Systems Reference Library

IBM System/360 Operating System
FORTRAN IV Library—
Mathematical and Service Subprograms

Program Number 360S-LM-501

This publication describes the mathematical and service subprograms contained in the library supplied with Basic FORTRAN IV and FORTRAN IV compilers. Detailed information on each mathematical routine is provided: its algorithm, speed, accuracy, range, and error conditions. Information on the use of library subprograms in either a FORTRAN or assembler language program is also provided.

Preface

The purpose of this publication is to describe the mathematical and service subprograms in the FORTRAN IV library supplied with Basic FORTRAN IV (OS) and FORTRAN IV (OS). As an aid to the programmer in his use of this publication, the contents of each chapter is described briefly below:

1. "Introduction" describes the three types of subprograms in the FORTRAN library (FORTLIB)and defines their use in either a FORTRAN or an assembler language program.
2. "Mathematical Subprograms" describes the subprograms which perform computations frequently needed by the programmer. A mathematical subprogram is invoked *explicitly* whenever one of its entry names appears in a source statement, or *implicitly* through use of certain notation in the source statement.
3. "Service Subprograms" contains information about those subprograms which perform utility functions or test machine indicators.
4. "Algorithms" contains information about the method used in the library to compute a mathematical function and describes the effect of an argument error upon the accuracy of the answer returned.
5. "Performance Statistics" gives accuracy and timing statistics for the explicitly called mathematical subprograms.
6. The appendixes provide a list of diagnostic messages, a list of module names, a sample storage printout, storage estimates, and information for the assembler language programmer.

It is assumed that the reader is familiar with one of the following publications:

IBM System/360 FORTRAN IV Language, Form C28-6515

IBM System/360 Basic FORTRAN IV Language, Form C-28-6629

IBM System/360 Operating System: Assembler Language, Form C28-6514

In addition, references are made within this publication to information contained in the following publications:

IBM System/360 Principles of Operation, Form A22-6821

IBM/360 Operating System: Supervisor and Data Management Macro-Instructions, Form C28-6647

IBM System/360 Operating System: Basic FORTRAN IV (E) Programmer's Guide, Form C28-6603

IBM System/360 Operating System: FORTRAN IV (G and H) Programmer's Guide, Form C28-6817

Standard mathematical notation is used in this publication. The reader is expected to be familiar with this notation and with common mathematical terminology.

SECOND EDITION (September 1972)

This is a reprint of GC28-6818-0 incorporating changes released in Technical Newsletter GN28-0589, dated June 1, 1970.

This edition corresponds to Release 19 of the IBM System/360 Operating System.

Changes are periodically made to the specifications herein; any such changes will be reported in subsequent revisions or Technical Newsletters.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM Branch Office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, 1271 Avenue of the Americas, New York, New York 10020.

Contents

Illustrations

TABLES

FIGURES

The FORTRAN IV library contains three types of subprograms: (1) mathematical functions such as SIN and SQRT, (2) service subprograms such as DUMP and EXIT, and (3) interface routines such as IBCOM and IBERR.

The library subprograms may be used in either a FORTRAN or an assembler language program. (Appendix A contains calling information for the assembler language programmer.) In FORTRAN, calls to the library subprograms are either at the programmer's request or in response to the program requirements. Subprograms required by the program being compiled are provided by a linkage editor or loader, which takes the subprograms from the library.

Mathematical Function Subprograms

Many commonly used mathematical functions or calculations are provided by the FORTRAN IV language. Programmer requests for these calculations are satisfied by a compiler in one of two ways: by inserting machine-language code directly within the object module; or by inserting an external reference to a library subprogram that contains the necessary code, and having the linkage editor include that subprogram in the load module. The first method is called *in-line*, the second *out-of-line*.

This publication discusses only out-of-line routines—these mathematical functions contained in the library. The American National Standards Institute (ANSI) defines several arithmetic functions—such as absolute value (ABS), positive difference (DIM), and transfer of sign (SIGN)—as *intrinsic functions*. For the most part, code for these functions is inserted in-line by the FORTRAN compiler at the point in the source module where the function's symbolic name is used. Consequently, they are not discussed in this book. However, the following ANSI-defined *intrinsic* functions have been implemented as a part of the FORTRAN IV library and are provided out-of-line for all systems: MAX0/MIN0, AMAX0/AMIN0, MAX1/MIN1, AMAX1/AMIN1, and DMAX1/DMIN1. These are, therefore, documented in this publication. In Basic FORTRAN IV only, four other *intrinsic* functions (MOD, AMOD, AINT, and IFIX/INT/IDINT) are provided out-of-line as a part of the library and are also covered here.

Service Subroutine Subprograms

Each of the service subprograms corresponds to a subroutine form as defined by a SUBROUTINE statement in a FORTRAN source module. These subprograms perform machine indicator tests and utility functions and may or may not return a value to the calling module. These subroutines are discussed in this publication.

Interface Routines

The library contains certain input/output and error processing routines that act as interfaces with the compiled program and the operating system. Frequently, the mathematical and service subprograms require assistance from these routines for input/output, interruption, and error processing. Storage estimates for these routines are included in Appendix B.

Mathematical Subprograms

The mathematical subprograms supplied in the FORTRAN library perform computations frequently needed by the programmer. The mathematical subprograms are called in two ways: explicitly, when the programmer includes the appropriate entry name in a source language statement (see Table 1); and implicitly, when certain notation (e.g., raising a number to a power) appears within a source language statement (see Table 6).

The following text describes the individual mathematical subprograms and explains their use in a FORTRAN program. Detailed information about the actual method of computation used in each subprogram, the performance of the subprogram, interruption and error procedures, and storage estimates can be found elsewhere in this publication.

Table 1. Explicitly Called Mathematical Subprograms

General Function	Specific Function	Entry Name(s)
Logarithmic and exponential subprograms (described in Table 2)	Exponential	EXP DEXP CEXP* CDEXP*
	Logarithmic, common and natural	ALOG, ALOG10 DLOG, DLOG10 CLOG* CDLOG*
	Square root	SQRT DSQRT CSQRT* CDSQRT*
Trigonometric subprograms (described in Table 3)	Arcsine and arccosine	ARSIN*, ARCOS* DARSIN*, DARCOS*
	Arctangent	ATAN, ATAN2* DATAN, DATAN2*
	Sine and cosine	SIN, COS DSIN, DCOS CSIN*, CCOS* CDSIN*, CDCOS*
	Tangent and cotangent	TAN*, COTAN* DTAN*, DCOTAN*
Hyperbolic function subprograms (described in Table 4)	Hyperbolic sine and cosine	SINH*, COSH* DSINH*, DCOSH*
	Hyperbolic tangent	TANH DTANH
Miscellaneous subprograms (described in Table 5)	Absolute value	CABS* CDABS*
	Error function	ERF*, ERFC* DERF*, DERFC*
	Gamma and log-gamma	GAMMA*, ALGAMA* DGAMMA*, DLGAMA*
	Maximum and minimum value	AMAX0, AMIN0, MAX0, MIN0 AMAX1, AMIN1, MAX1, MIN1 DMAX1, DMIN1
	Modular arithmetic	MOD AMOD, DMOD
	Truncation	AINT INT, IDINT
*Not available in Basic FORTRAN IV (OS)		

Explicitly Called Subprograms

Each explicitly called subprogram performs one or more mathematical functions. Each mathematical function is identified by a unique entry name.

A subprogram is called whenever the appropriate entry name is included in a FORTRAN arithmetic expression. The programmer must also supply one or more arguments. These arguments follow the entry name and are separated by commas; the list of arguments is enclosed in parentheses.

For example, the source statement:

```
RESULT = SIN (RADIAN)
```

causes the sine and cosine subprogram to be called. The sine of the value in RADIAN is computed and the function value is stored in RESULT.

In the following example, the square root subprogram is called to compute the square root of the value in AMNT. The function value is then added to the value in STOCK and the result is stored in ANS.

```
ANS = STOCK + SQRT (AMNT)
```

The explicity called subprograms are described in Tables 2 through 5. The following information is provided:

General Function: This column states the nature of the computation performed by the subprogram.

Entry Name: This column gives the entry name that the programmer must use to call the subprogram. A subprogram may have more than one entry name; the particular entry name used depends upon the computation to be performed. For example, the sine and cosine subprogram has two entry names: SIN and COS. If the sine is to be computed, entry name SIN is used; if the cosine is to be computed, entry name COS is used.

Table 2. Logarithmic and Exponential Subprograms *(Part 1 of 2)*

General Function	Entry Name	Subset	Definition	Argument(s)			Function Value Type[1] and Range[4]	Error Code
				No.	Type[1]	Range		
Common and natural logarithm	ALOG	Yes	$y=\log_e x$ or $y=\ln x$	1	REAL *4	$x > 0$	REAL *4 $y \geqq -180.218$ $y \leqq 174.673$	253
	ALOG10	Yes	$y=\log_{10} x$	1	REAL *4	$x > 0$	REAL *4 $y \geqq -78.268$ $y \leqq 75.859$	253
	DLOG	Yes	$y=\log_e x$ or $y=\ln x$	1	REAL *8	$x > 0$	REAL *8 $y \geqq -180.218$ $y \leqq 174.673$	263
	DLOG10	Yes	$y=\log_{10} x$	1	REAL *8	$x > 0$	REAL *8 $y \geqq -78.268$ $y \leqq 75.859$	263
	CLOG	No	$y=PV \log_e (z)$ See Note 2	1	COMPLEX *8	$z \neq 0 + 0i$	COMPLEX *8 $y_1 \geqq -180.218$ $y_1 \leqq 175.021$ $-\pi \leqq y_2 \leqq \pi$	273
	CDLOG	No	$y=PV \log_e (z)$ See Note 2	1	COMPLEX *16	$z \neq 0 + 0i$	COMPLEX *16 $y_1 \geqq -180.218$ $y_1 \leqq 175.021$ $-\pi \leqq y_2 \leqq \pi$	283
Exponential	EXP	Yes	$y=e^x$	1	REAL *4	$x \leqq 174.673$	REAL *4 $0 \leqq y \leqq \gamma$	252
	DEXP	Yes	$y=e^x$	1	REAL *8	$x \leqq 174.673$	REAL *8 $0 \leqq y \leqq \gamma$	262
	CEXP	No	$y=e^z$ See Note 3	1	COMPLEX *8	$x_1 \leqq 174.673$ $\lvert x_2 \rvert < (2^{13} \cdot \pi)$	COMPLEX *8 $-\gamma \leqq y_1, y_2 \leqq \gamma$	271, 272
	CDEXP	No	$y=e^z$ See Note 3	1	COMPLEX *16	$x_1 \leqq 174.673$ $\lvert x_2 \rvert < (2^{50} \cdot \pi)$	COMPLEX *16 $-\gamma \leqq y_1, y_2 \leqq \gamma$	281, 282

NOTES: (See end of table.)

Table 2. Logarithmic and Exponential Subprograms *(Part 2 of 2)*

General Function	Entry Name	Sub-set	Definition	Argument(s)			Function Value Type[1] and Range[4]	Error Code
				No.	Type[1]	Range		
Square root	SQRT	Yes	$y=\sqrt{x}$ or $y=x^{1/2}$	1	REAL *4	$x \geqq 0$	REAL *4 $0 \leqq y \leqq \gamma^{1/2}$	251
	DSQRT	Yes	$y=\sqrt{x}$ or $y=x^{1/2}$	1	REAL *8	$x \geqq 0$	REAL *8 $0 \leqq y \leqq \gamma^{1/2}$	261
	CSQRT	No	$y=\sqrt{z}$ or $y=z^{1/2}$ See Note 3	1	COMPLEX *8	any COMPLEX argument	COMPLEX *8 $0 \leqq y_1 \leqq 1.0987\,(\gamma^{1/2})$ $\lvert y_2 \rvert \leqq 1.0987\,(\gamma^{1/2})$	—
	CDSQRT	No	$y=\sqrt{z}$ or $y=z^{1/2}$ See Note 3	1	COMPLEX *16	any COMPLEX argument	COMPLEX *16 $0 \leqq y_1 \leqq 1.0987\,(\gamma^{1/2})$ $\lvert y_2 \rvert \leqq 1.0987\,(\gamma^{1/2})$	—

NOTES:

[1] REAL*4 and REAL*8 arguments correspond to REAL and DOUBLE PRECISION arguments, respectively, in Basic FORTRAN IV. Complex arguments cannot be used in a Basic FORTRAN IV program.

[2] PV = principal value. The answer given ($y_1 + y_2 i$) is that one whose imaginary part (y_2) lies between $-\pi$ and $+\pi$. More specifically: $-\pi < y_2 \leqq \pi$, unless $x_1 < 0$ and $x_2 = -0$, in which case, $y_2 = -\pi$.

[3] z is a complex number of the form $x_1 + x_2 i$.

[4] $\gamma = 16^{63}\,(1 - 16^{-6})$ for regular precision routines, and $16^{63}\,(1 - 16^{-14})$ for double-precision routines.

Subset: This column indicates those subprograms that belong to the Basic FORTRAN IV library. Unless otherwise indicated, all such subprograms also belong to the FORTRAN IV libraries.

Definition: This column gives a mathematical equation that represents the computation. An alternate equation is given in those cases where there is another way of representing the computation in mathematical notation. (For example, the square root can be represented either as $y = \sqrt{x}$ or $y = x^{1/2}$.)

Argument Number: This column gives the number of arguments that the programmer must supply.

Argument Type: This column describes the type and length of the argument(s). INTEGER, REAL, and COMPLEX represent the type; the notations *4, *8, and *16 represent the size of the argument in number of storage locations.

Argument Range: This column gives the valid range for arguments. If an argument is not within this range, an error message is issued and execution of the load module is terminated unless the extended error handling facility was specified at system generation (see *FORTRAN IV (G and H) Programmer's Guide* listed in the Preface for a full description of this facility). Appendix C of this publication contains a description of the error messages.

Function Value Type and Range: This column describes the type and range of the function value returned by the subprogram. Type notation used is the same as that used for the argument type. Range symbol $\gamma = 16^{63}(1-16^{-6})$ for regular precision routines, and $16^{63}(1-16^{-14})$ for double-precision routines.

Error Code: This column gives the number of the message issued when an error occurs. Appendix C contains a description of the error messages.

Throughout this publication, the following approximate values are represented by $2^{18} \cdot \pi$ and $2^{50} \cdot \pi$:

$2^{18} \cdot \pi = .82354966406249996D + 06$

$2^{50} \cdot \pi = .35371188737802239D + 16$

Detailed information for the assembler language programmer is given in Appendix A.

Table 3. Trigonometric Subprograms. (*Part 1 of 2*)

General Function	Entry Name	Subset	Definition	Argument(s)			Function Value Type[1] and Range[4]	Error Code
				No.	Type[1]	Range		
Arcsine and arccosine	ARSIN	No	y=arcsin (x)	1	REAL *4	$\|x\| \leqq 1$	REAL *4 (in radians) $-\frac{\pi}{2} \leqq y \leqq \frac{\pi}{2}$	257
	ARCOS	No	y=arccos (x)	1	REAL *4	$\|x\| \leqq 1$	REAL *4 (in radians) $0 \leqq y \leqq \pi$	257
	DARSIN	No	y=arcsin (x)	1	REAL *8	$\|x\| \leqq 1$	REAL *8 (in radians) $-\frac{\pi}{2} \leqq y \leqq \frac{\pi}{2}$	267
	DARCOS	No	y=arccos (x)	1	REAL *8	$\|x\| \leqq 1$	REAL *8 (in radians) $0 \leqq y \leqq \pi$	267
Arctangent	ATAN	Yes	y=arctan (x)	1	REAL *4	any REAL argument	REAL *4 (in radians) $-\frac{\pi}{2} \leqq y \leqq \frac{\pi}{2}$	—
	ATAN2	No	$y=\arctan\left(\frac{x_1}{x_2}\right)$	2	REAL *4	any REAL arguments (except 0, 0)	REAL *4 (in radians) $-\pi < y \leqq \pi$	255
	DATAN	Yes	y=arctan (x)	1	REAL *8	any REAL argument	REAL *8 (in radians) $-\frac{\pi}{2} \leqq y \leqq \frac{\pi}{2}$	—
	DATAN2	No	$y=\arctan\left(\frac{x_1}{x_2}\right)$	2	REAL *8	any REAL arguments (except 0, 0)	REAL *8 (in radians) $-\pi < y \leqq \pi$	265
Sine and cosine	SIN	Yes	y=sin (x)	1	REAL *4 (in radians)	$\|x\| < (2^{18} \cdot \pi)$	REAL *4 $-1 \leqq y \leqq 1$	254
	COS	Yes	y=cos (x)	1	REAL *4 (in radians)	$\|x\| < (2^{18} \cdot \pi)$	REAL *4 $-1 \leqq y \leqq 1$	254
	DSIN	Yes	y=sin (x)	1	REAL *8 (in radians)	$\|x\| < (2^{50} \cdot \pi)$	REAL *8 $-1 \leqq y \leqq 1$	264
	DCOS	Yes	y=cos (x)	1	REAL *8 (in radians)	$\|x\| < (2^{50} \cdot \pi)$	REAL *8 $-1 \leqq y \leqq 1$	264
NOTES: (See end of table.)								

Table 3. Trigonometric Subprograms (*Part 2 of 2*)

General Function	Entry Name	Subset	Definition	Argument(s) No.	Argument(s) Type[1]	Argument(s) Range	Function Value Type[1] and Range[4]	Error Code
Sine and cosine (continued)	CSIN	No	y=sin (z) See Note 2	1	COMPLEX *8 (in radians)	$\|x_1\| < (2^{18} \cdot \pi)$ $\|x_2\| \leqq 174.673$	COMPLEX *8 $-\gamma \leqq y_1, y_2 \leqq \gamma$	274, 275
	CCOS	No	y=cos (z) See Note 2	1	COMPLEX *8 (in radians)	$\|x_1\| < (2^{18} \cdot \pi)$ $\|x_2\| \leqq 174.673$	COMPLEX *8 $-\gamma \leqq y_1, y_2 \leqq \gamma$	274, 275
	CDSIN	No	y=sin (z) See Note 2	1	COMPLEX *16 (in radians)	$\|x_1\| < (2^{50} \cdot \pi)$ $\|x_2\| \leqq 174.673$	COMPLEX *16 $-\gamma \leqq y_1, y_2 \leqq \gamma$	284, 285
	CDCOS	No	y=cos (z) See Note 2	1	COMPLEX *16 (in radians)	$\|x_1\| < (2^{50} \cdot \pi)$ $\|x_2\| \leqq 174.673$	COMPLEX *16 $-\gamma \leqq y_1, y_2 \leqq \gamma$	284, 285
Tangent and cotangent	TAN	No	y=tan (x)	1	REAL *4 (in radians)	$\|x\| < (2^{18} \cdot \pi)$ See Note 3	REAL *4 $-\gamma \leqq y \leqq \gamma$	258, 259
	COTAN	No	y=cotan (x)	1	REAL *4 (in radians)	$\|x\| < (2^{18} \cdot \pi)$ See Note 3	REAL *4 $-\gamma \leqq y \leqq \gamma$	258, 259
	DTAN	No	y=tan (x)	1	REAL *8 (in radians)	$\|x\| < (2^{50} \cdot \pi)$ See Note 3	REAL *8 $-\gamma \leqq y \leqq \gamma$	268, 269
	DCOTAN	No	y=cotan (x)	1	REAL *8 (in radians)	$\|x\| < (2^{50} \cdot \pi)$ See Note 3	REAL *8 $-\gamma \leqq y \leqq \gamma$	268, 269

NOTES:

[1] REAL *4 and REAL *8 arguments correspond to REAL and DOUBLE PRECISION arguments, respectively, in Basic FORTRAN IV. Complex arguments cannot be used in a Basic FORTRAN IV program.

[2] z is a complex number of the form $x_1 + x_2 i$.

[3] The argument for the cotangent functions may not approach a multiple of π; the argument for the tangent functions may not approach an odd multiple of $\pi/2$.

[4] $\gamma = 16^{68} (1 - 16^{-6})$ for regular precision routines, and $16^{68} (1 - 16^{-14})$ for double-precision routines.

Table 4. Hyperbolic Function Subprograms

General Function	Entry Name	Subset	Definition	Argument(s) No.	Argument(s) Type[1]	Argument(s) Range	Function Value Type[1] and Range[2]	Error Code
Hyperbolic sine and cosine	SINH	No	$y = \frac{e^x - e^{-x}}{2}$	1	REAL *4	$\|x\| < 175.366$	REAL *4 $-\gamma \leqq y \leqq \gamma$	256
	COSH	No	$y = \frac{e^x + e^{-x}}{2}$	1	REAL *4	$\|x\| < 175.366$	REAL *4 $1 \leqq y \leqq \gamma$	256
	DSINH	No	$y = \frac{e^x - e^{-x}}{2}$	1	REAL *8	$\|x\| < 175.366$	REAL *8 $-\gamma \leqq y \leqq \gamma$	266
	DCOSH	No	$y = \frac{e^x + e^{-x}}{2}$	1	REAL *8	$\|x\| < 175.366$	REAL *8 $1 \leqq y \leqq \gamma$	266
Hyperbolic tangent	TANH	Yes	$y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	1	REAL *4	any REAL argument	REAL *4 $-1 \leqq y \leqq 1$	—
	DTANH	Yes	$y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	1	REAL *8	any REAL argument	REAL *8 $-1 \leqq y \leqq 1$	—

NOTES:

[1] REAL *4 and REAL *8 arguments correspond to REAL and DOUBLE PRECISION arguments, respectively, in Basic FORTRAN IV. Complex arguments cannot be used in a Basic FORTRAN IV program.

[2] $\gamma = 16^{68} (1 - 16^{-6})$ for regular precision routines, and $16^{68} (1 - 16^{-14})$ for double-precision routines.

Table 5. Miscellaneous Mathematical Subprograms (*Part 1 of 2*)

General Function	Entry Name	Sub-set	Definition	Argument(s) No.	Argument(s) Type[1]	Argument(s) Range	Function Value Type[2] and Range[4]	Error Code
Absolute value	CABS	No	$y=\|z\|=(x_1^2+x_2^2)^{1/2}$	1	COMPLEX *8	any COMPLEX argument See Note 2	REAL *4 $0 \leqq y_1 \leqq \gamma$ $y_2 = 0$	—
	CDABS	No	$y=\|z\|=(x_1^2+x_2^2)^{1/2}$	1	COMPLEX *16	any COMPLEX argument See Note 2	REAL *8 $0 \leqq y_1 \leqq \gamma$ $y_2 = 0$	—
Error function	ERF	No	$y=\frac{2}{\sqrt{\pi}}\int_0^x e^{-u^2}du$	1	REAL *4	any REAL argument	REAL *4 $-1 \leqq y \leqq 1$	—
	ERFC	No	$y=\frac{2}{\sqrt{\pi}}\int_x^\infty e^{-u^2}du$ $y=1-\text{erf}(x)$	1	REAL *4	any REAL argument	REAL *4 $0 \leqq y \leqq 2$	—
	DERF	No	$y=\frac{2}{\sqrt{\pi}}\int_0^x e^{-u^2}du$	1	REAL *8	any REAL argument	REAL *8 $-1 \leqq y \leqq 1$	—
	DERFC	No	$y=\frac{2}{\sqrt{\pi}}\int_x^\infty e^{-u^2}du$ $y=1-\text{erf}(x)$	1	REAL *8	any REAL argument	REAL *8 $0 \leqq y \leqq 2$	—
Gamma and log-gamma	GAMMA	No	$y=\int_0^\infty u^{x-1}e^{-u}du$	1	REAL *4	$x > 2^{-252}$ and $x < 57.5744$	REAL *4 $0.88560 \leqq y \leqq \gamma$	290
	ALGAMA	No	$y=\log_e \Gamma(x)$ or $y=\log_e \int_0^\infty u^{x-1}e^{-u}du$	1	REAL *4	$x > 0$ and $x < 4.2913 \cdot 10^{73}$	REAL *4 $-0.12149 \leqq y \leqq \gamma$	291
	DGAMMA	No	$y=\int_0^\infty u^{x-1}e^{-u}du$	1	REAL *8	$x > 2^{-252}$ and $x < 57.5744$	REAL *8 $0.88560 \leqq y \leqq \gamma$	300
	DLGAMA	No	$y=\log_e \Gamma(x)$ or $y=\log_e \int_0^\infty u^{x-1}e^{-u}du$	1	REAL *8	$x > 0$ and $x < 4.2913 \cdot 10^{73}$	REAL *8 $-0.12149 \leqq y \leqq \gamma$	301
Maximum and minimum values	MAX0	Yes	$y=\max(x_1,\ldots,x_n)$	$\geqq 2$	INTEGER *4	any INTEGER arguments	INTEGER *4	—
	MIN0	Yes	$y=\min(x_1,\ldots,x_n)$	$\geqq 2$	INTEGER *4	any INTEGER arguments	INTEGER *4	—
	AMAX0	Yes	$y=\max(x_1,\ldots,x_n)$	$\geqq 2$	INTEGER *4	any INTEGER arguments	REAL *4	—
	AMIN0	Yes	$y=\min(x_1,\ldots,x_n)$	$\geqq 2$	INTEGER *4	any INTEGER arguments	REAL *4	—
	MAX1	Yes	$y=\max(x_1,\ldots,x_n)$	$\geqq 2$	REAL *4	any REAL arguments	INTEGER *4	—

NOTES: (See end of table.)

Table 5. Miscellaneous Mathematical Subprograms (*Part 2 of 2*)

General Function	Entry Name	Subset	Definition	Argument(s) No.	Argument(s) Type[1]	Argument(s) Range	Function Value Type[2] and Range[6]	Error Code
Maximum and minimum values (continued)	MIN1	Yes	$y=\min(x_1,\ldots,x_n)$	$\geqq 2$	REAL *4	any REAL arguments	INTEGER *4	—
	AMAX1	Yes	$y=\max(x_1,\ldots,x_n)$	$\geqq 2$	REAL *4	any REAL arguments	REAL *4	—
	AMIN1	Yes	$y=\min(x_1,\ldots,x_n)$	$\geqq 2$	REAL *4	any REAL arguments	REAL *4	—
	DMAX1	Yes	$y=\max(x_1,\ldots,x_n)$	$\geqq 2$	REAL *8	any REAL arguments	REAL *8	—
	DMIN1	Yes	$y=\min(x_1,\ldots,x_n)$	$\geqq 2$	REAL *8	any REAL arguments	REAL *8	—
Modular arithmetic	MOD	See Note 3	$y=x_1$ (modulo x_2) See Note 4	2	INTEGER	$x_2 \neq 0$ See Note 5	INTEGER *4	—
	AMOD	See Note 3	$y=x_1$ (modulo x_2) See Note 4	2	REAL *4	$x_2 \neq 0$ See Note 5	REAL *4	—
	DMOD	See Note 3	$y=x_1$ (modulo x_2) See Note 4	2	REAL *8	$x_2 \neq 0$ See Note 5	REAL *8	—
Truncation	AINT	See Note 3	$y = (\text{sign } x) \cdot n$ where n is the largest integer $\leqq \lvert x \rvert$	1	REAL *4	any REAL argument	REAL *4	—
	INT	See Note 3	$y = (\text{sign } x) \cdot n$ where n is the largest integer $\leqq \lvert x \rvert$	1	REAL *4	any REAL argument	INTEGER *4	—
	IDINT	See Note 3	$y = (\text{sign } x) \cdot n$ where n is the largest integer $\leqq \lvert x \rvert$	1	REAL *8	any REAL argument	INTEGER *4	—

NOTES:

[1] REAL*4 and REAL*8 arguments correspond to REAL and DOUBLE PRECISION arguments, respectively, in Basic FORTRAN IV. Complex arguments cannot be used in a Basic FORTRAN IV program.

[2] Floating-point overflow can occur.

[3] The coding that performs this function is out-of-line in Basic FORTRAN IV (OS) and in-line in FORTRAN IV. Out-of-line coding is taken from the FORTRAN library by the linkage editor or loader and processed with the calling module. In-line coding is inserted by the FORTRAN compiler at the point in the source module where the function is referenced. This means that the in-line functions are invoked in FORTRAN IV by using the appropriate entry name, but that they are not part of the library. In-line functions are described in the FORTRAN IV language publications listed in the Preface.

[4] The expression x_1 (modulo x_2) is defined as $x_1 - \left[\frac{x_1}{x_2}\right] \cdot x_2$, where the brackets indicate that an integer is used. The largest integer whose magnitude does not exceed the magnitude of $\frac{x_1}{x_2}$ is used. The sign of the integer is the same as the sign of $\frac{x_1}{x_2}$.

[5] If $x_2 = 0$, then the modulus function is mathematically undefined. In addition, a divide exception is recognized and an interruption occurs. (A detailed description of the interruption procedure is given in Appendix C.)

[6] $\gamma = 16^{63}(1 - 16^{-6})$ for regular precision routines, and $16^{63}(1 - 16^{-14})$ for double precision routines.

Implicitly Called Subprograms

The implicitly called subprograms are executed as a result of certain notation appearing in a FORTRAN source statement. When a number is to be raised to a power or when multiplication or division of complex numbers is to be performed, the FORTRAN compiler generates the instructions necessary to call the appropriate subprogram. For example, if the following source statement appears in a source module,

```
ANS = BASE**EXPON
```

where BASE and EXPON are REAL*4 variables, the FORTRAN compiler generates a reference to FRXPR#, the entry name for a subprogram that raises a real number to a real power.

The implicitly called subprograms in the FORTRAN library are described in Table 6. The column headed "Implicit Function Reference" gives a representation of a source statement that might appear in a FORTRAN source module and cause the subprogram to be called. The rest of the column headings in Table 6 have the same meaning as those used with the explicitly called subprograms. Algorithms for implicitly called subprograms are given in the chapter "Algorithms." Additional information for assembler language programmers is given in Appendix A.

Table 6. Implicitly Called Mathematical Subprograms

General Function	Entry[1] Name	Sub-set	Implicit Function Reference[2]	Argument(s)		Function Value Type[3]	Error Code
				No.	Type[3]		
Multiply and divide complex numbers	CDMPY#	No	$y = z_1 * z_2$	2	COMPLEX *16	COMPLEX *16	—
	CDDVD#		$y = z_1/z_2$	2	COMPLEX *16	COMPLEX *16	—
	CMPY#	No	$y = z_1 * z_2$	2	COMPLEX *8	COMPLEX *8	—
	CDVD#		$y = z_1/z_2$	2	COMPLEX *8	COMPLEX *8	—
Raise an integer to an integer power	FIXPI#	Yes	y = i**j	2	i = INTEGER *4 j = INTEGER *4	INTEGER *4	241
Raise a real number to an integer power	FRXPI#	Yes	y = a**j	2	a = REAL *4 j = INTEGER *4	REAL *4	242
	FDXPI#	Yes	y = a**j	2	a = REAL *8 j = INTEGER *4	REAL *8	243
Raise a real number to a real power	FRXPR#	Yes	y = a**b	2	a = REAL *4 b = REAL *4	REAL *4	244
	FDXPD#	Yes	y = a**b	2	a = REAL *8 b = REAL 8*	REAL *8	245
Raise a complex number to an integer power	FCDXI#	No	y = z**j	2	z = COMPLEX *16 j = INTEGER *4	COMPLEX *16	247
	FCXPI#	No	y = z**j	2	z = COMPLEX *8 j = INTEGER *4	COMPLEX *8	246

NOTES:

1. This name must be used in an assembler language program to call the subprogram; the character # is a part of the name and must be included.
2. This is only a *representation* of a FORTRAN statement; it is not the only way the subprogram may be called.
3. REAL *4 and REAL *8 arguments correspond to REAL and DOUBLE PRECISION arguments, respectively, in Basic FORTRAN IV. Complex arguments cannot be used in a Basic FORTRAN IV program.

For subprograms that involve exponentiation, the action taken within a subprogram depends upon the types of the base and exponent used. Tables 7 through 10 show the result of an exponentiation performed with the different combinations and values of base and exponent. In these tables, *I* and *J* are integers; *A* and *B* are real numbers; *C* is a complex number.

Table 7. Exponentiation with Integer Base and Exponent

Base (I)	Exponent (J)		
	J > 0	J = 0	J < 0
I > 1	Compute the function value	Function value = 1	Function value = 0
I = 1	Compute the function value	Function value = 1	Function value = 1
I = 0	Function value = 0	Error message 241	Error message 241
I = −1	Compute the function value	Function value = 1	If J is an odd number, function value = −1. If J is an even number, function value = 1.
I < −1	Compute the function value	Function value = 1	Function value = 0

Table 8. Exponentiation with Real Base and Integer Exponent

Base (A)	Exponent (J)		
	J > 0	J = 0	J < 0
A > 0	Compute the function value	Function value = 1	Compute the function value
A = 0	Function value = 0	Error message 242 or 243	Error message 242 or 243
A < 0	Compute the function value	Function value = 1	Compute the function value

Table 9. Exponentiation with Real Base and Exponent

Base (A)	Exponent (B)		
	B > 0	B = 0	B < 0
A > 0	Compute the function value	Function value = 1	Compute the function value
A = 0	Function value = 0	Error message 244 or 245	Error message 244 or 245
A < 0	Error message 253 or 263	Function value = 1	Error message 253 or 263

Table 10. Exponentiation with Complex Base and Integer Exponent

Base (C) C = P + Qi	Exponent (J)		
	J > 0	J = 0	J < 0
P > 0 and Q > 0	Compute the function value	Function value = 1 + 0*i*	Compute the function value
P > 0 and Q = 0	Compute the function value	Function value = 1 + 0*i*	Compute the function value
P > 0 and Q < 0	Compute the function value	Function value = 1 + 0*i*	Compute the function value
P = 0 and Q > 0	Compute the function value	Function value = 1 + 0*i*	Compute the function value
P = 0 and Q = 0	Function value 0 + 0*i*	Error message 246 or 247	Error message 246 or 247
P = 0 and Q < 0	Compute the function value	Function value = 1 + 0*i*	Compute the function value
P < 0 and Q > 0	Compute the function value	Function value = 1 + 0*i*	Compute the function value
P < 0 and Q = 0	Compute the function value	Function value = 1 + 0*i*	Compute the function value
P < 0 and Q < 0	Compute the function value	Function value = 1 + 0*i*	Compute the function value

The service subprograms supplied in the FORTRAN library are divided into two groups: one group tests machine indicators and the other group performs utility functions. Service subprograms are called by using the appropriate entry name in a FORTRAN language CALL statement.

Machine Indicator Test Subprograms

The machine indicator subprograms test the status of pseudo indicators and may return a value to the calling program. When the indicator is zero, it is off; when the indicator is other than zero, it is on. In the following descriptions of the subprograms, *i* represents an integer expression and *j* represents an integer variable.

Pseudo Sense Light Subprogram
Entry Names: SLITE/SLITET

This subprogram is used to alter, test, and/or record the status of pseudo sense lights. Either of two entry names (SLITE or SLITET) is used to call the subprogram. The particular entry name used in the CALL statement depends upon the operation to be performed.

If the *four* sense lights are to be turned off or *one* sense light is to be turned on, entry name SLITE is used. The source language statement is:

CALL SLITE (*i*)

where *i* has a value of 0, 1, 2, 3, or 4.

If the value of *i* is 0, the four sense lights are turned off; if the value of *i* is 1, 2, 3, or 4, the corresponding sense light is turned on. If the value of *i* is not 0, 1, 2, 3, or 4, error message 216 is issued and execution of this module or phase is terminated. (This error handling is explained in Appendix C.) Execution can continue, however, if the extended error handling facility was selected at system generation (FORTRAN IV (OS) only). This facility is explained in detail in the *FORTRAN IV (G and H) Programmer's Guide* listed in the Preface.

If a sense light is to be tested and its status recorded, entry name SLITET is used. Regardless of its status before the test, after a sense light is tested, it is always set off. The source language statement is:

CALL SLITET (*i,j*)

where:

i has a value of 1, 2, 3, or 4, and indicates which sense light to test.

j has a value returned by the subprogram. 1 indicates the sense light was on; 2 indicates the sense light was off.

If the value of *i* is not 1, 2, 3, or 4, error message 216 is issued and execution of this module or phase is terminated unless the extended error handling facility is in effect.

Overflow Indicator Subprogram
Entry Name: OVERFL

This subprogram tests for an exponent overflow or underflow exception and returns a value that indicates the existing condition. After testing, the overflow indicator is turned off. This subprogram is called by using the entry name OVERFL in a CALL statement. The source language statement is:

CALL OVERFL (*j*)

The value of *j* is returned by the subprogram to indicate the following:

1 = floating-point overflow condition has occurred last.
2 = no overflow or underflow condition has occurred.
3 = a floating-point underflow condition has occurred last.

Note: A value for *j* of 1 or 3 indicates that that condition was the last one to occur. An overflow followed by an underflow in the same statement would be recorded as condition 3 – "underflow occurred last."

A detailed description of each exception is given in the programmer's guides listed in the Preface.

Divide Check Subprogram
Entry Name: DVCHK

This subprogram tests for a divide-check exception and returns a value that indicates the existing condition. After testing, the divide-check indicator is turned off. This subprogram is called by using entry name DVCHK in a CALL statement. The source language statement is:

CALL DVCHK (*j*)

where:

j is set to 1 if the divide-check indicator was on; or to 2 if the indicator was off.

Utility Subprograms

The utility subprograms perform two operations for the FORTRAN programmer: they either terminate execution (EXIT) or dump a specified area of storage (DUMP/PDUMP).

End Execution Subprogram
Entry Name: EXIT

The end execution subprogram terminates execution of the load module or phase and returns control to the operating system. (Except that no operator message is produced, EXIT performs a function similar to that performed by the STOP statement.) This subprogram is called by using the entry name EXIT in a CALL statement. The source language statement is:

```
CALL EXIT
```

Storage Dump Subprogram
Entry Names: DUMP/PDUMP

This subprogram dumps a specified area of storage. Either of two entry names (DUMP or PDUMP) can be used to call the subprogram. The entry name is followed by the limits of the area to be dumped and the format specification. The entry name used in the CALL statement depends upon the nature of the dump to be taken.

If execution of the load module or phase is to be terminated after the dump is taken, entry name DUMP is used. The source language statement is:
where:

CALL DUMP ($a_1, b_1, f_1, \ldots, a_n, b_n, f_n$)

a and *b* are variables that indicate the limits of storage to be dumped (either *a* or *b* may represent the upper or lower limits of storage).

f indicates the dump format and may be one of the integers given in Table 11. The formats available depend upon the compiler in use. A sample printout for each format is given in Appendix E.

Table 11. DUMP/PDUMP Format Specifications

Basic FORTRAN IV	FORTRAN IV
0 specifies hexadecimal	0 specifies hexadecimal
4 specifies INTEGER	1 specifies LOGICAL *1
5 specifies REAL	2 specifies LOGICAL *4
6 specifies DOUBLE PRECISION	3 specifies INTEGER *2
	4 specifies INTEGER *4
	5 specifies REAL *4
	6 specifies REAL *8
	7 specifies COMPLEX *8
	8 specifies COMPLEX *16
	9 specifies literal

If execution is to be resumed after the dump is taken, entry name PDUMP is used. The source language statement is:

CALL PDUMP ($a_1, b_1, f_1, \ldots, a_n, b_n, f_n$)

where *a*, *b*, and *f* have the same meaning as for DUMP.

Programming Considerations

A load module or phase may occupy a different area of storage each time it is executed. To ensure that the appropriate areas of storage are dumped, the following conventions should be observed.

If an array and a variable are to be dumped at the same time, a separate set of arguments should be used for the array and for the variable. The specification of limits for the array should be from the first element in the array to the last element. For example, assume that A is a variable in COMMON, B is a REAL number, and TABLE is an array of 20 elements. The following call to the storage dump subprogram could be used to dump TABLE and B in the hexadecimal format and terminate execution after the dump is taken:

CALL PDUMP ($a_1, b_1, f_1, \ldots, a_n, b_n, f_n$)

If an area of storage in COMMON is to be dumped at the same time as an area of storage not in COMMON, the arguments for the area in COMMON should be given separately. For example, the following call to the storage dump subprogram could be used to dump the variables A and B in REAL*8 format without terminating execution:

```
CALL PDUMP (A,A,6,B,B,6)
```

If variables not in COMMON are to be dumped, each variable must be listed separately in the argument list. For example, if R, P, and Q are defined implicitly in the program, the statement

```
CALL PDUMP (R,R,5,P,P,5,Q,Q,5)
```

should be used to dump the three variables. If the statement

```
CALL PDUMP (R,Q,5)
```

is used, all main storage between R and Q is dumped, which may or may not include P, and may include other variables.

If an array and a variable are passed to a subroutine as arguments, the arguments in the call to the storage

dump subprogram in the subroutine should specify the parameters used in the definition of the subroutine. For example, if the subroutine SUBI is defined as:

```
SUBROUTINE SUBI (X,Y)
DIMENSION X (10)
```

and the call to SUBI within the source module is:

```
DIMENSION A (10)
•
•
•
CALL SUBI (A, B)
```

then the following statement should be used in SUBI to dump the variables in hexadecimal format without terminating execution:

```
CALL PDUMP (X(1), X(10), 0, Y, Y, 0)
```

If the statement

```
CALL PDUMP (X(1), Y, 0)
```

is used, all storage between A(1) and Y is dumped because of the method of transmitting arguments.

When hexadecimal (0) or literal (9) is specified, the programmer should realize that the upper limit is assumed to be of length 4.

Algorithms

This chapter contains information about the method by which each mathematical function is computed. The information for explicitly called subprograms is arranged alphabetically according to the specific function of each subprogram (i.e., absolute value, exponentiation, logarithmic, etc.). The individual entry names associated with each subprogram are arranged logically from simple to complex within each function. For example, the heading "Square Root Subprograms" will have algorithms arranged in the following order by entry name: SQRT, DSQRT, CSQRT, CDSQRT.

Information for the implicity called subprograms is arranged alphabetically according to function, and alphabetically by entry name within that function. For example, the heading "Complex Multiply and Divide Subprograms" will have algorithms arranged in the following order: CDDVD#/CDMPY#, CDVD#/CMPY#.

The information for each subprogram is divided into two parts. The first part describes the algorithm used; the second part describes the effect of an argument error upon the accuracy of the answer returned.

The presentation of each algorithm is divided into its major computational steps; the formulas necessary for each step are supplied. For the sake of brevity, the needed constants are normally given only symbolically. (The actual values can be found in the assembly listing of the subprograms.) Some of the formulas are widely known; those that are not so widely known are derived from more common formulas. The process leading from the common formula to the computational formula is sketched in enough detail so that the derivation can be reconstructed by anyone who has an understanding of college mathematics and access to the common texts on numerical analysis.[1] Many approximations were derived by the so-called "minimax" methods. The approximation sought by these methods can be characterized as follows. Given a function $f(x)$, an interval I, the form of the approximation (such as the rational form with specified degrees), and the type of error to be minimized (such as the relative error), there is normally a unique approximation to $f(x)$ whose maximum error over I is the smallest among all possible approximations of the given form. Details of the theory and the various methods of deriving such approximation are provided in the reference.[1] The accuracy figures cited in the algorithm sections are theoretical, and they do not take round-off errors into account. Minor programming techniques used to minimize round-off errors are not necessarily described here.

The accuracy of an answer produced by these algorithms is influenced by two factors: the performance of the subprogram (see the chapter, "Performance Statistics") and the accuracy of the argument. The effect of an argument error upon the accuracy of an answer depends solely upon the mathematical function involved and not upon the particular coding used in the subprogram.

A guide to the propagational effect of argument errors is provided because argument errors always influence the accuracy of answers whether the errors are accumulated prior to use of the subprogram or introduced by newly converted data. This guide (expressed as a simple formula where possible) is intended to assist users in assessing the effect of an argument error.

---

[1] Any of modern numerical analysis texts may be used as a reference. One such text is A. Ralston's *A First Course in Numerical Analysis* (McGraw-Hill Book Company, Inc., New York, 1965). Background information for algorithms that use continued fractions may be found in H. S. Wall's *Analytic Theory of Continued Fractions* (D. Van Nostrand Co. Inc., Princeton, N. J., 1948).

The following symbols are used in this chapter to describe the effect of an argument error upon the accuracy of the answer:

SYMBOL	EXPLANATION
$g(x)$	The result given by the subprogram.
$f(x)$	The correct result.
ϵ	$\left\|\frac{f(x) - g(x)}{f(x)}\right\|$ The relative error of the result given by the subprogram.
δ	The relative error of the argument.
E	$\|f(x) - g(x)\|$ The absolute error of the result given by the subprogram.
Δ	The absolute error of the argument.

The notation used for the continued fractions complies with the specifications set by the National Bureau of Standards.[2]

Although it is not specifically stated below for each subroutine, the algorithms in this chapter were programmed to conform to the following standards governing floating-point overflow/underflow.

1. Intermediate underflow and overflows are not permitted to occur. This prevents the printing of irrelevant messages.
2. Those arguments for which the answer can overflow are excluded from the permitted range of the subroutine. This rule does not apply to CDABS and CABS.
3. When the magnitude of the answer is less than 16^{-65}, zero is given as the answer. If the floating-point underflow exception mask is on at the time, the underflow message will be printed.

Control of Program Exceptions in Mathematical Functions

The FORTRAN mathematical functions have been coded with careful control of error situations. A result is provided whenever the answer is within the range representable in the floating-point form. In order to be consistent with FORTRAN control of exponent overflow/underflow exceptions, the following types of conditions are recognized and handled separately.

When the magnitude of the function value is too large to be represented in the floating-point form, the condition is called a terminal overflow; when the magnitude is too small to be represented, a terminal underflow. On the other hand, if the function value is representable, but if execution of the chosen algorithm causes an overflow or underflow in the process, this condition is called an intermediate overflow or underflow.

Function subroutines in the FORTRAN library have been coded to observe the following rules for these conditions:

1. Algorithms which can cause an intermediate overflow have been avoided. Therefore an intermediate overflow should not occur during the execution of a function subroutine of the library.
2. Intermediate underflows are detected and not allowed to cause an interrupt. In other words, spurious underflow signals are not allowed to be given. Computation of the function value is successfully carried out.
3. Terminal overflow conditions are screened out by the subroutine. The argument is considered out of range for computation and an error diagnostic is given.

[2] For more information, see Milton Abramowitz and Irene A. Stegun (editors), *Handbook of Mathematical Functions*, Applied Mathematics Series-55 (National Bureau of Standards, Washington, D.C., 1965).

4. Terminal underflow conditions are handled by forcing a floating-point underflow exception. This provides for the detection of underflow in the same manner as for an arithmetic statement. Terminal underflows can occur in the following function subroutines: EXP, DEXP, ATAN2, DATAN2, ERFC, and DERFC.

For implicit arithmetic subroutines, these rules do not apply. In this case, both terminal overflows and terminal underflows will cause respective floating-point exceptions. In addition, in case of complex arithmetic (implicit multiply and divide), premature overflow/underflow is possible when the result of arithmetic is very close to an overflow or underflow condition.

Explicitly Called Subprograms

Absolute Value Subprograms

CABS/CDABS

1. Write $|x + iy| = a + ib$.
2. Let $v_1 = \max(|x|, |y|)$, and $v_2 = \min(|x|, |y|)$.
3. If characteristics of v_1 and v_2 differ by 7 (15 for CDABS) or more, or if $v_2 = 0$, then $a = v_1$, $b = 0$.
4. Otherwise,

$$a = 2 \cdot v_1 \cdot \sqrt{1/4 + 1/4\left(\frac{v_2}{v_1}\right)^2}, \text{ and } b = 0.$$

If the answer is greater than 16^{63}, the floating-point overflow interruption will take place (see Appendix C). The algorithms for both complex absolute value subprograms are identical. Each subprogram uses the appropriate real square root subprogram (SQRT or DSQRT).

Arcsine and Arccosine Subprograms

ARSIN/ARCOS

Algorithm

1. If $0 \leqq x \leqq 1/2$, then compute arcsin (x) by a continued fraction of the form:

$$\arcsin(x) \cong x + x^3 \cdot F \text{ where}$$

$$F = \frac{d_1}{(x^2 + c_1) + \dfrac{d_2}{(x^2 + c_2) + \cdots}}.$$

The coefficients of this formula were derived by transforming the minimax rational approximation (in relative error, over the range $0 \leqq x^2 \leqq 1/4$) for $\arcsin(x)/x$ of the following form:

$$\frac{\arcsin(x)}{x} \cong a_0 + x^2 \cdot \left[\frac{a_1 + a_2 x^2}{b_0 + b_1 x^2 + x^4}\right].$$

Minimax was taken under the constraint that $a_0 = 1$ exactly. The relative error of this approximation is less than $2^{-28.3}$.

If $0 \leqq x \leqq 1/2$, $\arccos(x)$ is computed as:

$$\arccos(x) = \frac{\pi}{2} - \arcsin(x).$$

2. If $1/2 < x \leqq 1$, then compute $\arccos(x)$ essentially as:

$$\arccos(x) = 2 \cdot \arcsin\left(\sqrt{\frac{1 - x}{2}}\right).$$

This case is now reduced to the first case because within these limits,

$$0 \leqq \sqrt{\frac{1 - x}{2}} \leqq 1/2.$$

This computation uses the real square root subprogram (SQRT)

If $1/2 < x \leqq 1$, $\arcsin(x)$ is computed as:

$$\arcsin(x) = \frac{\pi}{2} - \arccos(x).$$

Implementation of the above algorithms (steps 1 and 2) were carried out with care to minimize the round-off errors.

3. If $-1 \leqq x < 0$, then $\arcsin(x) = -\arcsin|x|$
and $\arccos(x) = \pi - \arccos|x|$.

This reduces these cases to one of the two positive cases.

Effect of an Argument Error

$E \sim \frac{\Delta}{\sqrt{1-x^2}}$. For small values of x, $E \sim \Delta$. Toward the limits (± 1) of the range, a small Δ causes a substantial error in the answer. For the arcsine, $\epsilon \sim \delta$ if the value of x is small.

DARSIN/DARCOS

Algorithm

1. If $0 \leqq x \leqq \frac{1}{2}$, then compute $\arcsin(x)$ by a continued fraction of the form:

$$\arcsin(x) \cong x + x^3 \cdot F \text{ where}$$

$$F = c_1 + \frac{d_1}{(x^2 + c_2) + \dfrac{d_2}{(x^2 + c_3) + \dfrac{d_3}{(x^2 + c_4) + \dfrac{d_4}{(x^2 + c_5)}}}}.$$

The relative error of this approximation is less than $2^{-57.2}$.
The coefficients of this formula were derived by transforming the minimax rational approximation (in relative error, over the range $0 \leqq x^2 \leqq \frac{1}{4}$) for $\arcsin(x)/x$ of the following form:

$$\frac{\arcsin(x)}{x} \cong a_0 + x^2\left[\frac{a_1 + a_2x^2 + a_3x^4 + a_4x^6 + a_5x^8}{b_0 + b_1x^2 + b_2x^4 + b_3x^6 + x^8}\right].$$

Minimax was taken under the constraint that $a_0 = 1$ exactly.
If $0 \leqq x \leqq \frac{1}{2}$, $\arccos(x)$ is computed as:

$$\arccos(x) = \frac{\pi}{2} - \arcsin(x).$$

2. If $\frac{1}{2} < x \leqq 1$, then compute $\arccos(x)$ essentially as:

$$\arccos(x) = 2 \cdot \arcsin\left(\sqrt{\frac{1-x}{2}}\right).$$

This case is now reduced to the first case because within these limits,

$$0 \leqq \sqrt{\frac{1-x}{2}} \leqq \frac{1}{2}.$$

This computation uses the real square root subprogram (DSQRT).
If $\frac{1}{2} < x \leqq 1$, $\arcsin(x)$ is computed as:

$$\arcsin(x) = \frac{\pi}{2} - \arccos(x).$$

Implementation of the above algorithms (steps 1 and 2) were carried out with care to minimize the round-off errors.

3. If $-1 \leqq x < 0$, then $\arcsin(x) = -\arcsin|x|$, and $\arccos(x) = \pi - \arccos|x|$. This reduces these cases to one of the two positive cases.

Effect of an Argument Error

$E \sim \frac{\Delta}{\sqrt{1-x^2}}$. For small values of x, $E \sim \Delta$. Toward the limits (± 1) of the range a small Δ causes a substantial error in the answer. For the arcsine, $\epsilon \sim \delta$ if the value of x is small.

Arctangent Subprograms

ATAN — Basic FORTRAN IV (OS)

Algorithm

1. Reduce the computation of arctan (x) to the case $0 \leqq x \leqq 1$, by using

   $\arctan(-x) = -\arctan(x)$, or

   $$\arctan\left(\frac{1}{|x|}\right) = \frac{\pi}{2} - \arctan|x|.$$

2. If necessary, reduce the computation further to the case $|x| \leqq \tan 15°$ by using

   $$\arctan(x) = 30° + \arctan\left(\frac{\sqrt{3} \cdot x - 1}{x + \sqrt{3}}\right).$$

   The value of $\left|\frac{\sqrt{3} \cdot x - 1}{x + \sqrt{3}}\right| \leqq \tan 15°$ if the value of x is within the range, $\tan 15° < x \leqq 1$. The value of $(\sqrt{3} \cdot x - 1)$ is computed as $(\sqrt{3} - 1)\,x - 1 + x$ to avoid the loss of significant digits.
3. For $|x| \leqq \tan 15°$, use the approximation formula:

   $$\frac{\arctan(x)}{x} \cong 0.60310579 - 0.05160454x^2 + \frac{0.55913709}{x^2 + 1.4087812}.$$

   This formula has a relative error less than $2^{-27.1}$ and can be obtained by transforming the continued fraction

   $$\frac{\arctan(x)}{x} = 1 - \frac{x^2}{3 + \dfrac{\dfrac{x^2}{5}}{\left(\dfrac{5}{7} + x^{-2}\right) - w}}$$

   where w has an approximate value of $\left(-\frac{75}{77}x^{-2} + \frac{3375}{77}\right) 10^{-4}$, but the true value of w is $\dfrac{\frac{4 \cdot 5}{7 \cdot 7 \cdot 9}}{\left(\frac{43}{7 \cdot 11} + x^{-2}\right) + \cdots}$.

   The original continued fraction can be obtained by transforming the Taylor series into continued fraction form.

Effect of an Argument Error

$E \sim \frac{\Delta}{1 + x^2}$. For small values of x, $\epsilon \sim \delta$; as the value of x increases, the effect of δ upon ϵ diminishes.

ATAN/ATAN2 — FORTRAN IV (OS)

Algorithm

1. For arctan (x_1, x_2):
   If $x_1 < 0$, use the identity $\arctan(x_1, x_2) = -\arctan(-x_1, x_2)$.
   Hence we may assume that $x_1 \geqq 0$. Then:

   If either $x_2 = 0$ or $\left|\frac{x_1}{x_2}\right| > 2^{24}$, the answer $= \frac{\pi}{2}$.

   If $x_2 < 0$ and $\left|\frac{x_1}{x_2}\right| < 2^{-24}$, the answer $= \pi$.

For the general case, if $x_2 > 0$, the answer = $\arctan\left(\left|\frac{x_1}{x_2}\right|\right)$, and

if $x_2 < 0$, the answer = $\pi - \arctan\left(\left|\frac{x_1}{x_2}\right|\right)$.

2. The computation of $\arctan\left(\left|\frac{x_1}{x_2}\right|\right)$ above, or of $\arctan(x)$ for the single argument case, follows the algorithm given for the subprogram ATAN in Basic FORTRAN IV (OS).

Effect of an Argument Error

$E \sim \frac{\Delta}{1+x^2}$. For small values of x, $\epsilon \sim \delta$; as the value of x increases, the effect of δ upon ϵ diminishes.

DATAN — Basic FORTRAN IV (OS)

Algorithm

1. Reduce the computation of arctan (x) to the case $0 \leqq x \leqq 1$ by using

   $\arctan(-x) = -\arctan(x)$ and

   $\arctan\frac{1}{|x|} = \frac{\pi}{2} - \arctan|x|$.

2. If necessary, reduce the computation further to the case $|x| \leqq \tan 15°$ by using

   $\arctan(x) = 30° + \arctan\left(\frac{\sqrt{3}\cdot x - 1}{x + \sqrt{3}}\right)$.

   The value of $\left|\frac{\sqrt{3}\cdot x - 1}{x + \sqrt{3}}\right| \leqq \tan 15°$, if the value of x is within the range $\tan 15° < x \leqq 1$. The value of $(\sqrt{3}\cdot x - 1)$ is computed as $(\sqrt{3} - 1)\,x - 1 + x$ to avoid the loss of significant digits.
3. For $|x| \leqq \tan 15°$, use a continued fraction of the form:

   $$\frac{\arctan(x)}{x} \cong 1 + x^2\left[b_0 - \frac{a_1}{(b_1 + x^2) -} \frac{a_2}{(b_2 + x^2) -} \frac{a_3}{(b_3 + x^2)}\right].$$

   The relative error of this approximation is less than $2^{-60.7}$.
   The coefficients of this formula were derived by transforming a minimax rational approximation (in relative error, over the range $0 \leqq x^2 \leqq 0.071797$) for $\arctan(x)/x$ of the following form:

   $$\frac{\arctan(x)}{x} \cong a_0 + x^2\left[\frac{c_0 + c_1x^2 + c_2x^4 + c_3x^6}{d_0 + d_1x^2 + d_2x^4 + x^6}\right].$$

   Minimax was taken under the constraint that $a_0 = 1$ exactly.

Effect of an Argument Error

$E \sim \frac{\Delta}{1+x^2}$. For small values of x, $\epsilon \sim \delta$, and as the value of x increases, the effect of ϵ upon δ diminishes.

DATAN/DATAN2 — FORTRAN IV (OS)

Algorithm

1. For $\arctan(x_1, x_2)$:
   If $x_1 < 0$, use the identity $\arctan(x_1, x_2) = -\arctan(-x_1, x_2)$.
   Hence we may assume that $x_1 \geqq 0$. Then:

   If either $x_2 = 0$ or $\left|\frac{x_1}{x_2}\right| > 2^{56}$, the answer $= \frac{\pi}{2}$.

If $x_2 < 0$ and $\left|\frac{x_1}{x_2}\right| < 2^{-56}$, the answer $= \pi$.

For the general case, if $x_2 > 0$, the answer $= \arctan\left(\left|\frac{x_1}{x_2}\right|\right)$, and

if $x_2 < 0$, the answer $= \pi - \arctan\left(\left|\frac{x_1}{x_2}\right|\right)$.

2. The computation of $\arctan\left(\left|\frac{x_1}{x_2}\right|\right)$ above, or of $\arctan(x)$ for the single argument case, follows the algorithm given for the subprogram DATAN in Basic FORTRAN IV (OS).

Effect of an Argument Error

$E \sim \frac{\Delta}{1+x^2}$. For small values of x, $\epsilon \sim \delta$, and as the value of x increases, the effect of ϵ upon δ diminishes.

Error Functions Subprograms

ERF/ERFC

Algorithm

1. If $0 \leqq x \leqq 1$, then compute the error function by the following approximation:

$$\text{erf}(x) \cong x(a_0 + a_1x^2 + a_2x^4 + \ldots + a_5x^{10}).$$

The coefficients were obtained by the minimax approximation (in relative error) of $\text{erf}(x)/x$ as a function of x^2 over the range $0 \leqq x^2 \leqq 1$. The relative error of this approximation is less than $2^{-24.6}$. The value of the complemented error function is computed as $\text{erfc}(x) = 1 - \text{erf}(x)$.

2. If $1 < x < 2.040452$, then compute the complemented error function by the following approximation:

$$\text{erfc}\,(x) \cong b_0 + b_1z + b_2z^2 + \ldots + b_9z^9$$

where $z = x - T_0$ and $T_0 \cong 1.709472$. The coefficients were obtained by the minimax approximation (in absolute error) of the function $f(z) = \text{erfc}(z + T_0)$ over the range $-0.709472 \leqq z \leqq 0.33098$. The absolute error of this approximation is less than $2^{-31.5}$. The limits of this range and the value of the origin T_0 were chosen to minimize the hexadecimal round-off errors. The value of the complemented error function within this range is between $\frac{1}{256}$ and 0.1573. The value of the error function is computed as $\text{erf}(x) = 1 - \text{erfc}(x)$.

3. If $2.040452 \leqq x < 13.306$, then compute the complemented error function by the following approximation:

$$\text{erfc}(x) \cong e^{-z} \cdot F/x \text{ where } z = x^2 \text{ and}$$

$$F = c_0 + \frac{c_1 + c_2z + c_3z^2}{d_1z + d_2z^2 + z^3}.$$

The coefficients for F were obtained by transforming a minimax rational approximation (in absolute errors, over the range $13.306^{-2} \leqq w \leqq 2.040452^{-2}$) of the function $f(w) = \text{erfc}(x) \cdot x \cdot e^{x^2}$, $w = x^{-2}$, of the following form:

$$f(w) \cong \frac{a_0 + a_1w + a_2w^2 + a_3w^3}{b_0 + b_1w + w^2}.$$

The absolute error of this approximation is less than $2^{-26.1}$. This computation uses the real exponential subprogram (EXP).

If $2.040452 \leqq x < 3.919206$, then the error function is computed as $\text{erf}(x) = 1 - \text{erfc}(x)$.

If $3.919206 \leqq x$, then the error function is $\cong 1$.

4. If $13.306 \leqq x$, then the error function is $\cong 1$, and the complemented error function is $\cong 0$ (underflow).
5. If $x < 0$, then reduce to a case involving a positive argument by the use of the following formulas:

$$\text{erf}(-x) = -\text{erf}(x), \text{ and } \text{erfc}(-x) = 2 - \text{erfc}(x).$$

Effect of an Argument Error

$E \sim e^{-x^2} \cdot \Delta$. For the error function, as the magnitude of the argument exceeds 1, the effect of an argument error upon the final accuracy diminishes rapidly. For small values of x, $\epsilon \sim \delta$. For the complemented error function, if the value of x is greater than 1, $\text{erfc}(x) \sim \frac{e^{-x^2}}{2x}$. Therefore, $\epsilon \sim 2x^2 \cdot \delta$. If the value of x is negative or less than 1, then $\epsilon \sim e^{-x^2} \cdot \Delta$.

DERF/DERFC

Algorithm

1. If $0 \leqq x < 1$, then compute the error function by the following approximation:

$$\text{erf}(x) \cong x(a_0 + a_1x^2 + a_2x^4 + \ldots + a_{11}x^{22}).$$

The coefficients were obtained by the minimax approximation (in relative error) of $\text{erf}(x)/x$ as a function of x^2 over the range $0 \leqq x^2 \leqq 1$. The relative error of this approximation is less than $2^{-56.9}$. The value of the complemented error function is computed as $\text{erfc}(x) = 1 - \text{erf}(x)$.

2. If $1 \leqq x < 2.040452$, then compute the complemented error function by the following approximation:

$$\text{erfc}(x) \cong b_0 + b_1z + b_2z^2 + \ldots b_{18}z^{18}$$

where $z = x - T_0$ and $T_0 \cong 1.709472$. The coefficients were obtained by the minimax approximation (in absolute error) of the function $f(z) = \text{erfc}(z + T_0)$ over the range $-0.709472 \leqq z \leqq 0.33098$. The absolute error of this approximation is less than $2^{-60.3}$. The limits of this range and the value of the origin T_0 were chosen to minimize the hexadecimal round-off errors. The value of the complemented error function within this range is between $\frac{1}{256}$ and 0.1573. The value of the error function is computed as $\text{erf}(x) = 1 - \text{erfc}(x)$.

3. If $2.040452 \leqq x < 13.306$, then compute the complemented error function by the following approximation:

$$\text{erfc}(x) \cong e^{-z} \cdot F/x \text{ where } z = x^2 \text{ and}$$

$$F = c_0 + \frac{d_1}{(z + c_1) +} \frac{d_2}{(z + c_2) +} \cdots \frac{d_6}{(z + c_6) +} \frac{d_7}{(z + c_7)}.$$

The coefficients for F were derived by transforming a minimax rational approximation (in absolute errors, over the range $13.306^{-2} \leqq w \leqq 2.040452^{-2}$) of the function $f(w) = \text{erfc}(x) \cdot x \cdot e^{x^2}$, $w = x^{-2}$, of the following form:

$$f(w) \cong \frac{a_0 + a_1w + a_2w^2 + \ldots + a_7w^7}{b_0 + b_1w + b_2w^2 + \ldots + b_6w^6 + w^7}.$$

The absolute error of this approximation is less than $2^{-57.9}$. This computation uses the real exponential subprogram (DEXP). If $2.040452 \leqq x < 6.092368$, then the error function is computed as $\text{erf}(x) = 1 - \text{erfc}(x)$.

If $6.092368 \leqq x$, then the error function is $\cong 1$.

4. If $13.306 \leqq x$, then the error function is $\cong 1$, and the complemented error function $\cong 0$ (underflow).
5. If $x < 0$, then reduce to a case involving a positive argument by the use of the following formulas:

$$\text{erf}(-x) = -\text{erf}(x), \text{ and } \text{erfc}(-x) = 2 - \text{erfc}(x).$$

Effect of an Argument Error

$E \sim e^{-x^2} \cdot \Delta$. For the error function, as the magnitude of the argument exceeds 1, the effect of an argument error upon the final accuracy diminishes rapidly. For small values of x, $\epsilon \sim \delta$. For the complemented error function, if the value of x is greater than 1, $\text{erfc}(x) \sim \frac{e^{-x^2}}{2x}$. Therefore, $\epsilon \sim 2x^2 \cdot \delta$. If the value of x is negative or less than 1, then $\epsilon \sim e^{-x^2} \cdot \Delta$.

Exponential Subprograms

EXP

Algorithm

1. If $x < -180.218$, then 0 is given as the answer via floating-point underflow.
2. Otherwise, divide x by $\log_e 2$ and write

$$y = \frac{x}{\log_e 2} = 4a - b - d$$

where a and b are integers, $0 \leqq b \leqq 3$ and $0 \leqq d < 1$.

3. Compute 2^{-d} by the following fractional approximation:

$$2^{-d} \cong 1 - \frac{2d}{0.034657359\,d^2 + d + 9.9545948 - \dfrac{617.97227}{d^2 + 87.417497}}.$$

This formula can be obtained by transforming the Gaussian continued fraction

$$e^{-z} = 1 - \frac{z}{1+}\,\frac{z}{2-}\,\frac{z}{3+}\,\frac{z}{2-}\,\frac{z}{5+}\,\frac{z}{2-}\,\frac{z}{7+}\,\frac{z}{2}.$$

The maximum relative error of this approximation is 2^{-29}.

4. Multiply 2^{-d} by 2^{-b}.
5. Finally, add the hexadecimal exponent a to the characteristic of the answer.

Effect of an Argument Error

$\epsilon \sim \Delta$. If the magnitude of x is large, even the round-off error of the argument causes a substantial relative error in the answer because $\Delta = \delta \cdot x$.

DEXP

Algorithm

1. If $x < -180.2187$, then 0 is given as the answer via floating-point underflow.
2. Divide x by $\log_e 2$ and write

$$x = \left(4a - b - \frac{c}{16}\right) \cdot \log_e 2 - r$$

where a, b, and c are integers, $0 \leqq b \leqq 3$, $0 \leqq c \leqq 15$, and the remainder r is within the range $0 \leqq r < \frac{1}{16} \cdot \log_e 2$. This reduction is carried out in an extra precision to ensure accuracy. Then $e^x = 16^a \cdot 2^{-b} \cdot 2^{-c/16} \cdot e^{-r}$.

3. Compute e^{-r} by using a minimax polynomial approximation of degree 6 over the range $0 \leqq r < \frac{1}{16} \cdot \log_e 2$. In obtaining coefficients of this approximation, the minimax of relative errors was taken under the constraint that the constant term a_0 shall be exactly 1. The relative error is less than $2^{-56.87}$.
4. Multiply e^{-r} by $2^{-c/16}$. The 16 values of $2^{-c/16}$ for $0 \leqq c \leqq 15$ are included in the subprogram. Then halve the result b times.
5. Finally, add the hexdecimal exponent of a to the characteristic of the answer.

Effect of an Argument Error

$E \sim \Delta$. If the magnitude of x is large, even the round-off error of the argument causes a substantial relative error in the answer because $\Delta = \delta \cdot x$.

CEXP/CDEXP

Algorithm

The value of e^{x+iy} is computed as $e^x \cdot \cos(y) + i \cdot e^x \cdot \sin(y)$. The algorithms for both complex exponential subprograms are identical. Each subprogram uses the appropriate real exponential subprogram (EXP or DEXP) and the appropriate real sine/cosine subprogram (COS/SIN or DCOS/DSIN).

Effect of an Argument Error

The effect of an argument error depends upon the accuracy of the individual parts of the argument. If $e^{x+iy} = R \cdot e^{iH}$, then $H = y$ and $\epsilon(R) \sim \Delta\ (x)$.

Gamma and Log Gamma Subprograms

GAMMA/ALGAMA

Algorithm

1. If $0 < x \leqq 2^{-252}$, then compute log-gamma as $\log_e\Gamma(x) \cong -\log_e(x)$. This computation uses the real logarithm subprogram (ALOG).
2. If $2^{-252} < x < 8$, then compute log-gamma by taking the natural logarithm of the value obtained for gamma. The computation of gamma depends upon the range into which the argument falls.
3. If $2^{-252} < x < 1$, then use $\Gamma(x) = \frac{\Gamma(x+1)}{x}$ to reduce to the next case.
4. If $1 \leqq x \leqq 2$, then compute gamma by the minimax rational approximation (in absolute error) of the following form:

$$\Gamma(x) \cong c_0 + \frac{z\,[a_0 + a_1 z + a_2 z^2 + a_3 z^3]}{b_0 + b_1 z + b_2 z^2 + z^3}$$

where $z = x - 1.5$. The absolute error of this approximation is less than $2^{-25.9}$
5. If $2 < x < 8$, then use $\Gamma(x) = (x-1)\,\Gamma(x-1)$ to reduce step by step to the preceding case.
6. If $8 \leqq x$, then compute log-gamma by the use of Stirling's formula:

$$\log_e\Gamma(x) \cong x(\log_e(x) - 1) - \tfrac{1}{2}\log_e(x) + \tfrac{1}{2}\log_e(2\pi) + G(x).$$

The modifier term $G(x)$ is computed as

$$G(x) \cong d_0 x^{-1} + d_1 x^{-2}.$$

These coefficients were obtained by a form of minimax approximation minimizing the ratio of the absolute error to the value of x. The absolute error is less than $x \cdot 2^{-26.2}$. Remembering the fact that $x < \log_e\Gamma(x)$ in this range, the contribution of this error to the relative error of the value for log-gamma is less

than $2^{-26.2}$. This computation uses the real logarithm subprogram (ALOG).

For gamma, compute $\Gamma(x) = e^y$, where y is the value obtained for log-gamma. This computation uses the real exponential subprogram (EXP).

Effect of an Argument Error

$\epsilon \sim \psi(x) \cdot \Delta$ for gamma, and $E \sim \psi(x) \cdot \Delta$ for log-gamma, where ψ is the digamma function.

If $\frac{1}{2} < x < 3$, then $-2 < \psi(x) < 1$. Therefore, $E \sim \Delta$ for log-gamma. However, because $x = 1$ and $x = 2$ are zeros of the log-gamma function, even a small δ can cause a substantial ϵ in this range.

If the value of x is large, then $\psi(x) \sim \log_e(x)$. Therefore, for gamma, $\epsilon \sim \delta x \cdot \log_e(x)$. In this case, even the round-off error of the argument contributes greatly to the relative error of the answer. For log-gamma with large values of x, $\epsilon \sim \delta$.

DGAMMA/DLGAMA

Algorithm

1. If $0 < x \leqq 2^{-252}$, then compute log-gamma as $\log_e\Gamma(x) \cong -\log_e(x)$. This computation uses the real logarithm subprogram (DLOG).
2. If $2^{-252} < x < 8$, then compute log-gamma by taking the natural logarithm of the value obtained for gamma. The computation of gamma depends upon the range into which the argument falls.
3. If $2^{-252} < x < 1$, then use $\Gamma(x) = \frac{\Gamma(x+1)}{x}$ to reduce to the next case.
4. If $1 \leqq x \leqq 2$, then compute gamma by the minimax rational approximation (in absolute error) of the following form:

$$\Gamma(x) \cong c_0 + \frac{z[a_0 + a_1 z + \ldots + a_6 z^6]}{b_0 + b_1 z + \ldots + b_6 z^6 + z^7}$$

where $z = x - 1.5$. The absolute error of this approximation is less than $2^{-59.8}$.
5. If $2 < x < 8$, then use $\Gamma(x) = (x-1)\,\Gamma(x-1)$ to reduce to the preceding case.
6. If $8 \leqq x$, then compute log-gamma by the use of Stirling's formula:

$$\log_e\Gamma(x) \cong x(\log_e(x) - 1) - \tfrac{1}{2}\log_e(x) + \tfrac{1}{2}\log_e(2\pi) + G(x).$$

The modifier term $G(x)$ is computed as

$$G(x) \cong d_0x^{-1} + d_1x^{-3} + d_2x^{-5} + d_3x^{-7} + d_4x^{-9}.$$

These coefficients were obtained by a form of minimax approximation minimizing the ratio of the absolute error to the value of x. The absolute error is less than $x \cdot 2^{-56.1}$. Remembering the fact that $x < \log_e\Gamma(x)$ in this range, the contribution of this error to the relative error of the value for log-gamma is less than $2^{-56.1}$. This computation uses the real logarithm subprogram (DLOG). For gamma, compute $\Gamma(x) = e^y$, where y is the value obtained for log-gamma. This computation uses the real exponential subprogram (DEXP).

Effect of an Argument Error

$\epsilon \sim \psi(x) \cdot \Delta$ for gamma, and $E \sim \psi(x) \cdot \Delta$ for log-gamma, where ψ is the digamma function.

If $\frac{1}{2} < x < 3$, then $-2 < \psi(x) < 1$. Therefore, $E \sim \Delta$ for log-gamma. However, because $x = 1$ and $x = 2$ are zeros of the log-gamma function, even a small δ can cause a substantial ϵ in this range.

If the value of x is large, then $\psi(x) \sim \log_e(x)$. Therefore, for gamma, $\epsilon \sim \delta \cdot x \cdot \log_e(x)$. In this case, even the round-off error of the argument contributes greatly to the relative error of the answer. For log-gamma with large values of x, $\epsilon \sim \delta$.

Hyperbolic Sine and Cosine Subprograms

SINH/COSH

Algorithm

1. If $|x| < 1.0$, then compute $\sinh(x)$ as:
   $$\sinh(x) \cong x + c_1x^3 + c_2x^5 + c_3x^7.$$
   The coefficient c_i were obtained by the minimax approximation (in relative error) of $\dfrac{\sinh(x)}{x}$ as the function of x^2. The maximum relative error of this approximation is $2^{-25.6}$.
2. If $x \geqq 1.0$, then $\sinh(x)$ is computed as:
   $$\sinh(x) = (1 + \delta)\,[e^{x+\log_e v} - v^2/e^{x+\log_e v}].$$
   Here, $1 + \delta = \dfrac{1}{2v}$, so that this expression is theoretically equivalent to $[e^x - e^{-x}]/2$. The value of v (and consequently those of $\log_e v$ and δ) was so chosen as to satisfy the following conditions:
   a) v is slightly less than ½, so that $\delta > 0$ and small.
   b) $\log_e v$ is an exact multiple of 2^{-16}.
   The condition *b*) insures that the addition $x + \log_e v$ is carried out exactly. This maneuver was designed to reduce the round-off errors and also to enlarge the limits of acceptable arguments. This computation uses the real exponential subprogram (EXP).
3. If $x \leqq -1.0$, use $\sinh(x) = -\sinh(|x|)$ to reduce to case 2 above.
4. If $\cosh(x)$ is desired, then for all valid values of arguments use the identity: $\cosh(x) = (1 + \delta)\,[e^{x+\log_e v} + v^2/e^{x+\log_e v}]$. Here the notation and the consideration are identical to case 2 above. This computation uses the real exponential subprogram (EXP).

Effect of an Argument Error

For the hyperbolic sine, $E \sim \Delta \cdot \cosh(x)$ and $\epsilon \sim \Delta \cdot \coth(x)$.
For the hyperbolic cosine, $E \sim \Delta \cdot \sinh(x)$ and $\epsilon \sim \delta \cdot \tanh(x)$.

Specifically, for the cosine, $\epsilon \sim \Delta$ over the entire range; for the sine, $\epsilon \sim \delta$ for small values of x.

DSINH/DCOSH

Algorithm

1. If $|x| < 0.881374$, then compute $\sinh(x)$ as:
   $$\sinh(x) \cong c_0x + c_1x^3 + c_2x^5 + \ldots + c_6x^{13}.$$
   The coefficients c_i were obtained by the minimax approximation (in relative error) of $\dfrac{\sinh(x)}{x}$ as the function of x^2. Minimax was taken under the constraint that $c_0 = 1$ exactly. The maximum relative error of this approximation is $2^{-55.7}$.
2. If $x \geqq 0.881374$, then $\sinh(x)$ is computed as:
   $$\sinh(x) = (1 + \delta)\,[e^{x+\log_e v} - v^2/e^{x+\log_e v}].$$

Here, $1 + \delta = \frac{1}{2v}$, so that this expression is theoretically equivalent to $[e^x - e^{-x}]/2$. The value of v (and consequently those of $\log_e v$ and δ) was so chosen as to satisfy the following conditions:

a) v is slightly less than ½, so that $\delta > 0$ and small.

b) $\log_e v$ is an exact multiple of 2^{-16}.

The condition *b*) insures that the addition $x + \log_e v$ is carried out exactly. This maneuver was designed to reduce the round-off errors and also to enlarge the limits of acceptable arguments. This computation uses the real exponential subprogram (DEXP).

3. If $x \leqq -0.881374$, then use $\sinh(x) = -\sinh(|x|)$ to reduce to case 2 above.
4. If $\cosh(x)$ is desired, then, for all valid arguments use the identity: $\cosh(x) = (1 + \delta)\,[e^{x+\log_e v} + v^2/e^{x+\log_e v}]$. Here the notation and the consideration are identical to case 2 above. This computation uses the real exponential subprogram (DEXP).

Effect of an Argument Error

For the hyperbolic sine, $E \sim \Delta \cdot \cosh(x)$ and $\epsilon \sim \Delta \cdot \coth(x)$.
For the hyperbolic cosine, $E \sim \Delta \cdot \sinh(x)$ and $\epsilon \sim \Delta \cdot \tanh(x)$.

Specifically, for the cosine, $\epsilon \sim \Delta$ over the entire range; for the sine, $\epsilon \sim \delta$ for the small values of x.

Hyperbolic Tangent Subprograms

TANH

Algorithm

1. If $|x| \leqq 2^{-12}$, then $\tanh(x) \cong x$.
2. If $2^{-12} < |x| \leqq 0.7$, use the following fractional approximation:

$$\frac{\tanh(x)}{x} \cong 1 - x^2\left[0.0037828 + \frac{0.8145651}{x^2 + 2.471749}\right].$$

The coefficients of this approximation were obtained by taking the minimax of relative error, over the range $x^2 < 0.49$, of approximations of this form under the constraint that the first term shall be exactly 1.0. The maximum relative error of this approximation is $2^{-26.4}$.

3. If $0.7 < x < 9.011$, then use the identity $\tanh(x) = 1 - \frac{2}{(e^x)^2 + 1}$.

The computation for this case uses the real exponential subprogram (EXP).

4. If $x \geqq 9.011$, than $\tanh(x) \cong 1$.
5. If $x < -0.7$, then use the identity $\tanh(x) = -\tanh(-x)$.

Effect of an Argument Error

$E \sim (1 - \tanh^2 x)\,\Delta$, and $\epsilon \sim \frac{2\Delta}{\sinh(2x)}$. For small values of x, $\epsilon \sim \delta$, and as the value of x increases, the effect of δ upon ϵ diminishes.

DTANH

Algorithm

1. If $|x| \leqq 2^{-28}$, then $\tanh(x) \cong x$.
2. If $2^{-28} < |x| < 0.54931$, use the following fractional approximation:

$$\frac{\tanh(x)}{x} \cong c_0 + \frac{d_1 x^2}{x^2 + c_1 +} \; \frac{d_2}{x^2 + c_2 +} \; \frac{d_3}{x^2 + c_3}.$$

This approximation was obtained by rewriting a minimax approximation of the following form:

$$\frac{\tanh(x)}{x} \cong c_0 + x^2 \cdot \frac{a_0 + a_1x^2 + a_2x^4}{b_0 + b_1x^2 + b_2x^4 + x^6}.$$

Here the minimax of relative error, over the range $x^2 \leqq 0.30174$, was taken under the constraint that c_0 shall be exactly 1.0. The maximum relative error of the above is 2^{-63}.

3. If $0.54931 \leqq x < 20.101$, then use the identity $\tanh(x) = 1 - \frac{2}{e^{2x} + 1}$.

   This computation uses the double precision exponential subprogram (DEXP).
4. If $x \geqq 20.101$, then $\tanh(x) \cong 1$.
5. If $x \leqq -0.54931$, then use the identity $\tanh(x) = -\tanh(-x)$.

Effect of an Argument Error

$E \sim (1 - \tanh^2 x)\,\Delta$, and $\epsilon \sim \frac{2\,\Delta}{\sinh(2x)}$. For small values of x, $\epsilon \sim \delta$. As the value of x increases, the effect of δ upon ϵ diminishes.

Logarithmic Subprograms (Common and Natural)

ALOG/ALOG10

Algorithm

1. Write $x = 16^p \cdot 2^{-q} \cdot m$ where p is the exponent, q is an integer, $0 \leqq q \leqq 3$, and m is within the range, $\frac{1}{2} \leqq m < 1$.
2. Define two constants, a and b (where a = base point and $2^{-b} = a$), as follows:

   If $\frac{1}{2} \leqq m < \frac{1}{\sqrt{2}}$, then $a = \frac{1}{2}$ and $b = 1$.

   If $\frac{1}{\sqrt{2}} \leqq m < 1$, then $a = 1$ and $b = 0$.
3. Write $z = \frac{m - a}{m + a}$. Then, $m = a \cdot \frac{1 + z}{1 - z}$ and $|z| < 0.1716$.
4. Now, $x = 2^{4p-q-b} \cdot \frac{1 + z}{1 - z}$, and $\log_e(x) = (4p - q - b)\log_e 2 + \log_e\left(\frac{1 + z}{1 - z}\right)$.
5. To obtain $\log_e\left(\frac{1 + z}{1 - z}\right)$, first compute $w = 2z = \frac{m - a}{0.5m + 0.5a}$ (which is represented in our system with slightly more significant digits than z itself), and apply an approximation of the following form:

   $$\log_e\left(\frac{1 + z}{1 - z}\right) \cong w\left[c_0 + \frac{c_1w^2}{c_2 - w^2}\right].$$

   These coefficients were obtained by the minimax rational approximation of $\frac{1}{2z}\log_e\left(\frac{1 + z}{1 - z}\right)$ over the range $z^2 \in (0, 0.02944)$ under the constraint that c_0 shall be exactly 1.0. The maximum relative error of this approximation is less than $2^{-25.33}$.
6. If the common logarithm is desired, then $\log_{10}x = \log_{10}e \cdot \log_e x$.

Effect of an Argument Error

$E \sim \delta$. Specifically, if δ is the round-off error of the argument, e.g., $\delta \sim 6 \cdot 10^{-8}$, then $E \sim 6 \cdot 10^{-8}$. Therefore, if the argument is close to 1, the relative error can be very large because the value of the function is very small.

DLOG/DLOG10

Algorithm

1. Write $x = 16^p \cdot 2^{-q} \cdot m$ where p is the exponent, q is an integer, $0 \leqq q \leqq 3$, and m is within the range $\frac{1}{2} \leqq m < 1$.
2. Define two constants, a and b (where a = base point and $2^{-b} = a$), as follows:

   If $\frac{1}{2} \leqq m < \frac{1}{\sqrt{2}}$, then $a = \frac{1}{2}$ and $b = 1$.

   If $\frac{1}{\sqrt{2}} \leqq m < 1$, then $a = 1$ and $b = 0$.
3. Write $z = \dfrac{m - a}{m + a}$. Then, $m = a \cdot \dfrac{1 + z}{1 - z}$ and $|z| < 0.1716$.
4. Now, $x = 2^{4p-q-b} \cdot \dfrac{1 + z}{1 - z}$, and $\log_e x = (4p - q - b)\log_e 2 + \log_e\left(\dfrac{1 + z}{1 - z}\right)$.
5. To obtain $\log_e\left(\dfrac{1 + z}{1 - z}\right)$, first compute $w = 2z = \dfrac{m - a}{0.5m + 0.5a}$ (which is represented in our system with slightly more significant digits than z itself), and apply an approximation of the following form:

$$\log_e\left(\frac{1 + z}{1 - z}\right) \cong w\left[c_0 + c_1 w^2\left(w^2 + c_2 + \frac{c_3}{w^2 + c_4 + \dfrac{c_5}{w^2 + c_6}}\right)\right].$$

   These coefficients were obtained by the minimax rational approximation of $\dfrac{1}{2z}\log_e\left(\dfrac{1 + z}{1 - z}\right)$ over the range $z^2 \in (0, 0.02944)$ under the constraint that c_0 shall be exactly 1.0. The maximum relative error of this approximation is less than $2^{-60.55}$.
6. If the common logarithm is desired, then $\log_{10} x = \log_{10} e \cdot \log_e x$.

Effect of an Argument Error

$E \sim \delta$. Therefore, if the value of the argument is close to 1, the relative error can be very large because the value of the function is very small.

CLOG/CDLOG

Algorithm

1. Write $\log_e(x + iy) = a + ib$.
2. Then, $a = \log_e|x + iy|$ and b = the principal value of arctan (y, x).
3. $\log_e|x + iy|$ is computed as follows:
   Let $v_1 = \max(|x|, |y|)$, and $v_2 = \min(|x|, |y|)$.

   Let t be the exponent of v_1, i.e., $v_1 = m \cdot 16^t$, $\dfrac{1}{16} \leqq m < 1$.

   Finally, let $t_1 = \begin{cases} t \text{ if } t \leqq 0 \\ t - 1 \text{ if } t > 0 \end{cases}$,

   and $s = 16^{t_1}$.

   Then, $\log_e|x + iy| = 4t_1 \cdot \log_e(2) + \frac{1}{2}\log_e\left[\left(\dfrac{v_1}{s}\right)^2 + \left(\dfrac{v_2}{s}\right)^2\right]$.

   Computation of v_1/s and v_2/s are carried out by manipulation of the characteristics of v_1 *and* v_2. In particular, if $v_2/s \ll 1$, it is taken to be 0. The algorithms for both complex logarithm subprograms are identical. Each subprogram uses the appropriate real natural logarithm subprogram (ALOG or DLOG) and the appropriate arctangent subprogram (ATAN2 or DATAN2).

Effect of an Argument Error

The effect of an argument error depends upon the accuracy of the individual parts of the argument. If $x + iy = r \cdot e^{ih}$ and $\log_e (x + iy) = a + ib$, then $h = b$ and $E(a) = \delta(r)$.

Sine and Cosine Subprograms

SIN/COS

Algorithm

1. Define $z = \frac{4}{\pi} \cdot |x|$ and separate z into its integer part (q) and its fraction part (r). Then $z = q + r$, and $|x| = \left(\frac{\pi}{4} \cdot q\right) + \left(\frac{\pi}{4} \cdot r\right)$.

2. If the cosine is desired, add 2 to q. If the sine is desired and if x is negative, add 4 to q. This adjustment of q reduces the general case to the computation of $\sin(x)$ for $x \geqq 0$ because

$$\cos(\pm x) = \sin\left(\frac{\pi}{2} + x\right), \text{ and}$$
$$\sin(-x) = \sin(\pi + x).$$

3. Let $q_0 \equiv q \bmod 8$.

Then, for $q_0 = 0, \sin(x) = \sin\left(\frac{\pi}{4} \cdot r\right)$,

$$q_0 = 1, \sin(x) = \cos\left(\frac{\pi}{4}(1 - r)\right),$$
$$q_0 = 2, \sin(x) = \cos\left(\frac{\pi}{4} \cdot r\right),$$
$$q_0 = 3, \sin(x) = \sin\left(\frac{\pi}{4}(1 - r)\right),$$
$$q_0 = 4, \sin(x) = -\sin\left(\frac{\pi}{4} \cdot r\right),$$
$$q_0 = 5, \sin(x) = -\cos\left(\frac{\pi}{4}(1 - r)\right),$$
$$q_0 = 6, \sin(x) = -\cos\left(\frac{\pi}{4} \cdot r\right),$$
$$q_0 = 7, \sin(x) = -\sin\left(\frac{\pi}{4}(1 - r)\right).$$

These formulas reduce each case to the computation of either $\sin\left(\frac{\pi}{4} \cdot r_1\right)$ or $\cos\left(\frac{\pi}{4} \cdot r_1\right)$ where r_1 is either r or $(1 - r)$ and is within the range, $0 \leqq r_1 \leqq 1$.

4. If $\sin\left(\frac{\pi}{4} \cdot r_1\right)$ is needed, it is computed by a polynomial of the following form:

$$\sin\left(\frac{\pi}{4} \cdot r_1\right) \cong r_1 (a_0 + a_1 r_1^2 + a_2 r_1^4 + a_3 r_1^6).$$

The coefficients were obtained by the interpolation at the roots of the Chebyshev polynomial of degree 4. The relative error is less than $2^{-28.1}$ for the range.

5. If $\cos\left(\frac{\pi}{4} \cdot r_1\right)$ is needed, it is computed by a polynomial of the following form:

$$\cos\left(\frac{\pi}{4} \cdot r_1\right) \cong 1 + b_1 r_1^2 + b_2 r_1^4 + b_3 r_1^6.$$

Coefficients were obtained by a variation of the minimax approximation which provides a partial rounding for the short precision computation. The absolute error of this approximation is less than $2^{-24.57}$.

Effect of an Argument Error

$E \sim \Delta$. As the value of x increases, Δ increases. Because the function value diminishes periodically, no consistent relative error control can be maintained outside the principal range, $-\frac{\pi}{2} \leqq x \leqq +\frac{\pi}{2}$.

DSIN/DCOS

Algorithm

1. Divide $|x|$ by $\frac{\pi}{4}$ and separate the quotient (z) into its integer part (q) and its fraction part (r). Then, $z = |x| \cdot \frac{4}{\pi} = q + r$, where q is an integer and r is within the range, $0 \leqq r < 1$.
2. If the cosine is desired, add 2 to q. If the sine is desired and if x is negative, add 4 to q. This adjustment of q reduces the general case to the computation of $\sin(x)$ for $x \geqq 0$, because

$$\cos(\pm x) = \sin\left(|x| + \frac{\pi}{2}\right), \text{ and}$$
$$\sin(-x) = \sin(|x| + \pi).$$

3. Let $q_0 \equiv q \bmod 8$.

Then, for $q_0 = 0, \sin(x) = \sin\left(\frac{\pi}{4} \cdot r\right)$,

$q_0 = 1, \sin(x) = \cos\left(\frac{\pi}{4}(1 - r)\right)$,

$q_0 = 2, \sin(x) = \cos\left(\frac{\pi}{4} \cdot r\right)$,

$q_0 = 3, \sin(x) = \sin\left(\frac{\pi}{4}(1 - r)\right)$,

$q_0 = 4, \sin(x) = -\sin\left(\frac{\pi}{4} \cdot r\right)$,

$q_0 = 5, \sin(x) = -\cos\left(\frac{\pi}{4}(1 - r)\right)$,

$q_0 = 6, \sin(x) = -\cos\left(\frac{\pi}{4} \cdot r\right)$,

$q_0 = 7, \sin(x) = -\sin\left(\frac{\pi}{4}(1 - r)\right)$.

These formulas reduce each case to the computation of either $\sin\left(\frac{\pi}{4} \cdot r_1\right)$ or $\cos\left(\frac{\pi}{4} \cdot r_1\right)$; where r_1 is either r or $(1 - r)$, and is within the range, $0 \leqq r_1 \leqq 1$.

4. Finally, either $\sin\left(\frac{\pi}{4} \cdot r_1\right)$ or $\cos\left(\frac{\pi}{4} \cdot r_1\right)$ is computed, using the polynomial interpolations of degree 6 in r_1^2 for the sine, and of degree 7 in r_1^2 for the cosine. In either case, the interpolation points were the roots of the Chebyshev polynomial of one higher degree. The maximum relative error of the sine polynomial is 2^{-58} and that of the cosine polynomial is $2^{-64.3}$.

Effect of an Argument Error

$E \sim \Delta$. As the value of the argument increases, Δ increases. Because the function value diminishes periodically, no consistent relative error control can be maintained outside of the principal range, $-\frac{\pi}{2} \leqq x \leqq +\frac{\pi}{2}$.

CSIN/CCOS

Algorithm

1. If the sine is desired, then
$$\sin(x + iy) = \sin(x) \cdot \cosh(y) + i \cdot \cos(x) \cdot \sinh(y).$$
If the cosine is desired, then
$$\cos(x + iy) = \cos(x) \cdot \cosh(y) - i \cdot \sin(x) \cdot \sinh(y).$$
2. The value of $\sinh(x)$ is computed within the subprogram as follows. Assume $x \geqq 0$ for this, since $\sinh(-x) = -\sinh(x)$.
3. If $x \geqq 0.346574$, then use $\sinh(x) = \frac{1}{2}\left(e^x - \frac{1}{e^x}\right)$.
4. If $0 \leqq x < 0.346574$, then compute $\sinh(x)$ by use of a polynomial:
$$\frac{\sinh(x)}{x} \cong a_0 + a_1x^2 + a_2x^4.$$
The coefficients were obtained by the minimax approximation (in relative error) of $\sinh(x)/x$ over the range $0 \leqq x^2 \leqq 0.12011$ under the constraint that a_0 shall be exactly 1.0. The relative error of this approximation is less than $2^{-26.18}$.
5. The value of $\cosh(x)$ is computed as $\cosh(x) = \sinh|x| + \frac{1}{e^{|x|}}$.
This computation uses the real expoential subprogram (EXP) and the real sine/cosine subprogram (SIN/COS).

Effect of an Argument Error

To understand the effect of an argument error upon the accuracy of the answer, the programmer must understand the effect of an argument in the SIN/COS, EXP, and SINH/COSH subprograms.

CDSIN/CDCOS

Algorithm

1. If the sine is desired, then
$$\sin(x+iy) = \sin(x) \cdot \cosh(y) + i \cdot \cos(x) \cdot \sinh(y).$$
If the cosine is desired, then
$$\cos(x + iy) = \cos(x) \cdot \cosh(y) - i \cdot \sin(x) \cdot \sinh(y).$$
2. The value of $\sinh(x)$ is computed within the subprogram as follows. Assume $x \geqq 0$ for this, since $\sinh(-x) = -\sinh(x)$.
3. If $x \geqq 0.481212$, then use $\sinh(x) = \frac{1}{2}\left(e^x - \frac{1}{e^x}\right)$.

4. If $0 \leqq x < 0.481212$, then compute $\sinh(x)$ by use of a polynomial:

$$\frac{\sinh(x)}{x} \cong a_0 + a_1x^2 + a_2x^4 + a_3x^6 + a_4x^8 + a_5x^{10}.$$

The coefficients were obtained by the minimax approximation (in relative error) of $\sinh(x)/x$ over the range $0 \leqq x^2 \leqq 0.23156$ under the constraint that a_0 shall be exactly 1.0. The relative error of this approximation is less than $2^{-56.07}$.

5. The value of $\cosh(x)$ is computed as $\cosh(x) = \sinh|x| + \frac{1}{e^{|x|}}$.

This computation uses the real exponential subprogram (DEXP) and the real sine/cosine subprogram (DSIN/DCOS).

Effect of an Argument Error

To understand the effect of an argument error upon the accuracy of the answer, the programmer must understand the effect of an argument error in the DSIN/DCOS, DEXP, and DSINH/DCOSH subprograms.

Square Root Subprograms

SQRT

Algorithm

1. If $x = 0$, then the answer is 0.
2. Write $x = 16^{2p-q} \cdot m$, where $2p - q$ is the exponent and q equals either 0 or 1; m is the mantissa and is within the range $\frac{1}{16} \leqq m < 1$.
3. Then, $\sqrt{x} = 16^p \cdot 4^{-q}\sqrt{m}$.
4. For the first approximation of $\sqrt{x}$, compute the following:

$$y_0 = 16^p \cdot 4^{-q} \cdot \left(1.681595 - \frac{1.288973}{0.8408065 + m}\right).$$

This approximation attains the minimax relative error for hyperbolic fits of $\sqrt{x}$. The maximum relative error is $2^{-5.748}$.

5. Apply the Newton-Raphson iteration

$$y_{n+1} = \tfrac{1}{2}\left(y_n + \frac{x}{y_n}\right)$$

twice. The second iteration is performed as

$$y_2 = \tfrac{1}{2}\left(y_1 - \frac{x}{y_1}\right) + \frac{x}{y_1},$$

with a partial rounding. The maximum relative error of y_2 is theoretically $2^{-25.9}$.

Effect of an Argument Error

$$\epsilon \sim \frac{1}{2}\delta.$$

DSQRT

Algorithm

1. If $x = 0$, then the answer is 0.
2. Write $x = 16^{2p-q} \cdot m$, where $2p - q$ is the exponent and q equals either 0 or 1; m is the mantissa and is within the range $\frac{1}{16} \leqq m < 1$.

3. Then, $\sqrt{x} = 16^p \cdot 4^{-q} \sqrt{m}$.
4. For the first approximation of $\sqrt{x}$, compute the following:

   $$y_0 = 16^p \cdot 4^{1-q} \cdot 0.2202\,(m + 0.2587).$$

   The extrema of relative errors of this approximation for $q = 0$ are $2^{-3.202}$ at $m = 1$, $2^{-3.265}$ at $m = 0.2587$, and $2^{-2.925}$ at $m = \frac{1}{16}$. This approximation, rather than the minimax approximation, was chosen so that the quantity $\frac{x}{y_3} - y_3$ below becomes less than 16^{p-8} in magnitude. This arrangement allows us to substitute short form counterparts for some of the long form instructions in the final iteration.
5. Apply the Newton Raphson iteration

   $$y_{n+1} = \tfrac{1}{2}\left(y_n + \frac{x}{y_n}\right)$$

   four times to y_0, twice in the short form and twice in the long form. The final step is performed as

   $$y_4 = y_3 + \tfrac{1}{2}\left(\frac{x}{y_3} - y_3\right)$$

   with an appropriate truncation maneuver to obtain a virtual rounding. The maximum relative error of the final result is theoretically $2^{-63.23}$.

Effect of an Argument Error

$$\epsilon \sim \frac{1}{2}\,\delta$$

CSQRT/CDSQRT

Algorithm

1. Write $\sqrt{x + iy} = a + ib$.
2. Compute the value $z = \sqrt{\frac{|x| + |x + iy|}{2}}$ as $k \cdot \sqrt{w_1 + w_2}$ where k, w_1 and w_2 are defined in 3, or 4, below. In any case let $v_1 = \max(|x|, |y|)$ and
   $$v_2 = \min(|x|, |y|).$$
3. In the special case when either $v_2 = 0$ or $v_1 \gg v_2$, let $w_1 = v_2$ and $w_2 = v_1$ so that $w_1 + w_2$ is effectively equal to v_1.
   Also let $k = 1$ if $v_1 = |x|$ and
   $$k = 1/\sqrt{2} \text{ if } v_1 = |y|.$$
4. In the general case, compute $F = \sqrt{\tfrac{1}{4} + \tfrac{1}{4}\left(\frac{v_2}{v_1}\right)^2}$.
   If $|x|$ is near the underflow threshold, then take
   $$w_1 = |x|,\ w_2 = v_1 \cdot 2F, \text{ and } k = 1/\sqrt{2}.$$
   If $v_1 \cdot F$ is near the overflow threshold, then take
   $$w_1 = |x|/4,\ w_2 = v_1 \cdot F/2, \text{ and } k = \sqrt{2}.$$
   In all other cases, take $w_1 = |x|/2$, $w_2 = v_1 \cdot F$, and $k = 1$.
5. If $z = 0$, then $a = 0$ and $b = 0$.
   If $z \neq 0$ and $x \geqq 0$, then $a = z$, and
   $$b = \frac{y}{2z}.$$
   If $z \neq 0$ and $x < 0$, then $a = \left|\frac{y}{2z}\right|$, and
   $$b = (\text{sign } y) \cdot z.$$

The algorithms for both complex square root subprograms are identical. Each subprogram uses the appropriate real square root subprogram (SQRT or DSQRT).

Effect of an Argument Error

The effect of an argument error depends upon the accuracy of the individual parts of the argument. If $x + iy = r \cdot e^{ih}$ and $\sqrt{x + iy} = R \cdot e^{iH}$,

then $\epsilon(R) \sim \frac{1}{2}\,\delta(r)$, and $\epsilon(H) \sim \delta(h)$.

Tangent and Cotangent Subprograms

TAN/COTAN

Algorithm

1. Divide $|x|$ by $\frac{\pi}{4}$ and separate the result into integer part (q) and the fraction part (r). Then $|x| = \frac{\pi}{4}(q + r)$.
2. Obtain the reduced argument (w) as follows:
   if q is even, then $w = r$
   if q is odd, then $w = 1 - r$.
   The range of the reduced argument is $0 \leqq w \leqq 1$.
3. Let $q_0 \equiv q \bmod 4$.
   Then for $q_0 = 0$, $\tan|x| = \tan\left(\frac{\pi}{4} \cdot w\right)$ and $\cot|x| = \cot\left(\frac{\pi}{4} \cdot w\right)$,
   $q_0 = 1$, $\tan|x| = \cot\left(\frac{\pi}{4} \cdot w\right)$ and $\cot|x| = \tan\left(\frac{\pi}{4} \cdot w\right)$,
   $q_0 = 2$, $\tan|x| = -\cot\left(\frac{\pi}{4} \cdot w\right)$ and $\cot|x| = -\tan\left(\frac{\pi}{4} \cdot w\right)$,
   $q_0 = 3$, $\tan|x| = -\tan\left(\frac{\pi}{4} \cdot w\right)$ and $\cot|x| = -\cot\left(\frac{\pi}{4} \cdot w\right)$.
4. The value of $\tan\left(\frac{\pi}{4} \cdot w\right)$ and $\cot\left(\frac{\pi}{4} \cdot w\right)$ are computed as the ratio of two polynomials:
   $$\tan\left(\frac{\pi}{4} \cdot w\right) \cong \frac{w \cdot P(u)}{Q(u)}, \quad \cot\left(\frac{\pi}{4} \cdot w\right) \cong \frac{Q(u)}{w \cdot P(u)}$$
   where $u = \frac{1}{2}w^2$ and
   $P(u) = -8.460901 + u$
   $Q(u) = -10.772754 + 5.703366 \cdot u - 0.159321 \cdot u^2$.
   These coefficients were obtained by the minimax rational approximation (in relative error) of the indicated form. The maximum relative error of this approximation is 2^{-26}. Choice of u rather than w^2 as the variable for P and Q is to improve the round-off quality of the coefficients.
5. If $x < 0$, then $\tan(x) = -\tan|x|$, and $\cot(x) = -\cot|x|$.
6. This program is provided with two kinds of error controls. One is for arguments whose magnitude is greater than $2^{18} \cdot \pi$. The other is for arguments which are very close to a singularity of the function. In either case, the precision of the argument is deemed insufficient for obtaining a reliable result. More specifically, the second control screens out the following arguments:
   a) $|x| \leqq 16^{-63}$ for COTAN (the result would overflow).
   b) x is such that one can find a singularity within eight units of the last digit

value of the floating-point representation of the sum $q + r$. Singularities are cases when the cotangent ratio is to be taken and $w = 0$.

The test threshold of this control can be dynamically modified by assembler code programs.

Effect of an Argument Error

$E \sim \frac{\Delta}{\cos^2(x)}$, and $\epsilon \sim \frac{2}{\sin(2x)}$ for $\tan(x)$. Therefore, near the singularities $x = \left(k + \frac{1}{2}\right)\pi$, where k is an integer, no error control can be maintained. This is also true for $\text{cotan}(x)$ for x near $k\pi$, where k is an integer.

DTAN/DCOTAN

Algorithm

1. Divide $|x|$ by $\frac{\pi}{4}$ and separate the result into integer part (q) and the fraction part (r). Then $|x| = \frac{\pi}{4}(q + r)$.
2. Obtain the reduced argument (w) as follows:
   if q is even, then $w = r$
   if q is odd, then $w = 1 - r$.

   The range of the reduced argument is $0 \leqq w \leqq 1$.
3. Let $q_0 \equiv q \bmod 4$.

   Then for $q_0 = 0$, $\tan|x| = \tan\left(\frac{\pi}{4} \cdot w\right)$ and $\cot|x| = \cot\left(\frac{\pi}{4} \cdot w\right)$,

   $q_0 = 1$, $\tan|x| = \cot\left(\frac{\pi}{4} \cdot w\right)$ and $\cot|x| = \tan\left(\frac{\pi}{4} \cdot w\right)$,

   $q_0 = 2$, $\tan|x| = -\cot\left(\frac{\pi}{4} \cdot w\right)$ and $\cot|x| = -\tan\left(\frac{\pi}{4} \cdot w\right)$,

   $q_0 = 3$, $\tan|x| = -\tan\left(\frac{\pi}{4} \cdot w\right)$ and $\cot|x| = -\cot\left(\frac{\pi}{4} \cdot w\right)$.
4. The value of $\tan\left(\frac{\pi}{4} \cdot w\right)$ and $\cot\left(\frac{\pi}{4} \cdot w\right)$ are computed as the ratio of two polynomials:

   $$\tan\left(\frac{\pi}{4} \cdot w\right) \cong \frac{w \cdot P(w^2)}{Q(w^2)}, \text{ and } \cot\left(\frac{\pi}{4} \cdot w\right) \cong \frac{Q(w^2)}{w \cdot P(w^2)}.$$

   where both P and Q are polynomials of degree 3 in w^2. The coefficients of P and Q were obtained by the minimax rational approximation (in relative error) of $\frac{1}{w}\tan\left(\frac{\pi}{4}w\right)$ of the indicated form. The maximum relative error of this approximation is $2^{-55.6}$.
5. If $x < 0$, then $\tan(x) = -\tan|x|$, and $\cot(x) = -\cot|x|$.
6. This program is provided with two kinds of error controls. One is for arguments whose magnitude is greater than $2^{50} \cdot \pi$. The other is for arguments which are very close to a singularity of the function. In either case, the precision of the argument is deemed insufficient for obtaining a reliable result. More specifically, the second control screens out the following arguments:
   a) $|x| \leqq 16^{-63}$ for COTAN (the result would ov flow).
   b) x is such that one can find a singularity v in eight units of the last digit value of the floating-point representation o the sum $q + r$. Singularities are cases when the cotangent ratio is to be take and $w = 0$.

The test threshold of this control can be dynamically modified by assembler code programs.

Effect of an Argument Error

$E \sim \frac{\Delta}{\cos^2(x)}$, and $\epsilon \sim \frac{2}{\sin(2x)}$ for $\tan(x)$. Therefore, near the singularities of $x = \left(k + \frac{1}{2}\right)\pi$, where k is an integer, no error control can be maintained. This is also true for $\text{cotan}(x)$ for values of x near $k\pi$, where k is an integer.

Implicitly Called Subprograms

The entry point names of the following implicitly called subprograms are generated by the compiler.

Complex Multiply and Divide Subprograms

CDVD#/CMPY# (Divide/Multiply for COMPLEX*8 Arguments)

CDDVD#/CDMPY# (Divide/Multiply for COMPLEX*16 Arguments)

Algorithm

Multiply: $(A + Bi)(C + Di) = (AC - BD) + (AD + BC)i$

Divide: $(A + Bi)/(C + Di)$

1. If $|C| \leqq |D|$, set
   $A = B, B = -A, C = D, D = -C$, since
   $\frac{A + Bi}{C + Di} = \frac{B - Ai}{D - Ci}$ before step 2.
2. Set $A' = \frac{A}{C}, B' = \frac{B}{C}, D' = \frac{D}{C}$;
   then compute
   $$\frac{A + Bi}{C + Di} = \frac{A' + B'i}{1 + D'i} = \frac{A' + B'D}{1 + D'D'} + \frac{B' - A'D'}{1 + D'D'}i.$$

Error Conditions

Partial underflows can occur in preparing the answer.

Complex Exponentiation Subprograms

FCDXI# (COMPLEX*16 Arguments)

FCXPI# (COMPLEX*8 Arguments)

Algorithm

The value of $y_1 + y_2 i = (z_1 + z_2 i)^j$ is computed as follows.

Let $|j| = \sum_{k=0}^{K} r_k \cdot 2^k$ where $r_k = 0$ or 1 for $k = 0, 1, \ldots, K$.

Then $z^{|j|} = \prod_{r_k \neq 0} z^{2^k}$, and the factors z^{2^k} can be obtained by successive squaring.

More specifically:

1. Initially: $k = 0, n^{(0)} = |j|, y_1^{(0)} + y_2^{(0)} i = 1 + 0i$,
   $z_1^{(0)} + z_2^{(0)} i = z_1 + z_2 i$.

2. Raise the index k by 1, and let $n^{(k-1)} = 2q + r$, where q is the integer quotient and $r = 0$ or 1.
3. Let $n^{(k)} = q$.
4. If $r = 0$, then $y_1^{(k)} + y_2^{(k)}i = y_1^{(k-1)} + y_2^{(k-1)}i$.
   If $r = 1$, then $y_1^{(k)} + y_2^{(k)}i = (y_1^{(k-1)} + y_2^{(k-1)}i)\ (z_1^{(k-1)} + z_2^{(k-1)}i)$.
5. If $n^{(k)} \neq 0$, then $z_1^{(k)} + z_2^{(k)}i = (z_1^{(k-1)} + z_2^{(k-1)})^2$, and steps 2 through 5 are repeated until $n^{(k)} = 0$.
6. When $n^{(k)} = 0$, and $j \geqq 0$, then $y_1 + y_2i = y_1^{(k)} + y_2^{(k)}i$.
   If $j < 0$, then $y_1 + y_2i = (1 + 0i) / (y_1^{(k)} + y_2^{(k)}i)$.

Exponentiation of a Real Base to a Real Power Subprograms

FDXPD# (REAL*8 Arguments)

FRXPR# (REAL*4 Arguments)

Algorithm

1. If $a = 0$ and $b \leqq 0$, error return.
   If $a = 0$ and $b > 0$, the answer is 0.
2. If $a \neq 0$ and $b = 0$, the answer is 1.
3. All other cases, compute a^b as $e^{b \cdot \log a}$. In this computation the exponential subroutine and the natural logarithm subroutine are used. If a is negative or if $b \cdot \log a$ is too large, an error return is given by one of these subroutines.

Error estimate

The relative error of the answer can be expressed as $(\epsilon_1 + \epsilon_2)\ b \cdot \log\ (a) + \epsilon_3$ where ϵ_1, ϵ_2, and ϵ_3 are relative errors of the logarithmic routine, machine multiplication, and the exponential routine, respectively.

For FDXPD#, $\epsilon_1 \leqq 3.5x10^{-16}$, $\epsilon_2 \leqq 2.2x10^{-16}$, and $\epsilon_3 \leqq 2.0x10^{-16}$. Hence the relative error $\leqq 5.7x10^{-16}x \mid b \cdot \log a \mid + 2.0x10^{-16}$. Note that $b \cdot \log a$ is the natural logarithm of the answer.

For FRXPR#, $\epsilon_1 \leqq 8.3x10^{-7}$, $\epsilon_2 \leqq 9.5x10^{-7}$, and $\epsilon_3 \leqq 4.7x10^{-7}$. Hence the relative error $\leqq 1.8x10^{-6}\,x \mid b \cdot \log a \mid + 4.7x10^{-7}$.

Effect of an Argument Error

$[a(1 + \delta_1)]\ b(1 + \delta_2) \cong a^b(1 + \delta_2 b \cdot \log a + b\delta_1)$. Note that if the answer does not overflow, $|b \cdot \log a| < 175$. On the other hand b can be very large without causing an overflow of a^b if $\log a$ is very small. Thus, if $a \cong 1$ and if b is very large, then the effect of the perturbation δ_1 of a shows very heavily in the relative error of the answer.

Exponentiation of a Real Base to an Integer Power Subprograms

FDXPI# (REAL*8 Arguments)

FRXPI# (REAL*4 Arguments)

Algorithm

The value of $y = a^j$ is computed as follows: Let $|j| = \sum_{k=0}^{K} r_k 2^k$ where $r_k = 0$ or 1 for $k = 0, 1, \ldots, K$. Then $a^{|j|} = \prod_{r_k \neq 0} a^{2^k}$ and the factors a^{2^k} can be obtained by successive squaring.

More specifically:

1. Initially: $k = 0$, $n^{(0)} = |j|$, $y^{(0)} = 1$, and $z^{(0)} = a$.

2. Raise the index k by 1, and decompose $n^{(k-1)} = 2q + r$, where q is the integer quotient and $r = 0$ or 1.
3. Let $n^{(k)} = q$.
4. If $r = 0$, then $y^{(k)} = y^{(k-1)}$.
   If $r = 1$, then $y^{(k)} = y^{(k-1)}z^{(k-1)}$.
5. If $n^{(k)} \neq 0$, then $z^{(k)} = z^{(k-1)}z^{(k-1)}$, and steps 2 through 5 are repeated until $n^{(k)} = 0$.
6. When $n^{(k)} = 0$, and $j \geqq 0$, then $y = y^{(k)}$. If $j < 0$, then $y = \frac{1}{y^{(k)}}$.

Note: The negative exponent is computed by taking the reciprocal of the positive power. Thus it is not possible to compute 16.0**−64 because there is a lack of symmetry for real floating-point numbers – i.e., 16.0**−64 can be represented, but 16.0**64 cannot. The result is obtained by successive multiplications and is exact only if the answer contains less than 14 significant hexadecimal digits.

Exponentiation of an Integer Base to an Integer Power Subprogram

FIXPI# (INTEGER*4 Arguments)

Algorithm

The value of $L = I^j$ is computed as follows: Let $j = \sum_{k=0}^{K_1} r_k \cdot 2^k$ where $r_k = 0$ or 1 for $k = 0, 1, \ldots, K$. Then $I^j = \prod_{r_k \neq 0} I^{2^k}$, and the factors I^{2^k} can be obtained by successive squaring.

More specifically:

1. Initially: $k = 0$, $n^{(0)} = j$, $y^{(0)} = 1$, and $m^{(0)} = I$.
2. Raise the index k by 1, and decompose $n^{(k-1)} = 2q + r$, where q is the integer quotient and $r = 0$ or 1.
3. Let $n^{(k)} = q$.
4. If $r = 0$, then $y^{(k)} = y^{(k-1)}$.
   If $r = 1$, then $y^{(k)} = y^{(k-1)} \cdot m^{(k-1)}$.
5. If $n^{(k)} \neq 0$, then $m^{(k)} = m^{(k-1)} \cdot m^{(k-1)}$, and steps 2 through 5 are repeated until $n^{(k)} = 0$.
6. When $n^{(k)} = 0$, $L = L^{(k)}$.

Note: The result is obtained by successive multiplications. The result is exact only if it is less than (2**31) − 1. Results are meaningless when this limit is exceeded and may even be of changed sign.

This chapter contains accuracy and timing statistics for the explicitly called mathematical subprograms. These statistics are presented in Tables 12 and 13 and are arranged in alphabetical order, according to the entry names. The following information is given in the two tables:

Entry Name: This column gives the entry name that must be used to call the subprogram.

Argument Range: This column gives the argument range used to obtain the accuracy figures. For each function, accuracy figures are given for one or more representative segments within the valid argument range. In each case, the figures given are the most meaningful to the function and range under consideration.

The maximum relative error and standard deviation of the relative error are generally useful and revealing statistics; however, they are useless for the range of a function where its value becomes 0, because the slightest error in the argument can cause an unpredictable fluctuation in the magnitude of the answer. When a small argument error would have this effect, the maximum absolute error and standard deviation of the absolute error are given for the range. For example, absolute error is given for $\sin(x)$ for values of x near π.

Sample: This column indicates the type of sample used for the accuracy figures. The type of sample depends upon the function and range under consideration. The statistics may be based either upon an exponentially distributed (E) argument sample or a uniformly distributed (U) argument sample.

Accuracy Figures: This column gives accuracy figures for one or more representative segments within the valid argument range. The accuracy figures supplied are based upon the assumption that the arguments are perfect (i.e., without error and, therefore, having no error propagation effect upon the answers). The only errors in the answers are those introduced by the subprograms. The chapter, "Algorithms," contains a description of some of the symbols used in this chapter; the following additional symbols are used in the presentation of accuracy figures:

$$M(\epsilon) = \mathrm{Max}\left|\frac{f(x) - g(x)}{f(x)}\right|$$

The maximum relative error produced during testing.

$$\sigma(\epsilon) = \sqrt{\frac{1}{N}\sum_i \left|\frac{f(x_i) - g(x_i)}{f(x_i)}\right|^2}$$

The standard deviation (root-mean-square) of the relative error.

$$M(E) = \mathrm{Max}\,|f(x) - g(x)|$$

The maximum absolute error produced during testing.

$$\sigma(E) = \sqrt{\frac{1}{N}\sum_i |f(x_i) - g(x_i)|^2}$$

The standard deviation (root-mean-square) of the absolute error.

In case of complex functions, the absolute value signs employed in the above definitions are to mean the complex absolute values. In the formulas for the standard deviation, N represents the total number of arguments in the sample; i is a subscript that varies from 1 to N.

Average Speed: The average time given for each function was determined by executing a job using an interval timer to measure time differences. Because of the various methods of performance enhancement used in some models of the System/360 (e.g., interleaving storage, buffered storage, execution overlap, anticipatory branching analysis), instruction order and mix can affect performance on certain models. Consequently, the times given must be considered as highly generalized and differences may be observed during normal use of the functions.

Test ranges, where they do not cover the entire legal range of a subroutine, were selected so that users may infer from the accuracy figures presented the trend of errors as an argument moves away from the principal range. The accuracy of the answer deteriorates substantially as the argument approaches the limit of the permitted range in several of the subroutines. This is particularly true for trigonometric functions. An error generated by any of these subroutines, however, is at worst comparable in order of magnitude to the effect of the inherent rounding error of the argument.

Table 12. Accuracy Figures

Entry Name	Argument Range	Sample E/U	Accuracy Figures			
			Relative		Absolute	
			M (ϵ)	σ (ϵ)	M (E)	σ (E)
ALGAMA	$0 < X < 0.5$	U	1.16×10^{-6}	3.54×10^{-7}		
	$0.5 \leqq X < 3.0$	U			9.43×10^{-7}	3.42×10^{-7}
	$3.0 \leqq X < 8.0$	U	1.25×10^{-6}	3.04×10^{-7}		
	$8.0 \leqq X < 16.0$	U	1.18×10^{-6}	3.80×10^{-7}		
	$16.0 \leqq X < 500.0$	U	9.85×10^{-7}	1.90×10^{-7}		
ALOG	$0.5 \leqq X \leqq 1.5$	U			6.85×10^{-8}	2.33×10^{-8}
	$X < 0.5, X > 1.5$	E	8.32×10^{-7}	1.19×10^{-7}		
ALOG 10	$0.5 \leqq X \leqq 1.5$	U			7.13×10^{-8}	2.26×10^{-8}
	$X < 0.5, X > 1.5$	E	1.05×10^{-6}	2.17×10^{-7}		
ARCOS	$-1 \leqq X \leqq +1$	U	8.85×10^{-7}	3.19×10^{-7}		
ARSIN	$-1 \leqq X \leqq +1$	U	9.34×10^{-7}	2.06×10^{-7}		
ATAN	The full range	Note 7	1.01×10^{-6}	4.68×10^{-7}		
ATAN 2	The full range	Note 7	1.01×10^{-6}	4.68×10^{-7}		
CABS	The full range	Note 1	9.15×10^{-7}	2.00×10^{-7}		
CCOS	$\lvert X_1\rvert \leqq 10, \lvert X_2\rvert \leqq 1$	U	2.50×10^{-6} See Note 2	7.66×10^{-7}		
CDABS	The full range	Note 1	2.03×10^{-16}	4.83×10^{-17}		
CDCOS	$\lvert X_1\rvert \leqq 10, \lvert X_2\rvert \leqq 1$	U	3.98×10^{-15} See Note 3	2.50×10^{-16}		
CDEXP	$\lvert X_1\rvert \leqq 1, \lvert X_2\rvert \leqq \pi/2$	U	3.76×10^{-16}	1.10×10^{-16}		
	$\lvert X_1\rvert \leqq 20, \lvert X_2\rvert \leqq 20$	U	2.74×10^{-15}	9.64×10^{-16}		
CDLOG	The full range except (1 + Oi)	Note 1	2.72×10^{-16}	5.38×10^{-17}		
CDSIN	$\lvert X_1\rvert \leqq 10, \lvert X_2\rvert \leqq 1$	U	2.35×10^{-15} See Note 4	2.25×10^{-16}		
CDSQRT	The full range	Note 1	1.76×10^{-16}	4.06×10^{-17}		
CEXP	$\lvert X_1\rvert \leqq 170, \lvert X_2\rvert \leqq \pi/2$	U	9.93×10^{-7}	2.67×10^{-7}		
	$\lvert X_1\rvert \leqq 170,$ $\pi/2 < \lvert X_2\rvert \leqq 20$	U	1.07×10^{-6}	2.73×10^{-7}		
CLOG	The full range except (1 + Oi)	Note 1	7.15×10^{-7}	1.36×10^{-7}		
COS	$0 \leqq X \leqq \pi$	U			1.19×10^{-7}	4.60×10^{-8}
	$-10 \leqq X < 0,$ $\pi < X \leqq 10$	U			1.28×10^{-7}	4.55×10^{-8}
	$10 < \lvert X\rvert \leqq 100$	U			1.14×10^{-7}	4.60×10^{-8}
COSH	$-5 \leqq X \leqq +5$	U	1.27×10^{-6}	2.63×10^{-7}		
COTAN	$\lvert X\rvert \leqq \pi/4$	U	1.07×10^{-6}	3.58×10^{-7}		
	$\pi/4 < \lvert X\rvert \leqq \pi/2$	U	1.40×10^{-6} (Note 5)	2.56×10^{-7}		
	$\pi/2 < \lvert X\rvert \leqq 10$	U	1.30×10^{-6} (Note 5)	3.11×10^{-7}		
	$10 < \lvert X\rvert \leqq 100$	U	1.49×10^{-6} (Note 5)	3.15×10^{-7}		
NOTES: (See end of table.)						

Table 12. Accuracy Figures (*Continued*)

Entry Name	Argument Range	Sample E/U	Accuracy Figures			
			Relative		Absolute	
			M (ϵ)	σ (ϵ)	M (E)	σ (E)
CSIN	$\lvert X_1\rvert \leqq 10, \lvert X_2\rvert \leqq 1$	U	1.92×10^{-6} See Note 6	7.38×10^{-7}		
CSQRT	The full range	Note 1	7.00×10^{-7}	1.71×10^{-7}		
DARCOS	$\lvert X\rvert \leqq 1$	U	2.07×10^{-16}	7.05×10^{-17}		
DARSIN	$\lvert X\rvert \leqq 1$	U	2.04×10^{-16}	5.15×10^{-17}		
DATAN	The full range	Note 7	2.18×10^{-16}	7.04×10^{-17}		
DATAN2	The full range	Note 7	2.18×10^{-16}	7.04×10^{-17}		
DCOS	$0 \leqq X \leqq \pi$	U			1.79×10^{-16}	6.53×10^{-17}
	$-10 \leqq X < 0$, $\pi < X \leqq 10$	U			1.75×10^{-16}	5.93×10^{-17}
	$10 < X \leqq 100$	U			2.64×10^{-15}	1.01×10^{-15}
DCOSH	$\lvert X\rvert \leqq 5$	U	3.63×10^{-16}	9.05×10^{-17}		
DCOTAN	$\lvert X\rvert \leqq \pi/4$	U	2.46×10^{-16} (Note 5)	8.79×10^{-17}		
	$\pi/4 < \lvert X\rvert \leqq \pi/2$	U	2.78×10^{-13} (Note 5)	8.61×10^{-15}		
	$\pi/2 < \lvert X\rvert \leqq 10$	U	5.40×10^{-13} (Note 5)	1.13×10^{-14}		
	$10 < \lvert X\rvert \leqq 100$	U	8.61×10^{-13} (Note 5)	4.61×10^{-14}		
DERF	$\lvert X\rvert \leqq 1.0$	U	1.89×10^{-16}	2.60×10^{-17}		
	$1.0 < \lvert X\rvert \leqq 2.04$	U	2.87×10^{-17}	9.84×10^{-18}		
	$2.04 < \lvert X\rvert < 6.092$	U	1.39×10^{-17}	8.02×10^{-18}		
DERFC	$-6 < X < 0$	U	2.08×10^{-16}	6.52×10^{-17}		
	$0 \leqq X \leqq 1$	U	1.40×10^{-16}	2.59×10^{-17}		
	$1 < X \leqq 2.04$	U	4.11×10^{-16}	8.86×10^{-17}		
	$2.04 < X < 4$	U	3.26×10^{-16}	8.65×10^{-17}		
	$4 \leqq X < 13.3$	U	3.51×10^{-15}	1.96×10^{-15}		
DEXP	$\lvert X\rvert \leqq 1$	U	2.04×10^{-16}	5.43×10^{-17}		
	$1 < \lvert X\rvert \leqq 20$	U	2.03×10^{-16}	4.87×10^{-17}		
	$20 < \lvert X\rvert \leqq 170$	U	1.97×10^{-16}	4.98×10^{-17}		
DGAMMA	$0 < X < 1$	U	2.14×10^{-16}	7.84×10^{-17}		
	$1 \leqq X \leqq 2$	U	2.52×10^{-17}	6.07×10^{-18}		
	$2 < X < 4$	U	2.21×10^{-16}	8.49×10^{-17}		
	$4 \leqq X < 8$	U	5.05×10^{-16}	1.90×10^{-16}		
	$8 \leqq X < 16$	U	6.02×10^{-15}	1.78×10^{-15}		
	$16 \leqq X < 57$	U	1.16×10^{-14}	4.11×10^{-15}		
DLGAMA	$0 < X \leqq 0.5$	U	2.77×10^{-16}	9.75×10^{-17}		
	$0.5 < X < 3$	U			2.24×10^{-16}	7.77×10^{-17}
	$3 \leqq X < 8$	U	2.89×10^{-16}	8.80×10^{-17}		
	$8 \leqq X < 16$	U	2.86×10^{-16}	8.92×10^{-17}		
	$16 \leqq X < 500$	U	1.99×10^{-16}	3.93×10^{-17}		
DLOG	$0.5 \leqq X \leqq 1.5$	U			4.60×10^{-17}	2.09×10^{-17}
	$X < 0.5, X > 1.5$	E	3.32×10^{-16}	5.52×10^{-17}		
NOTES: (See end of table.)						

Table 12. Accuracy Figures (*Continued*)

Entry Name	Argument Range	Sample E/U	Accuracy Figures			
			Relative		Absolute	
			M (ϵ)	σ (ϵ)	M (E)	σ (E)
DLOG10	$0.5 \leqq X \leqq 1.5$	U			2.73×10^{-17}	1.07×10^{-17}
	$X < 0.5, X > 1.5$	E	3.02×10^{-16}	6.65×10^{-17}		
DSIN	$\|X\| \leqq \pi/2$	U	3.60×10^{-16}	4.82×10^{-17}	7.74×10^{-17}	1.98×10^{-17}
	$\pi/2 < \|X\| \leqq 10$	U			1.64×10^{-16}	6.49×10^{-17}
	$10 < \|X\| \leqq 100$	U			2.68×10^{-15}	1.03×10^{-15}
DSINH	$\|X\| \leqq 0.88137$	U	2.06×10^{-16}	3.74×10^{-17}		
	$0.88137 < \|X\| \leqq 5$	U	3.80×10^{-16}	9.21×10^{-17}		
DSQRT	The full range	E	1.06×10^{-16}	2.16×10^{-17}		
DTAN	$\|X\| \leqq \pi/4$	U	3.41×10^{-16}	6.27×10^{-17}		
	$\pi/4 < \|X\| \leqq \pi/2$	U	1.43×10^{-12} (Note 5)	2.95×10^{-14}		
	$\pi/2 < \|X\| \leqq 10$	U	2.78×10^{-13} (Note 5)	7.23×10^{-15}		
	$10 < \|X\| \leqq 100$	U	3.79×10^{-12} (Note 5)	9.50×10^{-14}		
DTANH	$\|X\| \leqq 0.54931$	U	1.91×10^{-16}	3.86×10^{-17}		
	$0.54931 < \|X\| \leqq 5$	U	1.54×10^{-16}	1.87×10^{-17}		
ERF	$\|X\| \leqq 1.0$	U	8.16×10^{-7}	1.10×10^{-7}		
	$1.0 < \|X\| \leqq 2.04$	U	1.13×10^{-7}	3.70×10^{-8}		
	$2.04 < \|X\| \leqq 3.9192$	U	5.95×10^{-8}	3.41×10^{-8}		
ERFC	$-3.8 < X < 0$	U	9.10×10^{-7}	2.96×10^{-7}		
	$0 \leqq X \leqq 1.0$	U	7.42×10^{-7}	1.27×10^{-7}		
	$1.0 < X \leqq 2.04$	U	1.54×10^{-6}	3.78×10^{-7}		
	$2.04 < X \leqq 4.0$	U	2.28×10^{-6}	3.70×10^{-7}		
	$4.0 < X \leqq 13.3$	U	1.55×10^{-5}	8.57×10^{-6}		
EXP	$\|X\| \leqq 1$	U	4.65×10^{-7}	1.28×10^{-7}		
	$1 < \|X\| \leqq 170$	U	4.42×10^{-7}	1.15×10^{-7}		
GAMMA	$0 < X < 1.0$	U	9.86×10^{-7}	3.66×10^{-7}		
	$1.0 \leqq X \leqq 2.0$	U	1.13×10^{-7}	3.22×10^{-8}		
	$2.0 < X \leqq 4.0$	U	9.47×10^{-7}	3.79×10^{-7}		
	$4.0 < X < 8.0$	U	2.26×10^{-6}	8.32×10^{-7}		
	$8.0 \leqq X \leqq 16.0$	U	2.20×10^{-5}	7.61×10^{-6}		
	$16.0 < X \leqq 57.0$	U	4.62×10^{-5}	1.51×10^{-5}		
SIN	$\|X\| \leqq \pi/2$	U	1.32×10^{-6}	1.82×10^{-7}	1.18×10^{-7}	4.55×10^{-8}
	$\pi/2 < \|X\| \leqq 10$	U			1.15×10^{-7}	4.64×10^{-8}
	$10 < \|X\| \leqq 100$	U			1.28×10^{-7}	4.52×10^{-8}
SINH	$-5 \leqq X \leqq +5$	U	1.26×10^{-6}	2.17×10^{-7}		
SQRT	The full range	E	4.45×10^{-7}	8.43×10^{-8}		

NOTES: (See end of table)

Table 12. Accuracy Figures (*Continued*)

Entry Name	Argument Range	Sample E/U	Accuracy Figures			
			Relative		Absolute	
			M (ϵ)	σ (ϵ)	M (E)	σ (E)
TAN	$\lvert X\rvert \leqq \pi/4$	U	1.71×10^{-6}	2.64×10^{-7}		
	$\pi/4 < \lvert X\rvert \leqq \pi/2$	U	1.05×10^{-6} (Note 5)	3.59×10^{-7}		
	$\pi/2 < \lvert X\rvert \leqq 10$	U	6.49×10^{-6} (Note 5)	3.38×10^{-7}		
	$10 < \lvert X\rvert \leqq 100$	U	1.57×10^{-6} (Note 5)	3.07×10^{-7}		
TANH	$\lvert X\rvert \leqq 0.7$	U	8.48×10^{-7}	1.48×10^{-7}		
	$0.7 < \lvert X\rvert \leqq 5$	U	2.44×10^{-7}	4.23×10^{-8}		

NOTES:

[1] The distribution of sample arguments upon which these statistics are based is exponential radially and is uniform around the origin.

[2] The maximum relative error cited for the CCOS function is based upon a set of 2000 random arguments within the range. In the immediate proximity of the points $\left(n + \frac{1}{2}\right)\pi + 0i$ (where $n = 0, \pm 1, \pm 2, \ldots,$) the relative error can be quite high, although the absolute error is small.

[3] The maximum relative error cited for the CDCOS function is based upon a set of 1500 random arguments within the range. In the immediately proximity of the points $\left(n + \frac{1}{2}\right)\pi + 0i$ (where $n = 0, \pm 1, \pm 2, \ldots,$) the relative error can be quite high, although the absolute error is small.

[4] The maximum relative error cited for the CDSIN function is based upon a set of 1500 random arguments within the range. In the immediate proximity of the points $n\pi + 0i$ (where $n = \pm 1, \pm 2, \ldots,$) the relative error can be quite high, although the absolute error is small.

[5] The figures cited as the maximum relative errors are those encountered in a sample of 2500 random arguments within the respective ranges. See the appropriate section in the chapter "Algorithms" for a description of the behavior of errors when the argument is near a singularity or a zero of the function.

[6] The maximum relative error cited for the CSIN function is based upon a set of 2000 random arguments within the range. In the immediate proximity of the points $n\pi + 0i$ (where $n = \pm 1, \pm 2, \ldots,$) the relative error can be quite high, although the absolute error is small.

[7] The sample arguments were tangents of numbers uniformly distributed between $-\frac{\pi}{2}$ and $+\frac{\pi}{2}$.

Table 13. Average Machine Timings (*Part 1 of 4*)

Entry Name	Argument Range	Sample E/U	Timing Averages in Microseconds for Various Models								
			30	40	50	65	75 (Note 3)	85 (Note 4)		91	195
ALGAMA	$0.5 \leqq X < 3.0$	U	8873	2531	731	178	105	44	33	30	16
	$0 < X < 0.5$	U	8920	2564	751	185	105	42	34	28	17
	$3.0 \leqq X < 8.0$	U	10640	3038	897	218	125	52	37	32	19
	$8.0 \leqq X < 16.0$	U	7193	2178	584	152	92	33	29	24	14
	$16.0 \leqq X < 500.0$	U	7526	2231	611	152	92	34	30	25	15
ALOG	$0.5 \leqq X \leqq 1.5$	U	4021	1191	324	85	45	17	13	10	7
	$X < 0.5, X > 1.5$	E	4108	1238	331	85	45	15	12	10	7
ALOG10	$0.5 \leqq X \leqq 1.5$	U	4384	1324	351	85	45	17	14	11	7
	$X < 0.5, X > 1.5$	E	4384	1324	351	85	45	18	13	11	7
ARCOS	$-1 \leqq X \leqq +1$	U	5020	1624	424	105	65	20	21	16	10
ARSIN	$-1 \leqq X \leqq +1$	U	2620	1571	411	105	65	23	19	15	9
ATAN	The full range	Note 2	4038	1024	304	72	38	17	12	9	5
	Note 5	Note 2	3945	981	285	67	34	16	11	9	
ATAN2	The full range	Note 2	6114	1717	452	106	58	22	17	10	7
CABS	The full range	Note 1	6007	1838	458	116	74	26	23	19	13
CCOS	$\|X_1\| \leqq 10, \|X_2\| \leqq 1$	U	16923	4705	1425	332	191	74	54	61	35
CDABS	The full range	Note 1	17009	4628	774	157	91	34	28	20	14
CDCOS	$\|X_1\| \leqq 10, \|X_2\| \leqq 1$	U	59957	13961	2896	546	313	117	75	63	37
CDEXP	$\|X_1\| \leqq 1, \|X_2\| \leqq \pi/2$	U	51091	11350	2507	479	268	105	62	55	33
	$\|X_1\| \leqq 20, \|X_2\| \leqq 20$	U	51315	11406	2518	479	268	105	61	54	33
CDLOG	The full range (except $(0 + 0i)$)	Note 1	46248	10239	2074	413	235	85	69	46	32
CDSIN	$\|X_1\| \leqq 10, \|X_2\| \leqq 1$	U	60406	13995	2907	546	302	118	71	62	37
CDSQRT	The full range	Note 1	30436	9217	1396	302	168	60	55	35	27
CEXP	$\|X_1\| \leqq 170, \|X_2\| \leqq \pi/2$	U	14720	4046	1250	282	174	65	47	55	32
	$\|X_1\| \leqq 170, \pi/2 < \|X_2\| < 20$	U	15017	4138	1275	291	174	66	47	54	32
CLOG	The full range (except $(0 + 0i)$)	Note 1	12700	3596	1033	257	157	60	49	43	26
COS	$0 \leqq X \leqq \pi$	U	4448	1091	337	78	38	15	9	11	6
	$-10 \leqq X < 0,$ $\pi < X \leqq 10$	U	4529	1111	344	78	38	15	9	11	6
	$10 < \|X\| \leqq 100$	U	4529	1118	344	72	38	15	9	11	6
COSH	$-5 \leqq X \leqq +5$	U	6399	2011	597	145	92	30	23	28	18
COTAN	$\|X\| \leqq \pi/4$	U	5068	1311	391	92	58	19	14	16	8
	$\pi/4 < \|X\| \leqq \pi/2$	U	5214	1378	397	92	58	18	13	14	8
	$\pi/2 < \|X\| \leqq 10$	U	6540	1378	404	92	58	18	13	14	7
	$10 < \|X\| \leqq 100$	U	6540	1378	397	92	58	18	13	14	7
CSIN	$\|X_1\| \leqq 10, \|X_2\| \leqq 1$	U	16894	4680	1417	332	199	75	55	61	35
CSQRT	The full range	Note 1	10446	3555	808	207	124	47	43	33	24
DARCOS	$\|X\| \leqq 1$	U	19127	5011	931	185	98	34	31	19	12
NOTES: (See end of table.)											

Table 13. Average Machine Timings (*Part 2 of 4*)

Entry Name	Argument Range	Sample E/U	Timing Averages in Microseconds for Various Models								
			30	40	50	65	75 (Note 3)	85 (Note 4)		91	195
DARSIN	$\|X\| \leqq 1$	U	19255	5064	937	185	98	35	32	18	12
DATAN	The full range	Note 2	19937	3779	769	145	78	29	24	12	7
	Note 5	Note 2	19881	3745	750	139	74	28	23	12	
DATAN2	The full range	Note 2	20646	4104	858	171	95	35	30	13	9
DCOS	$0 \leqq X \leqq \pi$	U	16541	3406	762	138	72	29	17	12	7
	$-10 \leqq X < 0$, $\pi < X \leqq 10$	U	16541	3419	762	138	78	29	17	12	7
	$10 < X \leqq 100$	U	16541	3406	769	145	72	29	17	13	7
DCOSH	$\|X\| \leqq 5$	U	21010	4931	1031	212	118	43	29	27	17
DCOTAN	$\|X\| \leqq \pi/4$	U	17612	3498	777	145	85	29	19	16	8
	$\pi/4 < \|X\| \leqq \pi/2$	U	17945	3644	817	152	78	28	19	15	8
	$\pi/2 < \|X\| \leqq 10$	U	18284	3651	817	152	85	29	19	15	8
	$10 < \|X\| \leqq 100$	U	18284	3638	817	152	85	29	19	15	8
DERF	$\|X\| \leqq 1.0$	U	21539	4493	1029	178	98	42	22	13	8
	$1.0 < \|X\| \leqq 2.04$	U	29202	6213	1382	238	132	57	29	15	10
	$2.04 < \|X\| < 6.092$	U	40028	9313	1842	365	198	76	59	34	23
DERFC	$-6 < X < 0$	U	35447	7993	1642	312	178	66	48	27	19
	$0 \leqq X \leqq 1$	U	21898	4546	1049	185	105	42	22	13	8
	$1 < X \leqq 2.04$	U	29047	6159	1362	232	125	56	28	16	10
	$2.04 < X < 4$	U	39800	9266	1822	365	192	75	59	34	23
	$4 \leqq X < 13.3$	U	39987	9326	1849	372	205	77	60	35	24
DEXP	$\|X\| \leqq 1$	U	14305	3613	715	145	78	30	18	13	9
	$1 < \|X\| \leqq 20$	U	14305	3639	722	145	78	31	18	13	9
	$20 < \|X\| \leqq 170$	U	14305	3646	722	138	72	30	18	13	9
DGAMMA	$0 < X < 1$	U	25481	5559	1162	212	118	49	29	17	10
	$1 \leqq X \leqq 2$	U	24180	5386	1149	205	112	49	29	18	10
	$2 < X < 4$	U	26290	5719	1249	218	125	55	30	19	11
	$4 \leqq X < 8$	U	30931	6706	1469	265	145	63	36	22	12
	$8 \leqq X < 16$	U	45811	10833	2189	432	238	96	65	48	32
	$16 \leqq X < 57$	U	45936	10899	2195	425	238	96	65	48	32
DLGAMA	$0 < X \leqq 0.5$	U	39586	9533	1922	358	198	83	54	32	20
	$0.5 < X < 3$	U	41884	9326	1929	358	205	83	55	33	20
	$3 \leqq X < 8$	U	43378	10479	2202	412	232	94	58	37	23
	$8 \leqq X < 16$	U	31041	7139	1442	278	152	62	44	31	21
	$16 \leqq X < 500$	U	31622	7226	1469	285	158	62	44	32	21
DLOG	$0.5 \leqq X \leqq 1.5$	U	15934	3806	729	145	78	28	22	11	8
	$X < 0.5, X > 1.5$	E	16391	3886	749	145	72	30	22	11	8
DLOG10	$0.5 \leqq X \leqq 1.5$	U	17237	4079	782	145	78	30	23	12	8
	$X < 0.5, X > 1.5$	E	17698	4166	802	145	85	31	24	12	8

NOTES: (See end of table.)

Table 13. Average Machine Timings (*Part 3 of 4*)

Entry Name	Argument Range	Sample E/U	Timing Averages in Microseconds for Various Models								
			30	40	50	65	75 (Note 3)	85 (Note 4)		91	195
DSIN	$\|X\| \leqq \pi/2$	U	16598	3406	769	138	78	30	16	13	7
	$\pi/2 < \|X\| \leqq 10$	U	16598	3426	769	138	78	30	15	13	7
	$10 < \|X\| \leqq 100$	U	16598	3419	769	145	72	30	15	13	7
DSINH	$\|X\| \leqq 0.88137$	U	11957	2604	564	105	52	23	8	8	4
	$0.88137 < \|X\| \leqq 5$	U	21585	5058	1057	212	118	44	29	26	17
DSQRT	The full range	E	9416	2679	409	92	45	13	16	8	7
DTAN	$\|X\| \leqq \pi/4$	U	17527	3424	751	132	72	29	16	13	7
	$\pi/4 < \|X\| \leqq \pi/2$	U	18587	3671	824	158	85	28	19	13	8
	$\pi/2 < \|X\| \leqq 10$	U	18525	3644	817	152	85	28	18	13	8
	$10 < \|X\| \leqq 100$	U	18525	3624	817	152	85	28	18	13	8
DTANH	$\|X\| \leqq 0.54931$	U	14169	2673	515	92	52	19	15	6	4
	$0.54931 < \|X\| \leqq 5$	U	18432	4593	955	198	105	43	30	23	15
ERF	$\|X\| \leqq 1.0$	U	4151	1124	364	92	45	20	14	11	7
	$1.0 < \|X\| \leqq 2.04$	U	5180	1438	457	112	58	26	16	22	8
	$2.04 < \|X\| \leqq 3.9192$	U	9124	2644	771	185	105	44	34	31	21
ERFC	$-3.8 < X < 0$	U	6924	1984	597	145	78	33	24	22	14
	$0 \leqq X \leqq 1.0$	U	4215	1158	371	92	45	21	14	12	7
	$1.0 < X \leqq 2.04$	U	5059	1411	444	105	58	25	16	14	8
	$2.04 < X \leqq 4.0$	U	9020	2624	757	178	105	43	34	32	21
	$4.0 < X \leqq 13.3$	U	9300	2684	791	178	112	43	35	34	21
EXP	$\|X\| \leqq 1$	U	4129	1351	384	85	52	16	14	16	11
	$1 < \|X\| \leqq 170$	U	4150	1338	384	85	52	16	14	16	11
GAMMA	$0 < X < 1.0$	U	4633	1278	384	92	52	22	17	12	7
	$1.0 \leqq X \leqq 2.0$	U	4493	1251	391	92	52	23	16	18	7
	$2.0 < X \leqq 4.0$	U	5147	1424	444	105	58	27	19	16	8
	$4.0 < X < 8.0$	U	6694	1824	571	138	78	34	24	18	10
	$8.0 \leqq X \leqq 16.0$	U	11729	3571	997	245	145	53	45	41	28
	$16.0 < X \leqq 57.0$	U	11729	3591	997	238	152	54	45	42	28
SIN	$\|X\| \leqq \pi/2$	U	4383	1078	337	78	38	16	9	12	6
	$\pi/2 < \|X\| \leqq 10$	U	4560	1124	351	85	45	16	10	12	6
	$10 < \|X\| \leqq 100$	U	4560	1124	351	78	45	16	9	12	7
SINH	$-5 \leqq X \leqq +5$	U	5803	1764	537	125	72	24	22	24	15
SQRT	The full range	E	3162	1051	224	58	38	10	10	9	6
TAN	$\|X\| \leqq \pi/4$	U	2959	1258	364	85	52	17	12	14	7
	$\pi/4 < \|X\| \leqq \pi/2$	U	5378	1391	404	98	58	18	14	13	7
	$\pi/2 < \|X\| \leqq 10$	U	5205	1364	391	92	52	18	13	12	7
	$10 < \|X\| \leqq 100$	U	5205	1358	397	98	58	18	13	12	7

NOTES: (See end of table.)

Table 13. Average Machine Timings (*Part 4 of 4*)

Entry Name	Argument Range	Sample E/U	Timing Averages in Microseconds for Various Models								
			30	40	50	65	75 (Note 3)	85 (Note 4)		91	195
TANH	$\|X\| \leqq 0.7$	U	2658	598	177	45	25	10	6	5	3
	$0.7 < \|X\| \leqq 5$	U	6092	1911	564	132	78	27	22	23	17

NOTES:

[1] The distribution of sample arguments upon which these statistics are based is exponential radially and is uniform around the origin.

[2] The sample arguments were tangents of numbers uniformly distributed between $-\frac{\pi}{2}$ and $+\frac{\pi}{2}$.

[3] The statistics for the Model 75 are based upon four-way interleaving.

[4] The second column of speeds for the Model 85 applies to that machine with high-speed multiply feature.

[5] Timing figures on this line apply to the version for Basic FORTRAN IV (OS).

Appendix A: Assembler Language Information

The mathematical and service subprograms in the FORTRAN IV library can be used by the assembler language programmer. Successful use depends on three things: (1) making the library available to the linkage editor; (2) setting up proper calling sequences, based on either a call macro instruction or a branch; and (3) supplying correct parameters—*i.e.*, arguments.

Library Availability

The System/360 Operating System FORTRAN IV library is a partitioned data set named SYS1.FORTLIB. The assembler language programmer must arrange for the desired subprograms (modules) to be taken from this library and brought into main storage, usually as a part of his load module. This can be done by employing the techniques described in the publication *IBM System/360 Operating System: Linkage Editor and Loader*, Form C28-6538.

For example, the FORTRAN IV library could be made part of the automatic call library by using these job control statements:

```
//jobname       JOB     desired operands
//stepname      EXEC    ASMFCLG, PARM.LKED
                        ='XREF, LIST,
                        MAP'
.//ASM.SYSIN    DD      *
        (assembler language program source deck)
/*
//LKED.SYSLIB   DD      DSNAME=SYS1.FORTLIB,
                        DISP=SHR
/*
```

Subprograms requested in the source program would then be available to the linkage editor for inclusion in the load module.

Calling Sequences

Two general methods of calling are possible: (1) coding an appropriate macro instruction (see the publication *IBM System/360 Operating System: Supervisor and Data Management Macro Instructions*, Form C28-6647), such as CALL; or (2) coding assembler language branch instructions.

In all cases, a save area must be provided that:

- is aligned on a fullword boundary
- is at least as large as the size specified in Tables 14, 15, and 16, but preferably the standard 18 words to ensure future compatibility
- has its address in general register 13 at the time of the call macro instruction or branch

NOTE: FORTRAN subprograms use certain floating-point registers (see Table 14), but do not save and restore original register contents. If the programmer wishes floating-point information retained, he must save it himself before calling the subprogram.

If the called subprogram is one that uses FORTRAN input/output, error, or interruption routines (see Table 17), the calling program must include the following two instructions before the branch is made:

```
L     15, = V (IBCOM#)
BAL   14, 64 (15)
```

These instructions cause a branch into the IBCOM subprogram, which initializes return coding and prepares routines to handle interruptions. If this initialization is omitted, an interruption or error may cause abnormal termination. (After initialization, IBCOM returns to the intruction following the BAL.)

NOTE: When these instructions are included, the occurrence of a decimal divide exception within the assembler language program will cause the character B to appear in the program interruption message.

When a branch instruction rather than a call macro instruction is used to invoke a subprogram, several additional conventions must be observed:

- An argument (parameter) list must be assembled on a fullword boundary. It consists of one 4-byte address constant for each argument, with the last address constant containing a 1 in its high order bit.
- The address of this argument list must be in general register 1.
- The address of the entry name of the called subprogram must be in general register 15.
- The address of the point of return to the calling program must be in general register 14.

The total requirements for an assembler language calling sequence are illustrated in Figure 1.

Supplying Correct Parameters

Arguments must be of the proper type, length, and quantity, and, in certain cases, within a specified range for the subprogram called.

For mathematical subprograms, this information can be found in Tables 2 through 6. INTEGER*4 denotes a signed binary number four bytes long. REAL*4 and REAL*8 are normalized floating-point numbers, 4 and 8 bytes long, respectively. COMPLEX*8 and COMPLEX*16 are complex numbers, 8 and 16 bytes long, respectively, whose left half contains the real part, and right half

```
         L      15, = V(IBCOM#)      These two statements are necessary if the called subprogram uses FORTRAN
         BAL    14, 64(15)           input/output, error, or interrupt routines (see Appendix B).
         * *    * *

         LA     13,area              General register 13 contains the address of the save area.
         LA     1,arglist            General register 1 contains the address of the argument list.
         L      15,entry             General register 15 contains the address of the subprogram.
         BALR   14,15                General register 14 contains the address of the point of return to the calling
                                     program.
         NOP    X'id'                This statement is optional. The id represents an identification number.
                                     This number is supplied by the programmer and may be any hexadecimal
         * *    * *                  integer less than 2^16 - 1.

entry    DC     V (entry name)
              or
entry    DC     A (entry name)       NOTE: In this case, the entry name must be defined by an EXTRN instruc-
         * *    * *                        tion to obtain proper linkage.

area     DS     xxF                  This statement defines the save area needed by the subprogram. The xx
                                     represents the minimum size of the save area required; however, the pro-
                                     grammer is advised to use a save area of 18 fullwords for all subprograms.
                                     (The minimum save area requirements are given in Tables 14 and 15
                                     for the mathematical subprograms and in Table 16 for the service sub-
         * *    * *                  programs.)

For one argument
         CNOP                        Aligns the argument list at a fullword boundary.
arglist  DC     X'80'                Places a 1 in the high order bit of the only argument.
         DC     AL3 (arg)            Contains the address of the argument.

For more than one argument
         CNOP                        Aligns the argument list at a fullword boundary.
arglist  DC     A (arg1)             Contains the address of the first argument.
         DC     A (arg2)             Contains the address of the second argument.
                 .
                 .
                 .
         DC     X'80'                Places a 1 in the high order bit of the last argument.
         DC     AL3 (argn)           Contains the address of the last argument.
```

Figure 1. General Assembler Language Calling Sequence

contains the imaginary part. Each part is a normalized floating-point number. Four-byte argument types must be aligned on fullword boundaries, and 8-byte and 16-byte types must be aligned on double-word boundaries.

Argument information for nonmathematical subprograms can be found in "Service Subprograms."

Error messages resulting from incorrect arguments are explained in "Appendix C: Interruption and Error Procedures."

Results

Each mathematical subprogram returns a single answer of a type listed in Tables 2 through 6 (see "Function Value Type"). The INTEGER answers are returned in general register 0, REAL answers are returned in floating-point register 0, and COMPLEX answers are returned in floating-point register 0 and 2. Result registers are listed by subprogram entry name in Table 14.

The location and form of the service subroutine results can be determined from the discussion in "Service Subprograms."

Example

To find the square root of the value in AMNT, the library square root subprogram (entry name SQRT) could be invoked, using the following statements:

```
         L      15, = V (IBCOM#)
         BAL    14,64(15)
                 .
                 .
         LA     13,SAVE
         CALL   SQRT,(AMNT),VL
                 .
                 .
SAVE     DS     18F
                 .
                 .
AMNT     DC     E'144'
```

Table 14. Assembler Information for the Explicitly Called Mathematical Subprograms

Entry Name(s)	Save Area (Fullwords)	Registers Used[1]	
		Result	Intermediate
AINT	9	0	2, 4, 6
ALGAMA, GAMMA	9	0	2, 4, 6
ALOG, ALOG10	7	0	2, 4, 6
AMAX0, AMIN0	6	0	
MAX0, MIN0	9	0*	
AMAX1, AMIN1	6	0	
MAX1, MIN1	9	0*	
AMOD, DMOD	9	0	2, 4, 6
ARCOS, ARSIN	10	0	2, 4
ATAN	5	0	2, 4, 6
ATAN, ATAN2	7	0	2, 4, 6
CABS	7	0, 2	4, 6
CCOS, CSIN	9	0, 2	4
CDABS	7	0, 2	4, 6
CDCOS, CDSIN	9	0, 2	4
CDEXP	8	0, 2	4, 6
CDLOG	8	0, 2	4, 6
CDSQRT	9	0, 2	4, 6
CEXP	8	0, 2	4, 6
CLOG	8	0, 2	4, 6
COS, SIN	7	0	2, 4
COSH, SINH	8	0	2, 4
COTAN, TAN	7	0	2, 4
CSQRT	9	0, 2	4, 6
DARCOS, DARSIN	13	0	2, 4
DATAN	5	0	2, 4, 6
DATAN, DATAN2	7	0	2, 4, 6
DCOS, DSIN	7	0	2, 4
DCOSH, DSINH	8	0	2, 4
DCOTAN, DTAN	7	0	2, 4, 6
DERF, DERFC	11	0	2, 4, 6
DEXP	7	0	2
DGAMMA, DLGAMA	11	0	2, 4, 6
DLOG, DLOG10	9	0	2, 4, 6
DMAX1, DMIN1	9	0	
DSQRT	7	0	2, 4
DTANH	5	0	2, 4, 6
EXP	11	0	
ERF, ERFC	11	0	2, 4, 6
IDINT, INT, IFIX	9	0*	
MOD	9	0*	
SQRT	7	0	2
TANH	5	0	2, 4, 6

[1]Floating-point; asterisk indicates general.

Table 15. Assembler Information for the Implicitly Called Mathematical Subprograms

Entry Name(s)	Save Area (Fullwords)	Registers Used[1]	
		Result	Intermediate
CDMPY#, CDDVD#	5	0, 2	4, 6
CDVD#, CMPY#	5	0, 2	4, 6
FIXPI#	18	0*	
FRXPI#	18	0	
FDXPI#	18	0	
FRXPR#	18	0	
FDXPD#	18	0	
FCDXI#	18	0, 2	
FCXPI#	18	0, 2	

[1]Floating-point; asterisk indicates general.

(The VL operand in CALL indicates the end of the parameter list.)

Employing only assembler language instructions, the sequence would be:

```
                L       15,=V(IBCOM#)
                BAL     14,64(15)
                .
                .
                .
                LA      13,SAVE
                LA      1,ARG
                L       15,ENTRY
                BALR    14,15
                .
                .
                .
ENTRY           DC      V(SQRT)
                .
                .
                .
SAVE            DS      18F
                .
                .
                .
ARG             DC      X'80'
                DC      AL3(AMNT)
                .
                .
                .
AMNT            DC      E'144'
```

Note that, in both cases, a branch to IBCOM is provided, a save area is set up, and AMNT meets argument specifications by being a four-byte normalized floating-point number, ≥ 0, aligned on a fullword boundary.

In both cases, the answer (12, in this instance) is returned in floating-point register 0 as a 4-byte floating-point number.

Space Considerations

Many of the mathematical subprograms require other mathematical subprograms for their calculations. In addition, most of the subprograms use the input/output, error, and interruption library subroutines. (This interdependence is outlined in "Appendix B: Storage Estimates.") Thus, although the programmer may request just one FORTRAN subprogram, the requirements of that subprogram may make the resultant load module quite large. The SQRT routine, for example, takes only 344 bytes of storage itself, but requires other subroutines that increase the load module size by approximately 20,000 bytes.

Table 16. Assembler Information for the Service Subprograms

Entry Name(s)	Save Area (Fullwords)
DUMP, PDUMP	18
DVCHK	10
EXIT	5
OVERFL	10
SLITE, SLITET	10

Appendix B: Storage Estimates

Appendix B contains decimal storage estimates (in bytes) for the library subprograms. The estimate given does not include any additional mathematical subprograms or FORTRAN routines that the subprograms use during execution. The entry names of any additional mathetical library subprograms used are given in Table 17. Tables 17 and 18 also indicate which mathematical and service subprograms require FORTRAN routines for input/output, interruption, and error procedures.

Table 17. Mathematical Subprogram Storage Estimates

Entry Name(s)	Decimal Estimates	Additional Mathematical Subprograms Used	I/O, Error & Interrupt Routines
AINT	80		No
ALGAMA, GAMMA	848	ALOG, EXP	Yes
ALOG, ALOG10	464		Yes
AMAX0, AMIN0, MAX0, MIN0	224		No
AMAX1, AMIN1, MAX1, MIN1	224		No
AMOD, DMOD	120		No
ARCOS, ARSIN	496	SQRT	Yes
ATAN	200		No
ATAN, ATAN2	488		Yes
CABS	192	SQRT	Yes
CCOS, CSIN	760	EXP, SIN/COS	Yes
CDABS	200	DSQRT	Yes
CDCOS, CDSIN	832	DEXP, DSIN/DCOS	Yes
CDDVD#, CDMPY#	240		No
CDEXP	624	DEXP, DSIN/DCOS	Yes
CDLOG	488	DLOG, DATAN2	Yes
CDSQRT	328	DSQRT	Yes
CDVD#, CMPY#	216		No
CEXP	592	EXP, SIN/COS	Yes
CLOG	464	ALOG, ATAN2	Yes
COS, SIN	504		Yes
COSH, SINH	504	EXP	Yes
COTAN, TAN	648		Yes
CSQRT	312	SQRT	Yes
DARCOS, DARSIN	648	DSQRT	Yes
DATAN	312		No
DATAN, DATAN2	648		Yes
DCOS, DSIN	696		Yes
DCOSH, DSINH	592	DEXP	Yes
DCOTAN, DTAN	760		Yes
DERF, DERFC	808	DEXP	Yes
DEXP	704		Yes
DGAMMA, DLGAMA	1056	DLOG, DEXP	Yes
DLOG, DLOG10	538		Yes
DMAX1, DMIN1	120		No
DSQRT	352		Yes
DTANH	304	DEXP	Yes
EXP	424		Yes
ERF, ERFC	520	EXP	Yes
FCDXPI#	560	CDMPY#/CDDVD#	Yes
FCXPI#	536	CDVD#/CMPY#	Yes
FDXPD#	464	DEXP, DLOG	Yes
FDXPI#	368		Yes
FRXPR#	432	EXP, ALOG	Yes
FIXPI#	368		Yes
FRXPI#	360		Yes
IDINT, INT	136		No
MOD	56		No
SQRT	344		Yes
TANH	256	EXP	Yes

The programmer must add the estimates for all subprograms and routines needed to determine the amount of storage required. If the programmer has not made allowances for the storage required by any of these additional routines (see Table 19), the amount of available storage may be exceeded and execution cannot begin.

System/360 Operating System

The names of execution-time routines sometimes vary according to whether or not the Extended Error Handling facility is in effect. In the following discussion, the names that are used are those when the facility has not been specified. Table 19 presents a cross listing of names for both circumstances. A full discussion of extended error handling is given in the *FORTRAN IV (G and H) Programmer's Guide* listed in the Preface.

The IHCFIOSH routine performs input/output procedures for both Basic FORTRAN IV and FORTRAN IV. [This routine refers to a table (IHCUATBL) for information about the input/output devices used during execution.] The IHCFCOME routine performs interruption and error procedures for Basic FORTRAN IV library subprograms; the IHCFCOMH, IHCFCVTH, IHCFINTH, IHCTRCH, and IHCUOPT routines perform the procedures for FORTRAN IV library subprograms. If a system contains both basic and full FORTRAN IV compilers, the IHCFCOMH-IHCFCVTH routines are used. Tables 17 and 18 indicate which library subprograms require these execution-time routines.

Table 18. Service Subprogram Storage Estimates

Entry Name(s)	Decimal Estimates	I/O, Error, & Interrupt Routines
DUMP/PDUMP	544	Yes
DVCHK	80	Yes
EXIT	32	Yes
OVERFL	88	Yes
SLITE/SLITET	392	Yes

Table 19. OS Execution-Time Routines Storage Estimates

Routine Name	Decimal Estimate	Used by	Extended Error Handling Facility	
			Routine Name	Decimal Estimate
IHCADJST	1,156	FORTRAN IV	Same	Same
IHCDBUG	2,152	FORTRAN IV	Same	Same
IHCFCOME	6,196	Basic FORTRAN IV	Not applicable	
IHCFCOMH	4,218	FORTRAN IV	IHCECOMH	5,368
IHCCOMH2	520	FORTRAN IV	Same	1,120
IHCDIOSE	2,688 (See Note 1)	Both	IHCEDIOS	3,856 (See Note 1)
IHCFIOSH	3,744 + IHCUATBL (See Notes 2 and 3)	Both	IHCEFIOS	4,976 + IHCUATBL (See Notes 2 and 3)
IHCFINTH	926	FORTRAN IV	IHCEFNTH	1,368
IHCFCVTH	4,736	FORTRAN IV	Same	Same
IHCCGOTO	64	Basic FORTRAN IV	Not applicable	
IHCIBERH	224	FORTRAN IV	Same	Same
IHCIBERR	136	Basic FORTRAN IV	Not applicable	
IHCNAMEL	2,880	FORTRAN IV	Same	Same
IHCSTAE	450	FORTRAN IV	Same	Same
IHCTRCH	792	FORTRAN IV	IHCETRCH	706
IHCUOPT	8	FORTRAN IV	Same	800 + 8n (See Note 4)
	Not applicable	FORTRAN IV	IHCERRM	1,552
	Not applicable	FORTRAN IV	IHCFOPT	824

NOTES:

1. This module also acquires dynamic storage. Its amount, in bytes, may be computed by the formula 184 + buffer size. Each buffer value should be 800, except for a card read punch which is 80 and a printer which is 133. FORTRAN utilizes double buffering.
2. This module also acquires dynamic storage. Its amount, in bytes, may be computed by the formula 128 + buffer size. Buffer lengths are listed in Note 1.
3. The number of bytes in table IHCUATBL may be computed by the formula 12n + 8, where *n* is the number of data set reference numbers requested during system generation.
4. The number of additional entries supplied in the Option Table during system generation is represented by *n*.

In addition, several other execution-time routines may be needed to resolve external references in a FORTRAN IV object module.

1. If a source module specifies direct-access input/output operations, the compiler generates a call to the IHCDIOSE routine.
2. At the point that errors are encountered during compilation, the compiler generates a call to an error routine (IHCIBERR for Basic FORTRAN IV and IHCIBERH for FORTRAN IV). If execution of the load module is attempted, the error routine is called, a message is issued, and the execution is terminated.
3. If a Basic FORTRAN IV source module contains a computed GO TO, the compiler generates a call to the IHCCGOTO routine.
4. If a FORTRAN IV source module contains any input/output operations that refer to a NAMELIST name, compiler generates a call to the IHCNAMEL routine.
5. If a FORTRAN IV source module uses the debug facility, the compiler generates a call to the IHCDBUG routine.
6. If boundary alignment was specified during system generation, the IHCADJST routine will be loaded if a boundary-alignment error occurs.

Appendix C: Interruption and Error Procedures

This appendix contains brief explanations of the program interruption and error procedures used by the FORTRAN library. A full description of program interrupts is given in the publication *IBM System/360 Principles of Operation*, Form A22-6821. The programmer's guides listed in the Preface give detailed information about error processing and message formats.

Interruption Procedures

The FORTRAN library processes those interrupts that are described below; all others are handled directly by the system Supervisor:

1. When an interrupt occurs, indicators are set to record exponent overflow, underflow, fixed-point, floating-point or decimal divide exceptions. These indicators can be interrogated dynamically by the subprograms described in the chapter, "Service Subprograms."
2. A message is printed on the object error unit when each interrupt occurs. The old program status word (PSW) printed in the message indicates the cause of each interrupt.
3. Result registers are changed when exponent overflow or exponent underflow (codes C and D) occur. (For a description of the format of floating-point numbers, see the publication *IBM System/360 Principles of Operation*, Form A22-6821.) Result registers are also set when a floating-point instruction is referenced by an assembler language execute (EXEC) instruction.
4. Condition codes set by floating-point addition or substraction instructions are altered for exponent underflow (code D).
5. After the foregoing services are performed, execution of the program continues from the instruction following the one that caused the interrupt.

Error Procedures

During execution, the mathematical subprograms assume that the argument(s) is the correct type. No checking is done for erroneous arguments (i.e., the wrong type, invalid characters, the wrong length, etc.); therefore, a computation performed with an erroneous argument has an unpredictable result. However, the nature of some mathematical functions requires that the input be within a certain range. For example, the square root of a negative number is not permitted. If the argument is not within the valid range given in Tables 2 through 6, an error message is written on the object error unit data set defined by the installation during system generation. The execution of this load module or phase is terminated and control is returned to the operating system. However, execution can continue, (in FORTRAN IV (OS) only), if extended error handling was selected during system generation. This facility provides for standard corrective action by the user. For a full description of extended error message handling, see the publication *IBM System/360 FORTRAN IV (G and H) Programmer's Guide*, Form C28-6817.

Table 20 lists the error messages in numeric order, gives the entry name(s) associated with that number and an explanation of the error. The programmer's guides listed in the Preface of this publication show the text and format of the message as it appears in the error listing. In the following explanations, *x* represents the argument supplied by the programmer.

Table 20. Error Messages

Error Code	Entry Name(s)	Explanation
216	SLITE, SLITET	An invalid sense-light number was detected in the argument list of a call to the SLITE or SLITET subprogram.
241	FIXPI#	In the FIXPI# subprogram, a base number of zero and an exponent less than or equal to zero is an error.
242	FRXPI#	In the FRXPI# subprogram, a base number of zero and an exponent less than or equal to zero is in error.
243	FDXPI#	In the FDXPI# subprogram, a base number of zero and an exponent less than or equal to zero is an error.
244	FRXPR#	In the FRXPR# subprogram, a base number of zero and an exponent less than or equal to zero is an error.
245	FDXPD#	In the FDXPD# subprogram, a base number of zero and an exponent less than or equal to zero is an error.
246	FCXPI#	In the FCXPI# subprogram, a base number of zero and an exponent less than or equal to zero is an error.
247	FCDXI#	In the FCDXI# subprogram, a base number of zero and an exponent less than or equal to zero is an error.
251	SQRT	In the SQRT subprogram, the value of the argument is less than zero.
252	EXP	In the EXP subprogram, the value of the argument is greater than 174.673.
253	ALOG, ALOG10	In the ALOG/ALOG10 subprogram, the value of the argument is less than or equal to zero. Because this subprogram is called by an exponential subprogram, this message also indicates that an attempt has been made to raise a negative base to a real power.
254	SIN, COS	In the SIN/COS subprogram, the absolute value of an argument is greater than or equal to $2^{18}\pi$.
255	ATAN2	When the entry name ATAN2 is used, the value of both arguments cannot be zero.
256	SINH, COSH	In the subprogram SINH/COSH, the value of the argument is greater than or equal to 175.366.
257	ARSIN, ARCOS	In the ARSIN/ARCOS subprogram, the absolute value of the argument is greater than one.
258	TAN, COTAN	In the TAN/COTAN subprogram, the absolute value of the argument is greater than or equal to $2^{18}\pi$.
259	TAN, COTAN	In the TAN/COTAN subprogram, the value of the argument is too close to one of the singularities ($\pm \pi/2, \pm 3\pi/2, \ldots$ for the tangent; $\pm \pi, \pm 2\pi, \ldots$ for the cotangent).
261	DSQRT	In the DSQRT subprogram, the value of the argument is less than zero.
262	DEXP	In the DEXP subprogram, the value of the argument is greater than 174.673.
263	DLOG, DLOG10	In the DLOG/DLOG10 subprogram, the value of the argument is less than or equal to zero. Because the subprogram is called by an exponential subprogram, this message also indicates that an attempt has been made to raise a negative base to a real power.
264	DSIN, DCOS	In the DSIN/DCOS subprogram, the absolute value of the argument is greater than or equal to $2^{50}\pi$.
265	DATAN2	When the entry name DATAN2 is used, the value of both arguments cannot be zero.
266	DSINH, DCOSH	In the DSINH/DCOSH subprogram, the absolute value of the argument is greater than or equal to 175.366.
267	DARSIN, DARCOS	In the DARSIN/DARCOS subprogram, the absolute value of the argument is greater than one.

Table 20. Error Messages (*Continued*)

Error Code	Entry Name(s)	Explanation
268	DTAN, DCOTAN	In the DTAN/DCOTAN subprogram, the absolute value of the argument is greater than or equal to $2^{50}\pi$.
269	DTAN, DCOTAN	In the DTAN/DCOTAN subprogram, the value of the argument is too close to one of the singularities ($\pm \pi/2, \pm 3\pi/2, \ldots$ for the tangent; $\pm \pi, \pm 2\pi, \ldots$ for the cotangent).
271	CEXP	In the CEXP subprogram, the absolute value of the real part of the argument is greater than 174.673.
272	CEXP	In the CEXP subprogram, the absolute value of the imaginary part of the argument is greater than or equal to $2^{18}\pi$.
273	CLOG	In the CLOG subprogram, the value of both the real and imaginary parts of the argument cannot be zero.
274	CSIN, CCOS	In the CSIN/CCOS subprogram, the absolute value of the real part of the argument is greater than or equal to $2^{18}\pi$.
275	CSIN, CCOS	In the CSIN/CCOS subprogram, the absolute value of the imaginary part of the argument is greater than 174.673.
281	CDEXP	In the CDEXP subprogram, the value of the real part of the argument is greater than 174.673.
282	CDEXP	In the CDEXP subprogram, the absolute value of the imaginary part of the argument is greater than or equal to $2^{50}\pi$.
283	CDLOG	In the CDLOG subprogram, the value of both the real and imaginary parts of the argument cannot be zero.
284	CDSIN, CDCOS	In the CDSIN/CDCOS subprogram, the absolute value of the real part of the argument is greater than or equal to $2^{50}\pi$.
285	CDSIN, CDCOS	In the CDSIN/CDCOS subprogram, the absolute value of the imaginary part of the argument is greater than 174.673.
290	GAMMA	In the GAMMA subprogram, the value of the argument is outside the valid range of $2^{-852} < X < 57.5744$.
291	ALGAMA	In the ALGAMA subprogram, the value of the argument is outside the valid range of $0 < X < 4.2937 \cdot 10^{78}$.
300	DGAMMA	In the DGAMMA subprogram, the value of the argument is outside the valid range of $2^{-852} < X < 57.5744$.
301	DLGAMA	In the DLGAMA subprogram, the value of the argument is outside the valid range of $0 < X < 4.2937 \cdot 10^{78}$.

Appendix D: Module Names

Table 21. Mathematical Subprogram Module Names (*Part 1 of 2*)

Entry Name	Module Name	
	FORTRAN IV (G & H)	Basic FORTRAN IV
AINT	-------	IHCFAINT
ALGAMA	IHCSGAMA	-------
ALOG	IHCSLOG	IHCSLOG
ALOG10	IHCSLOG	IHCSLOG
AMAX0	IHCFMAXI	IHCFMAXI
AMAX1	IHCFMAXR	IHCFMAXR
AMIN0	IHCFMAXI	IHCFMAXI
AMIN1	IHCFMAXR	IHCFMAXR
AMOD	-------	AMOD
ARCOS	IHCSASCN	-------
ARSIN	IHCSASCN	-------
ATAN	IHCSATN2	IHCSATAN
ATAN2	IHCSATN2	-------
CABS	IHCCSABS	-------
CCOS	IHCCSSCN	-------
CDABS	IHCCLABS	-------
CDCOS	IHCCLSCN	-------
CDDVD#	IHCCLAS	-------
CDEXP	IHCCLEXP	-------
CDLOG	IHCCLLOG	-------
CDMPY#	IHCCLAS	-------
CDSIN	IHCCLSCN	-------
CDSQRT	IHCCLSQT	-------
CDVD#	IHCCSAS	-------
CEXP	IHCCSEXP	-------
CLOG	IHCCSLOG	-------
CMPY#	IHCCSAS	-------
COS	IHCSSCN	IHCSSCN
COSH	IHCSSCNH	-------
COTAN	IHCSTNCT	-------
CSIN	IHCLSCN	IHCLSCN
CSQRT	IHCCSSQT	-------
DARCOS	IHCLASCN	-------
DARSIN	IHCLASCN	-------
DATAN	IHCLATN2	IHCLATAN
DATAN2	IHCLATN2	-------
DCOS	IHCLSCN	IHCLSCN
DCOSH	IHCLSCNH	-------
DCOTAN	IHCLTNCT	-------

• Table 21. Mathematical Subprogram Module Names (*Part 2 of 2*)

Entry Name	Module Name	
	FORTRAN IV (G & H)	Basic FORTRAN IV
DERF	IHCLERF	-------
DERFC	IHCLERF	-------
DEXP	IHCLEXP	IHCLEXP
DGAMMA	IHCLGAMA	-------
DLGAMA	IHCLGAMA	-------
DLOG	IHCLLOG	IHCLLOG
DLOG10	IHCLLOG	IHCLLOG
DMAX1	IHCFMAXD	IHCFMAXD
DMIN1	IHCFMAXD	IHCFMAXD
DMOD	-------	IHCFMODR
DSIN	IHCLSCN	IHCLSCN
DSINH	IHCLSCNH	-------
DSQRT	IHCLSQRT	IHCLSQRT
DTAN	IHCLTNCT	-------
DTANH	IHCLTANH	IHCLTANH
EXP	IHCSEXP	IHCSEXP
ERF	IHCSERF	-------
ERFC	IHCSERF	-------
FCDXI#	IHCFCDXI	-------
FCXPI#	IHCFCXPI	-------
FDXPD#	IHCFDXPD	IHCFDXPD
FDXPI#	IHCFDXPI	IHCFDXPI
FIXPI#	IHCFIXPI	IHCFIXPI
FRXPI#	IHCFRXPI	IHCFRXPI
FRXPR#	IHCFRXPR	IHCFRXPR
GAMMA	IHCSGAMA	-------
IDINT	-------	IHCIFIX
INT	-------	IHCIFIX
MAX0	IHCFMAXI	IHCFMAXI
MAX1	IHCFMAXR	IHCFMAXR
MIN0	IHCFMAXI	IHCFMAXI
MIN1	IHCFMAXR	IHCFMAXR
MOD	-------	IHCFMODI
SIN	IHCSSCN	IHCSSCN
SINH	IHCSSCNH	-------
SQRT	IHCSSQRT	IHCSSQRT
TAN	IHCSTNCT	-------
TANH	IHCSTANH	IHCSTANH

Appendix E: Sample Storage Printouts

A sample printout is given below for each dump format that can be specified for the storage dump subprogram. The printouts are given in the following order: hexadecimal, LOGICAL*1, LOGICAL*4, INTEGER*2, INTEGER*4, REAL*4, REAL*8, COMPLEX*8, COMPLEX*16, and literal (see Figure 2). Note that the headings on the printouts are not generated by the system, but were obtained by using FORMAT statements. The number printed at the left of each output line is the storage location (in hexadecimal) of the first data item tabulated.

The output of the storage dump subprogram (for both entry names, DUMP and PDUMP) is placed on the object error unit data set defined by the installation during system generation.

```
CALL PDUMP WITH HEXADECIMAL FORMAT SPECIFIED
00A3E0     485F5E10  00000000  485F5E10  10000000  42100000

006DC8     42800000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
006DF8     C0000000  00000000  41200000  41566666  0000000C  41100000

CALL PDUMP WITH LOGICAL*1 FORMAT SPECIFIED
006E1E     T  F

CALL PDUMP WITH LOGICAL*4 FORMAT SPECIFIED
006E10     F  T

CALL PDUMP WITH INTEGER*2 FORMAT SPECIFIED
006E18       10
006E1A     -100
006E1C       10

CALL PDUMP WITH INTEGER*4 FORMAT SPECIFIED
006E20            1         2         3         4         5         6         7         8         9        10
006E48           11        12

CALL PDUMP WITH REAL*4 FORMAT SPECIFIED
006E00     0.20000000E 01   0.53999996E 01

CALL PDUMP WITH REAL*8 FORMAT SPECIFIED
006DC8     0.17599999999999990 03

CALL PDUMP WITH COMPLEX*8 FORMAT SPECIFIED
006DD0               (3.0000000,4.0000000)            (4.0000000,8.0000000)

CALL PDUMP WITH COMPLEX*16 FORMAT SPECIFIED
006DE0            (0.99999999999999990,0.99999999999999990)        (-0.99999999999999990,-0.99999999999999990)

CALL PDUMP WITH LITERAL FORMAT SPECIFIED
006E5C     THIS ARRAY CONTAINS ALPHAMERIC DATA
```

Figure 2. Sample Storage Printouts

IBM Sys/360 FORT. IV Libr., Math & Serv Subprogs. Printed in U.S.A. GC28-6818-1

GC28-6818-1

READER'S COMMENTS

TITLE: IBM System/360
FORTRAN IV Library:
Mathematical and Service Subprograms

FORM: GC28-6818-1

Your comments assist us in improving the usefulness of our publications; they are an important part of the input used in preparing updates to the publications. All comments and suggestions become the property of IBM.

Please do not use this form for technical questions about the system or for requests for additional publications; this only delays the response. Instead, direct your inquiries or requests to your IBM representative or to the IBM Branch Office serving your locality.

Corrections or clarifications needed:

Page	*Comment*

Please include your name and address in the space below if you wish a reply.

Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

fold

fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM CORPORATION
1271 Avenue of the Americas
New York, New York 10020

Attention: PUBLICATIONS

fold

fold

IBM

GC28-6818-01

IBM System/360 FORTRAN IV Library, Math & SSP Subprogs. Printed in U.S.A. GC28-6818-1