used for all data movement of more than one word at a time. This includes disk storage read and write, core-to-core block transmissions, movement of input data from the input buffers and output data to the output buffers, inquiry and reply information, and others.

Scatter Read/Gather Write

A powerful programming tool in 7070 operations is scatter read/gather write. This is illustrated in Fig. 9. A single record read from tape can be divided into as many parts as desired by the programmer. These parts are distributed to the different blocks of core storage as the tape is read. The figure shows how an inventory record is separated by field and category during the tape reading procedure. This operation is controlled by the table of record definition words shown in Fig. 8. This feature applies to writing on tape as well as reading from it. It enables the program to gather data from various blocks and automatically assemble them into one tape record.

Record Mark Words

Words containing the record mark character give an additional flexibility to the scatter read/gather write feature. Detection of the record mark word automatically denotes the end of a block in a scatter read/gather write operation. In scatter read, a tape record will not continue to fill the storage block if a record mark word is read into the storage area. Instead, it goes to the next block and begins to fill it with the characters immediately following the record mark word. In tape writing, a record mark word in core storage will signal the system to stop moving data from that block and start sending

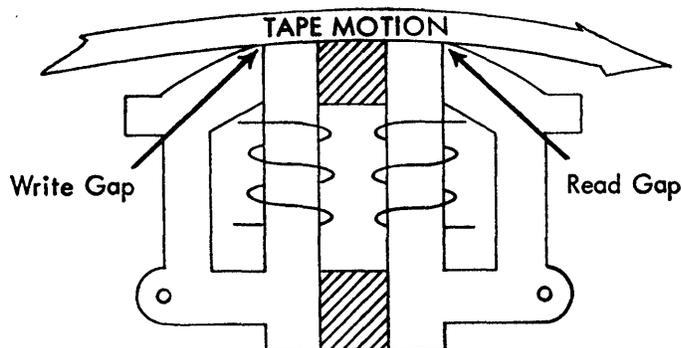


Fig. 7—Two-gap read-write head.

S 0 1 2 3 4 5 6 7 8 9			
LOCATION	SGN	STARTING ADDRESS	ENDING ADDRESS
3211	+00	2371	2380
3212	+00	0021	0021
3213	+00	4175	4194
3214	+00	1716	1720
3215	+00	3617	3621
3216	+00	0104	0110
3217	+00	4401	4425
3218	+00	2803	2803
3219	+00	3913	3919
3220	+00	0305	0307
3221	-00	4853	4877

Fig. 8—Record definition words.

from the next block. Tape read or write instruction determines whether the record mark word will be operative. If not, a record mark word is treated as a normal alphabetic word.

Zero Elimination

When a numeric word in storage is written on tape, as many as five high-order zeros can be automatically eliminated. The sign of a word is combined with its unit position digit. When the tape is read, the presence of a sign combined with a digit indicates the unit position of a word. The core storage word will be filled automatically with zeros in the high-order positions to replace those eliminated during the write operation. The functions of record mark word and zero elimination are completely automatic and require no additional execution time other than the time normally required to read and write magnetic tape at full tape rate. These functions result in a variable word and variable record size on magnetic tape.

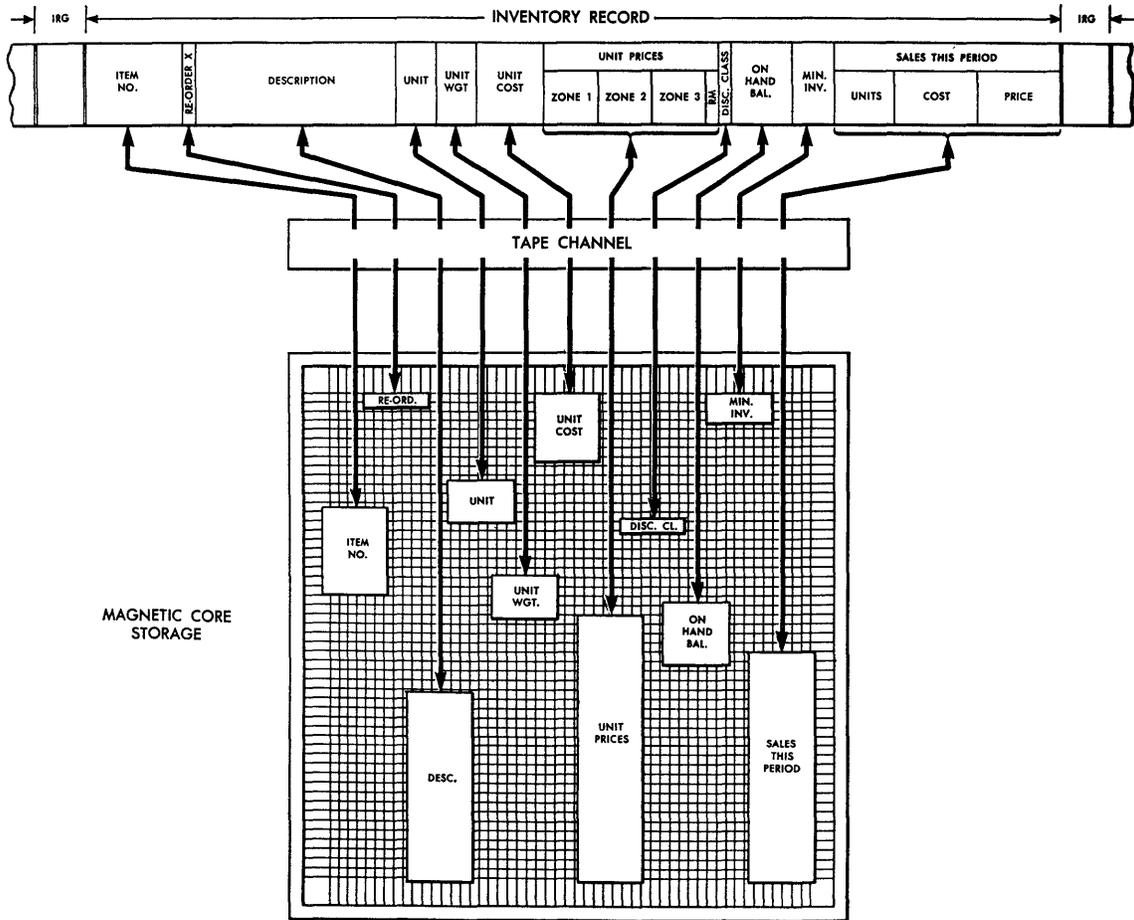


Fig. 9—Scatter read/gather write example.

CHANNEL CONTROL

The magnetic tape system of the 7070 has two channels, each connecting as many as six tape units with magnetic core storage. The channels also connect to as many as four disk storage units. These disk units can be programmed to use either channel, whereas a tape unit can use only the one connecting it to the central processing area.

The primary purpose of the additional channel is to allow the IBM 7070 to perform two operations simultaneously—reading two tapes, writing two tapes, or reading one and writing one. Each control channel performs one of the two tape read/write operations; while these simultaneous operations are taking place the program can continue. The function of the tape channel is shown diagrammatically in Fig. 10.

Information is moved between core storage and disk or tape storage by the transmission registers. The purpose of the transmission registers is to change the type of transmission from serial to parallel, or vice versa, and to synchronize the speed of the tape or disk with the speed of magnetic core storage. Data must move to and from tape and disk storage serially, but they are always

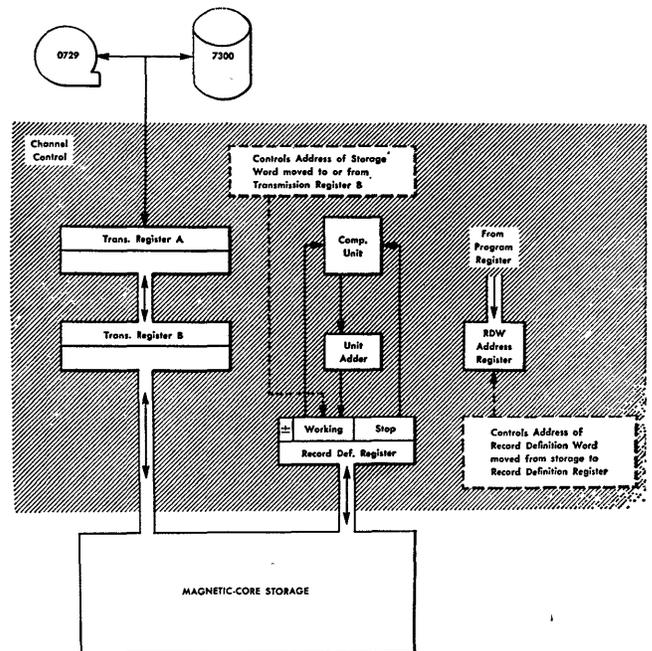


Fig. 10—Tape/disk channel control schematic.

sent to and from core storage in parallel. In reading tape or disk storage record, the characters are read serially into transmission register A until the ten-digit positions and sign position are filled. At this point they are sent in parallel to transmission register B. Register A then starts to fill up serially again with the next characters on the tape or disk record. The contents of register B are sent in parallel to the magnetic core storage location specified. In tape or disk storage write operations, data go in parallel to transmission register B, then in parallel to transmission register A, and serially to a tape or disk storage unit.

The storage locations are controlled by the record definition register, one of which is used for each tape channel. The starting (or working) and stop addresses in the registers are obtained from a table of record definition words stored in the memory. As a word is read from tape, the content of the working address is increased by one and compared with the stop address. After each word is read from tape, these addresses are compared. If they are unequal, the working address is increased by one and the next word is moved from tape to the new storage location specified.

This process continues until the working and stop addresses become equal, or until a record mark word is read. At that point, the last word is transferred, and the sign position in the record definition register is tested. If it is plus, the record definition word address register is increased by one, and a new record definition word is brought in from storage to designate a new block of words. When the sign of the record definition register is minus, this means the last record definition word in the operation has been reached and the operation is completed.

The operations described here are accomplished automatically within the record read or write time. The combination of zero elimination, record definition words, and record mark words provide a completely automatic method of obtaining variable word size, record size, and block size on magnetic tape.

Segment Forward or Backward Space per Count

These are new commands designed to allow automatic spacing over a series of tape records. The spacing operation can be performed with the tape moving in a forward or reverse direction. The tape is moved at full speed until one or several segment marks are sensed. The number of segments skipped is defined by a record definition word. The segment marks are recorded on the tape under program control and can be used for any type of demarcation required, such as a fixed number of tape records or a change in the file sequence control.

Table Look-Up Operations

There are three types provided in the IBM 7070. They are 1) equal or high, 2) equal, and 3) lowest. In equal or high, a sequential table is searched for the first entry that is equal to, or higher than, the search criteria.

Table look-up equal searches a random table for an entry equal to the search criteria. Table look-up lowest searches a random table for the entry with the lowest search criteria.

Floating Decimal Arithmetic

An optional computing feature of the IBM 7070 is the use of floating decimal operations. This includes automatic floating point double precision operations which provide eight high-order and eight low-order Mantissa digits, each with its correct, two-digit characteristic.

Automatic Priority Processing

This new programming feature makes it possible to process more than one program during the same period. This procedure eliminates time lost in waiting for input/output operation to be completed, since one program or another is constantly functioning. One of the programs, called the main routine, has a comparatively large number of program steps. The others, called the priority routines, have relatively few instructions, but involve almost continuous use of card reader, card punch, printer, tape unit, or disk file.

The main routine functions normally, while the tape, disk storage, or input/output unit in the priority routine is operating. These operations may include reading a card, punching a card, reading tape, writing tape, seeking a disk file record, reading a record file, or writing a file record. When any one of the preceding operations is completed, the main routine is signalled automatically. The priority routine carries out its program, stops its input/output or storage unit, and releases priority. The main routine then takes up exactly where it left off. It is possible to have more than one tape, disk storage, or input/output unit operating on a priority basis during the main routine. However, only the main routine can be signalled for priority; it is not possible to do this to a priority routine. If a second priority is ready while the first one is in progress, it will wait until the first one is completed. The main routine is resumed only when there are no priority routines waiting.

Priority is determined by the setting of the stacking latches, shown diagrammatically in Fig. 11. There are stacking latches for the card input/output units, one for each tape unit, and one for each of the three read/write heads in each disk file. Each program step in the main routine tests any or all of these latches to see if the priority routine is ready. These tests are made without any delay in the program steps of the main routine.

There are four types of priority: card, tape, disk file, and inquiry. Card priority is caused by the completion of a card read, card punch, or print operation. A tape priority routine is initiated by the completion of a tape read or write operation, and disk file priority is started at the conclusion of a disk file read, write, or seek. A manual inquiry automatically initiates an inquiry priority routine. In all cases, the program may test, set, or reset any latch. This allows complete control of priority

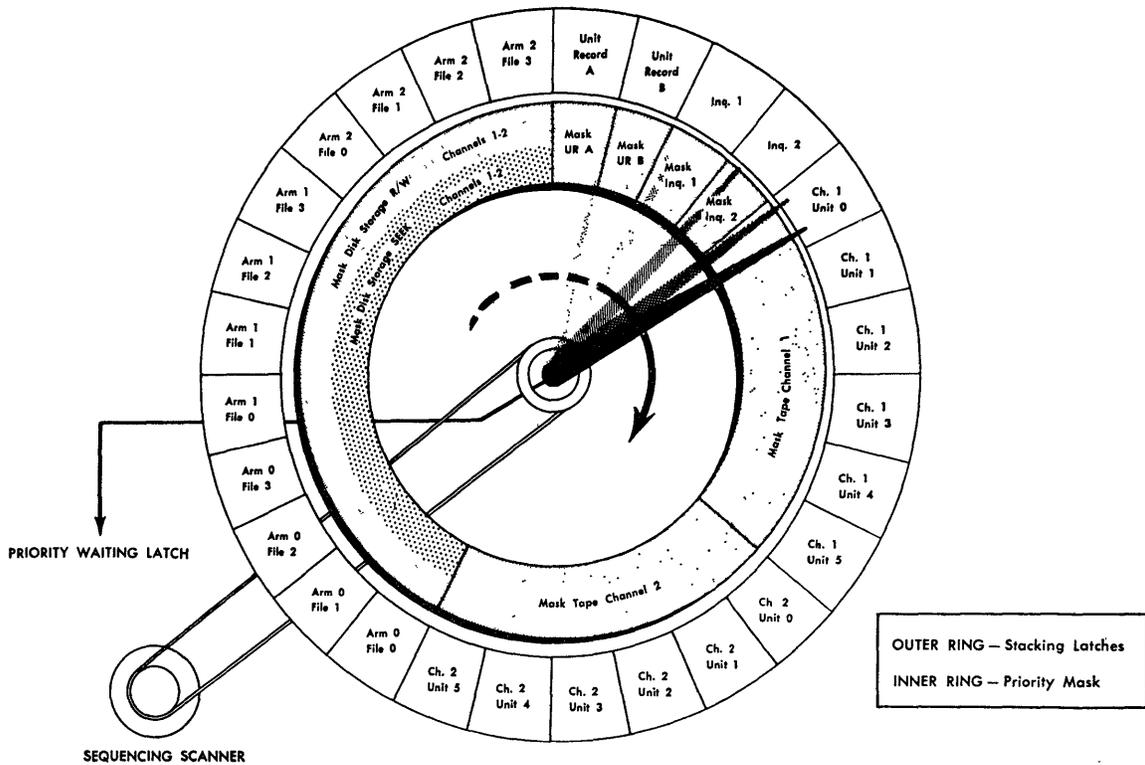


Fig. 11—Stacking-latch scanning schematic.

processing by the program although the entire scanning process is continuous and automatic. In any tape operation requiring special attention, an automatic tape priority is initiated. This includes, for example, operations in which an end-of-tape is recognized.

Physical and Electrical Characteristics of the IBM 7070 Data-Processing System Printed Circuit Cards

The system electronics is housed in a group of sliding gate cabinets, one of which is shown in Fig. 12. These cabinets are designed to accommodate printed circuit cards of the type shown in Fig. 13. These printed cards, which mount up to six transistors, are designed for automatic fabrication. Approximately 14,000 cards are employed in the basic 7070 system shown in Fig. 1. A total of 30,000 alloy junction germanium transistors and 22,000 point contact germanium diodes are used in this system.

Chassis intercard signal wiring is provided by jumper wires which are connected to the card socket by wire-wrap connections. Wiring is accomplished automatically by wire-wrap machinery. This is shown in Fig. 14. Distribution of power supply voltages to the transistor cards is accomplished by printed circuit strips. These features enable a major portion of the electronic system to be fabricated by automatic equipment.

Physical Planning Requirements

The use of a fully transistorized system has resulted in a reduction of the physical facilities needed to sup-

port the IBM 7070 Data-Processing System. Table II compares the savings for the IBM 7070 system with those of a comparable IBM 705 II.

TABLE II

Floor space reduced up to	50 per cent
Power reduced up to	58 per cent
Air conditioning reduced up to	58 per cent
Weight reduced up to	50 per cent

These reductions will result in substantial savings during the physical installation phase of each IBM 7070 system. In addition, existing facilities will accommodate more powerful IBM 7070 systems within physical facilities now using nontransistorized computers.

Library and Programs for the IBM 7070 Data-Processing System

An extensive library of programs is provided with the IBM 7070 system. These reduce the amount of time and effort normally required in a computer installation. The major automatic and library programs available are as follows:

Autocoder: This is a commercial assembler program based on comparable programs used with the 705 systems.

FORTTRAN: This program is a complete compiler for scientific problems. The FORTRAN language is identical to that currently used on the IBM 650, 704, 705, and 709 systems.

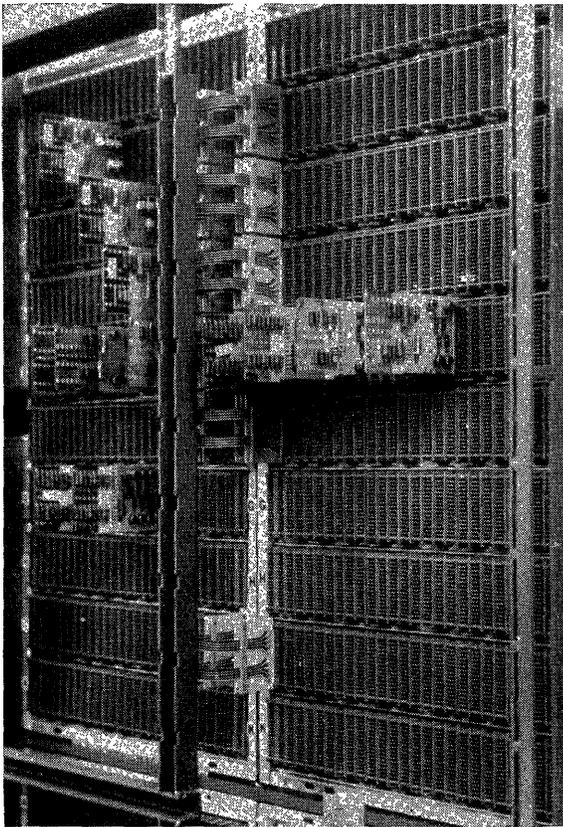


Fig. 12—Sliding gate housing showing printed circuit cards.

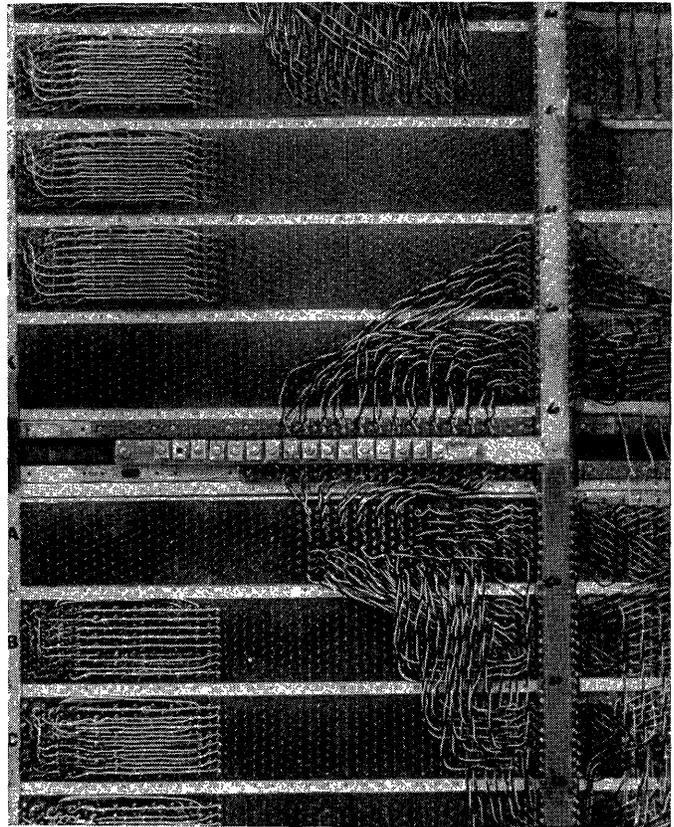


Fig. 14—IBM 7070 sliding gate housing showing wire-warp intercard wiring.

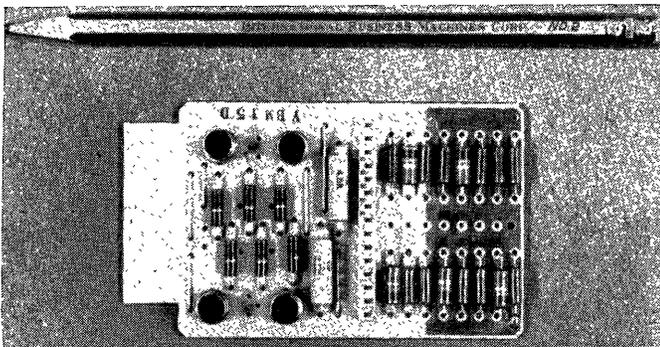


Fig. 13—Printed circuit card.

Sort and Merge: This program provides from two- to five-tape merging according to equipment available for the program. It defines a wide range of record length and control data.

Report Generator: The report generator describes and assembles a program to produce any desired report in about 30 to 45 minutes over-all time, including sorting and merging in the preparation of the report.

Utility Programs: These are a comprehensive, integrated set of utility programs to be used primarily with the Autocoder. They include the use of modern testing techniques, loads, memory prints, tape prints, etc.

Ramac Programs: These utility programs are for the Ramac file, loading, unloading, and searching operations. They are similar to programs now provided for

the IBM 650. In addition, a set of programs facilitates use of the chaining system, including an evaluation, a loading, additions, and deletions program, and optimum sequencing program.

General: Several programs assist in input/output areas such as tape labeling, end of file, error routines, and restart procedures. An input/output routine automatically schedules simultaneous reading, writing, and processing functions for IBM 7070 card, tape systems, and Ramac systems. This new routine enables the programmer to think of his job as a serial operation and automatically provides an efficient overlap of all functions. An analysis of total programming requirements of previous systems indicates that input/output has averaged approximately 40 per cent of the total coding job. With this new program, the IBM 7070 users can reduce the time substantially since they are not concerned with the implications of input/output scheduling.

Simulation: An IBM 704 program simulates and tests 7070 programs. A 7070 program simulates the 650 card, tape, and Ramac machine with a similar 7070 configuration. This program will run at a speed at least equivalent to the 650 execution of a program.

SUMMARY

The evaluation of a data processing system depends upon more than its inherent machine characteristics.

It includes the procedures, methods, and programs provided with the system and the manufacturers' support needed during the pre-installation, installation, and post-installation phases. All of these necessary and important features are provided with the IBM 7070 Data Processing System.

Specifically, they are:

Balance System: The high speed of computing and internal flow of data are balanced by high-speed tape units, rapid access storage in the disk files, and high-speed card readers and punches.

Maximum Utilization: Automatic priority processing allows efficient time-sharing and multiprogramming abilities for input, output, tape, disk file, and inquiry operations.

Building Blocks: A variety of units in varying capacities permit custom-made systems with the ability to grow as the user's needs are increased.

Application Range: The 7070 can handle a wide range of applications, including batch processing, in-line processing, and computing. It covers the area of medium- to large-scale systems.

Transistors: Solid-state components offer such advantages as high reliability and reduced requirements for floor space, electric power, and air conditioning.

Access and Use of Storage: Each word in core storage can be used for a program step, input or output. Scatter read and gather write minimizes the need for additional

steps which arrange data or assemble them for tape operations.

Simultaneous Operations: Transferring data to and from the system can be overlapped with computing operations.

Fully Alphanumeric: A complete 80-column card can be read in or punched for any combination of numeric and alphabetic data.

Programming Logic: Field definition, 99 index words, single address instruction, and many other factors contribute to direct and simple programming logic.

Variable-Length Records: Full flexibility in handling grouped records of variable length on tape is provided automatically by the scatter read and gather write feature and automatic zero elimination.

Reliability: Complete checking both of input, output, internal operations, and tape and disk storage insure the ultimate in performance.

Programming Systems: Assembly programs and a number of other library routines assist in the planning and programming.

Programming Testing: Programs can be tested prior to delivery permitting full operation immediately after installation.

IBM Services: IBM offers training, planning and programming assistance, customer engineering, and other services. These are the vital steps necessary to insure that the man-machine-methods team is complete and will function properly.

An Organizational Approach to the Development of an Integrated Data-Processing Plan

GEORGE J. FLEMING†

THE dictionary defines organization in four ways. One of these definitions is: "The way a thing's parts are arranged to work together," and this is the one that most nearly describes the subject.

The term integrated data processing has various meanings. Although originally used to describe common-language machine procedure it has gradually been extended to include all phases of the processing of data and is often used to describe the wedding of two or more data systems. For the purposes of this paper, the broader definition will be used.

The reasons for which data are processed may be categorized as follows:

- 1) Top-level management reports.
- 2) Middle-management reports.
- 3) Functional reports.
- 4) First-level management and operating reports.

Although these categories are a bit arbitrary, they are a useful classification for establishing the requirements of an integrated system. Each of these categories competes with the other for data and each has the following characteristics:

- 1) Requirement for data and supporting records.
- 2) Cycles on which they are produced.
- 3) The degree of accuracy that needs to be maintained.
- 4) Manner or method of presentation.

† Boeing Airplane Co., Seattle, Wash.

The degree to which each category is compatible to mechanization varies as does the method of collecting, controlling, transporting, transcribing and processing of the data.

Top-level reports are usually prepared by a skilled staff from data already processed and recorded. They often reflect the considered judgment of the staff and may be accompanied by documents that serve to analyze and evaluate the results they reflect. The volume of data required is relatively small, although data may be collected from many sources. These reports are usually produced on a monthly or longer cycle and the degree of accuracy required is often exacting. Current data-processing techniques are not usually adaptable to these reports.

Middle-management reports are also generally compiled from data that have been summarized and recorded for other purposes. The interests in this category are as wide as the data-collection and processing systems will permit. Requirements range from the critical to the superficial and the pressures that can be brought to bear are considerable. The cycles vary from weekly to monthly and the volumes of data required can be tremendous. The degree of accuracy is high and methods of presentation are complex.

The need to present data that are meaningful in a manner that requires minimum analysis is important as is careful disciplining of the requirements. Generally these requirements are adaptable to mechanization.

The functional reports include payrolls, inventories, production control, and other similar types of data processing that service the many functions of the company. The processing of large amounts of data collected from many sources is a major characteristic. The reporting cycle is most often weekly and the work usually involves the maintaining and up-dating of records. The degree of accuracy required is high as accumulative errors may lead to considerable distortion.

The cooperation of many departments is usually required in order to combine families of associated processing into functional applications. An example of such a combination is the labor handling function which would include data processing required for the personnel, timekeeping, payroll, accounting, treasury, and labor relations departments. Another example is the material handling function which might include some data-handling problems that originate in the purchasing, receiving, storing, accounts payable and manufacturing areas of the company.

These processes are highly compatible to machines, and are generally the forte of the data-processing service center. Most of the applications being processed on electronic data-processing equipment are, at least for the moment, in these functional reporting areas. The new data available to middle management are largely a by-product of these processes.

First-level management and operating reports are

often the stepchild of our modern high-speed data-processing systems.

The operating levels of management need reports that reflect yesterday's results the first thing this morning. The need for corrective action is urgent and must be taken immediately if it is to be effective. The report formats must be easily understood by many people and the contents flagged so that a cursory scanning of the detail will reveal troubled areas. The volume of data required is often large, as these reports are prepared for the control of detailed operations. A relatively low level of accuracy may be tolerated. A daily cycle and simplicity of format are the important considerations. Data collected in or close to the area being served are normally the source of these reports.

Format, collection systems and methods need to be individually tailored to the department served as the manufacturing or service processes they are designed to control usually differ within each department.

The data-processing center is rarely equipped with either the machines or manpower to cope with this category of reporting. Yet, the data needed for these reports are usually the same data that are required for the functional reporting category.

In order to promote efficient, practical, and economical integrated processing, the data-processing center must maintain an influential position regarding the determination of data-handling procedures within the company. The quality and cost of data processing for the functional and middle-management categories of reporting will be dependent on the center's success in establishing informational pipelines into all departments and the development of uniform and standardized methods of operation. The center will need to maintain staffs of intelligent personnel trained in data-processing methods in order to set up and operate the sophisticated procedures that will be required.

Experience indicates that even this is not enough. Only when machine techniques are combined with practical operating experience can integrated results be successfully achieved. The establishment of informational pipelines is only useful when they can be carefully maintained and vigilantly guarded. Sophisticated procedure may be satisfactory for the highly-skilled machine technician but breaks down rapidly when entrusted to less-informed employees.

It appears that we may have reached an impasse on our trail towards the integration of data processing.

The requirements of the data-processing center are, in many respects, in conflict with aspirations of the department being served. The forcing of informational pipelines and procedural restriction on other departments tends to create awkward human relations problems.

The various departments in need of "on the spot" reporting to keep their first-line management informed will be inclined to develop their own methods which will

compete for data and make standardizing of data-handling procedures difficult. Arranging these parts to work together is a challenging problem.

The proposed solution is not an easy way out. However, where the suggested type of organization has been established by managers who were aware of the many incompatible aspects of the problem and who have had a sincere desire to promote efficient methods of data processing, it has been quite successful.

The proposed plan assumes the following:

- 1) A centralized concept of data processing has been established, at least for the use of major equipment items.
- 2) Integration of the various data processes is considered advisable.
- 3) The central data service is equipped to take proper care of most of the functional and middle-management reporting and record-keeping requirements.
- 4) The center reports to a level of management that is influential in all departments of the organization.
- 5) A capable staff of analysts and programmers is available.
- 6) Initial applications on major equipment have been satisfactorily installed.

The major feature of the proposed organizational plan is an outside (of the center) operation designed to provide individual service to the using departments and provide for the informational pipelines required by the center. Another way of describing these service groups would be branch or satellite data-processing operations.

The satellite operation may be as large or small as required to provide for the needs of the department being served and may operate minor data-processing equipment. The equipment may range from paper-tape-equipped adding machines, typewriters, and bookkeeping machines, to small punched-card installations or, where justified, small-scale electronic machines.

Normally, these satellite operations will be located in the using departments in order to establish an atmosphere conducive to the wedding of technical machine-processing skills with the departmental experience.

The supervisor of the satellite operator will report directly to the data-processing center. However, he will be dedicated to serving the department manager to whom he is assigned and must be approved by the department manager. It is expected that this operation will be staffed by a mixture of personnel drawn from both the data-processing center and the using departments.

The responsibility of this group will, in addition to serving the using department, also have the secondary responsibility for establishing and maintaining the pipe-

lines necessary to provide the data-processing center with the data that are originated or perpetuated by the using department. It is further expected that as the satellite group acquires the proper skills they will assume an active role in preparing suggested methods and procedure for the approval of the department head and, when approved, assist in their implementation.

Although the size of the satellite operation and the manner in which it is equipped will vary with the needs of each department, the duties will remain the same. Implementing the plan will require a clear understanding of the objectives and duties of each satellite group. A letter or memo signed by the data-processing manager and the department head is suggested in order to be certain that the operating ground rules are firmly established. These rules may be expanded as the group gains familiarity with the area. However, the line of reporting must be to the data-processing center if maximum benefits are to be attained.

Organization within the data-processing center is flexible. However, as the number of satellite operations increases, it may be advisable to appoint a supervisor over these operations to insure that expected standards of service are maintained and to coordinate the procedural and data-flow activities.

Several problems in human relations are apparent in this proposal, such as the acceptance of the satellite operation into a department, the divided responsibilities of the supervisor, and the relationship between the department head and the data-processing manager. Difficulties may be expected in this regard, but experience indicates that wherever a sincere effort is made to overcome these difficulties, they are less serious than those generated by other types of organization. The proposal is often questioned from the standpoint of the utilization of equipment and manpower. This factor, if present at all, is normally offset by superior service rendered the using department. The data-processing center will benefit by receiving preprocessed data under carefully administered controls and in presummarized form.

The most serious problem will prove to be in adequately manning the satellite operation with qualified personnel. Most often the success of this plan will be reflected in the ability of the appointed supervisor.

The establishment of these satellite data processing groups provides an organization which may well serve to overcome most of the day-to-day problems connected with integrated data processing. Its many benefits include:

- 1) Providing the using department with a specialized data-processing service. When properly equipped, this group can prepare records and reports tailored to the department's requirements without seriously interfering with the schedule of the central data service. It also provides a vital service to the data-processing center by maintaining surveil-

lance over the data-collection system that is so necessary to high-speed processing.

- 2) Such an organization provides a coordination medium through which standards of operations, procedures and practices may be established. It also provides a natural working medium for the exchange of mechanical techniques and departmental operating experience.
- 3) By being directly related to the service center, the satellite group is in a position to call for and get expert service and advice from the service center. It may borrow a keypunch operator to relieve a temporary situation, obtain an experienced operator if required, or call in an analyst to assist them. The shifting of temporary overloads to the service centers standby equipment is facilitated and it should be noted that the center will have a built-in additional capacity for weekend or emergency work.
- 4) The proposed arrangement is useful in establishing control over data-processing equipment to be used within the company.
- 5) It allows departmental management to concentrate on their prime objective with the assurance that records vital to the department's functions will be properly maintained. The system analysis

group within the service center is provided a means of reviewing departmental reports to assure themselves and the company that costly and unnecessarily redundant reports are not being maintained.

- 6) This plan also provides a means for giving the technically-trained employee management and administrative experience which will help to assure a pool of prospective management talent. The service center will be benefited as this plan provides a line of advancement for their superior employees.

Precedents for the proposed plan are not too difficult to find. For example: The timekeeping organization, although reporting to one authority, provides services to many areas. Transportation units often operate from central pools while maintaining specialized services in remote departments. Purchasing normally concentrated in a center provides associated operations to service outlying branches. In fact, wherever overlapping services are a requirement, such organizations are not uncommon.

It seems probable that recently devised equipment combined with advanced techniques and organized in this manner will permit another step toward the goal of truly integrated data processing.

Developing a Long-Range Plan for Corporate Methods and the Dependence on Electronic Data Processing

NORMAN J. REAM†

INTRODUCTION

I HAVE been asked to speak to you on the subject of the impact of electronic data-processing innovations on corporate systems planning. This subject matter could be a recitation of how we have approached our planning effort at Lockheed followed by a recitation of how it has been adjusted from time to time by innovations announced by various manufacturers of electronic data processing equipment.

However, I feel that this subject can best be approached by first spelling out some of the major problems facing all industry, pointing up some areas that are lacking in development. Then I shall attempt to discuss what appears to be a logical approach to these difficult management problems, what contributions electronic

data processing has made to date, and what contributions future innovations in electronic data processing will or will not make to the solution of this multitude of problems.

While my remarks are directed to a corporate administrative systems planning effort, we all realize that the subjects that are under discussion at this conference have broad social significance and we must stand ready to assume our responsibilities.

Our economic system is designed in a manner in which a majority of the decisions affecting it are made by thousands of independent managements. This is an advantage to our country and to industry, but it also poses heavy responsibilities on the shoulders of members of management. As Americans, we are convinced that this freedom of action awarded these managements will, when working within the proper social framework and business environment, result in the greatest good for our

† Lockheed Aircraft Corp., Burbank, Calif.

citizenry. As a nation, we have flourished under the concept of individual freedom and I believe that it holds the only promise for our future.

You are all aware that at the present time we are engaged in an economic "cold war." Khrushchev has publicly announced that it is the Russian intention to bring the United States to its knees by means of economic war. He has also announced that at the end of their latest seven-year plan, Russia will have surpassed the United States in production and that the living standard of the average Russian will be better than we Americans now experience or will be experiencing at that time.

Fantastic claims? Perhaps, but we are engaged in a life or death struggle and the boasted intent of our economic adversary must be taken seriously, for these same intentions were first announced by Karl Marx over one hundred years ago. Therefore, the seriousness of our responsibility cannot be overstated. We must seek ways and means of increasing our over-all productivity and continuing the elevation of our living standards.

To maintain our cherished world position ahead of other world political doctrines will require that the thousands of independent managements in our country provide initiative and leadership in the use of the resources at their command. They are charged with the responsibility of securing greater internal efficiencies within their individual organizations and at the same time maintaining and improving their satisfactory relationships with employees, customers, stockholders, and others.

Drucker¹ stated in a recent article that one of the major problems of business is "the lack of any bridge of understanding between the 'macro-economics' of an economy and the 'micro-economics' of the most important actor of this economy, the business enterprise." He said that the only micro-economic concept to be found today in our economic theory is that of profit maximization which may mean short-run immediate revenue or long-range basic profitability of wealth-producing resources that may have to be qualified by a host of unpredictables, such as managerial power drives, union pressures, technology, etc. But this fails to account for business behavior in a growing economy.

According to Drucker, profit maximization is the wrong concept. The relevant question is: "What minimum does a business need?" not "What maximum can it make?" Companies that have attempted to think through the risks of business have found that the survival minimum exceeded the present "maxima" in many cases.

Drucker further pointed out that another crying need is the development of an integrated organization. Twenty years ago it was possible to see a business enterprise as a mechanical assemblage of functions, but today we know that when we talk of business, functions

simply do not exist. We speak of business profit, risk, product, investment, and customer relationships. The functions are irrelevant to any of them, yet we also recognize that work has to be done by people who specialize because no one knows thoroughly the ins and outs of a given function today, let alone all functions of a business. A basic problem then is how to transmute functional knowledge and functional contribution into general direction and profitable general results.

To my knowledge the problem of integration has escaped solution and will only be answered by devoted research and development. This will not be easy for we must recognize we are faced with the problem of developing a means of measuring and controlling a complex assortment of interacting groups of variously motivated entities in a flux of decision-making situations that comprise a normal company. The degree of the complexity involved is usually in direct proportion to the size of the organization.

RESPONSIBILITIES OF MANAGEMENT

Let us turn for a moment to a discussion of the responsibilities of management, for certainly the administrative systems planning effort of any company must be addressed to the responsibilities of management.

In the broadest sense we could define the responsibilities of management as the guidance, leadership, and control of a group of individuals toward a common objective. This broad definition indicates a purpose, but fails to give us an insight of how results are obtained. Therefore, it is necessary to define the responsibilities of management by defining their five basic processes:

- 1) *Planning*—that is, determining what shall be done. As used here, planning covers a wide range of decisions, including the clarification of objectives, establishment of policies, establishment of programs, and determining specific methods and procedures.
- 2) *Organizing*—or grouping the activities necessary to carry out the plans into management units and defining the relationships among the executives and workers in such units.
- 3) *Assembling resources*—that is, obtaining for the use of the business the personnel, capital, facilities, and other things needed to execute the established plans.
- 4) *Directing*—*i.e.*, issuing management directives. This includes the vital matter of indicating plans to those who are responsible for carrying them out.
- 5) *Controlling*—or seeing that operating results conform as nearly as possible to the established plans. This involves the establishment of standards, motivation of people to achieve these standards, comparison of actual results against the predetermined standard, and initiating necessary corrective action when performance deviates from the plan.

¹ P. F. Drucker, "Business objectives and survival needs," *J. Business*, Univ. Chicago Press, Chicago, Ill., vol. 31, pp. 81-90; April, 1958.

All management engages in the processes I have enumerated, and it is clear that various individuals who comprise management spend varying amounts of time at each.

The different members of management are divided into their functional specialties, such as sales, research and development, engineering, manufacturing, finance, industrial relations, etc. Each of these may in turn be divided into the five basic processes of management previously enumerated. These are then established as two different approaches to the same management activities. For example, the vice-president of engineering must plan, organize, assemble resources, and direct and control in the same manner as any other member of management. His problems may differ in degree, but they are interrelated and interdependent upon those problems of the balance of the management echelons.

The increase in the scope and complexity of modern-day business has resulted in management recognition of the necessity of further development and increased use of scientific management techniques. Management recognizes that our simplifying processes must move forward in balance with business complicating processes. Unless our simplifying processes keep pace, we will become a casualty of our self-developed economic and business complexity.

The next few years will see a tremendous increase in the use of data-processing systems in the development of new management methods; however, I wish to emphasize that in my opinion they will only be a tool of management in the over-all improvement of the management abilities.

OBJECTIVES OF CORPORATE SYSTEMS PLANNING

In any corporate systems planning effort, it is axiomatic that we direct our attentions to the problem of determining what information is required to operate business in a coordinated and profitable manner.

Good communication aids in coordinating the known activities of management. For instance, management must know promptly whether operations are proceeding in accordance with plans so that adjustments can be made when required. Moreover, there are a wide variety of activities, particularly those of a detailed nature, that are impractical to plan too far in advance, and coordination of these is achieved only when the personnel directing and performing them have current information upon which to base decisions.

Management has awakened to the realization that a business is essentially controlled and directed by decisions based on information supplied by its data-processing system. It has realized that major policy evolves from a whole series of day-to-day decisions based on the information currently at hand. And it has awakened to the bare fact that much desirable information is not available.

Further, management is realizing that it is much

simpler to set down the logic of problems in the field of physical sciences than it is to set down the assumed logic of a business executive when he is making a decision based on incomplete data.

Having established that the communication of essential information within a company is paramount in developing an integrated approach to management problems, it is necessary that we turn our attention to the administrative systems that are used to supply management intelligence.

The basic objectives in the development of an integrated approach to administrative systems are:

- 1) Development of improved management intelligence for use in decision-making processes.
- 2) The reduction and control of time spans.
- 3) Improved accuracy.
- 4) Increased productivity.
- 5) Reduced costs of operation.

None of these basic objectives are new, but the advent of electronic data-processing systems has given new life to this whole area. We are now and will be seeking the proper application of data-processing devices in order to take full advantage of interrelationships between the data problems of various segments of management and to recognize appropriately the dependence of a number of these segments on the same basic input information.

PLANNED IMPROVEMENT

In attempting to devise a planned administrative systems improvement program for Lockheed, we recognized we could not realize the desired results merely by studying, appraising, and converting our existing systems and procedures. We were faced with a research problem of considerable magnitude, and it could only be solved by an analysis of the requirements based upon a knowledge of systems parameters.

We found we had to solve this logical problem: *How can we best take the basic information from our day-to-day operations and process and distribute it so as to maximize our profits and minimize our costs?* The problem required that we devise a well-planned program to be carried out by creative people acquainted with research methods.

The most difficult part of any complicated problem, whether administrative or scientific, is the devising of a clear formulation and the establishment of a systematic manner of proceeding. The administrative-systems problem being primarily concerned with the processing and flow of information has three basic parts:

- 1) *Formulation of the problem*—We must determine here what inputs are required and what outputs are required.
- 2) *Logical design*—In this part of the study, we set up the internal relationships and describe the detailed information flow so given inputs will produce required outputs.

- 3) *Detailed systems design*—This part is concerned with the techniques and the tools of the system and spells out how the operations required in 2) will be accomplished.

We found that we had to attack these in their order (although at times it is possible to accomplish some of the logical design and the detailed system design in parallel).

Again, I emphasize that the most difficult area is the proper formulation of the problem. Until this is accomplished, little return can be expected in attempting to attack small pieces of the existing system.

The basic requirement is to tie the various segments of management into an effective whole through proper flow of information. It is binding the entire organization together into an effective, integrated whole through this flow of information that permits the information pipelines to serve not only as a means of improving day-to-day operations, but the projected operations as well.

Regarding this as a logical problem we realized that it is not necessary that *each* person working on the formulation of the problem or the logical design have a detailed knowledge of the existing systems, accounting, manufacturing control, etc. The actual requirement is an ability to use research techniques and an objective attitude which will not necessarily be influenced by existing administrative systems.

EFFECT OF ELECTRONIC DATA-PROCESSING INNOVATIONS

In attempting to discuss the effect of equipment innovations on the systems planning effort of a corporation, I would like to preface my comments by stating that in my opinion industry in general has not learned to use efficiently the logical abilities of existing data processing systems. I do not mean to discount the tremendous contribution that electronic data processing systems have made to management, but I wish to emphasize that in my opinion, we are only on the frontiers of exploiting their potential values.

Certainly the advent of solid-state systems will increase considerably the over-all productivity of equipment systems and will improve their reliability. However, until we improve our ability to use the logical designs of such new systems, we shall be using them as inefficiently as we are using existing systems, and our costs will undoubtedly be proportionately higher than they should be.

Since the delivery of the first Univac system to the Census Bureau in March, 1951, we have seen tremendous strides in the development of equipment systems. For instance, in the 700 series we have seen the 702 and then the successive introduction of multiple versions of the 705. Unfortunately, companies that have followed the trend of accepting all equipment changes that have come down the pike have done so at a tremendous cost.

I believe changes of electronic data processing systems must be dictated by management's ability to use them efficiently after all costs have been considered. Too many times equipment changes have been made without recognizing the multiple of such hidden costs as site preparation, turn-around costs, reprogramming effort, etc. A word of caution—evaluate any proposed equipment change carefully.

With the rapid introduction of more advanced data-processing systems by various manufacturers, many managements are utterly confused and are continually pressed to answer the supposed problem of equipment obsolescence. It seems to me that equipment obsolescence falls into four categories:

- 1) Economically obsolete—It would be beneficial economically to discard equipment and replace it with equipment of more advanced design.
- 2) Technically obsolete—Better equipment is immediately available. It is not warranted to discard presently installed equipment, but if new equipment were being acquired, it would be more feasible to acquire the latest design.
- 3) Technically obsolescent—Better equipment is in the design stage, but is not presently available.
- 4) Conceptually obsolete—Better equipment is theoretically possible, but has not yet been designed.

If you will reflect for a moment you will recognize the four stages of obsolescence and realize that all equipments proceed from the stage of being conceptually obsolete to the point of being economically obsolete. However, the point of becoming economically obsolete is usually a long time after they become technically obsolete, because the marginal cost of operating and maintaining them is small. If we were in a detailed discussion of leased or purchased equipment, then we would have to concede that when equipments are being leased, the point at which they reach economic obsolescence is much earlier than when they are purchased.

There are two major reasons for a change in electronic data-processing systems:

- 1) It is desirable to decrease the unit cost, or
- 2) It is desirable to do jobs that are not possible of accomplishment on existing equipment.

One of these two must be the basic reason for change to equipments of increased speed or increased memory capacity.

The physical difficulties as well as the economic problems of changing equipment, both for the manufacturer and the user, acts as an inertia or a brake on the introduction of too many new models, each being a slight improvement over the old. In my opinion, an order of magnitude of three to five times in improvement is needed before a going machine can economically be supplanted.

Because of the long cycles and lead times involved in

the development in the new data-processing systems, there is an upper limit to the speed with which new electronic data-processing systems can be introduced by any one manufacturer. However, as additional manufacturers start at varying places on the time scale, the result for the consumer is a practically continuous curve of increased technological advantage. The management problem then becomes one of choosing at which point to make the change and of assessing the reliability of the various manufacturers' claims.

Another factor that must be considered is whether or not the user of the proposed new electronic data processing system actually has a fully developed requirement. In other words, you can find operations where electronic data processing systems have been changed only to learn that they have acquired too large a system for the amount of work to be processed.

To date, most of the attention of management has been directed towards gathering historical information and not too much progress has been made in the area of development of management information for determination of a company's objectives. The advent of electronic data processing systems makes possible the application of techniques that had hitherto been impossible by manual or electromechanical means. We can look forward to great strides in this area in the next few years in the development and application of mathematical techniques (operations research) to the problems of management.

The area of financial planning and control is wide open, for example, in the applications of scientific management techniques. I am sure that electronic data-processing equipment will be a major contributor to the eventual solution of many existing problems. However, the equipment will only be a means of developing an answer after we have determined what information is required and how it can be developed. Here again, I would like to emphasize that progress will not come about merely by having the equipment, but must be preceded by a fully developed definition of the problem and the logical design of the system required.

In my initial work in the financial forecasting area, the first unforeseen difficulty was the problem of language. As a simple example, discussion with people in financial operations activities revealed instances of several distinct concepts answering to the same name and a multiplicity of names applied to essentially one concept. As a result, it was considered necessary to establish for the accounting aspects of the problem a structural framework which was reasonably complete, self-consistent, and within which the problem could be formulated with some assurance that the terms employed were clearly defined. Eventually it was determined that the problem could be solved in four steps. First, obtaining a mathematical model for the entire accounting feature of the forecast. Second, a study of prediction techniques used in obtaining basic information for the various inputs to

the accounting structure. Third, a study of the procedures necessary to make certain quantities of the output a maximum or minimum, while others are held under specified restrictions. Fourth, the application of high-speed data processing equipment to the results of the three previous studies.

I might add that work in these areas does not lend itself to rapid solution but is a result of long and arduous effort on the part of persons who are dedicated to re-searching this type of management problem.

Information retrieval is another problem area that holds promise of solution. However, data processing equipment cannot make a contribution until such time as the problem is defined and a logical retrieval system is designed.

These latter two are only representative of a host of management problems that must be sought out and solved to insure eventual management survival.

SELLING ELECTRONIC DATA PROCESSING

Those of you who are in the data-processing area of your management are faced with a real challenge to sell your ideas to management as well as to employees. You not only have the problem of selling the use of electronic data-processing equipment, but you ultimately will have the responsibility of proving that the introduction of these systems has reached or exceeded the break-even point. The break-even point of electronic data-processing systems is still most dubious; in fact, many existing installations are not economically sound. You are faced with the challenge of the control of costs after installation and of the continuous effort of insuring that the programming effort is of prime efficiency.

ORGANIZATION CHANGE REQUIRED

One of the most difficult problems which faces the advocates of the use of electronic data-processing systems is to awaken managements to their responsibility of insuring that they are using the systems to their best abilities. I agree that computers have been the impetus behind the tremendous interest that is now being focused on the development of improved administrative systems, but at the same time most managements have failed to recognize the necessity of integrating their organization in a systematic and purposeful manner. Usually these changes just happen over a period of time, arise from temporary expediency, or emerge as a solution to a crisis situation. However, in most instances functional reorganization has not kept abreast of the technological change.

The management organization structure is not inviolate and should be treated accordingly. In many cases changes in the organization structure can eliminate many of the procedures that complicate an administrative system, thus reducing costs and contributing to the simplifying process.

CONSOLIDATION OF SCIENTIFIC AND ADMINISTRATIVE ELECTRONIC DATA-PROCESSING OPERATIONS

One of the most interesting and promising developments in the application of electronic data-processing systems in business is the apparent determination of most manufacturers to develop a data-processing system that can be used for both business and scientific data processing with equal effectiveness. Also, the trend towards the development of these all-purpose data-processing systems in the medium scale field, with an ability to expand, opens the door for the economic use for computers in many smaller industries. It also gives good reason to speculate that large companies will find that the use of these dual-purpose systems will decrease considerably their over-all data-processing systems rental costs by consolidation of scientific and business data-processing operations and maximizing available computer time.

The establishment of two definite data-processing groups within a given area of a business to handle independently scientific and administrative data-processing operations is a very costly venture. The now recognized means to handle many independently programmed operations simultaneously, that is, business, scientific, or both, eliminates many of the arguments that were formerly used to substantiate the independence of the two establishments referred to above. From the standpoint of the future economics of data-processing systems installation, I believe that this is a major step forward.

PRE-ANNOUNCEMENT OF EQUIPMENTS

One of the most disturbing factors in the application of electronic data-processing equipments to management problems is the continued practice of some manufacturers of announcing new equipment innovations far in advance of their actual design completion or availability. I think that manufacturers would serve their purpose well if they would withhold announcements of new equipment until such time as they were able to discuss these new equipment systems with the support of factual information.

I recognize that the industry is extremely competitive and that various companies are jockeying for the best possible position. However, I believe that many of their sales practices have done much to retard progress by throwing many managements into a state of utter confusion by moving equipments into a business enterprise long before the planned installation is ready for their use.

I know of instances where certain manufacturers have attempted to have management withhold decision on equipment in order to deliver their own new system many months in the future when full economic justification of immediately available equipments has been documented. Also, some manufacturers' sales efforts have attempted to replace a competitor's equip-

ment with their own equipments when there is no economic justification to the user.

Usually the sales tactic behind these equipment pre-announcements has been to prove to a potential customer that the manufacturer in question has a corner on the knowledge and abilities in the data-processing equipment field. In my opinion, there are many reliable manufacturers and I know of no company that has a corner in the area of development or know-how.

LACK OF DEVELOPMENT IN THE INPUT AREA

Just as in the usage of punched-card equipments, the weakest area of electronic data-processing systems development today is in the input areas, *i.e.*, the collection of input data at the point of origin. It is true that this area is one of the most difficult to define—and perhaps has been side-stepped by both the users and the manufacturers because of this. Yet, I have the feeling that some manufacturers have long recognized the problem but have been more interested in protecting their investment in equipments and card plants than in making a definite contribution to the state of the art.

ELECTRONIC DATA-PROCESSING SYSTEMS CONTRACTS

I would also like to comment briefly on the present type of contract normally available for the lessee and the lessor of the equipments. Up to a short time ago, the rental contracts were certainly all vague and meaningless and were written solely for the benefit of the lessor. However, recently there has been a trend on the part of some lessors—that is, the manufacturers of the equipment—to write more definitive contracts than were previously available to the user. Even these are not definitive enough. It is time for the users to demand an even more definitive contract, one that charges the manufacturers with performance responsibility. I can see no reason why any manufacturer should make fantastic performance claims for given equipments during his sales efforts and then fail, if they are valid claims, to put these into a leasing agreement. Refusal only means that the abilities of the equipments have been overstated.

The present type of contract, as I previously stated, is a lessor's contract, and I believe that its continued use through the years has caused the user to absorb many costs due to equipment malfunction that should have rightfully been borne by the manufacturer.

DEVELOPMENT OF PROGRAMMING TECHNIQUES

Charges have been levelled at the manufacturers of data-processing equipment that they have failed to develop means of using the logical ability of existing systems to the fullest extent through neglecting to furnish advanced programming aids and techniques. To a degree it is a valid criticism. However, I feel that this is buck-passing, and to date our failure to exploit more fully the use of presently available data-processing

equipment in business is caused by the lack of the development of logical systems that will fully exploit the logical abilities of the equipment available. Too much effort has been expended on trying to transfer available human systems to these equipments rather than attempting to develop a proper definition of the logical system required.

CONCLUSION

Electronic data processing is becoming a byword in the evolution of the techniques of a scientific approach to the problems of management. The equipments are not an end in themselves and cannot be considered a panacea for the ills of management. Rather they are a tool of management. Their contribution to the improvement of management is entirely dependent on how well the problems of management are defined by the indi-

vidual practitioners and how ingenious they are at developing means of formulating information for management review and decision-making processes.

Again, we are only on the frontiers of the potential we seek. Hard work and ingenuity will bring success.

We must move carefully and at all times must be in a position to justify our activities. The introduction of improved electronic data-processing systems will undoubtedly contribute to the advancement of the state of the art, but the feeding of bad inputs into faster and more capable equipments will only generate more bad information at a faster pace.

I have tried to point out some of the difficult areas we are encountering to avoid overoptimism. Yet, there is no reason to be overpessimistic. Our eventual goal can be attained, and with the high stakes involved, the significance of the results warrants the all-out effort.

A General Approach to Planning for Management Use of EDPM Equipment

GOMER H. REDMOND†

DURING the past decade we have all been aware of, or have played a part in, a maximum effort on the part of certain dedicated people. The objective of this informal effort seemed to be to convert all clerical and computational work efforts into an automatic, "fire-all-the-clerks, we-can-do-anything" approach to all known business practices. These dedicated people were mainly comprised of data-processing manufacturers, management consultants, scientists, engineers, and administrative line and staff executives and assistants. They usually had a good grasp of a specific problem and felt that this problem or series of problems were sound EDP applications and by extrapolation, proceeded to teach a number of management executives the economics of EDPM installations.

The influence of this effort and the span of time over which it took place were both beneficial. Decision-making management level executives encouraged and at the same time discouraged the advance of clerical automation. Equipment marketing people were forced to compete in areas that they frankly knew were not practical. "If I don't submit a proposal for this application, competition will get the business and it's a rough row getting back in," were frequent comments that we

all have heard upon questioning EDP salesmen regarding doubtful or submarginal installation.

Management consultants were asked into the clerical automation area by top executives who respected previous neutral and objective work assignments, many of these management consultants accepted work assignments with the assumption that old and proven "standard" techniques would serve their purposes in this area as they had in many others; others went in with good staffs and did a "down town" yeoman-like job. Still others capitalized on the confusion and helped equalize mass optimism by forecasting dire results if EDPM installations were contemplated and planned without specific help from *their* firm.

Not to be outstripped, many business and newspaper writers and reporters joined in and began to point out the miracles of electronic hardware and its possible effect upon American business organizations and operations. Articles in magazines, in newspapers, in trade journals praised use of EDP equipment as a new industrial revolution and played up the hardware and its speed with almost no references to planning and application problems and costs. Other articles damned the rapid acquisition of EDP equipment. These write-ups blew way out of proportion some pioneering marginal commercial installations, threw serious doubt into management's mind as to the capabilities of their EDP planning

† Chrysler Corp., Detroit, Mich.

people and condemned organizational structure for permitting such a cancerous growth to survive.

Systems people and EAM operators also feeling a need to keep abreast were schooled in electronics. In many cases, they forgot traditional good systems concepts by tailoring work areas and good systems concepts to the convenience of a preselected "brain"; this was done to get into business as soon as possible.

To sum this up I would like to quote two individuals who have different viewpoints and yet a mutual understanding of this problem. John Diebold, President of John Diebold and Associates, in an address to the Eleventh International Management Congress said, "Automation has presented management with a major new problem. As yet management has not faced up to this problem and is hardly even grappling with it in any true sense. This is through no lack of energy or good intentions. On the contrary, the very activity of management in this sphere attests to the progressive spirit and desire for improvement that characterize the modern manager. The trouble lies elsewhere. Automation has turned out to be a much more complex and difficult problem than was originally thought. This being the case, the current disposition to minimize its revolutionary and novel aspects is more hindrance than help in putting automation to work."

The other gentleman that I must quote, to successfully set the stage for this general topic, is an EDP equipment manufacturing top executive. He is one of a half-dozen human beings who have successfully bridged the gap between an exacting technical knowledge of EDPM and the work that management *should* plan for this equipment. Just recently, he asked when American military, industrial and institutional management was going to use *properly* the equipment already produced and *plan* the right scientific and data-processing jobs for it to accomplish. He went on to point out that there is enough data handling equipment already produced to process all problems that our economy needs to handle—providing this equipment is properly programmed and properly distributed. Sound planning is the answer.

The need for planning prior to major commitment in most endeavors is apparent. A football team plans so that all eleven men work together to achieve a first down or a touchdown. In business, profit planning, sales planning, product planning, and production planning are present in most organizations and normally are predominant factors in successful achievement of objectives.

Formal planning has not been adopted for orderly consideration of electronic hardware by all business, military, and other forms of organizations. This is primarily due to the apparent desire of many functions of an organization to 1) jump on this bandwagon (or get left behind), 2) to control this monster which could completely dissolve current organization, 3) to merely extend current punch card applications through the electronic barrier, or 4) to ignore the entire subject

until EDP has become a proven practical factor in other organizations.

With the possibility of huge expenditures for programming, for building computer sites, for selection of appropriate systems applications, and for the hardware itself, an organization should begin to plan for successful planning, for it is essential in all forms of planning that a fundamental design and approach to planning a subject be created. Such a design for EDP planning must take into consideration scope, organization, and the ground rules or administration of a planning function itself.

To discuss this on a general basis, due to the number of extremely different forms and types of organizations represented here, let us first examine some possibilities of designing "scope" into an EDP planning function.

SCOPE

Planning latitude may be extremely broad if you are functionally responsible for data processing in a multi-plant industrial complex or it can be extremely limited if the characteristics of the enterprise are small and processing requirements are simple. To my knowledge, there is no formula for determination of planning scope. Some questions may help in formulating this portion of your design for EDP planning:

- 1) Should the plan point long range, at the immediate, or both? What should the plan achieve? Are planning objectives understood by all concerned?
- 2) Is the planning concerned with a part or all of a specific application?
- 3) Should planning review present punch card applications, present manual operations, or should it also encompass data processing for new management decision-making techniques?
- 4) Is planning required for all portions of an industrial enterprise or are we only concerned with a division, a plant, or a single function at corporate, division, or plant level?
- 5) Should planning be pointed toward data-processing economy or in specific work areas do accuracy and speed take precedence?

The scope of planning cannot be completely determined without a review of the planning organization and its place in the organization.

ORGANIZATION

- 1) Where should the staff work of EDP planning take place in the organization? Is central planning required for product planning, for profit planning, etc? Can economies be gained by centralized planning or are products and data-processing problems so different that plant or divisional planning is essential?
- 2) Can a combination centralized-decentralized form of planning work? Does this provide greater flexibility?

- 3) How should the planning group(s) be organized? What should the relationships be to systems employees?
- 4) What should be the make-up of the planning group or groups? How many staff workers should participate? What background, educational and experience levels should be utilized?
- 5) Should committee action be employed?
- 6) Should line employees contribute?
- 7) What part, if any, should outside consultants perform?
- 8) Equipment manufacturers—should they be asked to contribute data or ideas during planning and decision making?

After a proper organization is decided upon, management must decide upon a proper set of ground rules or administrative routine within which the EDP planning will operate.

PLANNING GROUND RULES

- 1) Is planning to be specific or broad? Is the plan to be fixed or flexible? What happens to the plan if broad procedures or organizational patterns are changed during or after completion of the plan?
- 2) How much documentation will be required? Should progress be periodically or occasionally reported, and if so, to what level or levels of management?
- 3) What preplanning bench markers can be established to enable management measurement of planning progress?
- 4) Should management decision be requested during, or only upon completion of, a review of the entire planning scope?
- 5) In what detail should management participate during planning steps? What information must management have to make final decisions properly—all the detailed technical data or broad approximations of cost and estimates of anticipated results?
- 6) Should fact finding be accomplished by a planning staff or should all components contribute factual data?
- 7) Is it necessary to sell staff work and EDP planning prior to the planning action?

These question areas are representative of the many decisions that should be made before initiating a thorough corporate-wide planning survey, or in performing a feasibility study at division or plant level. The list of questions is not a complete check list, but may be helpful in creating a list for a specific organization.

Planning work that can be measured and evaluated is difficult in itself, but when initially directed toward EDP, difficulties in the planning process may be amplified due to communication problems and due to many organizations' inherent caution and fear of possible changes.

Planning work and staff work are not new to most American military men or to business firms. An important contribution to the acceptance of a complex program is the attainment of prior success to create an aura of reliability in those executives that have so performed. Therefore, design for the plan is especially important during an initial planning process; in many cases the "chips" are great, and as in baseball, the batter either gets on base or he is out. In many more cases, however, successful use of EAM equipment plays a significant role in preparing for formal planning.

At Chrysler our planning had to be initiated by superimposing a fact-finding and investigatory stage over an existing broad and, in many cases, extremely *good* data-processing system. This system utilized medium and large EDP equipment along with EAM equipment at various levels of organization. It was largely created by breaking off an existing installation (manpower and programs) and, for geographic purposes, a new installation was then originated. A good amount of systems work had been accomplished, but this area needed extensive work to restrengthen approach concepts as well as to institute much-needed machine room documentation, administration, and control. A total of 42 machine rooms were in existence with over 1200 employees working on data-processing equipment. The comptroller was then informed of the need for a formal plan to investigate work areas and techniques; this review would then expand into an over-all systems study that envisioned examination of source data pickup to finished management action or information reports. He agreed.

EDP planning at Chrysler is the staff responsibility of the Manager, Systems, and Procedures. Data processing is an important management instrument in our company, and we have tailored our systems and procedures organization to place proper emphasis and yet not over-emphasis upon the subject of data processing. Our management believes that a correct balance between the work and programming effort of good systems concepts planning and mechanical and EDP planning people must prevail.

An Assistant Manager, Corporate Systems and Procedures, has been delegated the specific and full-time responsibility of EDP and EAM planning and control. His organization comprises three sections, each headed by a supervisor; they include 1) Operations Research, 2) EDP and EAM Administration and Control, and 3) Planning. Our Planning Section is made up of eight employees who have between 3 and 8 years of practical operating and staff experience in this specific field. Most have college degrees in mathematics, in business, or in engineering; however, some of our finest staff work has been accomplished by personnel with much less formal education.

Initially our planning was purposely geared on a broad basis to securing factual data and, on the first pass, to point out specific areas in which immediate new or remedial work should be initiated. Our preliminary

plan was half completed when we discovered a need; a need for a better way to handle those jobs (in a large industrial complex) that due to low unit costs, minimum elapsed time requirements, and with little additional communication costs, made a large computer a natural.

As in any flexible planning technique, new base lines and facts were developed, and after proper delineation and documentation of the new computer's planning and operating organization, the over-all plan was reinstated. You will note that specific and detailed operating planning, after a management decision has been made, is separated from the staff planning work and in effect proceeds through its own planning cycle within a specific predetermined scope and time limitation.

The broad-gauge, corporate-wide planning is then continued with new factors and perhaps with new ground rules that have resulted from the major decision passed down by the top management. Continuation does not mean starting over, but it does embrace a review of the planning process to date. This planning process entails 1) reaffirmation of objectives, 2) analysis of the new situation, 3) determination of planning routes and manpower requirements, 4) choice of alternate course to be initiated, 5) conversion of choice into action steps, 6) creation of internal and external communication channels, 7) determination of planning and methods of appraisal, and 8) then a procession through these same steps after evaluations indicate that changes in the planning process should be incorporated.

Specifically, our current plan which will be presented again to top management in the near future has the following *objective*.

It is the objective of the Corporate Systems and Procedures Department, through the development and eventual approval and adoption of this and succeeding planning efforts, to provide Chrysler Corporation with the most effective management information systems in the Industry.

The need for *constant* and *continual* planning is a fundamental requirement to successfully integrate, coordinate and effect such a dynamic data handling system. Competitively this is a must for Chrysler.

In general, we have just scratched the surface in the data handling field with the mechanization of some of the "safe" clerical functions. These functions have provided valuable experience in data processing techniques. On occasion, we have used the logical abilities of data handling equipment to provide meaningful data for the control and regulation of manufacturing processes and to directly help to improve our product quality. However, at the present time most of the data generated merely reports the status of operations, produces necessary checks and notices, but is rarely employed to provide recommended actions for timely operational decisions. Ultimately, a data handling system will be evolved that will provide this needed decision-making information, and select action courses for more consistent and precise control at all levels in the organization. The program to effect this plan will not remain static for any long period of time, but must be continually reviewed to reflect new data processing product development, new systems concepts, and progressive changes within the corporation.

After formally stating a purpose, it is frequently necessary to state the base line, or present situation, which helps form the scope and direction of our planning in the minds of the planners and decision makers. The present status should be concise and complete; any de-

isions that have been firmly made, but not as yet implemented, should be "baked in" as part of the plan initiating point. As decisions are made, this part of a plan package will change; such decisions should be formally reflected and should be documented so that at any time that management decision making may occur (and this process could be almost continuous) an up-to-date document can be presented to insure fact-founded decision.

A typical *Present Status* should include:

- 1) A statement and exhibits to indicate present data-processing installations.
- 2) An exhibit indicating the size, geographical location of these installations, along with an indication of communication media.
- 3) The basic type of equipment employed and their cost or rental.
- 4) An analysis of the number of people involved in data processing and the annual cost including fringe benefits.
- 5) A statement and chart that indicates the current organizational placement of data-processing components throughout the corporation.
- 6) A chart indicating functions and work areas that are being handled by data processing for each installation within the corporation.
- 7) A detailed statement indicating the history of a representative data-processing component; how it grew, what jobs it is now performing, what basic costs would be incurred if automated techniques were not employed.
- 8) An indication of the training and supervisory talents that present job incumbents possess.
- 9) Examples of areas in which improper control and improper organization, lack of documentation, poor systems concepts, etc., are contributing to erratic and faulty data processing.

Current planning progress should then be indicated to show positive staff and line, central and decentralized work effort that has resulted from the over-all planning efforts. This should include such progress as:

- 1) Training programs initiated by corporate or machine manufacturers to increase the scope of employees' work efforts through greater knowledge of new concepts or machines.
- 2) Training programs to teach supervisors and machine procedural planners the latest administrative, documentation, and control techniques.
- 3) Installation progress of approved EDP equipment or of new or combined EAM installations.
- 4) Major conceptual developments initiated at any EDP installation.
- 5) Disclosure of *preliminary* detail plans that look promising (although in which sufficient staff work has not been accomplished to secure final decision).

- 6) Development of community or employee communications designed to "sell" data processing.
- 7) Progress made in report and problem solution areas entirely foreign to data processing at this time; using techniques such as simulation, or by tailoring mathematical models of problem areas, thereby establishing entirely different methods for better and more accurate management decision.

The next section of a planning package may be a review of the development of short-range programs, as they relate to a long-range program. This program development review should include:

- 1) A concise statement of problem areas; personnel outside of the planning section should be asked to supplement and/or modify a list of these areas.
- 2) A brief statement as to the proper remedial action that must be taken to correct existing problem areas. Again, the considered opinion of line and staff executives should be requested during the formulation of the short-range action plan.
- 3) New problem areas currently not processed mechanically or electronically; these should be defined and should be investigated in terms of possible application and solution by use of different source information or processing techniques.
- 4) A firm schedule against which periodic progress reports may be measured.

- 5) Using the aforementioned four steps, a general statement recording 1-, 3-, and 5-year goals of information handling should be spelled out. This should include a statement specifying the planning philosophy and the general direction in which the plan is approaching the stated objectives.

In closing, EDP planning is a repetitive, dynamic process that, to be effective, is flexible and when intermediate decisions are made, should be re-created. It may be worthless if used as a pure academic exercise. A plan is considered and then may be rejected or partially or completely adopted. In any event, it has served a purpose. EDP planning should be tailored to the total enterprise, to top management individuals, and to the background of previous good or bad experiences with EAM or EDP applications. Planning recognizes that staff members do not have a patent on brains. Factual and idea contribution must be encouraged and developed. It should not always result in an EDP application; in many cases, byproducts of EDP plans and feasibility studies are more valuable than the EDP results themselves.

De-emphasizing hardware, it should place much more stress upon approach concepts and related facts.

EDP planning should be kept practical and should engender continuing enthusiasm on the part of the planning staff. It is not an end in itself. The only reward of good planning is its influence upon management, which results in the achievement of stated objectives.

Dynamic Production Scheduling of Job-Shop Operations on the IBM 704 Data-Processing Equipment

L. N. CAPLAN[†] AND V. L. SCHATZ[†]

FIRST, let us explain what we mean by job-shop operations. In this case we are talking about our Jet Engine Department in the General Electric Company, Aircraft Gas Turbine Division. The Jet Engine Department is the step between the basic concept development and the production shop. Their manufacturing facility is responsible for building engines and components and for testing these engines and components to prove practicability of design, serviceability, manufacturability, and reliability. The prototype engines are also within their manufacturing responsibility.

From this, one can visualize the type of operation about which we are speaking. It is largely one where Engineering requests parts to be manufactured for installation into an assembly which is to be tested. The important thing is that there are a large number of requestors and relatively few items per request. Stated another way, this means no schedule of incoming work. Hence, the name, job shop. Fig. 1 represents the flow of operations just discussed.

Now that we have seen what a job shop represents in our case, consider the size of our operation. There are approximately 9000 operations on order in the manufacturing shop at any one time. These operations may represent two or three thousand requests with several

[†] General Elec. Co., Business Systems Computations, Evendale, Ohio.

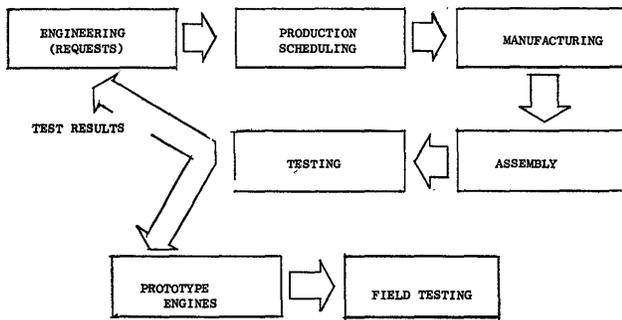


Fig. 1—Jet Engine Department operations diagram.

machine operations per request. This shop contains 50 machine-tool areas. An area in this case represents machines which would perform one type of function—for example, 5 lathes or 3 grinders or 15 inspectors. The cycle time for one job through the shop averages about 22 days.

At this point the problem becomes evident. We must be able to provide a schedule which will process work through the shop in the most efficient manner. We must work to a schedule that is set by the requestors' due dates. We must minimize waiting time for the part requested, we must minimize farm-out to vendors while idle time exists in our own shop, and we must minimize the idle operator time in our own machine shop. Also, we would like maximum utilization of our expensive machine tools. The scheduling must be completed quickly; otherwise, it is obsolete by the time it is to be used. In addition to the size of the job to be done, other complications to scheduling include the necessity to make allowance for weekends, allowance for holidays, consideration for overtime authorizations for extra shifts, holidays and weekends, sequential machine operations, simultaneous completion of machine operations on all parts going into an assembly, and operations being performed by outside vendors. It is difficult to develop the start and finish time for each job when all these facts are considered.

A punched-card and tabulator system had been used with some degree of success in the past. The primary output of this system was a punched card provided to the shop foremen showing the job to be worked on and the starting date. In spite of the fact that this is a relatively small shop, the scheduling became nearly impossible on tab equipment. The scheduling was inefficient to the extent that we were having to farm out work and at the same time we had idle machine time in our own shop. The resultant loss ran into thousands of dollars monthly. Also, in one case we were scheduling for two months' backlog of grinding work when actually less than two days' work was available for grinding.

This was more or less the climax which brought about consideration of the computer by the Production Scheduling people. Initially they were discouraged because of some of the things they had been told. For example, a great portion of the machine work would be sorting of the 9000 work-information cards into priority sequence and then resorting into job-order sequence. The job num-

ber is 19 digits in length. Therefore, to sort and resort the information deck would require passing 342,000 ($9000 \times 2 \times 19$) cards through a conventional sorter. This would take well over 10 hours and the schedule would be obsolete before the sorting was completed. The opinion was that sorting of this type would be prohibitive in cost on a computer. A second opinion held that the great number of exceptions would make this scheduling impractical to program for a computer. Nevertheless, Production Scheduling brought the problem to the Computations facility to see what could be done. After some study, we decided that the job could be accomplished on our 704. Painstaking programming would be required and new techniques or so-called "tricks" could be developed to do the sorting in a practical time. The existing card system was adapted to computer handling with a bare minimum of change. One man from Production Scheduling, working half-time, and a man from Computations, working full-time, completed and placed the program in operation in about 10 months.

Let us consider the system and the computer's role. When a work authorization is received by the machine shop, the scheduling group assigns a job number and priority, and breaks the order down into operations and estimated hours required for each operation. Cards are then made up for each operation. The card contains priority, job number, operation number, type of overtime allowed, men required, work area and alternate work area, a minimum starting date, estimated hours, part name, quantity, and some other information non-pertinent to the computer operation. The cards for new work are brought to Computations on a weekly basis along with cards on work in process. A date card to tell the machine when the scheduling begins heads the deck. Priority cards are placed ahead of the operation's cards. (See Fig. 2.) Within Computations these cards, 9000 on the average, are placed on tape via card-to-tape converter. The tape and program are placed on the machine. As the tape reads into the machine, the information is checked for possible errors, such as blanks in a numerical field, impossible data such as February 29th, non-leap year, and 31 days in 30-day months, or out-of-sequence job number. Some errors may be corrected. For example, if a 31st day is found in a 30-day month, the machine will make the day read as the first day of the next month. All errors are listed by job and operation number. If 50 errors are detected in the input deck, the operator is notified by an on-line comment to halt the program. The computer then prints the rest of the errors, if any, in the deck. As the information is being checked, it is also being transformed into binary words so that all scheduling information for the 9000 cards may be fitted into the 32,000-word memory at one time. This is still not possible so that job number (2 words) is placed on a tape sequentially. The technique for sorting this information is as follows. The memory was split into three parts. Storage 0-9000 will contain words with priority number and sequence number stored sequentially in the same word; 9001-18,000 contains words of scheduling information in sequential order; and 18,001-

JET ENGINE DEPT.		BADGE NUMBER		NAME	
 PRODUCTION SCHEDULING CARD		PERCENT COMPLETE		FOREMANS APPROVAL	
		COMMENTS		START TIME	
				STOP TIME	
		HRS		TENTHS	
SHOP #	AREA	OPER	JOB NUMBER	START DATE	PART NAME
1	2	3	4	5	6

00199-		PRIORITY CARD	
00000000	0000	0000	0000
12345678	9012	3456	7890
22222222	3333	4444	5555
66666666	7777	8888	9999
12345678	9012	3456	7890
22222222	3333	4444	5555
66666666	7777	8888	9999
12345678	9012	3456	7890
22222222	3333	4444	5555
66666666	7777	8888	9999
12345678	9012	3456	7890

041258		DATE CARD	
00000000	0000	0000	0000
12345678	9012	3456	7890
22222222	3333	4444	5555
66666666	7777	8888	9999
12345678	9012	3456	7890
22222222	3333	4444	5555
66666666	7777	8888	9999
12345678	9012	3456	7890

Fig. 2—IBM cards making up data deck.

27,000 contains other words of scheduling information sequentially. (See Fig. 3.)

The first 9000 words are sorted into priority sequence. By looking at the first word and adding 9000 and 18,000 respectively, the computer may pick up the second and third words belonging to that operation. Sorting into order requires approximately two minutes. We write this on a tape. Similarly, we read back into spaces 9001-18,000 and 18,001-27,000 the job number and pick it up in the same manner as above to merge with the tape containing our scheduling information. Finally, we have a tape containing the operations, including job number written in priority order. This tape feeds back into the computer and scheduling takes place. Assemblies and sequential operations are taken care of by using buckets to hold dates until the required pieces are finished. (See Fig. 4.) From this, we get a tape containing start dates in priority order and sequence number. This tape is read into memory and merged with our original input tape to place a starting date on each operation. (See Fig. 5.) The result is punched out as dispatch cards for the shop areas. Of course, while scheduling is going on, the computer is tabulating load per area per day. If maximum capacity is reached in any area per day, the tabulation and schedule move to the next day.

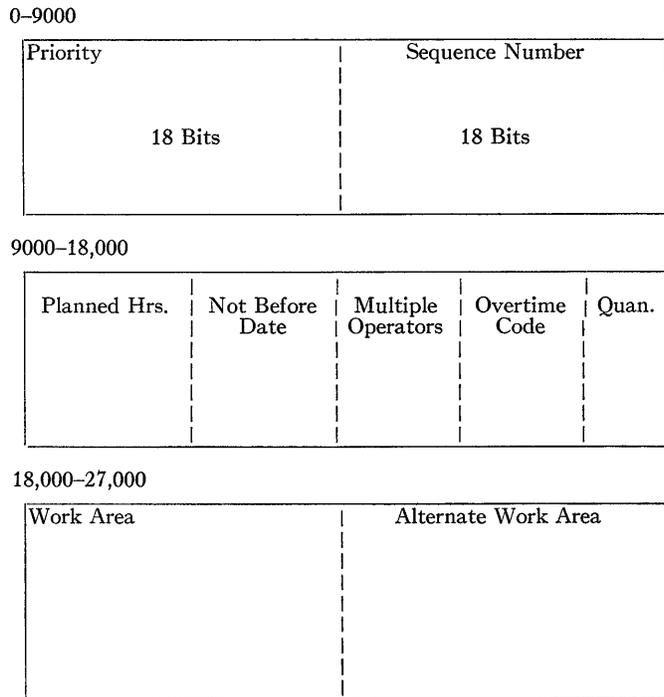


Fig. 3—Illustration of memory division for sorting purposes.

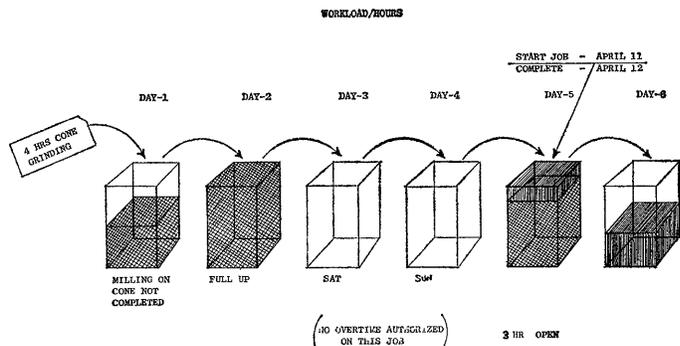


Fig. 4—Example of machine-tool area loading logic.

The printed work load (see Fig. 6) is sent to the Production Scheduling Office. The number at the far left denotes the machine tool areas, such as grinding, milling, etc. The schedule starts as of the date printed at the top of the page. The work allotted to each area each day for the next 42 days is printed. Finally, the area's maximum capacity per day is printed on the far right. If they feel they can do better on work load, they may reshuffle priority and run again. (Our future plans are to have the computer determine optimum schedule.) The punched cards are sent to the machine tool areas. The workmen fill in the hours worked or finished on the card and the cards are returned to the scheduling office to be remade for next week's run. That is, if a workman has completed 15 hours of a 40-hour job, he marks 15 on the card. The scheduling office will remake the card with 25 instead of 40 estimated hours and place it in the deck to be sent to Computations. (See Fig. 7.)

Some interesting sidelights regarding the computer

Part Name	Area	Job Number	Planned Hrs.	Start Date
Shaft Gear Spur	202 245	5-6-00169 000 04 7001	0000	10/23
Shaft Gear Spur	212 255	5-6-00169 000 04 7001	0017	10/27
Shaft Gear Spur	201 265	5-6-00169 000 04 7001	0030	10/27
Shaft Gear Spur	Mag 275	5-6-00169 000 04 7001	0000	10/28
Shaft Gear Spur	Ins 285	5-6-00169 000 04 7001	0000	10/30
Shaft Gear Spur	212 295	5-6-00169 000 04 7001	0032	11/03
Shaft Gear Spur	Ins 305	5-6-00169 000 04 7001	0000	11/06
Ring Spl Adapt	274 020	5-6-40151 000 01 0001	0050	10/01
Ring Spl Adapt	212 030	5-6-40151 000 01 0001	0010	10/02
Ring Spl Adapt	Ins 040	5-6-40151 000 01 0001	0000	10/17
Comp Rotor	268 050	5-7-80042 803 01 0001	0120	10/02
Comp Rotor	212 060	5-7-80042 803 01 0001	0030	10/03
Comp Rotor	Ins 070	5-7-80042 803 01 0001	0000	10/07
Blade Stg 3	Zyg 020	5-7-80045 803 01 0001	0000	09/29
Blade Stg 3	Ins 030	5-7-80045 803 01 0001	0000	10/03
Comp Blade Stg 4	212 010	5-7-80045 803 01 0002	0204	09/29
Comp Blade Stg 4	Zyg 020	5-7-80045 803 01 0002	0000	09/30
Comp Blade Stg 4	Ins 030	5-7-80045 803 01 0002	0000	10/06
Comp Blade Stg 8	Zyg 020	5-7-80045 803 01 0003	0000	09/29
Comp Blade Stg 8	Ins 030	5-7-80045 803 01 0003	0000	10/03
Bld Comp Stg 16	Ins 030	5-7-80045 803 01 0004	0000	10/03
Bld Comp Stg 9	Zyg 020	5-7-80045 803 01 0005	0000	09/29
Bld Comp Stg 9	Ins 030	5-7-80045 803 01 0005	0000	10/03
C/R Blade Stg 10	212 010	5-7-80045 803 01 0006	0495	09/29
C/R Blade Stg 10	Zyg 020	5-7-80045 803 01 0006	0000	10/03
C/R Blade Stg 10	Ins 030	5-7-80045 803 01 0006	0000	10/09
C/R Blade Stg 1	Ins 030	5-7-80045 803 01 0007	0000	09/29
C/R Blade Stg 1	Zyg 120	5-7-80045 803 01 0007	0000	10/01
C/R Blade Stg	212 010	5-7-80045 803 01 0008	0609	09/29
C/R Blade Stg	Zyg 020	5-7-80045 803 01 0008	0000	10/06
C/R Blade Stg	Ins 030	5-7-80045 803 01 0008	0000	10/10
C/R Blade Stg	Zyg 020	5-7-80045 803 01 0009	0000	09/30
C/R Blade Stg	Ins 030	5-7-80045 803 01 0009	0000	10/06
C/R Blade Stg 12	Zyg 020	5-7-80045 803 01 0010	0000	09/30
C/R Blade Stg 12	Ins 030	5-7-80045 803 01 0010	0000	10/06
C/R Blade	Zyg 020	5-7-80045 803 01 0011	0000	09/30
C/R Blade	Ins 030	5-7-80045 803 01 0011	0000	10/06
C/Disc Stg 5	212 020	5-7-80045 806 01 0001	0050	09/29
C/Disc Stg 5	268 030	5-7-80045 806 01 0001	0100	10/07
C/Disc Stg 5	212 040	5-7-80045 806 01 0001	0025	10/09
C/Disc Stg 5	Ins 050	5-7-80045 806 01 0001	0000	10/14
C/Disc Stg 6	212 020	5-7-80045 806 01 0002	0050	09/29
C/Disc Stg 6	268 030	5-7-80045 806 01 0002	0100	10/09
C/Disc Stg 6	212 040	5-7-80045 806 01 0002	0025	10/13
C/Disc Stg 6	Ins 050	5-7-80045 806 01 0002	0000	10/15
C/Disc Stg 7	212 020	5-7-80045 806 01 0003	0050	09/29
C/Disc Stg 7	268 030	5-7-80045 806 01 0003	0100	10/10
C/Disc Stg 7	212 040	5-7-80045 806 01 0003	0025	10/14
C/Disc Stg 7	Ins 050	5-7-80045 806 01 0003	0000	10/16
Stub Shaft	Ovn 010	5-7-80045 807 01 0001	0000	09/29
Stub Shaft	263 020	5-7-80045 807 01 0001	0040	10/03
Stub Shaft	Ins 030	5-7-80045 807 01 0001	0000	10/07
Comp Disc Stg	Ins 050	5-7-80045 809 01 0001	0000	10/09

Fig. 5—Output for Scheduling Office.

PRODUCTION SCHEDULING																
DAY BY DATE LOAD 09/22/58																
Area	Day 1 to 42															
201	48.0	48.0	48.0	48.0	48.0	0.0	0.0	48.0	48.0	48.0	48.0	48.0	0.0	0.0	Capacity/day	48.0 hrs.
—	48.0	48.0	48.0	48.0	24.0	0.0	0.0	13.1	0.0	2.4	1.8	0.9	0.0	0.0	—	—
—	0.6	0.8	0.6	3.7	3.5	0.0	0.0	0.0	2.5	2.0	1.0	0.0	0.0	0.0	—	—
203	32.0	32.0	32.0	32.0	31.0	0.0	0.0	13.5	13.5	30.5	30.5	0.0	0.0	0.0	Capacity/day	32.0 hrs.
—	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	—	—
—	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	—	—
206	32.0	14.5	23.5	15.5	0.0	0.0	0.0	6.0	16.5	20.0	14.9	12.1	0.0	0.0	Capacity/day	32.0 hrs.
—	16.0	13.0	10.0	8.0	0.0	0.0	0.0	0.0	12.0	8.0	0.0	0.0	0.0	0.0	—	—
—	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	—	—
207	32.0	32.0	32.0	32.0	32.0	0.0	0.0	32.0	32.0	25.0	2.5	0.0	0.0	0.0	Capacity/day	32.0 hrs.
—	0.0	17.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	—	—
—	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	—	—
208	32.0	32.0	32.0	32.0	32.0	0.0	0.0	17.0	3.5	5.0	0.0	18.0	0.0	0.0	Capacity/day	32.0 hrs.
—	0.0	12.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	—	—
—	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	—	—
209	16.0	15.5	0.0	0.0	0.0	0.0	0.0	0.0	2.5	0.0	6.0	12.0	0.0	0.0	Capacity/day	16.0 hrs.

Fig. 6—Listing of punched-card output.

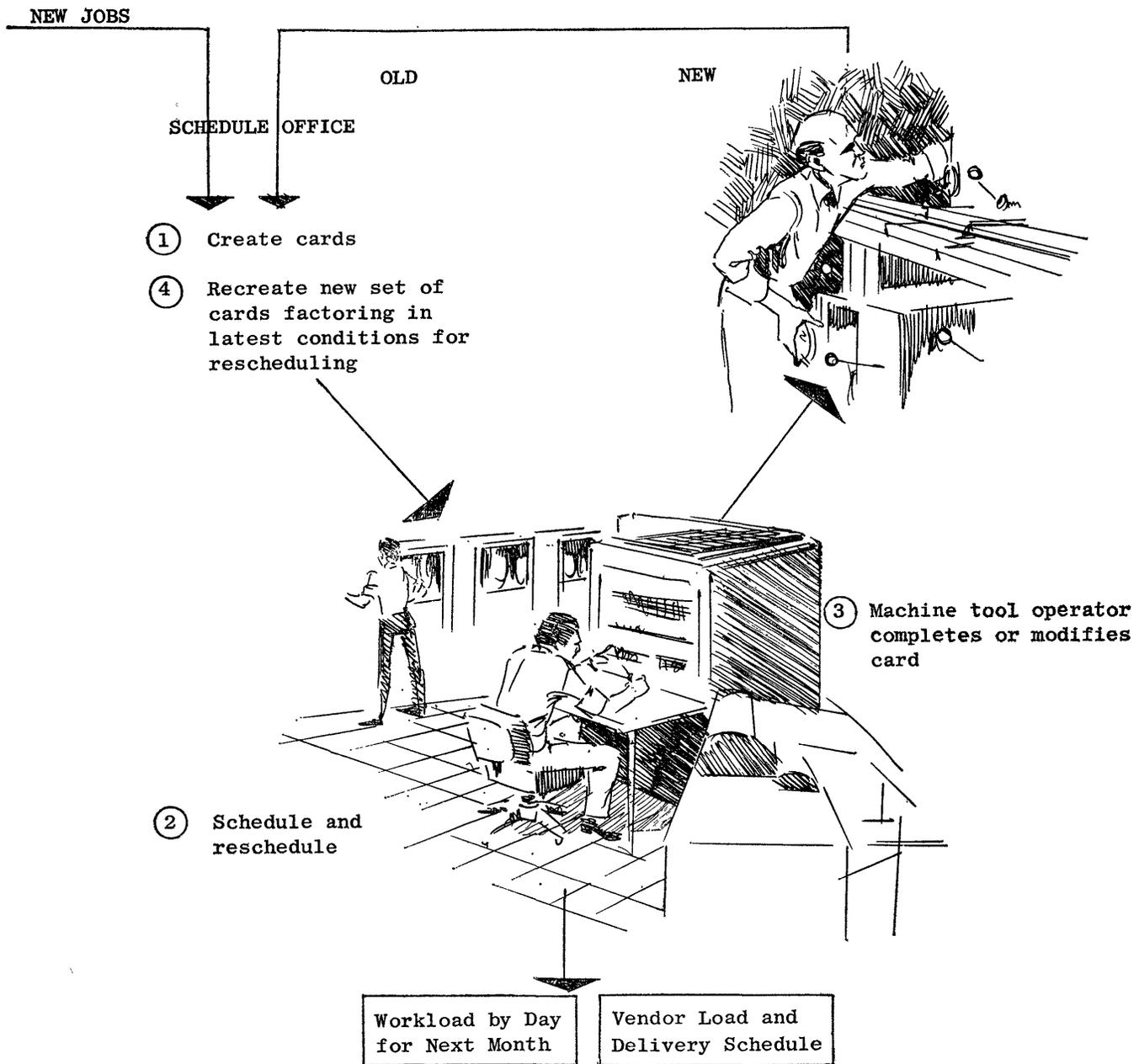


Fig. 7—Information flow to Computations facility and back to Production Shop.

are the following. Our 32K memory is completely filled twice during running. Six tape units are required. The computer running time is 30 minutes. In addition, two hours' handling time is required to prepare the input.

At one time an experimental task force tried to do the complete scheduling job by hand. By the end of four days of round-the-clock work, they threw up their hands. The machine system has now been in operation several months with a resultant 4 per cent drop in idle labor. Savings the first year, above the cost of the program, are estimated at a minimum of \$31,000, but are probably much greater.

In summary, the advantages of the system are

1) realistic, up-to-date and on-time schedules,

2) better work loading for minimum idle time of men and machines,

3) compact and easy method of placing work in areas,

4) use of the program to predict results of crash programs on delivery schedules and dates.

An added advantage not foreseen is that a tool dispatcher now places tools at all machines prior to the workmen starting on the job, instead of having the workmen obtain their own tooling after receiving the job. This is possible since we know what type of work will be at each machine tool and on what day. Delays in schedule due to tool shortages no longer occur.

As for future plans, we are making a study of scheduling the entire integrated operation as seen in Fig. 1.

Numerical Methods for High-Speed Computers— A Survey*

GEORGE E. FORSYTHE†

CLASSIFICATION OF PROBLEMS

NUMERICAL analysis is the science and art of using digital computing machines to carry out scientific computations, excluding pure data processing. Because of the recent changes in computers, the field is dominated today by the problems of using the large stored-program digital computers. Analog computers, though important both for their output and as a source of methods (too seldom exploited) for digital computers, are not considered here. It is possible to give a rough mathematical classification of the types of computing problems ordinarily met. The following classification is adapted from Forsythe [4].

Approximation

- 1) Evaluate functions of one or more variables. Integrate, differentiate, and interpolate them. Sum a series.
- 2) Approximate functions of one or more variables by simpler functions in the sense of least squares, least deviation, or other norms.
- 3) Test empirical data for blunders and fill in missing data. Smooth such data, and fit them with typical curves. Integrate and differentiate them. Analyze their spectra.

Algebra and Number Theory

- 4) Solve a number of simultaneous algebraic or transcendental equations.
 - a) Linear.
 - b) Nonlinear, including algebraic eigenvalue problems and polynomial equations.
- 5) Solve systems of linear or nonlinear inequalities.
 - a) Programming problems.
 - b) Problems from game theory.
- 6) Combinatorial problems (dealing with functions of permutations).
- 7) Problems from number theory.

Analysis and Functional Equations

- 8) Find the maximum of a function of one or more variables.
- 9) Solve initial-value problems for one or more ordinary or partial differential equations.
- 10) Solve boundary-value problems for one or more ordinary or partial differential equations.
- 11) Solve problems for other functional equations, including, for example,

- a) Differential equations with retarded arguments.
- b) Difference-differential equations.
- c) Conformal mapping problems.

Simulation

- 12) Simulation of random noise—for example, with a prescribed power spectrum.
- 13) Simulation of physical systems as an alternative to setting up equations for them, *e.g.*, traffic flow, or animal nervous systems.

THE WIDE SCOPE OF NUMERICAL ANALYSIS

In treating any of the above classes of problems, there is a very wide class of activities which comes under the heading “numerical analysis,” and an additional group of almost inseparable activities called “mathematical analysis” at the origin of the problem, and “programming” or “coding” on the machine end. Let us briefly review these.

Problem Formulation

The formulation of the mathematical problem to be solved is applied mathematical analysis, and not numerical analysis. However, the computational feasibility of the problem is a very important factor in the selection of a mathematical problem by which to model a given physical problem.

Replacement by an Algebraic Problem

Many of the mathematical problems of problem formulation A are not algebraic. Basically, digital computers can only add, subtract, multiply, and divide. Hence, to be tractable by a computer a problem must be algebraic—dealing with only a finite number of variables. The reduction of a transcendental problem to a related algebraic problem (or sequence of algebraic problems) may be called “discretization,” and the corresponding error, the “discretization error.” The study of discretization is an important part of numerical analysis.

Design of a Theoretical Algorithm

Having an algebraic problem to deal with, numerical analysts design algorithms with which to solve it. At this stage it is expedient and customary to think of ideal algorithms in which exact arithmetic operations are carried out with real numbers. Some algorithms are “direct,” terminating with the answer to the algebraic problem in a finite number of steps. Others are “iterative,” and attain the answer only as the limit of a sequence of steps.

* The writing of this exposition was sponsored by the Office of Naval Res., under Contract Nonr-225(37), NR 044 211.

† Stanford University, Stanford, Calif.

Convergence and Errors of Iterative Algorithms

Numerical analysts study the convergence and other asymptotic properties of iterative algorithms. The fact of convergence is almost essential to their use, while the speed of convergence is economically important. Knowledge of the asymptotic behavior assists one in accelerating the convergence, which is almost always too slow. Finally, at a higher level of sophistication, one ought to have at any stage of the iteration a means of bounding the difference between the current iterate and the exact solution of the algebraic problem.

Digitalization and Round-Off Error

We know that digital computers do not deal with real numbers, but only with rational numbers. In practice, computers usually use only special kinds of rational numbers—short terminating fractions to base 2 or 10, called “digital numbers.” The arithmetic operations heretofore discussed are replaced by various pseudo-operations. This introduces round-off errors and raises all kinds of difficult questions about the algorithm and its utility. These, too, are treated by numerical analysis, and represent a major point of contact with actual computers.

Programming and Coding

Actually getting problems on a digital computer, the goal of programming and coding, is so closely related to numerical analysis that it is difficult to separate them. The selection of an algorithm is bound up with the available means for approximating it digitally on a computer. For many problems, like the solution of Dirichlet's problem over a complicated region, the machine determination of difference equations (hand determination would be unthinkable for large problems) requires a code far more difficult than the machine solution of the equations.

The wide range of activity naturally forces practicing numerical analysts to specialize. I think it is vital that such specialization be along “vertical” lines, *i.e.*, that each analyst deal with problems from their formulation right through their coding and operation, confining himself, if necessary, to a few problems. In this way the feedback between machine peculiarities and problem formulation is most apt to be effective. This feedback should lead to the most rapid evolution of mathematical methods for using the new computers. As an immediate benefit, such a plan of operation should be the truest safeguard against major blunders, for such blunders arise most easily in areas of misunderstanding between different responsible persons.

Unfortunately, the organization of mathematical groups in many companies is in just the opposite direction. Engineering groups and mathematical analysis groups emerge, as well as programming groups and machine operating groups, and there is resistance to relaxed communication among them. It would be a great

contribution to the use of computers—and to our entire technology—if some organizational genius could find a method of keeping problems in the hands of very small responsible groups from their inception to their solution.

CURRENT STATUS OF THE MATHEMATICAL AREAS

It is manifestly impossible to survey here the current status of all areas of numerical analysis. Let me merely point to a few sources of new research in the fields, following the outline of the first section. Many exceedingly important new contributions will necessarily fail to be mentioned here.

Approximation

A major conference on approximation was held at the Army Mathematical Center in Madison, Wis., in April, 1958. The proceedings of that conference, to be published, will be an excellent synopsis of current thought on approximating functions. Problems of numerical integration, differentiation, etc., can be given a formulation in terms of finding nearest vectors in such function spaces as Hilbert space. (See Sard [16].) While more conventional approaches, based largely on polynomial interpolation, are probably an adequate basis for dealing with functions of one variable, it is probable that approximation problems for functions of two or more variables will be solved effectively only in terms of function spaces. And it was quite clear at the Madison conference that the great need now is for suitable methods for integrating, interpolating, and approximating functions of several variables.

The problem of estimating the power spectrum of a random process from a sample time function has become exceedingly important in modern engineering. The problem is one of statistical estimation, and progress is coming largely from statisticians like Tukey [13].

One area in approximation theory, that of minimax approximation, is receiving considerable attention. It has been known for years that polynomial and rational functions of least deviation from $f(x)$ over an interval are characterized by residuals with a certain equal-alternation property. Since a computer can generate any rational function of x , and only rational functions of x (we ignore problems of digitalization), it is natural to seek rational approximations to the common transcendental functions of analysis. Hastings and his collaborators [6] have published a book of useful approximations obtained by hand-tailoring methods. But it is widely agreed that digital computers can and should generate such approximations. Work of Remez [14] led to an algorithm by Novodvorskiĭ and Pinsker [11] for this purpose. The algorithm has been improved and coded at various computer centers, but it is not yet clear that a best procedure has been found. The problem is really one of linear programming with an infinite number of conditions.

While it is an attractive and interesting numerical analysis problem, for example, to find the rational function of degree 10 which best approximates $\cos x$ over an interval, this may not in fact be the most appropriate approximation to $\cos x$ in practice. It may be that a piecewise quadratic approximation would be faster to use, only slightly more bulky to store, and easier to determine. Professor C. B. Tompkins has mentioned that such an approximation is very easily tailored by an automatic computer to fit detailed specifications. Thus we must always ask whether numerical analysts are in fact applying their talents to solve the right problems.

Algebra

Problems of linear algebra and the solution of polynomial equations have been studied extensively during the present decade. In his 1958 lectures in Ann Arbor, Wilkinson [19] greatly increased our understanding of several common methods for these problems. His forthcoming book will contain the same material and more, while a little of the material is already available [2]. Wilkinson's most important basic contribution seemed to be a satisfactory and realistic analysis of the round-off error in Crout's method for Gaussian elimination. The importance of "searching for a pivot" was demonstrated conclusively.

Wilkinson discusses the calculation of the eigenvectors of a tridiagonal matrix, a necessary part of either the Givens or Lanczos methods for solving the complete eigenvalue-eigenvector problem for finite matrices [20]. He also gave the following startling example of the instability of the roots of a polynomial. Let

$$\begin{aligned} f(x) &= (x+1)(x+2)\cdots(x+20) \\ &= x^{20} + 210x^{19} + \cdots, \end{aligned}$$

with zeros $-1, -2, \dots, -20$. Suppose just one coefficient of $f(x)$ is slightly perturbed, say because of rounding, so that instead of $f(x)$ one deals with $g(x) = f(x) + 2^{-23}x^{19}$. Then Wilkinson has found that the zeros of g include numbers near -20.846 , and $-13.99 \pm 2.5i$. In face of this evidence, no one should ever again blindly ask any automatic routine to find the zeros of a polynomial of importance!

Householder [7] has reported a new algorithm which might be an improvement on Givens' very excellent routine for getting eigenvalues of symmetric matrices. For unsymmetric matrices the eigenvalue problem remains one for which there are many routines, with none of them outstanding enough to become standard. Ostrowski [12] has published one of the first papers bounding the variation in the eigenvalues of an unsymmetric matrix, due to perturbations in its elements. This has very important consequences for round-off error. It will be important to improve Ostrowski's results for various special cases where root multiplicity can be guaranteed to be less than the order of the matrix.

I shall not attempt to report on the progress in dealing with inequalities, combinatorial problems, or number theory.

Functional Equations

The status of the numerical solution of ordinary differential equations is being reported in a forthcoming book by Henrici, starting from the points of view of Rutishauser, Dahlquist, and others. The well-known methods of Milne [9] have generally proved unstable on automatic computers, although recent modifications may save them [10].

A recent book by Richtmyer [15] treats difference methods for marching problems of partial differential equations. A forthcoming book by Forsythe and Wasow will deal with difference methods for linear partial differential equations of second order, including both marching and jury problems. The concept of stability will be quantified somewhat for marching problems. The work of Young, and Peaceman-Rachford and successors, will be recorded. Work by Kahan and others on overrelaxation without Young's property A will be mentioned.

As indicated earlier, a large practical problem in putting Dirichlet's problem on a computer is to generate the difference equations. A very substantial project, called SPADE, is operating under SHARE to automate the generation and solution of difference equations. Such work is as important as it is difficult, and needs solid sponsorship.

The work of Gerschgorin is vital in bounding the discretization error due to replacing an elliptic boundary-value problem by difference equations. The Gerschgorin result has been extended by Wasow [18]. Recent Russian work is extending the results to cases where the unknown solution does not have bounded third derivatives. (See Volkov [17].)

The Gerschgorin theory has not been developed to deal with internal interfaces, and it would be highly desirable to make the extension. There are many questions open for investigation in connection with graded nets, regions with corners, and so on.

Simulation

About simulation there is only one topic I wish to mention. An increasingly important area in modern engineering is that of random noise. The simulation of random noise is important in many ways, and methods for analog computers are discussed, for example, by Laning and Battin [8]. The author holds that a well-designed digital computer can do anything an analog computer can, and better, provided that the coders know what to code. It seems to me time that someone investigated and wrote about the simulation on a digital computer of random noise with a prescribed power spectrum or autocorrelation function.

TREATMENT OF ERROR

Before the advent of computers one thought about discretization errors, and these are still important. In dealing with desk computations one had to worry about arithmetic blunders, but these are comparatively rare in automatic computation. But the characteristic error in much automatic computation is an error due to the digitalization of the numbers in a computer, with resultant approximation of the arithmetic operations, together with the cumulative effects of these errors through numerous subsequent operations. One now finds several points of view emerging with regard to these errors.

1) Let me illustrate one new point of view in the context of the solution of a linear algebraic system $Ax=b$, although it has far wider applications. Here A is a given nonsingular matrix of digital elements, while b is a given vector of digital components. Let $h=A^{-1}b$ be the true solution of the system. Suppose one has found a digital vector x which approximately solves the system. How does one measure the error in x ?

The conventional concept of error is the vector $x-h$, or some number $\|x-h\|$ which measures its smallness. The vector $x-h$ answers the question, how wrong is x as a solution of the problem with the given data A, b ? It is comparatively difficult to estimate $\|x-h\|$, since it requires knowledge of the size of A^{-1} .

Givens [5] and Wilkinson [19] are asking a different question in formulating the concept of the error in solving $Ax=b$. Namely, how necessary is it to alter the given data A, b to new values A_1, b_1 , in order that the approximate solution x correctly solve the altered problem $A_1x=b_1$? They consider the matrix $A-A_1$ and the vector $b-b_1$ as a definition of the error in x . Again, they might adopt some number of the form $\epsilon = \{\|A-A_1\|^2 + \|b-b_1\|^2\}^{1/2}$ as a measure of the smallness of $A-A_1$ and $b-b_1$.

Of course A_1 and b_1 are not uniquely determined by x . With suitably chosen norm functions one could force uniqueness by demanding that A_1, b_1 be *closest* to A, b , in the sense of minimizing the ϵ defined above; let $\min \epsilon$ denote that minimum value. However, it would probably be difficult to compute $\min \epsilon$ or the minimizing quantities A_1, b_1 , and all that is really needed is a reasonable upper bound for $\min \epsilon$. The real power of the Givens-Wilkinson suggestion is that it is easy to bound $\min \epsilon$ in solving linear systems by a number of methods. I shall not be able to go into this, except to mention the use of the *residual* $b-Ax$.

One simple way to bound $\min \epsilon$ has been used for a long time. Let $b_1=Ax$, and let $A_1=A$. Then $A_1x=b_1$, and $\min \epsilon \leq \|b-b_1\| = \|b-Ax\|$. Here one has concentrated the necessary alteration of the data in the vector b . The vector $b-Ax$ is easy to compute and is often already available as a byproduct of a solution of $Ax=b$. The idea of measuring the error in x by the size of the

residual $b-Ax$ is not a new one, of course, but it has usually been considered only as an expedient substitute for the "real error" $x-h$. What is new in recent thinking is to consider that such a number as $\|b-Ax\|$ (or the more general ϵ) may often be a fair measure of the error in solving $Ax=b$. Of course, the decision as to what is an appropriate measure of the error in solving a system can only be decided after an examination of the physical problem underlying the computation.

2) A second major development is the advent of codes which precisely bound the error of a calculation. Perhaps the most promising are the *range arithmetic codes* prepared under the supervision of Ramon Moore at the Lockheed Missile Systems Division, Palo Alto, Calif. In these codes, the operands are the range numbers of Dwyer [3], *i.e.*, real intervals. The answer to a range multiplication, for example, is the smallest digital interval which includes all products of pairs of numbers chosen from the intervals of both factors.

Range arithmetic seems to be the most efficient form of error computation by a machine, and machine computation seems far easier and more efficient than human bounding of errors. It is to be hoped that range arithmetic will find its way into all algebraic translators, as an optional form of arithmetic.

3) There have been suggestions by Wilkinson, Carr, and others, that problems be computed several times—for example, one each with eight, nine, ten, eleven, twelve, and thirteen decimals. From a comparison of the answers, one should be able to guess the round-off errors quite satisfactorily in some problems. Such ability to do arithmetic with variable word length is not easily available on most machines, but should be made available.

4) Givens has emphasized the importance of having programs which print out guaranteed bounds for answers to problems. For example, if a polynomial is input (perhaps by means of its coefficients as range numbers), an ideal routine might output in effect several rectangles in the complex plane, together with the number of zeros in each rectangle. Such a routine should be without the slightest mathematical error; its assertions should be known to be correct for a specified class of polynomials. And, if the input polynomial turns out not to be in that class, that fact should be output.

One knows in principle how to write such routines for various types of problems. What is lacking is definitive experience with them, for very few have been actually written. With polynomial zero-finders, it is certain that acceptably small rectangles can be found, by use of multiple-precision arithmetic, if necessary. But for ordinary differential equations, for example, it may be that no realistic solution bounds could ordinarily be found, if they must be rigorously correct. A good way to settle this question is to accumulate much experience with well-written codes taking full advantage of the speed of current machines.

FREE EXPERIMENTATION AND TEAMWORK OF
MAN AND MACHINE

It is safe to say that we are all using automatic digital computing machines very badly. It is characteristic of humans to use new tools as though they were extensions of old tools, and only to devise appropriate new methods very slowly. It would be most desirable to speed up our process of accommodation to the new machines, since otherwise one wonders if numerical analysts will ever catch up with computer engineers!

I suppose that the surest road to progress is to encourage imaginative experimentation by those close to machines, and especially to attract imaginative people to computer laboratories. I feel it would help if machine administrators would allow—yes, and encourage—their most imaginative numerical analysts to play personally with the newest machines as much as possible.

In the early days of automatic computation, research workers with important computing problems, physical or mathematical, would always be present when their computations were being carried out on automatic computers. They would help trouble-shoot the code, and use the preliminary results to suggest new cases to try and parameters to vary. The same method of operation is customary now with analog computation.

However, it was discovered that inquisitive research workers would waste machine time while they pondered the problem, and probably they sometimes got in the way of the machine operators. As a result, in many companies programmers and research workers are effectively discouraged from attending machine runs. Elaborate automonitoring and other efficient routines enable the machine to operate automatically for hours at a time, running codes which are preassembled on tapes. As a result, the *apparent* efficiency of a computing laboratory increases, when measured in good operating time per month. However, the *real* efficiency may well be found to decrease, when measured in problems solved per month. The reason is that the machine is being deprived of its most valuable component—an intelligent human who knows the problem. Knowing he will not be present at the console, the coder tries to cover every eventuality, and sometimes asks for ten times as much computing as would be necessary if he could be present. Is this efficiency?

There are sound economic reasons for insisting on reasonable efficiency in the machine room, and I do not advocate wasteful practices. But the gain from reasonable efficiency should include the gain from reasonable use of man as the most important ally of the machine. I feel that if a scientist wishes to work with the machine, it is up to the administrators to design methods whereby this collaboration can be efficiently and effectively carried out. Machine designers should also bear this in mind, and provide enough visual monitors and enough conveniently adjustable console registers, so that man

and machine may intercommunicate quickly and effectively. In this way man can return to the automatic digital computing loop in the effective way he is present in other types of computation.

It must be admitted that many administrators disagree with the last paragraph. They feel that an automatic digital computing loop can operate quite effectively at the slower rate which occurs with unattended runs, when the analyst ponders the output overnight, returning next day with his new input. And the collaboration of man and machine may sometimes be more effective because of the extra time spent by the analyst in pondering his next move.¹

I am sure that unattended runs are often adequate, and especially often for the more easily analyzed and better understood classes of problems. But it is characteristic of difficult problems that they are only imperfectly understood, so that the analyst may be in grave doubt about the choice of computing method and about various other matters, and even about where trouble will occur. But even at high speed it is often possible to read orders of magnitude from console registers and thus monitor a computation. For some problems an analyst in attendance can come prepared with various cards, for example, and can interpose changed values or alternate codes very early in a long computation, at a large net saving of computer time over just letting the machine run unattended. Can't we agree to give a scientist a few seconds to make such adjustments during a run? This is not to ask for many minutes for pondering a surprising turn of events—that would be very wasteful by any criterion.

I suppose the decision as to attending a run should be the decision of how a creative but responsible man feels he can best operate. A creative scientist is a rare person, and in his research should be permitted wide latitude in choosing his hours, location, and methods—including his technique of collaboration with a machine. In fundamental terms, computers can be manufactured in great numbers, and there are many persons capable of administering their operation. The really critical shortage remains that of creative scientists and engineers who can make effective use of their tools. Hence, at least on any problem of importance, the time of a scientist is really more valuable than that of a machine, and all obstacles should be cleared away from his use of a machine in the way he feels will solve his problem most effectively. No one should ever compare a creative man with a machine on the basis of their dollar costs per hour!

LITERATURE ON NUMERICAL ANALYSIS

In a field growing as fast as numerical analysis, it is difficult even to learn the names of important new books and journals, let alone read them. I shall mention here

¹ The author is indebted to Dr. Walter F. Bauer for an exchange of ideas on this matter.

the names of a few journals which have a fair number of articles or abstracts on the numerical analysis appropriate to high-speed digital computers.

Journals

Chiffres (France), *The Computer Journal* (England), *Communications of the Association for Computing Machinery*, *Journal of the Association for Computing Machinery*, *Journal of Research of the National Bureau of Standards*, *Journal of the Society for Industrial and Applied Mathematics*, *Mathematical Reviews*, *Mathematical Tables and Other Aids to Computation*, *Zeitschrift für numerische Mathematik* (West Germany), *Proceedings of the Cambridge Philosophical Society*, *Quarterly of Applied Mathematics*, *Quarterly Journal of Mechanics and Applied Mathematics* (England), *Referativnyi Zhurnal—Matematika* (Soviet Union), *Review of the Society for Industrial and Applied Mathematics*, *Vychislitel'naiia Matematika* (Soviet Union), *Zeitschrift für angewandte Mathematik und Mechanik* (East Germany), *Zeitschrift für angewandte Mathematik und Physik* (Switzerland), *Zentralblatt für Mathematik und ihre Grenzgebiete*.

Any library associated with a digital computer center should have current subscriptions of all these journals, together with as complete a back file as possible.

Books

Books on numerical analysis which take real account of automatic computation have been slow to appear, perhaps because the field changes so rapidly. I should like merely to call your attention to two books which are too recent to be well known. Complete citations are in the bibliography below. The first is "Modern Computing Methods" [2], written anonymously by four mathematicians in the National Physical Laboratories, near London. It is thin but full of good material by persons intimately involved in computation, both before and after the computer revolution. The bibliography of 128 titles is completely annotated.

The second is a general survey [1] of machines by Alt, including coding and problem analysis. There is a 12-page bibliography, implicitly annotated by cross-references to the text.

REFERENCES

- [1] F. L. Alt, "Electronic Digital Computers, Their Use in Science and Engineering," The Academic Press, Inc., New York, N. Y., and London, Eng., 336 pp.; 1958.
- [2] Anonymous, "Modern Computing Methods," The Philosophical Library, Inc., New York, N. Y., 129 pp.; 1958 (said to be written by L. Fox, E. T. Goodwin, F. W. J. Olver, and J. H. Wilkinson).
- [3] P. S. Dwyer, "Linear Computations," John Wiley and Sons, Inc., New York, N. Y., 344 pp.; 1951.
- [4] G. E. Forsythe, "Contemporary state of numerical analysis," in G. E. Forsythe and P. C. Rosenbloom, "Numerical Analysis and Partial Differential Equations," John Wiley and Sons, Inc., New York, N. Y., pp. 1-42; 1958.
- [5] W. Givens, "Numerical Computation of the Characteristic Values of a Real Symmetric Matrix," Oak Ridge Natl. Lab., Oak Ridge, Tenn., Rep. ORNL 1574, 107 pp.; 1954.
- [6] C. Hastings, Jr., "Approximations for Digital Computers," Princeton University Press, Princeton, N. J., 201 pp.; 1955.
- [7] A. S. Householder, "Some Mathematical Problems Arising in Matrix Computations," Oak Ridge Natl. Lab., Oak Ridge, Tenn.
- [8] J. H. Laning, Jr. and R. H. Battin, "Random Processes in Automatic Control," McGraw-Hill Book Co., Inc., New York, N. Y., 434 pp.; 1956.
- [9] W. E. Milne, "Numerical Solution of Differential Equations," John Wiley and Sons, Inc., New York, N. Y., 275 pp.; 1953.
- [10] W. E. Milne and R. R. Reynolds, "Stability of a numerical solution of differential equations," *J. Assoc. Comput. Mach.*, to be published.
- [11] E. N. Novodvorskiĭ and I. S. Pinsker, "On a process of equalization of maxima" (Russian), *Uspehi Matem. Nauk*, vol. 6, no. 6, pp. 174-181; 1951.
- [12] A. Ostrowski, "Mathematische Miszellen XXVII. Über die Stetigkeit von charakteristischen Wurzeln in Abhängigkeit von den Matrizenelementen," *Jahresber. Deutsch. Math. Verein.*, vol. 60, pp. 40-42; July, 1957.
- [13] H. Press and J. W. Tukey, "Power Spectral Methods of Analysis and their Application to Problems in Airplane Dynamics," reprinted by Bell Telephone System as Monograph 2606, 41 pp.; 1956.
- [14] E. I. Remez, "Obshchie Vychislitel'nye Metody Chebyshevskogo Priblizheniia," Akademiya Nauk Ukrainskoi SSR, Kiev, U.S.S.R., 454 pp.; 1957.
- [15] R. D. Richtmyer, "Difference Methods for Initial-Value Problems," Interscience Publishers, Inc., New York, N. Y., 238 pp.; 1957.
- [16] A. Sard, "Best approximate integration formulas; best approximation formulas," *Amer. J. Math.*, vol. 71, pp. 80-91; January, 1949.
- [17] E. A. Volkov, "On the question of solving the interior Dirichlet problem for Laplace's equation by the method of nets," (Russian), *Vychislitel'naia Matemat.*, Akademiya Nauk SSSR, Moscow, U.S.S.R., no. 1, pp. 34-61; 1957.
- [18] W. Wasow, "Discrete approximations to elliptic differential equations," *Z. angew. Math. Phys.*, vol. 6, pp. 81-97; March, 1955.
- [19] J. H. Wilkinson, lecture notes on matrix methods, to be published by the University of Michigan as part of notes on Advanced Numerical Analysis for Summer, 1958, by the University of Michigan Engineering Summer Conferences, East Engineering Bldg., Ann Arbor, Mich.
- [20] J. H. Wilkinson, "The calculation of the eigenvectors of codiagonal matrices," *Computer J.*, vol. 1, pp. 90-96; July, 1958.

More Accurate Linear Least Squares

RICHARD E. VON HOLDT†

THE LINEAR LEAST-SQUARES PROBLEM

LET A be a given matrix of m rows and n columns, ($m \geq n$), so that $AW=0$ implies $W=0$ (i.e., the column vectors of A constitute a linearly independent set), and let Y be a given m -dimensional vector. We seek an n -dimensional vector \bar{X} , so that $|R(\bar{X})|^2$ is the minimum value of $|R(X)|^2$ where

$$R(X) = AX - Y. \quad (1)$$

A GEOMETRIC DERIVATION OF THE CLASSICAL SOLUTION TO THE LINEAR LEAST-SQUARES PROBLEM

Let S be that subspace of m -dimensional Euclidean space which is spanned by the column vectors of A . Then for arbitrary X , AX is a vector in S , and $R(X)$ is a vector with initial point at the terminal point of Y and terminal point in S .

Let $A\bar{X}$ be the orthogonal projection of Y onto S . Then

$$R(\bar{X}) = A\bar{X} - Y, \quad (2)$$

is orthogonal to S , or

$$A^T R(\bar{X}) = 0. \quad (3)$$

For arbitrary X , we have

$$\begin{aligned} R(X) &= [R(X) - R(\bar{X})] + R(\bar{X}) \\ &= A(X - \bar{X}) + R(\bar{X}). \end{aligned} \quad (4)$$

From (3) and (4), for arbitrary X , we have

$$|R(X)|^2 = |A(X - \bar{X})|^2 + |R(\bar{X})|^2 \geq |R(\bar{X})|^2.$$

Thus $|R(\bar{X})|^2$ is the minimum value of $|R(X)|^2$ for arbitrary X . Substituting (2) into (3), \bar{X} must satisfy the relation:

$$A^T A \bar{X} = A^T Y. \quad (5)$$

From the hypothesis on A , we have

$$A^T A W = 0 \rightarrow W^T A^T A W \equiv |AW|^2 = 0 \rightarrow W = 0. \quad (6)$$

Hence $A^T A$ is a nonsingular $n \times n$ matrix and (5) has a unique solution.

THE SOLUTION OF (5) BY DIAGONAL PIVOTS

Let M_k , ($k=1, 2, \dots, n$), be the matrix obtained from A by deleting all but the first k columns of A . Then the columns of M_k form a linearly independent set and by the argument of (6), $M_k^T M_k$ is a nonsingular matrix, ($k=1, 2, \dots, n$).

Let P_1 be the determinant of $M_1^T M_1$ and P_k be the determinant of $M_k^T M_k$ divided by the determinant of $M_{k-1}^T M_{k-1}$, ($k=2, 3, \dots, n$). Then

$$P_k \neq 0, \quad (k=1, 2, \dots, n). \quad (7)$$

Using the diagonal elements, in increasing order, as pivots, and combining proper multiples of each row into all following rows to produce zeros below the pivot elements in the column containing the pivots in (5) does not change the value of any of the minors of $A^T A$ formed by deleting all but the first k rows and all but the first k columns of $A^T A$. Thus this process of replacing (5) by an equivalent upper-triangular system of equations yields the successive nonzero pivots:

$$P_1, P_2, \dots, P_n. \quad (8)$$

The above described process is equivalent to premultiplying both sides of (5) by a lower-triangular matrix L , with unit diagonal elements, yielding

$$L A^T A \bar{X} = L A^T Y \quad (9)$$

and this upper-triangular system is solved by back substitution.

THE SOLUTION OF (5) BY ORTHOGONALIZATION

Using the columns of A , in increasing order, as pivot columns, and combining proper multiples of each pivot column into all following columns so that the resulting columns are orthogonal to the pivot column, replaces the columns of A by an orthogonal basis for S . Let the matrix which results be denoted by B . Then

$$B = AU \quad (10)$$

where U is an upper-triangular matrix with unit-diagonal elements, and furthermore

$$B^T B = D, \quad (11)$$

where D is a diagonal matrix.

Premultiplying (5) by U^T and replacing \bar{X} by $UU^{-1}\bar{X}$, we have

$$DU^{-1}\bar{X} = B^T Y \quad (12)$$

and from this,

$$\bar{X} = UD^{-1}B^T Y. \quad (13)$$

COMPARISON OF METHODS

Since both $LA^T A$ and L^T are upper-triangular matrices, their product, $LA^T AL^T$, is also upper triangular, besides being symmetric, and is therefore a diagonal matrix. Thus AL^T is a matrix of mutually orthogonal

† Lawrence Rad. Lab., University of California, Livermore, Calif.

columns and since L^T is upper-triangular with unit-diagonal elements, and

$$L^T = U. \tag{14}$$

Again, since L^T is upper-triangular with unit-diagonal elements, the diagonal elements of $LA^T A$ and $D = U^T A^T A U = LA^T A L^T$ are identical, or

$$|B_k|^2 = P_k, \quad (k = 1, 2, \dots, n). \tag{15}$$

Let A_k and B_k be the k th columns of A and B , respectively, and let ϵ_k be a scala, defined by

$$|B_k| = \epsilon_k |A_k|, \quad (k = 1, 2, \dots, n). \tag{16}$$

Since B_k is a projection of A_k , we have

$$0 < \epsilon_k \leq 1, \quad (k = 1, 2, \dots, n), \tag{17}$$

and ϵ_k is a measure of the figure loss encountered in constructing B_k from A_k by orthogonalization, and there is no further figure loss by cancellation in computing,

$$P_k = |B_k|^2, \quad (k = 1, 2, \dots, n). \tag{18}$$

In the method of diagonal pivots, $|B_k|^2$ is formed from $|A_k|^2$ by repeated subtractions, and since

$$|B_k|^2 = \epsilon_k^2 |A_k|^2, \tag{19}$$

our measure of the figure loss in this method is ϵ_k^2 . Thus the method of orthogonalization has half the figure loss of the method of diagonal pivots.

R(\bar{X}) AS A BY-PRODUCT OF ORTHOGONALIZATION

Let Z_k , ($k=1, 2, \dots, n+1$) be defined by

$$Z_1 = Y \tag{20}$$

and

$$Z_{k+1} = Y - \sum_{i=1}^k B_j(B_j^T Z_j)/(B_j^T B_j) \quad (k = 1, 2, \dots, n). \tag{21}$$

Then

$$B_{k+1}^T Z_{k+1} = B_{k+1}^T Y, \quad (k = 1, 2, \dots, n - 1), \tag{22}$$

since the columns of B are mutually orthogonal. Setting $k=n$ in (21) and using (22), we have

$$Z_{n+1} = Y - \sum_{j=1}^n B_j(B_j^T Y)/(B_j^T B_j). \tag{23}$$

Thus Z_{n+1} is the component of Y orthogonal of S , or

$$Z_{n+1} = -R(\bar{X}). \tag{24}$$

THE INVERSE OF $A^T A$ AFTER THE APPLICATION OF THE METHOD OF ORTHOGONALIZATION

From (10), (11), and (14), we have

$$LA^T AL^T = D. \tag{25}$$

Since L and L^T are nonsingular,

$$A^T A = L^{-1} D (L^T)^{-1} \tag{26}$$

$$(A^T A)^{-1} = L^T D^{-1} L. \tag{27}$$

Since the calculation of D by the method of orthogonalization has half the figure loss of the method of diagonal pivots, the former method using (27) yields a computationally more accurate inverse. The elements of this inverse matrix are useful to statisticians in the Theory of Error Analysis.

DETAILS OF THE METHOD OF ORTHOGONALIZATION

The matrix $U = L^T$ need not be formed explicitly except in the evaluation of $(A^T A)^{-1}$ by (27).

Let U_k be the matrix which is the $n \times n$ identity except for the elements in the k th row to the right of the diagonal. These elements are the multiples of the k th column of B which are added to the corresponding following columns to yield new following columns, which are orthogonal to the k th column. Then

$$U = U_1 U_2 \dots U_{n-1}. \tag{28}$$

When: 1) the nonidentity elements of U_k have been formed and used to orthogonalize the following columns and Z_k to B_k ; 2) $B_k^T Y = B_k^T Z_k$ has been formed; and 3) $|B_k|^2$ has been formed, then B_i is no longer needed and the storage cells used for B_i are now available for storing the nonidentity elements of U_k and the scalar $B_k^T Y = (B^T Y)_k$. Repeating this process until $k=n$, \bar{X} is evaluated by

$$\bar{X} = U_1 U_2 \dots U_{n-1} D^{-1} (B^T Z) \tag{29}$$

where we take advantage of all the known zero elements of the matrices involved in performing the indicated matrix premultiplications.

For calculation of $(A^T A)^{-1}$, we have from (27) and (14)

$$(A^T A)^{-1} = (L_{n-1} L_{n-2} \dots L_2 L_1)^T D^{-1} (L_{n-1} L_{n-2} \dots L_2 L_1). \tag{30}$$

The nonidentity elements of the product

$$L_k (L_{k-1} L_{k-2} \dots L_1)$$

may be stored in the locations occupied by the non-identity elements of the two factors, ($k=2, \dots, n-1$). Having formed L , the diagonal and subdiagonal elements of $(A^T A)^{-1}$ can be formed and stored in the locations occupied by D and L .

CONCLUSION

Although the number of operations involved is greater in the method of orthogonalization than in the method of diagonal pivots, the increased accuracy is well worth the time and effort. It is to be noted that the method of orthogonalization for weighted polynomial fitting is equivalent to forming a set of weighted orthogonal polynomials, fitting the data to these polynomials, and reducing the combination of these polynomials to a single polynomial in the manner of Tchebycheff.

The CORDIC Computing Technique

JACK VOLDER†

THE “COordinate Rotation DIgital Computer” computing technique can be used to solve, in one computing operation and with equal speed, the relationships involved in plane coordinate rotation; conversion from rectangular to polar coordinates; multiplication; division; or the conversion between a binary- and a mixed-radix system.

The CORDIC computer can be described as an entire transfer computer with a special serial arithmetic unit, consisting of 3 shift registers, 3 adder-subtractors, and special interconnections. The arithmetic unit performs a sequence of simultaneous conditional additions or subtractions of shifted numbers to each register. This performance is similar to a division operation in a conventional computer.

Only the trigonometric algorithms used in the CORDIC computing technique will be covered in this paper. These algorithms are suitable only for use with a binary code. This fact possibly accounts for their late appearance as a numerical computing technique. Matrix theory, complex-number theory, or trigonometric identities can be used to prove rigorously these algorithms. However, to help give a more intuitive and pictorial understanding of the basic technique, plane trigonometry and analytical geometry are used in this explanation whenever possible.

First, consider two given coordinate components Y_i and X_i in the plane coordinate system shown in Fig. 1.

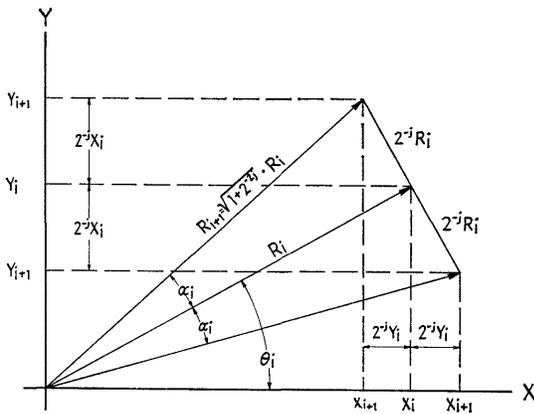


Fig. 1—Geometry of a typical rotation step.

The subscript i , as used in this report, will identify all quantities with a particular step in the computing sequence. The given components, Y_i and X_i , actually describe a vector of magnitude R_i at an angle θ_i from the origin according to the relationship,

$$Y_i = R_i \sin \theta_i \tag{1}$$

$$X_i = R_i \cos \theta_i. \tag{2}$$

With a very simple control of an arithmetic unit operating in a binary code, the sign of a number can be changed and/or the number can be divided by a power of two. Thus, if it is assumed that the numerical values of Y_i and X_i are available, the numerical values of both coordinates of one of the proportional quadrature vectors, R'_i , can be easily obtained.

$$Y'_i = 2^{-j} X_i \tag{3}$$

$$X'_i = -2^{-j} Y_i \tag{4}$$

where j is a positive integer or zero.

The vector addition of R'_i to R_i , by the algebraic addition of corresponding components, produces the following relationships:

$$Y_{i+1} = \sqrt{1 + 2^{-2j}} R_i \sin (\theta_i + \tan^{-1} 2^{-j}) = Y_i + 2^{-j} X_i \tag{5}$$

$$X_{i+1} = \sqrt{1 + 2^{-2j}} R_i \cos (\theta_i + \tan^{-1} 2^{-j}) = X_i - 2^{-j} Y_i \tag{6}$$

$$R_{i+1} = \sqrt{1 + 2^{-2j}} R_i. \tag{7}$$

Likewise, the addition of the other proportional quadrature vector at $\theta - 90^\circ$ to the vector R_i produces the following relationships:

$$Y_{i+1} = \sqrt{1 + 2^{-2j}} R_i \sin (\theta_i - \tan^{-1} 2^{-j}) = Y_i - 2^{-j} X_i \tag{8}$$

$$X_{i+1} = \sqrt{1 + 2^{-2j}} R_i \cos (\theta_i - \tan^{-1} 2^{-j}) = X_i + 2^{-j} Y_i \tag{9}$$

$$R_{i+1} = \sqrt{1 + 2^{-2j}} R_i. \tag{10}$$

If the numerical values of the components Y_i and X_i are available, either of the two sets of components Y_{i+1} and X_{i+1} may be obtained in one word-addition time with a special arithmetic unit (as shown in Fig. 2) operating serially in a binary code.

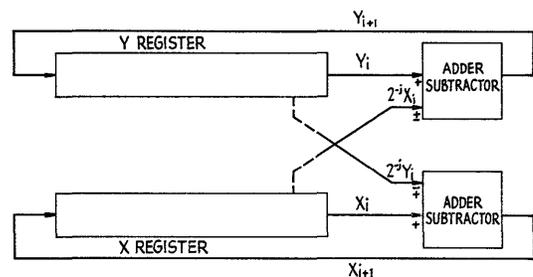


Fig. 2—Arithmetic unit for cross addition.

This particular operation of simultaneously adding (or subtracting) the shifted X value to Y and subtracting (or adding) the shifted Y value to X is termed “cross addition.”

† Convair, Fort Worth, Tex.

The effect of either of these two choices can be considered as a *rotation* of the vector R_i through the special angle plus (or minus) α_i where

$$\alpha_i = \tan^{-1} 2^{-i} \quad (11)$$

accompanied by an *increase in magnitude* of each component by the factor $(1+2^{-2i})^{\frac{1}{2}}$.

Note that this increase in magnitude is a function of the value of the exponent j and is independent of whichever of the two choices of direction is used. If a particular value of j is specified to correspond to a particular value of i in the general expression, and if it is specified that, for every i th term, one and only one of the two permissible directions of rotation is used to obtain the $i+1$ terms, then the choice may be identified by the binary variable ξ_i where $\xi_i = +1$ for positive rotation, or -1 for negative rotation. This gives a general expression for the $i+1$ terms as

$$Y_{i+1} = \sqrt{1+2^{-2i}} R_i \sin(\theta_i + \xi_i \alpha_i) = Y_i + \xi_i 2^{-i} X_i \quad (12)$$

$$X_{i+1} = \sqrt{1+2^{-2i}} R_i \cos(\theta_i + \xi_i \alpha_i) = X_i - \xi_i 2^{-i} Y_i \quad (13)$$

$$R_{i+1} = \sqrt{1+2^{-2i}} R_i. \quad (14)$$

After the components Y_{i+1} and X_{i+1} are obtained, another similar operation can be undertaken to obtain the $i+2$ terms.

$$\begin{aligned} Y_{i+2} &= \sqrt{1+2^{-2(j+1)}} [\sqrt{1+2^{-2j}} R_i \sin(\theta_i + \xi_i \alpha_i + \xi_{i+1} \alpha_{i+1})] \\ &= Y_{i+1} + \xi_{i+1} 2^{-(j+1)} X_{i+1} \end{aligned} \quad (15)$$

$$\begin{aligned} X_{i+2} &= \sqrt{1+2^{-2(j+1)}} [\sqrt{1+2^{-2j}} R_i \cos(\theta_i + \xi_i \alpha_i + \xi_{i+1} \alpha_{i+1})] \\ &= X_{i+1} - \xi_{i+1} 2^{-(j+1)} Y_{i+1} \end{aligned} \quad (16)$$

$$R_{i+2} = \sqrt{1+2^{-2(j+1)}} \sqrt{1+2^{-2j}} R_i. \quad (17)$$

Likewise, the pseudo-rotation steps can be continued for any finite, pre-established number of steps of pre-established increments but arbitrary values of sign. After these steps have been completed, the increase in magnitude of the vector as a result of these steps will be the constant factor

$$\begin{aligned} &\sqrt{1+2^{-2j}} \cdot \sqrt{1+2^{-2(j+1)}} \cdot \sqrt{1+2^{-2(j+2)}} \cdots \\ &\quad \cdot \sqrt{1+2^{-2m}}. \end{aligned} \quad (18)$$

The effective angular rotation λ of the vector system will be the value of the algebraic summation of the individual rotations.

$$\lambda = \xi_1 \alpha_1 + \xi_2 \alpha_2 + \xi_3 \alpha_3 + \cdots + \xi_n \alpha_n \quad (19)$$

where

$$\alpha_i = \tan^{-1} 2^{-i} \quad (20)$$

and

$$\xi_i = +1 \text{ or } -1. \quad (21)$$

Therefore, although the magnitude of each individual rotation step is fixed, there now appears the possibility

that, by an appropriate choice for each ξ , the algebraic summation of all steps can be made to equal any desired angle.

The requirements for making this sequence of steps suitable for use with any angle as the basis of a computing technique are: 1) a value must be determined for each angle α_i so that for any angle θ from -180° to $+180^\circ$ there is at least one set of values for the ξ operators that will satisfy (19), and 2) these chosen values must permit the use of a simple technique for determining the value of each ξ to specify λ .

The following relationships are necessary and sufficient for a sequence of constants to meet these requirements:

$$180^\circ \leq \alpha_1 + \alpha_2 + \alpha_3 + \cdots + \alpha_n + \alpha_n \quad (22)$$

$$\alpha_i \leq \alpha_{i+1} + \alpha_{i+2} + \cdots + \alpha_n + \alpha_n. \quad (23)$$

The following sequence meets the requirements of (22), (23) and (11):

$$\text{First term: } \alpha_1 = 90^\circ \quad (24)$$

$$\text{Second term: } \alpha_2 = \tan^{-1} 2^{-0} = 45^\circ \quad (25)$$

$$\text{Third term: } \alpha_3 = \tan^{-1} 2^{-1} \approx 26.5^\circ \quad (26)$$

$$\text{General term: } \alpha_i = \tan^{-1} 2^{-(i-2)} \quad (i > 1). \quad (27)$$

Any angle can now be represented by the expression

$$\begin{aligned} \lambda &= \xi_1(90^\circ) + \xi_2 \tan^{-1} 2^{-0} + \xi_3 \tan^{-1} 2^{-1} + \cdots \\ &\quad + \xi_n \tan^{-1} 2^{-(n-2)}. \end{aligned} \quad (28)$$

The combination of values of the operators $\xi_1 \xi_2 \cdots \xi_n$ form a special binary code which is based on a system of Arc Tangent Radices and will be identified as the ATR code. The values of α selected for this computing technique will be called ATR (Arc Tangent Radix) constants.

In addition, only one more term is required in this ATR system than that required in a perfect binary-radix system for equivalent angular resolution;

$$\begin{aligned} (n-1)\text{th term of perfect binary system } (\pm \text{variable}) \\ = 2^{-n} \text{ revolutions} \end{aligned} \quad (29)$$

$$\begin{aligned} n\text{th term of ATR system (for large } n) \\ \approx \frac{2^{-(n-1)}}{\pi} \text{ revolutions} \end{aligned} \quad (30)$$

$$\frac{2^{-(n-1)}}{\pi} < 2^{-n}. \quad (31)$$

Note that all terms except the first are terms of the natural sequence $\tan^{-1} 2^{-j}$ ($j=0, 1, 2$, etc.) and may be instrumented as shown in Fig. 2.

The computation step corresponding to the most significant radix is simply

$$Y_2 = R_1 \sin(\theta_1 + \xi_1 90^\circ) = \xi_1 X_1 \quad (32)$$

$$X_2 = R_1 \cos(\theta_1 + \xi_1 90^\circ) = -\xi_1 Y_1 \quad (33)$$

where

$$R_2 = R_1. \quad (34)$$

This step is unique in that no magnitude change is introduced. It may be instrumented with the same circuitry required for all of the other steps by simply disabling the direct input to the adder-subtractor during this step.

The change in magnitude of the components resulting from the use of all of the terms in the series of (28) is the constant factor

$$\sqrt{1 + 2^{-0}} \cdot \sqrt{1 + 2^{-2}} \cdot \sqrt{1 + 2^{-4}} \cdots \cdot \sqrt{1 + 2^{-2(n-2)}}. \quad (35)$$

The value of this magnitude factor is a function of n which can be a constant for any given computer. By arbitrarily solving for the factor for $n=24$ and by denoting the value of the magnitude change factor as K ,

$$K = 1.646760255. \quad (36)$$

At this point, these individual steps can be fitted into a complete computing technique. Of the two basic algorithms that will be described here, the problem of "vectoring" will be considered first. Vectoring is the term given to the conversion from rectangular-coordinate components to polar coordinates, that is, given the Y and X components of a vector, the vector magnitude R and its angular argument θ are to be computed. In this technique, R and θ are computed simultaneously and in separate register locations.

First consider the problem of computing R . Except for the known magnitude change K , the Pythagorean relationship of the coordinate components is maintained regardless of the value of the summation of rotation angles, λ . Then, if the individual directions of rotations, ξ_i , can be controlled so that, after the end of the computing sequence, the Y component is zero and the X component is positive,

$$\begin{aligned} R_{n+1} &= \sqrt{X_{n+1}^2 + Y_{n+1}^2} = X_{n+1} = KR_1 \\ &= K\sqrt{X_1^2 + Y_1^2}. \end{aligned} \quad (37)$$

The technique for driving θ to zero is based on a numerical nulling sequence similar to nonrestoring division.

Since the vector R_i is described only in terms of its rectangular-coordinate components, the angle of this vector θ_i from the origin (positive X axis) is not known. However, if θ_i is considered to be expressed in a form so that

$$-180^\circ \leq \theta_i < 180^\circ, \quad (38)$$

then it can be shown that the sign of the Y_i component always corresponds to the sign of the angle θ_i .

Therefore, before each step of the computation, the sign of Y_i may be examined to determine which of the two possible values of ξ_i will drive θ_{i+1} opposite in direc-

tion from θ_i and then set the action of the adder-subtractor accordingly to obtain the relationship

$$|\theta_{i+1}| = \left| |\theta_i| - \alpha_i \right|. \quad (39)$$

Regardless of the choice for ξ_i , the shift gates and the register gates can be controlled so that the increments of rotation prescribed by the sequence of ATR constants are used in the same order (most significant first) as shown in (24)–(27).

It can be shown that, by adding another term α_n to the summation of all ATR constants, the summation is greater than or equal to 180° for any value of n :

$$180^\circ \leq \alpha_1 + \alpha_2 + \alpha_3 + \cdots + \alpha_n + \alpha_n. \quad (40)$$

By expressing the angle of any vector in the form given by (37), the following relationship is obtained:

$$|\theta_i| \leq \alpha_1 + \alpha_2 + \alpha_3 + \cdots + \alpha_n + \alpha_n. \quad (41)$$

Although it has been previously stated, without proof, it can be readily shown that, for the i th term,

$$\alpha_i \leq \alpha_{i+1} + \alpha_{i+2} + \cdots + \alpha_n + \alpha_n. \quad (42)$$

Therefore, if the same rules given for (39) are applied to determine θ_{i+1} ,

$$-\alpha_1 \leq |\theta_1| - \alpha_1 \leq \alpha_2 + \alpha_3 + \cdots + \alpha_n + \alpha_n. \quad (43)$$

Then, by applying the inequality of (42) to the left-hand term of the above equation,

$$|\theta_2| \equiv \left| |\theta_1| - \alpha_1 \right| \leq \alpha_2 + \alpha_3 + \cdots + \alpha_n + \alpha_n. \quad (44)$$

Likewise, this process may be continued through α_n to give

$$|\theta_{n+1}| \leq \alpha_n. \quad (45)$$

As an illustration of the step-by-step value of the vector, as described by the coordinate components at each step during the vectoring operation, consider the example in Fig. 3.

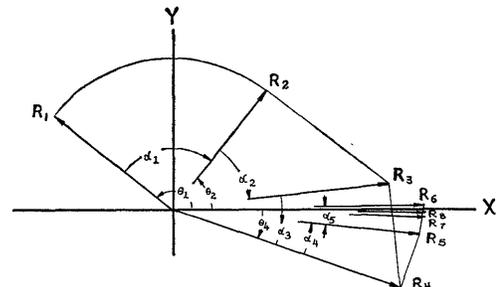


Fig. 3—Step-by-step relationships during nulling.

If, at the end of the computing sequence, the coordinate components specify a θ_{i+1} equal to zero, the total amount of rotation performed was equal in magnitude but opposite in sign to the angle θ_1 as specified by the original coordinate components Y_1 and X_1 .

At this point another register (identified as the angle register) and another adder-subtractor may be introduced, and it shall be assumed that the numerical value of each of the preselected ATR constants is stored within the computer and can be made available to the arithmetic unit in the same order as specified in (28). Since each ξ controls the action of the cross addition, each may also control the action of the additional adder-subtractor so that a subtraction or an addition may be made simultaneously in the angle register of the numerical value of the corresponding ATR constant to an accumulating sum to obtain the numerical value of the original angle θ_1 . Then, at the end of the computation, the desired numerical value of θ_1 is in this additional register, and the quantity KR_1 is in the X register.

A block diagram of the complete arithmetic unit necessary for computing both R_1 and θ is shown in Fig. 4.

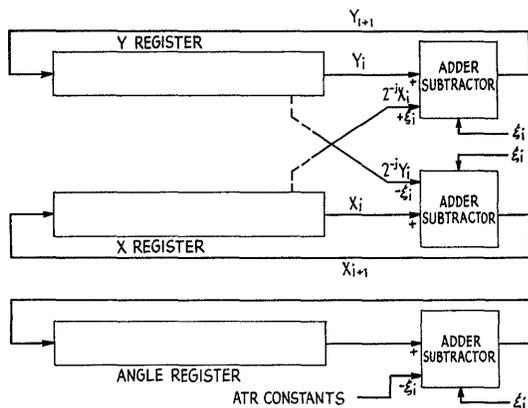


Fig. 4—Block diagram of complete arithmetic unit.

In summarizing the vectoring operations, the initial coordinate components Y_1 and X_1 of a vector are given, and the quantities R_{n+1} and θ_1 are computed where

$$R_{n+1} = KR_1 = K\sqrt{X_1^2 + Y_1^2} \quad (46)$$

and

$$\theta_1 = \tan^{-1} \frac{Y_1}{X_1} \quad (47)$$

and K is the constant magnitude-change factor described in (36) and (37).

A typical computing sequence is shown in Table I. The two's complement rotation is used for all quantities and for simplicity, shifted quantities are simply truncated without round-off.

Next, consider the application of the previous step-by-step relationships for the solution of the problem of coordinate rotation. It will help in applying this technique to consider, as before, the coordinate system as being fixed and the vector as being rotated. Then the solution for coordinate rotation requires the solution of the numerical values of the two coordinate components of a vector that has been rotated by a given

TABLE I
TYPICAL VECTOR COMPUTING SEQUENCE

Y Register	X Register	Angle Register
$Y_1 = 0.0101110$	$1.1000101 = X_1$	0.0000000
-1.1000101	$+0.0101110$	$+0.1000000 \quad \tan^{-1} \infty$
0.0111011	0.0101110	0.1000000
-0.0101110	$+0.0111011$	$+0.0100000 \quad \tan^{-1} 1$
0.0001101	0.1101001	0.1100000
-0.0110100	$+0.0000110$	$+0.0010010 \quad \tan^{-1} 2^{-1}$
1.1011001	0.1101111	0.1110010
$+0.0011011$	-1.1110110	$-0.0001001 \quad \tan^{-1} 2^{-2}$
1.1110100	0.1111001	0.1101001
$+0.0001111$	-1.1111110	$-0.0000101 \quad \tan^{-1} 2^{-3}$
0.0000011	0.1111011	0.1100100
-0.0000111	$+0.0000000$	$+0.0000010 \quad \tan^{-1} 2^{-4}$
1.1111100	0.1111011	0.1100110
$+0.0000011$	-1.1111111	$-0.0000001 \quad \tan^{-1} 2^{-5}$
1.1111111	$0.1111100 = KR_1$	$0.1100101 = \theta$

angle λ from its original position, as defined by the given initial coordinate components Y_1 and X_1 .

The desired angle of rotation will be given in binary coded form. By placing the angle λ in the angle register and sensing the sign of the quantity in this register before each step, the quantity in the angle register may be nulled to zero by sequentially subtracting or adding each of the ATR constants to the remaining quantity.

The explanation and proof of this nulling operation, in which the actual numerical values of the angles are employed, is the same as that for the previous nulling operation for vectoring.

Immediately following the determination of each ATR digit, and concurrently with the operation of the subtraction or addition nulling operation in the angle register, the operation of cross addition of shifted quantities may be performed in the Y and X registers to rotate the vector in a direction determined by each ξ with an angular magnitude as specified by α_i corresponding to the ATR constant being used. Then, at the end of the computing sequence, the numerical values of the desired components Y_{n+1} and X_{n+1} will be in the Y and X registers, respectively.

In summarizing the coordinate-rotation operation, the initial coordinates Y_i and X_i and the desired angle of rotation λ_1 are given where

$$Y_1 = R_1 \sin \theta_1 \quad (48)$$

$$X_1 = R_1 \cos \theta_1. \quad (49)$$

The results available at the end of the computation sequence are

$$Y_{n+1} = R_{n+1} \sin (\theta_1 + \lambda_1) = KR_1 \sin (\theta_1 + \lambda_1) \quad (50)$$

$$X_{n+1} = R_{n+1} \cos (\theta_1 + \lambda_1) = KR_1 \cos (\theta_1 + \lambda_1) \quad (51)$$

where K is the same constant magnitude-change factor, given in (36) and (37), that applies for the vectoring operation.

A typical coordinate rotation computing sequence is shown in Table II. (Note that in this computation, the vector is rotated through the negative X axis.)

Since the change of magnitude will be exactly known beforehand, it may be compensated for exactly either by scaling or by a magnitude-correction multiplication. It may then be said that, except for the practical consideration of limiting the number of digits and the number of steps to some finite value, both algorithms produce an exact solution. In a practical computer, no approximations are necessary except round-off.

In applying this computing technique to practical problems, the complete solution may be programmed by considering the computer as the digital equivalent of an analog resolver.

If the analog-resolver solution flow for the particular problem is known, then the number of operations and the information-flow diagram can be obtained simply by substituting for each resolver, on a time-shared basis, a CORDIC operation.

TABLE II
TYPICAL ROTATION COMPUTING SEQUENCE

Y Register	X Register	Angle Register
$Y_1 = \underline{0.0101110}$	$\underline{1.1000101} = X_1$	$0.1100101 = \lambda$
$\underline{+1.1000101}$	$\underline{-0.0101110}$	$\underline{-0.1000000} \quad \tan^{-1} \infty$
$\underline{1.1000101}$	$\underline{1.1010010}$	$\underline{0.0100101}$
$\underline{+1.1010010}$	$\underline{-1.1000101}$	$\underline{-0.0100000} \quad \tan^{-1} 1$
$\underline{1.0010111}$	$\underline{0.0001101}$	$\underline{0.0000101}$
$\underline{+0.0000110}$	$\underline{-1.1001011}$	$\underline{-0.0010010} \quad \tan^{-1} 2^{-1}$
$\underline{1.0011101}$	$\underline{0.1000010}$	$\underline{1.1110011}$
$\underline{-0.0010000}$	$\underline{+1.1100111}$	$\underline{+0.0001001} \quad \tan^{-1} 2^{-2}$
$\underline{1.0001101}$	$\underline{0.0101001}$	$\underline{1.1111100}$
$\underline{-0.0000101}$	$\underline{+1.1110001}$	$\underline{+0.0000101} \quad \tan^{-1} 2^{-3}$
$\underline{1.0001000}$	$\underline{0.0011010}$	$\underline{0.0000001}$
$\underline{+0.0000001}$	$\underline{-1.1111000}$	$\underline{-0.0000010} \quad \tan^{-1} 2^{-4}$
$\underline{1.0001001}$	$\underline{0.0100010}$	$\underline{1.1111111}$
$\underline{-0.0000001}$	$\underline{+1.1111100}$	$\underline{+0.0000001} \quad \tan^{-1} 2^{-5}$
$\underline{1.0001000}$	$\underline{0.0011110}$	$\underline{0.0000000}$

Monte Carlo Calculations in Statistical Mechanics

W. W. WOOD† AND J. D. JACOBSON†

I. STATISTICAL MECHANICAL INTRODUCTION

ACCORDING to classical statistical mechanics, the thermodynamic properties of a system of N molecules at temperature T and volume V are obtainable from the Gibbs configurational phase integral,

$$Z_N(T, V) = \int_V \dots \int_V e^{-U/kT} d\mathbf{r}_1 \dots d\mathbf{r}_N, \quad (1)$$

where k is Boltzmann's constant; U is the potential energy of the system of N molecules, and is a function of the position vectors \mathbf{r}_α , $\alpha = 1, 2, \dots, N$, of the molecules. For suitably simple molecules one usually assumes U to be expressible as a sum of spherically symmetric pair interactions $u(r)$:

$$U(\mathbf{r}_1, \dots, \mathbf{r}_N) = \frac{1}{2} \sum_{\alpha=1}^N \sum'_{\beta=1}^N u(r_{\alpha\beta}), \quad (2)$$

where $r_{\alpha\beta} = \|\mathbf{r}_\beta - \mathbf{r}_\alpha\|$, and the prime indicates omission of terms for which $\alpha = \beta$.

Most of the thermodynamic functions are expressible in terms of "ensemble averages" of some related function of the configurational coordinates. In these averages the factor $e^{-U/kT}$ appears as a weighting factor, so that Z_N

given by (1) is the associated normalizing factor. For example, the pressure p is given by the average of the "virial" V_R of the total intermolecular force:

$$pV/NkT = 1 - (1/3NkT)\langle V_R \rangle, \quad (3)$$

where

$$\langle V_R \rangle = (1/Z_N) \int_V \dots \int_V \left(1/2 \sum_{\alpha} \sum'_{\beta} r_{\alpha\beta} du(r_{\alpha\beta})/dr_{\alpha\beta} \right) \times e^{-U/kT} d\mathbf{r}_1 \dots d\mathbf{r}_N \quad (4)$$

The "radial distribution function" $g(r)$ is of considerable importance in the study of fluids. Let us first define $n(r)$, the "cumulative radial distribution function," giving the average number of molecules lying within the distance r from any representative molecule:

$$n(r) = (1/N) \left\langle \sum_{\alpha} \sum'_{\beta} A(r_{\alpha\beta}, r) \right\rangle, \quad (5)$$

where $A(x, r)$ is the step function

$$A(x, r) = \begin{cases} 1 & 0 \leq x < r \\ 0 & r \leq x \end{cases}. \quad (6)$$

Then $g(r)$, which is the number density of molecules at the distance r from any representative molecule, relative to the over-all macroscopic density N/V , is given by

† Los Alamos Scientific Lab., Los Alamos, N. M.

$$g(r) = (V/4\pi Nr^2)dn(r)/dr. \quad (7)$$

In terms of $g(r)$ (3) can be written

$$pV/NkT = 1 - \frac{2\pi}{3} \frac{N}{V} \frac{1}{kT} \int_0^\infty ru'(r)g(r)r^2dr. \quad (8)$$

Although more involved potential functions $u(r)$ are required to predict the thermodynamic properties of real molecular systems [1], there is much theoretical interest in the simple hard sphere (sometimes called billiard ball, or elastic sphere) model for which

$$u(r) = \begin{cases} \infty & r < \sigma, \\ 0 & r \geq \sigma; \end{cases} \quad (9)$$

σ is the molecular diameter. This form of $u(r)$ also leads to considerable computational simplification in the Monte Carlo procedure, and we shall refer most of our discussion to this case. Eq. (8) then becomes

$$\begin{aligned} pV/NkT &= 1 + \frac{2\pi}{3} \frac{N}{V} \sigma^3 g(\sigma), \\ &= 1 + \sigma n'(\sigma)/6; \end{aligned} \quad (10)$$

the pressure depends only on the value of the radial distribution function at the surface of the molecule.

II. THEORETICAL BASIS OF THE MONTE CARLO METHOD

For a macroscopic system, $V \sim 1 \text{ cm}^3$, N is of the order of 10^{22} , which is a considerable obstacle to either analytical or numerical treatment. Just as the various theories in one way or another radically reduce the dimensionality of the problem, a similar reduction is necessary before undertaking numerical work. Here this is done by applying the *periodic boundary condition* to a basic system of perhaps a few hundred molecules. The basic volume V is taken to have a shape such that it fills space compactly under successive translation along three axes in space. There are N molecules in each replica V , in the same relative positions, and in evaluating U from (2) the sum over α is limited to the N molecules in a particular reference V , but the sum over β is allowed to extend over the adjacent replicas out to some appropriately determined cutoff point. Particularly at high densities, near close-packed, it is important that N be chosen so that a regular lattice of N points fixed in V generates under the above translations the corresponding regular lattice throughout space. Most of our work has used a cubical V for which the natural choice of lattice is cubic-close-packed (face-centered cubic); then the possible choices of N are $4n^3$, $n=1, 2, \dots$, or $N=4, 32, 108, 256, 500, \dots$. $N=32$ is the first reasonably realistic choice, and most of our calculations have been done for this value. The periodic boundary condition is illustrated in Fig. 1 for a two-dimensional square V containing four molecules.

For hard spheres pV/NkT can readily be shown to be a function only of the reduced volume parameter τ ,

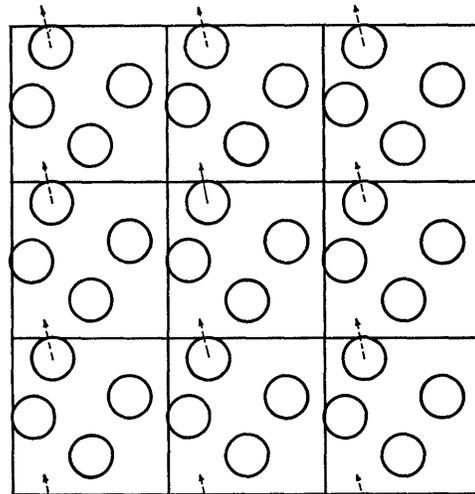


Fig. 1—The periodic boundary condition in two dimensions with four molecules in a square V .

defined by

$$\tau = \frac{N}{V\sigma^3\sqrt{2}}. \quad (11)$$

(The parameter so defined is the ratio of the volume V actually occupied by the system to the volume occupied at regular close packing.) For finite values of N , with the periodic boundary condition, some additional dependence upon N , for fixed τ , may be expected, and one object of the investigation should be to estimate this dependence and perhaps extrapolate to $N = \infty$, since it is this limiting value which will correspond to the desired thermodynamic value.

By use of the periodic boundary condition the problem has been approximated by one of a few hundred degrees of freedom, which is in the region where analytical methods fail but where Monte Carlo methods on electronic calculators may be expected to be feasible. However, the most obvious Monte Carlo procedure, the evaluation of Z_N as given by (1) by means of independent sampling of points in $3N$ dimensional configuration space, proves not to be feasible at interesting values of V/N [2]. The method to be described here is due to Metropolis, *et al.* [3], and involves instead a Markov chain whose states are points in $3N$ dimensional configuration space. A Markov chain may be regarded as the next level of stochastic complexity beyond the simple series of independent events such as a series of throws of a die, or the usual Monte Carlo integration procedure just mentioned. In contrast to the latter, in a Markov chain the probability of obtaining a given result at the t th step depends explicitly on the result on the $t-1$ th step (and therefore implicitly on all preceding steps). The simplest example is the well-known random walk, and the present method is in fact a random walk in $3N$ dimensional configuration space. The Markov chain method usually does not result in an evaluation of Z_N (1), but instead permits the direct evaluation of con-

figurational averages such as appear in (4) and (5).¹ The Markov chain is so constructed that the average of any function of configuration over all states in the chain tends toward the corresponding ensemble average [e.g., (4)] as the chain is developed indefinitely.

The discussion of Markov chains in which the admissible states form a discrete, finite set is somewhat simpler than that required for the case of a continuum of states, and we shall therefore take advantage of the fact that the use of digital calculation itself imposes a "quantization" of configuration space determined by the number of binary bits (in our calculations, 17) used to specify the molecular coordinates. Let us then represent possible configuration states by the letters j , k , and l , and let $p(j, k)$ be the probability, if at time t the configuration state is j , that at time $t+1$ the state will be k . To describe the development of the chain we use the usual "time" terminology; it should be emphasized that this is not a time directly related to molecular times, but may conveniently be thought of as corresponding to calculating machine time. The $p(j, k)$ are independent of t , and for the desired averaging property must satisfy the following conditions [1]:

$$\sum_k p(j, k) = 1, \quad (12)$$

and

$$p(j, k)e^{-U(j)/kT} = p(k, j)e^{-U(k)/kT}. \quad (12a)$$

In addition, there is an ergodic condition on the matrix $p(j, k)$: all states included (with nonzero weight) in the integral in (4) must belong to a single ergodic class [5]. That is, for every pair of such states j, k there must exist at least one finite n such that $p^{(n)}(j, k) \neq 0$; $p^{(n)}(j, k)$ is the probability of passing from state j at time t to state k at time $t+n$:

$$p^{(1)}(j, k) = p(j, k); p^{(n)}(j, k) = \sum_l p^{(n-1)}(j, l)p(l, k). \quad (13)$$

Among the $p(j, k)$ matrices satisfying (12) and (12a) we use the following:

$$\left. \begin{aligned} p(j, k) &= A(j, k) && \text{if } U(k) \leq U(j); \\ &= A(j, k) \exp[-(U(k) - U(j))/kT], && \text{if } k \neq j; \\ & && \text{if } U(k) > U(j); \end{aligned} \right\} (14)$$

$$p(j, j) = 1 - \sum_k p(j, k), \quad (15)$$

where

$$\begin{aligned} A(j, k) &= 1/8N\delta^3, \text{ if } r(\alpha; k) = r(\alpha; j), \\ &\alpha = 1, 2 \cdots \beta - 1, \beta + 1 \cdots N; \\ &\text{and } r(\beta, k) \in C_\delta[r(\beta; j)], \beta = 1, 2, \cdots N; \\ &= 0, \text{ otherwise.} \end{aligned} \quad (16)$$

In (16) $C_\delta[r(\beta, j)]$ denotes a cube of half edge δ , oriented parallel to the coordinate planes, with center at $r(\beta, j)$; the latter notation means the position vector of molecule β in configuration j . The length δ is a parameter of the Monte Carlo process whose value is found to influence the rate of convergence of the process. It is ordinarily small (see Section IV) compared to the edge of the cube V , and in (16) it is expressed in units of the smallest increment in the molecular coordinates; in our calculations, therefore, in units of 2^{-17} times the edge of the cube V .

The motivation underlying this specification of $p(j, k)$ is as follows. The factor $A(j, k)$ corresponds to a uniform choice among the limited number of neighboring states contained in C_δ ; this limitation is related to the small mean free path in molecular systems at interesting densities. The unsymmetrical form of the Boltzmann weighting factor (14) is chosen in preference to the alternative symmetrical form

$$p(j, k) = A(j, k) \frac{e^{-U(k)/kT}}{e^{-U(j)/kT} + e^{-U(k)/kT}} \quad (17)$$

mostly for computational convenience, since it leads to fewer calculations of the exponential function (in the case of more complicated models than hard spheres), but also because the unsymmetrical form leads to a more rapid motion in configuration space.

For hard spheres (14) reduces to

$$\begin{aligned} p(j, k) &= A(j, k), && \text{if } U(k) = 0, \\ &= 0, && \text{if } U(k) = \infty, \end{aligned} \quad (18)$$

assuming $U(j) = 0$, the only case with which we will be concerned since the initial state of the chain has $U = 0$.

It can be easily shown that if $u(r)$ has only point infinities (say, at $r=0$), then $p(j, k)$ defined above also satisfies the ergodic condition. However, the hard sphere $u(r)$ is infinite over a finite interval in r , with the result that U in $3N$ dimensional space is infinite over regions of considerable extent, from which the state point representing our system is excluded. The topological question of whether the accessible region in which U is zero is then connected or not, depending on the values of N/V and σ , is one to which the answer is unknown, and as a result the equivalence of the Markov and "ensemble" averages is somewhat doubtful. Under such circumstances the $p(j, k)$ defined above may separate the configuration states into separate ergodic classes. Within each class (*i.e.*, depending on the choice of initial configuration) the Markov average will converge to an average equivalent to the ensemble average taken over a restricted region of configuration space. However, it should also be mentioned that the $p(j, k)$ defined by (14) and (15) may actually connect spatially disconnected regions of configuration space, if the thickness of the dividing barrier is not too large compared to the parameter δ . (This is the only circumstance in which the value of δ will affect the convergent average of the Markov chain.)

¹ For a case in which the Monte Carlo method leads to an evaluation of Z_N , see Salsburg, *et al.* [4].

Such a compartmentalization of configuration space also raises the well-known quasi-ergodic problem in statistical mechanics concerning the equivalence of "ensemble" averages to the time-average behavior of an actual physical system of N molecules moving according to the Newtonian equations of motion for the given potential function. In this connection we call attention to the work of Alder and Wainwright [6], in which these equations of motion for the same systems of hard spheres with periodic boundary conditions have been integrated. Thus it is possible to compare the results of this kinetic theory calculation with the present statistical calculation. In a later part of the paper some comparisons of this sort will be made.

As mentioned already, the topological question of the connectivity of configuration space has not been solved. It is quite evident, for example, that at sufficiently high densities, near close packed, it will not be possible to interchange the positions of two molecules without crossing a region in which U is infinite. Configuration space is then evidently compartmentalized into something like $N!$ disconnected regions. However, for systems of N identical molecules (*i.e.*, a one-component system) this compartmentalization is of no consequence since all ensemble averages have identical values in all such compartments (*i.e.*, for hard spheres, the volumes of the compartments are all the same). It is the question of whether, associated with this trivial partitioning of the configuration space, there is a further, nontrivial partitioning into regions of different properties, which is uncertain. In particular, for example, if such a nontrivial compartmentalization does occur, how does it depend on the choice of N , at constant N/V and σ ? Is it an artifact of the periodic boundary condition? We will have occasion to return to these questions when we describe below the results obtained with the method.

III. CALCULATIONAL PROCEDURES

It is convenient to take as unit of distance the edge of the cubical volume V , so that in these units $V=1$. For a given choice of N , specification of τ then determines the value of σ , by (11). We first outline briefly, then discuss in greater detail, the steps by which the calculator is caused to develop a particular sample chain according to the Markov process defined by (14)–(16), for a system of hard spheres.

1) Assume that at "time" t the system is in the state $j(t)$, corresponding to subscript j in (14)–(16).

2) A random choice of one of the N molecules is made, corresponding to subscript β in (16). We call this molecule $\beta(t)$.

3) Each coordinate of molecule $\beta(t)$ is given a tentative random displacement, uniform on the interval $(-\delta, \delta)$, corresponding to $r(\beta, k)$ in (16). Call the configuration in which molecule $\beta(t)$ is in this displaced position, and the other $N-1$ molecules α have position $r(\alpha, j(t))$ configuration $j'(t)$.

4) Configuration $j'(t)$ is tested for overlaps; *i.e.*, one or more pairs of molecules whose distance between centers is less than the molecular diameter σ .

5) a) If no overlap is found in Step 4, then the next configuration in the chain is $j'(t)$; *i.e.*, $j(t+1)=j'(t)$.

b) If configuration $j'(t)$ contains an overlap, the configuration at time $t+1$ is identical with that at time t : $j(t+1)=j(t)$.

6) Certain procedures concerned with the averaging process are performed; the procedure then repeats beginning at Step 1, except for occasional interruptions for checking and census procedures. We now describe each of these steps in greater detail.

A. Specification of a Configuration

Here we describe the information carried in the calculator memory for purposes of specifying a particular configuration. Fundamentally all that is required are the values of the $3N$ coordinates $r(\alpha)$, which are stored in the r -Table. For economy of calculating time, however, it is desirable to carry along additional redundant information, since the calculation requires a rather large amount of machine time. For example, in connection with the calculation of the "cumulative radial distribution function" $n(r)$ (5), it is desirable to have available for each configuration $j(t)$ a tally, called the C -Table, of the intermolecular distances (squared) $r_{\alpha\beta}^2$ into the N_R intervals $C(\nu) = [\sigma^2 + \nu \cdot 2^{-s}, \sigma^2 + (\nu+1)2^{-s}]$, $\nu=0, 1, \dots, N_R-1$. Here s is chosen to be an integer in order to expedite the tallying process. When we are interested only in estimating the pressure, using (10), so that only $g(\sigma)$ is required, the range of the C -Table can be made quite small, with a consequent economy of machine time, as will appear below.

The search for overlaps in Step 4 can be considerably expedited by a device which avoids the necessity of examining all $N-1$ values of $r_{\alpha\beta}^2$.² The basis for the device is the fact that the displacement parameter δ is in all interesting cases relatively small compared to unity (edge of V), so that any actual overlap $r_{\alpha\beta}$ which occurs must be with a molecule α which was fairly close to β in configuration $j(t)$. We proceed as follows. At some point t_0 in the chain (in particular, at its beginning $t=0$) we establish a set of tables called the $\mu(\alpha)$ -Tables, one for each $\alpha=1, 2, \dots, N$. Each $\mu(\alpha)$ -Table consists of an indefinite number of entries $\mu(\alpha, \gamma)$, $\gamma=1, 2, \dots$, and gives all molecule numbers $\mu(\alpha, \gamma)$ for which $r_{\alpha\mu}^2(\alpha, \gamma) < d_4^2$ in configuration $j(t_0)$. The parameter d_4 must satisfy the relation

$$d_4^2 > d_3^2 = \sigma^2 + 2^{-s}N_R, \quad (19)$$

but is otherwise arbitrary, and can be adjusted for optimum calculation rate. Also at t_0 the positions of all N

² Note that since configurations $j'(t)$ and $j(t)$ differ only in the coordinates of molecule β , only these $N-1$ distances have changed. Thus if $j(t)$ is free of overlaps (which will be the case unless an error has occurred at some previous value of t), any overlap in $j'(t)$ will occur among this set of $N-1$ interactions.

molecules are recorded in the r_0 -Table; we shall call the configuration specified by this r_0 -Table "configuration j_0 ." A simple application of the triangle inequality shows that as long as all molecules remain within a distance

$$d = (d_4 - d_3)/2 \quad (20)$$

of their r_0 -Table positions, only those molecules in the $\mu(\alpha)$ -Table can approach closer than d_3 to molecule α . Thus as long as this "diffusion condition" is satisfied we need test in Step 4 only the molecules contained in the $\mu[\beta(t)]$ -Table.³

The recent availability of considerably larger core memories on the Los Alamos 704 calculators has made it possible to consider carrying other redundant information in order to increase further the calculation rate. Some of these possibilities are mentioned later in the discussion.

The starting configuration most often used is a regular face-centered lattice arrangement.

B. Choosing Molecule $\beta(t)$

This is done by the usual technique of taking the integer part of the product ξN , where ξ is a nominally randomly, uniformly distributed variable on the interval $[0, 1]$, obtained from a sequence of pseudo-random numbers generated by the middle square process [7]. We use a sequence of 70 bit numbers (generated by double precision techniques) which has not been statistically analyzed except to verify that there are no repetitions among the first 10^6 70 bit numbers. Faster techniques [8] can be used to generate pseudo-random numbers, and are currently being incorporated in the codes. The generation of all the random numbers required in the complete calculation uses at most about 7 per cent of the calculation time, however, so that no large improvement can be made by this change alone.

C. Displacing Molecule $\beta(t)$

The procedure is similar to that in Step 2, the displaced position being given by

$$r[\beta(t); j'(t)] = r[\beta(t); j(t)] + \delta\xi \quad (21)$$

where the three components of ξ are independent random numbers uniform on the interval $(-1, 1)$. We may mention that as a consequence of the periodic boundary condition, if molecule $\beta(t)$ due to the displacement given by (21) leaves the volume V through one face, it re-enters V through the opposite face; thus the

³ Actually, one might set up two $\mu(\alpha)$ -tables, $\mu_1(\alpha)$ giving interactions which are potential overlaps with α [entries being determined by $d_4' = \sigma + 2d$ (22)]; $\mu_2(\alpha)$ defined as in the text, giving potential C -Table interactions. This might be worthwhile when d_3 is relatively large compared to σ [for example, if $g(r)$ is to be calculated over a range of several molecular diameters]. In the present investigations we have concentrated on determining the pressure p ; as we have seen since only $g(\sigma)$ must then be determined, d_3 may be quite close to σ (we use $d_3 = 1.05 \sigma$), and not much advantage would be gained from the additional complication.

number of molecules N contained in V remains constant (see Fig. 1).

Steps 2 and 3 together require about 19 per cent of the average calculating time per configuration, with the present code.

D. Testing $j'(t)$ for Overlaps, and Tallying its Interactions into the C -Table

The distances⁴ $\|r[\beta(t); j'(t)] - r[\mu(\beta(t), \gamma); j(t)]\|$ are computed for $\gamma = 1, 2, \dots$; if for any γ an overlap is found, control passes immediately to Step E2. Those distances less than d_3 are tallied into an E -Table whose structure is identical with that of the C -Table already described. If none of the entries in $\mu[\beta(t)]$ produce an overlap, the diffusion test is made to see whether $r[\beta(t), j'(t)]$ is within the distance d of the r_0 -Table position stored for molecule β . If it is, control passes immediately to Step E1. If the diffusion condition is violated, control enters a routine called "refresh" which has the following functions:

1) All $N-1$ interactions $\|r[\beta(t); j'(t)] - r[\alpha; j_0]\|$ for $\alpha \neq \beta(t)$ are calculated, and the values of α for which these distances are less than d_4 are entered in a temporary W_2 -Table, which may eventually replace the current $\mu[\beta(t)]$ -Table.

2) As the procedure described in 1) is carried out, the distance $\|r[\beta(t); j'(t)] - r[\alpha; j(t)]\|$ is also calculated for every value of α which is placed in the W_2 -Table but which is not found in $\mu[\beta(t)]$. If any of these interactions in configuration $j'(t)$ is found to be an overlap, control passes immediately to Step E1. Those which lie in the interval (σ, d_3) are tallied into the E -Table already mentioned.

3) Those values of α which are entered in the W_2 -Table, but which are not contained in $\mu[\beta(t)]$ are tabulated, and if no overlap is found in $j'(t)$, the entry $\beta(t)$ is inserted in $\mu(\alpha)$ for these values of α .

4) Those values of α which are *not* entered in the W_2 -Table, and which appear in $\mu[\beta(t)]$, are tabulated, and if no overlap is found in $j'(t)$, the entry $\beta(t)$ is removed from $\mu(\alpha)$ for these values of α .

5) If no overlap is found in $j'(t)$, $r[\beta(t); j'(t)]$ replaces $r[\beta(t); j_0]$, and control passes to Step E1.

The testing of $j'(t)$ for overlaps, exclusive of the time spent in the "refresh" routine, is the most time-consuming part of the calculation, using about 46 per cent of the total time; the "refresh" routine uses about 16 per cent of the time. These figures are for values of the parameters controlling the calculation rate which seem to be optimum (see below).

E1) No Overlap in $j'(t)$

In this case, the next configuration of the chain, $j(t+1)$, is configuration $j'(t)$. The necessary modifica-

⁴ When we speak of computing a distance between two molecules we always mean the computation of the square of the distance; no square root procedures are required.

tion of the r_0 -Table and of the μ -Tables has been described under Step 4. The only further complication appears in the modification of the C -Table. Evidently what is required is to remove the interactions $\|r[\beta(t); j(t)] - r[\mu(\beta(t), \gamma); j(t)]\|$ which have been tallied into the C -Table at earlier t , and replace them by the interactions tallied into the E -Table during Step 4. The present code requires the recomputation of these $j(t)$ interactions at this point; current improvements will add to the redundant information already carried a set of $\phi(\alpha)$ -Tables recording for each α - $\mu(\alpha, \gamma)$ interaction in $j(t)$ the corresponding interval in the C -Table. In any case, the procedure consists of a tally of these $j(t)$ interactions into the E -Table, but with a negative unit increment; at the conclusion of this procedure the E -Table then contains the desired changes in the C -Table, so that the next procedure is $C+E \rightarrow C$. The modification of the r -Table is simply $r[\beta(t); j'(t)] \rightarrow r[\beta(t); j(t)]$. If the "refresh" procedure for modifying the μ -Tables was required in Step 4, we now replace $\mu[\beta(t)]$ by the W_2 -Table described there. This delay is necessary since $\mu[\beta(t)]$ must be used for the computation of $j(t)$ interactions.

We have so far not established a good criterion of the rapidity of the convergence process, other than simply a visual inspection of the trend of the results. In order to obtain some quantitative criterion for the choice of the parameter δ , we calculate the sum of the squares of the distances in $3N$ dimensional space between successive configuration points:

$$\phi(t) = \sum_{t'=1}^t \|r(\beta(t'); j(t')) - r(\beta(t'), j(t' - 1))\|^2. \quad (22)$$

At present about 7 per cent of the calculation time is spent in Step E1.

E2) Overlap in $j'(t)$

In this case $j(t+1) = j(t)$, no changes in the tables are required, and there is no contribution to $\phi(t)$.

F. Census and Checking Procedures

For convenience in monitoring the progress of the calculation, averages of the C -Table are taken over successive intervals of fixed length in time t , as well as over the entire length of the chain. For the former purpose an A -Table is carried which is set to zero at the beginning of each time interval, and at successive time steps is incremented by the C -Table. For the over-all average an S -Table is carried which is augmented by the A -Table at the end of each time interval. The cumulative distribution function $n(r)$, (5), is then estimated by

$$n(\sigma^2 + v \cdot 2^{-s}) = \frac{1}{Nt} \sum_{v'=0}^v S(v'; t). \quad (23)$$

Numerical differentiation (performed later by hand) then gives $g(\sigma)$, and the pressure p .

Numerous checks are built into the problem, particularly in the "refresh" routine in Step 4, and at the census intervals just described, where the current C -Table is checked by calculating all $N(N-1)/2$ interactions. These procedures have proven to be very worthwhile in giving notice of the occurrence of machine errors.

About 12 per cent of the calculation time is involved in these procedures.

IV. RESULTS AND DISCUSSION

The Monte Carlo method was previously applied to the system of hard spheres by Rosenbluth and Rosenbluth [9]. However, the results obtained by Alder and Wainwright [6] at an early point in their kinetic investigation suggested that the Rosenbluth results might be partly in error, so that we undertook a concurrent re-investigation of the question. The new results shown in Fig. 2 are indeed in rather good agreement with those of Alder and Wainwright.

The new Monte Carlo results have been described briefly elsewhere [9] from the standpoint of their statistical mechanical interest. Here we shall consider them briefly from the standpoint of computational interest, concentrating on aspects of the behavior which are in some sense exceptional.

Plotted as the reduced pressure pv_0/kT vs the reduced volume τ , where v_0 is the close-packed volume per molecule ($v_0 = \sigma^3/\sqrt{2}$), the results fall on two distinct curves, as shown in Fig. 2. This behavior may well be related to the presence of a first-order phase transition; we leave aside this point in the present discussion. In the region $\tau < 1.6$ the figure shows double-valued pressures. In the region $\tau = 1.52-1.60$ these arise because of the behavior of a representative single chain shown in Fig. 3. These chains show in addition to the usual statistical fluctuation, a tendency to oscillate between two rather well-defined classes of states, with interclass transitions being so rare (about four hours of 704 time were required to generate the chain pictured in Fig. 3) as to make the over-all averages very poorly determined within a single chain, and differing widely in independent chains generated for the same values of N and τ . If the two classes are averaged separately, however, the inter- and intrachain agreement is quite good; it is these separate averages which are plotted in Fig. 2, and which give rise to the double-valuedness. It is noteworthy that Alder and Wainwright observe the same rare interclass transitions.

In the course of investigating the effect of varying the parameter δ , we generated 14 different Markov chains for the 32-molecule system at $\tau = 1.55$, the chain shown in Fig. 3 being one of these. These chains comprised a total of 12,800,000 configurations, the longest chain in the set containing 3,500,000 configurations and the shortest, 560,000 configurations. A total of 14 well-defined interclass transitions were observed; four chains produced no transitions, eight produced one transition,

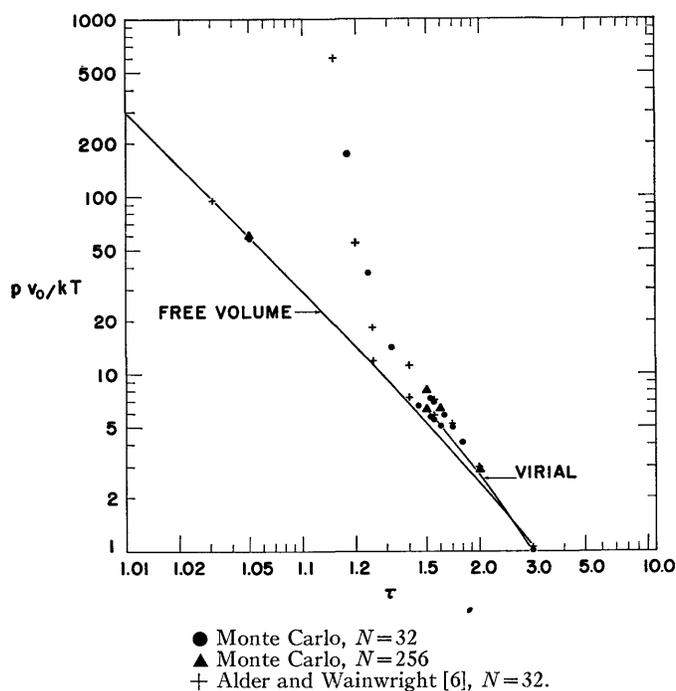


Fig. 2—For the sake of clarity many of the calculated points have been omitted from the figure, particularly those of Alder and Wainwright for $N=108$ and 256 .

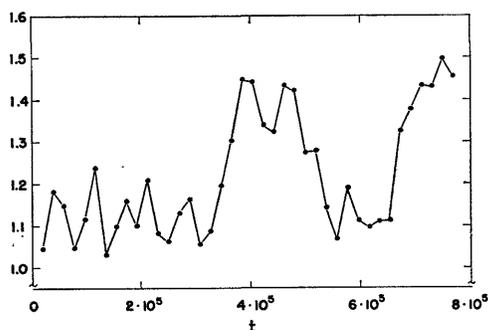


Fig. 3—The number of molecules inside a sphere of radius 1.026σ around a representative molecule, averaged over intervals of 19,200 configurations, plotted against the total number of configurations t , in a representative chain at $\tau=1.55$, $N=32$, $\delta=0.051$. The over-all average of these points should converge to the cumulative distribution function $n(1.026 \sigma)$, but the convergence is evidently very slow due to the secular fluctuation between the groups of high- and low-pressure states.

and two chains showed three transitions. Grouped all together, the results suggest that a transition occurs on the average of about once in 900,000 configurations. These 14 chains gave values of $p v_0 / kT$ (averaged over the entire length of each chain, not the separate high- and low-class averages mentioned previously) ranging from 5.45 to 6.92, with the average over all 14 chains being 6.23. This value of the average is obtained for two different weightings: 1) each configuration of every chain is given equal weight; or 2) each chain is weighted with its final value of $\phi(t)$ (22). The calculated standard deviation of this average is 0.11. We do not wish to stress the significance of this average, though it is interesting that it lies midway between the two curves of Fig. 2.

We interpret this transition behavior to be due to a compartmentalization of phase space (see the discussion of this point in Section II) into two regions which are narrowly connected in the region $1.52 < \tau < 1.60$. At larger values of τ the distinction between the two regions disappears rather quickly, and the over-all averages are easily determined. At smaller values of τ neither we nor Alder and Wainwright have observed any interclass transitions, and it is not known whether the two regions of configuration space corresponding to the two branches of Fig. 2 are in fact connected or not for $\tau < 1.52$: the lower curve is obtained when the initial configuration is taken as the regular lattice arrangement; the upper curve was obtained from initial configurations generated by a step-wise compression of the system (*i.e.*, gradually increasing σ as t increases), starting from a randomly chosen configuration from the upper class in the region $\tau = 1.55$ – 1.60 . The behavior does not change as N is varied, and is therefore probably not an artifact of the periodic boundary condition.

For optimum calculation time it has been found best to choose $d_4/\sigma \cong 1.3$ for $N=32$, $\cong 1.45$ for $N=256$, both for $\tau \sim 1.5$. The choice of this parameter does not affect the sequence of configurations generated in the chain, but only the rate (in terms of machine time) at which they are generated. The parameter δ , on the other hand, affects both the calculation rate and the motion through configuration space. The dependence of the optimization parameter $[\phi(t)/t]^{1/2}/\theta$ on the choice of δ is shown in Fig. 4, for the 32-molecule system at $\tau=1.55$. Here θ is the average actual machine (IBM 704) time required for the generation of one configuration in the chain, and $[\phi(t)/t]^{1/2}$ (taken for large t) is the root mean square displacement per configuration. Thus the maximum in Fig. 4 corresponds to a maximum movement through configuration space per unit of machine time, on the basis of the root-mean-square displacement criterion; it occurs at a value of $\theta \cong 20$ milliseconds/configuration. Also shown in Fig. 4 is the behavior of the "reject ratio" R , which is the fraction of recurring configurations due to overlaps in $j'(t)$. It is rather striking that the apparent optimum calculation rate occurs at a rather large value of R (~ 0.90). So large an optimum value of R was something of a surprise to us, but is at least partially supported by our qualitative impressions of the convergence of the radial distribution function, and probably also by the frequency of interclass transitions, which appears to be greatest at similarly large values of R .

It is of some interest to inquire how the rate at which the 704 calculator moves the state point through configuration space compares with that at which an actual molecular system executes its thermal movement, even though the two processes are of radically different character. If we take the temperature to be 25°C , the molecular diameter $\sigma = 3.5 \times 10^{-8}$ cm and the molecular mass as 40 atomic weight units (values which are approximately correct for the argon molecule), we find that a

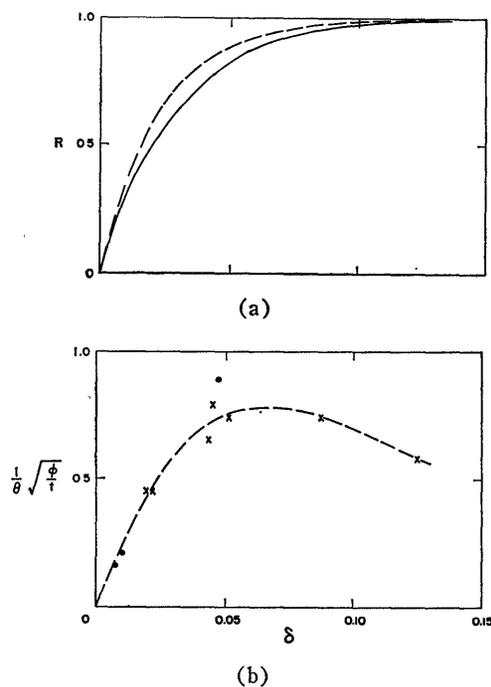


Fig. 4—The reject ratio R and calculation rate $\sqrt{\phi/t}/\theta$ (in units of the linear edge of the Monte Carlo cell V , per second of IBM 704 calculator time) as functions of δ , for $N=32$ and $\tau=1.55$. In (a) the dashed curve is from the high pressure class of states, the solid curve from the lower pressure class. Similarly in (b): ● lower pressure class; × high pressure class.

system of 32 molecules under these conditions would have a rate of root-mean-square displacement about 3×10^{12} times the maximum in Fig. 4(b). Thus we should not be surprised at the rareness of the interclass transitions during the machine calculation—on the scale of physical time they are relatively frequent, perhaps several hundred per microsecond. For the same values of the parameters, we can also compare directly the physical collision rate, which is estimated as 4×10^{14} collisions per second in a system of 32 molecules, with Alder and Wainwright's [6] calculation rate, which is about two collisions per second. Thus we see that in either case, the calculational procedures are much slower than the actual molecular system. One should not conclude from these figures that the Monte Carlo procedure is more efficient than the Alder and Wainwright procedure, since the root-mean-square displacement criterion is a very crude basis for comparison. Empirically, in terms of machine time, they seem to produce the interclass transitions at roughly the same rate.

The most important task remaining is to establish the significance of the two branches of the p - V curve at $\tau < 1.5$, *i.e.*, the connectivity and relative volumes of the corresponding regions of configuration space. Aside from possibly developing some more complicated stochastic approach, the chief hope of further progress lies in increasing the calculation speed. We have two approaches in mind, one with respect to the method, the other involving the use of faster machines.

A possible improvement in method lies in the question of how fine a subdivision of configuration space is required in order for a digital calculation to give results practically equivalent to the continuum of states implied by the analytical integral in (1). As already mentioned, we currently use $(2^{17})^3 = 2^{51}$ points in $3N$ dimensional space. If the number of bits required to represent the molecular coordinates can be reduced from the present 17 to perhaps 13–14 or less, then in the computation of the intermolecular distances we can replace arithmetic multiplication by a faster table-look-up process. With this and other similar modifications we hope to gain a considerable factor, but certainly less than 10, of improvement in speed.

The "Stretch" calculator currently under contract with IBM, for which coding is now in progress, will increase the calculation rate by a factor of 20–50, and we hope that the combination of improved programming and faster calculator will make possible a determination of the over-all average in the region near $\tau=1.55$. In another direction in computer design, it is probably worthwhile to mention that this problem falls in the class of those for which parallel computation by several arithmetic units with common memory and control could result in a considerable increase in calculating rate.

We shall close with a brief mention of some applications in which the results have been interesting without being confusing. For the hard sphere system at $\tau > 1.6$, where the chains are well convergent, the results are believed to be essentially exact and have been of considerable utility in calibrating various theoretical approximations leading to analytical treatments. At sufficiently low densities the method gives results in agreement with the "virial expansion" whose first five coefficients are known [9]. On the other hand, at sufficiently high densities ($\tau < 1.3$) the lower of the two pressures calculated from the two separate classes agrees increasingly well with the "free-volume" approximation. This suggests that the latter may be asymptotically correct at high densities.

In the case of "Lennard-Jones molecules," the method yields results [1] which applied to argon agree very well with experimental observations at pressures below about 4000 atmospheres, over a region where no analytical approximation has given agreement. At higher pressures there is disagreement, with a suggestion that the experimental observations may be in error. If the disagreement is real it has interesting consequences in the study of intermolecular forces. Just as with hard spheres, there is agreement with the free-volume approximation at very high densities.

The method has also been tested successfully on the so-called "lattice gas" with nearest neighbor interactions, by comparison with known analytical results [4].

We hope in the future to apply the method to systems of molecules of different kinds (mixtures) and to non-spherical molecules.

If we use the notation

$$\overline{M}_D(s) = \left[\frac{1}{s^2 + \omega_0^2 s \overline{q}_R} \right] = \left[\frac{1}{s^2 + \omega_0^2 \frac{sh}{1 - e^{-sh}}} \right], \quad (5)$$

then we can write

$$\overline{M}_D + \frac{\omega_0^2 h \overline{M}_D}{s(1 - e^{-sh})} = \frac{1}{s^2}. \quad (6)$$

Multiplying both sides of (6) by the factor $(1 - e^{-sh})^2$, we obtain

$$M_D(1 - e^{-sh})^2 + \frac{\omega_0^2 h \overline{M}_D}{s} (1 - e^{-sh}) = \frac{1}{s^2} (1 - e^{-sh})^2. \quad (7)$$

Inverse Laplace transforming (7), we obtain

$$M_D(t) - 2M_D(t - h) + M_D(t - 2h) + \omega_0^2 h \int_{t-h}^t M_D(\tau) d\tau = 0 \quad (8)$$

the integro-difference equation for $M_D(t)$.

Imposing the mathematical constraint that, except at $t=0, h, 2h, \dots, kh$, $M_D(t)$ is a constant staircase-type function, we see that the integro-difference equation (8) reduces to

$$M_D(t) - (2 - \omega_0^2 h)M_D(t - h) + M_D(t - 2h) = 0 \quad (9)$$

a linear, homogeneous, difference equation with constant coefficients for the eigenfunction, $M_D(t)$.

It turns out that

$$M_D(t) = \frac{h}{\sin(\gamma)} \overline{\sin}(\gamma t), \quad (10)$$

where

$$\overline{\sin}(\gamma t) = \sin(\gamma kh) \text{ for } kh \leq t < (k+1)h \\ k = 0, 1, 2, \dots$$

and

$$\gamma = \frac{1}{h} \arctan \left[\frac{h\omega_0 \sqrt{4 - \omega_0^2 h^2}}{2 - \omega_0^2 h^2} \right].$$

In a similar fashion, it can be shown that the other eigenfunction is given by

$$N_D(t) = L^{-1} \left[\frac{s}{s^2 + \omega_0^2 s \overline{q}_R} \right] \\ = \overline{\cos}(\gamma t) + \left[\frac{1 - \omega_0^2 h^2 - \cos(\gamma)}{\sin(\gamma)} \right] \overline{\sin}(\gamma t) \quad (11)$$

where $\overline{\cos}(\gamma t) = \cos(\gamma kh)$ for $kh \leq t < (k+1)h$

$$k = 0, 1, 2, \dots$$

Thus, we see that the digital mechanization of

$$\ddot{x} + \omega_0^2 x = f(t) \quad (1)$$

by a DDA employing the rectangular rule yields a digital solution given by

$$x_D(t) = \dot{x}(0)M_D(t) + x(0)N_D(t) + \int_0^t f(\tau)M_D(t - \tau)d\tau \\ = \dot{x}(0) \frac{h}{\sin \gamma} \overline{\sin}(\gamma t) \\ + x(0) \left[\cos(\gamma t) + \left(\frac{1 - \omega_0^2 h^2 - \cos \gamma}{\sin \gamma} \right) \overline{\sin}(\gamma t) \right] \\ + \frac{h}{\sin \gamma} \int_0^t f(\tau) \overline{\sin}[\gamma(t - \tau)]d\tau. \quad (12)$$

METHODS OF COMPENSATING FOR ERRORS IN THE DIGITAL SOLUTION

The exact solution for

$$\ddot{x} + \omega_0^2 x = f(t) \quad (1)$$

is

$$x(t) = \frac{\dot{x}(0) \sin(\omega_0 t)}{\omega_0} + x(0) \cos(\omega_0 t) \\ + \frac{1}{\omega_0} \int_0^t f(\tau) \sin[\omega_0(t - \tau)]d\tau$$

as contrasted with the digital solution

$$x_D(t) = \frac{\dot{x}(0)h + [1 - \omega_0^2 h^2 - \cos(\gamma h)]x(0)}{\sin(\gamma h)} \overline{\sin}(\gamma t) \\ + x(0) \overline{\cos}(\gamma t) \\ + \frac{h}{\sin(\gamma h)} \int_0^t f(\tau) \overline{\sin}[\gamma(t - \tau)]d\tau. \quad (13)$$

This comparison of exact and digital solutions suggests three successively applied techniques for compensation of digital truncation errors.

The first and most obvious compensation we can introduce is to force γ to equal ω_0 . We do this by using a value ω_1^2 in the serial multiplier so that ω_1^2 satisfies the relationship

$$\omega_0 = (1/h) \arctan \left[\frac{h\omega_1 \sqrt{4 - \omega_1^2 h^2}}{2 - \omega_1^2 h^2} \right]. \quad (14)$$

The partially compensated digital solution then takes the form

$$x_D'(t) = \left[\frac{x(0)h + (1 - \omega_1^2 h^2 - \cos \omega_1 h)x(0)}{\sin \omega_0 h} \right] \overline{\sin}(\omega_0 t) \\ + x(0) \overline{\cos}(\omega_0 t) \\ + \frac{h}{\sin(\omega_0 h)} \int_0^t f(\tau) \overline{\sin}[\omega_0(t - \tau)]d\tau. \quad (15)$$

The solution $x_D'(t)$ suggests that a well-chosen but incorrect value for $x(0)$ be used called $\dot{x}_1(0)$, so that

$$\frac{\dot{x}_1(0)h + (1 - \omega_1^2 h^2 - \cos \omega_1 h)x(0)}{\sin \omega_0 h} = \frac{x(0)}{\omega_0}. \quad (16)$$

With these two digital compensations, the digital solution looks like

$$x_D''(t) = \frac{x(0) \overline{\sin}(\omega_0 t)}{\omega_0} + x(0) \overline{\cos}(\omega_0 t) + \frac{h}{\sin(\omega_0 h)} \int_0^t f(\tau) \sin \omega_0(t - \tau) d\tau. \quad (17)$$

The third and last compensation technique consists of multiplying f by a constant, $c = c(h)$, so that

$$\frac{c(h)}{\sin(\omega_0 h)} = \frac{1}{\omega_0}. \quad (18)$$

With the three compensations introduced, the digital solution is given by

$$x_D(t) = \frac{x(0) \overline{\sin}(\omega_0 t)}{\omega_0} + x(0) \overline{\cos}(\omega_0 t) + \frac{1}{\omega_0} \int_0^t f(\tau) \overline{\sin}[\omega_0(t - \tau)] d\tau \quad (19)$$

as contrasted with the exact solution given by (13). The only errors left are those generated by the convolution integral

$$\frac{1}{\omega_0} \int_0^t f(\tau) \overline{\sin}[\omega_0(t - \tau)] d\tau.$$

INSTABILITY CAN BE INTRODUCED BY WRONG CHOICE OF ALGORITHM OR BY DELAY BETWEEN INTEGRATIONS

If the trapezoidal algorithm is used instead of the rectangular algorithm for integration in the digital mechanization of

$$\dot{x} + \omega_0^2 x = f(t)$$

the following expression is obtained as the digital solution:

$$x_D(t) = \dot{x}(0) \left[\frac{he^{-\alpha h}}{\sin(\gamma h)} e^{\alpha t} \overline{\sin}(\gamma t) \right] + x(0) \left\{ e^{\alpha t} \overline{\cos}(\gamma t) + \left[\frac{e^{-\alpha h} - \overline{\cos}(\gamma h)}{\sin(\gamma h)} \right] e^{\alpha t} \overline{\sin}(\gamma t) \right\} + \frac{he^{-\alpha h}}{\sin(\gamma h)} e^{\alpha t} \int_0^t f(\tau) \overline{\sin}[\gamma(t - \tau)] d\tau \quad (20)$$

where

$$\alpha = \frac{1}{2h} \log_e \left[1 + \frac{\omega_0^2 h^2}{2} \right]$$

$$\gamma = \frac{1}{h} \arctan \left[\frac{\omega_0 h}{4} \sqrt{16 - \omega_0^2 h^2} \right].$$

The exponential term, $e^{\alpha t}$, where α is positive as indicated, reveals that the truncation errors, incurred by use of the trapezoidal algorithm increase exponentially whereas there is no such tendency toward instability introduced by the rectangular algorithm.

Another source of instability can be delays between successive integrations. For example, suppose we purposely delay one integration, say, one sampling period, behind the other integration. That is, suppose we mechanize according to

$$\int_0^t \dot{x}_D(\tau) d\tau - \int_0^t \dot{x}(0) d\tau + \omega_0^2 \int_0^t \int_0^{t-h} x_D(\tau) d\tau dt_1 = \int_0^t \int_0^{t_2} f(\tau) d\tau dt \quad (21)$$

rather than by (3).

Taking the transform of (21) and assuming the rectangular rule, we obtain

$$\tilde{x}_D(s) = \frac{\dot{x}(0)}{s^2 + \omega_0^2(\overline{sf}_R)e^{-hs}} + \frac{sx(0)}{s^2 + \omega_0^2(\overline{sf}_R)e^{-hs}} + \frac{\tilde{f}(s)}{s^2 + \omega_0^2(\overline{sf}_R)e^{-hs}}.$$

The eigenfunctions corresponding to

$$\frac{1}{s^2 + \omega_0^2(\overline{sf}_R)e^{-hs}} \quad \text{and} \quad \frac{s}{s^2 + \omega_0^2(\overline{sf}_R)e^{-hs}}$$

are

$$\exp\left(\frac{\log_e \sqrt{1 + \omega_0^2 h^2}}{h} t\right) \times \left[\cos\left(\frac{\arctan(h\omega_0)}{h} t\right) \right]$$

and

$$\exp\left(\frac{\log_e \sqrt{1 + \omega_0^2 h^2}}{h} t\right) \times \left[\sin\left(\frac{\arctan(h\omega_0)}{h} t\right) \right].$$

The eigenfunctions show that the effect of a one-sampling period delay between integrations is to introduce a positive exponential-type factor into the solution—in other words, the solution is rendered inherently unstable.

BIBLIOGRAPHY

- [1] E. I. Jury, "Synthesis and critical study of sampled-data control systems," *Trans. AIEE*, vol. 75, pt. 2, pp. 141-149; July, 1956.
- [2] R. H. Barker, "The pulse transfer function and its application to sampling servo systems," *Proc. IEE*, vol. 99, pt. 4, pp. 302-317; 1952.

Automatic Digital Matric Structural Analysis

M. CHIRICO†, B. KLEIN†, AND A. OWENS†

LIST OF SYMBOLS

- A = cross-sectional area of bar.
 A_{\max} = maximum cross-sectional area of bar.
 A_{\min} = minimum cross-sectional area of bar.
 A_p = area of panel.
 B_{\max} = length of longer parallel edge of tapered panel.
 B_{\min} = length of shorter parallel edge of tapered panel.
 $D_x(i)$ = x component of displacement at the joint i in the coordinate system (X, Y, Z). $D_y(i)$ and $D_z(i)$ are defined similarly.
 E = Young's modulus.
 $F_x(i)$ = x component of the external force applied at the joint i . $F_y(i)$ and $F_z(i)$ are defined similarly.
 G = shear modulus.
 L_{ij} = length of the bar with end joints i and j .
 P_{ij} = load at the joint i on the bar B_{ij} .
 q = shear flow.
 \bar{q} = average shear flow.
 δ_{ij} = the displacement of the joint i in the direction of the bar B_{ij} .
 $\Delta x_{ij} = x_i - x_j$. Δy_{ij} and Δz_{ij} are defined similarly.
 t_D = thickness of panel at intersection of its diagonals.
 i, j, k, m, n = subscripts.

INTRODUCTION

THE accelerated pace of computer development has made possible the rapid solution of complex problems. The time necessary to set up problems has begun to surpass greatly machine solution time. Consequently, more emphasis is needed on means of making use of digital machines and allied equipment in setting up problems automatically.

Heretofore, the preparation of certain matric equations appearing in structural analysis has been a tedious task. The procedures have required a large amount of judgment and tiresome hand computation. The chances for errors have been prevalent.

The present paper presents a method whereby the above factors can be minimized or negated. Input data are reduced to a minimum. All logical decisions are carried out completely automatically so as to arrange the matrix automatically. Machine time is found to be very small relative to the time previously needed to set up problems. Therefore, this coded program should prove very useful to structures and allied engineers.

Some familiarization with Klein^{1,2} is helpful but not necessary for the understanding of the development in this paper. The basic concepts are the ones of joint, bar, and panel. The joints connect the bars and the bars border the panels.

MACHINE SIMULATION OF ELEMENTS

The code is built around the basic elements: joints, bars, and panels. The information below noted by asterisk (*) is computed.

1) Joints

A group of words is assigned to each joint containing the following information:

- a) Its position coordinates $x, y,$ and z
- b) Whether it is fixed
- *c) All the bars attached to the joint.

2) Bars

A group of words is assigned to each bar with the following data:

- a) Its two end joints
- b) Whether it is tapered
- c) Its cross-sectional area (both maximum and minimum, if tapered)
- *d) The direction cosines $x, y,$ and z
- *e) The length of the bar L_{ij} .

3) Panels

A group of words is assigned to each panel with information on:

- a) The four corner joints
- *b) The area
- c) The thickness
- *d) All of the bordering bars
- *e) B_{\max} and B_{\min} (lengths of the parallel sides)
- f) Whether it is tapered.

The number of cells in a group must be multiples of eight to allow the use of multiple index registers. A joint, bar, and panel, grouped together in 40 cells to economize on space, are not necessarily related.

THE CODE

A) Input

The information without asterisks, described in the previous section, is read into the proper location. The numbers of the joints appearing in the bars and panels are converted into the two's complement of that joint's address. The joint then can be referred to with an index register.

¹ B. Klein, "A simple method of matric structures analysis," *J. Inst. Aeronaut. Sciences*, vol. 24, pp. 40-46; January, 1957.

² B. Klein, "A simple method of metric structural analysis, part II—effects of taper and a consideration of curvature," *J. Inst. Aeronaut. Sciences*, vol. 24, pp. 813-820; November, 1957.

B) Bar Cross Referencing

1) The following is computed for each bar B_{ij} :

$$a) \Delta x = x_i - x_j, \quad (1)$$

$$b) \Delta y = y_i - y_j, \quad (2)$$

$$c) \Delta z = z_i - z_j. \quad (3)$$

2) The bar address complement is stored in the groups of the joints i and j .

3) Every panel is examined. If both the joints i and j are in it, the bar B_{ij} borders it. In this case the two's complement of the bar address is placed in the panel group and the two's complement of the panel address is placed in the bar group.

C) Joint Equations

1) For each joint i the following force equilibrium equations are entered in the matrix

$$a) \sum_j P_{ij} \left(\frac{\Delta x}{L} \right)_{ij} = F_x(i), \quad (4)$$

$$b) \sum_j P_{ij} \left(\frac{\Delta y}{L} \right)_{ij} = F_y(i), \quad (5)$$

$$c) \sum_j P_{ij} \left(\frac{\Delta z}{L} \right)_{ij} = F_z(i). \quad (6)$$

D) Panel Area

1) The two parallel bars for each panel are found by comparing the direction cosines. The bars are rearranged so that the longest parallel side is first and the other parallel bar is second.

2) Twice the area A_p is computed by the following (see Fig. 1).

$$2A_p = h(B_{\min} + B_{\max}) \quad (7)$$

where

$$h = L_{ij} \sin \theta_1 \quad (8)$$

$$\sin \theta_1 = \sqrt{1 - \cos^2 \theta_1} \quad (9)$$

$$\begin{aligned} \cos \theta_1 = & \left(\frac{\Delta x}{L} \right)_{ij} \left(\frac{\Delta x}{L} \right)_{jm} + \left(\frac{\Delta y}{L} \right)_{ij} \left(\frac{\Delta y}{L} \right)_{jm} \\ & + \left(\frac{\Delta z}{L} \right)_{ij} \left(\frac{\Delta z}{L} \right)_{jm}. \end{aligned} \quad (10)$$

The cosine above is equal to the dot product only if both sides are directed away from or toward the joint. If the sides are directed in opposite ways, the dot product must be multiplied by minus one.

E) Panel Equation

1) The shear panel displacement equation for the panel in Fig. 1 is

$$\begin{aligned} - (2A_p/GI_D) \bar{q}_{jmk} - B_{\max}(\delta_{ij} + \delta_{jm}) + B_{\min}(\delta_{ki} + \delta_{ik}) \\ - L_{ij}(\delta_{ij} + \delta_{ji}) + L_{km}(\delta_{km} + \delta_{mk}) = 0. \end{aligned} \quad (11)$$

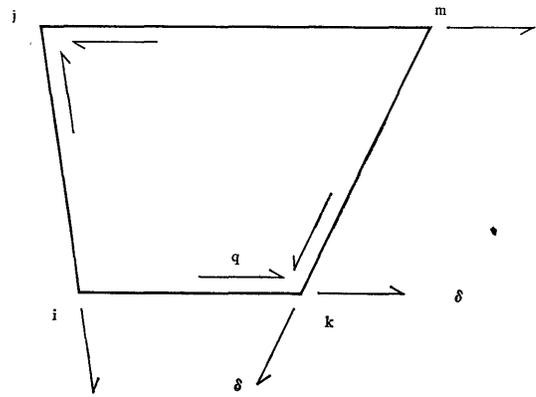


Fig. 1—Shear panel.

2) The arrows inside the panel indicate the direction of shear flow. The arrows outside the panel indicate the direction of displacements.

3) The corresponding panel and bar numbering for Fig. 1 is

panel: P_{jmk}

bars: $B_{mj}, B_{ki}, B_{ij},$ and B_{km} .

4) The panel numbering determines the positive direction of the shear flows, which are directed toward the joints designated by the first and third subscripts of the panel. (See Fig. 1.) The bar numbering determines the direction of positive displacement of the joint which is toward the joint determined by the first subscript of the bar.

5) The signs of the terms in the equation are:

- a) Shear term. This sign is minus.
- b) Displacement term. If the direction of positive displacement is the same as the direction of positive shear flow along a side, this sign is plus; if they are opposite, the sign is minus.

6) The code determines the signs by examining the joint numbers. If the first bar subscript is equal to the first or third panel subscript, the sign is plus; otherwise, it is minus.

7) The length of the side is the factor outside the parenthesis in the displacement term. Note that the length of the opposite side is used in the case of the parallel sides.

F) Bar Equations

1) The axial element equilibrium equation for each bar is

$$P_{ij} - P_{ji} + \sum_n b_n(\pm \bar{q}_n) = 0. \quad (12)$$

2) When the bar B_{ij} is one of the parallel bars bordering a panel, b_n of (12) is the length of the other parallel bar; otherwise, it is the length of the bar B_{ij} .

3) The sign of a shear term is positive if the positive direction of the shear along an edge of the

panel and the positive direction of the displacement in the bar bordering that edge are opposite; otherwise, the sign of the shear term is negative.

- 4) The axial force displacement equation for a tapered bar is

$$P_{ij} + L_{ij} \sum_n (\pm \bar{q}_n) f(B_{\min}/B_{\max}, A_{\max}/A_{\min}) + (E\bar{A}/L_{ij})(\delta_{ji} - \delta_{ij}) = 0 \quad (13)$$

where P_{ij} is the force at the narrow end of the bar. The sign of the shear term is plus if the shear and displacement directions oppose, and

$$f(B_{\min}/B_{\max}, A_{\max}/A_{\min}) = \frac{(A_{\min}/A_{\max})^{0.175}}{1 + (B_{\min}/B_{\max})^{0.7}} \quad (14)$$

$$A = A_{\min} + \left(\frac{A_{\min}/A_{\max}}{2} \right)^{0.175} (A_{\max} - A_{\min}) \quad (15)$$

if the taper is linear.

- 5) For a nontapered bar the equation is

$$P_{ij} + P_{ji} + (2EA/L_{ij})(\delta_{ji} - \delta_{ij}) = 0. \quad (16)$$

- G) The Flow Chart follows.

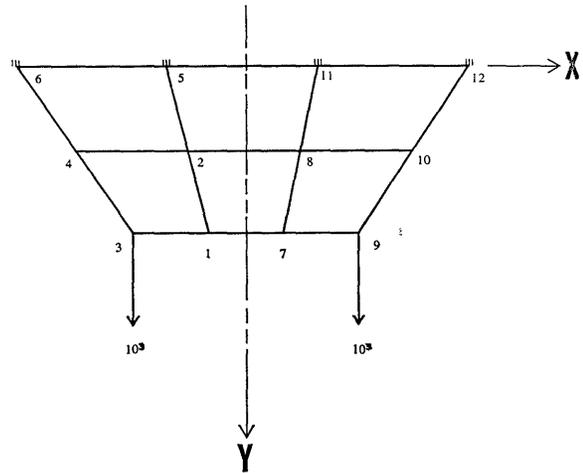
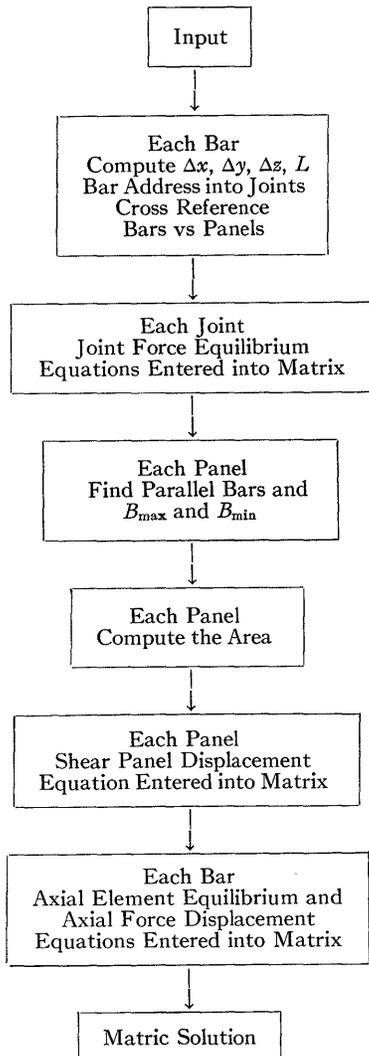


Fig. 2—Example problem.

Example Problem

The structure in Fig. 2 is analyzed by the code. The complete input data describing the structure appears in the next section, which is followed by the matrix generated and the solution with the matrix column numbers. The elements without matrix column numbers are obtained by a simple calculation from matrix computed elements; e.g., Delta 1-7 equals Delta 1-3.

Total computing time, i.e., the time for both matrix setup and solution, is about 0.01 hour.

Problems of a much more formidable nature have been arranged and solved by the program, and work is in progress for improving the code. For example, use of instantaneous coordinates may create many more zero elements in the matrix.

FORCES		INPUT		
JOINT		X	Y	Z
3		0.0	1000.0	0.0
9		0.0	1000.0	0.0

JOINTS		X	Y	Z	FIXED
NUMBER					
1		- 2.5	16.0	0.1	NO
2		- 3.75	8.0	0.1	NO
3		- 7.5	16.0	0.1	NO
4		-11.25	8.0	0.1	NO
5		- 5.0	0.0	0.1	YES
6		-15.0	0.0	0.1	YES
7		2.5	16.0	0.1	NO
8		3.75	8.0	0.1	NO
9		7.5	16.0	0.1	NO
10		11.25	8.0	0.1	NO
11		5.0	0.0	0.1	YES
12		15.0	0.0	0.1	YES

BARS		TAPERED	A _{MAX}	A _{MIN}	E
JOINT 1	JOINT 2				
3	4	YES	1.0	0.66667	1.0
4	6	YES	1.3333	1.0	1.0
9	10	YES	1.0	0.66667	1.0
10	12	YES	1.3333	1.0	1.0
1	2	YES	1.0	0.66667	1.0
2	5	YES	1.3333	1.0	1.0
7	8	YES	1.0	0.66667	1.0
8	11	YES	1.3333	1.0	1.0
3	1	NO	0.66667	0.66667	1.0
4	2	NO	1.0	1.0	1.0
9	7	NO	0.66667	0.66667	1.0
10	8	NO	1.0	1.0	1.0
1	7	NO	0.66667	0.66667	1.0
2	8	NO	1.0	1.0	1.0

PANELS

NUM-BER	JOINT	JOINT	JOINT	JOINT	G-SHEAR	THICK-NESS
1	3	4	2	1	0.4	0.07
2	4	6	5	2	0.4	0.1
3	1	2	8	7	0.5	0.07
4	2	5	11	8	0.4	0.1
5	7	8	10	9	0.4	0.07
6	8	11	12	10	0.4	0.1

THE MATRIX

ROW	COL	VALUE	ROW	COL	VALUE
1	1	0.42443388	20	19	-7.5
1	2	-0.42443388	20	37	-5.0
1	3	1.0	20	43	-5.0
2	1	-0.90545894	21	38	5.0
2	2	0.90545894	21	35	-1.0
3	4	-1.0	21	10	-1.0
3	5	0.90545894	21	23	1.0
3	6	-0.42443388	22	30	-0.26666667
4	7	0.15437688	22	37	-0.26666667
4	8	-0.15437688	22	10	1.0
4	9	-1.0	22	23	1.0
4	10	1.0	23	38	-7.5
5	7	-0.98801203	23	13	-1.0
5	8	0.98801203	23	26	1.0
6	11	-0.15437688	24	34	-0.26666667
6	12	-1.0	24	40	-0.26666667
6	13	1.0	24	13	1.0
7	11	0.98801203	24	26	1.0
8	14	-0.42443388	25	32	-5.0
8	15	0.42443388	25	27	10.0
8	16	-1.0	25	9	-1.0
9	14	-0.90545894	25	3	1.0
9	15	0.90545894	26	30	0.26666667
10	17	-1.0	26	31	-0.26666667
10	18	0.90545894	26	9	1.0
10	19	0.42443388	26	3	1.0
11	20	-0.15437688	27	32	7.5
11	21	0.15437688	27	12	-1.0
11	22	1.0	28	34	0.26666667
11	23	-1.0	28	6	-0.26666667
12	20	-0.98801203	28	12	1.0
12	21	0.98801203	29	44	5.0
13	24	0.15437688	29	41	-10.0
13	25	1.0	29	22	-1.0
13	26	-1.0	29	16	1.0
14	24	0.98801203	30	37	0.26666667
15	27	-3499.9999	30	43	-0.26666667
15	28	-8.8352984	30	22	1.0
15	29	8.0970674	30	16	1.0
15	30	-10.0	31	44	-7.5
15	31	-10.0	31	25	-1.0
16	32	-3571.4285	32	40	0.26666667
16	28	-8.8352984	32	19	0.26666667
16	4	-8.8352984	32	25	1.0
16	29	8.0970674	33	35	8.0970674
16	33	-8.0970674	33	27	-8.0970674
16	34	-7.5	33	45	-1.0
16	6	-7.5	33	8	1.0
16	30	5.0	34	35	4.2360996
16	31	5.0	34	27	-4.2360996
17	35	-3499.9999	34	29	-0.1430726
17	36	8.0970674	34	8	1.0
17	29	-8.0970674	35	38	8.0970674
17	30	-1.0	35	32	-8.0970674
17	37	1.0	35	7	-1.0
18	38	-3571.4285	35	11	1.0
18	36	8.0970674	36	38	4.3028433
18	39	8.0970674	36	32	-4.3028433
18	29	-8.0970674	36	33	-0.10150823
18	33	-8.0970674	36	29	0.10150823
18	34	-7.5	36	11	1.0
18	40	7.5	37	41	8.0970674
18	30	5.0	37	35	-8.0970674
18	37	-5.0	37	46	-1.0
19	41	-3499.9999	37	21	1.0
19	42	8.8352984	38	41	4.2360996
19	36	-8.0970674	38	35	-4.2360996
19	37	10.0	38	36	-0.14307260
19	43	10.0	38	21	1.0
20	44	-3571.4285	39	44	8.0970674
20	42	8.8352984	39	38	-8.0970674
20	17	8.8352984	39	20	-1.0
20	36	-8.0970674	39	24	1.0
20	39	-8.0970674	40	44	4.3028433
20	40	7.5	40	38	-4.3028433

ROW	COL	VALUE	ROW	COL	VALUE
40	39	-0.10150823	46	41	-4.6223159
40	36	0.10150823	46	42	-0.13111821
40	24	1.0	46	15	1.0
41	27	8.8352984	47	44	-8.8352984
41	47	-1.0	47	14	-1.0
41	2	1.0	48	44	-4.6951448
42	27	4.6223159	48	17	-0.093026737
42	28	-0.13111821	48	42	0.093026737
42	2	1.0	27	49	-468.75001
43	32	8.8352984	28	49	-468.75001
43	1	-1.0	31	49	-468.75001
44	32	4.6951448	32	49	-468.75001
44	4	-0.903026737	43	49	-1104.4123
44	28	0.093026737	44	49	-1104.4123
45	41	-8.8352984	47	49	-1104.4123
45	48	-1.0	48	49	-1104.4123
45	15	1.0			

THE SOLUTION

MATRIX COLUMN NUMBER	ELEMENT	VALUE	MATRIX COLUMN NUMBER	ELEMENT	VALUE
24	P 1 2	0.0	35	Q 4	0.0
25	P 1 3	1631.4448	32	Q 5	-407.4740
26	P 1 7	1631.4448	27	Q 6	-105.1129
20	P 2 1	3299.3447	39	DELTA 1 2	43445.283
22	P 2 4	986.2406	40	DELTA 1 3	6117.8879
21	P 2 5	3299.3447		DELTA 1 7	6117.8879
23	P 2 8	986.2406	36	DELTA 2 1	26172.822
	P 3 1	418.5939	37	DELTA 2 4	3698.4027
	P 3 4	986.2406		DELTA 2 8	3898.4027
16	P 4 2	0.0		DELTA 2 5	26172.822
14	P 4 3	7443.9683		DELTA 3 1	29813.814
15	P 4 6	7443.9683	17	DELTA 3 4	151221.70
46	P 5 2	4150.4514	43	DELTA 4 2	7396.8052
48	P 6 4	6515.2640	42	DELTA 4 3	53067.405
13	P 7 1	1631.4448		DELTA 4 6	53067.405
11	P 7 8	0.0		DELTA 7 1	6117.8879
12	P 7 9	1631.4448	33	DELTA 7 8	43445.282
10	P 8 2	986.2406	34	DELTA 7 9	6117.8875
7	P 8 7	3299.3448		DELTA 8 2	3698.4027
	P 8 10	986.2406	29	DELTA 8 7	26172.822
8	P 8 11	3299.3448	30	DELTA 8 10	3698.4022
	P 9 7	418.5939		DELTA 8 11	26172.822
9	P 9 10	986.2406		DELTA 9 7	-29813.814
3	P 10 8	0.0	4	DELTA 9 10	151221.70
1	P 10 9	7743.9684	31	DELTA 10 8	7396.8045
2	P 10 12	7743.9684	28	DELTA 10 9	53067.405
45	P 11 8	4150.4514		DELTA 10 12	53067.405
47	P 12 10	6515.2640	19	DX 3	-29813.814
44	Q 1	407.4740	18	DY 3	180986.33
41	Q 2	105.1129	6	DX 9	29813.813
38	Q 3	0.0	5	DY 9	180986.33

A New Approach to High-Speed Logic

W. D. ROWE†

INTRODUCTION

MOST approaches to the construction of high-speed computers and logic systems in the past have been first, to go from serial to parallel circuitry, and then to use faster and faster components to increase the operational speed of the circuitry. High-speed components have reached the state where we are not talking about switching times in the order of the light-foot. That is, if we have a switching circuit with a switching time of one millimicrosecond, the theoretical limit in which we can switch this information from one point to another requires that the separation distance be less than one foot. This is the theoretical limit imposed by the velocity of light and shows that we are rapidly approaching the limits of switching speed.

It is therefore necessary that we look again at the organization and utilization of logic techniques to determine whether there are not other means of obtaining high-speed operation from logic circuitry, and thereby of bypassing the need for extremely fast components.

It has long been recognized¹ that one solution to this problem would be the use of canonical logic forms of either the minterm (sum of products) or maxterm (product of sums) form.² The practical application of this technique has long been hampered by the lack of a suitable electronic device capable of responding to the multiple input-output loadings required. This paper discusses a transistor device, the "Modified NOR Circuit," which is capable of handling up to 25 inputs and 25 outputs with response times of about 80 mμsec. This device is applied to the design of a high-speed adder and a high-speed counter.

The design approach used here is termed "parallel-parallel" logic. This term arises from the fact that not only is the function constructed in parallel, but the logic is also constructed in parallel. A comparison of this logic with others is as follows:

Serial Logic	
Function—Serial	
Logic—Serial	
Parallel Logic	
Function—Parallel	
Logic—Serial	/
Parallel-Parallel Logic	
Function—Parallel	
Logic—Parallel	

† Westinghouse Electric Corp., Buffalo, N. Y.

¹ R. K. Richards, "Arithmetic Operations in Digital Computers," D. Van Nostrand Co., Inc., New York, N. Y.; 1955.

² M. Phister, Jr., "Logical Design of Digital Computers," John Wiley and Sons, Inc., New York, N. Y.; 1958.

FUNDAMENTALS OF PARALLEL-PARALLEL LOGIC

With a logic circuit that has an infinite number of inputs and outputs and if all input signals and their complements are available, logic arrays can be constructed using only one level of logic circuits. The advantage of this lies in the fact that the complete operation time of a logic array consists of only one logic circuit propagation time. This means that the maximum speed of a logic array is the same as the maximum operating speed of a single logic component.

To explain this procedure, it can be shown that any logical operation can be written using conventional Boolean notations (where + indicates an OR operation, and · represents an AND operation, and — a complement) as a sum of products or a product of sums, *i.e.*,

$$y = x_1 \cdot x_2 + \bar{x}_1 \cdot \bar{x}_2 = (x_1 + \bar{x}_2)(\bar{x}_1 + x_2). \quad (1)$$

The left-hand expression in the equation may be represented by a number of AND circuits (first level) working into a single OR gate (second level) (Fig. 1), while the right-hand (equivalent) expression can be represented by a number of OR gates (first level) working into an AND gate (second level) (Fig. 2). It is then obvious that if each logic circuit has no limit in its fan-in and fan-out, all logic can be done on two levels or a depth of two. Now, the OR operation is electronically unique, and can often be performed by a simple junction of leads without any logic elements. In the case of the NOR logic element, to be discussed below, the OR operation need not be performed at all, since this element accepts multiple inputs and performs the necessary OR operation implicitly.

If all signal complements are not available, a third level is required for negation. Since most signals come from bistable registers in parallel-parallel operation, both the signal and its complement are almost always available. The insertion of a third level in some inputs and not others can cause occurrence of certain race conditions if care is not exercised. Memory and counting circuits also require special consideration.

Previous papers have discussed the use of a particular universal logic circuit called a "NOR" circuit.^{3,4} The logic of this circuit is such that an output exists if, and only if, neither input *A* NOR, *B* NOR, *C* NOR, etc., is present. (See Fig. 3.) This circuit is universal in that it can perform all logic functions when combined in various forms with other NOR circuits. The right-

³ W. D. Rowe, "The transistor NOR circuit," 1957 WESCON CONVENTION RECORD, pt. 4, pp. 231-245.

⁴ W. D. Rowe and T. A. Jeeves, "The NORDIC II computer," 1957 WESCON CONVENTION RECORD, pt. 4, pp. 85-95.

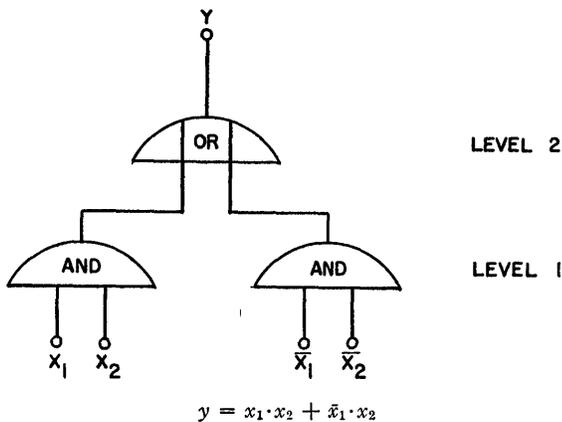


Fig. 1—Representation of two-level logic—AND-NOR.

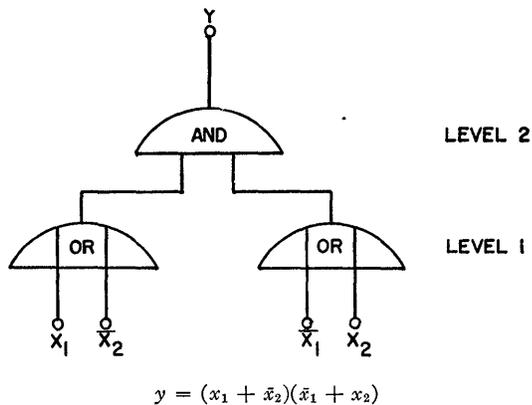


Fig. 2—Representation of two-level logic—OR-AND.

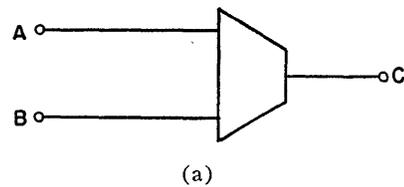
hand expression of (1) can be expressed simply by operation of several NOR circuits (first level) into another NOR circuit (second level). (See Fig. 4.) (Where the output of the array is fed to other similar circuits the second level may be omitted, as remarked above.)

The particular advantage of using the NOR circuit instead of English logic circuits is twofold. First, only one type of circuit is required, so that only a single propagation constant is necessary; and second, it is possible to build NOR circuits with numbers of inputs and outputs that are quite large, in the order of 25 inputs and 25 outputs. This number is sufficiently large for many applications.

APPLICATION OF PARALLEL-PARALLEL TECHNIQUE TO A FULL ADDER

The examples used so far are trivial in that they are two levels for both the parallel and the parallel-parallel cases. In order to illustrate more fully the advantages of the parallel-parallel method, a slightly more sophisticated example will be examined.

This case is that of a full adder circuit. A basic full adder (Fig. 5) provides the sum of two addend variables and the carry variable from the preceding stage in binary form. The circuit is built up of two half-adders



A \ B	0	1
0	1	0
1	0	0

Fig. 3—NOR logic; (a) diagrammatic NOR; (b) truth table. Logical expression: an output appears at C if neither input A nor input B is present.

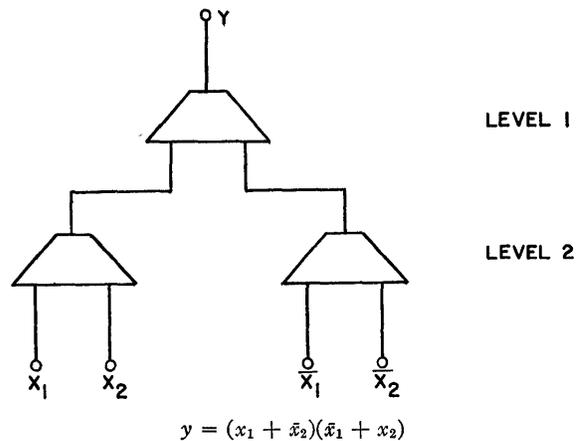


Fig. 4—Representation of two-level logic—NOR-NOR.

utilizing a maximum of two inputs per logic circuit. The depth is five, so that five propagation times are required.

The logic expression for this circuit is shown to be

$$s = [\overline{x\bar{y} + \bar{x}y + \bar{c}}][x\bar{y} + \bar{x}y + c], \tag{2}$$

which is derived directly by the substitution of the output of one-half adder which can be expressed in two equivalent ways:

$$H = \bar{x}y + x\bar{y} + (x + y)(\bar{x} + \bar{y}) \tag{3}$$

into the input of the second half adder, expressed as

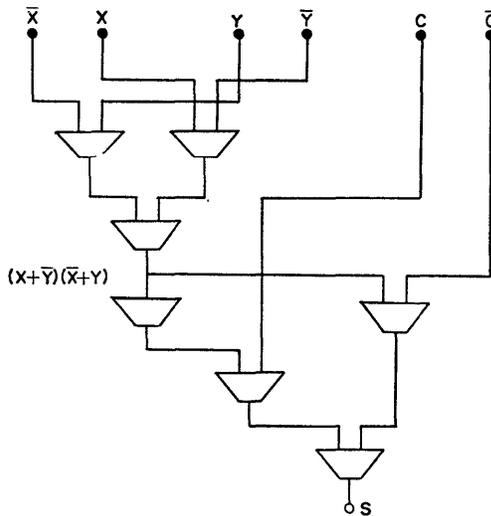
$$S = (\bar{H} + \bar{C})(H + C) \tag{4}$$

to form (2). Here x and y are the two addend variables and C the preceding carry variable.

Manipulation of this equation brings it into parallel-parallel form so that the equivalent expression is $s = (x + \bar{y} + \bar{c})(x + y + c)(\bar{x} + y + \bar{c})(\bar{x} + \bar{y} + c)$. (5)

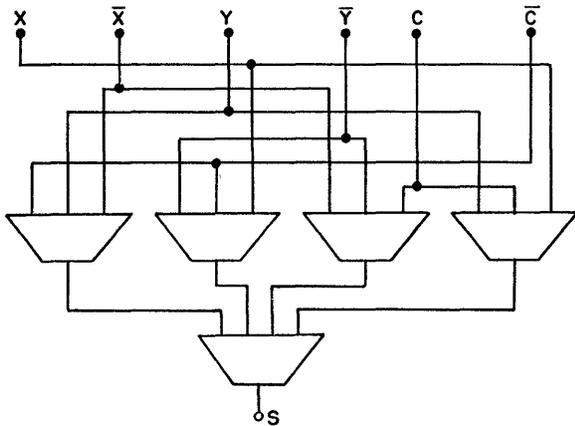
The derivation of this equation from (2) is shown in Appendix I.

This equation leads to the circuit of Fig. 6, where multi-input logic circuits are used. The depth of logic is now only two.



$$[(x + \bar{y})(\bar{x} + y) + c][(x + \bar{y})(\bar{x} + y) + c] = s$$

Fig. 5—Full adder consisting of two half-adders, using only two inputs per NOR. Depth = 5.



$$s = (\bar{x} + y + \bar{c})(x + \bar{y} + \bar{c})(\bar{x} + \bar{y} + c)(x + y + c)$$

Fig. 6—Full adder consisting of parallel-parallel logic.

In the example, fewer logic circuits are required in the parallel-parallel case, since most of the logic is accomplished in inputs of the logic circuits and the interwiring. This does not always hold true. The worst case of parallel-parallel logic has a maximum of $2^n - 1$ first-level logic circuits, where n is the number of inputs to each first-level logic circuit. There are q second-level logic circuits, where q is the number of desired outputs. Fortunately, most applications are so specialized that only a few first-level logic circuits are required. However, in the worst case, the second-level logic circuits must handle $2^n - 1$ inputs.

It is possible to reduce the number of logic circuits by compromising on depth, as was done by Weinberger and Smith.⁵

⁵ A Weinberger and J. L. Smith, "A one-microsecond adder using one megacycle circuitry," IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-5, pp. 65-73; June, 1956.

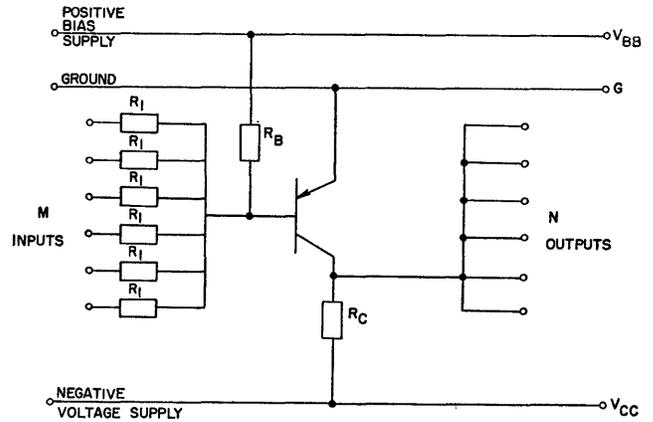


Fig. 7—Basic transistor NOR circuit.

THE MODIFIED TRANSISTOR NOR CIRCUIT

In order to facilitate the use of parallel logic it is necessary to have a circuit with as large a number of inputs (fan-in) and outputs (fan-out) as possible. Since it is extremely desirable to use only a single logic circuit, the transistor NOR circuit was selected for investigation.

The basic transistor NOR circuit is shown in Fig. 7. A negative voltage on any of the inputs, M , is sufficient to cause the transistor to saturate and supply a ground potential signal to the outputs, N . The absence of a negative voltage on the inputs causes the positive bias voltage to maintain the transistor at cut-off. Under this condition, the transistor being in a very high impedance state, the outputs see a negative voltage equal to

$$\text{output voltage} = V_{cc} \frac{R_I}{xR_c + R_I}$$

where R_I is the input resistor value of a NOR circuit being driven by one of the NOR circuit outputs, N ; R_c is the collector load resistor; and x is the number of outputs, N , actually connected to the inputs of other NOR circuits. Only one NOR circuit input is driven by one NOR circuit output. This equation shows that the output voltage is reduced as more inputs of succeeding circuits are connected to the outputs of the NOR circuit in question.

If the minimum voltage that appears under the fully loaded condition is called a "one" (this voltage being sufficient to saturate a succeeding NOR), and the absence of this voltage (ground) is called a "zero," the logic conditions of Fig. 3 are fulfilled and this is the basic NOR operation.

Since all inputs are resistive, and most of the logic is accomplished in the input network, only a small number of transistors are required as compared to the number of resistors. This is certainly desirable, as most of the logic is now accomplished by the wiring interconnections and resistors, which can be inexpensive, reliable, and miniaturized elements.

For extreme reliability, the basic NOR circuit seems to have a maximum fan-in, fan-out of six, which is

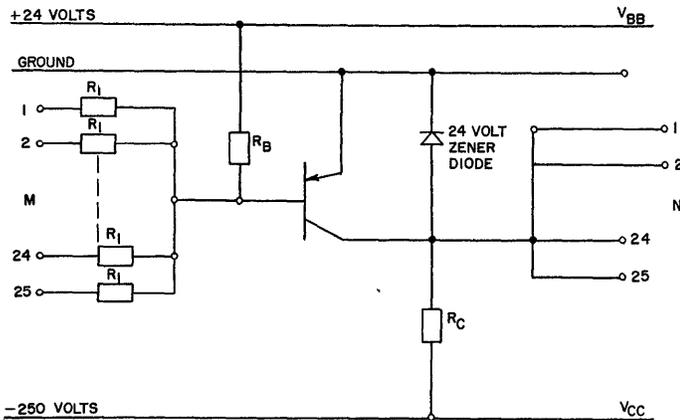


Fig. 8—The modified transistor NOR circuit—type A.

fairly small when one is considering parallel-parallel logic. In order to increase the fan-in and fan-out to about 25, a more suitable number, the basic NOR circuit has been modified so that the output voltage remains at essentially power-supply potential regardless of output loading. This allows the circuit to drive many more outputs than is possible in the basic circuit.

One means of modifying the NOR circuit is shown in Fig. 8. A 24-volt breakdown Zener diode is placed across the collector and emitter of the transistor such that output will never fall below 24 volts when the transistor is cut off (output load is restricted to never exceed a condition where the output voltage will fall below this value). Under this condition the current derived at the high voltage V_{cc} acting through the resistor is shunted between the load and the Zener diode. As the load varies (due to changes in the number of outputs used) the excess current is shunted through the diode so that the load is continually driven from a 24-volt output signal, during a "one" output condition.

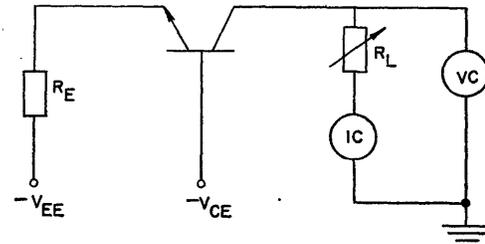
It is certainly possible to replace the diode with a resistor whose value is chosen such that output voltage is always in the fully loaded condition. This reduces the circuit flexibility since a different resistor value is required for every different condition of output loading.

A basic disadvantage of this circuit is its high power dissipation, arising from the high-voltage power supply acting across resistor, R_c . This disadvantage has been eliminated by the design of a second type of modified NOR circuit.

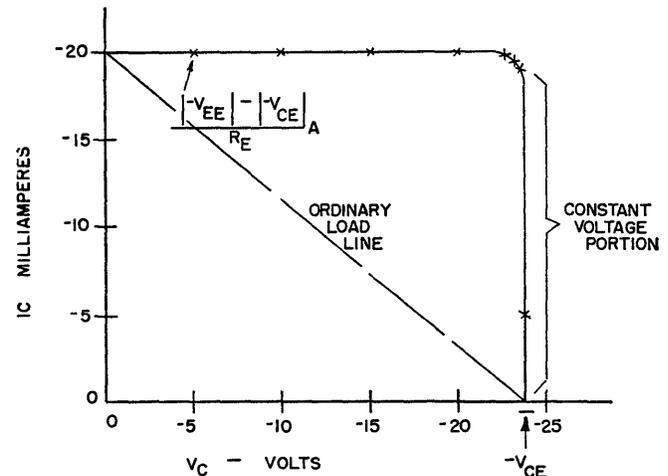
This circuit is based on the fact that under certain conditions a transistor makes an excellent constant voltage source. This operation is described in Fig. 9. A constant emitter current source is derived by the constant voltage difference between bias voltages (V_{EE} and V_{CE}) acting across emitter resistor R_E . This contains the maximum collector current to be

$$\frac{|-V_{EE}| - |-V_{CE}|}{R_E} A = I_{c_{max}},$$

where $|-V_{EE}| > |-V_{CE}|$ and A is the current gain of



(a)



(b)

Fig. 9—(a) Constant voltage circuit, (b) circuit operation plot.

the transistor. Below this value a range of constant potential exists. An actual plot of this can be made by observing the meters in the circuit of Fig. 9(a) when the load resistor R_L is varied. This plot is shown in Fig. 9(b) for a particular case where $I_{c_{max}} = 20$ ma and $V_{CE} = -24$ volts. The constant voltage portion is shown, and this is the operating range utilized.

This circuit is then combined with the NOR circuit to make a very effective modified NOR circuit. (See Fig. 10.) Transistor T_1 is the basic logic switch, and transistor T_2 is the constant voltage source. The power dissipation of this circuit is considerably less than that of Type A circuit.

The silicon diode placed in the forward direction between base and ground prevents the base from being overloaded when a plurality of inputs have "one" signals applied. This connection makes use of the high forward-voltage drop of most silicon diodes. This circuit has actually been constructed using micro-alloy diffused base transistors. The circuit has a simultaneous fan-in and fan-out of 25 with an average propagation time in the order of 80 μsec .

In actually testing circuits in various applications, evidence points out that the average propagation time is a more significant measurement of operation speed than rise, fall, and storage times considered individually. Average propagation time is measured by taking a series string of n logic circuits and applying a pulse to the first stage. The average propagation time is the time delay for the pulse to be propagated from the first to the

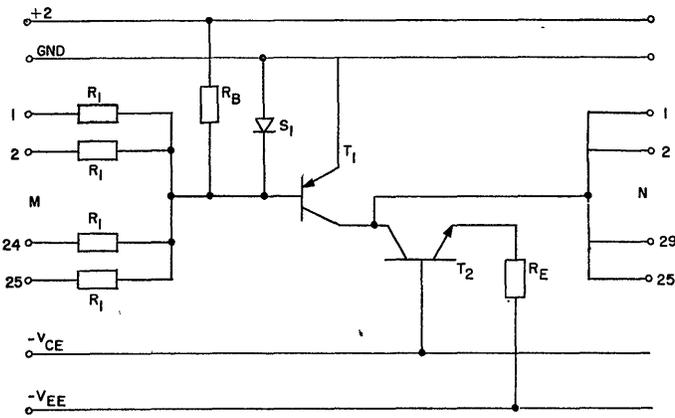


Fig. 10—Modified transistor NOR circuit—type B.

last stage divided by n , the number of logic circuits. In any particular NOR circuit the actual propagation time varies only a few per cent from the average, in the majority of cases. Since rise, fall, and delay times do not add directly to give actual operating time of compounded circuits, average propagation seems to be a more significant means of circuit operation time measurement.

APPLICATION OF PARALLEL-PARALLEL LOGIC TO HIGH-SPEED ADDITION

In order to illustrate the effectiveness of parallel-parallel logic, an application consisting of the circuitry for the carry operation of a high-speed adder will be shown in detail.

One of the major difficulties in designing high-speed adding devices is the drawback of ripple-through carry.¹ When making a binary addition, the most significant bit is dependent on the carry signal from the preceding stage, which is in turn dependent on the carry signal of its preceding stage, etc. This means that the most significant bit is dependent on the condition of the least significant and every other bit in order.

It has been shown^{1,5} that the expression for a carry signal from any particular stage k of a binary adder may be given as

$$C_K = A_K \cdot B_K + (A_K + B_K)C_{K-1}, \quad (7)$$

where C_K is the carry signal from stage K , C_{K-1} is the carry signal from the preceding stage, and A_K and B_K are the addends associated with bit K . If the substitutions

$$\begin{aligned} D_K &= A_K \cdot B_K \\ R_K &= A_K + B_K \end{aligned} \quad (8)$$

are used for convenience, the carry signals for binary adder of n bits may be expanded. Then

$$C_0 = C_0. \quad (9)$$

Generally, since there is no carry into the least significant stage, C_0 is 0.

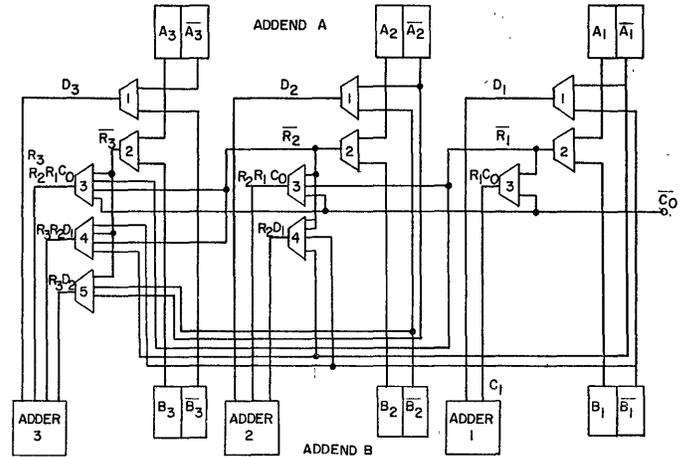


Fig. 11—Three-stage high-speed carry circuit.

$$C_1 = D_1 + R_1 C_0 = D_1 \quad (10)$$

$$C_2 = D_2 + R_2 D_1 + R_2 R_1 C_0 \quad (11)$$

$$C_3 = D_3 + R_3 D_2 + R_3 R_2 D_1 + R_3 R_2 R_1 C_0 \quad (12)$$

$$\begin{aligned} C_n &= D_n + R_n D_{n-1} + R_n R_{n-1} D_{n-2} + \dots \\ &\quad + R_n R_{n-1} \dots R_2 R_1 C_0. \end{aligned}$$

This shows that the carry for any bit n is immediately available from a single logic array for each bit. Furthermore, some of the logic expressions that are used in determining the carry of a particular bit are used in all the succeeding bits, so that much of the circuitry of each bit is necessarily repeated throughout. Actual circuitry for a three-bit carry circuit is shown in Fig. 11. The carry for each bit is arrived at after only two levels of logic circuitry, regardless of the bit position. Also, the adder of Fig. 6 may be used as the adder circuit, with the negation of the carry derived directly from similar circuitry. Total addition time is then exactly four propagation times or approximately 320 μsec , regardless of the number of bits in the adder.

For the basic carry circuit the number of logic circuits (L) is

$$L = \sum_{z=1}^n (X + 2) = \frac{n(n+5)}{2} = (1/2)n^2 + (5/2)n, \quad (14)$$

where n is the number of bits. The maximum number of inputs required by any logic circuit is $n+1$, and the maximum number of outputs is $\max k(n-k+1)$, which is $n(n+2)/4$ if n is even, and $(n+1)^2/4$ if n is odd, where k is bounded by $1 \leq k \leq n$. Then for the addition of two 20-bit words (without sign) a logic circuit with a fan-in of 21 and a fan-out of 110 is required. (The fan-out can be kept below 25 by using a logical design other than that given in Fig. 11. See Appendix II.) The carry circuit will require 250 NOR circuits. This compares with 80 logic circuits (fan-in fan-out of 3) for a particular twenty-bit ripple carry circuit in use in the NORDIC II computer (4 circuits per bit).⁴

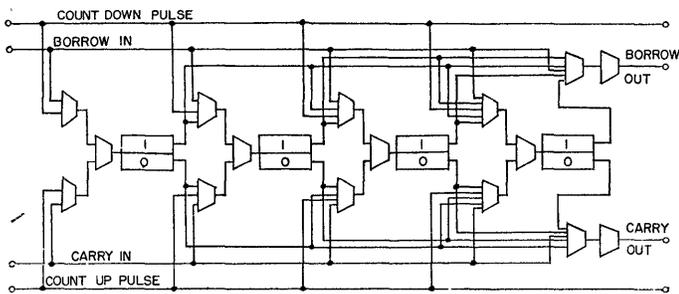


Fig. 12—Reversible binary counter.

Eq. (14) shows that by compromising on depth and thereby speed, the 20-bit carry circuit can be broken down into two 10-bit carry circuits, each requiring 75 logic circuits with a fan-in of 11 and a fan-out of 30. However, the depth is now four, half the speed of the 20-bit carry case. For a depth of eight, a total of 100 logic circuits are required, since four 5-bit carry circuits may be used. Only 6 inputs and 9 outputs are required for each level. This demonstrates the flexibility that is available by compromising on speed. An example of what can be done with only a limited fan-in and fan-out is the SEAC high-speed adder.⁵

APPLICATION OF PARALLEL-PARALLEL LOGIC TO COUNTING CIRCUITS

Some circuits, such as counter chains, do not fall directly into easy application of parallel-parallel logic. It is possible to treat such operation in a similar manner by making all counters and similar devices operated by parallel logical means, instead of direct sequential means. To illustrate this point, an ordinary binary counter has been examined to determine the logic involved in switching each stage by a count pulse. Then by considering each counter stage as a logical input, and determining from this the necessary changes in each counter stage for the next count pulse, we can effect a complete count cycle in only two propagation times, plus the switching time of a counter.

An example of such a binary counter is shown in Fig. 12 for 4 bits. It is reversible in that it can count up or down. The condition of the counter before a count pulse determines which counters should be changed when the count pulse appears. When all the counters have been changed, the new condition determines the change for the next succeeding count pulse, etc.

The limit of the size of a binary counter of this nature depends upon the maximum allowable fan-in, fan-out of the NOR circuits used. Thus for a 25-input-output NOR circuit, a 24-bit, simultaneous advance, binary counter is possible. It is also obvious that the counter code used is restricted only by the logic used. Therefore, a counter of any desired code is possible.

RACING AND TIMING CONDITIONS

When every input signal is required to travel in paths of equal length to the output (*i.e.*, all inputs extend through the same depth), the only conditions of racing that can occur are due to the variation of the propaga-

tion time of any logic circuit from the average propagation time. This condition is easily bypassed since parallel-parallel operation is exactly synchronous. This means that all output signals appear simultaneously after a predetermined time delay, when input conditions are changed. This time delay (taken for the deepest depth used) sets the minimum time between operations. False conditions due to racing occur only within this time period and disappear at the termination. All racing of this type is easily suppressed by gating the outputs so they do not read out during the circuit propagation time.

CONCLUSIONS

Parallel-parallel logic operation allows circuit operation at very high speed without the use of much higher speed components. The sacrifice for obtaining this operation is the number of logic circuits required and the topological and geometrical problems that arise from the complications of interwiring. Fortunately, these problems are not unduly severe for most actual applications.

The use of the modified NOR circuit allows operation with only one type of logic circuit with a large fan-in and fan-out. Furthermore, the logic is accomplished in the resistor gating and interwiring most of the time. This means that the actual number of transistors used is small compared to the amount of logic accomplished.

The attempt in this paper has been to outline briefly some of the possibilities that lie in this type of logic, along with its limitations. Although considerable work has been done in this area, it is too lengthy and detailed for inclusion here. Table I is a summary and comparison of the different types of logic and their applications, and simply expresses the basic ideas involved.

TABLE I

Serial	Parallel	Parallel-Parallel
Logic and function by sequence	Function is parallel Logic is sequence	Function is parallel Logic is parallel
Slow	Fast	Very fast
Minimum equipment redundancy	High equipment redundancy	High Equipment redundancy

APPENDIX I

LOGIC MANIPULATION OF THE FULL ADDER

$$\text{Half sum} = x\bar{y} + \bar{x}y = H. \quad (15)$$

$$\text{Full sum} = H\bar{C} + \bar{H}C = S, \quad (16)$$

where C is the carry from the preceding stage.

$$H = (\bar{x} + \bar{y})(x + y) = \bar{x}x + x\bar{y} + \bar{y}y + \bar{x}y = x\bar{y} + \bar{x}y \quad (17)$$

since $x\bar{x} = y\bar{y} = 0$.

$$S = (\bar{H} + \bar{C})(H + C). \quad (18)$$

By similar methods,

$$s = [\bar{x}\bar{y} + \bar{x}y + \bar{c}][x\bar{y} + \bar{x}y + c]. \quad (19)$$

By substituting (15) into (19),

$$s = [(\bar{x}\bar{y})(\bar{x}\bar{y}) + \bar{c}][x\bar{y} + \bar{x}y + c]. \quad (20)$$

Since $A + B = A \cdot B$,

$$s = [(\bar{x} + y)(x + \bar{y}) + \bar{c}][x\bar{y} + \bar{x}y + c]. \quad (21)$$

Since $\overline{A \cdot B} = \bar{A} + \bar{B}$,

$$s = [xy + \bar{x}\bar{y} + \bar{c}][x\bar{y} + \bar{x}y + c], \quad (22)$$

which gives

$$s = xyc + \bar{x}\bar{y}c + x\bar{y}\bar{c} + \bar{x}y\bar{c}, \quad (23)$$

since $x\bar{x} = y\bar{y} = 0$.

This is in AND-OR form. To get the more desirable (for NOR logic) OR-AND form,

$$s = \overline{(\bar{x} + \bar{y} + \bar{c})} + \overline{(x + y + \bar{c})} + \overline{(\bar{x} + y + c)} + \overline{(x + \bar{y} + c)}. \quad (24)$$

Since $\overline{A\bar{B}} = \bar{A} + \bar{B}$,

$$s = \overline{(\bar{x} + \bar{y} + \bar{c})(x + y + \bar{c})(\bar{x} + y + c)(x + \bar{y} + c)}, \quad (25)$$

$$S = (\bar{x}y + \bar{x}\bar{c} + x\bar{y} + \bar{y}\bar{c} + x\bar{c} + y\bar{c} + \bar{c})(\bar{x}\bar{y} + \bar{x}c + xy + yc + xc + \bar{y}c + c). \quad (26)$$

Since $A \cdot A = A$,

$$\bar{s} = \bar{x}yc + \bar{x}\bar{y}\bar{c} + x\bar{y}c + xy\bar{c}; \quad (27)$$

since $A + A = A$,

$$s = \overline{(\bar{x}yc)(\bar{x}\bar{y}\bar{c})(x\bar{y}c)(xy\bar{c})}; \quad (28)$$

since $\overline{A + B} = \bar{A} \cdot \bar{B}$,

$$s = \overline{(x + \bar{y} + \bar{c})(x + y + c)(\bar{x} + y + \bar{c})(\bar{x} + \bar{y} + c)}. \quad (29)$$

Eq. (29) is the final form.

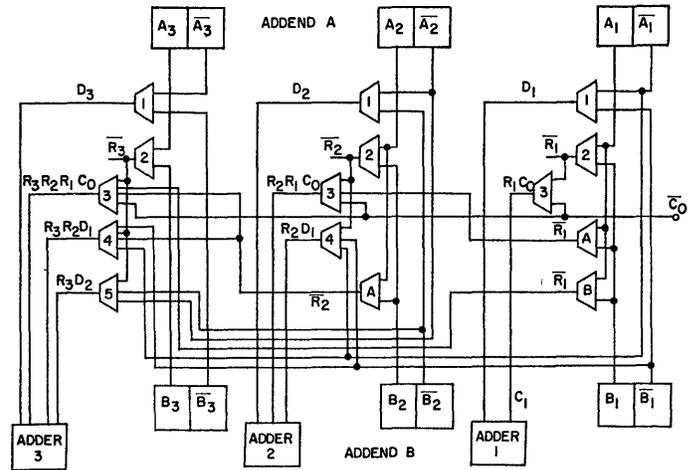


Fig. 13—Three-stage high-speed carry circuit, limited loading case.

APPENDIX II

In order to constrain the number of outputs to a single circuit for such a matrix as the simultaneous carry circuit to the limit of $n + l$, more circuits can be used to parallel the loading. This is indicated in Fig. 13 which shows the high speed for three bits with a maximum of $n + l$ inputs and outputs on each logic circuit.

The number of circuits required is now

$$L = n(n + 2) = n^2 + 2n,$$

which is certainly much larger than the previous case. This shows that an unlimited fan-out is desirable. For limited cases, this may be compromised by using parallel logic circuits. In either circumstance, the depth of levels remains unchanged.

ACKNOWLEDGMENT

The author is grateful for the efforts of Dr. T. A. Jeeves who initially suggested this approach and who has contributed to much of its development.

Information Retrieval Study*

ROBERT COCHRAN†

AS is well-known in the computer field, the basic scheme of information retrieval is to search a given set of data and extract those records which satisfy a certain set of criteria. While many schemes of information retrieval are being used, new techniques are always being considered and tested so that this very important task of computer technology can be perfected

relative to any given set of circumstances. The impetus to develop more effective methods for recalling and correlating recorded information originated in the field of science and technology. It was evident more than 10 years ago that considerable improvements in methods for using recorded knowledge were long overdue. The search still goes on—to study, to invent, and to perfect.

The utility of information retrieval is self-evident. As more and more records and documents come into being, the problem multiplies in complexity. The need for swift recall is paramount. For example, in medicine, the

* Based on work done for the U. S. Army Electronic Proving Ground, Ft. Huachuca, Ariz.

† Computer Applications Sec., Computer Dept., G.E. Co., Phoenix, Ariz.

ready access to the expanding record of laboratory and clinical observations is of decisive importance both to research and to professional practice. The advantages of basing decisions in business and Government on a broad range of coordinated data are well recognized. In the case of library and great document centers, the value of a collection of documents and other records, from simply a practical point of view, is determined by the benefits achieved in using the information contained in this collection.

One of the first entries into this vast field was the General Electric Psychological Matrix built for the armed forces during World War II. With this pseudo-computer, selections of qualified personnel could be made in accordance with a limited set of criteria. Since that time, progression to more complicated fields has been steady; the problem has been viewed more critically and with much greater sophistication. Essentially, it has come to be realized that the principal process of information retrieval requires that the material characterizing the subject content be inspected as to whether it matches the criteria characterizing the information requirement. In other words, from the viewpoint of logical theory, material selected as pertinent constitutes a class whose defining characteristics are determined by the information required. This process is not a blind search but the basis from which further searches may be made. In fact, to satisfy an information requirement, a multiplicity of searches may have to be performed.

Using this theory as a broad base, we may go on to specifics. The retrieval scheme to be discussed here encompassed three types of data to be searched using three magnetic tapes which had previously been written on a card-to-tape converter. Each of the three types of information were comprised of a master record and from one to nine trailer records. When a particular master record was extracted, its associated trailer records were also extracted.

In the first set of data, a master record was denoted by a zero in a particular card column. Trailer records were denoted by a number, one through nine, in the same column. In the second set, the denoting numbers were in a different card column. In the third set, no part of the record designated it as either master or trailer. Entries were limited to two records: first, a master; second, a trailer. A sense switch setting on the computer console was used to determine which tape to search.

The manner in which to specify criteria on which to base the master record search was the first problem to solved. This problem was complicated by the fact that each of the three sets of data had up to 20 fields which might have to be searched. These fields could be of varying length, and they could be continuous, interrupted, or overlapping. As an example, suppose digits 1 through 6 were a date group. Field A could be digits 1 and 2 for year; field B could be digits 3 and 4 for month; field C could be digits 5 and 6 for date of month; field D could be digits 1, 2, 3, and 4 for year and month; and field E

could be digits 1, 2, 3, 4, 5, and 6 for the entire date. To circumvent this problem, a decision was made to allow the criteria specification to designate the field also: fields would be defined and each field assigned a designating letter from A through T.

Let us now go to the query. Punched into a Hollerith card and read from the on-line card reader, this query should be of conjunctive normal form. That is, all criteria would be joined by the AND condition unless otherwise specified. All criteria joined by EITHER-OR would be at the end of a query. It would now be possible to have a query "A and B AND-EITHER C or D." A symbol for NOT and a symbol for BETWEEN-AND were included in the logic to enlarge the capability of the query. Thus, the query could now be written:

A and B AND NOT C AND BETWEEN D and E and EITHER F or G.

The letters *X*, *Y*, and *Z* were used to denote the three logic symbols and would precede the symbol for the field they would affect. Individually, the meaning assigned to these letters was: *X*=LIMITS; *Y*=NOT; and *Z*=EITHER-OR.

Since the criterion specified could be of variable length, the one remaining item to be resolved before it would be possible to write a query was the establishment of a means whereby the computer could recognize the end of a criterion. Since it is possible for a criterion to contain a character which would be the same as a field or logic symbol, the presence of such a symbol alone would not suffice. Therefore, a decision was made to let a slash mark (/) denote the end of a criterion and a comma (,) separate the lower and upper bounds of the limits.

In searching a tape, the first operation is reading the query from the decimal card and converting it to binary coded decimal form. This is the same form in which the information is written on magnetic tape. Then one record—the first master record to be interrogated—is read from the applicable magnetic tape. The program now begins a character-by-character analysis of the query.

If the first character of a query is a logic symbol, one of three flags is set so that these flags may later be interrogated and the correct action taken when needed. If it is a field symbol, it is saved, so that later an appropriate subroutine may be selected to isolate the descriptor from the tape record in the particular field under interrogation. After the field symbol has been detected and saved, the program examines the "limits" flag. If this flag has been set, the computer assembles character after character from the query until a comma is detected.

These assembled characters are then set up as the lower bound, limit A. Then, characters are assembled from the query until a slash mark is detected and set up as the upper bounds, limit B. If the limits flag has not been set, character after character is assembled from the query until a slash mark is detected. When this occurs,

the assembled characters are set up as the "criterion." Once either the limits or criterion has been set up, the next character or symbol from the query is examined and saved.

The first field symbol saved is examined and the appropriate digits from the tape record are isolated and set up as the "descriptor."

The program is now ready to begin the actual comparison. Again it is necessary to interrogate the limits flag to determine 1) if an exact match is desired or 2) if the descriptor is to be between two specified limits, numerically or alphabetically. If the descriptor is to be between two limits, it is first compared with the lower bounds. If it is less than the lower bounds, the NOT or negated condition flag is inspected. If the descriptor is not within the established bounds and the field is negated, a matched condition arises and control is transferred to the appropriate routine. A mismatched condition arises if the field is not negated, and again control is transferred to an appropriate routine.

The descriptor is compared with the upper bounds if it is equal to, or greater than, the lower bounds. If it is greater than the upper bounds it is not within the established limits. Therefore, the mismatched routine is used if the field is not negated; the matched routine is used if the field is negated. The descriptor is within the established limits if it is less than, or equal to, the upper bounds. Therefore, a matched routine is used if the field is not negated or a mismatched routine if the field is negated.

If the limits flag was not set we look for an exact comparison between the descriptor and the field criterion. If the descriptor agrees with the criterion, we must interrogate the negated condition flag. If the field was not negated, we have a match; if it was negated, we have a mismatch. If the descriptor did not agree with the criterion, we must also test the negated condition flag. This time, if the field was negated we have a mismatch and if the field was not negated we have a match.

After the comparison, control is transferred to either a matched routine or a mismatched routine whether the descriptor was to be between two limits or to agree exactly with the criterion. In the mismatched situation, the OR condition flag must be interrogated. If the OR

condition is not in effect, at least one item has been found which does not satisfy the query. This information is sufficient to eliminate this particular record.

Records are read from tape until another master record is found. Then all flags are reset to OFF and the query is processed as before. If the OR condition is set, it is possible that only one of a set of OR criteria has been mismatched. In view of this, it is now necessary to inspect the next symbol which has been saved. If this next symbol is a blank, the end of the query has been reached and the further possibility of a match falling under this OR is nil. It is then necessary to read tape records until another master record is encountered. Then the process is repeated. If this next symbol is a Z, or EITHER-OR symbol, again, there is no possibility of a match under the present OR; and this master record, together with its trailer, is skipped. If the next symbol is other than a blank or a Z, it indicates that the next criterion falls under this OR. Therefore, all of the flags except OR are reset and the next field is processed as outlined.

It is also necessary to examine the OR condition flag in the matched condition. If this flag is not set, one field criterion has been satisfied and the next field is examined. If an OR condition already exists, one of the set of OR criteria has been satisfied and no other criteria need be examined. The remainder of this EITHER-OR set is skipped until another EITHER-OR set of criteria is reached or the end of the query occurs. If another EITHER-OR set is met, it is processed in the same manner. If the end of query is reached, the record has satisfied all conditions for this query, and it, plus all its trailer records, are printed out. Another master record is then read and processed.

This method of inspecting the query and interrogating master records repeats for record after record until the end of a file of information is encountered. At this point, all the records have been examined and all those that satisfy the query have been printed. The tape is rewound and the program is now ready to process another query.

This retrieval system is atypical—its utility is manifold. As well as being a direct tool for retrieving past information and for solving immediate problems, the system is also used to plan for the future.

Communication Across Language Barriers*

W. F. WHITMORE†

I AM not sure whether my proposal is properly classed as a "blue sky" development; perhaps it's only on the edge of the stratosphere. What I am seeking is some means of communicating a fairly limited set of ideas rapidly and unambiguously across language barriers. It should treat all languages on an equal footing (if it does not evade the use of words entirely), and it might be advantageous if it did not demand literacy. I want to indicate here why I think such a device would be useful, and some possible methods for determining its characteristics. I am not an electronics specialist, and I don't propose to present circuit diagrams and the contents of black boxes. I have spent a good deal of my life writing operational requirements, and it is an operational requirement which I am going to present now. Once convinced of the need, I think computer experts can supply the device.

One of the most difficult aspects of the situation to understand is that such devices are not in heavy demand and are not already in existence. In some respects they do exist, of course: international flag signals for use by ships, the agreed code of highway warning signs in Europe, the Morse SOS and the voice call MAYDAY for distress, the reasonably universal symbols of mathematics and engineering. But, as far as computers go, the emphasis seems to be on translating literary or scientific texts, principally from Russian into English, with involved problems of shades of meaning and grammatical word order. This is admirable and difficult and challenging, and remarkable results are being achieved. But the machines involved are far from portable, and they don't work in real time. It should not take a high-speed digital computer to transmit a military order, control international air traffic at a busy airport, or order a meal in Southeast Asia.

The ultimate operational requirement was expressed to me once by a Marine colonel in the following terms: "Look, Doc, I want a walkie-talkie and a set of coils. When I've got a Greek regiment on my flank, I put in the 'Greek' coil. Then I talk English and he hears Greek." That expresses one form of the need admirably, and if that particular form of realization is possible, I should think the "blue sky" designation would be appropriate. However, something considerably short of the ideal would be most valuable. Another military example of the need: the *London Economist* reported that the first tactical order to the multilingual United Nations Force after arrival in Suez took four hours to

transmit. This would be a rather intolerable situation in the heat of battle.

With a stage set by these somewhat specific examples, let's consider in broader terms the situation which causes the need for communication across language barriers. It starts from the fact that, in regard to foreign languages, Americans are illiterate and provincial. "If those foreigners want to talk to us, let 'em learn English," regardless of the fact that in structure and spelling it is one of the more difficult languages to learn. This attitude did no great harm when the United States was isolated from the rest of the world, and when other nations mainly wanted things from us. But with the present resources of communication and transportation we are far from isolated, and we want (and desperately need) support from other nations in the cold war with a rival superstate. Precisely at this time, these other states are beginning to be aware of their own national entities (largely as a result of U. S. ideals of national self-determination), and are developing a feeling that their languages are just as good as ours. In a very real sense, the psychology of a nation is expressed through its language. The diplomatic consequences of American inability to speak foreign languages are evident in the daily papers, and everyone is aware of the stereotype of the American tourist who makes it obvious that inability to speak English is a sign of basic stupidity.

Another approach to this need is the universal language, such as Esperanto. This has been with us for decades without reaching any general acceptance, and I personally have strong doubts that it will do so even in the long run. The structure of proposed universal languages is in the Indo-European family of tongues, and there are a multitude of languages of importance to us, but outside this framework. Pei¹ gives a glimpse of some of them. As Whorf² has pointed out, some American Indian languages, such as Hopi, have a structure quite alien to the Indo-European grammar and word order. For example, the Hopi view of an event always includes both space and time, so that his language functions adequately without tenses for verbs. In any event, the present needs are too imperative to wait upon the slow progress of any of the universal languages. This is not to deny that the problem would largely vanish if a universal language were achieved. However, in the short term, we might hope for something nearer the situation of the written language in China. In all parts of China, the written ideographic language can be read, no matter

* The views expressed herein are those of the author and do not necessarily reflect the official views of the U. S. Navy.

† Special Projects Office, Bureau of Ordnance, USN, Washington, D. C.

¹ M. Pei, "The World's Chief Languages," S. F. Vanni, New York, N. Y.; 1946.

² B. L. Whorf, "Language, Thought, and Reality," J. B. Carroll, ed., Wiley-Technology Press, New York, N. Y., p. viii; 1956.

how widely the local dialects of the spoken language may vary. The sign language of the American Indians is another example.

I hope that you are convinced by now of the need for the proposed device. How should one set about getting it? Three or four years ago, when I first became interested in this question, I talked to some electronics people. None of them saw any great problem in constructing such a machine, providing they had a reasonable set of requirements from the user. So the first step would seem to be to collect a set of prospective users, and try to find out what their real needs for communication are, with a view to minimizing these requirements, rather than particularizing subtle shades of meaning. In a tactical military application, for example, it should be possible to assemble (perhaps at one of the War Colleges) a group of senior military officers with experience of field command, together with information theory specialists—maybe even advertising agents! This group would compile a vocabulary and grammar of the tactical orders and concepts which would need to be transmitted in situations involving the joint participation of U. S. and indigenous forces, with the requirement to keep these brief and unambiguous. Adequacy of the resulting lists might be tested with map exercises and war games involving foreign officers. The end result should be a body of tactical information of moderate size which is to be communicated across language barriers. A similar procedure might be used in other areas where the problem arises, such as airways control, ship handling, highway traffic regulation, and so on. I think basic English comes under my strictures against the universal language solution, but the underlying attempt to boil down English to a minimal structure which is still adequate for rudimentary communication is certainly a good precedent for the information groups to follow.

Once confronted with a manageable body of information to be processed, I feel sure the computer experts will be able to take it from there. The resulting equipment should certainly be portable and rugged enough to stand handling in the field. It would be nice to contemplate something as compact and personal as the present transistor pocket radios. For instance, is it possible to fit a device of this size with an indexed memory and some sort of transceiver so that a simple code transmission would cause the "radio" to repeat a particular selected command, *e.g.*, "Advance 100 yards at sunset," or "Mortar fire on Hill 209"? Or perhaps the memory

device will be so bulky that it will be an adjunct to field telephone installations, receiving a coded signal and playing a message in the appropriate language. Note that this compression of stereotyped messages is already employed, for example, by Western Union in its canned greeting messages for birthdays or Christmas.

The vocal communication may not be the best for all situations. In many cases, communication by sketches, maps, or conventional signals (flag hoists, again) may be better adapted to the particular problem at hand. In aircraft control, one can operate directly on the pilot's instruments. For example, in the integrated cockpit display system sponsored by the Office of Naval Research (ONR), one could indicate a desired destination directly on the navigational plot, or put maneuver signals on the contact analog display. Indeed, the ability shown in the ONR project to get completely free of preconceptions about knobs and dials would be a necessary requirement for the designers of the communications equipment. The obsession with the spoken or written word may, in fact, be an unnecessary handicap in approaching the problem. Certainly a miniaturized form of the facsimile transmission used to send weather maps would be a contribution to the basic problem.

It would be possible to develop at some length the advantages which might result from the type of device here proposed. Doubtless this audience can suggest many which have not occurred to me. But for the moment, it seems better to leave the problem fairly unadorned and let discussion bring out further points. It should be remarked in closing that the proposition is a "blue sky" one in an unexpected and frustrating sense—you have been presented with a suggestion for constructing a machine without having the basic design requirements formulated. I firmly believe that the essential first step is the conference of prospective users, and that the computer designers will have to wait in the wings until a specific body of information for communication has been proposed. So, in a sense, this paper is being given to the wrong audience. However, perhaps some of you have dual allegiance to requirements as well as to hardware; the rest of you probably have colleagues who should be prompted into the requirements type of activity. In any event, I hope there will be enough meat in this paper to persuade you to campaign for a set of requirements which will allow the computer experts to produce a useful and portable method for communicating across language barriers.

Symbolic Language Translation

EUGENE C. GLUESING†

INTRODUCTION

ALTHOUGH interest in the problems and development of techniques for machine translation of languages has been growing in the United States, no organized program comparable to that of the Russians has been developed. In that country, teams of specialists have been working in this area for several years. Announcement of a "breakthrough" in the field comparable to that of Sputnik in space travel is not to be unexpected from them.

The ideas expressed in the following report were first suggested by an article by Nagel.¹ Basically, they can be summarized, and at least partially justified, by the definition of language translation offered by Richens and Booth.²

"Taken in its most general sense, translation is the substitution of one language for another to express the same set of *ideas*. [The emphasis is mine.] It should proceed by a one-one substitution of symbols for each of the ideas expressed. . . ."

SYMBOLIC LANGUAGE TRANSLATION

The vagaries, anomalies, and ambiguities of language format and meaning, as well as widely variant linguistic structures have confused many literary translators; also, they offer many obstacles and complications to the adaptation of computers to language translation. For most practical translations of one language into another, it is necessary and sufficient that the meaning of the texts be conveyed from the reference language to the object language. Structure and harmony and beauty of phraseology need not be preserved provided this meaning is conveyed.

The intent of this paper is to indicate that through the use of an intermediate language, it is both possible and practical to apply electronic computers to the task of translating any existing language (or languages). The intermediate language proposed for use in translation consists basically of that of symbolic logic, with perhaps an extended set of symbols designed to simulate specific grammatical groupings or terms such as parts of speech, phrases, and possibly some idiomatic expressions, which are not traditionally represented in symbolic logic. The increase in the number of symbols

needed may entail the increase in machine bit representation of characters from six to seven (excluding the parity bit). This would give an additional 64 characters, deemed sufficient for the purpose. Alternatively, and perhaps more reasonably, would be the employment of combinations of presently existing machine symbols to fulfill the requirements of the additional symbolic notations.

In essence, the entire system may be compared in application to the pictogram type of symbolic characters of the Chinese language. The diagrammatic representation of objects or ideas there makes it frequently possible to conduct written discourse among some groups even though they cannot communicate orally.

THE PROBLEM DEFINED

The problem of language translation through symbolic conversion seems more and more to approach that of finding an appropriate symbolic vehicle, *i.e.*, a set of rules and expressions which will adequately convey the meaning of a statement from one language to another and at the same time lend itself quite readily to translation from language to the symbols and vice versa. Because of the complexity and variability of language structure, flexibility in the intermediate symbolic language is important.

For example:

$$1) @ ? [XR(XR'MW)]$$

That is to say, "How does one convince himself that he knows what a word means?"

$$2) (ALM)[T(P)(S \rightarrow M')] = 1$$

or, "In logic and mathematics the statement that the same statement always has the same meaning is largely true."

The above two statements,³ translated into symbolic form, are illustrative of both the problems and the probabilities of the use of symbolic notation in language translation. This translation was made by using the generally accepted symbols of symbolic logic supplemented by symbols supplied as needed by the writer. The use of any set of such symbols desired may be developed by the individual translator if the interpretation of the symbol is provided.

To illustrate, let us take statement 2) above and give the step-by-step translation of language to symbols to language, using English and German.

³ See [7], p. 71.

† Remington Rand Univac, St. Paul 16, Minn.

¹ E. Nagel, "Symbolic logic, haddock's eyes, and the walking dog ordinance," in "The World of Mathematics," Simon and Schuster, New York, N. Y., pp. 1878-1900; 1956.

² See [9], p. 25.

English	Symbols	German
In all logic and mathematics	(ALM)	Für alle Logik und Mathematik
the truth of a statement (P)	$T(P)$	die Wahrheit einer Festellung (P)
that a symbol implies a meaning	($S \rightarrow M'$)	dasz ein Symbol eine Bedeutung enthält
is near to 1.	$= \hat{1}$	ist nahezu Eins.

Since the symbols other than the conventional logic symbols should have an accompanying explanation, they should be supplied with each statement. Thus:

- L = "int" logic (Logik)
- M = "int" mathematics (Mathematik)
- S = "int" symbol (Symbol)
- M' = "int" meaning (Bedeutung)
- 1 = "int" nearly one (nahezu Eins).

The above statement designates how it is possible, given a constant meaning for a set of symbols, to translate from one language into another through a computer, given a set of bit configurations for each symbol and a dictionary of corresponding words and phrases, in any given language, for each symbol. The fact that the symbols can be interpreted in any language would make it possible, with limitations, to translate any language to any other through the symbols. For example, if the symbol, $T(P)$, always refers to the truth of a proposition or statement, going from the German, "die Wahrheit einer Festellung," to the symbol $T(P)$, it can then be matched with the corresponding phrase in French, Italian, Indian, Russian, English, or any other language.

TWO APPROACHES

One of the more significant questions to be resolved in the development of this system is: To what extent should the symbolism replace the language (*i.e.*, words)? Regardless of how the problem is handled, a large amount of storage would be needed—virtually enough to hold the vocabulary desired. Two relatively disparate approaches offer themselves:

- 1) To develop a large number of symbols (in essence a new, symbolic language) to represent words and/or phrases, or
- 2) To utilize a minimum number of symbols, enough to convey the meaning, with identifiers which are immediately translated within the construct of each statement.

The statement already discussed, 2) would fall into the first category, provided a separate library of symbols and meanings were instituted. An example of the second type of treatment would be the proverb:

$$(x)(x \text{ is in the hand}) \\ \rightarrow (x=y)(x \text{ is a bird})(y=2x \text{ in the bush}).$$

It is believed that for all practical purposes, the second method would be the faster and more efficient, in-

asmuch as less preparation, time, and effort would be required by the coder, and the alpha characters are already established in the computer.

It must be remembered that, regardless of which approach is used, the original translation, from language to symbols, must be manually (humanly) produced. The primary advantages would be derived from the fact that:

- 1) It is not necessary for the coder to know *any* foreign language in order to effect a translation.
- 2) Translation into as many foreign languages as exist in the libraries of the computer is equally possible, once the symbolic coding is completed.

A third approach to the translation problem would involve the possible combination of symbolic designation with the procedures involving grammatical and structural analysis often described. No attempt has been made thus far to explore this possibility.

SOME CONSIDERATIONS

To return to the first approach, namely that of developing a complete symbolic language, a few additional explanatory remarks include:

- 1) The necessity for establishing a large library of symbols would present considerable difficulty.
- 2) Certain well-defined rules of operation and identification procedures would require development. Examples of these:
 - a) Verbs could all be expressed as relations, *viz.*, R = seeing, testing, etc. The tense may be indicated by subscript notation, *i.e.*, R_1 = present, R_2 = past, R_3 = past progressive, etc. The progressive or participial form is used exclusively.
 - b) The subject will generally be designated (x).
 - c) Separate tables of symbols for prepositions, conjunctions, certain adverbs and other parts of speech.
 - d) Use of normal, logical symbolism wherever applicable.
 - e) Use of some method, such as over-or-underlining to designate various types of phrases and/or clauses. For example, a line under an element to indicate it is the object of a preposition.
 - f) Introductory phrases, parenthetical phrases, and words and phrases which do not have a definite function in the sense or meaning of the statement may be omitted in the symbolic form, unless they enter into the meaning.

- g) Where variant meanings of words exist, a subscript to designate the intended meaning could be employed. Take, for example, the word "fast":
 $F_1 = \text{"int" rapid}$
 $F_2 = \text{"int" motionless}$
 $F_3 = \text{"int" abstain from food}$
 $F_4 = \text{"int" having zest for living.}$
- h) Active and passive voice or verbs could be indicated by an arrow pointing left or right above R , indicating the direction of the relation:
 $\leftarrow R = \text{"int" passive } R$
 $\rightarrow R = \text{"int" active } R.$
- i) It may be better in many instances to translate idiomatic and colloquial phrases into conventional phraseology in order to convey the meaning, and thence into symbolic form.
- j) Generally, terms would be units of structure in the statement, and symbols would indicate relationships between terms.
- k) Grouping of an element with its modifiers is recommended.

It is anticipated that many other rules would develop as the system progresses. The ones listed above are merely suggestive; they are not to be construed as necessary or final.

Let us consider some further examples:

- 1) $(Ax)M'(RP \div R'W) \wedge (AS)(TUF) \leftrightarrow P$
 $x = \text{"int" Symbolic Logic}$
 $M' = \text{"int" meaning}$
 $R = \text{"int" attaching}$
 $\div = \text{"int" "but not"}$
 $\wedge = \text{"int" because, since, as, inasmuch as, for.}$

Here the symbol (\wedge) is used for the group of subordinating conjunctions similar to "because." The statement could be interpreted: "For all symbolic logic, the meaning has the relation of attaching (R) to a proposition (P), but not the relation of attaching to a word, because for all statements the statement is either true or false if and only if it is a proposition.

Illustrating the clarity of expression in a case where the English is not too clear, let us consider:

"The graduating boy did not know whether he preferred books and food, or money and clothes and pets, or a new automobile."

Symbolically:

$$(Ex)(xG) [x \div RR'(A \cap B)UC] \cap [(D \cap E) YF].$$

- | | |
|----------------------------|-----------------------------------|
| $A = \text{"int" books}$ | $F = \text{"int" new automobile}$ |
| $B = \text{"int" food}$ | $G = \text{"int" graduate}$ |
| $C = \text{"int" money}$ | $R = \text{"int" knowing}$ |
| $D = \text{"int" clothes}$ | $R' = \text{"int" preferring}$ |
| $E = \text{"int" pets}$ | $x = \text{"int" a boy.}$ |

Here the flexibility of the language is shown in that, if a running vocabulary is maintained for each statement, the corresponding terms can be translated to another language regardless of the symbols used.

Again, in the case of poetry, where in many cases meanings are obscure and subject to the whim of the interpreter, the following selection from "Music Hall Song"⁴ may be illustrative:

$$(Ex)(IR_2x)U(x \div R') \rightarrow (YR''?)(IR'''x). \text{ —!}$$

$x = \text{"int" donkey}$	$R' = \text{"int" going}$
$R_2 = \text{"int" having (past)}$	$R'' = \text{"int" thinking}$
$Y = \text{"int" you}$	$R''' = \text{"int" walloping.}$
$- = \text{"int" no}$	

Original:

"If I had a donkey wot wouldn't go
 D'ye think I'd wallop him? No, no, no!"

Two things are suggested here:

First, it may be a good idea to indicate verb mood with some symbol.

Second, it may be better to use numeric superscripts rather than prime designations to avoid awkwardness in sentences with many verbs.

In many cases, any attempt at literal translation would yield only failure. Such is the case in expressions involving puns, colloquialisms, modern jive talk, and idioms. For example, "Mud thrown is ground lost," would require rewriting to explain the meaning. The expression, "The team caught fire and won the game in the dying moments," would require some interpretation and revision, as would, "Dig that cra-a-a-zy cat!"

In summary, the effective translation of the meaning of language by machine involves:

- 1) The use of symbolic logic symbols as an intermediate language for transmitting the meanings of sentences.
- 2) An additional set of symbols to designate word groups, phrases, idioms, and other expressions not effected by the existing symbols of logic.
- 3) Either an extension of the bit structure of characters by one bit to include the additional symbols, or the combination of presently-used characters to form new symbols.
- 4) Dictionaries of corresponding terms in both the reference and target languages. These could be limited to the specific vocabularies required for the topic areas, such as science, mathematics, and electronics.
- 5) Trained and qualified individuals to translate the reference text into the intermediate symbols and to translate this symbolic language into the target language.

⁴ J. Beuler, in "The Oxford Dictionary of Quotations," Oxford University Press, Oxford, Eng., 2nd ed., p. 41; 1955.

CONCLUSION

It is not expected that the method outlined here will preserve the aesthetic values of poetry, the local color of colloquialisms and idioms, nor the fluidity of oratory. In many cases, it may even be difficult to preserve the meaning, as in the not-so-new example of the machine translation of the Biblical quotation, "The spirit is willing, but the flesh is weak," which came out in one Russian translation, "The whiskey is good, but the meat is rotten." A method which can be put into operation without too much time and difficulty is all that is intended.

The ideas presented here are neither complete nor fully developed. They are fragmentary and intended to stimulate further thinking and development. Entry into a new area of application is usually slow, incomplete, and often muddled and confused by human stupidity. Such attempts, however, are necessary even to the slow progress of fundamental understanding. Beginnings must be made somewhere, sometime. Perhaps this report will establish one such beginning.

BIBLIOGRAPHY

- [1] V. H. Yngve, "The translation of languages by machine," in "Proceedings of the Third London Symposium on Information Theory," Academic Press, New York, N. Y.; 1956.
- [2] S. N. Razumovskii, "On the question of automatizing the programming of problems of translation from one language into another," *Dokl. Akad. Nauk. USSR*, vol. 113, p. 760-761; 1957. Translation by M. D. Friedman, Lincoln Lab., Lexington, Mass.
- [3] I. K. Bel'skaya, L. N. Korolev, I. S. Mukhin, D. Yu. Panov, and S. N. Razumovskii, "Certain problems of automatic translation" (translation), U. S. Joint Publ. Res. Service, Washington, D. C.; April 1, 1958.
- [4] D. Yn. Panov, I. A. Mel'chuk, O. S. Kulagina, I. K. Bel'skaya, T. N. Moloshnaya, *et al.*, "Soviet experiments in machine translation" (translation), U. S. Joint Publ. Res. Service, Washington, D. C.; April 1, 1958.
- [5] "Abstracts of the Conference on Machine Translation," First Moscow State Pedagogical Inst. of Foreign Languages, Ministry of Higher Ed., USSR; May 15-21, 1958. Translation, U. S. Joint Publ. Res. Service, Washington, D. C.; July 22, 1958.
- [6] J. Lambeck, "The mathematics of sentence structure," *Amer. Math. Monthly*, pp. 154-170; March, 1958.
- [7] J. Whatmough, "Language, a Modern Synthesis," New American Library of World Literature, Inc., New York, N. Y.; 1946.
- [8] S. K. Langer, "Symbolic Logic," Dover Publications Inc., New York, N. Y.; 1953.
- [9] W. N. Locke and A. D. Booth, ed., "Machine Translation of Languages," M.I.T. Press and John Wiley and Sons, New York, N. Y.; 1955.

A Generalized Scanner for Pattern- and Character-Recognition Studies

W. H. HIGHLEYMAN† AND L. A. KAMENTSKY†

INTRODUCTION

IN the past 30 years, various ideas have been presented for machines which could recognize spatial patterns. With the advent of the digital computer and its use in data processing, there has been a great increase of interest in the automatic conversion of human language to language understandable by a machine. The translation to machine language of spatial symbols—pattern recognition—is important to this conversion.

A subclass of pattern recognition is character recognition, *i.e.*, the translation of a given set of handwritten or printed symbols to machine codes representing each symbol or a group of symbols. Many of the methods proposed for character recognition have not been evaluated critically. It is the purpose of the generalized scanner described in this paper to facilitate a systematic study of character-recognition techniques and an evaluation of methods prior to actual machine development.

The efficient study and evaluation of character-recognition methods requires the aid of a machine because of the quantity of data reduction involved. Such a machine might take one of two forms. For each recognition method a special-purpose machine may be constructed, tested, developed, modified, and perhaps reconstructed, until it either works or has proved unfeasible. The second type of machine is a general-purpose machine that is capable of simulating any recognition method, such that new methods and modifications of methods require only the generation of programs on paper.

One of the foremost problems in the development of a general-purpose character-recognition machine is that some methods require rather complex scans. These scans reduce the input information to a more efficient form. This problem of scan complexity might be solved by building a relatively simple scanner, such as a high-resolution raster scanner, to convert the spatial pattern to electrical signals. A general-purpose digital computer could then be used to translate this simple scan to the more complex scan. This same computer would then simulate the recognition logic of the particular system

† Bell Telephone Labs., Inc., Murray Hill, N. J.

under investigation, producing a recognition code corresponding to the simulated scan information.

Such a simulation method was found to be more expensive in terms of cost of equipment and required digital computer time than the machine which is the subject of this paper. This machine consists of a programmable scanner for simulating the desired scan, and a digital computer for simulating the recognition logic. In this case, no computer time is required for scan simulation.

This paper describes a programmable, or generalized, scanner that has been developed at the Bell Telephone Laboratories. In conjunction with the IBM 704 Electronic Data Processing Machine, the scanner is intended for use as a laboratory tool in the investigation of pattern- and character-recognition methods.

DESCRIPTION OF THE GENERALIZED SCANNER SYSTEM

The design of the generalized scanner system was motivated by several objectives including

- 1) Suitability for large data volume studies;
- 2) Suitability for studying different methods with reasonable scanner and computer times;
- 3) Suitability for scanning opaque forms containing handwriting as well as larger patterns;
- 4) Output in a form compatible with and efficient for the IBM 704 computer.

The generalized scanner is basically a flying-spot scanner with a scan configuration that can be programmed. An opaque pattern to be studied is illuminated by the cathode-ray tube light source, and the reflected light is detected and quantized into two intensity levels.

An output on magnetic tape is provided in a format compatible with the IBM 704, indicating the reflectivity and associated scan position of that part of the pattern being scanned.

METHOD OF SCANNING

The document to be studied is placed on a document carriage, and the desired pattern is positioned in the scanning area of the cathode-ray tube. The pattern is illuminated by the light spot from the cathode-ray tube. The spot moves in constant time (200 μ sec) from any given point on the pattern to any other point. The scan is established by specifying a sequence of coordinates of the light spot as a function of time (Fig. 1). Any one of 10,000 points in a 100 by 100 array can be specified as the end point of a scan segment. For each scan segment, the coordinates of such a point are specified; the beam moves from its present position to the specified point, following a straight line with uniform velocity. Here it will remain until the next point is specified. In this manner any scan can be approximated by a series of straight line segments. The 10,000-element resolution is sufficient to generate complicated scans, or give good resolution for rectilinear scans.

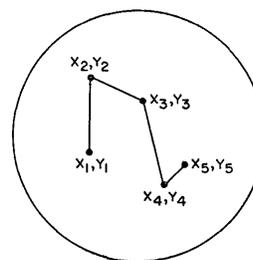


Fig. 1—Method of scanning.

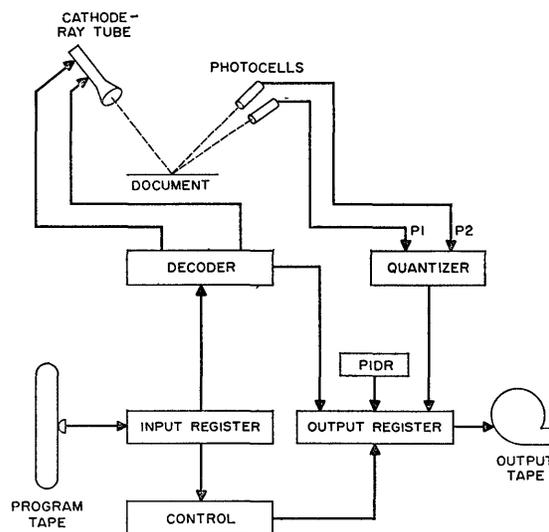


Fig. 2—Block diagram of generalized scanner.

GENERATION OF THE SCAN

Fig. 2 illustrates the major functional blocks of the generalized scanner system. The information necessary to control the scan pattern and the output format is contained on a continuous loop of magnetic tape. This program tape is produced by the IBM 704 by means of program-generation routines for each type of scan pattern. The generated program tape contains six bits plus a parity bit per line—the IBM 704 bit arrangement on tape.

Three lines of tape are assembled in the input register from the program tape to form an 18-bit instruction word. These 18 bits in the tape format shown in Fig. 3 are interpreted as one of five operations. These operations fall into two groups: scan operations and format operations. The scan operations specify a scan to a point (X, Y) determined by 14 bits of the total instruction. Two types of scan operations are available: 1) a scan with the beam unblanked (a "1" in the S position), and 2) a scan with the beam blanked (a "0" in the S position). The three format operations are start-of-program, write end-of-record (EOR), and write end-of-file (EOF). The start-of-program operation is represented by a unique 18-bit code that identifies the beginning of the program on the program tape loop. The end-of-record is used for the proper organization of output data for later use with the IBM 704; the end-of-file indicates

the end of the program (the scanning of one pattern has been completed). The end-of-records and end-of-files written on the output tape have the same tape representations as are used in the IBM 704. With an Ampex FL-100 tape machine, 60,000 instructions can be contained on a single tape loop, and instructions can be executed at a rate of 4000 per second.

The instruction contained in the input register is interpreted by the control unit. If it is a scan instruction, the 14-bit encoding of the coordinates is transferred to the decoder. A pair of 200- μ sec ramp currents are generated by the decoder (one for the X coordinate, one for the Y coordinate) and are used to drive the deflection yokes of the cathode-ray tube. The initial current of each ramp is proportional to the value of the originating coordinate; the final current is proportional to the value of the terminating coordinate of the scan segment. The set of ramp currents causes a scan to be produced in an image area that can be any size from 1/8 by 1/8 inch to 3 by 3 inches.

MARK DETECTION

Two opaque areas are actually scanned by the cathode-ray tube light source (Fig. 4). One contains the document under investigation; the other, a blank piece of paper. The reflections from the two areas are each focused on two photomultipliers, each sensitive to a different color. Hence, the scanner has a 2-color capability. The outputs of the photomultipliers for each color are differenced. This compensates for changes in CRT intensity, and for multiplier sensitivity as a function of beam position.

A DuMont Dual Opaque Flying-Spot Scanner has been modified for the scanning and pickup functions.

RECORDING OF THE SCAN INFORMATION

Before scanning a particular pattern, a code identifying that pattern is set into the toggle switch pattern identification register (PIDR, Fig. 2) by the machine operator. This code is the first piece of information written on the output tape (an Ampex FR 400) during the scanning of that pattern and serves to identify it in later processing by the IBM 704 computer. The scan information following the identification code includes the photomultiplier output information and the end-point coordinates for each scan segment using the tape format shown in Fig. 3.

A word length of 18 bits, or three tape lines, is written on the output tape for each scan segment. For a given scan segment, the output word contains the 14-scan address bits (X , Y) which were specified by the instruction word. This serves to identify the scan information with the corresponding scan segment. The outputs of the two photochannels are indicated by two other bits ($P1$, $P2$), a "1" being written if a mark of that particular color were crossed by the scan segment. A "1" in another bit (E) indicates an input parity error, and the last bit (S) identifies the type of scan (a reproduction of

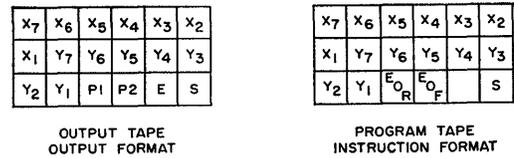


Fig. 3—Tape word formats.

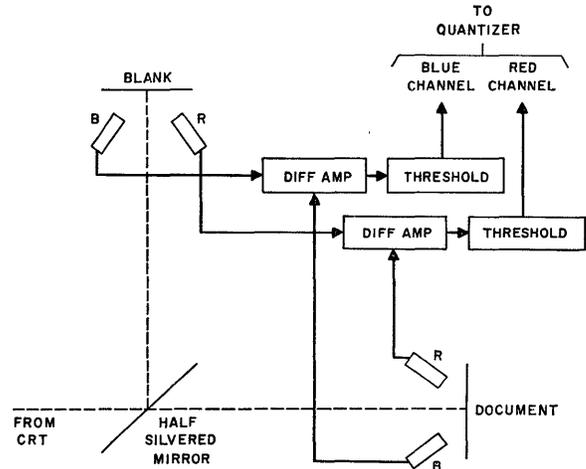


Fig. 4—Pickup system.

the corresponding bit in the instruction word). This information is all that is required to specify the results of the scanner operation to the computer.

The output tape may also be written in another form. In many cases, the photomultiplier outputs alone are sufficient for processing; hence, the third line only is required on the output tape. Although only two of these six bits are useful, writing only the third line is a more efficient input to the IBM 704 since the elimination of the other two lines will triple the input data rate to the IBM 704. (In many recognition procedures, the IBM 704 is input limited.) Therefore, the option exists in the scanner to write only the third line of the output word on the output tape. The output tape travels at one-third normal speed, thus giving the normal line density.

IBM 704 COMPATIBILITY

The program tape for the scanner is written by the IBM 704 computer on a reel of tape. This reel is then placed on the output tape machine, and the program information is transferred directly to the continuous loop tape machine under the control of the logic unit. Later, as the output tape is written, its timing is controlled by the program tape. That is, when a line enters the logic unit from the program tape, a line is also written on the output tape. Therefore, even though the speeds of the two scanner tape machines may differ from each other and from the IBM 704 tape machines, the line spacing on the output tape is, for all practical purposes, identical to that of the original program tape prepared by the IBM 704. By the same process, end-of-record and end-of-file gaps written on the input tape will appear as the same size gaps on the output tape.

Hence, the output tape is compatible with the IBM 704 computer.

PROGRAM EXAMPLE

An example of a simple program may clarify the operation of the generalized scanner. Fig. 5 illustrates the scanner's use in the study of recognizing numbers written with the double-dot system of constrained writing.¹ (A number written using two dots as a guide can be recognized by noting the crossings of the number with radial lines emanating from the dots.)

Assume that it is desired to program a set of radial lines in the form of a cross, and to move this cross through the field systematically. The two particular crosses that are centered on the constraining dots are to be used for the recognition procedure. One of these crosses is shown in Fig. 5(a). The first instruction (X_1, Y_1, B) positions the beam. (B stands for Blank; S stands for Scan.) The next instruction causes the beam to trace out one radial line, and so forth.

Let it be assumed that this particular scan pattern is positioned over a blue character written about red dots as shown in Fig. 5(b). Then the output tape will appear as shown (the first photochannel bit is the blue channel). Corresponding to the first instruction, $X_1, Y_1, 0, 0, B$ is written on the output tape (no outputs observed since this is a blanked scan). The next instruction will create a scan segment that crosses a blue mark, and this is so indicated in the output word. The next scan segment also crosses a blue line, and so on.

In interpreting these data, the computer will examine all of the crosses for a crossing of a red mark by the scan segment from X_6, Y_6 to X_7, Y_7 (the middle of the cross as shown in the figure) since this means that the cross is aligned with the dot. It will then use the blue crossings of that particular cross for the recognition procedure. Hence, in this case, the programmable recognition machine (scanner and computer) has been able to search for a character, to find it, and to identify it.

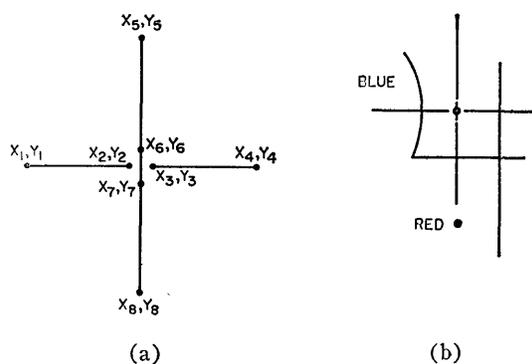
CONCLUSION

The generalized scanner is shown in Fig. 6. From left to right the units are: the power supply cabinet, the program tape machine, the output tape machine, the console, and the flying-spot scanner light source and pickup units.

The generalized scanner coupled with the IBM 704 computer is a powerful laboratory tool which is intended to greatly facilitate the investigation and evaluation of various character- and pattern-recognition methods. The most important characteristics of the scanner in review are

- 1) 100-line resolution (10,000 resolvable points);
- 2) A scan field variable from 1/8 by 1/8 inch to 3 by 3 inches;

¹ T. L. Dimond, "Devices for reading handwritten characters." *Proc. EJCC*, pp. 232-237; 1957.



PROGRAM TAPE		OUTPUT TAPE			
COORDINATE	S/B	COORDINATE	BLUE	RED	S/B
X_1, Y_1	B	X_1, Y_1	0	0	B
X_2, Y_2	S	X_2, Y_2	1	0	S
X_3, Y_3	B	X_3, Y_3	0	0	B
X_4, Y_4	S	X_4, Y_4	1	0	S
X_5, Y_5	B	X_5, Y_5	0	0	B
X_6, Y_6	S	X_6, Y_6	0	0	S
X_7, Y_7	S	X_7, Y_7	0	1	S
X_8, Y_8	S	X_8, Y_8	1	0	S

Fig. 5.

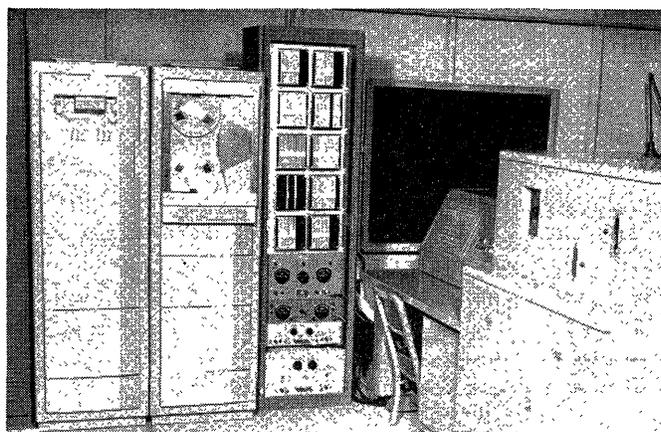


Fig. 6—The generalized scanner.

- 3) Two-color capability;
- 4) 60,000-instruction capacity;
- 5) Execution rate of 4000 instructions per second;
- 6) IBM 704 compatibility.

As experience is gained with the generalized scanner, it is planned to add a more flexible instruction code to allow the scanner to perform some of the functions now performed by the IBM 704. This will reduce or eliminate computer time for searching and for some data reduction operations. Additional instructions will allow scanner searching for characters and conditional transfers between scans. More information will be made available from the scanning process, and more efficient methods for writing on the output tape will be included. It must be stressed that this additional flexibility would serve only to reduce the amount of computer time required for simulation. The present generalized scanner coupled with a digital computer is sufficiently flexible to investigate any desired character-recognition method.

File Searching Using Variable Length Keys

RENE DE LA BRIANDAIS†

MANY computer applications require the storage of large amounts of information within the computer's memory where it will be readily available for reference and updating. Quite commonly, more storage space is required than is available in the computer's high-speed working memory. It is, therefore, a common practice to equip computers with magnetic tapes, disks, or drums, or a combination of these to provide additional storage. This additional storage is always slower in operation than the computer's working memory and therefore care must be taken when using it to avoid excessive operating time.

This paper discusses techniques for use in locating records stored within a low-speed memory medium where they are identifiable by a key word or words of variable length on a machine not equipped to accomplish this automatically. The technique is also applicable to the conversion of variable word length information into fixed length code words.

When records can be stored in a slower memory medium in such a fashion that their exact location may be determined from the nature of their designation, reasonably efficient handling procedures can be established. However, as is often the case, the records cannot be so easily located and it becomes necessary to examine each entry in order to locate a particular record. Sequential examination of the key words of each record, until the desired record is located, is not a satisfactory approach on machines not having automatic buffered searching facilities, and may not be satisfactory on machines so equipped, if, for instance, reels are searched which need not be because it is not known in advance that they are not needed. Because the average search time for the desired records is proportional to the number of records stored in this slower memory, the total operating time of a program is proportional to the product of the number of records stored and the number of records for which search is instigated. This product may approach the square of the number of records involved. This relationship between operating time and the number of records stored places a definite limitation on the number of records which may reasonably be stored by any particular program. Fortunately, if the records can be stored with the key words in some ordered arrangement, an educated guess can then be made as to the location of a particular record, and a better system will result. However, records cannot always be arranged in such a fashion.

When records are large compared to the key word or words, a useful technique is to form an index having in

it just the key words and the location on the corresponding record. A particular record is then located by searching the index to determine the record's location and then taking the most rapid approach in arriving at the record. Since only the key words and the locations of the corresponding records are stored in the index this technique reduces the amount of information which must be handled during a search. With the smaller amount of information involved it is often possible to utilize the computer's high-speed memory for the storage and searching of the index. Furthermore, this index can now be ordered or otherwise subjected to speed-up techniques. This index approach often can greatly improve the operating efficiency of record handling programs. In many instances this improvement is sufficient but there are also many cases where a further increase in efficiency is necessary. In particular, the time required to perform the search when consulting the index may still be objectionably large. If this is true then it is necessary to apply a speed-up technique to the searching operation. Of course these techniques can be applied to any table lookup problem where the nature of the key word or words does not lead directly to the desired entry.

Peterson¹ has suggested a method of arranging such an index which greatly reduces the lookup time when it can be applied. This method, referred to as the "bucket method," calls for randomizing the digits of the key word to produce a number which indicates that point in the available memory where a particular key and its corresponding record location should be stored. If this particular space is not available it is stored in the next highest (or lowest) available space. When seeking a particular record, the exact randomization process is repeated producing the same indicated point and a search is begun from that point in memory and in the previously used direction. When using this method, facility must be provided to continue from the other end when one limit of the available space is reached. During the process of placing an entry in this table a record is kept of the number of steps which must be taken before finding space to store the entry. This number is then compared with and, if necessary, replaces the previously occurring maximum. This maximum can then be used to limit the operation when a search is undertaken for an item for which there is no entry.

The object of this procedure is to distribute the records evenly throughout the available space in spite of uneven characteristics which occur because of similarities in the structure of the keys. A limitation of this

† U. S. Naval Ordnance Lab., Corona, Calif.

¹ W. W. Peterson, "Addressing for random-access storage," *IBM J. Res. Dev.*, vol. 1, pp. 130-146; April, 1957.

method is that it is necessary to store the key as a part of the entry in order to identify an item positively during the searching phase. This increases the storage space used and, as the memory fills up, the average number of spaces which must be examined before an entry is located increases. This saturation effect greatly decreases the operating efficiency. Furthermore, this method becomes far too involved from the bookkeeping standpoint when the keys are variable word-length words which exceed the length of one computer word when working with a fixed word-length machine.

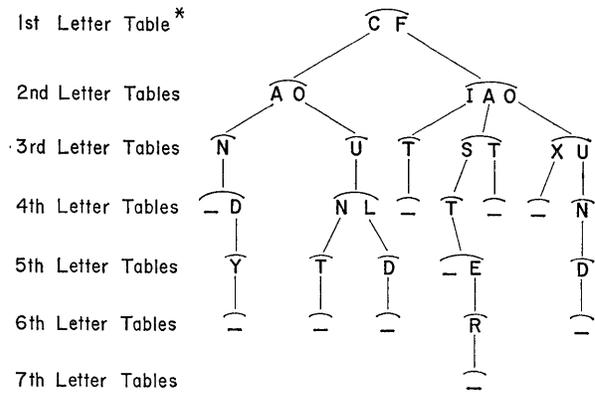
A problem involving variable word-length information confronted us in writing a FORTRAN type compiler for the Datatron 205. In this version of FORTRAN we allow the names of quantities to be of any length and they may consist of any number of separate words provided there are no intervening special characters. For reasons of simplicity in the internal handling, each of these external names must be converted to a code word of specific length such that it may be stored with other information within one cell. Further, it is necessary that the external name be preserved and made easily available for annotating the finished program. To accomplish this, each external word is placed in a file which, for future reference, is written on tape as it is formed. The position of each entry in this file is then used as the code word for the name. Knowing the position of a particular external word makes it very simple to recover for annotation purposes.

When it is necessary to have some method of determining whether any given word has occurred previously, sequentially scanning the previous entries is impractical because of the limitations mentioned previously. A more desirable situation would be a scheme that in no way depended upon the amount of previously stored information in the file. If this could be accomplished, the operating time would then be more nearly proportional to the number of items for which a search was performed rather than the product of this number and the total number of entries in the file.

The technique we have developed accomplishes the goal. The operating time is related to each letter of the external word and is, therefore, proportional to the number of letters in each word for which a search is performed. The size of the file has little effect on the operating time. Total operating time is proportional to the number of external words multiplied by the time required for a word of average length.

In our particular application each external name becomes a record which is stored on tape. The location of the first word of each record is the code for that external word. The external word itself is the key. In the computer's main (drum) memory we form an index of the key and its corresponding code. The organization of this index is the reason for this method's efficiency. We call this the "letter tables" method. The index consists of a set of tables with the number of tables as well as the number of entries in each table varying with

each running of the program. An address, usually the lowest numbered available cell, is assigned as the starting point of the table of first letters. The key words are examined letter by letter and each first letter which occurs is entered in the table of first letters if such an



*All entries of any one table are covered by a single arc (—).

Fig. 1—Formation of a set of tables.

entry does not already exist (Fig. 1). A new table is assigned to each of these letters. It is formed by assigning it a starting address. Each second letter which occurs is entered in the table assigned to the letter which it follows. To each of these second letter entries a new table is assigned as before. Each third letter is placed in the table assigned to the letter pair which it follows. Thus, there will be a table for the letters which follow "CA" and a different table for those letters which follow "CB" if these combinations occur. To each third letter a table is assigned and the technique is repeated until all letters of the key have been taken care of.

In our program the words which make up a key are compressed to eliminate blank spaces before being placed in the table as one word. A blank space is then used to signify the end of the word. This blank space is stored in the set of tables as a signal that the entry is complete and the code word which was previously determined is stored with this blank instead of an indication of a table assigned for the next letter. If preservation of the blanks is necessary, a special unique mark may be used to signal the end of a key.

When attempting to determine the previous occurrence of a particular word the procedure is first to scan the table of first letters until the desired letter is found. The second letter is then compared with the various entries in the assigned table until agreement is found for the second letter. The third letter is compared in a similar manner with the indicated table and the process is continued until comparison occurs with a blank. When this occurs the desired code may be found in the remainder of the entry with that blank. In the event that the desired letter does not occur in a particular list, it is known that this word is not in the index. If this word is to be added, it is now necessary to store the remaining

letters of the word in the index using the previously described technique. The first letter to be added will be the one which did not occur. Once a letter has been added to the tables there are no entries in the newly formed table so no further searching is necessary, and it is only necessary to add each letter remaining in the word to the new tables.

As previously mentioned, the code is stored with the blank which signifies the end of the word. This code is the next available location for a record in the external language file. As soon as the index is complete the external word, which is this next record, is placed in the file. The location indicator is adjusted to indicate the next available space in this file and this determines what the next code word will be.

Since the amount of space required for any of the tables in this type of operation depends upon the manner in which the letters happen to follow each other, it becomes necessary to assign space to each of the tables as it is needed. This is best done by assigning the next available space to whichever table is being expanded. The programming principles involved in this type of operation were first described by Newell and Shaw.² However, since in our application it is not necessary to remove entries from the tables as was the case in their application, a less involved method than the one they described can be applied. If a continuous portion of memory can be devoted to the storage of the tables it can be utilized in a sequential fashion with the next available space being the next cell. A simple counter can then be used to keep track of this next available space. This operation results in the storage of the various tables in an overlapping fashion and, therefore, it is necessary that each entry in a table have an indication of the location of the next entry in that table.

Sg	2-digit letter code	4-digit address of assigned table	4-digit address of next entry
----	---------------------	-----------------------------------	-------------------------------

(a)

Sg	0 0	CODE	4-digit address of next entry
----	-----	------	-------------------------------

(b)

Fig. 2—(a) Format of a letter entry. (b) Format of a word-terminating entry.

Fig. 2 shows the two types of entries in these tables. The letter entry shown in Fig. 2(a) has the letter in the two digits on the left end of the word, and the four digits on the right end are used for the address of the next entry in this table. The remaining four digits specify the address of the first word in the assigned table.

² A. Newell and J. C. Shaw, "Programming the logic theory machine," *Proc. WJCC*, pp. 230-240; February, 1957.

Fig. 2(b) shows the configuration used for the storage of a blank which signifies the end of a word. In this word, the first two digits are blank, the next four digits are the code, and as in the previous case, the last four digits indicate the address of the next entry. The end of a particular list is signified by the absence of an address indicating the next entry.

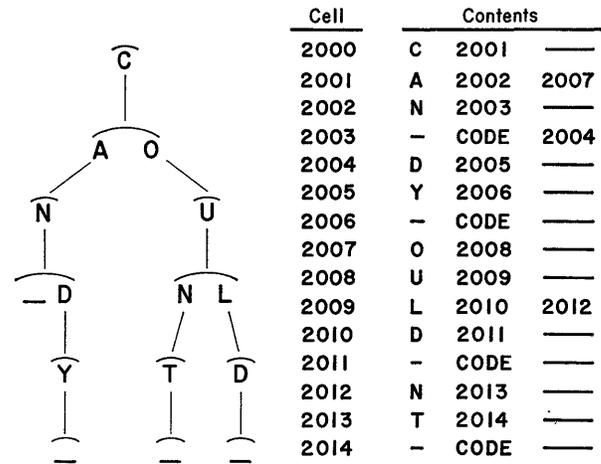


Fig. 3—Memory distribution of a set of tables.

Fig. 3 shows a memory distribution which would result from the storage of the four words: can, candy, count, and could. In this example there are a total of twelve separate tables stored in the fifteen spaces. Although the system may appear more complicated, no more bookkeeping is required than in a system of sequentially searching a list of entries where the different entries require different numbers of words in memory for their storage.

An additional time-saving feature that can be applied with a slight additional cost of memory space is the establishment of a full set of possible first letters with the corresponding second letter tables assigned starting points. By this we mean that if the first letter is "E" the first entry in the assigned second letter table will be in the fifth cell with respect to the beginning of the tables. Those first letters which do not occur are then wasting one memory word each.

Since in this scheme there is only one letter stored in each word, it requires far more space than other schemes where several letters are stored in one word. However, the advantage in scanning speed makes up for this disadvantage and it becomes practical to form several tables of this type, storing them in a slower memory medium until needed. When doing this, difficulty can arise if care is not taken to avoid excessive transfers to and from this slower memory. Methods of overcoming this problem depend upon the particular application. In our application each set of tables is no longer needed after one complete pass of the input, and when overflow of space occurs, during input, those words which cannot be converted are marked. After completion of the first

input pass the set of letter tables is erased and a second input pass begins at that point in the input data where overflow occurred. A new set of tables is formed to convert the marked words and the process can be repeated as often as necessary.

Now let us look more closely at the technique in an effort to determine the operating time. For the sake of the following discussion, we shall assume each character to be one of 40 possibilities. This list of possibilities could include the 26 letters of the alphabet, ten decimal digits, a blank, and three special characters. The maximum number of comparisons necessary to determine a word then comes to 40 for each character in the word including the blank which terminates the word. We find however that this maximum is seldom reached. To show this, let us assume a file contains 1000 words. If the first characters of these words are evenly distributed among the 40 possibilities there will be approximately 25 words starting with each character. Since 25 words can provide only five-eighths of the possible entries in the second letter tables we can expect that to determine a word, the average number of comparisons needed will be 20 to determine the first letter, 13 to determine the second letter, and thereafter only one per letter. Thus a nine-letter word including the blank might require an average of approximately 40 comparisons.

Now we increase the number of words to 10,000 and we find that we have an average of 250 words per character in the first letter table, $62\frac{1}{2}$ per character in the second letter tables, and approximately one and one-half in the third letter tables. The average number of comparisons for a nine-letter word now comes to 20 for each of the first three letters and one for each of the remaining letters. This new total of 66 is 1.65 times greater than the previous average.

When the words stored in such a fashion are taken from some formal language such as English, the number of words beginning with certain characters tends to increase and, therefore, the possibility of having all of the various characters occur in the table of first letters is decreased. This decreases the average number of comparisons needed to determine the first letter. Furthermore, the number of letters which normally might follow a particular letter is limited so that the average number of comparisons is reduced for subsequent letters also.

For example we normally expect "U" to follow "Q" and one of the vowels or the letters "H," "L," "R," or "Y" to follow the letter "C." Thus we find we are able to adjust favorably the averages we determined previously due to bunching, a phenomenon which usually leads to decreased efficiency in other methods. In the instance of the 10,000-word file we might expect the averages to be more like 16, 12, 8, 3, and one thereafter which would be 44 for the nine-letter word, an improvement of one-third.

We shall now attempt to compare this technique with Peterson's "bucket method." The bunching which we described as useful to us must be overcome when using the "bucket method." This is usually done by generating a number which is influenced by all characters of the word and yet appears to be random with respect to them. This is extremely difficult when dealing with variable word length information, especially with long words where only one letter differs or where the letters are the same but two have been interchanged. Other difficulties encountered with the "bucket method" include terminating the search when enough entries have been examined to know that the word is not in the file and then finding suitable space to insert the word. The resultant bookkeeping can actually consume many times more operating time than the actual comparison operation requires. Thus, although fewer comparisons may be required when using the "bucket method," due to the variable word length problems, the operating time is pushed up into the same range as that of the "letter tables method" which we have described.

Another way in which the two methods must be compared is with regard to the amount of memory required for the storage of similar amounts of information. In a fixed word length machine up to all but one character might be wasted with each word stored when using the "bucket method." Also, when a minimum of two adjacent cells are used, one for the word and one for the code, an occasional word will be lost due to storage of a word requiring an odd number of cells in such a position as to leave only one unused cell between itself and an adjacent entry. The amount of memory space required for the storage of a particular amount of information in the "letter tables method" cannot be specifically determined because it is quite dependent upon the number of repetitions of letter sequences which occur. The number of cells required will always be greater than the number of words and may in some instances approach the total number of characters stored. This means that the "letter tables method" will probably require from two to six times as much memory space as the "bucket method" for a similar amount of information.

The amount of code required by the "bucket method" may run three to five times as much as for the "letter tables method" depending on the application. Also, this latter method could quite probably be written as a subroutine or by a generator in a compiling routine with much greater ease than could the "bucket method." The final choice of method depends on details of the specific problem and also on operating characteristics of the machine on which it is to be run. It may in some cases be necessary to program and run tests before a final determination can be made. Both methods are an order of magnitude faster than the simple sequential search and we have found them both to be of value in different parts of the FORTRAN for Datatron Project.

Program Design to Achieve Maximum Utilization in a Real-Time Computing System

A. FREDERICK ROSENE†

IN THE DESIGN of a real-time computing system, a major problem is the selection of machine capacity and speed adequate to handle the maximum data-processing rates and yet not burden the system at other times with unused machine capability. All considerations—cost, efficiency, size, good design practice—require maximum utilization of the computer's complete capability. The design of the computer program to achieve maximum utilization is especially difficult in real-time systems which must be capable of handling a wide variation in data rate.

A real-time program design problem of this type was faced in the development of the Fire Control Center for the PLATO antimissile system. In addition to the requirement of handling a wide variation in traffic rate in real time, a Fire Control computing system must also be small and mobile. The program design which was developed illustrates one way in which a high-speed computer may be used efficiently for such a problem and is general enough to be applied to other real-time problems such as transportation or communications systems control.

The basic property of a real-time system is that certain commitments must be met at particular times no matter what the load on the system. The method of approach considered here uses as a design base a timing interval defined as the largest interval in which the essential outputs are required no more than once. The size of this interval, as a consequence of this definition, is determined by the external dynamic and physical environment, and is not dependent on the computer being used.

In addition to the essential set of outputs produced during a basic timing interval, there are also outputs, in most real-time systems, whose computation may be delayed to a later time. In the Fire Control application such tasks as predicting target position or selecting launch sites for a defensive missile could be delayed, whereas computing a command for a defensive missile which is airborne may not be delayed. In order to reduce computing time when an overload is present, two avenues of approach are available. (See Fig. 1.) First, the computation of the quantities which are not essential during the present interval can be delayed. However, before this is done, the expected load in the future, when

the output of this computation can be delayed no longer, will have to be considered. The second alternative would be a change in the method by which the essential outputs are formed; that is, a sacrifice of quality for quantity. The high speed at which computers work makes it impossible for human intervention to control the selection of what is processed and the manner in which it is processed. Therefore, it is necessary for the computer to determine what it should do and how it should do it by examining not only present conditions but also predicted future conditions.

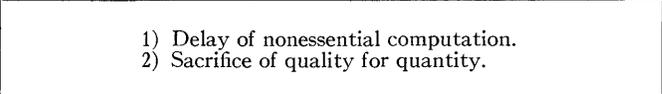
- 
- 1) Delay of nonessential computation.
 - 2) Sacrifice of quality for quantity.

Fig. 1—Methods for reducing computing time.

A program for providing this type of control for a large system would soon become a logical monster offering little flexibility of operation and an almost impossible debugging job unless some type of division of responsibility can be obtained. The basic principle of design which was applied is a division of the system into sections, each of which has a control program associated with it. These sections are connected and controlled by a central control program which imposes a given strategy on the system by the manner in which it connects the sections of the system. One outstanding advantage of this type of program design is the flexibility inherent in the control programs as well as in the operational programs. Also, this design enables several sections of the system to be programmed concurrently by different programmers without the worry of incompatible results.

In the design of the PLATO Fire Control Center, consideration was first given to defining levels of decision which provide a basis for proper division of the system into sections. This was followed by a detailed design of the different sections and their associated data-storage design. Of course, during this design period full consideration was given to possible orderings of simulation experiments for the efficient analysis of the system.

DIVISION OF THE SYSTEM BY DECISION LEVELS

The system recognizes three levels of decisions: those appearing in central control, subcontrol, and system significant blocks. (See Fig. 2.) The basic unit of the

† Missile Systems Lab., Sylvania Elec. Products, Inc., Waltham, Mass.

system is the system significant block in which all operational programs appear. Associated with each of these blocks is a subcontrol which controls its operations. The highest level, central control, controls the relationships between subcontrol-system significant block combinations.

The properties which a system significant block should satisfy are:

- 1) Any decision made in the system significant block must be a decision directly dependent on output within that system significant block, and must directly control the operation within that block.
- 2) Any functional change in the system significant block should not affect the rest of the system.
- 3) Any functional change in the rest of the system should not affect the system significant block.
- 4) Any section of the system significant block which satisfies the above three conditions must not be directly controlled by a timing decision (a decision determining how time is used).

Therefore, a system significant block becomes a black box provided with inputs which in turn generates outputs. The factor determining how the system is divided into system significant blocks is that no decision controlling operation of the rest of the system nor timing decision can be included within a block. Also, as the name implies, the system significant block should represent an operation which is meaningful to the system as

a whole. For example, a Fire Control Center which must interpret data from a radar and, on the basis of these data, launch and guide a defensive missile to interception of an enemy missile, might be divided into systems significant blocks in the following manner. (See Fig. 3.)

- 1) A block which detects the presence of a target by examining the data received from the radar.
- 2) A block which smooths the target data and predicts the position of the target.
- 3) A block which predicts the position of the defensive missile.
- 4) A block which computes commands which are sent to the defensive missile.

In any computing system, the juggling of the ordering of decision can make it possible for almost any block to satisfy the properties of a system significant block. Thus, one would begin the division of a system by determining the operations present which actually possess significance to the system as a whole. Then the blocks performing these operations can be designed according to the four properties already listed.

Each system significant block will have associated with it a subcontrol block which controls its operation. A subcontrol block is composed of decision operations which are directly dependent on the output of a central control and/or one or more subcontrol blocks, and the result of these decision operations directly control the operation of one system significant block. This intermediate level of control associated with each system significant block provides a division of control into subsections in the same manner as the system significant block division divided the operational program into more manageable subsections. The subcontrol block has the responsibility for making all decisions, time-dependent or otherwise, which affect the operation of its system significant block, that is, affect the method in which the block is used. By way of illustration, the subcontrol of a system significant block which predicts future target position might be required to determine if it has been allotted enough time to predict the position of all targets which have been detected by the best mathematical method. When there is not enough time, the

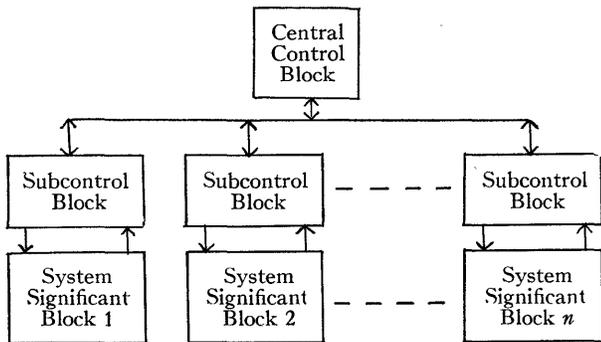


Fig. 2—Division of system by decision levels.

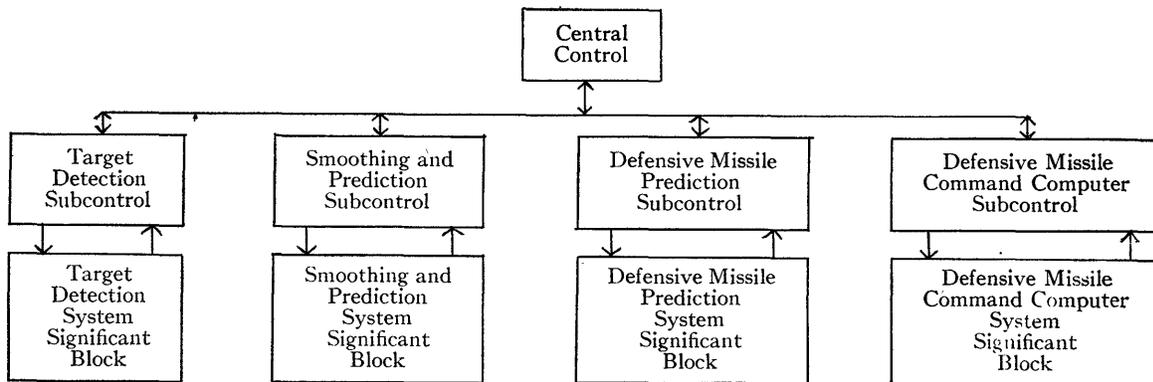


Fig. 3—Example of system significant block breakdown of a Fire Control Center.

positions of some targets will have to be predicted using a less accurate but shorter method; and the subcontrol will have to determine which targets should be processed using the shorter method.

The highest decision level is the central control which contains all decision operations directly concerning strategy. A decision at this level is directly dependent on the output of one or more subcontrol blocks and directly controls the operations of one or more subcontrol blocks. One important reason for separating decisions of this type from those appearing in the subcontrol blocks is that changes in strategy could be frequently required by a changing tactical situation.

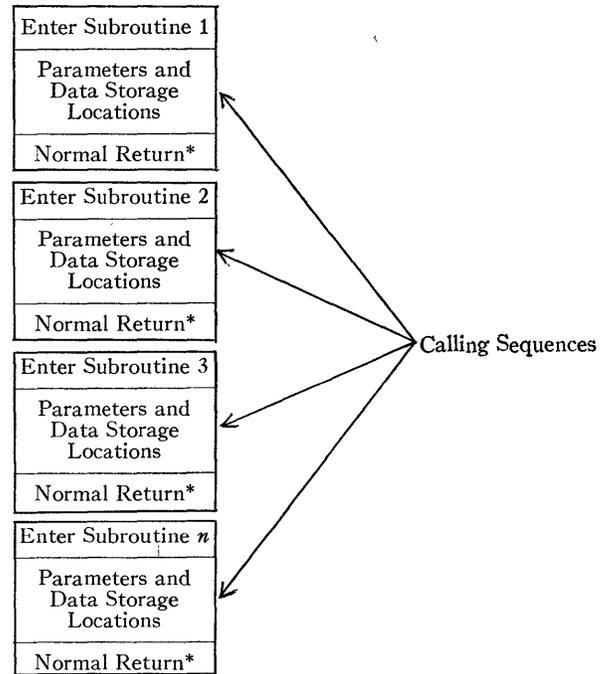
UNIT DESIGN

The properties already discussed, which are associated with each of the three levels of the system, set the bounds for the detailed design of each level.

The structure of the system significant block, which is the basic unit of the system, is a collection of calling sequences. (See Fig. 4.) These calling sequences may be considered as macro instructions from which a program is formed for carrying out a designated task. The normal return of each macro instruction contains an unconditional transfer instruction; the addresses of these transfer instructions are inserted by the subcontrol block of the system significant block. This insertion of addresses, accompanied by the storing or changing of parameters in the macro instructions where needed, is the only operation necessary to form a program. Hence, the subcontrol block effectively writes a program by inserting transfer addresses and parameters in macro instruction. However, in order to accomplish this the subcontrol must determine on the basis of data supplied by central control what tasks are waiting to be done, in what order they should be done, and how they should be done.

Provision must also be made for an overload of work, that is, a selective process for determining what tasks should be delayed until a later time. In order to do this, information such as present work load, time available for the subcontrol and system significant block combination being used, priorities of tasks waiting to be done, and any restrictions placed on waiting tasks must be available. (See Fig. 5.)

The first function of the subcontrol will be to select a task. This selection will be predominately based on a



* Normal returns are transfer instructions whose addresses are supplied by subcontrol.

Fig. 4—Structure of a system significant block.

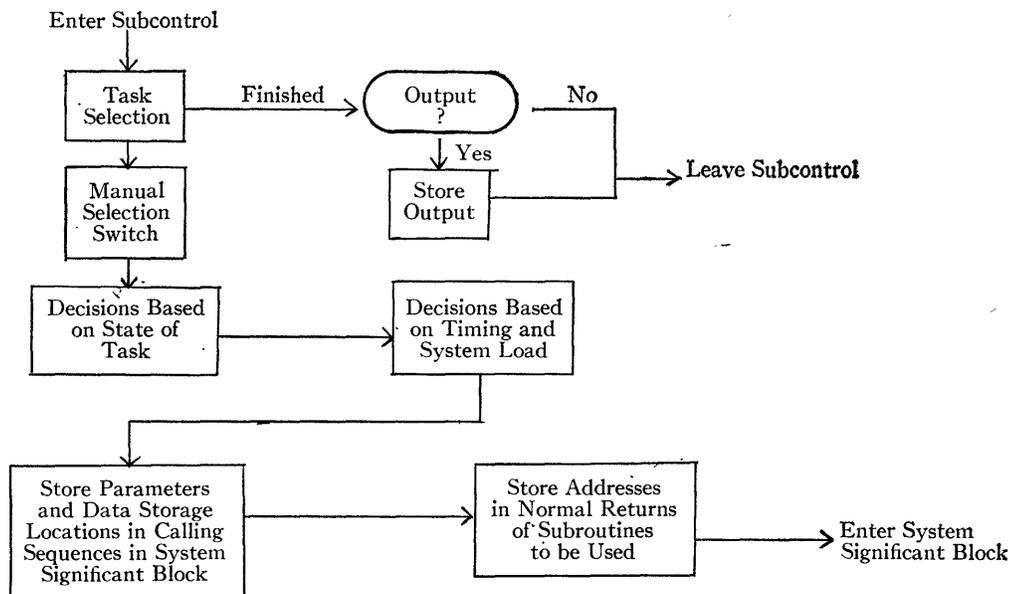


Fig. 5—Structure of a subcontrol block.

comparison of priority to time available; however, work load could also be a factor. Once a task has been selected, it is necessary to determine how this task should be carried out. There are actually two types of selections which must be made at this point: a selection of method based on the same quantities on which selections of tasks were based, and a selection which is completely independent of timing requirements but based on the state of a given task. The first type of selection is concerned with quantity vs quality. In other words, as demands on time increase, accuracy may be sacrificed in order to increase the number of outputs computed during a computational interval. This type of decision must be based on parameters computed by the central control since it is concerned with the strategy being employed. The second type of decision is concerned with how much has already been done on a certain task and on the basis of past results what should be done next. During initial simulations there will be the need for the analyst to select arbitrarily a given method he wishes to evaluate. For this purpose a manual selection switch not under control of the subcontrol is included.

One final duty of the subcontrol is to place all inputs and outputs of the system significant block on tape in the binary mode if they are desired for historical records and use in future analysis of system operation.

The central control block is only entered at the beginning of each basic timing interval. (See Fig. 6.) At this time, the subcontrol-system significant combinations which are to be used, the order in which they should be used, and the amount of time allotted to each, is determined. In order to accomplish this, the central control must be able to assess the work load expected during this and succeeding intervals, and the relative importance of each task to be performed by the system significant blocks. This type of information would be based on such quantities as probability of a task obtaining its objective, the size of the waiting line for a given task, time required by each system significant block, and necessary data rates allotted to a given task. It would also be the duty of central control to interpret external commands which have been supplied since the last interval. This would include information regarding functioning of different sections of the system, changes in

strategy, and instructions concerning the deletion of certain data.

Such things as error checks, system performance checks, and diagnostic programs will be handled by central control or at least controlled by central control. Signals indicating errors either in the computers or in other equipment such as data links will be interpreted by central control and the necessary action taken. This action would include such things as bypassing computations which cannot be handled with a given part of the system nonoperable, the switching-in of spare equipment, the calling-in of diagnostic programs to help locate the cause of an error, and the issuance of corrective instructions, if possible, where incorrect outputs have been given. System performance checks would indicate the way the system is operating and could be used to determine if an alternative strategy is required. Some examples of the types of decisions which central control may have to make are:

- 1) An external signal indicates that one launch site cannot be used. What action should be taken? The block which performs launch site selection would have to be notified; also, time must be allotted during the next few timing intervals for reassignment of launch sites for any interception already planned using the inoperative site.
- 2) There is a large number of targets requiring prediction and smoothing. How should the next time interval be used in order to reduce the size of this waiting line? During the next interval, as much time as possible would have to be allotted for the prediction system significant block. Hence, the other blocks would be allotted a minimum amount of time or bypass completely. The parameters sent to the prediction subcontrol would be such that all prediction would be done by the shortest method.

The exact structure of the central control cannot be specified in detail since the functions it must carry out are dependent on strategy. However, ideally, its function is to apply a strategy to the present conditions in order to produce a course of action which is consistent with immediate operational requirements and results in the least probability of system saturation.

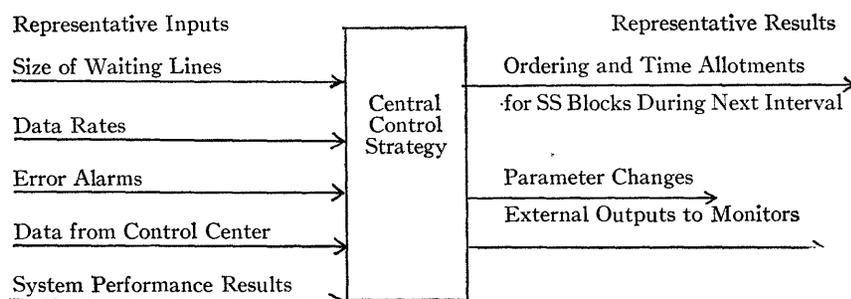


Fig. 6—Functions of central control.

SUBROUTINE DESIGN

The macro instructions which compose system significant block have been described as calling sequences to subroutines. (See Fig. 7.) These subroutines are divided into three levels according to their use in the system, and hence the manner in which they must be written. The calling sequences of those in the top level only appear in system significant blocks and usually in only one block. The function performed by this type of subroutine is usually system dependent; that is, the function could be done in many ways and simulation and analysis will determine which method is best for a given mode of operation. Calling sequences of subroutines from lower levels could appear within a subroutine at this level. The calling sequences of a second level subroutine could appear in a system significant block or in a top level subroutine. The functions performed by a second level subroutine are explicitly defined mathematical or logical tasks and their use does not directly depend on mode of operation of the system. This level subroutine can only contain calling sequences of subroutines from the lowest level.

All subroutines in the top two levels are written in closed form and the location of all inputs and outputs are specified in the calling sequence. Provision is made within each subroutine for storing all inputs and outputs on tape under the control of its calling sequence so that any data needed for analysis purposes are readily obtained if desired. One of the differences between these two levels of subroutines is the way in which they are written; that is, the second level programs are com-

DATA-STORAGE DESIGN

Another requirement of program design is the specification of the data storage. The data storage for a real-time system must be set up with the specific purposes of the system in mind and be compatible with the program design. Output requirements are one of the important considerations in designing data storage. System significant blocks and subroutines on the top two levels have provision for placing their inputs and outputs on tape. The data supplied by individual subroutines are necessary for analysis of the programs involved, whereas once the system is actually in operation, the output supplied by the subcontrol blocks is of prime importance. Since, in many cases, there is a minimum of time available for output in an operating system, the data-storage design should be such that the system significant block outputs require a minimum amount of computational time.

Even after a system is operating, it may be necessary to change or modify the strategy being employed. In order to expedite this type of change, separate storage for control data is advisable. In systems where random-access memory is limited, data not being used would be transferred to some auxiliary type storage such as drums. The data storage in this case would have to be designed with this transference of data in mind.

SIMULATION EXPERIMENTS

The first units of the system to be simulated are the subroutines. (See Fig. 8.) The purpose of this phase of simulation is to evaluate the mathematical models used

Level	Use	Form	Location of Calling Sequences	Subroutines Used by a Subroutine	Tape Output Provided	Error Returns
1	Dependent on Mode of System Operation	Specialized	SS	Levels 2 or 3	Yes	Yes
2	Not Dependent on Mode of System Operation	General	SS or Level 1	Level 3	Yes	Yes
3	Utility Programs	General	Levels 1 or 2	None	No	Yes

Fig. 7—Subroutine levels.

pletely general, whereas top level programs can be more specialized.

The third level of subroutines consists of utility programs, and the calling sequences of this type of program appear only within other subroutines of higher levels. All utility programs are written in a general closed form, and no provision for tape storage of inputs and outputs is provided.

Subroutines from all levels have error returns for overflow, division failure, and nonacceptable inputs.

Phase I	(a) Subroutine. (b) Groups of subroutines.
Phase II	(a) System significant block with all or part of its subcontrol. (b) Groups of system significant blocks each with its subcontrol and part of central control.
Phase III	Complete system including peripheral equipment.

Fig. 8—Ordering of simulation experiments.

over the range of inputs expected. In some cases, this simulation consists of connecting several subroutines together. The second phase consists of system significant block simulation accompanied by subcontrol control. During this phase, the different methods available for performing a function are assessed and the optimum selected. More than one method per function is kept in some cases because of time considerations, *e.g.*, time available vs accuracy needed. Sections of the control program used during this phase become part of the subcontrol of the blocks being simulated. The third phase, the entire simulation, is reached by successively combining system significant blocks and adding the appropriate sections of the central control. Adequate evaluation of programs at any level is impractical without realistic inputs which are most easily supplied by other subcontrol system significant block combinations and

central control; thus complete system simulation seems necessary. Also, a synthetic environment would have to be provided by peripheral subsystem simulation of such things as radar and missile.

CONCLUSION

For the PLATO antimissile system Fire Control Center, a program design was developed which embodies three decision levels: central control, subcontrol, and system significant block. This versatile program design provides for the maximum machine utilization, and permits a trade-off between quality and quantity of computation as well as expediting programming, debugging, and simulation.

The general concepts of this program-design approach should be applicable to other real-time computing systems in the business and scientific fields.

Pattern and Character Recognition Systems—Picture Processing by Nets of Neuron-Like Elements

L. A. KAMENSKY†

INTRODUCTION

SPATIAL pattern recognition, of which the recognition of alpha numeric characters is a subclass, is an important and practical problem. More efficient coding of transmitted pictorial information and more efficient utilization of humanly produced information could result from its solution. The problem of pattern recognition has been stated as the assignment of a meaningful code to a recognizable structure in a set of signals.¹ The signals, in this case organized spatially, are the result of a transformation from a visual picture field P to an electrical representation of this field. The points of this signal field S correspond to a characteristic of points in the picture. In this study, the reflectivity of given picture-point areas is quantized as black or white as a basis for a two-state electrical-signal representation of points of a pattern.

The total information content of this signal field is certainly less than that of the original picture. Useful pattern recognition requires that the code assigned to a pattern have even less information content than the signal field. For example, a code assigned to a pattern of a number in the set of 10-decimal-number symbols

indicates the number value but need not reflect the size, position, nor quality of the original pattern. The pattern-recognition machine is thus a device for performing an information-destructive transformation on the signal field to yield an output code assigned to the value of the pattern. Internally, the machine may perform a sequence of information-destructive transformations. Furthermore, the machine may internally produce transformations yielding parameters of the picture as an intermediate step, rather than the output code directly. Such parameters may have a physical meaning. For example, in useful number recognition the presence of the pattern parameters, straight lines, openings in curves lines, or corners is significant for generating the output code.

This paper describes an approach to the solution of pattern recognition that may be characterized as "spatial operations by neuron-like elements." Signal fields are transformed by a predetermined network of threshold-responsive elements. These elements have been called neuron-like in that they have many inputs and a single all-or-nothing output; they are connected in spatial arrays with excitation or inhibition gating between the signal field and the elements, or between the output and inputs of elements. More correctly, they should be called spatial neurons, since only their spatial properties approach the assumed properties of neurons.

† Bell Telephone Labs., Inc., Murray Hill, N. J.

¹ L. D. Harmon, "Computer simulation of pattern recognition," presented at Symp. on Pattern Recognition, Ann Arbor, Mich.; October 22, 1957.

(The coined name "speuron" will be used in this paper.) As will be shown, spatial gating only has been used and none of the complicated temporal properties of neurons have been simulated. However, speurons have really been applied for their usefulness in performing many different information-destructive transformations. They operate on all signal-field points simultaneously under the centralized control of a program. A model of a simplified speuron net was built. Included in this paper are illustrations, from the output of this model, of useful pattern filtering (noise reduction, width reduction, etc.) and pattern-parameter (straight lines, corners, etc.) extraction transformations.

CLASSIFICATION OF METHODS

Solutions of the pattern-recognition problem may be classified by the nature of the required information-destructive transformations. Remembering that the input to a pattern recognizer is the signal field defined in the first paragraph, and that the output is a code assigned to represent a class of patterns having a recognizable structure, let us develop the place of the neuron-like net.

Element Matching

The signal field is resolved into n independent elements and each input pattern is represented by an n bit code. Recognition is effected by matching an input code with codes representing the configurations of *independent elements* of each of the set of recognizable patterns. All possible patterns would be represented by a code table containing $\leq 2^n$ entries. Logical² or statistical techniques³ may be used to find the correct or best fit of the input and recognizable pattern fields.

Feature Matching

The individual elements of many patterns, however, are *not independent*, since recognizable patterns contain constraints on form. Parts of patterns can be classified in terms of independent groups of elements, instead of by individual elements. Some of these groups include the geometrical parameters, straight, curved, closed or opened, and breaks or corners. These may be independent of absolute position, size, noise, and some changes of form. We shall call these "features of the pattern." Recognition of patterns is possible if a sufficient set of relevant features can be extracted from the signal field.

Searching for Features

This group of pattern recognizers effects feature-extracting transformations by searching specific areas of the signal field. Edge tracing⁴ can be applied to fol-

lowing the contours of a pattern and changes in direction of a tracing head can be used as the features of the pattern. In recognition of double-dot writing,⁵ searches are made for written lines in seven specific areas of the signal field. In effect this searching system codes the feature, "Are partially- or fully-closed loops of black area present in this pattern?"

Feature Extraction by Spatial Transformations

The signal field is usually resolved into n independent elements. During a spatial transformation, the state of each element is examined. The state of this element and other elements whose coordinates are specified with respect to the examined one are functionally related by a specific rule to determine the transformed state of the examined element. There are many rules for transforming patterns.⁶⁻⁹ A study of Fig. 1 may help to clarify this. Three such transformations are shown here. Consider neighbors in this case as adjacent horizontal-vertical elements.

Although one investigator⁹ has proposed a parallel transformation system, all experimental studies of spatial transformations have been made using a digital

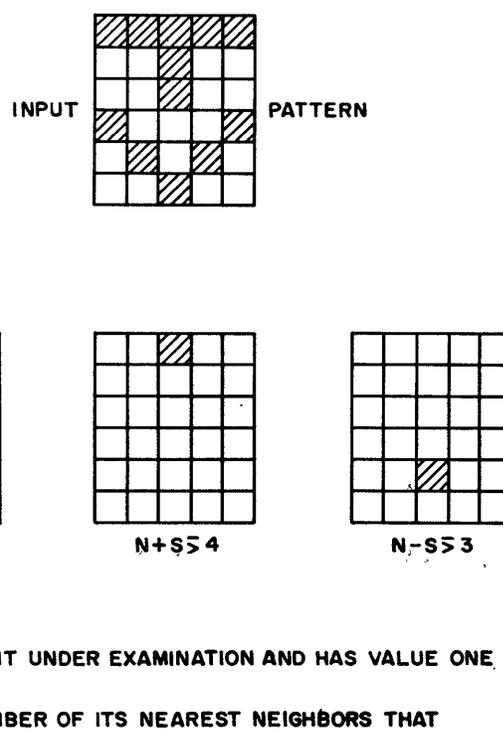


Fig. 1—Different transformations of the above pattern.

⁵ T. L. Dimond, "Devices for reading handwritten characters," *Proc. EJCC*, pp. 232-237; 1957.

⁶ O. G. Selfridge, "Pattern recognition and modern computers," *Proc. WJCC*, pp. 91-93; 1955.

⁷ G. P. Dinneen, "Programming pattern recognition," *Proc. WJCC*, pp. 94-100; 1955.

⁸ L. Cahn, R. A. Kirsch, L. C. Ray, and G. Urban, "Experiments in processing pictorial information with a digital computer," *Proc. EJCC*, pp. 221-229; 1957.

⁹ S. H. Unger, "A new type of computer oriented toward spatial problems," *Proc. WJCC*; 1958.

² "Electronic Reading Automation," The Solartron Electronics Group, Ltd.

³ H. T. Glantz, "On the recognition of information with a digital computer," *J. Assoc. Comp. Mach.*, vol. 4, pp. 178-188; 1957.

⁴ J. Loeb, "Communication theory of transmission of simple drawings," in "Communication Theory," Willis Jackson, ed., Butterworth Scientific Pubs., London, Eng., pp. 323-325; 1953.

computer. Each point must be examined sequentially until the complete field has been transformed. Although all spatial transformations to be described can be done on a digital computer, a relatively long time is required to relate spatial arrays of signals. As a result, parallel operations have been thought essential to performing useful pattern recognition.^{1,10}

Spatial Computers

Unger⁹ has proposed a program-controlled computer consisting of a spatial array of identical arithmetic modules which are connected to adjacent modules and to corresponding signal field points. Each module consists of a one-bit accumulator, about six bits of memory, and associated logic. The logical arithmetic operations this system could perform have been discussed by Unger.

The utility of serial computing has been questioned and has led to proposals for spatial computing. It is proposed that the utility of doing arithmetic operations at all be questioned as an answer to the pattern-recognition problem. Threshold-responsive elements (speurons) are proposed as one other answer, because:

- 1) They are simple; a model was constructed using a transistor per speuron.
- 2) They can be connected in different ways; each element can have many inputs; the associated logic between connections is simple.
- 3) They can transform by many different rules with a given connection by changing a common supply voltage.
- 4) They can be used to recognize patterns by successively transforming a signal field under program control.

It is demonstrated here that certain speuron operations will clean up patterns or extract certain features; that is, an output state or given sequence of states will appear, after a given sequence of transformations, only at points in the signal field where a certain feature is present. It is hoped that speuron subroutines will be found to uniquely extract sufficient sets of features of useful sets of patterns. One subroutine for each feature will be run sequentially on the machine. The presence or absence of a given set of relevant features will be the basis for recognizing patterns, like numbers or letters.

NEURON-LIKE ELEMENTS

The basic element of the neuron-like nets is shown in Fig. 2. The general elements N_j may have any number n of inputs X_{ij} each taking on the value 0, +1, or -1. All elements are controlled by setting a "threshold Z ." Z can take on the values 0 to n . The elements have a single output Y_j . Y_j is either 0 or 1 based on the criteria:

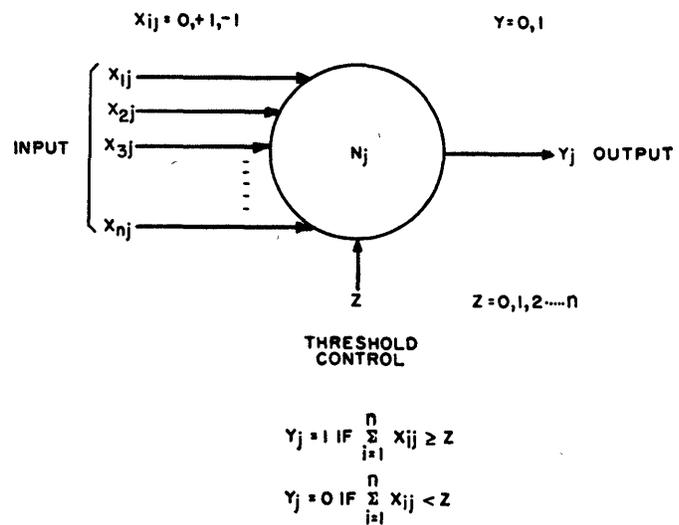


Fig. 2—The basic neuron-like element.

$$Y_j = 1 \quad \text{if} \quad \sum_{i=1}^n X_{ij} \geq Z$$

$$Y_j = 0 \quad \text{if} \quad \sum_{i=1}^n X_{ij} < Z.$$

In the general case, X_{ij} and Y_j are different for each element, but Z is the same for all elements of a given net.

Input to and Output from the Elements

Other logical elements must be connected to the basic element to provide for successive transformations and to increase the number of operations of a speuron net. A general form for the input and output of speurons is shown in Fig. 3. The inputs I_{ij} used to control gates can take on the value 0 or 1. The gated signals C_i , the control inputs to the speurons, can take on the values 0, +1, or -1. C_1 is not necessarily equal to C_2 or C_3 , etc.; however, all C_1 's, C_2 's, etc., all Z 's and flip-flop reset signals R are connected together. X_{ij} may be related to C_i and I_{ij} by the following table:

C_i	I_{ij}	X_{ij}
-	0	0
+1	+1	+1
-1	+1	-1
0	+1	0

NET CONNECTIONS

Two sets of net connections have been studied. In the first, called the nearest neighbor connection, a field point and neighbors surrounding it are connected to a corresponding speuron. In the second, called the directed connection, a field point and points along specific radii emanating from this point are connected to a corresponding speuron.

¹⁰ O. G. Selfridge, "Computers and pattern recognition," presented at meeting of Amer. Assoc. Advance Sci.; December 27, 1956.

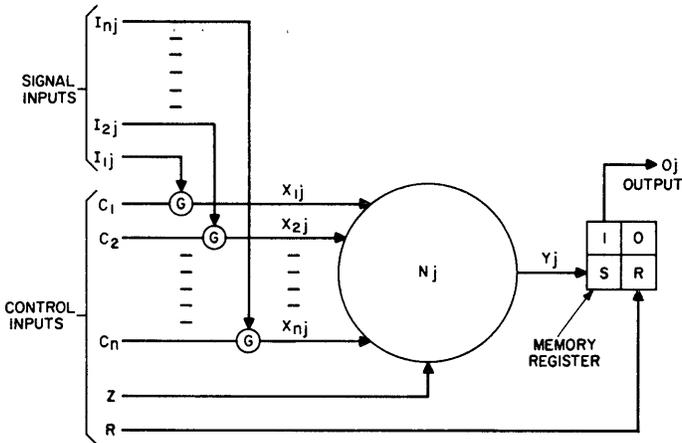


Fig. 3—General input and output of the elements.

Nearest Neighbors Connections

Each of nine inputs I_{ij} of a threshold-responsive element D_j is connected to a point in the signal field S_j corresponding to the position of D_j and to one of its eight nearest neighbors. This connectivity may be extended to next-nearest neighbors, etc. In a simplification of this connection, shown in Fig. 4, each of five inputs I_{ij} to speuron D_j is connected to S_j and to one of its four horizontal-vertical nearest neighbors. The latter connection was used in the model to be described. It should be pointed out that neighboring and corresponding input points add or subtract in a manner depending on the values of C_i . To effect successive transformations, the output of each element O_j is connected back to the corresponding input S_j through a gate which is opened for each transformation cycle. It is assumed that an input signal S_j can change to the same state as O_j after each transformation cycle.

Directed Connection

The inputs of an element D_j are connected to the corresponding signal point S_j and to signal points along radii emanating from S_j . A directed connection with horizontal-vertical radii is shown in Fig. 5. A range of influence and the number and the angles of the radii must be defined for each net. The directions of influence on all speurons are controlled by the voltages C_i . There is one common control for each radius and one control for the connection to the corresponding field point. Transformations can be generated which extract the feature, "Does a signal field point lie within a partially- or completely-closed loop of black area, or is it part of a black area?" Settings of Z will produce transformations which depend on the width of pattern lines or on the number of pattern lines in a given direction.

A SIMPLE MODEL

A 30-element model neuron-like net was built to study the feasibility of this device and to allow the au-

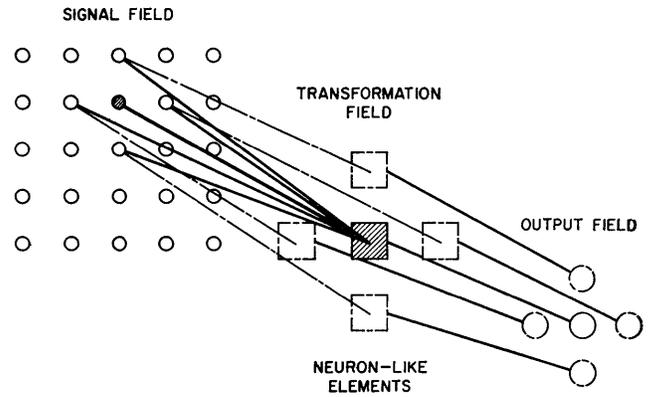


Fig. 4—Nearest neighbor connection.

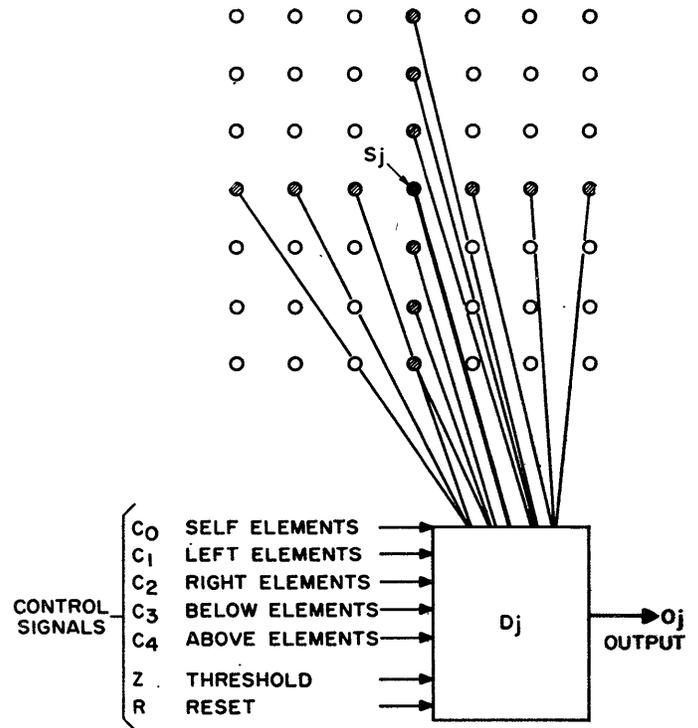


Fig. 5—Directed connection.

thor to experiment with different transformations. Although all transformations can be simulated on a computer, the use of the model enabled the transformations to be evaluated faster.

The model's input signal-field toggle switches, neuron-like elements, and lamp outputs were arranged in a six-by-five matrix. The input-toggle switches were each connected to four leads with phone-tip ends to allow any input to be connected to the input jacks of any speurons. Each element is a Kirchhoff adder driving a biased transistor. A circuit diagram of five elements and their input-output is shown in Fig. 6. These correspond to the simple N_j elements shown in Fig. 2, except for one input which can be controlled (as in Fig. 3) to be made positive, negative, or zero if the signal point connected to that input is one. This input was always connected to

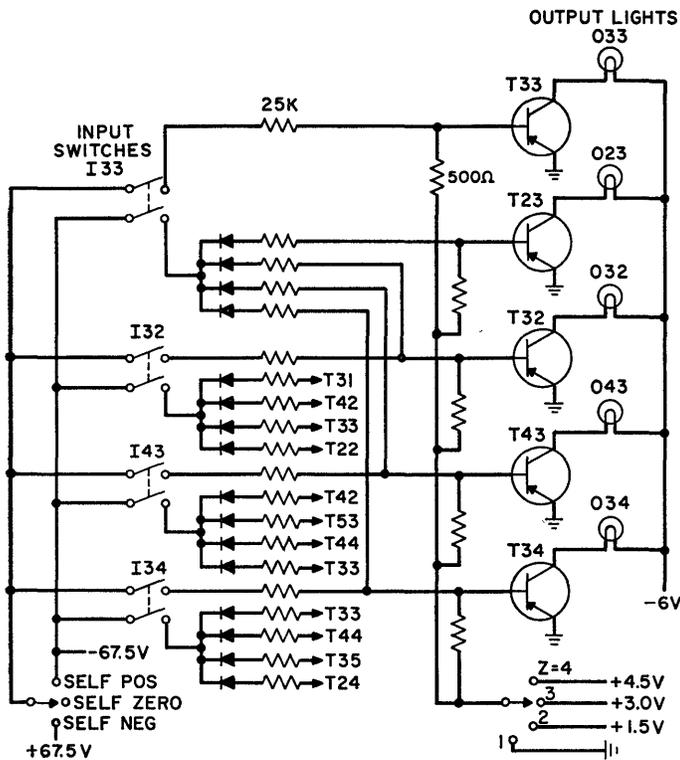


Fig. 6—Circuit diagram of 5 model elements with nearest neighbor connection.

the signal pattern position corresponding to the neuron-like element itself. This resulted in positive summing inputs from the four neighboring field points and adding, subtracting, or zero inputs from the corresponding or self-position. Four Z settings were built into the model. With the three self-input settings, 12 operations are possible. A lamp was used to indicate if the output pattern of each element was a one. Either the input or output pattern can be displayed on the lamp array. A photograph of the model is shown in Fig. 7.

Successive transformations were made by noting the pattern of output lights and setting up this pattern on the toggle-switch input array. The model was connected using both nearest neighbor and directed connections. The model successfully indicated useful nearest neighbor connection transformations. It was found that there were not enough elements to do anything useful with the directed connection. Transformations by the model are illustrated in the next section.

Model's Nearest Neighbor Transformations

Illustrations from the model's display of its 12 operations are shown in Fig. 8. To avoid confusion a different input pattern was used to illustrate each threshold setting. In the first column are the input patterns. In the second column are the patterns obtained with the positive self-connection; the third column shows zero self-connections; the fourth column illustrates negative self-

connections. The four threshold settings of $Z=1, 2, 3,$ and 4 are shown on rows 1, 2, 3, and 4. Note that the transformations in the upper left fill in patterns, while those in the lower right reduce patterns. If all of the inputs were controlled as in Fig. 3, the C_i controls would make the transformations directionally selective among 15 alternatives.

Some basic picture-processing operations are shown in Fig. 9. The results are self-explanatory. The values of Z are the threshold setting; the values of S are E for an adding or self-exciting connection, 0 for a self-zero or ignoring connection, and I for a subtracting or self-inhibiting connection. If additional controls C_i were used, pattern thinning and corner finding could be made directional operations. For example, only lower-left corners could be indicated on the output.

Fig. 10 illustrates operations on horizontal, vertical, and diagonal lines, vees and tees. Transformations are possible which reduce vees or tees to a single point. Some of the other patterns are interesting, especially the ones showing the invariance of the vee to the self-settings. The lower four photographs show the behavior of isolated lines to the transformation with $Z=2$ and $S=I$.

Successive transformations with $Z=2$ and $S=0$ are shown in Fig. 11. In one case, that of a nonsquare "L," successive transformations result in an alternation of two patterns after a number of transformations, the number depending on the pattern size. For a square "L," the resulting pattern is stable and a filled square.

CONCLUSION

A classification of the character-recognition problem was used to show the distinctions among the various methods. The neuron-like net method gives a large class of spatial operations, transforming all input pattern points simultaneously.

The neuron-like elements are relatively simple but capable of many operations. All neurons are connected alike to elements of an input presentation and are centrally controlled. Two distinct types of net connection have been studied. The first related neighboring points of the pattern; the second related points along given radii in the pattern.

A simple model has been constructed and shown to yield useful transformations which reduce irregularities and extract pattern features. Transformations obtainable by other types of elements and net connections have been indicated. A number-recognition method using one connection is currently being evaluated. The neuron-like net will be investigated more extensively by simulation on the IBM 704 with input patterns produced by a scanner.¹¹

¹¹ W. H. Highleyman and L. A. Kamensky, "A generalized scanner for character and pattern recognition studies," this issue, p. 294.

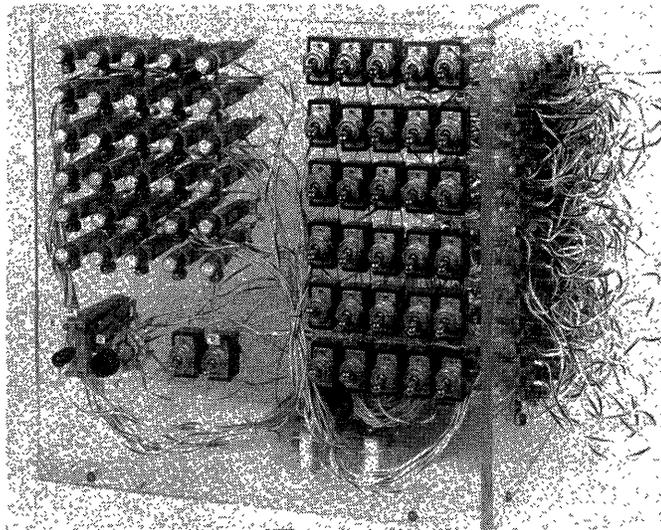


Fig. 7—Photograph of the model.

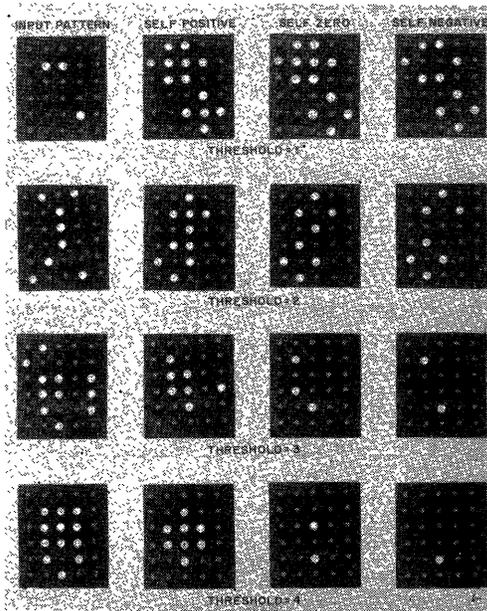


Fig. 8—The 12 basic operations of the model.

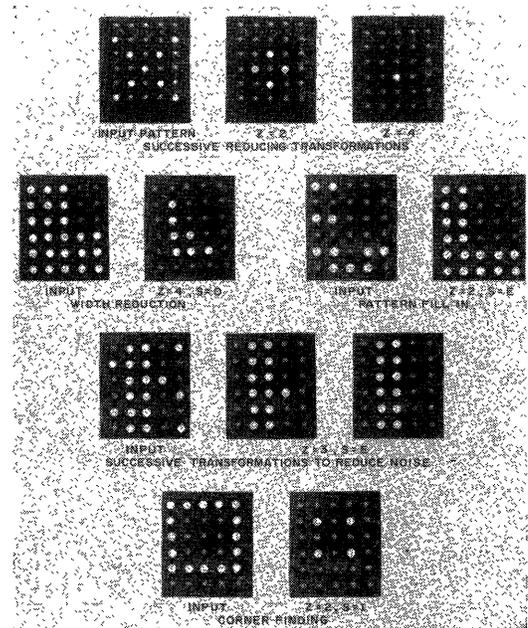


Fig. 9—Picture-processing operations.

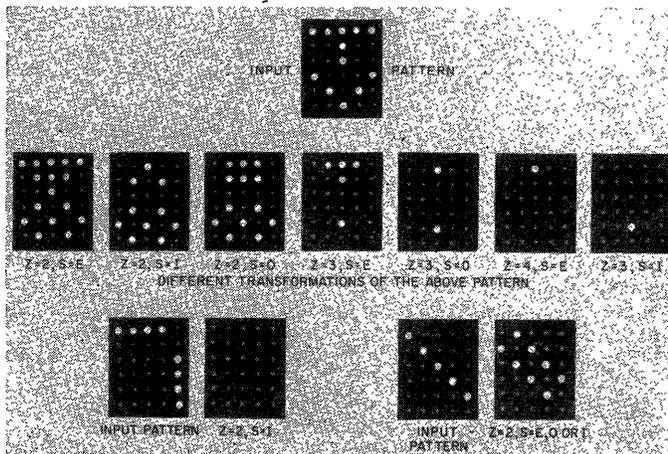


Fig. 10—Picture-processing operations.

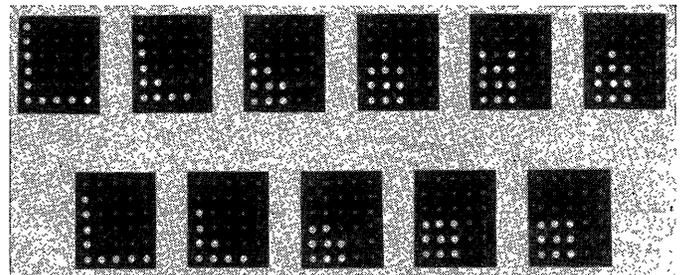


Fig. 11—Successive transformations of a nonsquare and square "L" with $Z=2$, $S=0$.

The Social Responsibility of Engineers and Scientists

F. B. WOOD†

INTRODUCTION

RECENTLY there has been some interest in the question of the social responsibility of engineers. A series of articles and letters to the editor appeared in the early part of 1958 in *Computers and Automation*¹⁻⁶ which dealt first with whether a journal such as *Computers and Automation* should publish articles on the social responsibility of computer scientists. Then specific topics such as the possibility of the destruction of civilization due to some component failure in the computer linked to a missile-warning radar network were treated. A series of viewpoints has been presented ranging from conscientious objection to working on a computer system that might be used for destructive purposes at one end of the scale, to a viewpoint of no concern with the use of one's work at the other end. My interpretation of these discussions is that people are arguing about the implied hypothesis: there is a danger to the existence of our civilization because social institutions have too long a time lag in making adjustments to utilize the latest technological advances wisely.

This apparently sudden interest in the social responsibility of computer scientists was preceded by a long and fluctuating development of concern for social responsibility in science and engineering. Meier has reviewed the status of social consequences of scientific discovery and has made specific recommendations concerning the social responsibility of administrative scientists.⁷ Layton has studied the history of the idea of social responsibility in the American engineering profession.⁸ Rothstein has discussed some of the deeper philosophical aspects of these problems in his book.⁹ The Western Joint Com-

puter Conference at Los Angeles, Calif., May 6, 1958, conducted a panel on "The Social Problems of Automation."¹⁰

The various viewpoints appearing in *Computers and Automation* present an uncoordinated distribution of differing ideas. The views of the 1958 WJCC panel have a certain amount of coherence. It would be desirable to find a straightforward way for an individual or engineer to determine his responsibilities in this area. The ideas which I am about to develop are hypotheses brought forward for the purpose of obtaining discussion on this important subject. At this stage, they represent my own personal views and are not to be construed as representing a policy of my employer. I would have preferred to have this paper follow a historical analysis of this problem of the social responsibility of engineers so that I could be sure that I am not repeating the same mistakes made in previous periods of interest in the subject. Perhaps by next year we will have a sounder base to operate from in discussing the subject of the social responsibility of engineers.

DISTINCTION BETWEEN THE SOCIAL RESPONSIBILITY OF CITIZENS IN GENERAL AND THAT OF SPECIALISTS

In a democratic nation such as the United States, all citizens have a responsibility to keep aware of the major problems of our country. This is necessary to be prepared to make wise decisions in electing public officials and in voting on basic policies. Specialists such as engineers and scientists of course share this basic responsibility with all citizens. I maintain that specialists have an additional responsibility beyond that of the citizen because of their special knowledge which is not readily accessible to the layman.

WHAT SOCIAL RESPONSIBILITIES DO ENGINEERS AND SCIENTISTS HAVE?

In the long run, technology is undoubtedly making changes in the organization of our society. We cannot expect engineers and physical scientists to become sociologists. However, we can expect engineers to ask questions and urge that appropriate social scientists study the social problems related to their work. Each scientist or engineer can ask himself where his own specialty fits in the development of devices or new knowledge which may affect social organization. Then he can speculate as to what problems might come up in the future due to the application of his work.

† IBM Corp., San Jose, Calif.

¹ Readers and Editor's Forum, "Curse or blessing?" *Computers and Automation*, vol. 7, pp. 9-10; January, 1958.

² E. C. Berkeley, "Cooperation in horror," *Computers and Automation*, vol. 7, p. 3; February, 1958.

³ A. A. Burke (I), W. H. Pickering (II), and Editor (III), "Destruction of civilized existence by automatic computing controls," *Computers and Automation*, vol. 7, pp. 13-14; March, 1958.

L. Sutro, "Comments on 'Destruction of civilized existence by automatic computing controls,'" vol. 7, pp. 6, 31; May, 1958.

⁴ Editor (I, III) and Readers (II), "The social responsibility of computer scientists," *Computers and Automation*, vol. 7, pp. 6, 9; April, 1958.

⁵ "Ballot on discussion of social responsibility of computer scientists," *Computers and Automation*, vol. 7, p. 6; May, 1958.

Later results, vol. 7, p. 6; July, 1958.

⁶ N. Macdonald, "An attempt to apply logic and common sense to the social responsibility of computer scientists," *Computers and Automation*, vol. 7, pp. 22-29; May, 1958.

Discussion: "Locks for front doors," vol. 7, p. 24; August, 1958.

⁷ R. L. Meier, "Analysis of the social consequences of scientific discovery," *Amer. J. Phys.*, vol. 25, pp. 609-613; December, 1957.

⁸ E. Layton, "The American engineering profession and the idea of social responsibility," Ph.D. dissertation, Univ. of Calif. at Los Angeles; December, 1956.

⁹ J. Rothstein, "Communication, Organization and Science," The Falcon's Wing Press, Indian Hills, Colo.; 1958.

¹⁰ H. T. Larson (chairman), H. D. Lasswell, B. J. Shafer, and C. C. Hurd, "The social problems of automation," panel discussion, *Proc. WJCC*, pp. 7-16; May, 1958. (AIEE Publication T-107.)

This may be a further stage in the development of the last decade in which it has become popular to use human factors engineering studies directed by psychologists to determine if proposed electromechanical devices requiring human reading or manipulation are consistent with the way human beings function. As our industrial society becomes more complex, it may be necessary to extend this concept to "social factors" studies where the engineer calls in sociologists to investigate the social effects of applying his new knowledge or devices.

At this stage the engineer's responsibility may be to see if there is someone or some group studying these problems, and if there is not, he can recommend to the appropriate agency that such a project be undertaken. In this way the engineer can shorten the time lag between the introduction of a new technology and the appreciation of its social consequences. I have noticed that even specialists sometimes fail to recognize the division point between their domain and that of other specialists. The important thing here is to obtain the advice of the appropriate specialists, instead of just relying upon our own ideas and feelings.

A CHECKING CHART TO AID THE ENGINEER IN DEVELOPING SOCIAL RESPONSIBILITY

Let us construct a chart to outline the factors involved in determining what an engineer's social responsibility should be. Such a chart is shown in Fig. 1. Starting in the lower left-hand corner, there is a box to write in the "Engineer's Special Work." To the right appears a box for the "New Knowledge and Devices" which may result from the work of the engineer.

The next step is more speculative, namely the listing of "Potential Social Consequences" in the third box. The next box, "Find Expert Advice," is for statement of the problems that the potential social consequences indicate as requiring investigation by social science advisors. A sample of the principal fields of science advisors who might be consulted is listed with boxes for checking to see if they are needed on this problem. Some engineers' special work may lead to problems of a biological or medical nature, while the work of others may require the aid of psychologists or social scientists.

After preliminary contact has been established by the engineer with the required science advisors, the engineer must determine how far he will go himself in taking action. In the box on the right, four different magnitudes of action are indicated. The engineer may find all he has to do is to inform the appropriate social scientists about the problems, and they will pick up the responsibility from there on. In other cases there may be no funds to support the social scientists, and the engineer may feel it is his responsibility to campaign for appropriation of funds to support social science projects or to convince industrial management to include social scientists on their staffs.

I claim that the engineer, who does not have much spare time because of his basic engineering work and

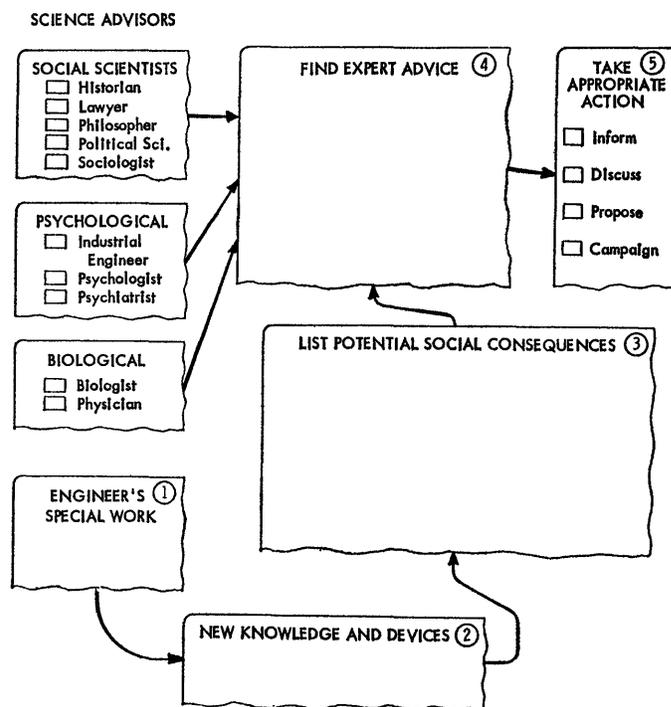


Fig. 1—A checking chart for analyzing the social responsibility of engineers and scientists.

his family responsibilities, can find short cuts to understanding the social implications of his work through devices such as the checking chart of Fig. 1. I have faith that the engineer can fulfill his social responsibility to help utilize the results of his work in keeping with mankind's highest aspirations.

To fulfill his social responsibility the engineer must understand that it is a responsibility he shares with many people both inside and outside his profession. He may not need to devote a tremendous amount of time and energy to the social implications of his work. The key to success lies in developing a fruitful perspective of the relationship of his work to the society in which he lives.

A SAMPLE USE OF THE CHECKING CHART

Consider an engineer working on the problems of data communication in connecting remote stations to a central computer. This is entered in the first block in Fig. 2. A successful solution to the data communication problem might result in a universal credit system, where every store, airline, doctor's office, race track, stock exchange, etc., would have terminal sets which would make transactions when the customer's coded credit card is inserted in the set. This would eliminate the need for money for most transactions. This new device is entered in the second block in Fig. 2.

Then we go on to block 3, "List Potential Social Consequences," such as:

- 1) The elimination of money might mean there would be no more armed robberies, which would be a step forward in the development of civilization.

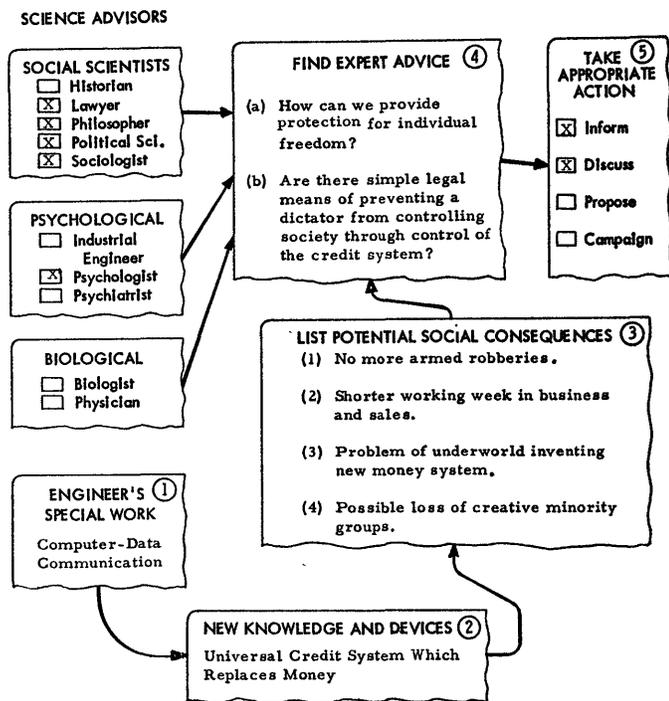


Fig. 2—A sample use of the checking chart.

- 2) The universal credit system might permit a shorter working week in sales and business administration work, permitting individuals to devote more time to creative hobbies which would enrich our community life.
- 3) New problems might arise such as gangsters inventing a new money system to finance illegal activities.
- 4) Police measures instituted to suppress the underworld gangsters might interfere with groups working on important social problems. For example, some public officials might be violating some of the provisions of the United States Constitution by discriminating against some minority racial or religious group. People in the community involved might feel like contributing a few dollars each to hire a lawyer to look into the case. These people might be afraid to contribute to this important cause when the accounting system would keep a record of each transaction. How do they know whether some future official will be able to distinguish between supporting a legal test case to protect the Constitution, and supporting some subversive activities? In such a situation the existence of this universal accounting system might inhibit people from protecting our constitutional government.

The next step for the engineer is to find expert advice to evaluate which of the potential social consequences pose real problems that won't just solve themselves in

the natural course of events. It would be desirable if we engineers could just refer these questions to some agency such as the National Science Foundation (NSF) for consideration. At present the NSF has a limited representation from the social sciences, so the engineer may have to find appropriate experts wherever he can. To assist the engineer in finding advice, I have listed some of the more obvious classifications on the checking chart under the principal categories of biological, psychological, and social science. At present the potential science advisors can usually be found on the staffs of nearby colleges or research institutes. In this case I have checked the boxes opposite the relevant categories in this sample case.

Informal discussion with these expert advisors results in a restatement of the problems as follows:

- 1) How can we provide protection for individual freedom in a more complex society where new technology such as computer-data communication systems permit a centralized accounting system covering all financial transactions in the community?
- 2) Are there simple legal means and technical characteristics of a computer-data communication system which permit safeguards to prevent potential dictators from seizing control of the system as means of gaining control of our country?

These questions as now restated are questions of importance to all citizens. The social scientists and the engineers both have additional responsibilities over and above their basic responsibility as citizens. However, the citizens at large have the basic responsibility of providing for financial support of such studies. The extent to which the engineer is responsible for taking action on these matters depends upon the state of development of social science research projects.

On the checking chart I have shown four degrees of action the engineer might take:

- 1) Inform: If through government agencies or private foundations there exist social science research projects adequate to study the problems, the engineer may discharge his social responsibility by simply informing these social scientists about the potential technological changes that may result from his work. In some cases an engineering research organization, in order to protect its proprietary interests, may prefer to hire social science consultants instead of releasing technological data to outside institutions.
- 2) Discuss: If the social science consultants are available and are financed, but do not have sufficient understanding of the technology involved, the engineer may have to organize discussions with the social scientist in order to pass enough of his special knowledge on to the people otherwise qualified to investigate these problems.

- 3) Propose: If there are insufficient social scientists available and the funds available are inadequate to support such research, the engineer may find it necessary to propose new appropriations and scholarships through his company, the existing research foundations, or through government agencies.
- 4) Campaign: If the agencies having the power to allocate funds for the study of these social problems fail to act, and the engineer is convinced that the problems will soon be urgent, he may have to plan stronger action such as campaigning to get political groups to pick up the problems. He may have to carry his campaign directly to the people, if the political leaders are insensitive to his proposals.

MAINTAINING A PERSPECTIVE

In his specialized engineering work the engineer has acquired through education and experience the portions of basic science that are most useful in his particular engineering assignment. The human needs on his job assignment usually have been evaluated by other people so that the human needs have already been translated into engineering objectives. To fulfill his role as "interpreter of science in terms of human needs," he needs some more direct contact with both science and with human needs. He can read such magazines as the *Scientific American*, which has popular articles on all levels of phenomena, as a way of keeping abreast of developments in science. To obtain a more direct contact with human needs, he can participate in a local church social problems study group. In order to develop a better understanding of the business world in which the results of his engineering work are used, he can read a magazine such as *Fortune*. He can develop a better perception of the social effects of science on a world scale by following the activities of the United Nations Educational, Scientific, and Cultural Organization (UNESCO) by reading one of their bulletins such as the quarterly *Impact of Science Upon Society*. The technical societies such as the AIEE and the IRE might eventually develop a monthly one-page abstract of significant articles relating to the social consequences of new technology.

CONCLUSIONS

Recent articles and panels on the social problems of computers and automation are a healthy sign that some engineers are developing a perspective of how their special field relates to the activities of mankind in general. Engineers need some kind of a framework to present an abstract but meaningful view of human activity to which they can correlate their own work.

A checking chart has been developed to assist the engineer in tracing the potential social consequences of his own work. A table of major sections of the biological, psychological, and social sciences is included to assist the engineer in selecting expert advisors.

In a democracy all citizens have a responsibility to keep aware of the major problems of our country. I believe that specialists such as engineers and scientists have an additional social responsibility because their knowledge is not readily accessible to the layman.

I believe that the engineer can carry out his social responsibility primarily by being concerned with the question: Are qualified experts investigating the potential social problems that might result from the engineer's work? The engineer can use the checking chart developed in this paper to assist in arriving at an answer to the question and in determining to what level of action his responsibility should extend. He shares with other specialists the responsibility for seeing that these problems are being studied and that provisions to inform the voters are made in our society.

I do not suggest that the engineer should be responsible for solving the social problems related to his work. The engineer's responsibility is more of a coordinator to alert the people of our country to the status of our coverage of the problems. If the engineer finds that a social problem relating to his engineering work is not being adequately investigated, he has a responsibility to refer questions to management, social scientists, government agencies, and to the citizens at large to stimulate the investigation of such problems.

ACKNOWLEDGMENT

I wish to express my appreciation to Dr. M. M. Astrahan for his valuable comments and discussion during the preparation of this paper.

Emergency Simulation of the Duties of the President of the United States

LOUIS L. SUTRO†

I. INTRODUCTION

A TECHNICAL problem is arising in our democratic government which engineers and mathematicians are equipped to assist in solving. The problem is how to approach making the kind of decision the President is called upon to make if missiles are detected on their way toward the United States. The number of facts on which a decision should be based appears to be increasing. The length of time in which to make the decision appears to be getting shorter.

Dr. Isador Rabi described the problem in a speech given in December, 1957.¹

Hydrogen bombs are going to be deployed at bases around the world under the control of many groups of persons. If an oncoming ICBM were detected 5000 miles away there might be time to intercept it with weapons not yet developed. But there will not be time to wake up the President to ask what to do, to call a meeting of the cabinet.

Facing a question that has not been mentioned before in the literature of computer engineering, we should give great consideration to method. I propose that as we approach each part of the problem we first describe it in the language most appropriate for the topic. Then let us attempt to translate this statement into computer and control terminology. Third, let us inquire to what extent an improved system can be built out of a combination of human beings and electronic equipment or electronic equipment alone.

The work of three men is the precedent for the attempt, in this paper, to describe human beings, human relations, and man-machine relations in terms of computer and control engineering. One is Dr. Warren McCulloch, a psychiatrist now at M.I.T., who is describing the human nervous system in this manner. One of his early papers was, "The Brain as a Computing Machine."² One of his more recent is on the design of reliable circuits out of unreliable components,³ giving one answer to the question of why the brain is as reliable as it is. The second man to supply precedent is Dr. Karl Deutsch, a political scientist now at Yale. He came to wide attention with the publication of a book explaining nationalism in terms of communication engi-

neering.⁴ The third is Jay W. Forrester who is now simulating business and economic systems by computer programs. He described his approach in, "Industrial Dynamics—A Major Breakthrough for Decision Makers."⁵ Prior to undertaking this he directed the development of the SAGE computer. I quote these three men extensively.

This paper was written during evenings, weekends, and holidays. The opinions expressed are mine or those whom I quote, and not necessarily those of my employer.

II. THE PROBLEM

We appear to be approaching an era of violence. The two major powers are manufacturing weapons to kill millions of people. They can be fired by the push of a button or by the signal from a computer. Many may soon be hidden so that they cannot be destroyed by bombing. As these weapons are built, installed, and connected to remote controls, the probability that one will be fired will rise rapidly, and the probability of a salvo to wipe out a nation will also rise, although more slowly.

The problem that engineers need to consider requires them to design controls that operate within limits. They must so arm the United States that another country considering an attack will know that it will receive a violent attack in return. Such armament is called deterrent power. On the other hand, they need to be concerned that building up deterrent power by the United States will lead to building up deterrent power by another country. This interaction is regenerative and leads to a rising probability of destruction of both sides.

The need for deterrent power was presented by Albert Wohlstetter in an article entitled, "The Delicate Balance of Terror."⁶ Wohlstetter is an economist for the Rand Corporation, a private nonprofit research corporation working on aspects of national defense and survival. He states that:

We must expect a vast increase in the weight of attack which the Soviets can deliver with little warning, and the growth of a significant Russian capability for an essentially warningless attack. . . . What can be said, then, as to whether general war is unlikely? Would not a general nuclear war mean "extinction" for the aggressor

† Instrumentation Lab., Dept. of Aeronautics and Astronautics, Mass. Inst. Tech., Cambridge, Mass.

¹ R. K. Plumb, "New weapons peril U. S. life, Rabi says," *New York Times*, vol. 107, pp. 1, 10; January 1, 1958.

² W. S. McCulloch, "The brain as a computing machine," *Trans. AIEE*, vol. 6, pp. 492-497; June, 1949.

³ W. S. McCulloch, "Stable, reliable and flexible nets of unreliable formal neurons," Res. Lab. of Electronics, M.I.T., Cambridge, Mass., Quart. Prog. Rep., pp. 118-129; October, 1958.

⁴ K. W. Deutsch, "Nationalism and Social Communication," John Wiley and Sons, New York, N. Y.; 1953.

⁵ J. W. Forrester, "Industrial dynamics—a major breakthrough for decision makers," *Harvard Business Rev.*, vol. 36; July-August, 1958.

⁶ A. Wohlstetter, "The delicate balance of terror," *Foreign Affairs*, vol. 37, pp. 217, 222; January, 1959.

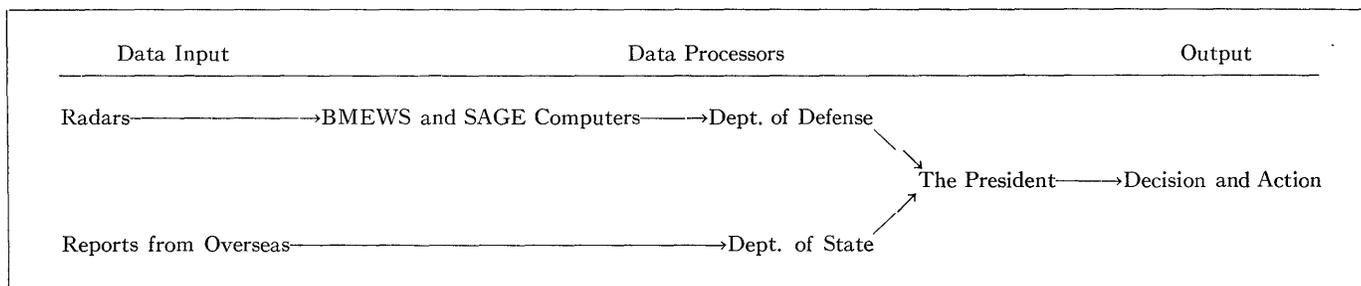


Fig. 1—Examples of channels in the man-machine system for making emergency decisions.

as well as the defender? “Extinction” is a state that badly needs analysis. Russian casualties in World War II were more than 20,000,000. Yet Russia recovered extremely well from this catastrophe. There are several quite plausible circumstances in the future when the Russians might be quite confident of being able to limit damage to considerably less than this number—if they make sensible strategic choices and we do not. On the other hand, the risks of not striking might at some juncture appear very great to the Soviets, involving, for example, disastrous defeat in peripheral war, loss of key satellites with danger of war spreading—possibly to Russia itself—or fear of attack by ourselves.⁶

Wohlstetter concludes that our ability to strike back in spite of attack should make a foreign country’s aggression less likely. This is deterrence. It consists of two parts: first, the weapons, and second, the ability to reach a decision to use them.

In arming against Russia, the United States is making a move which may be followed by more arming on the part of the Russians. This is positive feedback. It should be replaced by negative feedback of the kind to be described in the next section.

Let us return now to the problem, namely, how to approach making the kind of decision the President is called upon to make if missiles are detected on their way toward the United States. Dr. Karl Deutsch who has studied this problem suggests breaking it down into the following parts:⁷

- 1) Broaden the base of facts which lead to a decision.
- 2) Improve the reliability of the logic and computation used in processing these facts.
- 3) Shorten the time for making the decision.

Let us apply Dr. Deutsch’s analysis to a rough diagram of the man-machine system now used for making emergency decisions. (See Fig. 1.) The upper input illustrates electronic channels; the lower, written reports. The many other inputs have been purposely omitted. Data flow from these inputs through a stage of data processing before they enter the State and Defense Departments. In the executive departments, the new data are correlated with data stored in the files and memories of the personnel. They report to the President and they may recommend action. The President usually chooses between alternatives presented to him. If there is time he will consult with the National Security Council before deciding.

We can plot on this diagram the three improvements recommended by Dr. Deutsch. To broaden the facts on which a decision is based, there needs to be a greater input of data. In addition, there need to be better ways of tapping the facts stored in the executive departments. To improve the reliability of logic and computation requires improved data processors. To shorten the time requires an increase in speed of the entire decision-making system.

Pursuit of these three improvements can take us a long way toward a solution of our problem. To go further requires that we look closely first at the human being who holds the office of President, then at the biological computer which learns, remembers, and makes decisions. Delving into these biological mechanisms will allow us to examine possible simulators of memory, ability to learn, and ability to make decisions.

III. HISTORY OF OUR DECISION-MAKING SYSTEM

We have now described the problem this paper considers, in language appropriate to the problem. We began to convert this description to computer language when we made the simplified diagram of the system (Fig. 1) and observed that this is a man-machine system. To progress further in making a description in computer and control terminology, we need to go back to the origins of this man-machine system.

Perhaps by accident, the history of man-machine systems has never been told as a whole. To read present texts on the subject one might be led to believe that man-machine systems are not much more than a hundred years old. Yet books are a kind of machine. Their parts move with respect to one another. Moreover, as a human being reads words in a book, he is letting these words program the biological computer in his head.

Thus, a society that lives by rules written in books is a man-machine system. It has been evolving for 5000 years, from the days when men first wrote on stones and clay blocks to the present when recorded knowledge fills vast libraries. The evolutionary process has been carried forward by inventive people who created new systems when the need arose for them.

Benjamin Franklin might be called the first engineer to apply himself to the design of the American system. We know Franklin for his inventive work in the realms

⁷ K. W. Deutsch, private communication; February 21, 1959.

of electricity and heat. He discovered the identity of lightning and electricity and advanced the theory, still valid, that electricity is of two kinds, "positive" and "negative." He invented the lightning rod, a heating system for American homes, and the lending library. In 1754, he started work on the American system of government.⁸ The colonies were then threatened by the French and the Indians. The British government called a congress at Albany in the hope of getting the colonies to cooperate in raising troops and funds. Franklin, representing Pennsylvania, drafted the plan which the congress adopted, although the colonies did not.

Franklin's plan, redrafted twenty years later, became the Articles of Confederation, which were the system specifications for the first American government. When a more elaborate system was required, Franklin participated in the writing of the present Constitution.

James Madison was the leading designer this time. Unlike Franklin, he had specialized in the design and operation of governmental systems. He had helped to set up the state government of Virginia. He had served in Congress and observed the weaknesses of the Articles of Confederation. When the prospect arose of writing a Constitution he wrote out a proposal for it.

Adopted in 1789, the Constitution has grown since then by amendments and interpretation by courts. Congress has passed laws and administrators have made rules to carry out the laws. These rules are the programs which public officials pledge that their internal computers will obey.

The system devised by Franklin, Madison, and the other founding fathers is diagrammed in Figs. 2-4. Lines represent information flow. Fig. 2 suggests that each Congressman is ideally part of several feedback loops. The people in a congressional district elect him, then demand action of him. His action may be to participate in writing a new law or in opposing a proposed law. One feedback loop consists of reports by newspapers, radio, and TV. In another loop, the law is carried out by someone appointed by the President. Either the reports shown in the first loop or the impact of the law itself on wages, prices, and other interests of people shown in the second loop, may cause them to change their demand on their Congressman. If he acts to their satisfaction, they usually re-elect him.

The election of the President occurs in another loop which takes four years to traverse. Formation of the Constitution occurs in still another loop with the longest time period of all. Fig. 3 shows the same loops as Fig. 2, but now all of Congress and the whole electorate are represented. The whole body of law enacted by Congress is shown as a block at the center. To it is attached a small block below it, representing the newly enacted law.

A Congressman is also part of feedback loops that include very much larger groups of people than a congressional district. Such groups might be the automobile industry, the United States, or mankind. To show these feedback loops would require a very much more intricate drawing than Fig. 3. The number of these additional feedback loops and the quantity of people that they involve are a measure of the breadth of interests and the statesmanship of a Congressman.

Fig. 4 shows the response that the system was designed to make to an offensive incident or series of incidents by another nation. The incidents bore on the electorate or on special interest groups among the electorate who demanded action from Congress and the President. When a "threshold of tolerance" was crossed, Congress declared war and the President carried out the war through his secretaries of War and Navy. In practice, the incidents may have affected the owners and editors of mass media of communication so that they demanded action from Congress and the President. Or the incidents might come more fully to the attention of the executive than the public and thus the threshold of tolerance of the President would be crossed before that of the public and he would press Congress to a greater degree. This happened from 1939 to 1941.

Germany under Hitler, Russia then and today, lack the free flow of information and feedback controls of the kind described above. They are less stable in their relations with other nations. For example, a treaty, being a law, is part of the feedback control system of the United States. A treaty made by a dictatorship is observed or not as the dictator sees fit.

A system like that devised for the United States could be devised for the entire world and provide stability in that area also. The man part of the system needs to be educated for its task. The machine part needs to be capable of greater speed and reliability than the original system designed for the United States.

IV. CHANGES TO THE DECISION-MAKING SYSTEM, 1950 TO 1959

We have described in computer and control terminology the system that operated to repel an attack up to 1950. Let us now look at the changes that have been made in the present decade. Steps have been taken in each of the three directions that we considered desirable in Section II.

Fig. 1 showed the pattern of response that has been taking shape since 1950. Congress is no longer part of the loop of response. In January, 1955, Congress

handed to the President the power to defend Quemoy and Matsu if he likes, and to use atomic weapons there at his discretion. . . . The pattern is now clear; in the Middle East, as in the Far East, Congress has left it to the President to fight or retreat as he sees fit.⁹

⁸ H. C. Hockett, "Political and Social Growth of the United States, 1492-1852," The Macmillan Co., New York, N. Y., pp. 188, 189, 247, 286; 1935.

⁹ J. Reston, "War-making power; Quemoy crisis shows how control passed from Congress to President," *New York Times*, vol. 107, p. 4; September 4, 1958.

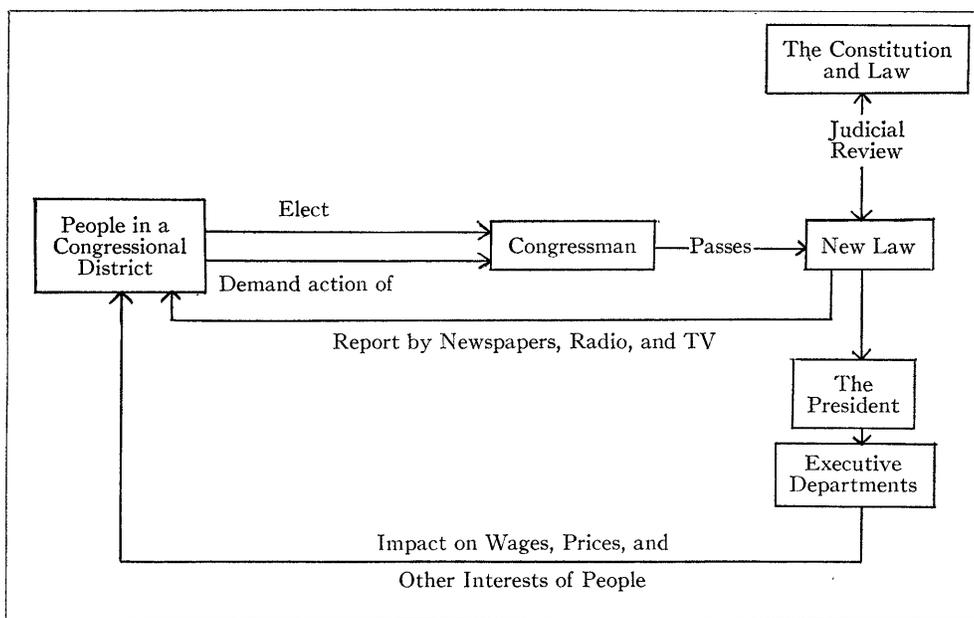


Fig. 2—Congressman in feedback loops.

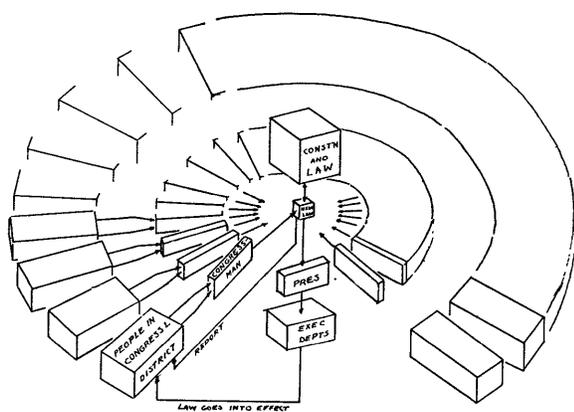


Fig. 3—Simplified block diagram of the United States Government.

This act of Congress formalized the practice begun by President Truman at the outbreak of the Korean Conflict in 1950. This practice has served to shorten the time for making the decision, but I question if it has increased the reliability of the decision. The older system requiring debate in Congress and across the nation brought more minds to bear on the problem.

The flow of facts into the decision-making system has been increased and speeded by two unique electronic systems, SAGE (Semi-Automatic Ground Environment) and BMEWS (Ballistic Missile Early Warning System).

Fig. 5 shows the contents of the upper half of the diagram of Fig. 1 arranged in pictorial fashion. For simplicity, it shows only the part of the system where data are detected and moved at electronic speeds. The flow of reports from overseas to the State Department is assumed to be present but not shown. SAGE computers with radars above them are shown in the inner ring. BMEWS computers with radars above them are

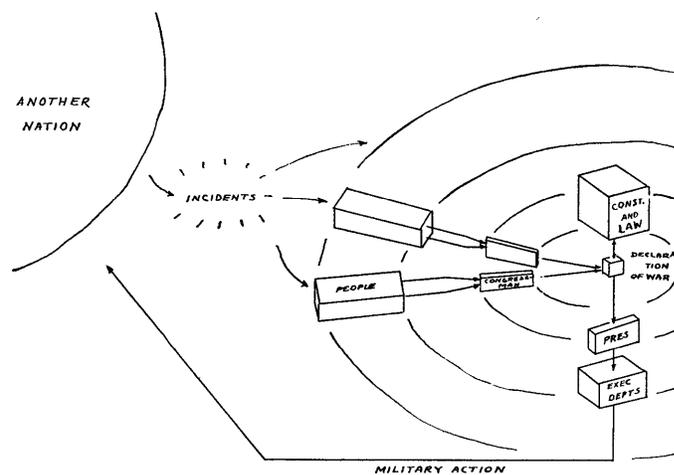


Fig. 4—Response of the United States to an attack, 1789-1950.

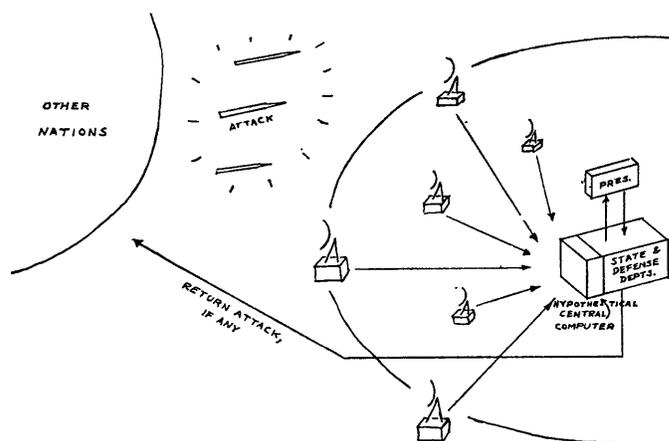


Fig. 5—Hypothetical response of the United States to an attack in the present and near future. (The flow of reports from overseas to the State Department is assumed but not shown.)

shown in the outer ring. Their signals are shown entering a hypothetical central computer which organizes them for presentation to personnel in the Defense Department. These people merge the new data with pertinent data from their own memories or their files. Then they make selections from these merged data to compose reports and make recommendations to the President.

The SAGE and BMEWS systems are part of the improvement to the decision-making process that we seek. They broaden the base of facts on which a decision would be made. Let us examine those systems in more detail.

Sage is a gigantic man-machine system whose radars watch the sky over the United States and feed information into the largest computers so far mass-produced.¹⁰⁻¹⁵ At each of about 30 "direction centers" in the United States, a 75,000-instruction program runs continuously to process the data and display them on large scopes. Fig. 6 shows a typical computer center. Fig. 7 shows this center being fed by information from radars at the left and giving out information to planes and missiles at the right. A tie is shown at the top to higher headquarters and at the bottom to an adjacent direction center.

Fig. 8 shows the point at which the SAGE computer gives up its data to a man who then makes a decision. Here an Air Force officer, looking at a displayed map on which approaching enemy planes are shown, orders planes or missiles to intercept them. The SAGE computer carries out his order by directing the plane or missile to the target.

In 1960 a system is scheduled to go into operation which will inspect in a similar fashion the air space between the United States and other countries. This is the Ballistic Missile Early Warning System (BMEWS) whose radars have a range of 3000 miles.¹⁶ The radar returns will be interpreted by a computer to discover whether each object seen moving at high speed is a meteor, a satellite, or an ICBM. As in the SAGE system, the conclusions reached could be used to generate a display, send a message, or fire an interceptor missile. But it can also do more than SAGE. By tracing the trajectory of a missile, BMEWS can determine where it

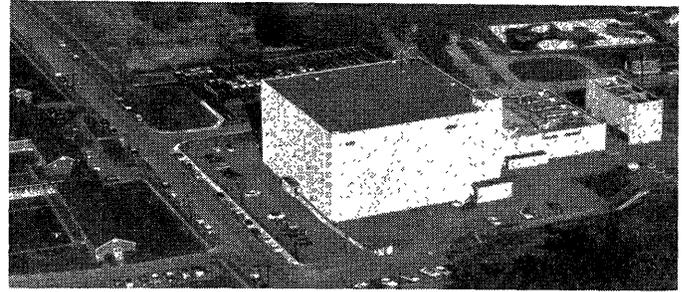


Fig. 6—A SAGE direction center building. (Photograph by Lincoln Laboratory, M.I.T.)

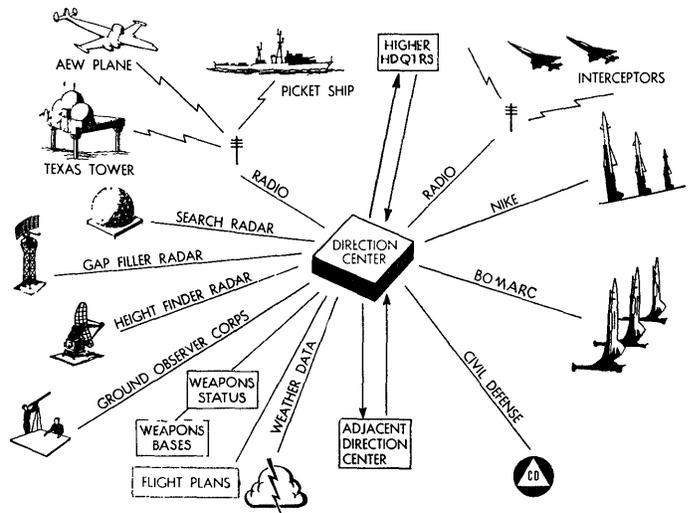


Fig. 7—Inputs to SAGE direction center are from radars at left, weather stations and commercial planes below. Outputs are to planes, missiles, adjacent direction centers, and higher headquarters. (Drawing by Lincoln Laboratory, M.I.T.)



Fig. 8—Air Force officer ordering interception of enemy plane or missile. (Photograph by Lincoln Laboratory, M.I.T.)

¹⁰ R. R. Everett, C. A. Zraket, and H. D. Bennington, "SAGE, a data processing system for air defense," *Proc. EJCC*, pp. 148-155; December, 1957.

¹¹ W. A. Ogletree, H. W. Taylor, E. W. Veitch, and J. Wylon, "AN/FST-2 processing for SAGE," *Proc. EJCC*, pp. 156-160; December, 1957.

¹² R. R. Vance, L. G. Dooley, and C. W. Diss, "Operation of the SAGE duplex computers," *Proc. EJCC*, pp. 160-163; December, 1957.

¹³ M. M. Astrahan, B. Housman, J. F. Jacobs, R. P. Mayer, and W. H. Thomas, "The logical design of the digital computer for the SAGE system," *IBM J. Res. Dev.*, vol. 1; January, 1957.

¹⁴ H. D. Bennington, "Production of large computer programs," *Proc. Symp. on Adv. Prog. Methods for Digital Computers*, ONR Symp. Rep. ACR-15; June 2, 1956.

¹⁵ D. R. Israel, "Simulation in large digital control systems," presented at the Natl. Simulation Conf., Houston, Texas; April, 1956.

¹⁶ "The ICBM's: danger—and deterrents," *Newsweek*, vol. 52, pp. 56-57; December 22, 1958.

came from; then, assuming the missile takes no evasive action at a later stage, BMEWS can predict where the missile is likely to go. The prediction may make it possible to destroy the missile in the air. The estimate of where the missile came from can be the basis for a decision to retaliate.

Congress' response to the threat of nuclear attack has been to increase the effectiveness of the President and at the same time weaken the feedback loops of which the President is a part. This has reduced the sensitivity of the control system to public demands and restraints. It appears that attention should be given to providing new control loops to replace those that have been weakened or removed. But first let us give our full attention to the problems of the President.

V. PROBLEMS OF THE PRESIDENT IN AN EMERGENCY

The following are two situations that he might have to face.

Fig. 9 shows country *A* (aggressor) launching an attack on country *N* (nonaggressor) intended to both destroy it and prevent it from retaliating. Let us assume that the deterrent power of country *N* is its ability to launch missiles. It appears that in the immediate future, the majority of launching sites are likely to be known, with the result that a retaliatory attack by missiles can be made only if it is started before the original attack arrives. There are several "ifs" here and if they are all to be satisfied, speed of decision is very important. However, when the retaliatory power is hidden, as we are led to believe it will be in a few years, great speed will not necessarily be needed. A reliable decision requiring days if necessary appears far more important, lest an error be made.

However, a circumstance that would demand speed of decision arises when the President's life is threatened by an approaching missile. He has two alternatives: to order a retaliatory attack on a suspected country, or to wait, knowing that if he is destroyed someone else may order the retaliatory attack. Needed is fast processing of data that give him a reliable basis for decision in the time he has available.

As we approach closer to an examination of the duties of the President, let us consider what Dr. Deutsch believes a data-processing system can and cannot do today:⁷

- 1) Compute trade-offs (if I do this, then what?).
 - a) What might be the effect of each of our actions on the civilians in this country?
 - b) What will be the effect of each of our actions on the capabilities of the attacking countries?
 - c) What will be the effect on third countries?
- 2) Prepare estimates of the over-all effect of an action.
- 3) Make recommendations to the President.

No computer today has the learning capacity of an individual, much less that of a community. Computers should facilitate human and community learning by evaluating and cross-checking relevant data. Progress consists of putting more and more of the information-handling burden on the mechanical and electronic equipment and leaving an ever-smaller amount of ever-higher decisions to the human agent.⁷

But suppose the human agent does not respond because he is asleep, as Dr. Rabi suggested, or for some other reason. It is the obligation of computer engineers and programmers to inquire what they can do to supplement the President. The American people may not

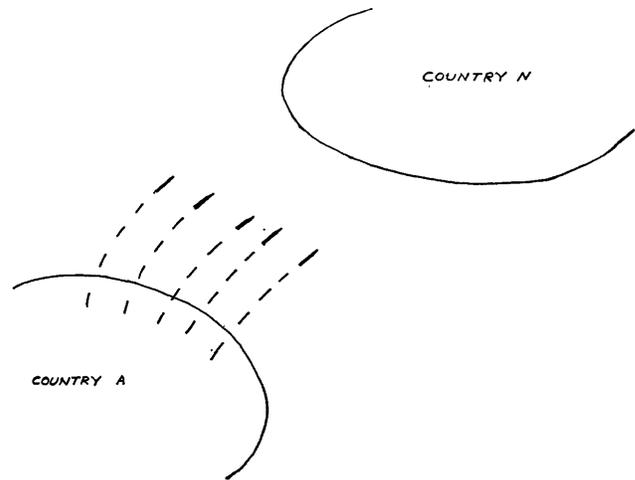


Fig. 9—Country *A* launching an attack on country *N*.

accept what they propose. But proposals should be made periodically and in greater detail as more techniques become available.

To that end a description will be made first of the emergency duties of the President, then of the qualities that led to his selection by the American people. These two descriptions can be regarded as part of the specifications of a simulator. With these specifications before us we will then inquire how far engineers have progressed toward the emergency simulation of the duties of the President.

VI. THE DUTIES AND THE QUALITIES OF A PRESIDENT

The President's task in the problem we are considering is to order or not to order the military to act. He is there to make sure that the military are effectors, not decision points. For example, in an international crisis, military men get poised, ready to use their weapons. The President, on the other hand, will act the way his personality dictates.

All that we ask of a President is that he be his best self. We mean by this that we ask him to apply to a major decision the traits that he demonstrated before taking office. Yet all of us have our ups and downs. There is always the possibility that a quick decision will be required when the President is not at his best. A system to back up the President, therefore, is being considered.

If such a system were to win the acceptance of the American people it would need some of the qualities of a President. What are some of these?

To avoid the mental images of actual Presidents, let us refer to the President for the moment as a system—a very elaborate biological system. This system is put into its key position by a process whose first milestone is nomination at a national convention. It is then tested for three to four months in a kind of trial presidency during which it is presented with the problems of the President and called upon to declare what decisions it would

make if it were the President. During this same time, the system is watched by reporters and TV: How does it treat its wife, its children, its friends? What are its beliefs? Does it get angry easily? During this testing period, an image is built up in the minds of the voters. The image is one of a predictable system, to the extent that the voter has made observations. On election day, at the end of this test period, voters choose between two or more systems.

Looking more closely at a system, we observe that what interests the voters most—or what we think should interest them most—is its information-processing subsystem. This is a network of switching and storage elements. Of the 30 million million cells that comprise a human system about one tenth make up its information-processing or nervous subsystem.

Dr. McCulloch calls this subsystem a “biological computer.”¹⁷ Feeding information into it are the senses of sight, hearing, touch, taste, smell, and acceleration. It contains three kinds of memory, a means of learning, and a means of making decisions. It appears that a system to simulate (see the Appendix) the duties of the President will require the following properties of biological computers:

- 1) Memory
- 2) Ability to learn
- 3) Ability to make decisions.

In the following three sections we briefly describe and evaluate the steps that the computer engineering profession has taken toward simulation of the duties of the President. These efforts are for other purposes, but they serve this purpose.

VII. SIMULATION OF HUMAN MEMORY

By computer memory we mean both the static storage and the continuously running program that up-dates this storage and presents alternatives for decision. Let us look at the memories in both SAGE and in Industrial Dynamics Research programs at M.I.T.

From the data received by its radars, a SAGE computer can predict the course of each aircraft in the airspace which it is monitoring. It can predict the points at which interception can be made by aircraft taking off from different airfields. An Air Force officer, watching the two predictions plotted on a scope, can select an aircraft to make an interception. This action is illustrated in Fig. 8.

Just as the SAGE computer contains a model of moving aircraft, so an Industrial Dynamics program contains the model of a company. In a diagram of a typical

model,¹⁸ a solid line represents the flow of goods from the factory to the warehouse, to the retailer, and finally to the customer. Dashed lines represent the flow of information from the customer to the retailer and all the way back to the factory. Numbers in the lines indicate the length of delays. Where a flow of goods and a flow of information touch, a decision is made.

Forrester's diagram represents a more advanced form of analysis than that shown in Figs. 3–5. The analysis itself consists of difference equations.

The following (typical) equation tells how to calculate the level of Unfilled Orders at (the) Retail (end of the business) at time K :

$$UOR_K = UOR_J + DT(RRR_{JK} - SSR_{JK}).$$

This equation tells us that the unfilled orders at retail at time, K , are equal to the unfilled orders at retail at the previous time, J , plus the inflow minus the overflow.¹⁸

The inflow is the product of a time interval DT and a rate, RRR , that holds from times J to K . The outflow is the product of the same time interval and another rate, SSR . Each equation is evaluated independently, using the results from the previous evaluation of all the equations. (See the Appendix.)

While the simulator of the President would require facts and figures bearing on current issues, its memory of environment can be approximate. Industrial dynamics models could serve this purpose. The model described above was intended to bring understanding of one company to its factory manager or corporation executive. Models of the groups of companies that make up an industry would be useful to the simulator of a President. Models of the United States government, its allies, and its adversaries would be necessary.

VIII. ABILITY TO LEARN

The present system for making emergency decisions is one that learns. The biological computers in the system learn by changing, or increasing, the storage in their memories. The system as a whole learns in several ways, one of which is illustrated in Fig. 3. Here trials and errors are recorded in the memories of human beings and lead to new rules.

The first method of learning we shall consider for the simulator is continual reprogramming. Dr. Richard C. Clippinger suggests:¹⁹

It will probably be necessary for the governmental simulator to operate in parallel with the President for a considerable time in order to learn. Computer learning is similar to the successive reprogramming of a complicated process by means of more and more efficient programs, drawing intelligently on more and more past experience. Probably the longer it has been in operation the more efficient it will be, that is, the more it can accomplish in a few microseconds.

¹⁷ W.S. McCulloch, “Reliability of Biological Computers,” lecture, University of Pittsburgh, Pittsburgh, Pa.; May 10, 1957. (Unpublished.)

¹⁸ J. W. Forrester, “Formulating Quantitative Models of Dynamic Behavior of Industrial and Economic Systems, Part I,” Industrial Dynamics Res., School of Industrial Management, M.I.T., Cambridge, Mass., Memo. D-16, pp. 8, 30, 31; April 5, 1958.

¹⁹ Private communication, October 19, 1958.

The SAGE system learns in the manner described by Dr. Clippinger. A staff of programmers at the System Development Corporation in Santa Monica, Calif., attends the system and incorporates what is learned in an improved program.¹⁴ To "get back into" a program of 75,000 instructions requires careful documentation augmented by computer methods for changing the program. The need to rework increasingly large programs is an incentive for the second method of computer learning we are considering here—heuristic programming or "artificial intelligence."

Dr. John McCarthy describes artificial intelligence:²⁰

These programs all use trial-and-error learning. A criterion for an acceptable solution is known. Then the machine "searches" a group of potential solutions for one answer that meets the criterion . . . Unfortunately the groups or classes of potential solutions of interesting problems are too large to be examined one at a time by any conceivable computer.

Therefore, we must devise methods called heuristics for replacing the search of the class of potential solutions by a number of searches of much smaller groups. It is in these heuristics that the intelligence, if any, lies.

Programs written by Newell, Shaw, and Simon have proved theorems of logic²¹ and played chess, each with increasing skill. A program written by Gelertner and Rochester containing the theorems and heuristics taught in a high-school geometry class has done the homework and taken the examinations of that class.²²

But each of these programs handles only a limited range of problems. To extend the range we need to tie together a learning system with many storing systems. Each of us needs only to look in a mirror to see a system that does all these things and, in addition, makes decisions of the kind described in the next section. Examination of this system is instructive. Its elaborate transducers facilitate learning. These transducers include the eyes, ears, sense of touch, and inertia-sensitive inner ears. For each transducer there is a corresponding part of the biological computer where information is processed before it is stored. Thus the transducers are not only detectors, they are filters, switching incoming information toward its place of storage. Furthermore, they are adjustable filters. When you are looking for something, you have tuned your detectors to find that thing and ignore other things. Searching for a red ribbon in your bureau drawer, you tune your eyes to search for red and need only make a yes or no decision about each thing you see.

The radars of the SAGE system report only targets moving at a speed greater than a certain amount. However, the filter here is not adjusted by the computer.

²⁰ J. McCarthy, "Getting closer to machines that think," *New York Herald-Tribune*, Engineering News Supplement; May 24, 1959.

²¹ A. Newell, J. C. Shaw, and H. A. Simon, "Empirical explorations of the logic theory machine, a case study in heuristic," *Proc., WJCC*, pp. 218-230; February, 1957.

²² H. L. Gelertner and N. Rochester, "Intelligent behavior in problem-solving machines," *IBM J. Res. Dev.*, vol. 2, pp. 336-345; October, 1958.

Moreover, radars "see" with very coarse resolution. Great sums of money have gone into the development of radar. There has yet to be a comparable effort at developing a high-resolution system with adjustable filtering to enable an electronic system to "see" the objects that human beings not only see but think about most of their waking hours.

In the absence of its own inputs, the simulator will have to take in the form of punched cards or electric signals the observations of those who do have these inputs. Lacking a filtering system, it will have to use the classifications of events made by these observers. The classification can determine what heuristics and what part of the memory are to be employed.

IX. ABILITY TO MAKE DECISIONS

Decisions can be made by computer programs according to predetermined rules. To run these rules *and* memory of the kind developed by Industrial Research *and*, possibly, a learning routine would make a slow simulator. Speed can be obtained by imitating the human decision system.

The decision-making apparatus in the human system is the reticular formation. It is the core of the brain stem. It is about as big around as a cigarette and about two inches long. Each of the several thousand large cells in this formation:

receives signals from almost every source in the human body, coded in pulse-interval modulation to convey whence the signal came from and what happened there . . . The reticular formation decides what he ought to do, what he should heed, how vigilant he ought to be and whether he has time for that idle fancy that inspires his future action.²³

The method by which the several thousand large cells of this formation reach a decision is similar to that used by a battle fleet.

Every ship of any size or consequence receives information from the others and sweeps the sky for hundreds of miles and the water for tens of miles with its own sense organs. In war games and in action, the actual control passes from minute to minute from ship to ship, according to which knot of communication has then the crucial information to commit the fleet to action . . . It is a redundancy of potential command, wherein knowledge constitutes authority.

In the reticular formation, each cell is like a ship of this battle fleet, able to take command when the information it has received is accepted, by all of the several thousand large cells, as that most requiring attention.

Having spent much of his life mapping the nervous systems of monkeys and men, Dr. McCulloch is now studying the nerve connections of the human reticular formation. Every one of the several thousand large cells in this formation is connected to nearly every other. In addition, every one of these cells receives signals from

²³ W. S. McCulloch, "Where is fancy bred," Bi-Centennial Conf. on Experimental Psychiatry sponsored by the Western Psychiatric Institute and Clinic, Dept. of Psychiatry, University of Pittsburgh School of Medicine, Pittsburgh, Pa.; March 5, 1959.

some of the afferent cells of the body and from some of the cells of the cerebral cortex. This much can be determined from dissection. What cannot be determined this way is how each cell influences every other.

Fortunately, much is known about how the reticular formation performs. From this knowledge, McCulloch is considering a possible logical diagram showing how its neurons may affect each other. The resulting design can be implemented by artificial neurons such as those being built by Jerome Lettvin.²⁴

Could the logical design also be implemented by a programmed computer? A small part of it could. Each neuron can be represented by storage registers containing the neuron threshold, the state of the neuron after the last cycle of excitation and inhibition, and the nature of the connections to other neurons. To simulate all of the interconnections in the clock time of the brain would require the processing of at least 1000(!) instructions in 0.1 second.

An assembly of artificial neurons is called a parallel computer, meaning that all logical operations are occurring at the same time instead of sequentially, as in a programmed computer. For the present, parallel logic is a goal to work towards while using programmed logic.

X. WHEN SHOULD THE SIMULATOR BE USED?

A programmed simulator, although slow, can render a service now by providing an operating model of the environment of the President, by demonstrating how new rules may be learned, and by demonstrating how rules may be applied to make decisions. Starting as a guide to decision-makers, a simulator could be gradually improved until it might be able to make decisions on its own. It would be for Congress, the President, and the American people to decide if the simulator should be allowed to do this.

Three measures will be suggested as aids in deciding when a simulator should be used in this way. One measure is the extent of internal restraint. As Dr. Deutsch puts it:

For any large . . . memory system, the specific content of all combinations that might become dominant . . . cannot be predicted. The possibilities are too numerous as to what combinations might arise in a human mind, or in any computer . . . remotely comparable. Hence we fear entrusting political control to any one human mind, or to any small committee, even though we trust them as being human personalities . . . who share the unspoken and unstated values and inhibitions of our culture and religion.

An electronic machine (at present) can include in its memory, at best, only those rules of law, morality and religion that have been stated explicitly in words. . . . These . . . rules a computer would then apply with terrible literal-mindedness. It might become the electronic embodiment of the letter that kills, rather than of the spirit that gives life.

²⁴ J. Y. Lettvin, "Nerve Models," Res. Lab. of Electronics, M.I.T., Cambridge, Mass., pp. 178-179; January 15, 1959. In the diagram, the unlabelled diode at the left is the excitatory input; that at the right, the inhibitory input. The wiper of the potentiometer determines the threshold.

Limitations of computers, when recognized by engineers, appear to stimulate efforts to overcome the limitations. This gives direction to the development of new techniques of memory, ability to learn, ability to make decisions and the additional categories mentioned by Dr. Deutsch.

A further challenge from him should be quoted:

To build into a computer the properties of perceptiveness, tolerance of ambiguity, mercy and spirituality—that is, perceptiveness toward second-order and higher-order patterns of preferences—would require capabilities far in excess of those available at present. So long as such vastly greater capabilities have not been developed, computers can aid human judgment but cannot safely replace it.

The second measure we shall consider is the extent and sensitivity of feedback control such as that in Fig. 3. If we find difficulty in trusting one human mind, we shall have greater difficulty in trusting a simulator. However, a control network is possible consisting of many simulators. Given authority to act, a decision would be made by a majority of those simulators that had not been destroyed by attack or sabotage. Each would simulate the duties of a Congressman or group of Congressmen. As Dr. Clippinger has suggested for a simulator of the President,¹⁹ each should be operated in parallel with the one it is simulating so as to:

. . . (a) learn, (b) demonstrate to Congress and the President that it is worthy of their respect and faith for at least a limited period, (c) provide time to educate and persuade the people of this democratic country that it should be used.

Such a network could have feedback controls as extensive as Congress itself, at least during trial periods.

The third measure of when a simulator should be used is the measure of the emergency when, if Congress, the President, and the American people have previously approved, the simulator would be permitted to act. Seeking this measure takes us back to the question raised by Dr. Rabi. In accord with that, two conditions would make the use of a simulator desirable. One condition is imminence of destruction such as a 90 per cent probability that 5,000,000 people will be killed, a 9 per cent probability that 50,000,000 people will be killed, or any of the equivalent probabilities. The other condition is the inability of the President to respond.

Equipment with extraordinary reliability is needed to determine both of these conditions. The estimate of probable deaths would need to be made by a computer that has both information about approaching missiles and models of population. The President's ability to respond in a predetermined time could be determined by interrogating him, by requiring him to report periodically, or by some other method.

The desired reliability should be obtained either by operating computers in parallel, which is done in the SAGE system, or by applying the theory of building reliable circuits out of unreliable components.³ The

latter requires the kind of parallel logic described in the last section with interconnections and thresholds so selected that the failure or erratic behavior of one or more elements will not affect the output.

APPENDIX

DEFINITION OF SIMULATION

The word "simulation" is used in this paper in its modern technical sense:²⁵

. . . to assume the appearance of, . . . without any intention to deceive. I refer to its use in the field of mechanical-electronic computation. Here the procedure is to simulate physical or mental processes in setting up a problem which is then given to a computer to solve.²⁵

The Industrial Dynamics Research program at M.I.T. uses the words "make a model of" in the place of

²⁵ J. C. Warner, "The fine art of simulation," *Carnegie Alumnus*, Carnegie Inst. of Tech., Pittsburgh, Pa.; 1959.

"simulate." The model in this case is a set of equations. These M.I.T. people save the word simulate to describe the evaluation of these equations, one at a time, for a given set of input conditions. They solve the equations at time intervals which are short compared to the shortest delay intervals of the system being modeled. They are thus simulating simultaneous solution.

In this paper "simulate" is given the meaning of the first paragraph above. Simulation here is intended to achieve a "quality" equal to or excelling the performance of the human being to be simulated, for the periods when it is given his responsibility. The "quality" of performance is a composite of breadth of facts which lead to a decision, reliability of the logic and computation used in processing these facts, speed, and human considerations. A simulator might attain acceptable quality by excelling in some of these considerations while falling short in others.

Can Computers Help Solve Society's Problems?

JEROME ROTHSTEIN†

INTRODUCTION

THE advent of large-scale computers gave new impetus to mechanizing the handling of tremendous quantities of data. It also indicated the possibility of carrying out many ventures of social significance which are now completely impractical. It is hard to see an important social revolution in the first, *per se*. Automatic billing of telephone subscribers or mechanization of clerical activities, for example, is only substitution of machine for manual activities. This has been going on continuously since the beginning of the industrial revolution. Exciting prospects emerge when one considers fields characterized by enormous amounts of data together with complicated intertwining causal relationships buried under statistical blur.

The present paper considers a few of very many possibilities. They were chosen mainly because one might expect them some day to have enormous impact, both on the individual and on society. The first group bears on the weather and on economic planning and policy, the second on various questions of public health, and the third on "the proper study of mankind," man him-

self. They are tentative groupings with no pretense to completeness or profundity. It is believed, however, that they make some general statements plausible. These are that modern computer and data handling techniques may

1) lead to making our economic system more productive, and to smoothing cycles of inflation and deflation, or employment, and of farm income;

2) revolutionize our ideas of public health, and make the world a more wholesome dwelling place;

3) revolutionize our knowledge of ourselves, our abilities, susceptibilities, mental, physical and genetic constitution, as well as diagnostic and preventive medicine.

We believe it is the responsibility of the computer engineer and scientist to point out such potentialities, to acquaint specialists in many fields with what computers can do, to collaborate with them in applying computer techniques to those fields, to keep research foundations and government agencies aware of areas worthy of support, to keep administrators, policy makers, and legislators informed and advised, and thereby to assist in the formulation of sound public policies.

In the discussion below, military, industrial, and scientific applications are very largely neglected. This is

† Edgerton, Germeshausen and Grier, Inc., Boston 15, Mass.

not because these fields are unimportant, but rather because their importance has been so well recognized that a paper of this length could now contribute little to them.

WEATHER, COMPUTERS, AND ECONOMIC POLICY

The historic role of weather and climate in human and economic terms needs no belaboring. Famine, drought, floods, plenty, epidemics, mass migrations, the rise and fall of civilizations, wars and their outcome have often been engendered or decided by long- and short-range weather and climatic variations. With foreknowledge, as in the biblical story of Joseph and Pharaoh, countermeasures become possible, with tremendous gain in economic and human terms. Modern data processing is making it possible to use the tremendous mass of accumulated and current meteorological, astronomical, and climatic data to make long- and short-range weather predictions more accurate. When continuous global cloud surveys by satellite are available, accuracy will advance even more, and even weather control will doubtless be more effective. The average citizen will know when to carry an umbrella or plan a vacation, retail establishments will plan inventories more intelligently, and farming will be less of a gamble. If one knows a drought is going to occur, for example, it is silly to plant extensively where there is no artificial irrigation. If it is known that a good crop is imminent and that the following year will be one of drought, the government might well consider making provisions for crop storage. Knowledge of this sort on a global scale can go a long way toward eliminating famine in one place and glut in another. The effect of this on international relations and on the likelihood of revolutions and wars may well be stupendous. More locally, public works programs in agricultural areas could be planned to dovetail with an expected decline in demand for farm labor, with lowered shock or recession of the whole economy. Production based on agricultural raw materials can be more intelligently programmed when yields are forecast even approximately. Inventory policy and servicing depots for farm machinery could be set up more efficiently. Railroad rolling stock, truck fleets, and the like can be administered more efficiently when demand is predicted more accurately. Clearly, the techniques of operational research, which could make extensive use of data-handling systems, would become much more readily applicable to tremendous areas of planning, allocation, production, servicing, inventory, and other policy problems. Expert consulting services for small enterprises, unable to support operational research on their own, would become economically justified, and might well lower failure rates for small business. Fuel, electric power generation, and hydroelectric policies are also weather dependent, and the economic value of long-range weather information in these fields is clearly tremendous. It hardly seems an exaggeration to say that little of the national and world economy is un-

affected by weather, and that most of it would benefit by better knowledge of weather trends.

There is no reason to restrict considerations like these to agriculture. Some control over inflation and deflation is exerted by the Federal Reserve System and the Treasury by control of discount rates and the sale of government securities, for example. Policies presently established as "curative," often with undesirable lag, may become "preventive" when economics, aided by large-scale computer techniques, has made economic forecasts more accurate. The forecast problem is comparable in complexity to the global weather problem, and may well be attacked with similar techniques. Fluctuations may be one of the prices of freedom, but the instability and suffering engendered by excessive swings can surely be much reduced by intelligent countermeasures, and with no real loss of freedom. It would seem almost suicidal not to develop such countermeasures. Computer techniques can be expected to play a vital role in this field.

COMPUTERS AND PUBLIC HEALTH

In many fields of public health, detailed cause-and-effect relationships are submerged by multitudinous accidental factors. Masses of data gathered over long periods of time must often be digested and tested by sophisticated statistical techniques before valid statements can be made about them. All statements, on which action is to be taken, should be treated as testable statistical hypotheses, with action justified when the hypothesis has reached some preassigned confidence level. The examples below were chosen at random, and represent but a small sample of the total one could cite.

Fluoridation of drinking water is a measure backed by much competent medical opinion and bitterly fought by a number of lay groups. Proponents of fluoridation claim that tremendous reductions can be made in dental caries with no ill effects. Its opponents make dire predictions about the effect of fluoride on the kidneys, the nervous system and almost every other organ in the body. There are millions of people at the present time who consume fluoridated drinking water. It therefore seems entirely feasible to amass completely convincing and compelling data on the correlation between fluoridation of drinking water and every ill to which the flesh is heir. Of course, even in the face of compelling evidence one often finds crackpots who refuse to be convinced. If one has faith in democracy, however, one must believe that in the long run the majority will recognize and accept the truth. If statistically valid evidence of this sort is gathered—and computers can certainly play a vital role in doing this—and it turns out that almost all tooth decay can be safely and permanently eliminated from the population, the gain would be incalculable.

A similar situation appears to be involved in estimating the effect of radioactive background and cosmic rays on stillbirths, cancer, and genetic impairment of large populations. It has been supposed by some that

there is a certain threshold of danger. Others maintain that any increment in the radiation background, no matter how small, will produce additional cases of defective births, mutations, and of cancer. One clearly cannot make controlled human experiments, but the city of Denver is a mile higher than New York. The inhabitants of Denver are thus exposed to a higher cosmic-ray background than the people of New York. In addition, uranium ores in Colorado and a number of other western states must subject the people in those regions to radioactive background, to uranium materials in their foods and the like which are absent in many coastal areas. Some parts of the world, such as Travancore, India, have high natural backgrounds due to the presence of monazite sands or other radioactive materials. It thus seems entirely feasible to get statistically convincing data on the incidence of conditions of all sorts in existing populations living under a variety of radiation backgrounds. With extensive, computer-processed data one can make a more realistic assessment of the dangers of atomic testing, for example. One can perhaps find whether a threshold for radiation damage exists, or even if there are beneficial effects of radiation in very small amounts. There is some evidence that normal individuals have some immunity to cancer. It therefore does not seem entirely impossible that fall-out in very small amounts might have no effect on normal individuals but could shorten the life expectancy of individuals fated to succumb to cancer by a small amount. Similarly, data on abnormal births and congenital defects might lead to a better understanding of the genetic hazards.

A third field is that of poisons, taken in a broad sense. Coal or oil smoke from the heating plants of private dwellings, and exhaust fumes from automobiles could conceivably have subclinical toxic effects. It is known that carcinogenic substances are produced in oil refineries. Such substances can also be produced by combustion of tobacco and many organic or carbonaceous materials. Some studies indicate a connection between smoking and the incidence of lung cancer and cardiovascular diseases. It seems possible that "chemical fall-out" from daily industrial, heating, smoking, and automobile-riding activities could be a hazard greater than radioactive fall-out. Chemical agents can also produce mutations. Among these are colchicine and other complex compounds related to coal tar derivatives and other substances with great biological activity. A large-scale statistical survey of the quantities and identities of atmospheric and other contaminants of all sorts, and their correlations with the incidence of various diseases, appears eminently desirable. If one found, for example, that small communities in the Rocky Mountains have far lower incidence of cardiovascular disease than smoggy industrial areas, with suburban residential areas intermediate, then laws requiring the chemical clearance of smog and precipitation of dust particles might ultimately be in order. Many industrial poisons are known

and in many places measures have been taken to prevent atmospheric contamination, but by and large a serious and obvious outbreak of some condition seems to be required before public opinion is aroused enough to take action. It is therefore quite possible that many cases of this sort are unnoticed because cause-and-effect relations are buried in a sea of accidental factors. The possibility that we can live under healthier conditions seems too real for us to overlook these potential applications of computer processing and analysis of the tremendous amounts of data required.

COMPUTERS, THE INDIVIDUAL, AND SOCIETY

Every human being is a unique complex universe. As life insurance companies well know, this does not prevent the drawing of valid statistical inferences about human populations. The more homogeneous a statistical population, the more accurately can inferences be made about unobserved individuals on the basis of observations on a sample. In between the heterogeneity of *homo sapiens* as a whole, and the homogeneity of identical twins, one senses the possibility of a classification scheme which would divide mankind into groups of sufficient homogeneity to make it possible to draw medically or otherwise useful inferences about a group. The scheme would certainly be complex; and there is no *a priori* reason to assert, for example, that even a million categories would enable one to predict that a particular ten-year-old boy, let us say, will develop angina pectoris between the ages of forty-five and fifty-five, or that a healthy young woman will have a cervical cancer before she is sixty.

The existence of recognizable hereditary characteristics, the small number of clinically distinguishable blood types, the "tendencies" or "predispositions" known to practicing physicians, the experience of plant and animal breeders, the development of strains of laboratory mice like waltzing mice or those who invariably develop cancer, and research in heredity (*e.g.*, on *Drosophila* or *Neurospora*), and other examples, all suggest that very useful classifications probably exist whose numbers of categories are not astronomically large. If a number of the order of ten thousand categories could adequately characterize an individual biologically, such a characterization would be of tremendous use in preventive medicine. Who knows the extent to which cancer and heart trouble could be anticipated, and perhaps prevented or corrected or ameliorated by proper control of diet, activity, or environment? The Rh factor which used to kill babies born to mothers of opposite Rh blood type is now understood well enough to predict the couples to whom this would happen. The tragedy is now avoidable. Who can say what anguish and burdens could be prevented if a similar understanding of congenital idiocy and other genetically based abnormalities or inferiorities could be achieved? Who knows but that the patterns of potential parents of future Einsteins might become recognizable?

Such knowledge, like all knowledge, would be a double-edged sword. Unattractive Orwellian prospects suggest themselves under totalitarian regimes. But under a form of government in which the sanctity of human personality is preserved, one sees the possibility of great good, of healthier individuals, and of a gradually improving human stock. The concept of eugenics is basically good, and if the individual is free to use eugenic information or not, if he is never coerced, and if the program is kept free of bias and fanaticism, only good can come of it.

Setting up such a program would ultimately require a tremendous amount of experimental data, data processing, analysis, and testing of statistical hypotheses. Years might pass before results with practical application could be obtained (though we doubt it), and automation of hundreds of complex laboratory procedures, many not now known, might be necessary before people could be adequately "typed." But if a program developed capable of coming anywhere near the goals described it would bring benefits even greater than those discussed. The old problems of nature vs nurture, or heredity vs environment as determinants of human capability and achievement might become better understood. Factors leading to more valuable and satisfying lives might stand out in bolder relief if the life patterns of a large number of biologically similar individuals were studied with the tools of psychology, sociology, and anthropology. A far deeper understanding of psychosomatic interactions would almost surely result. Cross-cultural studies, as part of the broad program, might give deeper insight into the interactions between individual psychology and the cultural milieu, perhaps to enable us to see how the values and goals of different cultures (which have much to do with the stresses and motivations of an individual) generate different statistics of stress syndromes, diseases, antisocial behavior, or mental conditions in biologically similar individuals. Could such knowledge be obtained, we could begin to develop a science of society which would permit conscious improvement of our customs, values, and motivations, provision of sanifying influences, and tend to maximum development of individual talents and satisfaction. These utopian dreams, distant as they may now seem, need not be unattainable. With computer techniques, automation, and the devoted labors of inspired, determined, and creative people, such dreams will come ever closer to reality.

In these times of international tension, omnipresent powder kegs and atom bombs, one can seriously maintain that the world cannot afford to neglect these possibilities.

CONCLUSION

There is an old witticism about the difference between a scientist and a philosopher. The former applies

increasingly refined techniques to an increasingly specialized and narrowing aspect of the world. As time goes on he knows more and more about less and less. The latter continually generalizes, going into increasingly abstruse abstractions. As time goes on he knows less and less about more and more. In the end, the story goes, the scientist knows everything about nothing, while the philosopher knows nothing about everything. While neither scientist nor philosopher would admit approaching the limits described, there is enough truth to the story to show that it might sometimes be good for the scientist to be a little more philosophical, and for the philosopher to be a little more scientific.

It is hard, perhaps, for the computer scientist or engineer, wrestling with detailed problems of hardware, logical design, budgets, deadlines, maintenance, and operation, to take the philosophic approach very often. The viewpoint here espoused is that once in a while a broad philosophic look at things can be valuable. This paper has therefore studiously avoided technical minutiae and conventional computer applications in favor of bold (we hope not reckless) extrapolations and generalizations to broad problems. What we dream today we achieve tomorrow. It is our responsibility not only to develop day-to-day technology, but also to philosophize enough to find fields where technology will promote advances measured in broad human terms. Just as we have learned to dig better with steam shovels than with our fingers, so must we learn to tackle civilization's problems with the aid of computer techniques rather than with our bare brains.

We cannot wait until the perfect master plan is worked out. Not only would this never be done, but even if some plan were set up as near perfect, it would probably be obsolete before it could be implemented. The same kinds of piecemeal attack that characterize all research would have to be employed. There is much that can be done immediately and in parallel (some being done already), such as 1) mechanizing public health, hospital research, and individual physicians' case history data, 2) developing "common language" techniques to permit easy exchange and consolidation of information gathered by independent agencies, individuals, and countries, 3) making it possible to consolidate scattered data on individuals in order to amass birth-to-death histories, 4) creating techniques of integrated cooperation between individuals, institutions, local, state, federal, and international organizations, 5) encouraging cross-disciplinary research, 6) sponsoring programs designed to uncover areas to work on immediately, 7) perfecting routines for testing increasingly complex statistical hypotheses, 8) formulating special fields so that computer techniques can be used on them, and 9) doing this whenever it becomes feasible. We must press for the nine parts of action to support the one part of philosophy.

The Measurement of Social Change

RICHARD L. MEIER†

IS IT possible to build a synoptic instrument, similar to a telescope or a radar network, for viewing one's own society? How may we interpret the myriads of social activities that are presently undertaken? Preliminary explorations suggest that we need sensing techniques, or transducers, that pick up changes going on inside the society. External indicators, like air photos, are too superficial. We are faced with a problem of discovering what operating characteristics a deep-probing instrument should have so that it may be as practical and useful as possible.

Economists judge change in society by modifications in the makeup of the gross national product and the level of expenditures, political scientists can analyze elections and polls, but sociologists and anthropologists have no cumulative sets of accounts or aggregate indexes. They have had hopes, however, similar to those expressed by Lazarsfeld [3]:

Our economic statistics are today quite well advanced. We know how much pig iron is produced and how much meat is exported every year. But we still have very little bookkeeping in cultural matters. The content of mass media of communication is an important and readily available source of social data, and it will not be surprising if this analysis becomes a regular part of our statistical services in the not too distant future.

These statistics have not yet come into being because the labor cost was high, the time lags were great, and the system description was incomplete, so it has been impossible to state how one set of measurements related to another.

Let us take a brief, searching look at the social system. Society is maintained and changed by the behavior of its members. Intuitively one feels that the basic unit of behavior is the *act*, but acts are not as easily counted and differentiated as particles, molecules, or organisms. Satisfactory data can only be obtained when *actors* are forced to confine their behavior within certain preset specifications or codes, which may be called languages, currencies, habits, or "standard operating procedures." This behavior must be observed in *public* spheres, since the objective, detached observer is missing in private affairs. The latter will require altogether different instruments and techniques for data accumulation, and will not be taken up here.

By far the most promising attack upon the problems of measurability is offered by lumping together small sets of acts into *transactions*. A *social transaction* involves, among other things, the emission of a *message*

together with evidence of its receipt—apparent to an observer, but also, through one or another form of feedback, to the agency responsible for emission.

At any given moment, the population of the society can be divided into *senders*, *receivers*, and *nonparticipants*, much as the economist divides his population into producers and consumers, and each participant must play both roles. The *message* normally contains some information that is novel to the *receiver*, more that is redundant, and some symbols that are quite unintelligible. Some messages are not communicated directly to *receivers*, but are stored in libraries, files, and artifacts where they become a resource embedded in the social environment. Uncoded information may be gleaned from the environment through systematic observation. Scientists, weather observers, diagnosticians, and other professionals have been trained to reduce these phenomena to coded, communicable form (Fig. 1). The information flows should be sampled where the wavy lines occur.

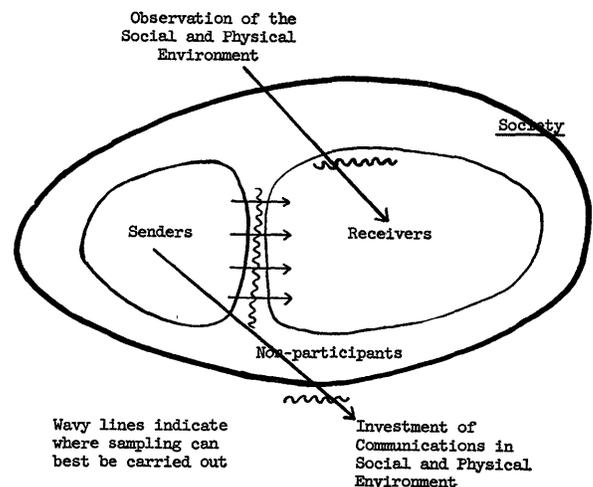


Fig. 1—Communications flows in society.

The term "information" at this point has been used in its intuitive sense. At a later stage, it will be shown that the demand for *information storage* (used now in its technical sense) in our instrument corresponds crudely with the volume of these flows in society—about as well as national income figures represent the combined satisfactions of consumers. The greatest difficulty in the design of our instrument is the conversion of all of the codes for human communication, oral, written, graphic, gestural, musical, etc., into a single code

† Univ. of Michigan, Ann Arbor, Mich.

which is convenient for machine handling. We may have to incorporate human operators, whose skills resemble those of cataloguers in a library, for the more difficult features of translation. Fortunately, as we shall see later, the bulk of the information flow in modern society is in the form of printed language, which seems amenable to automatic sensing, coding, and abstracting (Luhn [4]).

Given the complexities in social communications, how would a representative and comprehensive overview of the social transactions be obtained for our instrument? The mass media—television, radio, magazines, newspapers, books, records, catalogues, direct mail advertising, etc.—could be recorded at the source. Schools, conferences, committee meetings, shop talk, live performances, etc., would have to be random-sampled. But on what basis?

Here we are forced to refer to a fundamental property of modern society. A *sender* can have many simultaneous *receivers*, but any given *receiver* usually accepts messages from but one *sender* at a given moment; on rare occasions he may pay attention to two or three, but no more. The decision as to the completion of a social transaction depends upon the *receiver*. He pays for the message by spending time taking it in. Human time is a moderately scarce commodity, and cannot be wasted indefinitely. People tend to switch dials and scan the newspapers and magazines until they find messages that are interesting to themselves. Message types that gain few receivers tend to be dropped by senders in favor of those which get more attention. Therefore, the social value of broadcast messages may be determined, as a first approximation, by the amount of *time* people devote to them.

Thus a comprehensive time budget of the members of the society—how they allocate time to receiving messages of various kinds, and time to other matters that do not involve social communications—provides a simple, additive criterion of value. We could attach the probable number of receivers, with some estimates of the time and place of reception, to the records of the messages themselves that are held in storage for our instrument.

The formulation of a society-wide time budget has already been explored by Meier [5]. A quantitative description of time-use has applications in public affairs independent of its employment in our system, and the techniques required for making economical measurements already exist.

Economists have found that macro-analysis is greatly assisted by subdividing the economy into such sectors as agriculture, manufacturing, households, etc. The rules for simplifying the accounting may be different in each sector. The choice of sectors in social analysis will depend upon the kinds of reinforcement provided by other approaches to social measurement, such as public opinion evaluation, the Census, and historical analysis. A first guess regarding sectors is provided in Table I.

TABLE I
COMMUNICATIONS-ORIENTED ALLOCATION OF TIME
IN PUBLIC ACTIVITIES

Work School Radio and TV Shopping	Travel Reading Meetings and Parties Dining and Drinking	Play Ritual Personal Services Miscellaneous
Possible Subcategories under Work		
Factory Office Construction and Mining Agriculture and Forestry	Housework Maintenance of Property Services Miscellaneous	

Another feature of our instrument must be introduced. If it is to be economically constructed, it should be decentralized. The headquarters would contain communications which are subject to national distribution, plus some measurements of exports and imports over continental boundaries, while branches would exist in every metropolitan area (Fig. 2).

In the course of proposing a design for this apparatus, we have piled feature upon feature so that it has by now become quite elaborate. It is expected to intercept and store a huge volume of messages, but this is made feasible by eliminating most of the redundancy in social communications, and reducing all the messages to a common code. The instrument attaches weights to these messages according to the number of persons and the amount of time spent receiving them; it indicates the times and places the message is received, and it must store all of this in a permanent record which can be scanned quickly and automatically. Fortunately it need not get every message that is received, but may start modestly by sampling at, say, a one per million rate. As the instrument is refined, and the representation of social change that is required must be finer-grained, the sampling rate may be advanced.

We are now ready to discuss who would use such an instrument and for what purposes. Planners and administrators who must make decisions for the public regarding parks, playgrounds, schools, traffic patterns, and various social services should be able to develop criteria for deciding from studies of trends in social communications and from comparisons with other sources of social data. Advertisers may be expected to develop their craft on the basis of the more detailed measurements of response they would be able to obtain. Politicians should be able to sense better the distribution of sentiment on various issues. Educators may assess the impact of special programs. Changing tastes, the appearance of new patterns of social interaction, and the passage of fads should all be registered as factual data—"how much," "where," and "when." The natural emphasis is upon local public affairs, mass entertainment, and the functioning of work, school, and commerce, because these matters make up the bulk of our communications activity.

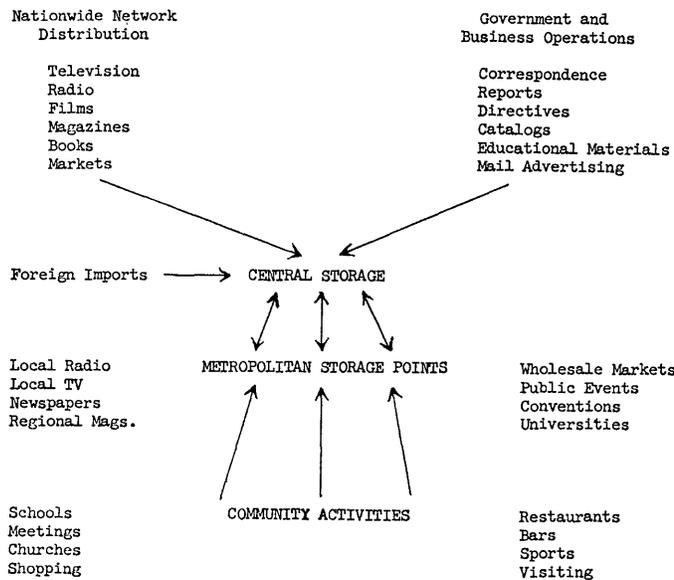


Fig. 2—Organization of social communications as imposed by the location of various activities.

A skilled operator would ask his questions in terms of key words or phrases appearing in the content with a frequency of 10^{-6} to 10^{-8} . They serve as "tracers" of message content as it is spread through the population. Maps and time series can be prepared which show their buildup and decline. More detailed information about the changing attitudes of people may be obtained by reconstructing the contexts within which the key words appeared.

The severest criticism to be made of a representative record of social communications is that the content of the messages tends to be superficial. In many, if not most, social transactions people disguise their true feelings about a subject. An investigator may nevertheless make many nontrivial observations, and can probe more deeply, if he desires, by using the "trial balloon" technique (Fig. 3). An event, closely relevant to the subject of interest, is purposely created—it may be an announcement, an incident, or a rumor. The subsequent wave of "talk" that is stirred up may then be analyzed. The effect that is triggered off provides a good indication of the sensitivity of the public to that issue at that time.

These and other small-scale tactical uses in government and commerce should grow rapidly to the point where large installations may be justified which allow hundreds of simultaneous operators.

The strategic uses of an instrument of this sort are still more interesting. The accumulation of socio-cultural "wealth," for example, may be estimated in a manner analogous to that developed by economists, and the flows of information through society may also be estimated. A very brief outline of the steps involved, and the kinds of conclusions to be obtained, will be presented.

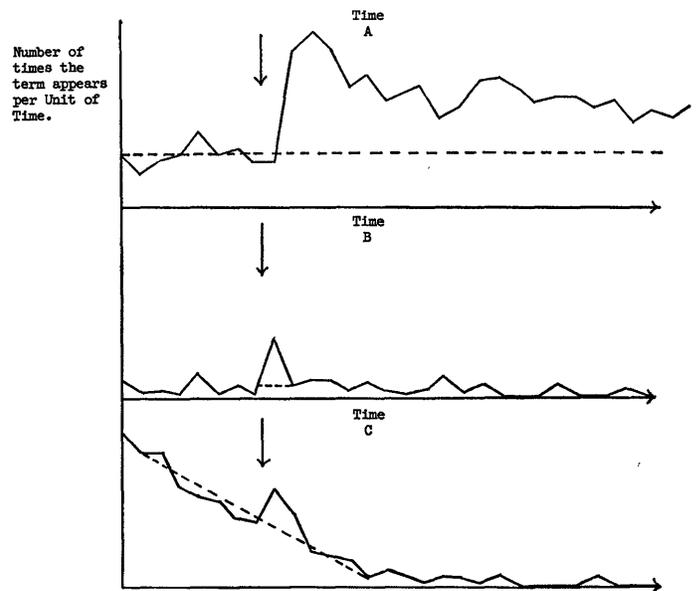


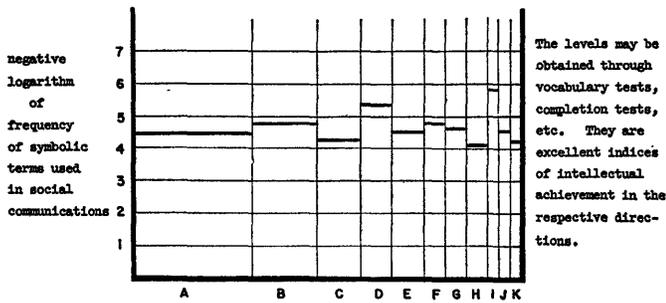
Fig. 3—The "trial balloon" stimulus as revealed by content analysis. Assume the stimulus contains concepts whose treatment in communications uses terms A, B, and C with high probability.

What is the total of all nonredundant information that is transmitted in society for a year? The limitation upon flow is the capacity of the receiver to understand the messages to which he exposed himself. A receiver has a limited repertory of terms. Reasonably good statistics exist only for English vocabulary.

The respective terms that are used in messages can be mapped according to their probability of occurrence in social communications, as in Fig. 4. The abscissa is some arbitrarily defined categorization of meanings, similar to the Dewey decimal system. When this same map is put onto polar coordinates, we can show stages in the development of a receiver as in Fig. 5. The protuberances on the periphery are associated with the specialties engaged in by the person. The map of transition probabilities between terms would have the same appearance.

There is a standing rule in society that a sender should have greater knowledge about the subject of the message than the receiver, if information is to be transmitted. Thus, on the average, the senders are more informed and more expert than the receiver, as shown in Fig. 6. Continued communication would cause the receiver's repertory of terms to grow in the direction of the sender's. He would learn something about the subject. Senders must choose their terms so that they lie on the periphery of the receiver's map, if they are to save time and maintain interest.

We are now in a position to estimate the amount of information flowing that is potentially useful to receivers. Let us define a restricted number of classes of receivers, say about a hundred, each representing a different segment of society, ranging from illiterates to various kinds of professionals, but exhaustive of the



Categories of social communications, spaced according to the allocation of the volume of transmitted terms to each of them respectively.

In American culture A might include terms used in newspapers, magazines and popular books, B those used in conversation, radio, and television, C those employed in business transactions . . . J those used in the fine arts, etc.

Fig. 4—Typical properties of a receiver of social communications.

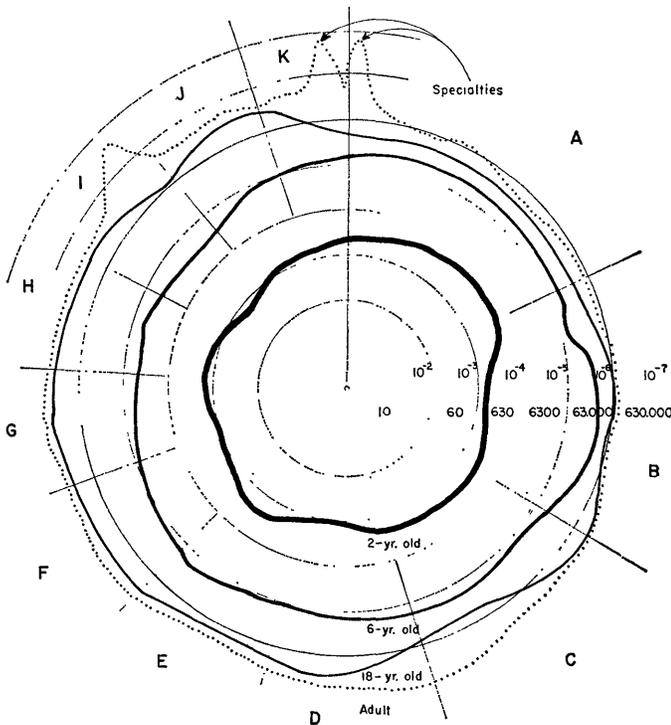
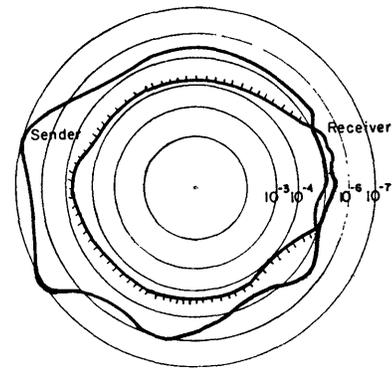


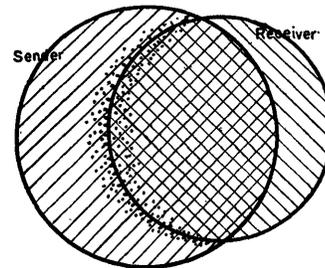
Fig. 5—Typical development of the vocabulary map in an individual. The powers of ten shown are levels for the frequencies with which terms appear. If the Zipf distribution (rank order times frequency is a constant) holds, each shell contains the indicated number of terms. Categories or contexts (A, B, C, . . . K) are assigned by convention as before.

population. Each would have a distinctive map. The messages transmitted in society and stored in our social record must have the receivers which choose to spend time on them classified according to category. Shannon [6] has described a method for using typical receivers for the measurement of redundancy.

Interestingly enough, the significant information flow rate tends to stabilize itself for a given context. Editing the rough spots out of manuscripts has this ef-



(a)



(b)

Fig. 6—Relationships between repertoires in social communications. (a) The receiver expands his map in the indicated direction as a consequence of communication. (b) Simplified maps showing how the sender chooses terms which lie on the boundary of the vocabulary that is shared, if he wishes to optimize the transfer of information.

fect, and directing has this function for mass media. The pauses unconsciously inserted into human speech have recently been shown to work on the same principle [1]. This property, combined with miscellaneous other available information about mass public behavior in American metropolitan areas, enables us to arrive at the first approximation of information flow (Table II).

TABLE II
INFORMATION TRANSMISSION IN METROPOLITAN SOCIETY*
(POPULATION 5,000,000)

Mode of Reception	Time Allocated hours/year	Estimated Receiving Rate bits/minute	Estimated Flow bits/year
Reading	4×10^9	1500	36×10^{13}
Television	3×10^9	500	9×10^{13}
Lecture and Discussion	4×10^9	200	5×10^{13}
Observation of Environment	3×10^9	100	2×10^{13}
Radio	1.5×10^9	300	3×10^{13}
Films	1.6×10^8	800	8×10^{12}
Miscellaneous	5×10^9	100	3×10^{13}
			6×10^{14}

per capita average $\sim 10^8$ bits/year

* Judged in terms of the probable repertoires of receivers, not in the accepted sense used in information theory. Possibly a new term should be coined for information distributed over a population.

Comparisons between the poorest and richest metropolitan areas can also be exceedingly suggestive (Table III). Observers now agree that socio-cultural growth parallels economic growth but the introduction of measurements suggests that social communications must either precede economic growth or grow more rapidly than income. Apparently an expansion of socio-cultural activity is a necessary but not sufficient precursor of economic development.

TABLE III
INCOME AND INFORMATION FLOW EXTREMES IN URBAN SOCIETY

	San Francisco	Addis Ababa or Jakarta
Income	\$3000 capita/year	\$150 capita/year
Non-redundant* information receipt	$\sim 10^8$ bits/year	$\sim 10^6$ bits/year

* Again in terms of probable repertoires of receivers. This assumes that 70–80 per cent of residents in the poorer cities are illiterate.

The heavy volume of information transmitted by reading is highly significant. A society like our own which is increasingly white collar reads more at work and at home. There are limits to human ability to receive information, however, which are believed to be in the neighborhood of 10^9 bits per capita per annum for a population with the present distribution of mental capacities. At the present estimated rate of gain, this

theoretical saturation level is likely to be reached within two generations. The prospect is startling enough to cause us to investigate more closely the stresses associated with communication saturation in human organizations.

My feeling is that our instrument is already technically feasible. Simple calculations show that sampling social communications at a rate of ten parts per million presents storage requirements within range of existing equipment, but the desired degree of access remains unclear. Message collection in the field is not a problem, but the programming for storage and the cataloguing of nonverbal materials has been inadequately developed. Much experimental work and formal analysis will be required before a truly comprehensive cross section of social change can be achieved.

REFERENCES

- [1] F. Goldman-Eisler, "Speech production and language statistics," *Nature*, vol. 180, p. 1497; December 28, 1957.
- [2] M. Kochen and M. Levy, "The logical nature of an action scheme," *Behavioral Science*, vol. 1, pp. 265–289; October, 1956.
- [3] P. Lazarsfeld, "Communications research and the sociologist," in "Current Trends in Social Psychology," W. Dennis, ed., University of Pittsburgh Press, Pittsburgh, Pa., pp. 232–233; 1948.
- [4] H. P. Luhn, "A statistical approach to mechanical encoding and searching of literary information," *IBM J. Res. Dev.*, vol. 1, pp. 309–313; October, 1957.
- [5] R. L. Meier, "Human time allocation; indices for the measurement of social change," *J. Amer. Inst. Planners*, vol. 25, pp. 27–33; February, 1959.
- [6] C. E. Shannon, "Prediction of entropy of printed English," *Bell Sys. Tech. J.*, vol. 30, pp. 50–64; January, 1950.

Simulation of Sampled-Data Systems Using Analog-to-Digital Converters

MICHAEL S. SHUMATE†

INTRODUCTION

UNTIL recently, systems simulation problems could generally be split into two classes: problems requiring only an analog computer for solution and problems requiring only a digital computer for solution. Any general problem has a number of characteristics which adapt it to either one or the other method of solution.

A systems problem is most readily adapted to analog computer simulation when it requires a relatively short solution time on the computer and a relatively inaccurate solution is acceptable; has relatively "high" frequencies; and has nonlinearities such as, for example,

saturation, deadzone, or hysteresis. To be adapted to digital computer simulation, a systems problem usually possesses relatively low frequencies and requires a long solution time; can be adapted to an iterative form of simulation without introduction of an instability (this is usually implied by a lack of nonlinearities such as those mentioned above); and has a range of variable which exceeds that possessed by an analog computer solution.

Certain problems involving combinations of both groups of properties may often be split into two separate problems, one involving high-frequency nonlinear effects, and one involving low-frequency effects.

An example of such a problem is the simulation of the flight of a liquid-propelled ballistic missile. The missile's

† Consultant for Space Technology Labs., Inc., and California Inst. of Tech., Pasadena, Calif.

reactions to external and internal forces can be simulated on an analog computer, and the motion of the missile along a trajectory can be simulated on a digital computer.

However, recent developments in control systems, particularly systems using sampled or discrete data, have generated a third class of simulation problems; this class of problem usually involves both sampled and continuous information, has nonlinearities and no general dividing line between high and low frequencies, and usually has a range of variable which exceeds that possessed by an analog computer. An example of such a problem is the simulation of an inertially guided missile. The guidance and control systems must have rapid response characteristics, and the system may involve a special-purpose digital computer, whose program makes it act like a multimode, adaptive controller.

SIMULATION OF SAMPLED DATA SYSTEMS

Large analog simulation facilities are thus faced with the problem of obtaining reasonable representations of such systems, and must thus have the capability of at least sampling continuous information, and hopefully be able to perform some intelligent and useful operations on such information.

Several straightforward methods of performing sampled-data simulation are presently available. Sampling may be accomplished by a so-called hold-amplifier [see Fig. 1(a)] or a passive hold circuit [see Fig. 1(b)]. These circuits are not easily adapted to simulation of digital computer operations, because of difficulty involved in holding past values of sampled information. This difficulty is partially eliminated by using transfluxors¹ as hold devices.

The most general method for simulation of sampled-data systems is to use analog-to-digital converters to connect an analog computer to a general-purpose digital computer (see Fig. 2). In such an installation, one or more analog-to-digital converters are used to sample continuous information and present it to a digital computer. Several more analog-to-digital converters are used open loop as digital-to-analog converters to present and hold digital information to the analog computer. Two major disadvantages of this system are:

- 1) The difficulty in starting the two computers simultaneously, and
- 2) The difficulty in scheduling operation time on the digital computer, since in most installations digital computing time is a premium quantity.

The first of these difficulties may be remedied by equipping the digital computer with an "interrupt" feature, and the analog computer with a "start" control. The interrupt feature allows the digital computer to

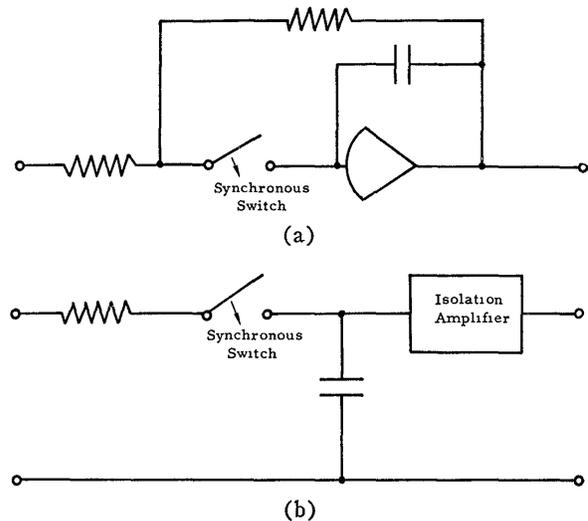


Fig. 1—Two examples of sample-and-hold circuits.

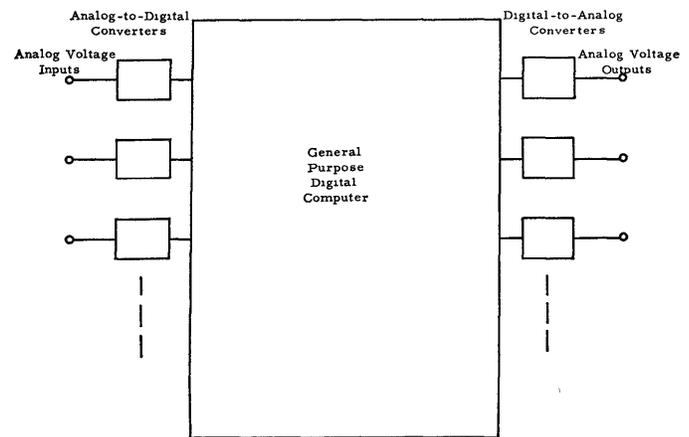


Fig. 2—General-purpose sampled-data simulator.

break its program when a pulse (perhaps a sample pulse) is received, and jumps to a special subroutine used for external communication purposes. The start control allows the analog computer hold relays to actuate simultaneously with the same pulse that first interrupted the digital computer.

The second difficulty is easily overcome if one is willing to pay for the digital computer time. It seems a little ridiculous, however, to use a large general-purpose digital computer if the digital program involved is relatively simple. For example, in order to use this system to simulate a sample-and-hold, the digital computer must be programmed to perform a transfer of a number from an analog-to-digital converter to a digital-to-analog converter, and would thus sit idle during most of a sample period.

It therefore becomes evident that such a large, general-purpose sampled data simulator should only be used for complex, sophisticated problems, and some auxiliary equipment should be developed to simulate simpler sampling operations.

¹ J. A. Rajchman and A. W. Lo, "The transfluxor," Proc. IRE, vol. 44, pp. 321-332; March, 1956.

EQUIPMENT

The construction of an auxiliary sampled-data simulator is, of course, dictated by equipment present in a simulation facility. The facility currently available to the author consists of a 300 amplifier Electronic Associates Analog Computer, an EPSCO Addaverter conversion system, and a Remington-Rand Univac Scientific Digital Computer, Model 1103A.

The approach used in the construction of an auxiliary sampled data simulator was based on the fact that the facility currently possessed equipment which would both sample analog voltages and hold analog voltages. This equipment, the Addaverter (see description below), would, of course, be in use part of the time as a communication link to the 1103A, but its time schedule was sufficiently open to warrant thinking of using it for other purposes.

The Addaverter is used to simulate a group of parallel sample-and-hold channels. A sample-and-hold is obtained by effectively placing an analog-to-digital converter in series with a digital-to-analog converter. The analog-to-digital converter is caused to sample its input voltage, the resulting digital number is then transferred to the digital-to-analog converter, and converted back into a voltage, which remains unchanged until the complete cycle is repeated again. Note that delaying of the conversion of the number back into a voltage is equivalent to delaying the sampled values of the input voltage and is useful for simulation of transportation lag, etc.

In order to obtain a better description of the implementation of the above concept into a complete sampled data simulator, some description of the Addaverter is necessary.

The Addaverter consists of 30 analog-to-digital converters (hereafter abbreviated ADC), 15 used as such, and the other 15 used open loop, as digital-to-analog converters (abbreviated DAC).

All 15 ADC's sample² the voltages present at their inputs simultaneously, under the control of a central sample controller, which is triggered by a single pulse, called a sample pulse; the resulting digital numbers remain intact in the memory of each ADC until the next sample pulse occurs.

The DAC's operate individually, each converting the digital number previously stored in its memory to an analog voltage when it receives a pulse, called a present pulse. The voltage at a DAC's output remains unchanged until a new number has been read into its memory *and* a present pulse has been received. A single present pulse input and 15 pulse inhibit inputs are used, instead of 15 pulse inputs.

In order to transfer the digital numbers from the ADC memories to the DAC memories without using the digital computer, an additional piece of equipment, nor-

mally used for Addaverter maintenance purposes, is employed. One particular operational mode of this equipment, when triggered by a single pulse, transfers the numbers stored in the ADC memories into the DAC memories in a sequential fashion: the content of the first ADC memory is transferred into an auxiliary register, and then into the memory of the first DAC;³ the process is then repeated for each of the other 14 ADC-DAC channels. Provision is made to prevent the number transfer for any individual channel by energizing an inhibit gate associated with that channel.

These three units, the 15 ADC's, the 15 DAC's used as DAC's, and the number transfer equipment, comprise the basic sampled-data simulator. In order to obtain satisfactory operation, it is necessary to supply a burst of three pulses each time it is desired for the simulator to sample. Another piece of equipment was constructed to supply the pulse burst necessary to control the simulator. This equipment consists of necessary control logic for the simulator, and a large quantity of pulse and dc logic with a prepatch capability, to permit flexible operation of the entire system. A source of timed pulses is available, to use as system clock; a "start" system is available, to synchronize starting of analog simulations with the clock pulses.

An ADC-DAC channel may thus be used to sample-and-hold an analog voltage with either of two transfer functions:

$$H_1(s) = \frac{1}{s} (1 - e^{-sT}), \text{ and} \quad (1)$$

$$H_2(s) = \frac{1}{s} (1 - e^{-sT})e^{-s\tau} \quad \tau \leq T \quad (2)$$

where T is the sampling period, and τ a delay time.

No problems are incurred if it is desired to have all 15 ADC-DAC channels operating in the same mode. However, it is sometimes desired to simulate multirate sampled-data systems, or monorate systems that have several samplers each with a different time delay. No difficulties would arise if each ADC-DAC channel operated independently of every other channel. This is not the case, however, since all channels sample simultaneously.

The simulator control equipment was so designed to permit simulation of up to four different sample-and-hold operations; however, their operating frequencies must be restricted to integral multiples of some frequency.⁴ A block diagram of the simulator control is shown in Fig. 3. The simulator control has a set of sample pulse inputs, a set of present pulse inputs, and a set of "enable" outputs, all located on the prepatch panel. The inhibit inputs for the present pulse (DAC

³ This process requires 40 μ sec.

² The Addaverter uses the "ripple-down" method for analog-to-digital conversion and requires 180 μ sec to complete one conversion.

⁴ This restriction is brought about by the simultaneous sampling of the ADC's. It is possible to use two asynchronous frequencies, but some external pulse logic is necessary to prevent double pulsing of the sample control.

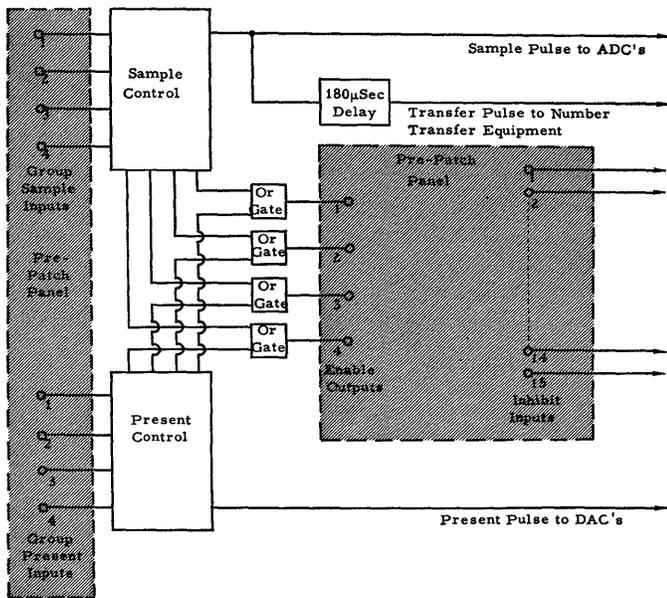


Fig. 3—Block diagram of simulator control logic.

inhibit), and inputs for the transfer inhibit are wired in parallel⁵ and brought up to the prepatch panel. The four "enable" outputs, one for each operating group, must be patched to the inhibit inputs of the channels, as predetermined by the programmer. If a particular channel is to operate according to the group 1 mode, its inhibit input must be patched to the group 1 enable. Several channels may operate from the same group enable.

A particular group's operating mode is completely determined by what pulses are fed to its sample input and its present input. For example, to operate a group as a sample-and-hold [see Fig. 4(a)], a clock pulse must be fed to the group sample input each time it is desired for a sample to occur. A present pulse must then be fed to the group present input 815 μ sec after each sample pulse. The sample input pulse causes all group enable outputs to be reset, then the particular group enable (whose input was pulsed) is set, and the ADC's are caused to sample (all 15 sample); 180 μ sec later, the number transfer is initiated. (Only those channels associated with the operating group have numbers transferred.) The present input pulse, which must be delayed at least 815 μ sec to allow the sample and number transfer to be completed, then causes the DAC's connected to the operating group to present the numbers transferred as voltages.

Note that the sample-and-hold is actually a sample-and-hold plus a short delay; this delay is not long enough to introduce unwanted effects in the majority of problems simulated. If it is desired to introduce more delay, it is only necessary to increase the delay of the group present input pulse. The maximum delay possible

⁵ Logically, if a number is transferred into a DAC, it is done so with the intention of presenting it.

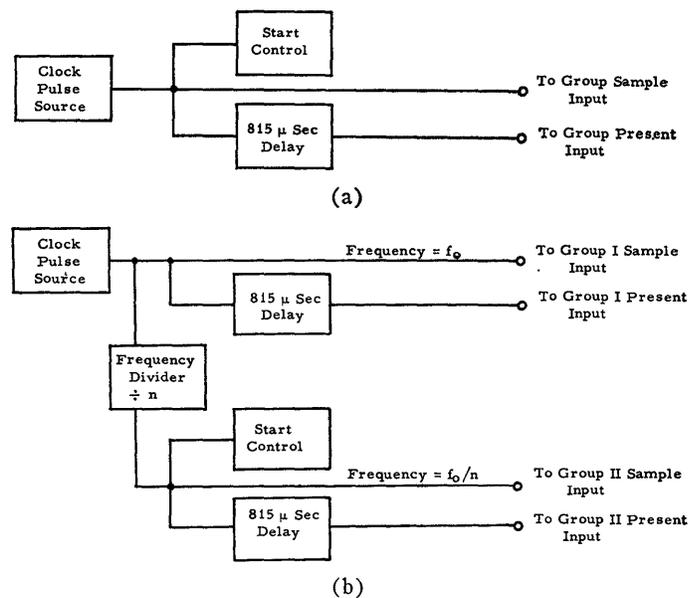


Fig. 4—Example pulse logic diagrams.

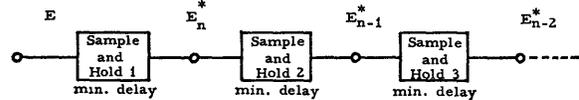


Fig. 5—Wiring of sample-and-hold channels to simulate long time delays.

for a group present input pulse is the sampling period for the particular group.

To operate two groups at different, related frequencies, the group running at the higher frequency is operated as described above, but the clock pulses for the lower frequency group must be obtained by dividing the higher frequency clock pulses by some integral number. [See Fig. 4(b).] The start control input is shown in Fig. 4(a) and 4(b).

A great many other modes of operation are possible, but will not be presented here because of space restrictions.

SIMULATION OF SIMPLE DIGITAL COMPUTER OPERATIONS⁶

Simulation of sampled-data systems which incorporate a digital computer may be accomplished by using sample-and-hold channels to simulate the digital computer's transfer function (provided the transfer function is not too complicated). Simple digital filters, second- or third-order difference equations, etc., may be simulated.

If a group of ADC-DAC channels are wired in series [the output of the first wired to the input of the second (see Fig. 5, with the control logic being pulsed by logic circuitry given in Fig. 4(a) of the example logic diagrams, the variable delay set to 815 μ sec, and all channel inhibit inputs wired to the operating group enable

⁶ J. R. Ragazzini and G. F. Franklin, "Sample-Data Control Systems," McGraw-Hill Book Co., Inc., New York, N. Y.; 1958

output], then it can be easily shown that successive channel outputs are each delayed from the previous channel output by one sample period.

E_n^* in this treatise will be used to denote the *sampled and held value* of E ; the subscript n refers to the n th sample pulse.

Suppose E is zero, and has been for a considerable length of time. Then E_n^* , E_{n-1}^* , etc., will all be zero. Now suppose that during the interval between the n th and the n th+1 samples, E changes to some nonzero variable value. Then, after the present pulse associated with the n th+1 sample, channel 1's output will change to E_{n+1}^* . However, during the sampling associated with the n th+1 sample, channel 1's output was still zero, hence the output of channel 2 will remain at zero after the n th+1 present pulse.

After time $n+2$, the output of channel 2 will have the value E_{n+1}^* , since that value was still at the output of channel 1 when the n th+2 sampling occurred. The output of channel 3, E_n^* , will still be zero. (This operation is a consequence of the fact that each sample-and-hold channel has a slight transportation lag.)

Therefore, a series string of sample-and-hold channels will act like a delay line which propagates values of E that have been sampled and held. Further consideration will show that this is equivalent to the way a digital computer would remember past values of a variable.

Thus

$$\begin{aligned} \text{Output of channel 1} &= E_n^* \\ \text{Output of channel 2} &= E_n^* e^{-sT} \\ \text{Output of channel 3} &= E_n^* e^{-2sT}. \end{aligned}$$

Using z transform notation ($z = e^{sT}$; see Ragazzini and Franklin⁶)

$$\begin{aligned} \text{Output of channel 1} &= E_n^* \\ \text{Output of channel 2} &= E_n^* \frac{1}{z} \\ \text{Output of channel 3} &= E_n^* \frac{1}{z^2}. \end{aligned}$$

The first channel has the transfer function

$$G_1 = \frac{1}{s} (1 - e^{-sT}) \quad (3)$$

and each succeeding channel has the transfer function

$$G_2 = G_3 = \dots = \frac{1}{z}. \quad (4)$$

Suppose it is desired to simulate the transfer function

$$G(z) = \frac{z(a_0z + a_1)}{z^2 + b_1z + b_2} = \frac{E_0^*}{E_i^*}. \quad (5)$$

Solving for E_0^* ,

$$E_0^* = a_0E_i^* + \frac{a_1E_i^*}{z} - \frac{b_1E_0^*}{z} - \frac{b_2E_0^*}{z^2}. \quad (6)$$

The block diagram for this is shown in Fig. 6. The implementation of a second-order difference equation can be accomplished with three sample-and-hold channels instead of four; the diagram in Fig. 6 was chosen because it made the error analysis more expedient.

Accuracy of Simulation

Each Addaverter unit (ADC or DAC) is accurate to within ± 0.1 per cent of its nominal input (or output) voltage. Therefore, the voltage out of each sample-and-hold channel is accurate to within 0.2 per cent of the voltage put into the channel. Because of the 0.2 per cent inaccuracy of the Addaverter, some uncertainty in the solution of a difference equation may arise. The following discussion treats the limitations caused by this inaccuracy.

A sample-and-hold channel may be visualized as having its output made up of the sum of a voltage which is identical to the voltage at the input during the last sample and an unknown random voltage. This is graphically explained in Fig. 7.

Using Fig. 7 as a model sample-and-hold channel, the complete diagram for a second-order difference equation is shown in Fig. 8.

The expression for the output is

$$\begin{aligned} E_0^* &= E_i^*G(z) + R_I^*G(z) + R_{II}^* \frac{a_1}{1 + \frac{b_1}{z} + \frac{b_2}{z^2}} \\ &\quad - R_{III}^* \frac{b_1 + \frac{b_2}{z}}{1 + \frac{b_1}{z} + \frac{b_2}{z^2}} - R_{IV}^* \frac{b_2}{1 + \frac{b_1}{z} + \frac{b_2}{z^2}}. \quad (7) \end{aligned}$$

Further analysis would be impossible without some simplifying assumptions about the character of R_i .

An R_i is a function of the voltage at the channel input and hence cannot be assumed Gaussian. Observations taken from the Addaverter have shown that the R_i are, to a first approximation, constant offsets. Therefore, let

$$R_i^* = \frac{z}{z-1} \epsilon_i \quad (8)$$

where ϵ_i = constant offset associated with the i th channel.

Furthermore, to make analysis expedient, the second-order difference equation is assumed to be that of a damped sinusoid:

$$G(z) = \frac{ze^{-\alpha T} \sin aT}{z^2 - 2ze^{-\alpha T} \cos aT + e^{-2\alpha T}}. \quad (9)$$

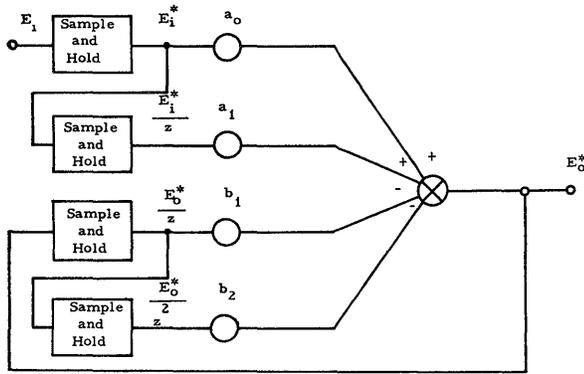


Fig. 6—Block diagram for simulation of second-order difference equation.

Substituting (8) and (9) into (7),

$$E_0^* = E_1^*G(z) + \frac{z}{z-1} \epsilon_I G(z) + \frac{z}{z-1} \epsilon_{II} \frac{z^2 e^{-\alpha T} \sin aT}{z^2 - 2ze^{-\alpha T} \cos aT + e^{-2\alpha T}} + \frac{z}{z-1} \epsilon_{III} \frac{2z^2 e^{-\alpha T} \cos aT - e^{-2\alpha T} z}{z^2 - 2ze^{-\alpha T} \cos aT + e^{-2\alpha T}} - \frac{z}{z-1} \epsilon_{IV} \frac{e^{-2\alpha T} z^2}{z^2 - 2ze^{-\alpha T} \cos aT + e^{-2\alpha T}} \quad (10)$$

The first term of (10) represents the desired output of the simulation. The remaining terms represent the error caused by the offsets in the channel outputs. The error is split into two parts: a constant term and an oscillatory term. These two parts are evident in the partial fraction expansion of the error terms of (10).

$$\text{Error of } E_0^* = \frac{[(e^{-\alpha T} \sin aT)(\epsilon_I + \epsilon_{II}) + (2e^{-\alpha T} \cos aT)(\epsilon_{III}) - (e^{-2\alpha T})(\epsilon_{III} + \epsilon_{IV})] z}{1 - 2e^{-\alpha T} \cos aT + e^{-2\alpha T}} \cdot \frac{z}{z-1} + \frac{\left\{ \begin{aligned} &(e^{-\alpha T} \sin aT)(\epsilon_I) + (e^{-\alpha T} \sin aT)(2e^{-\alpha T} \cos aT - e^{-2\alpha T})(\epsilon_{II}) \\ &+ [(2e^{-\alpha T} \cos aT - e^{-2\alpha T})(2e^{-\alpha T} \cos aT) - e^{-2\alpha T}](\epsilon_{III}) \\ &- (2e^{-\alpha T} \cos aT - e^{-2\alpha T})(\epsilon_{IV}) \end{aligned} \right\} z^2 + [(-e^{-2\alpha T} e^{-\alpha T} \sin aT)(\epsilon_I + \epsilon_{II}) - (2e^{-2\alpha T} e^{-\alpha T} \cos aT)(\epsilon_{III}) + (e^{-2\alpha T})(\epsilon_{III} + \epsilon_{IV})] z}{(1 - 2e^{-\alpha T} \cos aT + e^{-2\alpha T})(z^2 - 2ze^{-\alpha T} \cos aT + e^{-2\alpha T})} \quad (11)$$

The first term of (11) contributes a constant error, which would be most noticeable if aT were small. If $a=0$, then the error due to the first term would be

$$\text{error} = \frac{(2e^{-\alpha T} - e^{-2\alpha T})\epsilon_{III} - e^{-2\alpha T}\epsilon_{IV}}{1 - 2e^{-\alpha T} + e^{-2\alpha T}} \quad (12)$$

Suppose ϵ_{III} and ϵ_{IV} are offsets which are of such po-

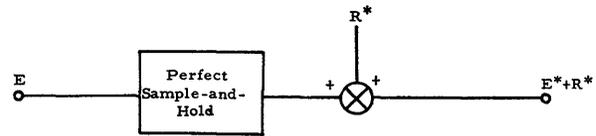


Fig. 7—Model of a noisy sample-and-hold channel.

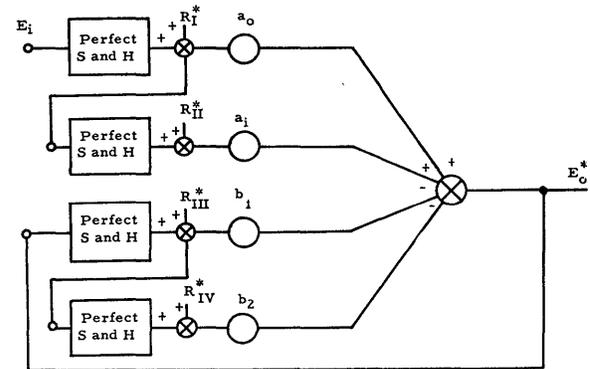


Fig. 8—Block diagram of second-order difference equation, including random disturbances.

larity that they give the largest error: namely, suppose $\epsilon_{III} = +\psi_f$ and $\epsilon_{IV} = -\psi_f$ where ψ_f is the average channel offset. Furthermore, suppose the total error of simulation is to be no larger than $100\psi_f$ (*i.e.*, the error-to-offset ratio is 100).

Then substituting into (12)

$$100\psi_f \geq \frac{2e^{-\alpha T}\psi_f}{1 - 2e^{-\alpha T} + e^{-2\alpha T}} \quad (13)$$

Solving for $e^{-\alpha T}$,

$$e^{-\alpha T} \leq 0.868. \quad (14)$$

$e^{-\alpha T}$ is the z -plane location of the double pole for the case when $a=0$. Hence, if it is desired to have an error no greater than 100 times the average channel offset, then the real part of the pole locations, for small aT , must be less than 0.868.

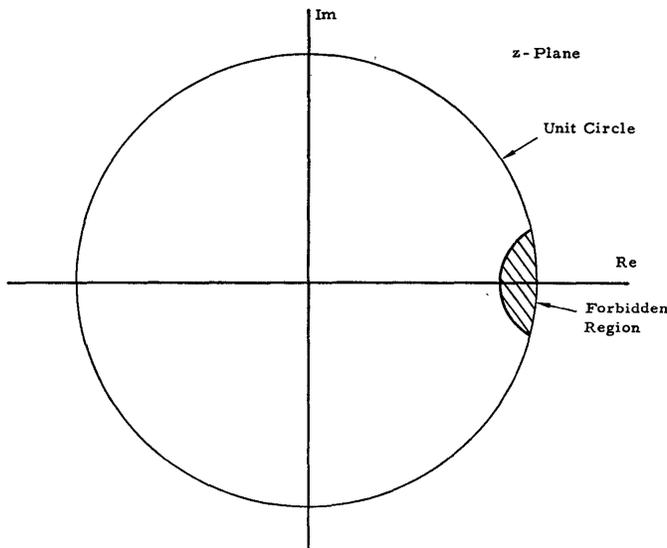


Fig. 9—Forbidden region of z plane for poles of a second-order difference equation.

Now consider the case where $\alpha=0$. The major source of error in the solution comes from the second term of (11). This error is

$$\text{error} = \frac{\{[\epsilon_{\text{I}} + (2 \cos aT - 1)\epsilon_{\text{II}}] \sin aT + [(2 \cos aT - 1)(2 \cos aT) - 1]\epsilon_{\text{III}} - (2 \cos aT - 1)\epsilon_{\text{IV}}\}z^2 + \{-(\epsilon_{\text{I}} + \epsilon_{\text{II}}) \sin aT - (2 \cos aT - 1)\epsilon_{\text{III}} + \epsilon_{\text{IV}}\}z}{(2 - 2 \cos aT)(z^2 - 2z \cos aT + 1)} \quad (15)$$

By assuming that $\cos aT$ is approximately unity, and that $\epsilon_{\text{III}} = \psi_f$ and $\epsilon_{\text{IV}} = -\psi_f$ (as before) and neglecting $(\epsilon_{\text{I}} + \epsilon_{\text{II}}) \sin aT$, the numerator of (15) reduces to

$$2(z^2 - z)\psi_f$$

which is the same as the numerator of the transform for $\cos aTn$. Hence, this error contributes

$$\frac{\psi_f}{1 - \cos aT} \cos aTn$$

to the output of the simulation. If as before, it is desired that this contribution be less than 100 times the average channel offset, ψ_f , then

$$100\psi_f \geq \frac{\psi_f}{1 - \cos aT} \quad (16)$$

Solving for aT ,

$$aT \geq 8.1^\circ \quad (17)$$

Thus, using (14) and (17), a region of the z plane can be enclosed, and dubiously named a forbidden region for the existence of poles of a second-order system. Fig. 9 illustrates this forbidden region.

Similar calculations made for an error contribution which is not to exceed 10 times the average channel offset, ψ_f , yield the limits

$$e^{-aT} \leq 0.641, \text{ and} \\ aT \geq 25.9^\circ \quad (18)$$

A similar analysis for a first-order lag may be made by letting $a_1 = b_2 = 0$ and $a_0 = 1$ in (5). For an error-to-offset ratio of 100

$$100\psi_f \geq \frac{b_1\psi_f}{1 - b_1} \quad (19)$$

Solving for b_1 ,

$$b_1 \leq 0.99 \quad (20)$$

Hence, for a first-order lag, the pole should be located at points less than 0.99 for a ratio of 100. This means that this technique of simulating difference equations cannot be used to do integration (the above case, for instance, where $b_1 = 1$).

Figs. 10 and 11 are illustrations of the type of solution uncertainty caused by the sample-and-hold channel offsets. For both figures, an undamped cosine function was used and the average channel offset was 0.02 volt. For Fig. 10, the value of aT was 5° , locating the poles inside the forbidden region; for Fig. 11 $aT = 10^\circ$, locating the poles just outside the forbidden region. The difference in solution uncertainty is clearly illustrated. The same type and magnitude of uncertainty arises in a damped sinusoid solution.

It is indeed unfortunate that the forbidden region exists where it does, since it excludes the area of the z plane most commonly used. Time scaling may sometimes be employed to shift system poles to more favorable locations, but this often leads to difficulties in the analog computer portion of a simulation, since frequency must be scaled up.

CONCLUSIONS

Simulation facilities possessing general-purpose sampled data simulators in the form of a conversion system and a digital computer may require an auxiliary sampled-data simulator for simulation of simpler system problems.

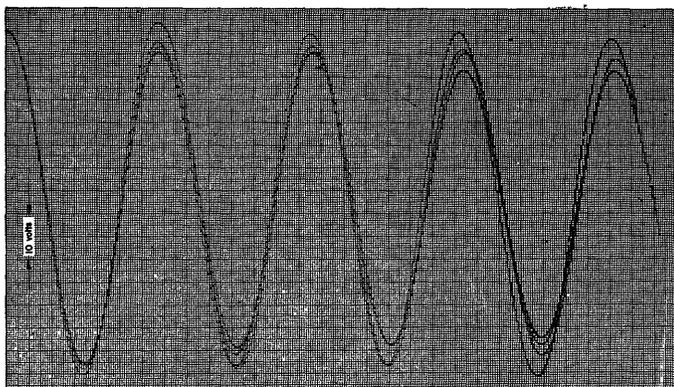


Fig. 10— $\cos aTn.aT=5^\circ$. Average channel offset=0.02 volt.

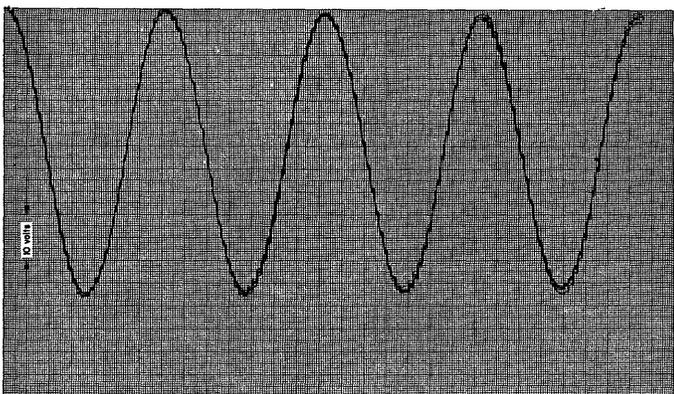


Fig. 11— $\cos aTn.aT=10^\circ$. Average channel offset=0.02 volt.

An auxiliary simulator which uses the main portions of the analog-to-digital conversion system satisfies most of the requirements that a sampled-data simulator must meet. It has the advantage that it uses the conversion system during otherwise idle periods.

The simulator consists basically of a number of sample-and-hold channels which, subject to certain inaccuracies, may introduce errors into a simulation. The magnitudes of these errors can be determined in cases only with prior knowledge of the complete system.

In the simulation of digital transfer functions, the effects of these inaccuracies on the representation of transfer functions may be analyzed for any given transfer function. Given some bound on the over-all error, a region of existence of the z -plane poles of the transfer function may be outlined.

ACKNOWLEDGMENT

The author is indebted to the members of the Simulation Department of Space Technology Laboratories for their assistance and guidance in the development of the auxiliary sampled-data simulator. Eli Anfenger of EPSCO, Inc., is credited with the original idea for the simulator.

The author is particularly grateful to R. P. Adams, who designed the control logic; E. A. Goldberg, who assisted with the error analysis; and George Chitel, who supervised the construction of all of the additional equipment.

FOXY 2: A Transistorized Analog Memory for Functions of Two Variables

L. J. KAMM[†], P. C. SHERERTZ[†], AND L. E. STEFFEN[†]

FOXY 2 is a high-speed function generator with two independent variables for use in analog computers and simulators.

FOXY 2 stands for *Function Of X Y*, 2nd type. It is a high-speed transistorized version of FOXY 1, which is a relay device.

The mathematical approach is shown in Fig. 1. The x, y plane is ruled in a grid and the value of z at each corner point is defined in the memory.

For any point (x, y) the 4 surrounding corner values z_A, z_B, z_C, z_D are switched to the output circuit, and $z(x, y)$ is obtained by so-called bilinear interpolation as follows: z'' is obtained by linear interpolation between z_A and z_B , and z' is obtained by linear interpolation between z_C and z_D . Then z is obtained by linear interpolation between z' and z'' .

The memory comprises a jack for each corner point and a set of fixed voltage jacks. To record a value of 47 for $z_{5,9}$ for example, corner jack 5, 9 is wired to voltage jack 47. To provide for wiring several corner jacks to a single voltage jack, the corner jacks are actually twin

[†] Convair, San Diego, Calif.

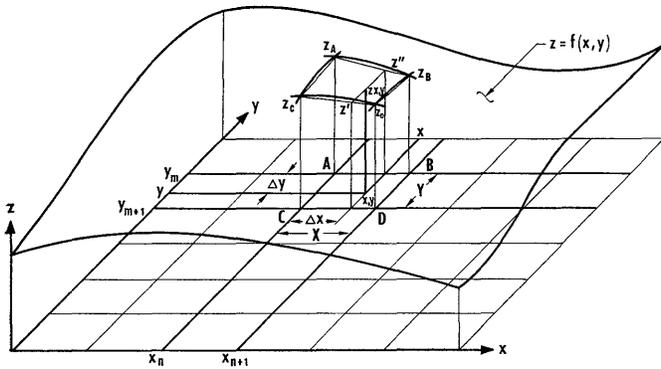


Fig. 1—Mathematical description $z=f(x, y)$.

jacks and chain wiring is used from a voltage jack to a first-corner jack to a second-corner jack to a third-corner jack, etc.

There are 201 fixed-voltage jacks having values from -100 through 0 to +100.

The x, y plane is divided up by 21 rows and 21 columns of corner points, so that z is defined at 441 points.

A bus is provided for each x and each y grid line. A corner-point voltage is switched out from the memory when both its x and its y busses are energized. To provide for the simultaneous selection of all four corners of a square needed for interpolation within the square, two adjacent x busses and two adjacent y busses are energized at the same time. Energizing x busses 3 and 4 and y busses 1 and 2 switch out the four voltages on corner jacks (3, 1), (3, 2), (4, 1), and (4, 2).

Ambiguity is prevented by the even-odd segregation scheme shown in Fig. 2. All corner points in the x, y plane are classified as $A, B, C,$ or D . Every square has one and only one corner of each classification, although not in the same relative positions. The switching circuits for each classification are kept separate. Thus, in the above example corner

- (3, 1) is class A
- (3, 2) is class C
- (4, 1) is class B
- (4, 2) is class D .

The simultaneous operation of busses $x_3, x_4, y_1,$ and y_2 connect one and only one corner jack to each of four memory-output wires, $A, B, C,$ and D .

A block diagram of the system is shown in Fig. 3. The x and y input voltages each go to a corner selector which energizes the appropriate grid busses. The grid busses selectively operate the 21×21 array of corner switches. These connect the corner voltages (z values) to the output interpolator, 4 at a time (the A corner, B corner, C corner, and D corner of a grid square). The corner selector circuits also send signals via the pre-interpolators to the output interpolator which then generates a weighted average of the four corner voltages to produce the output, z .

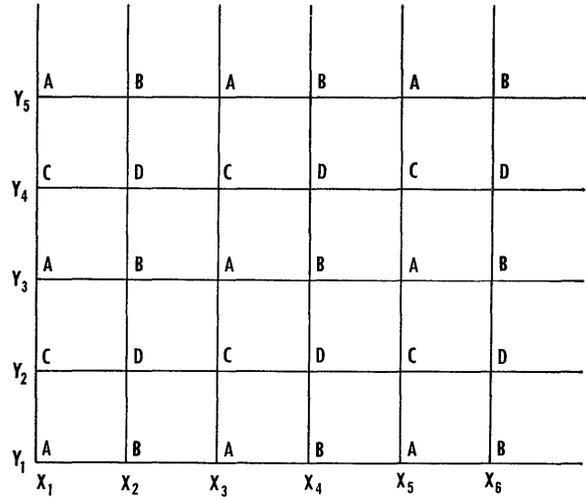


Fig. 2—Even-odd corner classification.

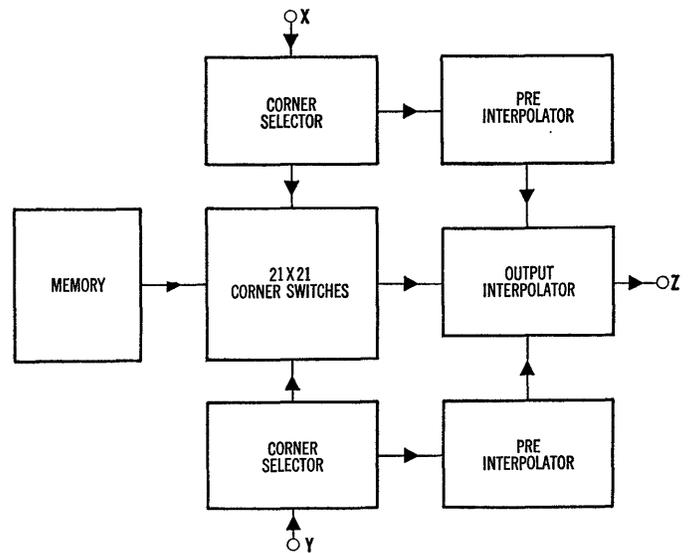


Fig. 3—Block diagram.

The corner point switching system is shown in Fig. 4. Here the outputs of the corner selectors of Fig. 3 are designated matrix decoders. The grid busses are designated x_1, x_2, \dots, x_{21} and y_1, y_2, \dots, y_{21} . Each corner switch is an AND gate comprising a 2N521A transistor whose base is connected to a y bus and to an x bus by 1N116 diodes and to a bias voltage by a resistor.

When both its busses are energized, a transistor is turned on and its corner-jack voltage is imposed on one of the four output wires $A, B, C,$ or D . All of the A -corner switches are connected to output wire A , and corners $B, C,$ and D are correspondingly related to output wires $B, C,$ and D . Thus, when two adjacent x busses and two adjacent y busses are energized, a corner voltage is imposed on each of the four output wires.

Each corner selector contains an analog-to-digital converter in which each count corresponds to a 10-volt

A Time-Sharing Analog Computer*

JOHN V. REIHING, JR.†

I. INTRODUCTION

THE transient behavior of physical systems is often studied by the use of electronic analog computers. If the system considered is characterized by a continuous distribution of properties, the describing system equations are of the partial differential class. An analogous set of equations, soluble on the analog computer, can often be formed by the application of finite difference approximations. For example, a physical system, originally space and time dependent, can sometimes be sectionalized into a number of space segments and then described by a set of ordinary differential and/or algebraic equations. Such sectionalization ordinarily results in a number of equation sets of similar form.

Most often the computational approach to the sectionalized problem is to associate with each section a block of analog equipment. Each section block is usually composed of identical analog computer components. The total system is then simulated by cascading the section blocks. For multisection systems, as are required for rapid transients, equipment requirements increase as n , the number of sections, increases. As a result, some problems cannot be accommodated by the analog computer facility. Further, the large number of potentiometers resulting from the sectionalized cascade solution increases problem setup time and the probability of operator error in the potentiometer-set-phase of setup.

A time-sharing analog solution, described in this paper, replaces the cascade of similar circuits by a single circuit whose components are time-shared. In conjunction with the actual computing elements of the time-shared circuit are circuits to provide time delay and timing functions. This sharing permits a reduction in equipment requirements permitting a smaller investment in computing equipment for a given problem size or increased problem capacity over that available without time-sharing.

II. DESCRIPTION AND METHOD OF OPERATION

A. A Typical Problem and Method of Solution

The description and method of operation of a time-sharing analog computer designed to solve a typical set of pressurized water, forced convection reactor core heat transfer equations follows. The machine to be described

illustrates the principal features of time-sharing and the solution of the sectionalized reactor core equations may be considered a typical application.

The set of differential-difference equations to be solved is given below and is shown, along with a sketch of the model, in Fig. 1. Coolant flow is assumed constant for the analysis:¹

$$\frac{dT_{m_k}(t)}{dt} = \alpha_1 \bar{q}_k(t) - \alpha_2' [T_{m_k}(t) - T_{w_k}(t)] \quad (1)$$

$$\frac{dT_{w_k}(t)}{dt} = \alpha_3' [T_{m_k}(t) - T_{w_k}(t)] - \alpha_4' [T_{o_k}(t) - T_{i_k}(t)] \quad (2)$$

$$T_{w_k}(t) = \frac{T_{o_k}(t) + T_{i_k}(t)}{2} \quad (3)$$

$$T_{o_k}(t) = T_{i_{k+1}}(t). \quad (4)$$

The forcing functions for the k th section are the inlet coolant temperature $T_{i_k}(t)$ and the heat flux $\bar{q}_k(t)$. The output coolant temperature is $T_{o_k}(t)$, and the average metal and coolant temperatures are $T_{m_k}(t)$ and $T_{w_k}(t)$, respectively.

The block diagram of Fig. 2 illustrates the conventional, cascaded method of solution. Each of the n sections is composed of the same computing elements. Fig. 3 indicates, in block diagram, the solution of the equation set by time-sharing. The substitution of a single time-shared circuit for the tandem string of circuits is evident by comparing these two diagrams.

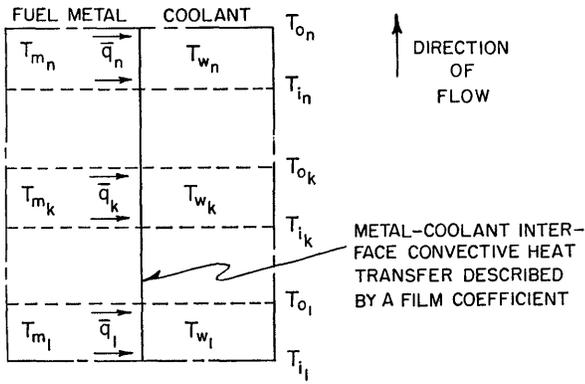
Fig. 4 shows a four-section analog circuit diagram for a time-sharing machine to solve the set under consideration. This analog circuit can be considered as a combination of two circuits, *i.e.*, an equation solving section and an auxiliary or service section. The equation solving section consists of integrators A and B , summer D , and the gain potentiometers with settings α_1 , α_2' , α_3' , α_4' . Integrator A solves (1) for $T_{m_k}(t)$ given the heat flux and the film drop. Eq. (2) is solved by integrator B for $T_{w_k}(t)$ by integrating the film drop and the coolant temperature rise across the k th section. Summer D produces the outlet coolant temperature $T_{o_k}(t)$ by solving (3). The auxiliary or service circuit, peculiar to this time-sharing machine, includes six special devices. The devices and their functions are as follows.

- 1) Delay circuits D_A , D_B , D_C , and D_D receive, store, and discharge voltages at times determined by control signals. The delay circuits can be classified into two types by considering the nature of the signals upon which they operate. First are those

* This paper is an abstract of a thesis presented at the University of Pittsburgh for the M.S. degree. The author is indebted to Drs. J. F. Calvert, T. W. Sze, and D. J. Ford, all of the University of Pittsburgh, for helpful criticism.

† Bettis Atomic Power Div., Westinghouse Electric Corp., Pittsburgh, Pa. Operated for the U. S. Atomic Energy Commission by the Westinghouse Electric Corp. under Contract AT-11-1-GEN-14.

¹ See Appendix for a list of symbols.



$$\frac{dT_{m_k}(t)}{dt} = \alpha_1 \bar{q}_k(t) - \alpha_2' [T_{m_k}(t) - T_{w_k}(t)]$$

$$\frac{dT_{w_k}(t)}{dt} = \alpha_3' [T_{m_k}(t) - T_{w_k}(t)] - \alpha_4' [T_{o_k}(t) - T_{i_k}(t)]$$

$$T_{w_k}(t) = \frac{T_{o_k}(t) + T_{i_k}(t)}{2}$$

$$T_{o_k}(t) = T_{i_{k+1}}(t)$$

Fig. 1—Reactor core heat transfer equations and model sketch.

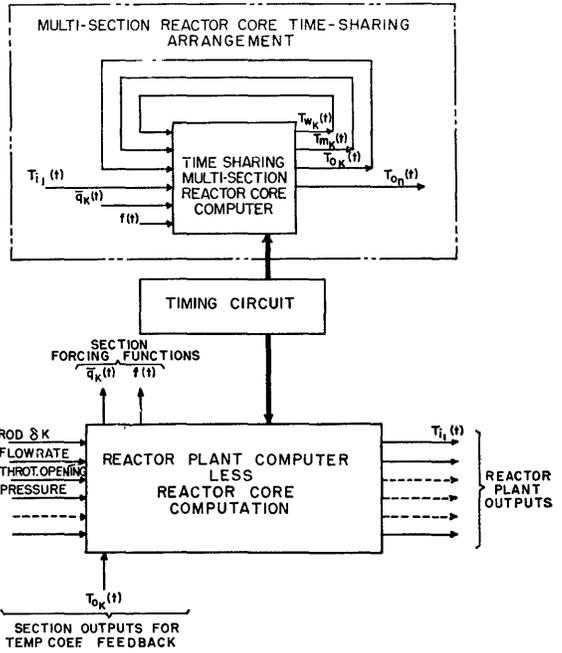


Fig. 3—Reactor plant simulator with multisection reactor core time-sharing computer.

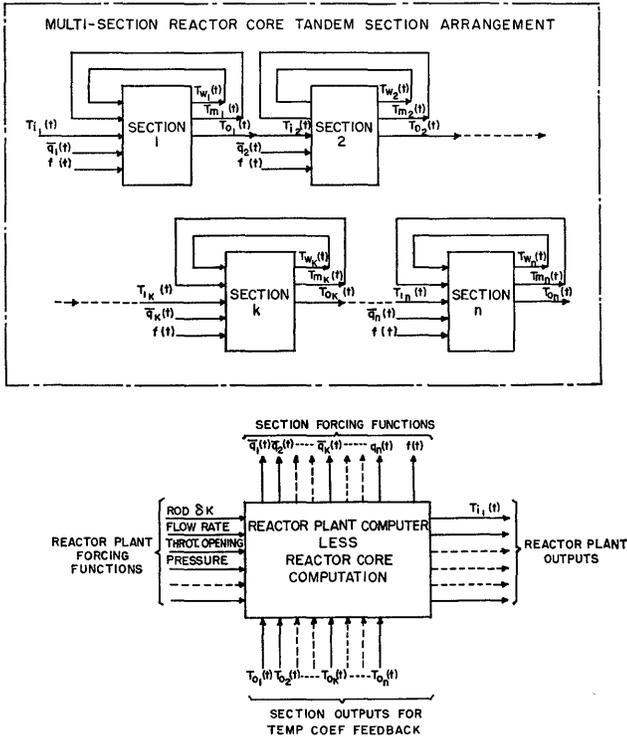


Fig. 2—Reactor plant simulator with multisection reactor core tandem section computer.

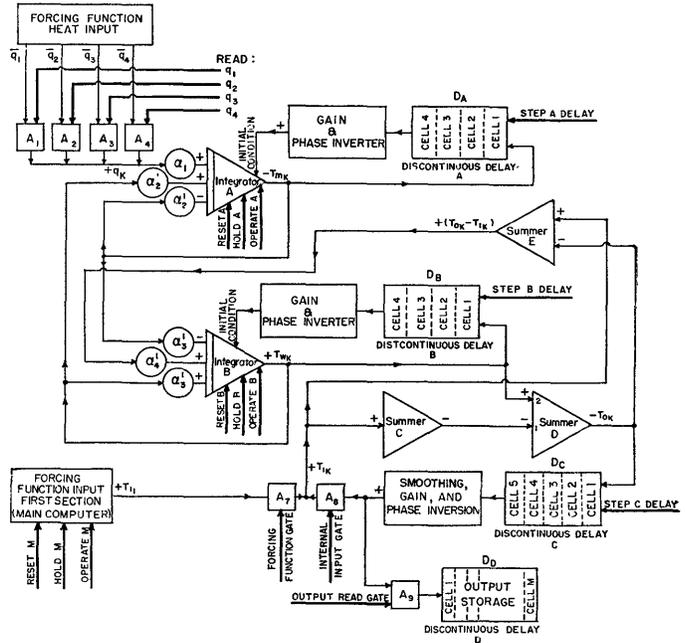


Fig. 4—Time-sharing computer for the solution of a sectionalized heat transfer problem-reactor core. Circuit model for: open loop, constant flow, non-uniform heat flux situation (four section model).

$$\frac{dT_{m_k}}{dt} = \alpha_1 \bar{q}_k - \alpha_2' (T_{m_k} - T_{w_k})$$

$$\frac{dT_{w_k}}{dt} = \alpha_3' (T_{m_k} - T_{w_k}) - \alpha_4' (T_{o_k} - T_{i_k})$$

$$T_{o_k} = 2T_{w_k} - T_{i_k}$$

$$T_{o_k} = T_{i_{k+1}}$$

which delay initial condition voltages, *e.g.*, D_A , D_B . These signals are discrete in nature and the necessary delay may be discontinuous, *i.e.*, discrete sampling, storage, and discharge. Second are those which delay continuous voltages, *e.g.*, D_C , D_D . This second type is extremely difficult to realize economically, particularly when the time delays are long. For this reason all of the delays designed for this prototype computer are of the first type. The continuous signal delays are approximated by smoothing operations performed on the discontinuous delays.

- 2) The integrator operation control circuit changes the operational state of integrators (Reset, Hold, Operate) in response to command signals from a timing circuit.
- 3) A heat input circuit provides the k th section heat forcing function when the k th section equations are being determined by the equation solving circuit.
- 4) Smoothing, gain, and phase inverting circuits perform the functions their names imply.
- 5) Gates A_1 , A_2 , A_3 , A_4 , A_7 , A_8 , and A_9 control the flow of signals.
- 6) A timing circuit controls the sequence of operations of the auxiliary devices and the main and time-sharing computer. Command signals from the timing circuit are shown as heavy lines in Fig. 4.

Prior to the initiation of operation of the time-sharing computer, steady-state calculations are performed to obtain the initial conditions for all n sections. These initial conditions are denoted by $T_{m_k}(o)$ and $T_{w_k}(o)$. The voltage analogs of these temperatures are stored in discontinuous delay circuits D_A and D_B , respectively. These circuits consist of four tandem cells denoted by C_1 through C_4 . The number of cells making up delays D_A and D_B does not necessarily equal the number of sections being simulated. The requirement is that the number of cells and the stepping rate yield a time delay of $n\tau_R + (n-1)\tau_s$ where n is the number of sections, τ_s the sampling time, and τ_R the reset time. For the subsequent discussion delay D_C is assumed to consist of five cells permitting five samples in each sampling interval τ_s , *i.e.*, $p=5$. The time delay employed is τ_s for the first sample and $\tau_R + (p-2)\tau_s/p$ for the four subsequent samples. If a continuous delay were to be used a delay of $\tau_s + \tau_R$ would be employed.

The heat input forcing function circuit (shown in the upper left of Fig. 4) computes \bar{q}_1 , \bar{q}_2 , \bar{q}_3 , and \bar{q}_4 , and these variables appear at the inputs of gates A_1 through A_4 , respectively. The outputs of these gates are multiplied to form the \bar{q}_k input line.

Two additional engineering considerations remain. A sampling period is chosen to establish the rate of com-

putation as controlled by the timing circuit. The choice of a sampling period is governed by the speed of the transient to be encountered, the degree of reactivity feedback via the temperature coefficient, the desired accuracy, the allowable machine running time, and other considerations. The sampling period is denoted by τ_s . The second consideration is the evaluation of the hot leg transport time τ_{dh} . This time fixes the number of tandem cells required in delay circuit D_D (output storage, D_D , could also be a continuous type delay, *e.g.*, tape, if economic considerations permit). With τ_{dh} established, the timing circuits are adjusted to provide such a delay.

The operation of the time-sharing analog computer proceeds as follows. The computer integrators are set to Reset, installing initial conditions $T_{m_1}(o)$ and $T_{w_1}(o)$ at the outputs of integrators A and B . With gates A_1 and A_7 open and all others closed, the main and time-sharing computers are set to Operate condition. The circuit remains in Operate for τ_s seconds during which time the forcing function $T_{i_1}(t)$ flows into the computer. Since gate A_1 is open, the heat flux presented to the circuit is \bar{q}_1 . The output of summer D is, consequently, the analog behavior of $T_{o_1}(t)$ for the period τ_s , *i.e.*, the output water temperature transient of the first section of the four-section model during the sampling period τ_s . This output temperature transient is to become the input forcing function for section two of the model during the subsequent operational period, and so provision is made to store discrete values of $T_{o_1}(t)$. Such storage is accomplished by stepping the discontinuous delay circuit D_C at intervals during the initial τ_s seconds. Such a stepping action is caused to take place every $\tau_s/4$ seconds by the timing circuit. The result of this action is the storage of five samples of $T_{o_1}(t)$ in delay circuit D_C at the end of τ_s seconds. These five voltage analogs denoted by $T_{o_1}(o)$, $T_{o_1}(1)$, $T_{o_1}(2)$, $T_{o_1}(3)$, and $T_{o_1}(4)$ appear in cells C_5 , C_4 , C_3 , C_2 , and C_1 of D_C , respectively. At the end of τ_s seconds the main computer (external to the time-sharing computer) and the time-sharing computer are set to the Hold condition. Shortly thereafter, the initial condition delay circuits D_A and D_B are stepped placing conditions $T_{m_2}(o)$ and $T_{w_2}(o)$ at the outputs of integrators A and B . Stepping delays D_A and D_B also causes the state of integrators A and B (at a time τ_s after the beginning of the transient) to be stored in C_1 of D_A and D_B . These analog voltages are the initial conditions required for the second complete cycle of computation. They displace, in the delay circuits, the "initial" initial conditions. The notation employed for these conditions is $T_{m_k}(4)$ and $T_{w_k}(4)$ where the parenthetical number denotes the state of the variable after a time $4\tau_s/4$ seconds. Then, with gates A_2 and A_8 open and all others closed, the time-sharing computer is placed in Reset and shortly thereafter in the Operate condition.

The input forcing function for the second section is

now the output of the first section as previously computed. This signal is introduced by stepping the D_C delay. Such stepping causes the discrete analog voltages to pass out of the delay, through the smoothing, gain, and phase inverting circuit, and into the computational circuit via gate A_8 . This same stepping of D_C causes the output transient of the second section to pass into the delay D_C for storage and future use in the next cycle of computation. Again, the output transient is sampled at five points $\tau_s/4$ seconds apart in time. The output transient from the third section is obtained during the third τ_s interval of time by the same sequential process employed in solving the section two response, and similarly for section four.

At the end of the fourth τ_s second interval the contents of delay circuit D_C are five voltages representing samples of the outlet water temperature transient during the initial sampling period of the input water temperature forcing function. At this time, with the time-sharing and main computer in the Hold condition, the initial condition delays D_A and D_B are stepped placing $T_{m_1}(4)$ and $T_{w_1}(4)$ upon integrators A and B . Further, gate A_8 is closed and A_7 opened permitting the second τ_s interval of the inlet coolant temperature forcing function to drive the first section computation when the computers are set to Operate. Gate A_1 is opened and the time-sharing computer is set to Reset. Both computers are now placed in Operate and the second cycle of computation begins. The four section computations proceed as previously described. During the first τ_s seconds of the second complete cycle gate A_9 is open. This open gate permits the output transient from the first cycle to pass into the output storage delay circuit D_D as the first section response to the second sampling interval displaces this information in storage device D_C .

As the computation proceeds in a cyclic fashion the output storage D_D becomes filled with samples of the desired output coolant temperature transient. Each analog sample, spaced $\tau_s/4$ seconds apart in time, is stored in sequential order in the D_D device. The earliest (time-wise) voltage appears in the highest order cell and the latest voltage sample in the lowest order cell, *i.e.*, the input cell number (C_1). As soon as sufficient τ_s second intervals have elapsed so that the sum of the intervals totals the hot leg transport delay τ_{dh} , the output storage delay begins to discharge the sampled output transient. This output information is sent out in spurts, four section computation times apart. Each data spurt consists of five voltage samples spaced $\tau_s/4$ apart in time. Such discrete data may be smoothed to convert to a continuous analog form.

B. Feedback Considerations

Time-sharing techniques must include provision for feedback signals such as the temperature feedback loop signal in the simulation of a reactor plant with a nonzero temperature coefficient of reactivity. The effect to be simulated can be described as

$$\partial[\delta k_{TC}(t)] = f \left\{ \sum_{k=1}^n K_{TCk} [\partial T_{w_k}(t)] \right\}. \quad (5)$$

Or if the temperature coefficient, K_{TCk} , is assumed spatially constant

$$\partial[\delta k_{TC}(t)] = f[K_T \partial T_{ave}(t)] \quad (6)$$

where

$$T_{ave}(t) = \frac{1}{n} \sum_{k=1}^n T_{w_k}(t). \quad (7)$$

An exact summation process, as required by (5) or (7), is not possible with time-sharing techniques. This inherent limitation is so because the instantaneous behavior of the average water temperature in all n sections is known only during the computation of the final or n th section, and then only if all previous $T_{w_k}(t)$ transients are stored.

The circuit next described approximates $T_{ave}(t)$ as given by (7). The method proposed is the repeated correction of the average existing at the start of any one complete cycle by the use of the section data as it becomes available. Listed below are equations which describe such a method.

T_{ave} at the start of a four-section cycle is

$$T_{ave}(0) = \frac{1}{4} \sum_{k=1}^4 T_{w_k}(0)$$

during the first section computation

$$T_{ave}(t) = \frac{1}{4} \sum_1^4 T_{w_k}(0) + \frac{1}{4} [T_{w_1}(t) - T_{w_1}(0)];$$

during the second section

$$T_{ave}(t) = \frac{1}{4} \sum_1^4 T_{w_k}(0) + \frac{1}{4} [T_{w_1}(4) - T_{w_1}(0)] \\ + \frac{1}{4} [T_{w_2}(t) - T_{w_2}(0)];$$

and the third section

$$T_{ave} = \frac{1}{4} \sum_1^4 T_{w_k}(0) + \frac{1}{4} [T_{w_1}(4) - T_{w_1}(0)] \\ + \frac{1}{4} [T_{w_2}(4) - T_{w_2}(0)] + \frac{1}{4} [T_{w_3}(t) - T_{w_3}(0)]$$

and finally, during the fourth section computation the average becomes

$$T_{ave}(t) = \frac{1}{4} \sum_1^4 T_{w_k}(0) + \frac{1}{4} [T_{w_1}(4) - T_{w_1}(0)] \\ + \frac{1}{4} [T_{w_2}(4) - T_{w_2}(0)] + \frac{1}{4} [T_{w_3}(4) - T_{w_3}(0)] \\ + \frac{1}{4} [T_{w_4}(t) - T_{w_4}(0)].$$

At the end of the first complete cycle of computation the T_{ave} signal is

$$T_{\text{ave}}(4) = \frac{1}{4} \sum_{k=1}^4 T_{w_k}(4)$$

and so, during the next cycle, the identical process can be repeated.

A circuit to accomplish this task is shown in Fig. 5. The expressions, indicated in terms of temperature, are the analogs of the voltages which, of course, actually occur. The state of the circuit is that which would exist at the start of the first section computation of the first cycle.

The operation of the circuit proceeds as follows. Prior to time zero,

$$\frac{1}{4} \sum_{k=1}^4 T_{w_k}(0)$$

is stored on capacitor C_1 . Relay T_{ave} is de-energized and $T_{w_1}(t) = T_{w_1}(0)$ so that the output of summer P is also

$$\frac{1}{4} \sum_{k=1}^4 T_{w_k}(0).$$

The analog voltage of this term is applied to capacitor C_2 through the NC_x contact on relay T_{ave} . The inputs of summer Q are $+T_{w_1}(t)$ and $-T_{w_1}(0)$ which are obtained from the output of integrator B and the gain and phase inverter following delay D_B , respectively. Integrator B and delay D_B are shown in Fig. 4. During the first section computation (computers set to Operate) $T_{w_1}(t)$ begins to differ from $T_{w_1}(0)$. This difference is computed by summer Q and added to the original summation stored on C_1 by summer P after being attenuated by $1/n$ by the input potentiometer shown. This new voltage is applied to capacitor C_2 .

At the end of the first section computation, relay T_{ave} is energized and maintained up during the second section period. Now capacitor C_2 "remembers" the initial voltage and the new sum consisting of

$$\frac{1}{n} \sum_{k=1}^4 T_{w_k}(0) + \frac{1}{4} [T_{w_1}(4) - T_{w_1}(0)] + \frac{1}{4} [T_{w_2}(t) - T_{w_2}(0)]$$

is applied to capacitor C_2 through the NO_x contact. During this period, the inputs to summer Q are $T_{w_2}(t)$ and $T_{w_2}(0)$. Relay T_{ave} is thus alternately de-energized and energized until all four sections have been computed. At the end of four periods capacitor C_2 has the analog of

$$\frac{1}{4} \sum_{k=1}^4 T_{w_k} \quad (4)$$

stored upon it. Summer R corrects the stored signals for attenuation and dc shift suffered in passing through the cathode-follower read-out circuit. The succeeding cycles proceed as the first.

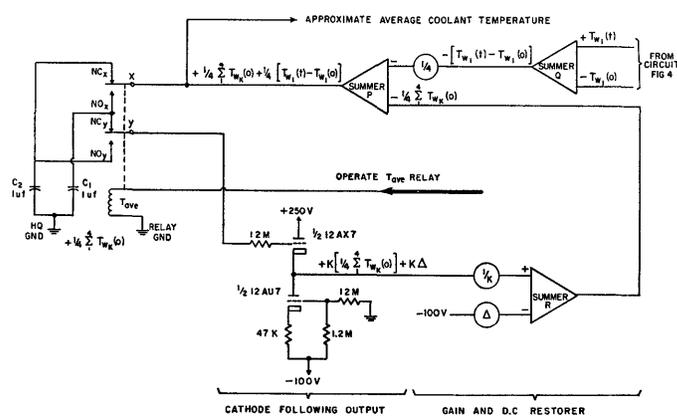


Fig. 5—Summation circuit to approximate the average coolant temperature.

C. Time and Space Dependent Forcing Functions

Provisions for forcing functions which are both time and space dependent require another novel time-sharing circuit. The heat flux input to the reactor core is a typical example.

By finite differencing techniques the k th section heat flux input can be approximated by

$$\bar{q}_k(t) = m_k \bar{q}_T(t),$$

where m_k is constant. Several system variables cause time variations in the reactor core—heat flux, $[\bar{q}_T(t)]$, *e.g.*, changes in rod position, coolant temperature, and pressure. If the sampling period of the time-sharing computer is chosen so that the core heat flux, $\bar{q}_T(t)$, changes appreciably during the period, provision must be made to include such variations in the computation.

Fig. 6 is a circuit diagram of a heat flux circuit which provides both a space and time variant forcing function. During the first sampling period relay Q is inoperative permitting the $\bar{q}_T(t)$ signal to flow to the input bus of the m_k scaling potentiometers and to the input of discontinuous delay D_Q (this delay could also be of the continuous type, *e.g.*, magnetic tape). While the \bar{q}_T signal flows, the D_Q delay is stepped causing five (arbitrary number) samples to be stored within the delay. Potentiometer m_1 scales $\bar{q}_T(t)$ yielding $\bar{q}_1(t)$. During this time, rotary switch R_Q is in position 1. Thus, $\bar{q}_1(t)$ appears at the output of the heat flux circuit. At the start of the second section computation relay Q is operated and switch R_Q is stepped to position 2 by pulsing the step R_Q lead. During the second interval delay D_Q is stepped periodically causing the initial $\bar{q}_Q(t)$ signal to reappear on the potentiometer input bus as well as at the input of the delay. Smoothing and gain are applied to the discontinuous signal as indicated. During this period $\bar{q}_2(t)$ appears as the output of the flux circuit. This sequence is continued until all n sections have been computed. At the completion of the computational cycle rotary switch R_Q is set to position 1 by pulsing the home R_Q lead which actuates the release magnet. Relay Q is released and the circuit is ready for the second sampling period. This cir-

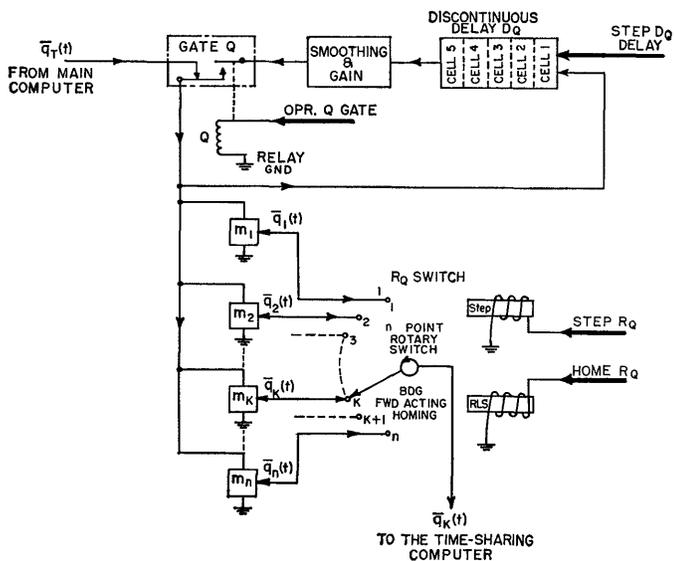


Fig. 6—Heat flux forcing function circuit for a time-sharing computer solving the reactor core equations.

circuit performs the functions of gates A_1 through A_4 of Fig. 4.

D. Time Dependent Forcing Functions

Another class of forcing functions which must be handled by time-sharing are those which are time dependent only. An example of this class is the coolant flow rate through the reactor core. For nonconstant flow (1) and (2) are amended to read:

$$\frac{dT_{m_k}(t)}{dt} = \alpha_1 \bar{q}_k(t) - \alpha_2 f(t)^{0.8} [T_{m_k}(t) - T_{w_k}(t)], \quad (8)$$

$$\frac{dT_{w_k}(t)}{dt} = \alpha_3 f(t)^{0.8} [T_{m_k}(t) - T_{w_k}(t)] - \alpha_4 f(t) [T_{o_k}(t) - T_{i_k}(t)]. \quad (9)$$

Clearly those terms in (8) and (9) with coefficients α_2 , α_3 , and α_4 are dependent upon flow rate. Fig. 7 is a circuit designed to include the effects of variable flow upon the reactor core analog simulation. The circuit is shown as it would appear when augmenting the time-sharing simulator illustrated in Fig. 4. Only integrators A and B of Fig. 4 are indicated and all other components are omitted for simplicity. The variable flow portion is set off by the heavy broken line in Fig. 7. At the start of the transient run relay F is de-energized allowing the flow signal $f(t)$ to pass into the computer as a continuous function. During the initial sampling period, delay D_F is stepped causing discrete samples of $f(t)$ to be stored in the delay (D_F could be a continuous delay, e.g., tape). The sample $f(t)$ also passes to multiplier M_B , and the function generator FG1 and hence to multiplier M_A . The outputs of these multipliers, M_A and M_B , are the desired functions

$$-\alpha_2 f(t)^{0.8} [T_{m_k}(t) - T_{w_k}(t)]$$

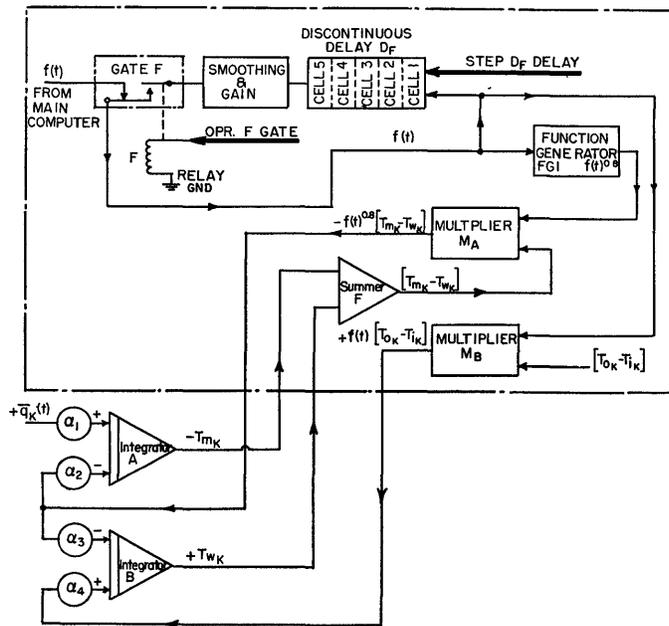


Fig. 7—Time variant flow rate forcing function circuit for a time-sharing computer solving the reactor core equations.

and

$$f(t) [T_{o_k}(t) - T_{i_k}(t)],$$

respectively. At the end of the section 1 computational period, relay F is energized by applying voltage to the Operate F gate lead. During the section 2 and succeeding section computation periods the discontinuous delay D_F is stepped periodically causing the initial $f(t)$ sample to pass out of the delay, through the smoothing and gain circuit, and into the computing circuit. Thus, the sample is reused in each section period. At the conclusion of the complete n -section computing cycle, relay F is de-energized and the circuit is prepared to receive the second $f(t)$ sample from the main computer.

III. PILOT MODEL

In order to determine the workability and accuracy of the time-sharing computing method a pilot model was designed and constructed with sufficient capacity to solve a four-section reactor core heat transfer problem with constant coolant flow and uniform axial heat flux.

The necessary delay circuits for the pilot model were designed by an extension of an invention attributed to Janssen² and later demonstrated by Philbrick.³ A block diagram of a delay circuit is shown in Fig. 8. Buffer amplifiers B_1 , B_2 , etc., have the following properties:

- 1) very high input impedance,
- 2) very low output impedance,
- 3) amplification close to unity, and
- 4) high available output power.

² J. M. L. Janssen, "Discontinuous low-frequency delay line with continuously variable delay," *Nature*, vol. 169, p. 148; January, 1952.

³ "A Palimpsest on the Electronic Analog Art," ed. by H. M. Paynter, G. A. Philbrick Researches, Inc., Boston, Mass., p. 163; 1955.

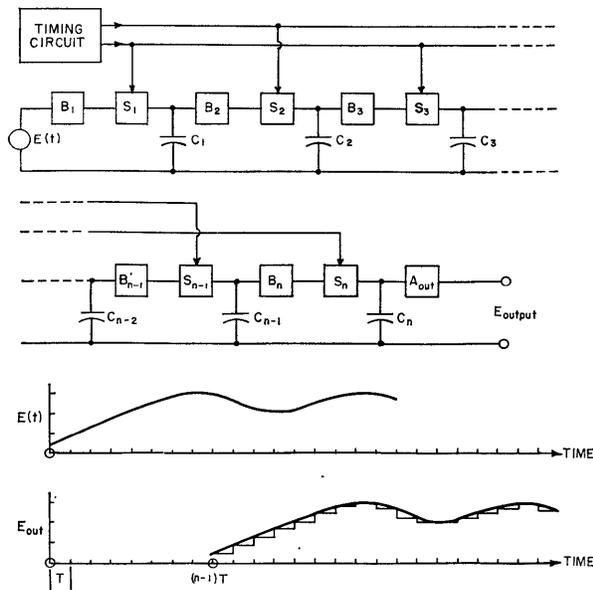


Fig. 8—Block diagram of a discontinuous delay line with continuous variable delay (after J. M. L. Janssen, Royal Dutch/Shell Laboratory, Delft, Netherlands, October 25, 1951).

These devices were obtained by the design of an extra-linear cathode follower. Switches S_1 , S_2 , etc. have characteristics as follows:

- 1) very low forward impedance,
- 2) very high reverse impedance, and
- 3) controllable by external command signals.

The switches for the delay circuits of the pilot model were designed for two different applications of the delays:

- 1) The short time delay circuits, *e.g.*, the recycled forcing function delays (delay D_C of Fig. 4), required bilateral electronic switches patterned after the work of Philbrick.³
- 2) The long time delay circuits, *e.g.*, the initial condition delays (D_A and D_B of Fig. 4), were designed with fast-acting relay contact switches.

Capacitors C_1 , C_2 , etc., are extremely low-leakage components. Output amplifier, A_{out} , has:

- 1) adjustable gain,
- 2) very high input impedance, and
- 3) low output impedance.

This component was obtained by cascading a buffer amplifier and a conventional analog computer dc amplifier. This arrangement permitted the required smoothing operation and the gain adjustment to be performed within the output device.

To begin the explanation of the delay circuit it is assumed that all switches are open and all capacitors initially uncharged. At time zero all odd number switches are closed for a sufficient time to cause capacitor C_1 to charge to $E(0)$. At time T all even number switches operate causing C_2 to assume voltage $E(0)$. The odd switches are then closed at time $2T$ allowing $E(2T)$ to

charge C_1 and $E(0)$ to pass to C_3 . At time $3T$ even switches are closed moving $E(2T)$ to C_2 and $E(0)$ to C_4 . The process of alternately closing the odd and even number switches is continued with closures every T seconds. Eventually, after $(n-1)T$ seconds, voltage $E(0)$ appears on C_n and hence becomes the first output sample. Following this voltage, every $2T$ seconds, are $E(2T)$, $E(4T)$, $E(6T)$, etc. Thus a delay of $(n-1)T$ is achieved. Since T , the switching period, can be controlled, the objective is achieved.

The timing circuit of the pilot model was synchronized with a master clock. Clock pulses were used to drive bistable multivibrators which performed desired frequency divisions. The resulting subharmonics of the clock pulse train were directed to a logic circuit which generated the necessary control pulses to execute the desired sequential switching plan. The control pulses, after receiving power amplification, actuated relays whose contacts formed a switching network. The signals from the network controlled the operation of the component devices which made up the time-sharing computer.

IV. TEST RESULTS AND CONCLUSIONS

In order to evaluate the performance and accuracy of the pilot model a series of tests were run in which the reactor core heat transfer equations were simulated with zero heat input and constant flow, *i.e.*, a simple transport delay problem described by:

$$\frac{dT_{wk}}{dt} = \frac{2n}{\tau_{do}} T_{ik} - \frac{2n}{\tau_{do}} T_{wk}, \quad (10)$$

$$T_{ok} = 2T_{wk} - T_{ik}, \quad (11)$$

$$T_{ik} = T_{ok-1}. \quad (12)$$

The forcing function was a cosine shaped increase in the inlet coolant temperature. The results of these experiments indicated:

- 1) Conventional and time-sharing circuits are compatible and reproducible results are obtainable. Switching transients, relay contact "races," and switching synchronism problems are evident but they can be overcome by proper circuit engineering.
- 2) Simulation accuracy is a function of the sampling interval employed in the delay circuits and the method of signal smoothing employed. Fig. 9 shows a typical input-output trace. The circuit was forced by a 0.785 rad/second cosine rise in inlet coolant temperature. Delay D_C of Fig. 4 was smoothed by a $1/(\tau_e s + 1)$ filter in which the optimum τ_e was found to be 0.03 second. The maximum per cent departure from the ideal delayed transient (also shown in Fig. 9) is 3.1 per cent occurring 3.6 seconds from the start of the transient.
- 3) The accuracy of the simulation of systems in which feedback signals dependent upon instan-

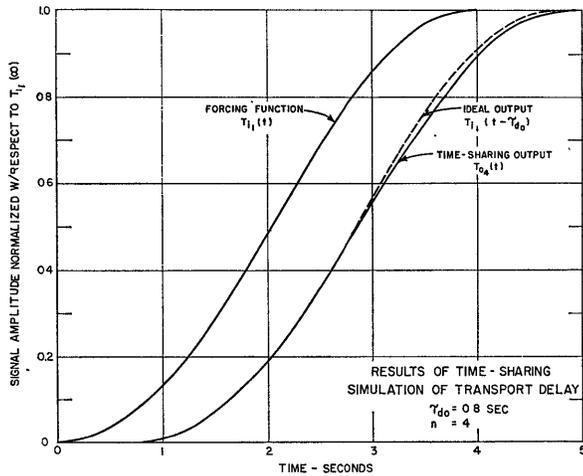


Fig. 9—Results of a time-sharing simulation of transport delay.

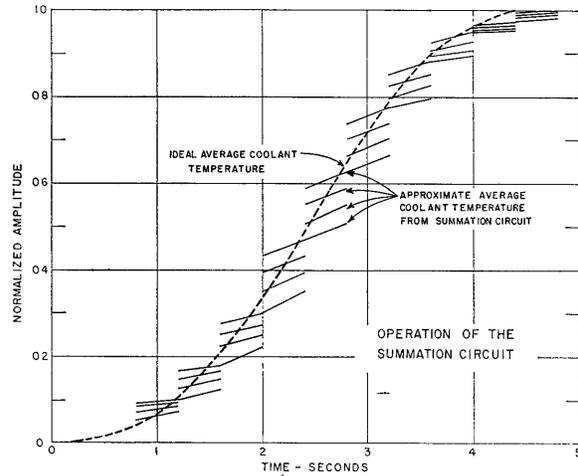


Fig. 10—Operation of the summation circuit.

taneous spatial averages, is limited by the inherent inability of the time-sharing computer to correctly obtain such averages. The operation of the circuits designed to obtain the approximate average coolant temperature was, however, successful. Fig. 10 illustrates the results of approximating the spatial average by employing the circuitry previously described. In this figure each τ_s segment of the input forcing function produces four output traces since the average water temperature is continuously corrected as new data become available from the four sections in turn. The nature of the approximation is seen by comparing the summation circuit output traces, shown as solid lines in Fig. 10, with the ideal average temperature shown as a dashed line.

V. COST AND EQUIPMENT REQUIREMENTS

In order to compare the approximate cost and equipment requirements of time-sharing computation with conventional analog computation a calculation of these requirements for a large but typical problem was performed. The problem considered was the simulation of the heat transfer phenomena of a reactor core coupled to a heat exchanger within a pressurized, forced convection system. The simulator was designed with sufficient capacity to provide the following:

- 1) variable coolant flow,
- 2) nonuniform axial heat flux,
- 3) temperature coefficient of reactivity feedback,
- 4) variable steam throttle opening, and
- 5) steam temperature feedback in the heat exchanger.

The most important results of the calculation are shown in the three graphs of Fig. 11. Fig. 11 (a) compares the cost of time-sharing solutions to conventional cascade solutions as a function of the number of sections.

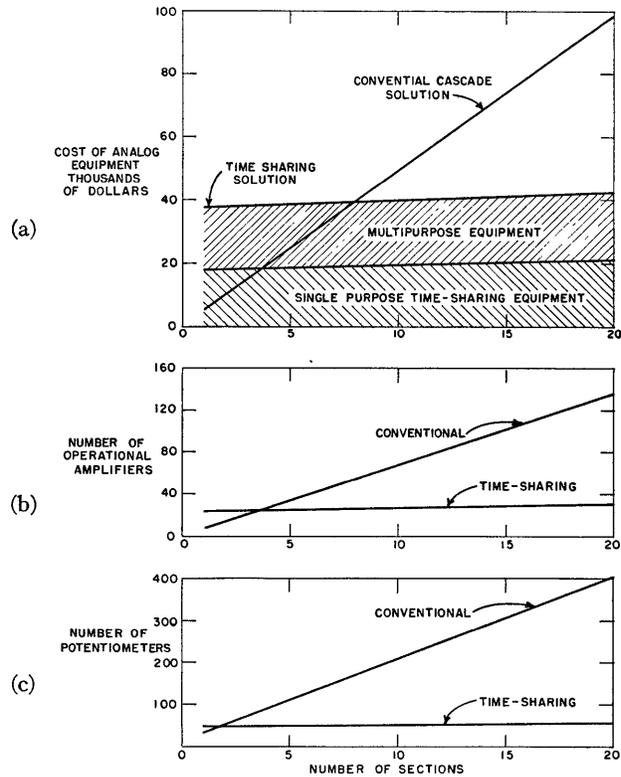


Fig. 11—Cost and equipment comparisons.

Time-sharing costs are divided into two parts. The first, shown cross-hatched in Fig. 11(a), represents the cost of that equipment which is peculiar to time-sharing and which could not readily be used in non-time-shared applications. The second component cost is that of multipurpose equipment, e.g., amplifiers, potentiometers, multipliers, etc., which, of course, could be used for non-time-shared problems. Fig. 11(b) and 11(c) shows the amplifier and potentiometer requirements as a function of the number of sections simulated.

The ratio of the machine running time required to

solve a sectionalized problem by time-sharing to the running time by the conventional cascade method is given by:

$$\frac{\text{time-sharing running time}}{\text{conventional running time}} = n \frac{(\tau_s + \tau_R)}{\tau_s}.$$

VI. CONCLUSIONS

The results of this study indicate that a time-sharing analog computer for the solution of differential-difference equations is realizable and is economically attractive in certain circumstances. If the number and size of multisection problems to be computed are large, the equipment conservation possible with time-sharing can conceivably outweigh the accuracy and running time penalties.

The three phases of this study, *viz.*, design, development, and optimization, yielded more specific conclusions. The design of a system to solve the reactor core and boiler heat transfer multisection equations revealed that considerable savings in operational amplifiers and potentiometers would be enjoyed when the number of sections simulated exceeded three. First cost of the auxiliary equipment required for the time-sharing solution of these equations is estimated to be equivalent to the first cost of a sufficient amount of conventional analog equipment to solve a four-section problem. Programming and set-up time would be markedly reduced by time-sharing particularly if a large number of physical parameters and analog scaling factors were common to many sections. Such a savings would tend to offset the increase in running time required by time-sharing. Design considerations further revealed the upper bound on the accuracy obtainable under time-sharing to be the accuracy achieved by the conventional cascade analog solution. This accuracy can be approached as

- 1) the sampling interval, τ_s , approaches zero,
- 2) the number of samples, p , obtained for the forcing function delays approaches infinity, and
- 3) the highest frequency components of the input forcing functions approach zero.

The development phase of this study produced working models of all of the circuits essential to the time-

sharing system. These units were successfully integrated with conventional analog computing equipment and test problems were run. The electronic and relay circuits proved reliable and the total system yielded reproducible results. The delay and timing circuit designs, as conceived for the four-section pilot model, can be readily extended to handle additional sections.

Much additional attention can be given to the optimization of the time-sharing system. The effect of the variation in computing parameters, *e.g.*, τ_s , upon accuracy might profitably be studied. Newly available analog devices, such as magnetic tape loop transport delays, could provide improvements in computational accuracy and thus merit investigation.

APPENDIX

LIST OF SYMBOLS

Symbol	Description	Units
α_1	Constant in heat transfer equations	ft ² °F/Btu
$\alpha_2', \alpha_3', \alpha_4'$	Constant in heat transfer equations	1/sec
σk_{TC}	Reactivity due to temperature effects	reactivity
K_T	Over-all temperature coefficient of reactivity	reactivity/°F
K_{TC_k}	Temperature coefficient of reactivity for the k th section	reactivity/°F
$f(t)$	Ratio of instantaneous coolant flow rate to the time-zero flow rate	dimensionless
m_k	Ratio of k th section average heat flux to the total average heat flux	dimensionless
n	Number of axial sections	dimensionless
p	Number of samples obtained for delays during τ_s interval	dimensionless
\bar{q}_k	Spatially averaged heat flux in the k th section	Btu/second ft ²
\bar{q}_T	Spatially averaged total heat flux	Btu/second ft ²
t	Time	second
T	Switching period of discontinuous delay	second
τ_c	Smoothing circuit time constant	second
τ_{do}	Coolant transport time through reactor core coolant channel	second
τ_{dh}	Hot leg transport time, core to heat exchanger	second
τ_R	Time-sharing reset time	second
τ_s	Time-sharing sampling time	second
T_{ave}	Spatially averaged mean coolant temperature	°F
T_{i_k}	Mean coolant temperature at entrance to the k th section	°F
T_{m_k}	Spatially averaged metal temperature in k th section	°F
T_{o_k}	Mean coolant temperature at exit of the k th section	°F
T_{w_k}	Spatially averaged mean coolant temperature in k th section	°F

Computers—The Answer to Real-Time Flight Analysis

GUENTHER HINTZE†

INTRODUCTION

THE development of real-time data processing facilities has become an important task for the missile test ranges. The requirement for efficient test programs and the desire of the missile developers to have the results of missile firings "as soon as possible" after the flight, are pressing and many studies and developments for improved test facilities with faster data processing are being undertaken. These developments can be broken down into two major categories: instrumentation and computers for the reduction and analysis of the measured data. As long as the data are only required "very fast," say one or a few hours after the flight, the main problem is the data collection, transmission, and conversion to make them in the proper forms available to high-speed computers for which the further reduction and print-out presents no particular difficulty if enough computer capacity and peripheral equipment is available.

However, the computational problem is of a different order for real-time data analysis whereby, based on this analysis, actions might be taken while the flight is still in progress in order to obtain conclusive information or to influence the flight experiment in such a way that the accomplishment of desired objectives might be secured. Here is a new, challenging task for electronic computers leading to the development of new concepts and the improvement of existing computers. Ground guidance computers which have been developed for several guided missile systems perform a similar job, directing a missile according to real-time data which are being continuously observed during the flight. There is, however, a main difference: while these computers have to direct a well or nearly well behaving missile, a fully developed real-time flight analysis and control facility is mainly called into action when things are not going well and something has to be done to save a costly flight test. It is evident that this is a formidable task and in this paper some ideas will be presented concerning the order of magnitude of the required computing equipment.

THE BASIC CONCEPT OF REAL-TIME FLIGHT ANALYSIS

Fig. 1 shows a simplified diagram of the real-time flight analysis and control concept. Simultaneously with the actual flight of the missile, the flight is also simulated on a missile simulator where initial conditions and parameters are set exactly in accordance with the real missile. Through real-time range instrumentation,

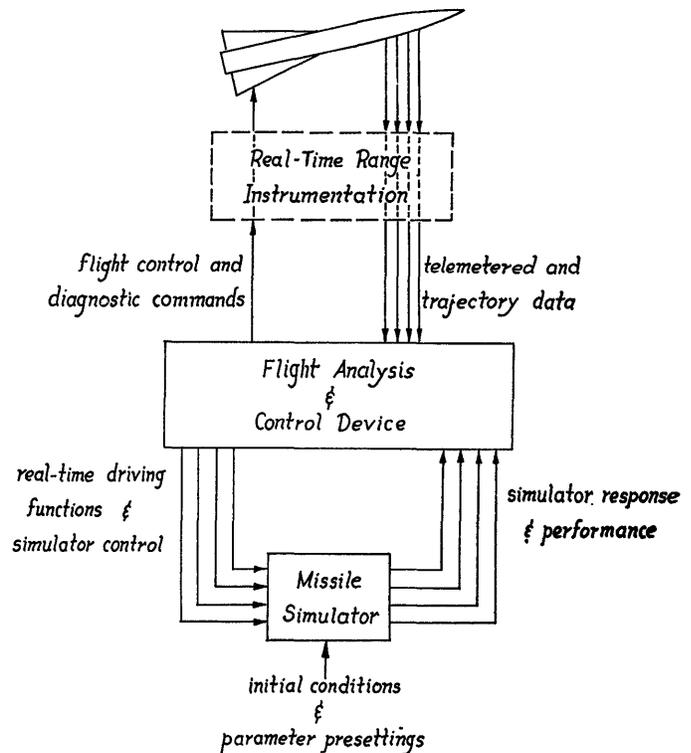


Fig. 1—Simplified diagram for real-time flight analysis.

including real-time data collection, transmission, and reduction, a ground computer receives continuously or in sampled form inputs, concerning the motion of and the events within the missile. In the diagram this computer is called flight analysis and control device. In order to make the simulated-flight compatible, the actual real-time driving and control functions are sent to the missile simulator according to the measurements which are obtained from the flight.

The data describing the response and performance of the simulated missile and its components are compared in the flight analysis computer with the actual flight data. As long as these values compare within certain predetermined and specified margins, no actions may be taken. If deviations develop which are still not critical but which warrant some investigation about the reasons for the deviation, diagnostic commands may be relayed to the missile switching over from some standard measurements in noncritical areas to additional measuring pickups which had been previously installed in suspected trouble areas. Sound engineering judgment is required to place such additional information sources. Case histories of previous flights and test experience will help in this regard. For example, a common trouble source are accelerometers which are used ac-

† Flight Simulation Lab., White Sands Missile Range, Las Cruces, N. Mex.

tively in guidance and control loops, and it is indicated to mount close to these accelerometers vibration pickups which can be switched in to determine quantitatively and timely the rise and propagation of vibrations which might cause erratic accelerometer outputs.

When failures occur which disable the missile to continue its flight properly and which will lead to aborted rounds, remedial commands to the missile shall be instituted. The analysis scheme must be detailed enough to decide if these commands should merely isolate a failing component, or send substitute command signals to the missile. For the future, it is even visualized to have some reserve jatos built in test missiles which could be used to cause a change in velocity from the ground.

It will require many diversified developments to advance real-time flight analysis and control to the point where it will be a workable and reliable facility which does not add to the difficulties of a timely execution of flight test programs, but which will be a valuable help for better understanding and utilization of missile flight experiments. It is felt that the part which computers will play in the accomplishment of such facilities is decisive, particularly if the test engineers and systems analysts succeed in formulating precisely the mathematical relations on which the analysis and the making of decisions is based.

MATHEMATICAL FORMULATION OF THE PROBLEM

The description of the tested guided missile system in form of a mathematical model is the necessary prerequisite for a further analysis of this system and the measured flight data on computers. The mathematical models which are presently used for flight simulation studies are deterministic or rigid analytical models where the input-output relations for missile motion and individual components are expressed in mechanistic equations. Fig. 2 shows the rigid analytical representation of the following portions of a missile system: the missile dynamics which determine the three vectors of lateral and angular accelerations as functions of force and moment inputs; the direction cosine matrix for transformation from missile body to radar axes using the angular rates of the missile body axes as input; the ground guidance computer which receives the rectangular missile coordinates in the radar frame; and finally, the control subsystem of the missile causing fin-deflections according to the command input from the ground and the missile motion.

This is an extremely simplified representation of a missile system which shall serve only the purpose to explain that most of the inputs are not fixed values, but they are characterized by some type of distribution. Even if the simulation state of the art is such that presently the number of stochastic inputs in simulation studies are quite limited, it must be borne in mind that a realistic simulation of a missile system which can be used as a reference in evaluating flight data must take into account these random variations of inputs.

Some of the areas where consideration should be given to stochastic inputs are the airframe characteristic anomalies such as airfoil malalignment and wind tunnel variation. In case of a spinning missile, fin malalignment may be a contributor to miss distance as much as some electronic failures. Burning anomalies affect the missile mass and inertia. Airframe flexure failure modes limit the required missile dynamics. Radar noise measured under one set of atmospheric conditions may be quite different from that measured under a changed condition. Component production variation effects, the malalignment of sensing elements, and the equipment response variations due to temperature or vibration are also stochastic variables.

It would be unjustified to draw conclusions from the comparison of flight data with some fixed value where in reality there is not such a thing as fixed values. The analysis must consider the distribution of these values and ultimately a probabilistic simulation model should be used as reference for real-time analysis. However, the development of such missile models is not yet completed. Great progress has been made in the realistic simulation of the random noise of track radars, but for many other stochastic missile variables, the amplitude distribution and frequency characteristics have not been determined. Therefore, in the following discussion, reference is made to existing, rigid analytical models, though the ultimate requirement for probabilistic analytical models is being recognized.

COMPUTER REQUIREMENTS

The computer requirements for conducting the real-time flight analysis are dictated principally by the degree of realism in the guided missile system simulation, and the complexity of the analysis schemes to be used.

Computer Requirements as a Function of Realistic Simulation

The simulation according to Fig. 3 when carried out in full without simplification will represent a guided missile system in 3 dimensions and 6° of freedom as it operates in rotating, oblate spheroidal earth, with external meteorological wind, gravitational attraction which varies according to latitude and missile altitude, and electromagnetic noise caused by track radars.

A great amount of computing equipment is required for the various coordinate transformations. If the aerodynamic forces and moments are not given in missile body axes, they have to be converted into these directions by use of a direction cosine matrix which is a function of the components of the missile velocity with respect to the air.

Since missile velocity is with reference to the body frame, the external wind velocity, expressed in the local ground frame, needs to be first transferred to a form referred to the body frame. Then this wind velocity is expressed in body frame components by use of a direction cosines matrix. From the missile rotational velocities,

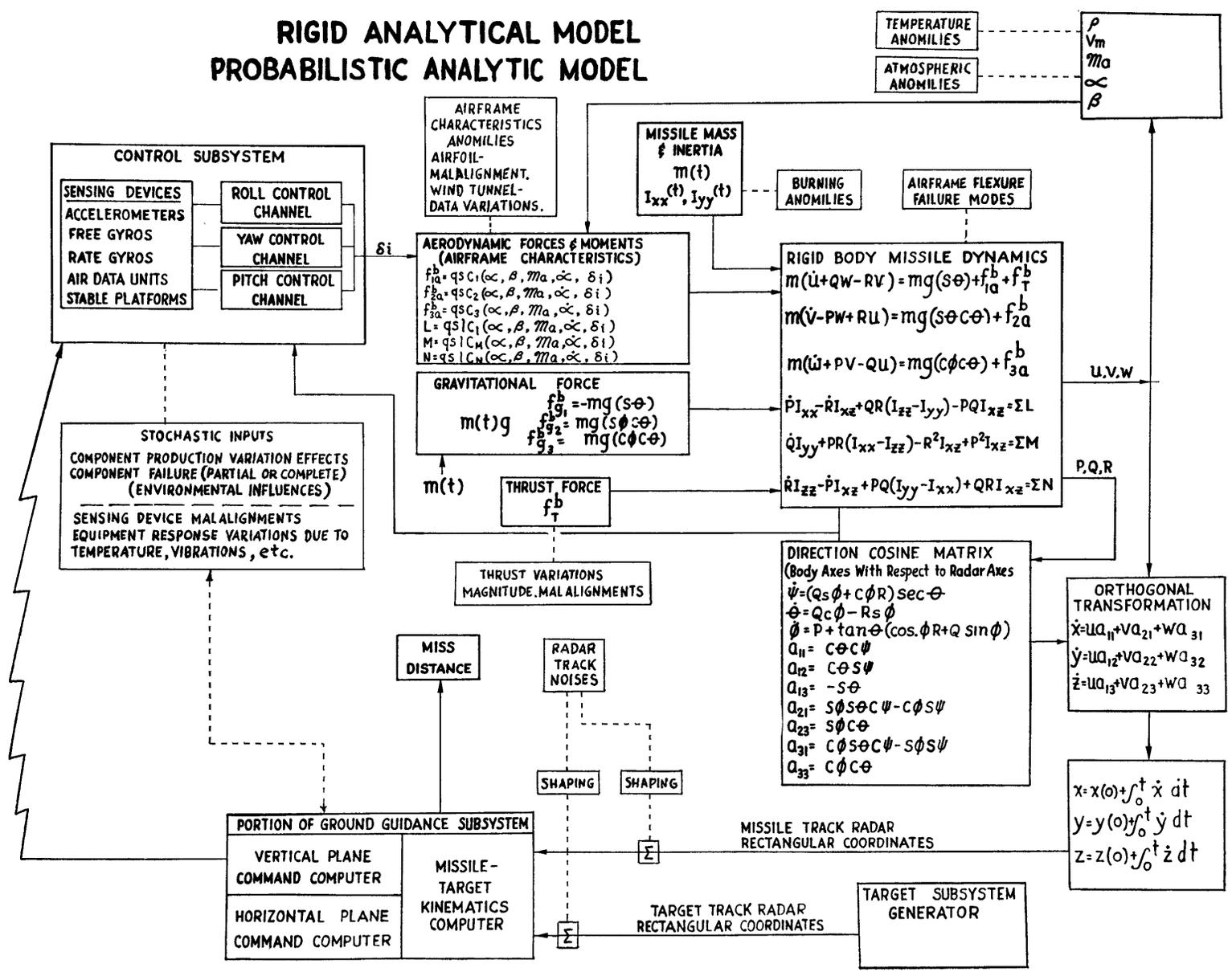


Fig. 2.

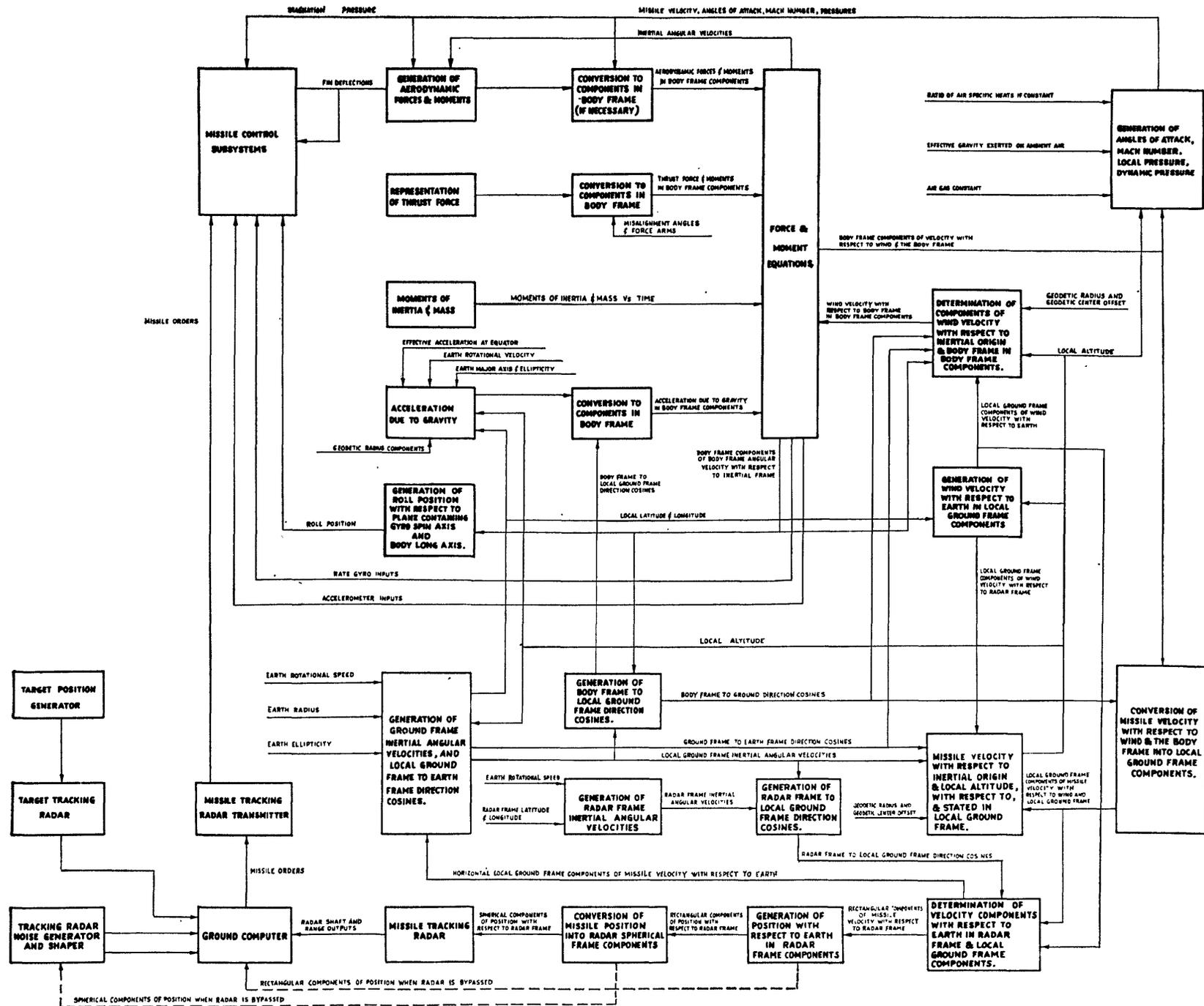


Fig. 3—Verbal diagram of a simulation of the guided missile system.

the earth rotational velocity, and the change of missile latitude and longitude with respect to time, this set of direction cosines is developed.

Another matrix of direction cosines is developed which relates the missile velocity components in the local ground frame to velocity components in the missile tracking radar frame. After the missile position is determined in this rectangular frame, the position is converted into components in the spherical frame. These components are inputs to the missile tracking radar.

Realistic simulation of thrust forces involves representation of starting and tail-off transients of the motor as well as variations in the thrust due to change in ambient air pressure. The force due to gravity, when realistically simulated, requires that the mass attraction vector be computed where missile latitude and altitude have been considered. In computation of the translational and rotational velocities and accelerations of the missile, the simulation needs to account for the change of mass and of inertia of the missile during its flight.

Fig. 3, from which a few details were discussed, represents somewhat realistically the guided missile system operation with the chief influencing factors. However, for such a simulation, about 600 amplifiers with associated nonlinear equipment would be required.

A more detailed description of the missile control subsystem, indicated by one box in the general simulation plan, is presented in Fig. 4. Control of the missile is achieved in each plane by comparing the command voltage to the sum of three feedback voltages.

The rate gyros and accelerometers are being simulated mathematically by transfer functions having second order denominators. The fin position voltages are seen to be functions of stagnation pressure as well as corresponding shaft position. The networks for each control channel are represented by a set of equations which together describe the behavior of the network. Then each steering amplifier and electrohydraulic assembly is expressed by a transfer function representing the dynamic characteristics of this link.

Computer Requirements as a Function of Complexity of the Analysis Scheme

The extent to which the complexity of the analysis scheme influences computer requirements may be observed from discussion of Fig. 5. This figure outlines briefly in diagram form the general form of a real-time flight analysis setup.

In this scheme, it is seen that the actual guided missile system acts as the input source. Quantities representative of accelerometer outputs, rate gyro outputs, missile altitude, and missile position are transmitted by telemetry or other means to a computer which uses these data to determine missile angular velocities, angular accelerations, aerodynamic moments, translational accelerations and velocities, angles of attack, and aerodynamic forces. If angles of attack are not measured

directly, external wind velocity as a function of location and altitude is needed in this computation. By use of this information, the position of the missile may be calculated, then compared to missile position as given by the missile tracking radar. If the results reasonably agree, according to errors allowable to data processing and telemetry, then the information, derived by processing the telemetered accelerometer, rate gyro, and altitude information, may be considered as useful in conducting an analysis of the actual guided missile system.

Telemetered information concerning voltages at various points in the missile control subsystems is conveyed along with the accelerometer and rate gyro information to a simulation of the missile control subsystems. Malfunctioning within a certain block of real missile equipment may then be determined by making the input to corresponding simulated block exactly equal to the block input in the actual missile. If the output of the simulated block is different from the real system block, then the real system block has malfunctioned. (Of course the assumption is made that the simulated system is quite validly represented.) At the same time, the difference between the unequal outputs may be determined in order to calculate an additional input to the real system causing it to perform as desired. For example, if the corresponding commands to the simulated missile and to the real missile are identical, but the corresponding voltages at the outputs of the receivers are greatly different, then it can be suspected that the receiving and translating circuits have malfunctioned.

Another section of the computer facility in the real-time analysis setup would calculate the aerodynamic force and moment coefficients starting with the use of theoretical expressions for the coefficients and the actual values for fin positions, angles of attack, dynamic pressure, and missile velocity. If these calculated values of the forces and moments are much different from the corresponding values as determined from rate gyro and accelerometer information, then commands would be generated which would force the real missile to respond only in the pitch or yaw plane or in some restricted fashion. Thus the complexity of the aerodynamic coefficient expressions would be greatly reduced and calculation of corrected partial derivatives would be simplified.

The mathematical relations as well as the required inputs and the produced outputs of the optimization scheme for the aerodynamic coefficients are shown in more detail in Fig. 6. For the computations which have to be performed for the coefficient optimization, it is assumed that sufficient different numerical values of the measured quantities are supplied for the determination of the wanted coefficients. These values have to be measured and processed in a time short enough that during this interval other variables, such as Mach number, are practically constant.

The technique of making the missile respond to pre-programmed commands is already being applied in the

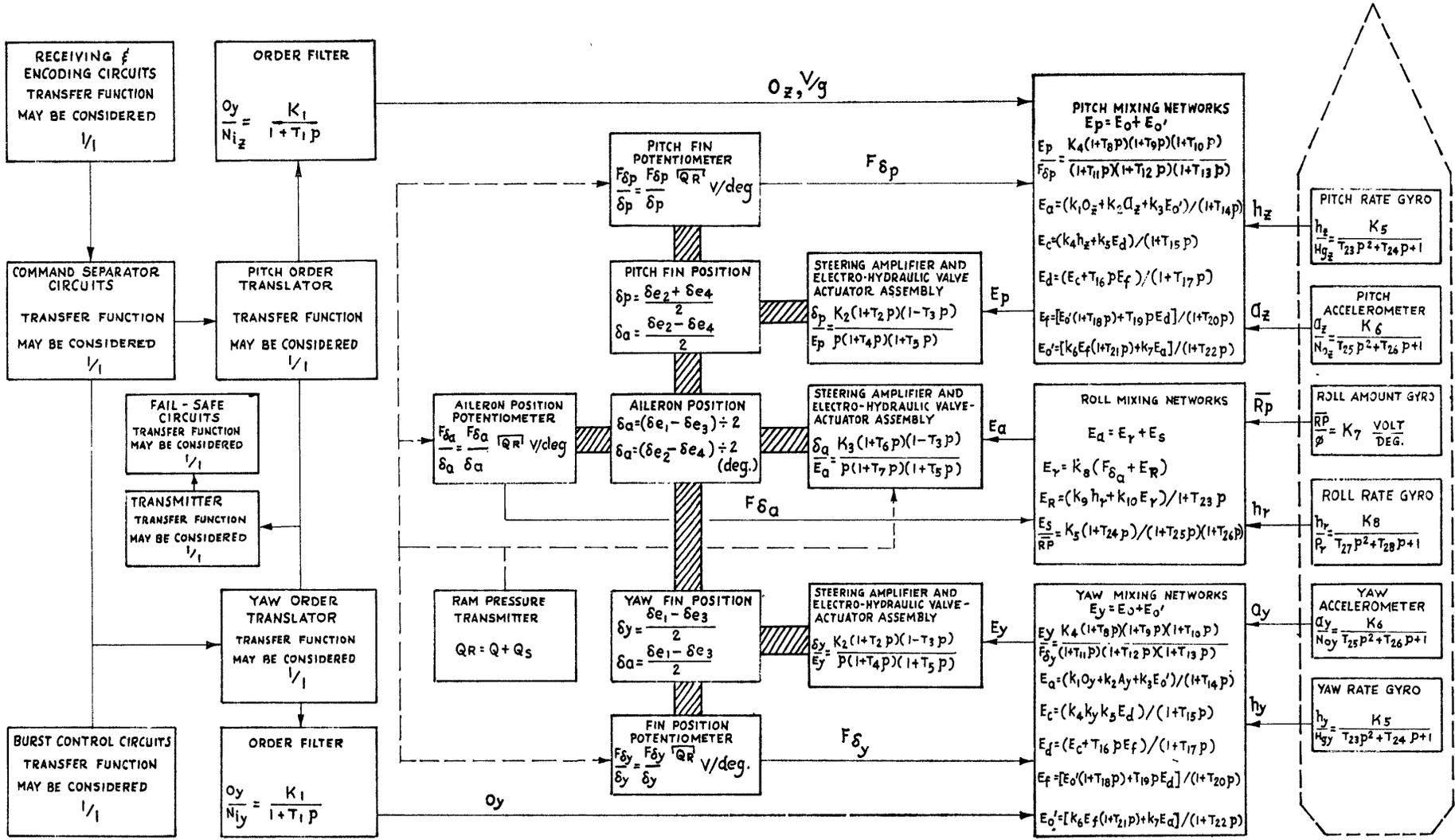


Fig. 4—Missile control system block diagram.

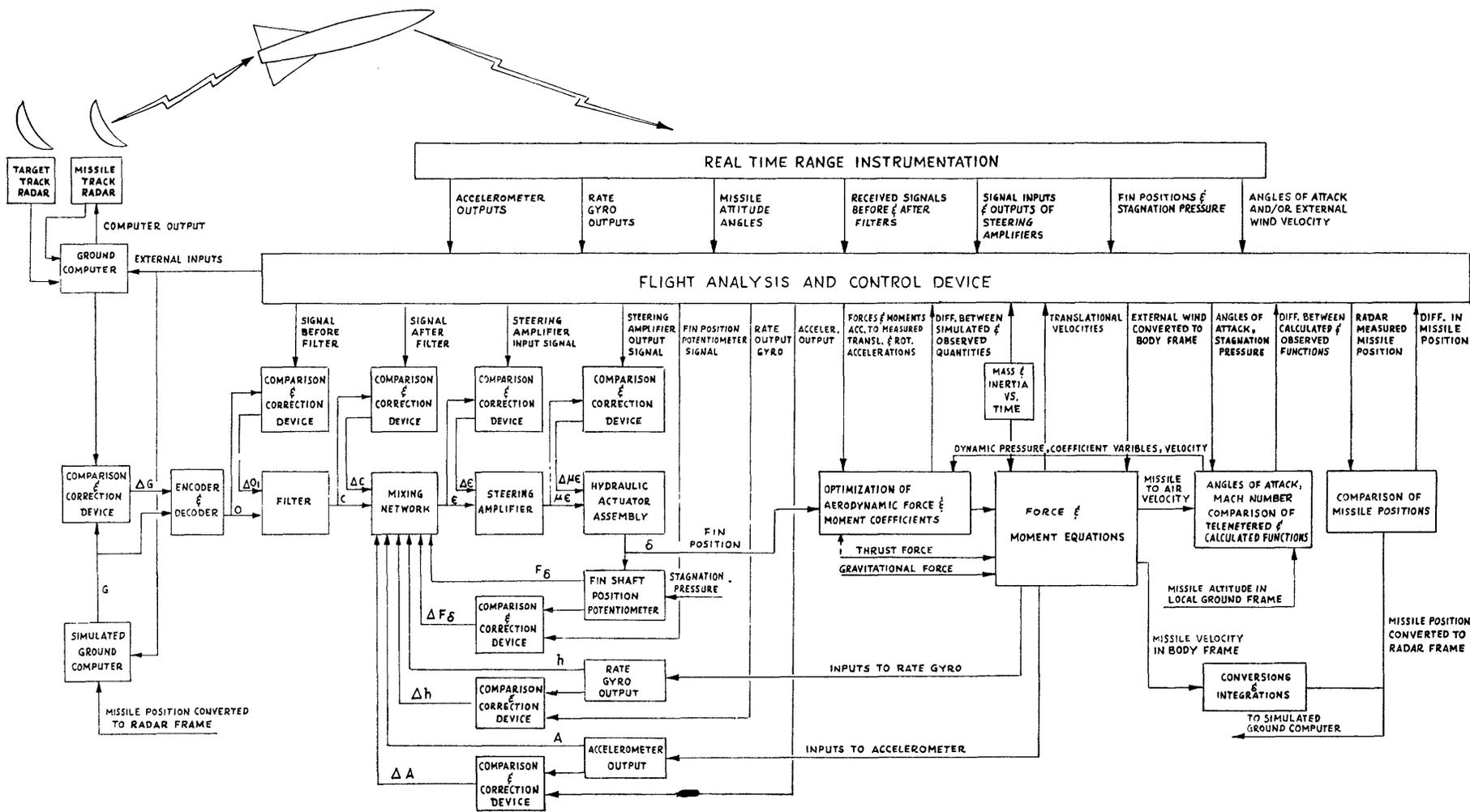


Fig. 5—Simplified presentation of analysis scheme.

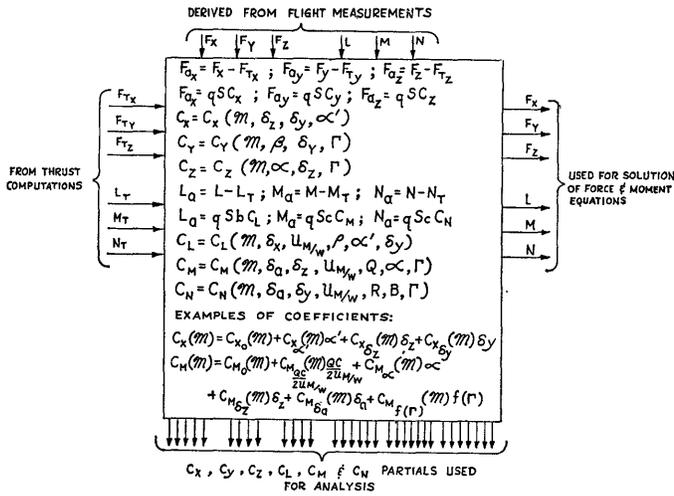


Fig. 6—Optimization of aerodynamic force and moment coefficients.

so-called aerodynamic runs in which guided missiles are flown with inactive guidance systems. However, this method does not exercise a closer control of the missile, whereby certain variables such as bank angle and undesired motion in several planes are kept to a minimum. Provisions are made that according to the motion of the missile, commands can be relayed to the missile keeping it in the motion which is desired for the check on its aerodynamic characteristics. If in one particular phase of the flight the lift coefficient shall be investigated, the missile is kept from lateral motion and rolling. For a small range of Mach number, the lift coefficient then becomes only a function of fin deflection and angle of attack. With accurate measurements of these two parameters, and the aerodynamic force by use of accelerometers, it is possible to obtain for this particular phase of the test good information on the lift coefficient. Other phases in the test can be used in a similar way for the determination of other force and moment coefficients.

The foregoing discussion may convey some idea of the complexity of a real-time flight analysis computer setup. Considering the processes of comparison, and generation of forcing command, as well as circuits for automatic optimization of the simulated coefficients, it is reasonable to estimate the analog computer requirements for the scheme discussed at about 700 amplifiers

with the usual proportion of associated nonlinear equipment.

CONCLUSION

It is felt that present-day computers could handle real-time flight analysis on a somewhat limited basis, either switching in automatically additional information channels as required, or performing a comparing analysis on a few variables at a time. However, a full-fledged real-time analysis giving a comprehensive, quantitative evaluation, adequate to understand the "how" and "why" of all flight events, may exceed present computer capability.

The described analysis scheme, which is one of various possibilities, requires approximately 1300 amplifiers and proportional nonlinear equipment. The mechanization and solution of a problem of this size is feasible with existing analog computers, if enough time is available to make the proper checks and adjustments before a run is made.

However, when tied up with actual missile firings, no additional time can be permitted for the readying of the computers, because there are already too many other delays caused by instrumentation which aggravate the flight experiment of missiles.

There seem to be two possible answers to this problem. One is that digital computers be large enough to solve in real time a set of equations of the indicated order of magnitude. The problem of having digital computers always ready on a standby basis could be more easily solved. The second possibility would be the addition of automatic checking and adjustment features to the analogs working reliably enough to have an accurate performance of the analog computer assured every time a missile is ready to go. The combination of analog and digital computers may offer some promise.

Although the outlined computer requirements may appear formidable, the benefits to be derived from these computer capabilities make the effort to be expended worthwhile. The achievement of real-time flight analysis for conclusive evaluation of missile flights and for the control of these expensive and complex experiments will bring an advance to all guided missile programs which is so significant that it should be given fullest attention.

Industry's Role in Supporting High-School Science Programs

J. O. PAIVINEN†, *Chairman*

AN introductory statement was presented by the chairman, a member of the WJCC Technical Program Committee. He pointed out that the objective of the meeting was to bring information on cooperative industry-school programs to the attention of interested computer people as well as invited high-school principals from the Bay area (20 out of 100 invitees attended). Through this meeting, the computer industry expressed interest and willingness to cooperate in additional similar programs wherever the need may be evidenced by schools or teachers. It is not the intent to restrict such programs to involve computer technology only, but to participate in broad science programs including a computer portion. In this way, students aiming at careers in science will be exposed to computer techniques as one of their available tools, while potential computer people will have attained a broader grasp of basic technology.

Dr. Paul Hurd, *Chairman (Professor of Science Education at Stanford University, Calif.)*

Dr. Hurd portrayed the program as illustrating examples of cooperation between industry, professional people, and civic groups, working with high schools to enrich the technical exposure of high-school students. The needs of our society have expanded from one in 300 employees in 1900 needing a scientific, engineering, or mathematical training for their positions to one in 37 needing such training in 1958. The trend is still increasing for such training. Conversely, schools need the help of industry and civic groups to establish such programs not only to avoid increasing the teacher's burdens but to bring to the teachers a knowledge and background in newer areas of technology and science. Examples in the program will point out that the success of cooperative programs depends always on both an interested teacher and interested community members to participate in the extra work. In summary, the examples in the program show how a new dimension can be brought to education in which the community itself takes an active part by contributing time, technical skills, and support to enrich high-school and junior high-school programs.

† General Elec. Co., Palo Alto, Calif.

Darryl Littlefield (*Physics and Science Teacher, Livermore Union High School, Livermore, Calif.*)

An elective course is offered as part of the curriculum at the high school to cover applications and programming of the IBM 650. The objective of the course is to convey a real appreciation of the power and application of computers in modern research, engineering, and business. Problems are generally of a nature that would not normally be encountered by high-school students. Littlefield and Herman Thomas of the Math Department cooperate in the project.

The course stemmed from Littlefield's contact with the 650 at the Lawrence Radiation Laboratory (University of California) at Livermore and his consequent interest toward using the community resources to establish programs of real educational growth for his students. Dr. Sidney Fernbach of the Radiation Laboratory acts as consultant for the course while the Laboratory makes available machine time for use by the students. The Radiation Laboratory has also made available charts, models, and speakers for the classroom.

The benefit of a week-long seminar on computer instruction for teachers and engineers, arranged by Dr. Van Etta of Hughes Aircraft Co. of Los Angeles, was stressed. As another part of the school program, the construction of a relay binary adder-subtractor by two students using parts donated by Pacific Telephone and Telegraph and IBM was mentioned. Dr. LaFrangi of the Radiation Laboratory acted as advisor.

Joanne Watkins (*Senior Student, Livermore High School, Livermore, Calif.*)

Programming to solve the motion of a projectile in vacuum was described and flow charts, coding sheets, and plots of trajectories for varying initial elevations were shown. The course was enjoyable because it was new and interesting and has given a background applicable even to other computers, together with an appreciation for what jobs can best be done on computers.

Doug McMilin (*Junior Student, Livermore High School, Livermore, Calif.*)

A payroll calculation on the 650 was described and the input and output card layouts, flow charts, and programming sheets shown. The main points of the calculations were described. Personal access to a ma-

chine was found fascinating and mathematical principles in practical use enjoyed. The training in logical thinking was valuable and resulted from working a problem as a whole, with simultaneous attention to detail, while observing the strict rules of computer programming.

Tom Doyen and Ross Harrower (*Students, Livermore High School, Livermore, Calif.*)

The operation of the binary adder they had constructed was explained, including mention of subtraction by complements. A demonstration followed the description.

Dr. Sidney Fernbach (*Staff Member, Lawrence Radiation Laboratory, Livermore, Calif.*)

Other cooperative programs supported by the Radiation Laboratory were described. A high-school committee attempts to get students interested in individual projects, providing advisers on such topics as rocketry and nuclear energy. Speakers are also provided for science clubs and classrooms, and student tours are arranged at both Berkeley and Livermore Laboratories. Training of high-school teachers in modern physics is arranged through summertime and parttime employment where three lectures per week are provided on computers, modern physics, and chemistry.

High-school students have been employed during the summer to convey a working knowledge of laboratory procedures; this summer, participation requiring even Q clearance will be arranged.

Finally, general scholarship is encouraged by trying to increase student enrollment in college preparatory courses and by arranging for awards from the community for outstanding students.

Henry Martin (*Physics Teacher, Palo Alto High School, Palo Alto, Calif.*)

Five to six years ago, industry help was sought in guiding the efforts of students who evidenced their eagerness to learn by after-hours and Saturday use of school facilities and the laboratory. However, little success was encountered due to the lack of any concrete program that Martin and the short-handed science staff had time to generate. This stalemate was broken in 1957 by a consultant from the Joe Berg Foundation, 1712 South Michigan Avenue, Chicago 16, Ill., an organization that volunteers help in establishing an initial relation between industry and schools. This resulted in the Palo Alto Science Seminar, which is an entirely locally conducted program. Fifty-two weekly sessions are conducted throughout the year, with 1 to 1½ hours of each session devoted to a general program given by an industry expert or a panel discussion with subsequent group discussions specializing in chemistry, geology, biology, physics, engineering, and mathematics. A student joins at about the tenth grade level and will ulti-

mately undertake a project of interest to him with the counselling available of a volunteer industry professional. Industry also provides equipment, speakers, demonstrations, and sometimes facilities where the student often works in proximity to an engineer or a scientist. Advantages to the students include familiarization with work conditions and opportunities, opportunity for individual creative study (since projects are not group efforts), personal satisfaction of a hobby with prestige value, and the opportunity to seek and earn scholarships.

Advantages to industry include an early encouraging hand to potential Ph.D. scientists, an opportunity to demonstrate to a broad slice of the community that scientists are normal human beings with families and a sense of humor, and an opportunity to contribute to the community and the future of our country.

Other programs recommended in closing for industry's consideration are:

- 1) Participation of students as well as teachers at professional dinners and educational programs arranged by industry.
- 2) The assignment to some member of a company management team the specific responsibility of support and cooperation in educational affairs.
- 3) The opportunity for students on field trips to spend sufficient time with the engineers and scientists to gain some insight into the significance of the projects or laboratories visited.
- 4) Summer jobs for students which will increase their knowledge of industry practices and expectations.
- 5) Screening and testing programs to select outstanding students to receive substantial scholarships.
- 6) Summer schools at local universities so that students can see what they will be up against in the future.
- 7) Public competitions for scholarship so that recognition could be given in a manner comparable to school athletic letters and Father's Club dinners for athletes.

Larry Hubbart (*Student, Palo Alto High School, Palo Alto, Calif.*)

Hubbart's construction of a test stand as well as his subsequent experiments in measuring the lift of rotating airfoils (similar to an inverted pie plate) were described. An engineer from Hiller Helicopter Co. acts as adviser: his contribution was described as helping to suggest directions of investigation to pursue as well as to help maintain morale when the project appears to bog down. The test stand consisted of a counter-balanced scale with a drive motor and a photocell rpm counter. Airfoils of varying shapes and textures (some with added ducts) have been measured. The project was described as representing a success "even if it never leaves the

ground" due to the experience it provided in how to conduct an experiment and how to present the results.

Mike Macauley (*Student, Palo Alto High School, Palo Alto, Calif.*)

An experiment in combatting muscle fatigue by injected chemical solutions is being conducted. A rabbit is anesthetized and muscle contractions are caused by mild electric shock until exhaustion occurs. Injection of a mild hydrogen peroxide solution results in acceleration of the recovery: recovery periods range from 3 minutes for a 3 per cent solution to 0.3 second for 15 per cent. Hydrogen peroxide was chosen due to the safe decomposition products; procedures are followed to safeguard against bubble formation in the blood.

The insights gained into medicine and medical research have reinforced interest in entering the field of medicine.

Wallace Burton (*Engineer, R-S Electronics, Palo Alto, Calif.*)

The YMCA Men's Club, in seeking to further encourage science students and to provide recognition for work done in the Palo Alto Science Seminar, chose to present an annual Palo Alto Industry-Youth Science Show. The show presents awards to outstanding student science projects and also presents an opportunity for exhibits by local industrial firms.

Announcements of the show with award categories are sent to all the high schools in the Palo Alto district and a participation of about 10 per cent of the students is experienced (the only other such event, the Bay Area Science Fair, accommodates only 1 per cent of the Palo Alto students). Three categories of participation are provided: 1) physical sciences, 2) biological sciences, and 3) technical reports (given orally in competition). About

one-half of the industrial firms contacted responded favorably to requests for financial support and industrial exhibits. In the first year, industry met \$650 of the \$850 expenditures, and this year the show will be self-supporting together with an expected elimination of the club deficit.

The Science Show provides wide recognition to students through substantial adult attendance as well as the interest and awareness of the industry people who work as judges and exhibitors for the show. The Men's Club provides the planning and organizing for the show so that the participation of the various schools can be drawn together into an integrated plan, this having been found necessary to obtain effective support from industry.

Even smaller communities can establish Science Shows. Community members are sure to lend help if given an opportunity while local commercial businessmen would undoubtedly help support the expenses.

A member of the audience (*Staff Member, Systems Development Corp., Los Angeles, Calif.*)

Educational support activities performed by the staff at Systems Development Corporation were described. At the junior high-school level, package lectures are available in 1) data processing, 2) automatic feedback, 3) binary and octal notation in programming, etc. For the high-school level, talks are available in 1) concepts of programming, 2) environmental simulation in air defense, etc. City colleges have available 1) computer design philosophies, 2) logic design, 3) computer system design, etc. A symposium has been held for teachers in "Implications of the Computer Age for Teaching Mathematics" while other lectures and special courses on programming and advanced math for high-school students have also been provided.
