File No. 7040-36
Form C28-6806-1

IBM Systems Reference Library

IBM 7040/7044 Operating System (16/32K)

Subroutine Library

(FORTRAN IV Mathematical Subroutines)

This publication contains information for both FORTRAN and MAP programmers who desire to use the FORTRAN IV Mathematical Subroutines.

The publication includes the information required to call each subroutine (i.e., calling sequences and valid argument ranges), optional returns for invalid arguments, algorithms and accuracy statistics for most of the subroutines, timing estimates, error conditions, and error messages.

Preface

This publication describes the FORTRAN IV mathematical subroutines. These subroutines are available to any FORTRAN IV or MAP programmer who wishes to use them.

The first part of this publication deals primarily with the rules for calling FORTRAN IV mathematical subroutines in either a FORTRAN IV or MAP program. The second part details the mathematical principles from which these subroutines have been constructed. Also included are performance statistics indicating the speed and accuracy of the subroutines.

The reader should be familiar with one of the following publications, depending upon the programming language with which he works:

IBM 7040/7044 Operating System (16/32K): FORTRAN IV Language, Form C28-6329

IBM 7040/7044 Operating System (16/32K): Macro Assembly Program (MAP) Language, Form C28-6335

MAJOR REVISION (MAY 1965)
This publication, Form C28-6806-1, supercedes the previous edition, Form C28-6806-0, which contained some errors regarding the performance of the subroutines. Changes in the text are indicated by a vertical line to the left of each change; revised illustrations are indicated by the symbol (●) to the left of each caption.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.
Address comments concerning the contents of this publication to:
IBM Corporation, Programming Systems Publications, Dept. D39, 1271 Avenue of the Americas, New York, N.Y. 10020

IBM Technical Newsletter

File Number	7040-36
Re: Form No.	C28-6806-1
This Newsletter No.	N28-0542-0
Date	February 1, 1966
Previous Newsletter Nos.	None

Errata to IBM 7040/7044 Operating System (16/32K)
Subroutine Library
(FORTRAN IV Mathematical Subroutines)

Attached is a replacement for page 8 of the publication IBM 7040/7044 Operating System (16/32K), Subroutine Library (FORTRAN IV Mathematical Subroutines), Form C28-6806-1. The only change is a correction in the valid argument range for subroutines XP2 and XP3.

File this newsletter at the back of the publication. It will provide a reference to changes, a method of determining that all amendments have been received, and a check for determining whether the publication contains the proper pages.

IBM Corporation, Programming Systems Publications, Dept. D39, 1271 Avenue of the Americas, New York, N.Y. 10020

PRINTED IN U.S.A.

Contents

Introduction

The FORTRAN IV mathematical subroutine library contains three types of subroutines: single-precision, double-precision, and complex. These subroutines may be used by a FORTRAN IV or a MAP programmer to perform mathematical computations. This publication provides the information required by a FORTRAN IV or MAP programmer who wishes to use these subroutines.

Calling Sequences

Each subroutine in the library provides one or two mathematical functions. Each function is identified by a unique entry point. For this reason, the name of each subroutine is distinct from the name of its entry point(s).

The method used to call a specific function depends upon the programming language used (i.e., FORTRAN IV or MAP). In each case, however, the programmer must specify an entry point and one or more arguments. (An argument is the name of a location that contains the value to be supplied as input to a function.) Figure 1 shows the general form by which a mathematical function is called in each language. The specific form for calling each function is shown in Figures 2 through 8 of the section "FORTRAN IV Mathematical Subroutines."

Arguments and Answers

The arguments of most of the functions provided by the subroutines must be normalized floating-point numbers. Only the MTN, XP1, XP2, and FDX1 subroutines differ in this respect. These four subroutines, along with XP3, FDX2, and FCA, require calling sequences that differ from the general forms shown in Figure 1. These exceptions are given in Figures 2 through 8 of the section, "FORTRAN IV Mathematical Subroutines."

Double-Precision Arguments

A double-precision argument consists of two adjacent words. The location of the first word is considered to be the location of the entire argument. The first word is the high-order part of the double-precision number, while the second word is the low-order part. MAP programmers should note that the location of the high-order part must be an even-numbered storage location if the processing unit they use is equipped with the double-precision instruction set. This restriction does not apply if double-precision operations are simulated by the Floating-Point Trap Subroutine.

Complex Arguments

A complex argument consists of two adjacent words. The first word contains the real part of the complex argument, while the second word contains the imaginary part. The location of the real part is considered to be the location of the entire complex argument.

Answers

Each function produces a single answer. For FORTRAN IV programs, the answer is stored in the leftmost variable of the arithmetic assignment statement that was used to call the function (see Figure 1). For MAP programs, upon return to the calling program, the answer is found either in the AC register (for single-precision functions), or in the AC and MQ registers (for double-precision and complex functions). More specifically, for double-precision functions, the high-order part of the answer is stored in the AC, and the low-order part in the MQ; for complex functions, the real part is stored in the AC, and the imaginary part is stored in the MQ.

Source Language	Calling Sequences for Functions that Require One Argument	Calling Sequences for Functions that Require Two Arguments
FORTRAN IV	y = entry point (argument) answer stored in y	y = entry point (arg1, arg2) answer stored in y
MAP	CALL entry point (argument) answer left in AC or AC-MQ	CALL entry point (arg1, arg2) answer left in AC or AC-MQ

Figure 1. General Form of Calling Sequences

Role of the Execution Error Monitor

The nature of the functions and the nature of the machine registers used in computations impose certain limits on the range of input arguments for most functions. Figures 2 through 8 of the section, "FORTRAN IV Mathematical Subroutines" give the valid argument range for each function.

Any function that places a limitation on the range of arguments it will accept also provides the user with an option for handling invalid arguments. These options are called optional returns and each type of optional return is identified by an error code. Thus, for each of the 28 optional returns, there is a specific error code. (The optional return and error code for each function are given in Figures 2 through 8 of the section, "FORTRAN IV Mathematical Subroutines.")

Whenever an invalid argument is detected by a function, the function immediately transfers control to the Executor Error Monitor (i.e., subroutine XEM—see the publication *IBM 7040/7044 Operating System (16/32K): System Programmer's Guide,* Form C28-6339, for information about XEM). The XEM subroutine then checks the option control bits (i.e., bits 1 through 28) of MATOP., a one-word control section within XEM. The position of each option control bit corresponds to an error code (and, thus, a specific optional return); that is, bit 1 corresponds to error code 1, bit 2 corresponds to error code 2, etc. If the option control bit corresponding to the error code of the function that called XEM is set to 1, then the optional return for that particular function is to be taken. The XEM subroutine prints a warning message to this effect and execution continues with the optional return having been taken. However, if the option control bit is set to 0, then the optional return for that particular function is not to be taken and execution is to be terminated. The XEM subroutine prints error messages for both the user and the machine operator and then terminates execution.

In the distributed version of XEM, all the option control bits in MATOP. are set to 1. Thus, if these control bits are not reset, all cases of invalid arguments will result in the taking of the appropriate optional return. Termination of execution because of an invalid argument can result only for those functions whose option control bits have been set to 0. This resetting can be accomplished in either of two ways:

1. It can be done during execution by a MAP program which reassembles the MATOP. control section. However, the resetting is effective only for that particular application.

2. It can be done prior to execution by reassembling XEM so that the configuration of the option control bits would permanently meet the user's requirements.

Whenever an optional return is taken, a conventional answer is returned to the user. Among the conventional answers given for invalid arguments is the largest possible floating-point number, $2^{127} - 2^{100}$ in single-precision cases, or $2^{127} - 2^{73}$ in double-precision cases. For the sake of brevity, these numbers are written as *omega* (i.e., Ω) in Figures 2 through 8 of the section, "FORTRAN IV Mathematical Subroutines."

Floating-Point Trap Subroutine

A floating-point trap occurs whenever the AC or MQ, or both, reach an overflow or underflow condition.

Floating-Point Overflow

Except for those subroutines that deal with complex numbers, floating-point overflow can never occur because the arguments are screened upon entry to a subroutine. If an argument is one that could cause overflow, it is immediately treated as an invalid argument and control passes to the Execution Error Monitor.

Floating-point overflow can occur during some of the complex subroutines when an answer approaches the overflow threshold even though both the real and imaginary parts of the complex argument lie within the valid range. Such is the case with those subroutines for which the cost (in space and time) of an effective prescreening of arguments would be so excessive as to hinder seriously the efficiency of the subroutine.

An occurrence of such an overflow causes a floating-point trap. The Floating-Point Trap Subroutine (i.e., system subroutine FPT) then sets the AC and the MQ according to the trap condition. (See the publication *IBM 7040/7044 Operating System (16/32K): System Programmer's Guide,* Form C28-6339, for information about trap conditions.) After the AC and MQ have been set, FPT prints an overflow message and then returns control to the subroutine from which the trap originated.

Floating-Point Underflow

Any subroutine can cause floating-point underflow. An occurrence of underflow causes a floating-point trap. The FPT subroutine then sets the AC and MQ according to the trap condition; after the AC and MQ have been set, FPT returns control to the subroutine from which the trap originated.

Double-Precision Simulation

If a 7040/7044 System is equipped with the double-precision instruction set, the Subroutine Library will take full advantage of it. However, for a 7040/7044 System that does not have the double-precision instruction set, the simulation of double-precision operations is performed by the FPT subroutine as described below.

A floating-point trap occurs whenever a double-precision instruction is encountered during execution. After the trap occurs, the FPT subroutine simulates the double-precision operation and then returns control to the point at which the trap occurred.

NOTE: If double-precision simulation is not required, the user may delete that part of FPT that provides the simulation capability. The publication *IBM 7040/7044 Operating System (16/32K): System Programmer's Guide,* Form C28-6339, gives the details for the deletion procedure.

Evaluating Accuracy

Because the size of each machine word is limited, small errors may be generated by mathematical subroutines. In an elaborate computation, slight inaccuracies can accumulate to become larger errors. Thus, in interpreting final results, the user should take into account any errors introduced during the various intermediate steps. For a detailed discussion of errors, see Appendix A, "Algorithms and Accuracy Considerations."

Fortran IV Mathematical Subroutines

The FORTRAN IV mathematical subroutine library is summarized in the figures that make up the rest of this section. The figures are organized as follows:

Figure 2. Single-Precision Exponential Subroutines
- Exponential (XPN)
- Exponential, fixed-point base and exponent (XP1)
- Exponential, floating-point base, fixed-point exponent (XP2)
- Exponential, floating-point base and exponent (XP3)
- Square Root (SQR)

Figure 3. Single-Precision Trigonometric Subroutines
- Sine/Cosine (SCN)
- Tangent/Cotangent (TNCT)
- Arctangent (ATN)
- Arcsine/Arccosine (ARSCN)

Figure 4. Single-Precision Miscellaneous Routines
- Logarithm (LOG)
- Hyperbolic Sine/Cosine (SCNH)
- Hyperbolic Tangent (TNH)
- Error Function (ERF)
- Gamma/Loggamma (GAMA)

Figure 5. Double-Precision Exponential Subroutines
- Exponential (FDXP)
- Exponential, with floating-point base and fixed exponent (FDX1; entry point DXP1.)
- Exponential, with double-precision base and single- or double-precision exponent (FDX2)
- Square Root (FDSQ)

Figure 6. Double-Precision Trigonometric and Logarithmic Subroutines
- Logarithm (FDLG)
- Sine/Cosine (FDSC)
- Arctangent (FDAT)

Figure 7. Complex Subroutines
- Square Root (FCSQ)
- Exponential (FCXP)
- Exponential, with complex base and fixed exponent (FDX1; entry point CXP1.)
- Logarithm (FCLG)
- Sine/Cosine (FCSC)
- Absolute Value (FCAB)
- Arithmetic (FCA)

Figure 8. Other Subroutines
- MTN Routine
- Modular Function (FDMD)

NOTE: Some of the ranges in these figures are represented in powers of 2. Listed below are these values and the approximate decimal value corresponding to each.

$2^{-127} \cong 5.878 \times 10^{-39}$
$2^{20} \cong 1.049 \times 10^{6}$
$2^{25} \cong 3.355 \times 10^{7}$
$2^{50} \cong 1.126 \times 10^{15}$
$2^{120} \cong 1.663 \times 10^{36}$

Subroutine Name	Entry Point	Definition	Calling Sequence	Valid Argument Range	Error Code	Options	
						If Argument Range Is	Then the Answer Is
XPN (exponential)	EXP	$y = e^x$	FORTRAN IV y = EXP(x) MAP CALL EXP(x)	$x \leq 88.029692$	8	$x > 88.029692$	Ω
XP1 (exponential, fixed-point base, *m*, and fixed-point exponent, *n*)	.EXP1.	$y = m^n$	FORTRAN IV y = m**n MAP CLA m LDQ n CALL .EXP1.	$m \neq 0$ $n \geq 0$	1	$m = n = 0$	0
					2	$m = 0$ $n < 0$	0
XP2 (exponential, floating-point base *x*, and fixed-point exponent *n*)	.EXP2.	$y = x^n$	FORTRAN IV y = x**n MAP CLA x LDQ n CALL .EXP2.	$x \neq 0$	1	$x = n = 0$	0
					2	$x = 0$ $n < 0$	0
XP3 (exponential, floating-point base x_1, and floating-point exponent x_2)	.EXP3.	$y = x_1^{x_2}$	FORTRAN IV $y = x_1**x_2$ MAP CLA x_1 LDQ x_2 CALL .EXP3.	$x_1 > 0$	1	$x_1 = x_2 = 0$	0
					2	$x_1 = 0$ $x_2 < 0$	0
					7	$x_1 < 0$ $x_2 \neq 0$	$\lvert x_1 \rvert^{x_2}$
SQR (square root)	SQRT	$y = x^{\frac{1}{2}}$	FORTRAN IV y = SQRT(x) MAP CALL SQRT(x)	$x \geq 0$	14	$x < 0$	$\lvert x \rvert^{\frac{1}{2}}$

●Figure 2. Single-Precision Exponential Subroutines

Subroutine Name	Entry Point	Definition	Calling Sequence	Valid Argument Range	Error Code	Options	
						If Argument Range Is	Then the Answer Is
SCN (sine and cosine functions, where argument x is expressed in radians)	SIN	$y = \sin(x)$	FORTRAN IV $y = SIN(x)$ MAP CALL SIN(x)	$\|x\| < 2^{25}$	12	$\|x\| \geq 2^{25}$	0
	COS	$y = \cos(x)$	FORTRAN IV $y = COS(x)$ MAP CALL COS(x)				
TNCT (tangent and cotangent functions, where argument x is expressed in radians)	TAN	$y = \tan(x)$	FORTRAN IV $y = TAN(x)$ MAP CALL TAN(x)	$\|x\| < 2^{20}$ and x may not be an odd integral multiple of $\pi/2$ (see Note)	3	$\|x\| \geq 2^{20}$	0
					4	$x \cong k\frac{\pi}{2}$ where k is an odd integer	Ω
	COTAN	$y = \cot(x)$	FORTRAN IV $y = COTAN(x)$ MAP CALL COTAN(x)	$\|x\| < 2^{20}$ and x may not be a multiple of π (see Note)	3	$\|x\| \geq 2^{20}$	0
					4	$x \cong k\pi$ where k is an integer	Ω
ATN (arctangent functions, where the result y is in radians)	ATAN	$y = \arctan(x)$	FORTRAN IV $y = ATAN(x)$ MAP CALL ATAN(x)	Any argument	not applicable	not applicable	not applicable
	ATAN2	$y = \arctan\left(\frac{x_1}{x_2}\right)$	FORTRAN IV $y = ATAN2(x_1,x_2)$ MAP CALL ATAN2(x_1,x_2)	$(x_1,x_2) \neq (0,0)$	9	$(x_1,x_2) = (0,0)$	0
ARSCN (arcsine and arccosine functions, where the result y is in radians)	ARSIN	$y = \arcsin(x)$	FORTRAN IV $y = ARSIN(x)$ MAP CALL ARSIN(x)	$\|x\| \leq 1$	13	$\|x\| > 1$	0
	ARCOS	$y = \arccos(x)$	FORTRAN IV $y = ARCOS(x)$ MAP CALL ARCOS(x)				

Note: For a more detailed discussion of the valid argument ranges for the TNCT subroutine and how these ranges may be expanded or reduced with the MTN subroutine, see Appendix A, "Algorithms and Accuracy Considerations."

Figure 3. Single-Precision Trigonometric Subroutines

Subroutine Name	Entry Point	Definition	Calling Sequence	Valid Argument Range	Error Code	Options	
						If Argument Range Is	Then the Answer Is
LOG (common and natural logarithm functions)	ALOG	$y = \log_e (x)$	FORTRAN IV y = ALOG(x) MAP CALL ALOG(x)	$x > 0$	10	$x = 0$	$-\Omega$
	ALOG10	$y = \log_{10} (x)$	FORTRAN IV y = ALOG10(x) MAP CALL ALOG10(x)		11	$x < 0$	$\log \lvert x \rvert$
SCNH (hyperbolic sine and cosine functions)	SINH	$y = \frac{1}{2}(e^x - e^{-x})$	FORTRAN IV y = SINH(x) MAP CALL SINH(x)	$\lvert x \rvert \leq 88.029692$	5	$\lvert x \rvert > 88.029692$	Ω
	COSH	$y = \frac{1}{2}(e^x + e^{-x})$	FORTRAN IV y = COSH(x) MAP CALL COSH(x)				
TNH (hyperbolic tangent)	TANH	$y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	FORTRAN IV y = TANH(x) MAP CALL TANH(x)	any argument	not applicable	not applicable	not applicable
ERF (error subroutine)	ERF	$y = \frac{2}{\sqrt{\pi}}\int_0^x e^{-u^2}du$	FORTRAN IV y = ERF(x) MAP CALL ERF(x)	any argument	not applicable	not applicable	not applicable
GAMA (gamma and loggamma functions)	GAMMA	$y = \int_0^\infty u^{x-1}(e^{-u})du$	FORTRAN IV y = GAMMA(x) MAP CALL GAMMA(x)	$2^{-127} < x < 34.843$	20	$2^{-127} \geqq x$ or $x \geqq 34.843$	Ω
	ALGAMA	$y = \log\left[\int_0^\infty u^{x-1}(e^{-u})du\right]$	FORTRAN IV y = ALGAMA(x) MAP CALL ALGAMA(x)	$0 < x < 1.54926(2^{120})$	21	$x \leq 0$ or $x \geqq 1.54926(2^{120})$	Ω

Figure 4. Miscellaneous Single-Precision Subroutines

Subroutine Name	Entry Point	Definition	Calling Sequence	Valid Argument Range	Error Code	Options	
						If Argument Range Is	Then the Answer Is
FDXP (exponential)	DEXP	$y = e^x$	FORTRAN IV y = DEXP(x) MAP CALL DEXP(x)	$x \leq 88.029692$	24	$x > 88.029692$	Ω
FDX1 (exponential, floating-point base *x*, and fixed-point exponent *n*) (see Note 1)	DXP1.	$y = x^n$	FORTRAN IV y = x**n MAP CLA x LDQ x + 1 TSL DXP1. PZE n	$x \neq 0$ $n \geq 0$	1	$x = n = 0$	0
					2	$x = 0$ $n < 0$	0
FDX2 (exponential, double-precision base x_1, and single- or double-precision exponent x_2)	DXP2.	$y = x_1^{x_2}$	FORTRAN IV $y = x_1 ** x_2$ MAP CLA x_1 LDQ $x_1 + 1$ TSL DXP2. pfx x_2 (see Note 2)	$x_1 > 0$ $x_2 \geq 0$	1	$x_1 = x_2 = 0$ or $x_2 < 0$	0
					7	$x_1 < 0$ $x_2 \neq 0$	$\lvert x_1 \rvert^{x_2}$
FDSQ (square root)	DSQRT	$y = x^{\frac{1}{2}}$	FORTRAN IV y = DSQRT(x) MAP CALL DSQRT(x)	$x \geq 0$	28	$x < 0$	$\lvert x \rvert^{\frac{1}{2}}$

Note 1: Subroutine FDX1 also has an entry point, CXP1., that provides an exponential function for a complex base with a fixed-point exponent. Information on CXP1. is found in Figure 7.

Note 2: pfx = PZE for a double-precision exponent
pfx = MZE for a single-precision exponent

Figure 5. Double-Precision Exponential Subroutines

Subroutine Name	Entry Point	Definition	Calling Sequence	Valid Argument Range	Error Code	Options	
						If Argument Range Is	Then the Answer Is
FDLG (natural and common logarithm functions)	DLOG	$y = \log_e(x)$	FORTRAN IV y = DLOG(x) MAP CALL DLOG(x)	$x > 0$	25	$x = 0$	$-\Omega$
	DLOG10	$y = \log_{10}(x)$	FORTRAN IV y = DLOG10(x) MAP CALL DLOG10(x)		26	$x < 0$	$\log \lvert x \rvert$
FDSC (sine and cosine functions, where the argument x is expressed in radians)	DSIN	y = sine(x)	FORTRAN IV y = DSIN(x) MAP CALL DSIN(x)	$\lvert x \rvert < 2^{50}\pi$	27	$\lvert x \rvert \geq 2^{50}\pi$	0
	DCOS	y = cosine (x)	FORTRAN IV y = DCOS(x) MAP CALL DCOS(x)				
FDAT (arctangent functions, where the result y is expressed in radians)	DATAN	y = arctan(x)	FORTRAN IV y = DATAN(x) MAP CALL DATAN(x)	any argument	not applicable	not applicable	not applicable
	DATAN2	$y = \arctan\left(\frac{x_1}{x_2}\right)$	FORTRAN IV $y = DATAN2(x_1,x_2)$ MAP $CALL\ DATAN2(x_1,x_2)$	$(x_1,x_2) \neq (0,0)$	22	$(x_1,x_2) = (0,0)$	0

Figure 6. Double-Precision Trigonometric and Logorithmic Subroutines

For the purpose of the following figure,

$z = (x_1, x_2) = x_1 + ix_2$

$y = (y_1, y_2) = y_1 + iy_2$

Subroutine Name	Entry Point	Definition	Calling Sequence	Valid Argument Range	Error Code	Options: If Argument Range Is	Options: Then the Answer Is
FCSQ (square root)	CSQRT	$y = z^{\frac{1}{2}}$ real $y \geqq 0$	FORTRAN IV y = CSQRT (z) MAP CALL CSQRT(z)	any argument (see Note 1)	not applicable	not applicable	not applicable
FCXP (exponential)	CEXP	$y = e^z$	FORTRAN IV y = CEXP(z) MAP CALL CEXP(z)	$x_1 \leqq 88.029692$ $\lvert x_2 \rvert < 2^{25}$	15	$x_1 > 88.029692$	$\Omega(\cos x_2 + i \sin x_2)$
					16	$\lvert x_2 \rvert \geqq 2^{25}$	$0 + 0i$
FDX1 (exponential, complex base z, and fixed-point exponent n) (see Note 2)	CXP1.	$y = z^n$	FORTRAN IV y = z**n MAP CLA z LDQ z + 1 TSL CXP1. PZE n	$z \neq 0$ $n \geqq 0$	1	z = n = 0	0
					2	z = 0 n < 0	0
FCLG (natural logarithm)	CLOG	$y = PV \log_e(z)$ (see Note 3)	FORTRAN IV y = CLOG(z) MAP CALL CLOG(z)	$z \neq 0 + 0i$ (see Note 1)	17	$z = 0 + 0i$	$-\Omega + 0i$
FCSC (sine and cosine functions)	CSIN	y = sine(z)	FORTRAN IV y = CSIN(z) MAP CALL CSIN(z)	$\lvert x_1 \rvert < 2^{25}$ $\lvert x_2 \rvert \leqq 88.029692$	18	$\lvert x_2 \rvert > 88.029692$	(see Note 4)
	CCOS	y = cosine(z)	FORTRAN IV y = CCOS(z) MAP CALL CCOS(z)		19	$\lvert x_1 \rvert \geqq 2^{25}$	$0 + 0i$ or $-0 - 0i$
FCAB (absolute value)	CABS	$y = \lvert z \rvert$	FORTRAN IV y = CABS(z) MAP CALL CABS(z)	any argument (see Note 1)	not applicable	not applicable	not applicable
FCA (arithmetic operations)	FCAOP.	Complex addition, subtraction, multiplication, and division	(see Note 5)	any argument (see Note 1)	not applicable	not applicable	not applicable

Note 1: Floating-point overflow can occur.

Note 2: Subroutine FDX1 also has an entry point, DXP1., to an exponential function for a floating-point base and a fixed-point exponent. (See Figure 5 for further information.)

Note 3: The letters "PV" indicate that the "Principal Value" is returned to the user. That is, the answer given will be from that branch where the imaginary part lies between $-\pi$ and π. More specifically, $-\pi < y_2 \leqq \pi$ unless $x_1 < 0$ and $x_2 = -0$ in which case $y_2 = -\pi$.

Note 4: Optional answers for FCSC subroutine when imaginary part of the complex argument is invalid:

if x_2 is	result of CSIN(z) is	result of CCOS(z) is
> 88.029692	$\frac{\Omega}{2}(\sin x_1 + i\cos x_1)$	$\frac{\Omega}{2}(\cos x_1 - i\sin x_1)$
< -88.029692	$\frac{\Omega}{2}(\sin x_1 - i\cos x_1)$	$\frac{\Omega}{2}(\csc x_1 + i\sin x_1)$

Note 5: Calling sequences for FCA subroutine:

Arithmetic Operation	FORTRAN IV	MAP z_1 in AC-MQ
Addition $z_1 + z_2$	$y = z_1 + z_2$	TSL FCAOP. PZE z_2
Subtraction $z_1 - z_2$	$y = z_1 - z_2$	TSL FCAOP. PON z_2
Multiplication $z_1 \cdot z_2$	$y = z_1 * z_2$	TSL FCAOP. PTW z_2
Division z_1/z_2	$y = z_1/z_2$	TSL FCAOP. PTH z_2

● Figure 7. Complex Subroutines

Subroutine Name	Entry Point	Definition	Calling Sequence	Valid Argument Range	Error Code	Options	
						If Argument Range Is	Then the Answer Is
MTN	MTAN	Resets the accuracy guarantee for the single-precision tangent subroutine.	FORTRAN IV CALL MTAN(k) MAP CALL MTAN(k)	$k \geq 0$ but *k* must be an integer (see Note)	not applicable	not applicable	not applicable
FDMD (modular function)	DMOD	$y = x \pmod z$ is computed as $y = x - [x/z]\, z$ where brackets denote the integral part of the quantity	FORTRAN IV $y =$ DMOD(x,z) MAP CALL DMOD(x,z)	any double-precision floating-point numbers	not applicable	not applicable	not applicable
Note: A detailed description of the purpose and use of MTN is found in Appendix A, "Algorithms and Accuracy Considerations."							

Figure 8. Other Subroutines

Appendix A: Algorithms and Accuracy Considerations

Introductory Information

This section contains information on the algorithms and performance of most of the subroutines in the FORTRAN IV mathematical subroutine library. The subroutines for which no such information is given are: XP1, XP2, XP3, FDX1, FDX2, and FDMD.

Accuracy

Because the size of a machine word is limited, small errors may be generated by mathematical subroutines. In an elaborate computation, slight inaccuracies can accumulate to become larger errors. Thus, in interpreting final results, the user should take into account any errors introduced during the various intermediate stages.

The accuracy of an answer returned by a subroutine is influenced by two factors: (1) the accuracy of the argument, and (2) the performance of the subroutine.

The Accuracy of the Argument

Most arguments contain errors. An error in a given argument may have accumulated over several steps prior to the use of the subroutine. Even data fresh from input conversion contain slight errors since decimal data cannot usually be exactly converted into the binary form required by the processing unit; the conversion process is usually only approximate. Argument errors always influence the accuracy of answers. The effect of an argument error on the accuracy of an answer depends solely on the nature of the mathematical function involved and not on the particular coding by which that function is computed within a subroutine. In order to assist users in assessing the accumulation of errors, a guide on the propagational effect of argument errors is provided for each function. Wherever possible, this is expressed as a simple formula.

The Performance of the Subroutines

The performance statistics supplied in this appendix are based upon the assumption that arguments are perfect (i.e., without errors, and therefore having no argument error propagation effect upon answers). Thus the only errors in answers are those introduced by the subroutines themselves.

For each subroutine, accuracy figures are given for one or more segments throughout the valid argument range(s). In each case the particular statistics given are those most meaningful to the function and range under consideration. For example, the maximum relative error and standard deviation of the relative error of a set of answers are generally useful and revealing statistics, but useless for the range of a function where its value becomes 0, since the slightest error of the argument value can cause an unbounded fluctuation on the relative magnitude of the answer. Such is the case with sin(x) for x near π, and in this range it is more appropriate to discuss absolute errors.

Symbols Used in Describing Accuracy

In the presentation of error statistics, the following symbols are employed:

$g(x)$ = the answer given by the subroutine for the mathematical function under discussion.

$f(x)$ = the correct extra precision answer for the mathematical function under discussion.

$\epsilon = \left|\frac{f(x) - g(x)}{f(x)}\right|$, the relative error of the answer.

δ = the relative error of the argument.

$E = |f(x) - g(x)|$, the absolute error of the answer.

Δ = the absolute error of the argument.

$M(E) = \text{Max}\,|f(x) - g(x)|$, the maximum absolute error produced during testing.

$M(\epsilon) = \text{Max}\left|\frac{f(x) - g(x)}{f(x)}\right|$, the maximum relative error produced during testing.

$\sigma(E) = \sqrt{\frac{1}{N}\sum_i \left|f(x_i) - g(x_i)\right|^2}$, the root-mean-square (standard deviation) absolute error.

$\sigma(\epsilon) = \sqrt{\frac{1}{N}\sum_i \left|\frac{f(x_i) - g(x_i)}{f(x_i)}\right|^2}$, the root-mean-square (standard deviation) relative error.

When applied to complex numbers, the absolute value signs in the above formulas should be regarded as denoting complex absolute value. Thus, the above formula for E represents the vector error when applied to a complex function.

Algorithms

Some of the formulas are widely known; those that are not widely known are derived from more commonly known formulas. In such cases, the steps leading from the common formula to the computational one have been detailed so that the derivation may be reconstructed by anyone who has a basic understanding of mathematics and who has access to the common texts on numerical analysis. Background information for algorithms involving continued fractions may be found in Wall, H.S., *Analytic Theory of Continued Fractions*, D. Van Nostrand Co., Inc., Princeton, N. J., 1948.

Algorithms and Performance Statistics

Single-Precision Exponential—XPN

Algorithm

1. Write $x[\log_2(e)] = n + r$ (where n is the integer part and r is the fraction part). Then

$$e^x = (2^n)(2^r), \quad -1 < r < 1$$

2. Compute 2^r by means of a rational approximation formula. This formula was derived in the following way:

Take the Gaussian-type continued fraction

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

truncate at the ninth term and rewrite it to obtain

$$e^z \cong \frac{1680 + 840z + 180z^2 + 20z^3 + z^4}{1680 - 840z + 180z^2 - 20z^3 + z^4}$$

Substituting $r[\log_e(2)]$ for z and rewriting the above, we get:

$$2^r \cong 1 + \frac{2r}{Cr^2 - r + D - \left(\frac{B}{r^2 + A}\right)}$$

where A, B, C, and D are constants.

The maximum relative error of this formula is 1.6×10^{-9}.

3. If $x < -89.415987$, then 0 is given as the answer.

4. Computation is carried out in fixed-point to minimize truncation errors.

Effect of an Argument Error

$\epsilon \sim \Delta$. Since $\Delta = \delta x$, for the larger value of x, even the round-off error of the argument causes a substantial relative error in the answer.

Performance Statistics

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$0 \leq x \leq 1$	3.37×10^{-9}	7.32×10^{-9}	859	339
$\lvert x \rvert \leq 88.028$	4.80×10^{-9}	1.50×10^{-8}	850	334
Note: The sample arguments upon which the above statistics are based were uniformly distributed over the specified range.				

Figure 9. XPN, Exponential Function

Single-Precision Square Root—SQR

Algorithm

1. Write $x = [2^{2p-q}]\,(m)$, where p is an integer, $q = 0$ or 1, and $1/2 \leq m < 1$. Then, $\sqrt{x} = 2^p \sqrt{2^{-q}(m)}$ for $1/4 \leq [2^{-q}(m)] < 1$.

2. The first approximation y_0 of $\sqrt{x}$ for $1/4 \leq x < 1$ is obtained by one of the four hyperbolic fits of the form $y_0 = a + b/(c + x)$ depending on the size of x. If $2^{-1/2} \leq x < 1$, then the constants a, b, and c are chosen to minimize the relative error of the fit while giving the exact value for $x = 1$. The maximum relative error of this fit is 0.5×10^{-4}, and its contribution to the final relative error is less than 0.13×10^{-8}.

If $1/4 \leq x < 2^{-3/2}$, or $2^{-3/2} \leq x < 1/2$, or $1/2 \leq x < 2^{-1/2}$, the constants a, b, and c are chosen to minimize the contribution to the absolute error of the answer. In these cases, the contribution to the final absolute error is less than 0.5×10^{-9}, or 0.6×10^{-9}, or 0.7×10^{-9} depending on the range.

3. Apply the Newton-Raphson iteration once to get the answer:

$$y_1 = \frac{1}{2}\left(y_0 + \frac{x}{y_0}\right)$$

Effect of an Argument Error

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

Performance Statistics

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
All positive numbers	3.32×10^{-9}	7.54×10^{-9}	688	243
Note: The sample arguments upon which the above statistics are based were exponentially distributed over the specified range.				

Figure 10. SQR, Square Root Function

Single-Precision Sine/Cosine—SCN

Algorithm

1. $\sin(-x) = -\sin(x)$, $\cos(-x) = \cos(x)$; assume $x \geq 0$.

2. Write $x = \frac{\pi}{4}(q) + r$, where q is an integer, and $0 \leq r < \frac{\pi}{4}$. Let $q_0 \equiv q \pmod 8$.

3. If cos (x) is desired, raise q_0 by 2, reduce it modulo 8 and compute sine. If sin (x) is desired and if $x < 2^{-13}$, give x as the answer.

4. Now the case is reduced to the computation of $\sin(\frac{\pi}{4} q_0 + r)$, $0 \leq q_0 \leq 7$. Using the formulas:

$$\sin\left[\frac{\pi}{4}(4 + q_0) + r\right] = -\sin\left[\frac{\pi}{4}(q_0) + r\right] \quad 0 \leq q_0 \leq 3$$

$$\sin\left[\frac{\pi}{4}+r\right]=\cos\left[\frac{\pi}{4}-r\right]$$
$$\sin\left[\frac{\pi}{2}+r\right]=\cos(r)$$
$$\sin\left[\frac{3\pi}{4}+r\right]=\sin\left[\frac{\pi}{4}-r\right]$$

the case is reduced to the computation of sin (r) or cos (r) for $0 \leq r \leq \frac{\pi}{4}$.

5. The coefficients of the approximation $\sin(r) \cong r\,(s_0 + s_1r^2 + s_2r^4 + s_3r^6)$ were obtained by the Chebyshev interpolation over the range $0 \leq r \leq \frac{\pi}{4}$.

The coefficients of the approximation $\cos(r) \cong 1 + c_1r^2 + c_2r^4 + c_3r^6 + c_4r^8$ were obtained by the Chebyshev interpolation over the range

$$-0.01 \leq r \leq \frac{\pi}{4}$$

The relative error of the sine formula is less than 0.34×10^{-8}.

The relative error of the cosine formula is less than 0.73×10^{-10}.

6. Computations are carried out in fixed-point to minimize truncation errors.

Effect of an Argument Error

$E \sim \Delta$. As the argument gets larger, Δ grows, and since the value of the function is periodically diminishing, no consistent relative error control can be maintained outside the principal range of $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$. This holds true for cosine as well.

Performance Statistics

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$\lvert x\rvert \leq \frac{\pi}{2}$	4.82×10^{-9}	1.64×10^{-8}	1063	437

Figure 11. SCN, Sine Function (I)

Argument Range	Root-Mean-Square Absolute Error $\sigma(E)$	Maximum Absolute Error $M(E)$	Average Speed in Microseconds	
			7040	7044
$\lvert x\rvert \leq \frac{\pi}{2}$	0.90×2^{-28}	1.20×2^{-27}	1063	437
$\frac{\pi}{2} < \lvert x\rvert \leq 10$	1.10×2^{-28}	1.84×2^{-27}	1150	475
$10 < \lvert x\rvert \leq 100$	1.11×2^{-28}	1.90×2^{-27}	1154	482

Figure 12. SCN, Sine Function (II)

Argument Range	Root-Mean-Square Absolute Error $\sigma(E)$	Maximum Absolute Error $M(E)$	Average Speed in Microseconds	
			7040	7044
$0 \leq x \leq \pi$	1.14×2^{-28}	1.85×2^{-27}	1236	501
$-20 \leq x < 0$ $\pi < x \leq 20$	1.11×2^{-28}	1.88×2^{-27}	1234	500

Figure 13. SCN, Cosine Function

Note: The sample arguments upon which the statistics in Figures 11, 12, and 13 are based were uniformly distributed over the specified ranges.

Single-Precision Tangent/Cotangent—TNCT

Algorithm

1. If $x < 0$, use $\tan(x) = -\tan(|x|)$, $\cot(x) = -\cot(|x|)$. Assume $x \geq 0$ now.

2. Write $x = \frac{\pi}{4}(q) + r$, where q is an integer and $0 \leq r < \frac{\pi}{4}$. Let $q_0 \equiv q \pmod 4$.

3. If $q_0 = 0$ or 2 (i.e., octant 1 and 3), define $r_0 = r$. If $q_0 = 1$ or 3 (i.e., octant 2 and 4), define $r_0 = \frac{\pi}{4} - r$.

4. Define the case number s as follows:
If tan (x) is desired, $s = q_0$.
If cotan (x) is desired, $s = 1$ for $q_0 = 0$, or $s = 0$ for $q_0 = 1$, or $s = 3$ for $q_0 = 2$, or $s = 2$ for $q_0 = 3$.

5. Compute the factor F as follows:

$$F = 1 + \frac{13.946\,r_0^2 - 313.11}{r_0^2 - 104.46 + \left(\frac{939.33}{r_0^2}\right)}, \text{ if } r_0 > 2^{-14}.$$

If $r_0 \leq 2^{-14}$, then $F = 1$.

This approximation can be obtained by rewriting the continued fraction:

$$\frac{\tan(r_0)}{r_0} \cong \frac{1}{1 - \frac{r_0^2}{3 - \frac{r_0^2}{5 - \frac{r_0^2}{7 - \frac{r_0^2}{8.946}}}}}$$

The maximum relative error of this formula is 10^{-9}.

6. Now the answer is r_0/F for $s = 0$, F/r_0 for $s = 1$, $-F/r_0$ for $s = 2$, $-r_0/F$ for $s = 3$.

Relative Error Control

Let $x = (2^n)\,m$. If the case number s above is 1 or 2, and if the reduced argument r_0 is less than 2^{-26+n} (with the exception of a cotangent entry with small arguments), an execution error is signalled. This is the case when the argument is so close to a singularity that the minimal indeterminate value of the argument (caused by pre-rounding) can cause a relative error of up to 1/3. No screening is given for arguments near a zero of the function.

The foregoing can be strengthened or eliminated by the use of the MTN subroutine.

An execution error is also signalled if $|x| \geq 2^{20}$, or if the cotangent function is requested with $|x| < 2^{-126}$.

Effect of an Argument Error

$E \sim \Delta/\cos^2(x)$, $\epsilon \sim 2\Delta/\sin(2x)$ for tan (x). Thus, near the singularities $x = (k + 1/2)\pi$, where k is an integer, neither absolute error control nor relative error control can be maintained. Similarly, this is true for cotan(x), where $x = k\pi$, and k is an integer.

Performance Statistics

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$\lvert x \rvert \leq \frac{\pi}{4}$	4.40×10^{-9}	1.41×10^{-8}	1106	458
$\frac{\pi}{4} < \lvert x \rvert \leq \frac{\pi}{2}$	5.29×10^{-9}	(6.46×10^{-8})*	1386	557
$\frac{\pi}{2} < \lvert x \rvert \leq 10$	9.35×10^{-9}	(6.09×10^{-8})*	1307	532
$10 < \lvert x \rvert \leq 100$	9.70×10^{-9}	(9.97×10^{-8})*	1311	533

Figure 14. TNCT. Tangent Function

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$\lvert x \rvert \leq \frac{\pi}{4}$	4.34×10^{-9}	1.34×10^{-8}	1234	498
$\frac{\pi}{4} < \lvert x \rvert \leq \frac{3\pi}{4}$	1.05×10^{-8}	(8.64×10^{-8})*	1345	537
$\frac{3\pi}{4} < \lvert x \rvert \leq 10$	9.08×10^{-9}	(2.41×10^{-8})*	1406	559

Figure 15. TNCT, Cotangent Function

*The figures cited as the maximum relative errors are those encountered among 2500 random samples in the respective ranges. For all the perfect arguments in the full legal range (under the standard error control), the maximum relative error is estimated to be 2×10^{-4}.

Note: The sample arguments upon which the statistics in Figures 14 and 15 are based were uniformly distributed over the specified ranges.

Single-Precision Arctangent—ATN

Algorithm

1. Assume $0 \leq x \leq 1$,
   $\arctan(-x) = -\arctan(x)$;
   $\arctan\left(\frac{1}{|x|}\right) = \frac{\pi}{2} - \arctan(|x|)$

2. If $\left[\tan(15°)\right] \leq x \leq 1$, reduce further to the range $|\bar{x}| \leq \left[\tan(15°)\right]$ by $\arctan(x) = 30° + \arctan(\bar{x})$, where $\bar{x} = \sqrt{3} - \frac{4}{x + \sqrt{3}}$.

3. By transforming the Taylor series into a continued fraction, we obtain

$$\arctan(x) = x \left[1 - \frac{x^2}{3} + \frac{\frac{x^2}{5}}{\frac{5}{7} + x^{-2}} - \frac{\frac{(4)(5)}{(7)(7)(9)}}{\frac{43}{(7)(11)} + x^{-2} + w} \right]$$

(where w is an abbreviation for further items)

Dropping w and rewriting the formula, we get

$$\arctan(x) = x \left\{ - \frac{64\,x^2}{(3)(5^2)(7^2)} + \frac{(41)(64)}{(5^3)(7^2)} + \frac{\frac{(3^5)(13)(79)}{(5^4)(7^3)}}{x^2 + \frac{34063}{(3)(5)(13)(79)} - \left[\frac{\frac{(14641)(1100)}{(3^2)(7)(13^2)(79^2)}}{x^2 + \frac{(11)(389)}{(3)(13)(79)}} \right]} \right\}$$

The maximum relative error of this approximation for $|x| \leq [\tan(15°)]$ is 6×10^{-11}.

4. Fixed-point computation is used to minimize truncation errors.

5. ATAN2 provides the extended answer range $-\pi \leq y \leq \pi$, depending on the combination of signs of the two arguments.

Effect of an Argument Error

$E \sim \frac{\Delta}{1 + x^2}$. For small x, $\epsilon \sim \delta$; and as x becomes large, the effect of δ on ϵ diminishes.

Performance Statistics

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
The Entire Range	3.56×10^{-9}	1.41×10^{-8}	1293	516

Note: The sample arguments upon which the above statistics are based were tangents of uniformly distributed numbers between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. (That is, the sample was such that the values returned by the function were uniformly distributed over the answer range.)

Figure 16. ATN, Arctangent Function

Single-Precision Arcsine/Arccosine—ARSCN

Algorithm

1. If $0 \leq x \leq 1/2$, compute arcsin(x) by the use of the Chebyshev interpolation polynomial of degree 5 over this range.

2. If $\frac{1}{2} < x \leq 1$,

$$\arcsin(x) = \frac{\pi}{2} - 2\left[\arcsin\left(\sqrt{\frac{1-x}{2}}\right)\right]$$

In this range we have $0 \leq \sqrt{\frac{1-x}{2}} < 1/2$, which reduces the case to that of item 1 above.

3. If $0 \leq x \leq 1$, $\arccos(x) = \frac{\pi}{2} - \arcsin(x)$. This reduces the case of arccosine to that of arcsine.

4. If $-1 \leq x < 0$, use $\arcsin(x) = -\arcsin(-x)$, and $\arccos(x) = \pi - \arccos(-x)$ to reduce to the earlier cases.

5. The SQR subroutine is used in step 2, above.

Effect of an Argument Error

$E \sim \frac{\Delta}{\sqrt{1-x^2}}$. Thus, for small x, $E \sim \Delta$. For ARSIN with a small x, $\epsilon \sim \delta$. Toward the limit of the range, a small argument error causes a substantial error in the answer.

Performance Statistics

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$\lvert x \rvert \leq 1$	5.39×10^{-9}	3.15×10^{-8}	1335	551

Figure 17. ARSCN, Arcsine Function

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$\lvert x \rvert \leq 1$	5.60×10^{-9}	1.93×10^{-8}	1421	575

Figure 18. ARSCN, Arccosine Function

Note: The sample arguments upon which the statistics in Figures 17 and 18 are based were uniformly distributed over the specified range.

Single-Precision Logarithm—LOG

Algorithm

1. If $|1-x| < 2^{-7}$, use the polynomial approximation:

$$\log(1+z) \cong z - [2^{-1} + 3(2^{-18})]\, z^2 + \frac{1}{3} z^3,$$

where $z = x - 1$.

The maximum relative error of this formula for $|z| < 2^{-7}$, is 3×10^{-8}.

2. If $|1-x| \geq 2^{-7}$, reduce the case as follows:

write $x = 2^p(m)$, where $\frac{1}{2} \leq m < 1$;

$$z = \frac{m - \frac{1}{\sqrt{2}}}{m + \frac{1}{\sqrt{2}}}, \text{ then } |z| < 0.1716.$$

Also $\frac{1+z}{1-z} = m\sqrt{2}$ and

$$\log_e(x) = \left[p - \frac{1}{2} + \log_2\left(\frac{1+z}{1-z}\right)\right] \log_e(2).$$

3. By transforming the Taylor series into a continued fraction, we obtain

$$\log_e\left(\frac{1+z}{1-z}\right) = 2z\left[1 + \frac{z^2}{3} + \frac{\frac{z^2}{5}}{-\frac{5}{7} + z^{-2} + w(z)}\right]$$

When $z = \sqrt{\frac{(7)(11)}{2843}}$, the approximate value of the remainder term w(z) is $\left[-\frac{11}{(7)(9)(140)}\right]$. Thus, we obtain the approximation:

$$\log_e\left(\frac{1+z}{1-z}\right) \cong \frac{(3^3)(4)(5)(7^2) - (3)(3371)(z^2) - (1019)(z^4)}{(3^3)(4)(5)(7^2) - (3)(6311)(z^2)}(2z) \quad (*)$$

This formula reduces to the form

$$\log_2\left(\frac{1+z}{1-z}\right) \cong z\left[c_1 + c_2 z^2 + c_3/(z^2 + c_4)\right].$$

The maximum relative error of (*) is 0.62×10^{-9} for $|z| < 0.1716$. However, since the procedure in item 2 may involve the cancellation of significant digits, this relative accuracy can not be maintained for the final result if the argument value is near 1.

Effect of an Argument Error

$E \sim \delta$. In particular, if δ is the round-off error of the argument, say $\delta \sim 7 \times 10^{-9}$, then $E \sim 7 \times 10^{-9}$. This means that if the argument is close to 1, the relative error can be very large, since the value of the function at that point is very small.

Performance Statistics

Argument Range	Root-Mean-Square Absolute Error $\sigma(E)$	Maximum Absolute Error $M(E)$	Average Speed in Microseconds	
			7040	7044
$\frac{15}{16} \leq x \leq \frac{17}{16}$	1.17×2^{-33}	1.44×2^{-31}	1024	419
Note: The sample arguments upon which the above statistics are based were uniformly distributed over the specified range.				

Figure 19. LOG, Natural Logarithm Function (I)

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
All positive numbers outside $\left(\frac{15}{16}, \frac{17}{16}\right)$	3.39×10^{-9}	8.95×10^{-9}	1104	452
Note: The sample arguments upon which the above statistics are based were exponentially distributed over the specified range.				

Figure 20. LOG, Natural Logarithm Function (II)

Argument Range	Root-Mean-Square Absolute Error $\sigma(E)$	Maximum Absolute Error $M(E)$	Average Speed in Microseconds	
			7040	7044
$\frac{15}{16} \leq x \leq \frac{17}{16}$	1.95×2^{-34}	1.14×2^{-31}	1151	471
Note: The sample arguments upon which the above statistics are based were uniformly distributed over the specified range.				

● Figure 21. LOG, Common Logarithm Function (I)

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
All positive numbers outside $\left(\frac{15}{16}, \frac{17}{16}\right)$	4.84×10^{-9}	1.75×10^{-8}	1231	504
Note: The sample arguments upon which the above statistics are based were exponentially distributed over the specified range.				

● Figure 22. LOG, Common Logarithm Function (II)

Single-Precision Hyperbolic Sine/Hyperbolic Cosine —SCNH

Algorithm

1. $\cosh(x) = \frac{e^x + e^{-x}}{2}$
2. If $|x| > 0.3465736$, use $\sinh(x) = \frac{e^x - e^{-x}}{2}$
3. If $|x| \leq 0.3465736$, use $\sinh(x) \cong x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!}$

The maximum relative error of this approximation is 6×10^{-10}.

4. This subroutine uses the exponential subroutine XPN.

Effect of an Argument Error

For the SINH function:

$$E \sim \Delta\,[\cosh(x)] + \frac{\Delta^2}{2}[\sinh(x)] \text{ and } \epsilon \sim \Delta\,[\coth(x)] + \frac{\Delta^2}{2}$$

For the COSH function:

$$E \sim \Delta\,[\sinh(x)] + \frac{\Delta^2}{2}\,[\cosh(x)] \text{ and } \epsilon \sim \Delta\,[\tanh(x)] + \frac{\Delta^2}{2}$$

In particular, for COSH, $\epsilon \sim \Delta$ over the entire range. However, for SINH given a small value for x, $\epsilon \sim \delta$.

Performance Statistics

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$\lvert x \rvert \leq 0.3466$	5.78×10^{-9}	1.45×10^{-8}	731	288
$0.3466 < \lvert x \rvert \leq 10$	7.10×10^{-9}	2.61×10^{-8}	1427	534

Figure 23. SCNH, Hyperbolic Sine Function

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$\lvert x \rvert \leq 10$	7.26×10^{-9}	2.22×10^{-8}	1314	500

Figure 24. SCNH, Hyperbolic Cosine Function

Note: The sample arguments upon which the above statistics in Figures 23 and 24 are based were uniformly distributed over the specified range.

Single-Precision Hyperbolic Tangent—TNH

Algorithm

1. For $|x| < 0.5493$, use a modified continued fraction

$$\tanh(x) = \frac{x}{1} + \frac{x^2}{3} + \frac{x^2}{5} + \frac{x^2}{7} + \frac{x^2}{9.02743}$$

The maximum relative error of this approximation is 4×10^{-9}.

2. For $0.5493 \leq x < 10.4$, use $\tanh(x) = 1 - \frac{2}{(e^{2x}) + 1}$
3. The XPN subroutine is used in step 2, above.
4. For $x \geq 10.4$, give $\tanh(x) = 1$.
5. For $x \leq -0.5493$, use $\tanh(-x) = -\tanh(x)$.

Effect of an Argument Error

$E \sim [1 - \tanh^2(x)]\Delta$, $\epsilon \sim \frac{2\Delta}{\sinh(2x)}$. Thus, for small values of x, $\epsilon \sim \delta$, and x gets larger, the effect of δ on ϵ diminishes.

Performance Statistics

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$\lvert x\rvert \leq 0.5493$	5.70×10^{-9}	1.44×10^{-8}	993	401
$0.5493 < \lvert x\rvert \leq 10.4$	4.28×10^{-9}	1.09×10^{-8}	1307	495
Note: The sample arguments upon which the above statistics are based were uniformly distributed over the specified range.				

Figure 25. TNH, Hyperbolic Tangent Function

Single-Precision Error Function Subroutine—ERF

Algorithm

1. $\text{erf}(-x) = -\text{erf}(x)$. Assume $x \geq 0$ now.

2. If $x > 4.17$, $\text{erf}(x) \cong 1$.

3. If $4.17 \geq x > 1.51$, use the following Gaussian-type continued fraction:

$$1 - \text{erf}(x) = \frac{2}{\sqrt{\pi}}\left[\int_x^\infty e^{-u^2}\,du\right]$$

where

$$\int_x^\infty e^{-u^2}\,du = e^{-x^2}\left[\frac{0.5x}{x^2+0.5} - \frac{0.5}{x^2+2.5} - \frac{3.0}{x^2+4.5} - \ldots - \frac{(n-\frac{1}{2})n}{x^2+2n+\frac{1}{2}} - \ldots\right]$$

$$\cong e^{-x^2}\left[\frac{0.5x}{x^2+0.5} - \frac{0.5}{x^2+2.5} - \frac{3.0}{x^2+4.5} - \frac{7.5}{x^2+6.5} - \frac{10.803}{x^2+4.269}\right]$$

The maximum absolute error of this approximation is 1.1×10^{-9}.

The two constants in the last term are obtained by requiring that the formula give the exact values at $x = \sqrt{2.3125}$ and $x = \sqrt{2.75}$.

4. If $1.51 \geq x \geq 0$, use the continued fraction obtained by transforming the Taylor expansion of *erf*:

$$\left(\frac{\sqrt{\pi}}{2x}\right)[\text{erf}(x)] = 1 - \frac{x^2}{3} + \frac{x^4}{2!5} - \frac{x^6}{3!7} + \frac{x^8}{4!9} - \cdots$$

$$= 1 - \cfrac{1.0281x^2}{x^2+10.216} + \cfrac{-167.17}{x^2+9.8103} + \cfrac{201.39}{x^2+11.570} + \cfrac{31.228}{x^2-1.7730} + \cfrac{64.244}{x^2+5.578+w(x)}$$

(Constants given in the second formula are approximate.)

Drop w(x) and modify the two constants of the last term so that the formula is exact at $x = 1$ and $x = \sqrt{2}$. The maximum relative error of the formula obtained in this way is 2×10^{-9}.

5. Fixed-point computation is employed to minimize truncation errors. This subroutine uses the exponential subroutine XPN.

Effect of an Argument Error

$E \sim \Delta(e^{-x^2})$. As the magnitude of the argument increases from 1, the effect of an argument error on the final accuracy diminishes rapidly. For small x, $\epsilon \sim \delta$.

Performance Statistics

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$\lvert x\rvert \leq 4$	4.75×10^{-9}	1.95×10^{-8}	1972	792
Note: The sample arguments upon which the above statistics are based were uniformly distributed over the specified range.				

Figure 26. ERF, Error Function

Single-Precision Gamma/Loggamma—GAMA

Algorithm

1. If $0 < x \leq 2^{-127}$, use $\log[\Gamma(x)] \cong -\log(x)$ for ALGAMA.

2. If $2^{-127} < x < 4$ for GAMMA, reduce the case to $1 \leq x \leq 2$, using $x\,[\Gamma(x)] = \Gamma(x+1)$. Compute $\Gamma(x)$ for $1 \leq x \leq 2$, using a continued fraction obtained in the following manner:

$$\Gamma(x) = \int_0^\infty e^{-t}\,t^{x-1}\,dt = \sum_{k=0}^{\infty} \frac{2^{k+1}\,a_k}{k!}\,(x-1.5)^k \quad (*)$$

where

$$a_k = \int_0^\infty (\log t)^k\,e^{-t^2}\,t^2\,dt$$

and

$a_0 = \frac{1}{4}\sqrt{\pi}$
$a_1 = 0.80845993 \times 10^{-2}$
.
.
.
$a_{21} = -0.1628 \times 10^{10}$

Then, by transforming (*) into a continued fraction, we obtain,

$$\Gamma(z+1.5) = \cfrac{\alpha_1}{z+\beta_1} + \cfrac{\alpha_2}{z+\beta_2} + \cfrac{\alpha_3}{z+\beta_3} + \cfrac{\alpha_4}{z+\beta_4} + \cfrac{\alpha_5}{z+\beta_5+w(z)} \quad (**)$$

where

$\alpha_1 = -1.2581927$	$\beta_1 = -11.746649$
$\alpha_2 = 33.814358$	$\beta_2 = -4.8326355$
$\alpha_3 = 59.285853$	$\beta_3 = 8.1171874$
$\alpha_4 = -3.6512651$	$\beta_4 = 0.20317850$
$\alpha_5 = 6.0985782$	$\beta_5 = 1.4063097$

Finally, drop w(z) in (**), and compensate for it by modifying the constants β_4, α_5, and β_5 to obtain an approximation formula accurate to within an absolute error of 2.3×10^{-9}.

3. If $2^{-127} < x < 4$ for ALGAMA, compute $\log[\Gamma(x)]$

by first computing $\Gamma(x)$ as in item 2, and then taking the logarithm of the result.

4. If $4 \leq x < 1.54926(2^{120})$ for ALGAMA, compute $\log[\Gamma(x)]$ as follows:

$$\log[\Gamma(x)] \cong x[\log(x) - 1] - \tfrac{1}{2}\log(x) + \tfrac{1}{2}\log(2\pi) + F(x)$$

where

$$F(x) \cong 0 \text{ if } x \geq 2^{12}$$

$$F(x) \cong \frac{1}{12x} - \frac{1}{360x^3} + \frac{1}{1260x^5} - \frac{1}{1760x^7} \text{ if } x < 2^{12}$$

This formula is the result of economizing Stirling's asymptotoc series. For the range considered, its absolute error is less than 2.1×10^{-9}.

5. If $4 \leq x < 34.843$ for GAMMA, compute $\Gamma(x)$ by first computing $\log[\Gamma(x)]$ as in item 4, and then taking its exponential base e.

6. Subroutines LOG and XPN are used by this subroutine.

Effect of an Argument Error

For $\Gamma(x)$, $\epsilon \sim [\Psi(x)]\Delta$, and for $\log[\Gamma(x)]$, $E \sim [\Psi(x)]\Delta$, where Ψ is the digamma function.

For $1/2 < x < 3$, $-2 < \Psi(x) < 1$, and $E \sim \Delta$ for $\log[\Gamma(x)]$. However, since $x = 1$ and $x = 2$ are zeros of $\log[\Gamma(x)]$, even a small δ can cause a substantial ϵ in this interval.

For large values of x, $\Psi(x) \sim \log(x)$. Hence, for $\Gamma(x)$, $\epsilon \sim \delta[x\log(x)]$. This shows that even the round-off error of the argument contributes greatly to the relative error of the answer. However, for $\log[\Gamma(x)]$ with large values of x, $\epsilon \sim \delta$.

Performance Statistics

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$2^{-127} < x < 1$	4.38×10^{-9}	1.26×10^{-8}	1542	596
$1 \leq x < 2$	5.32×10^{-9}	1.06×10^{-8}	1446	552
$2 \leq x < 4$	6.88×10^{-9}	2.84×10^{-8}	1617	632
$4 \leq x < 10$	9.00×10^{-8}	3.31×10^{-7}	3175	1308
$10 \leq x < 34$	5.88×10^{-7}	2.02×10^{-6}	3178	1310

Figure 27. GAMA, Gamma Function

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$0 < x \leq 1/2$	5.18×10^{-9}	2.14×10^{-8}	2620	1039
$3 \leq x < 4$	8.45×10^{-9}	3.47×10^{-8}	2761	1102
$4 \leq x < 10$	1.08×10^{-8}	3.61×10^{-8}	2221	943
$10 \leq x < 100$	1.30×10^{-8}	3.35×10^{-8}	2228	950

Figure 28. GAMA, Loggamma Function (I)

Argument Range	Root-Mean-Square Absolute Error $\sigma(E)$	Maximum Absolute Error $M(E)$	Average Speed in Microseconds	
			7040	7044
$1/2 < x < 3$	1.28×2^{-28}	1.94×2^{-27}	2574	1020

Figure 29. GAMA, Loggamma Function (II)

Note: The sample arguments upon which the statistics in Figures 27, 28, and 29 are based were uniformly distributed over the specified ranges.

Double-Precision Exponential—FDXP

Algorithm

1. $e^x = 2^y$, where $y = x\left[\log_2(e)\right]$.

Write $y = y_1 + y_2$, where y_1 is the integer part and y_2 is the fraction part.

If $y \geq 0$, then set $z_1 = y_1$, and $z_2 = y_2$.

If $y < 0$, then set $z_1 = y_1 - 1$, and $z_2 = y_2 + 1$.

Then, $2^y = (2^{z_1})(2^{z_2})$, where z_1 is an integer and $0 \leq z_2 \leq 1$.

2. For $0 \leq z_2 \leq 1$, 2^{z_2} is computed by use of the Chebyshev interpolation polynomial of degree 11 for the interval. The maximum relative error of this polynomial is 2^{-57}.

3. If $x \leq -89.415987$, 0 is given as the answer.

Effect of an Argument Error

$\epsilon \sim \Delta$. Since $\Delta = \delta x$, for the larger value of x even the round-off error of the argument causes a substantial relative error in the answer.

Performance Statistics

Most of the double-precision subroutine performance figures have two sets of statistics for each argument range. Those statistics preceded by an "S" were calculated on machines *not* having the double-precision instruction set; those preceded by a "D" were calculated on machines having the double-precision instruction set.

Argument Range		Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
				7040	7044
$0 \leq x \leq 1$	S	4.24×10^{-17}	1.68×10^{-16}	21008	—
	D	3.74×10^{-17}	1.07×10^{-16}	2619	1289
$\lvert x \rvert \leq 88.028$	S	4.82×10^{-17}	1.96×10^{-16}	21018	—
	D	4.38×10^{-17}	1.39×10^{-16}	2633	1293

Note: The sample arguments upon which the above statistics are based were uniformly distributed over the specified range.

Figure 30. FDXP, Exponential Function (Double-Precision)

Double-Precision Square Root—FDSQ

Algorithm

1. Write $x = 2^{2p-q}(m)$, where p is an integer, $q = 0$ or 1, and $\frac{1}{2} \leq m < 1$. Then

$$\sqrt{x} = 2^p \sqrt{2^{-q}(m)}$$

2. Take the first approximation y_0 to be

$$y = 2^p\left(\frac{m}{2} + \frac{1}{2}\right), \text{ if } q = 0$$

or

$$y = 2^p\left(\frac{m}{2} + \frac{1}{4}\right), \text{ if } q = 1$$

The relative error of y_0 is less than 2^{-4}.

3. Apply the Newton-Raphson iteration four times to y_0 (three times in single-precision and the fourth time in double-precision).

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

The maximum relative error of y_3 is 2^{-53}.

Effect of an Argument Error

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

Performance Statistics

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$10^{-30} \leq x < 10^{37}$	4.26×10^{-17}	1.59×10^{-16}	1346	496

Note: The sample arguments upon which the above statistics are based were exponentially distributed over the specified range.

Figure 31. FDSQ, Square Root Function (Double-Precision)

Double-Precision Logarithm—FDLG

Algorithm

1. Write $x = 2^n(m)$, where $\frac{1}{2} \leq m < 1$. Define the base value F as:

$$F = \frac{1}{2}, \text{ if } \frac{1}{2} \leq m < \frac{\sqrt{2}}{2}$$

$$F = 1, \text{ if } \frac{\sqrt{2}}{2} \leq m < 1$$

Let $z = \frac{m - F}{m + F}$. Then, $m = F \times \left(\frac{1+z}{1-z}\right)$ and $\lvert z \rvert \leq 0.1716$.

$$\log(x) = n\,[\log(2)] + \log(F) + \log\left(\frac{1+z}{1-z}\right)$$

$$\log(F) = -\log(2) \text{ for } \frac{1}{2} \leq m < \frac{\sqrt{2}}{2}$$

$$\log(F) = 0 \text{ for } \frac{\sqrt{2}}{2} \leq m < 1$$

2. $$\log\left(\frac{1+z}{1-z}\right) = 2z\left(1 + \frac{z^2}{3} + \frac{z^4}{5} + \ldots\right)$$
$$\cong z\,(c_0 + c_1 z^2 + \ldots + c_7 z^{14})$$

where the coefficients $c_0, c_1, \ldots, c_7$ are obtained by the Chebyschev interpolation.

The maximum relative error of this approximation is $2^{-59.9}$.

Effect of an Argument Error

$E \sim \delta$. In particular, if δ is the round-off error of the argument, say $\delta \sim 5.6 \times 10^{-17}$, then $E \sim 5.6 \times 10^{-17}$. This means that if the argument is close to 1, the relative error can be very large, since the function value is very small at that point.

Performance Statistics

Argument Range		Root-Mean-Square Absolute Error $\sigma(E)$	Maximum Absolute Error $M(E)$	Average Speed in Microseconds	
				7040	7044
$\frac{1}{2} \leq x \leq 2$	S	1.3×2^{-55}	1.64×2^{-53}	20801	—
	D	1.76×2^{-56}	1.89×2^{-54}	2278	1102

Note: The sample arguments upon which the above statistics are based were uniformly distributed over the specified range.

Figure 32. FDLG, Natural Logarithm Function (I)

Argument Range		Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
				7040	7044
Full Range excluding $\left(\frac{1}{2}, 2\right)$	S	9.65×10^{-17}	2.95×10^{-16}	20827	—
	D	6.09×10^{-17}	1.80×10^{-16}	2307	1123
Note: The sample arguments upon which the above statistics are based were exponentially distributed over the specified range.					

Figure 33. FDLG, Natural Logarithm Function (II)

Argument Range		Root-Mean-Square Absolute Error $\sigma(E)$	Maximum Absolute Error $M(E)$	Average Speed in Microseconds	
				7040	7044
$\frac{1}{2} \leq x \leq 2$	S	1.85×2^{-56}	1.12×2^{-53}	22340	—
	D	1.66×2^{-56}	1.64×2^{-54}	2446	1182
Note: The sample arguments upon which the above statistics are based were uniformly distributed over the specified range.					

● Figure 34. FDLG, Common Logarithm Function (I)

Argument Range		Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
				7040	7044
Full range excluding $\left(\frac{1}{2}, 2\right)$	S	1.18×10^{-16}	4.45×10^{-16}	22370	—
	D	9.10×10^{-17}	3.31×10^{-16}	2475	1203
Note: The sample arguments upon which the above statistics are based were exponentially distributed over the specified range.					

● Figure 35. FDLG, Common Logarithm Function (II)

Double-Precision Sine/Cosine—FDSC

Algorithm

1. If cos(x) is desired, reduce the case to a sine function by

$$\cos(x) = \sin\left(\frac{\pi}{2} - x\right)$$

2. If $x < 0$, use $\sin(-x) = -\sin(x)$. Assume $x \geq 0$. If $|x| < 2^{-26}$, give x as the answer.

3. Divide x by $\frac{\pi}{4}$ and separate the integer part, q_1, and the fraction part, q_2, of the quotient. Let $q_0 = q_1 [\text{mod } 8]$. Then

$$\sin(x) = \sin\left(\frac{\pi}{4} q_0 + \frac{\pi}{4} q_2\right)$$

4. Further reduce the case to the computation of either $\sin(\frac{\pi}{4} r)$ or $\cos(\frac{\pi}{4} r)$ (where $0 \leq r \leq 1$) by the formulas:

$$\sin\left[\frac{\pi}{4}(4 + q_0) + \frac{\pi}{4} q_2\right] = -\sin\left(\frac{\pi}{4} q_0 + \frac{\pi}{4} q_2\right) \quad 0 \leq q_0 < 3$$

$$\sin\left(\frac{\pi}{4} + \frac{\pi}{4} q_2\right) = \cos\left[\frac{\pi}{4}(1 - q_2)\right]$$

$$\sin\left(\frac{\pi}{2} + \frac{\pi}{4} q_2\right) = \cos\left(\frac{\pi}{4} q_2\right)$$

$$\sin\left(\frac{3\pi}{4} + \frac{\pi}{4} q_2\right) = \sin\left[\frac{\pi}{4}(1 - q_2)\right]$$

5. For $0 \leq r \leq 1$,

$$\sin\left(\frac{\pi}{4} r\right) = S_0 r + S_1 r^3 + S_2 r^5 + \cdots + S_6 r^{13}$$

$$\cos\left(\frac{\pi}{4} r\right) = 1 + C_1 r^2 + C_2 r^4 + \cdots + C_6 r^{12}$$

where the coefficients $S_0, \ldots, S_6$ are obtained by the Chebyschev interpolation over the range $0 \leq r \leq 1$, and the coefficients $C_0, \ldots, C_6$ are obtained by the Chebyshev interpolation over the range $-0.01 \leq r \leq 1$.

The maximum relative error of the sine approximation is 2^{-58}. The maximum relative error of the cosine approximation is 2^{-54}.

Effect of an Argument Error

$E \sim \Delta$. As the argument gets larger, Δ grows, and since the value of the function is periodically diminishing, no consistent relative error can be maintained outside the principal range $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$. This holds true for cosine as well.

Performance Statistics

Argument Range		Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
				7040	7044
$\lvert x \rvert \leq \frac{\pi}{2}$	S	1.66×10^{-16}	7.83×10^{-16}	14989	—
	D	9.08×10^{-17}	3.81×10^{-16}	1974	943

Figure 36. FDSC, Double-Precision Sine Function (I)

Argument Range		Root-Mean-Square Absolute Error $\sigma(E)$	Maximum Absolute Error $M(E)$	Average Speed in Microseconds	
				7040	7044
$\lvert x \rvert \leq \frac{\pi}{2}$	S	1.26×2^{-54}	1.26×2^{-52}	14989	—
	D	1.12×2^{-54}	1.00×2^{-52}	1974	943
$\frac{\pi}{2} < \lvert x \rvert \leq 10$	S	1.25×2^{-51}	1.46×2^{-49}	15014	—
	D	1.16×2^{-52}	1.32×2^{-50}	2086	988
$10 < \lvert x \rvert \leq 100$	S	1.27×2^{-48}	1.07×2^{-45}	15055	—
	D	1.08×2^{-49}	1.21×2^{-47}	2085	986

Figure 37. FDSC, Double-Precision Sine Function (II)

Argument Range		Root-Mean-Square Absolute Error $\sigma(E)$	Maximum Absolute Error $M(E)$	Average Speed in Microseconds	
				7040	7044
$0 \leq x \leq \pi$	S	1.68×2^{-54}	1.80×2^{-52}	15924	—
	D	1.52×2^{-54}	1.14×2^{-57}	2106	987
$-20 \leq x < 0$ $\pi < x \leq 20$	S	1.76×2^{-50}	1.26×2^{-47}	16034	—
	D	1.63×2^{-52}	1.33×2^{-49}	2202	1025

Figure 38. FDSC, Double-Precision Cosine Function

Note: The sample arguments upon which the statistics in Figures 36, 37, and 38 are based, were uniformly distributed over the specified ranges.

Double-Precision Arctangent—FDAT

Algorithm

1. Reduce the general case to $0 \leq x \leq 1$ by using the formulas:

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

2. Then reduce the case further to $|x| \leq \tan(15°)$ by using

$$\arctan(x) = 30° + \arctan\left(\sqrt{3} - \frac{4}{x+\sqrt{3}}\right),$$

if $\tan(15°) < x < \tan(45°)$.

3. For the basic range $|x| \leq \tan(15°)$, a continued fraction approximation of the following form is used:

$$\frac{\arctan(x)}{x} = 1 + \frac{\alpha_1 x^2}{\beta_1 + x^2} + \frac{\alpha_2}{\beta_2 + x^2} + \frac{\alpha_3}{\beta_3 + x^2} + \frac{\alpha_4}{\beta_4 + x^2} \quad (*)$$

This formula can be derived by transforming the Taylor series into the continued fraction

$$\frac{\arctan(x)}{x} = 1 - \cfrac{\frac{x^2}{3}}{\frac{3}{5} + x^{-2} - \cfrac{\frac{(3)(4)}{(5^2)(7)}}{\frac{23}{(5)(9)} + x^{-2} - \cfrac{\frac{(5^2)(2^4)}{(7)(9^2)(11)}}{\frac{59}{(9)(13)} + x^{-2} - \cfrac{\frac{(4)(7^2)(3)}{(5)(11)(13^2)}}{\frac{(3)(37)}{(13)(17)} + x^{-2} + w}}}}$$

Then, as the approximation of the value $w = w(x)$, use the value -0.00398, so that when the above fraction is rewritten with this value of w, the formula $(*)$ is obtained. The maximum relative error of the formula $(*)$ is less than 2^{-55}.

4. As a result of the subtraction involved in item 2, computational errors are heaviest for the range $\tan(15°) \leq |x| \leq \tan(45°)$, especially as $|x|$ approaches $\tan(15°)$.

5. DATAN2 provides the extended answer range $-\pi \leq y \leq \pi$, depending on the combination of the signs of the two arguments.

Effect of an Argument Error

$E \sim \frac{\Delta}{1 + x^2}$. For small x, $\epsilon \sim \delta$; and as x increases, the effect of δ on ϵ decreases.

Performance Statistics

Argument Range		Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
				7040	7044
Full range	S	1.87×10^{-16}	1.21×10^{-15}	18427	—
	D	2.12×10^{-16}	1.29×10^{-15}	2375	1137

Note: The sample arguments upon which the above statistics are based were tangents of random numbers uniformly distributed over the interval $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$. (That is, the argument sample was such that the values returned by the function were uniformly distributed over the answer range.)

Figure 39. FDAT, Double-Precision Arctangent Function

Complex Square Root—FCSQ

Algorithm

1. Write $\sqrt{x + iy} = \xi + i\eta$

2. If $x \geq +0$, use $\xi = \sqrt{\frac{|x| + |x + iy|}{2}}$, $\eta = \frac{y}{2\xi}$

3. If $x \leq -0$, use $\xi = \frac{y}{2\eta}$, $\eta = \text{sgn}(y)\sqrt{\frac{|x| + |x + iy|}{2}}$

Thus, if $x < 0$, the case $y = -0$ is differentiated from the case $y = +0$. That is,

$$\sqrt{x-0i} = \lim_{\epsilon \to +0} \sqrt{x-\epsilon i}$$

and

$$\sqrt{x+0i} = \lim_{\epsilon \to +0} \sqrt{x+\epsilon i}$$

4. If, in the foregoing computation,

$$\frac{|x| + |x+iy|}{2} < 2^{-129},$$

then 0 + 0i is given as the answer.

5. The SQR and FCAB subroutines are used by this subroutine.

NOTE: If $|x| + |x+iy| \geq 2^{127}$, floating-point overflow occurs.

Effect of an Argument Error

If we write $x+iy = r(e^{ih})$ and $\sqrt{x}+iy = R(^{iH})$, then $\epsilon(R) \sim \frac{1}{2}\delta(r)$ and $\epsilon(H) \sim \delta(h)$.

Performance Statistics

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$10^{-30} \leq \lvert x_1+ix_2 \rvert \leq 10^{37}$	5.06×10^{-9}	1.56×10^{-8}	2413	849
Note: The distribution of sample arguments upon which the above statistics are based is exponential radially and uniform around the origin.				

Figure 40. FCSQ, Complex Square Root Function

Complex Exponential—FCXP

Algorithm

1. $e^{x+iy} = e^x[\cos(y)] + i[e^x \sin(y)]$

2. The XPN and SCN subroutines are used by this subroutine.

Effect of an Argument Error

If we write $e^{x+iy} = R(e^{iH})$, then $H = y$ and $\epsilon(R) \sim \Delta(x)$.

Performance Statistics

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$\lvert x_1 \rvert \leq 85, \lvert x_2 \rvert \leq 10$	9.34×10^{-9}	2.58×10^{-8}	3885	1528
Note: The sample arguments upon which the above statistics are based were uniformly distributed over the specified range.				

Figure 41. FCXP, Complex Exponential Function

Complex Natural Logarithm—FCLG

Algorithm

1. Write $\log(x+iy) = \xi + i\eta$. Then, $\xi = \log|x+iy|$, and $\eta = \arctan(\frac{y}{x})$ (in the sense of ATAN2).

2. This routine differentiates the argument $x - 0i$ from the argument $x + 0i$. That is,

$$\log(x-0i) = \lim_{\epsilon \to +0} \log(x-\epsilon i)$$

$$\log(x+0i) = \lim_{\epsilon \to +0} \log(x+\epsilon i)$$

3. Subroutines FCAB, ATN, and LOG are use by this subroutine.

NOTE: If $|x+iy| \geq 2^{127}$, floating-point overflow occurs.

Effect of an Argument Error

When we write $x + iy = r(e^{ih})$ and $\log(x+iy) = \xi + i\eta$, we have $h = \eta$ and $E(\xi) \sim \delta(r)$. When the argument is near 1 + 0i, the answer is almost zero; therefore a small δ can cause a large ϵ.

Performance Statistics

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
Full range, except the immediate neighborhood of 1 + 0i	3.59×10^{-9}	1.14×10^{-8}	4406	1656
Note: the distribution of sample arguments upon which the above statistics are based is exponential radially and uniform around the origin.				

Figure 42. FCLG, Complex Natural Logarithm Function

Complex Sine/Cosine—FCSC

Algorithm

1. $\sin(x+iy) = [\sin(x)][\cosh(y)] + [i][\cos(x)][\sinh(y)]$

$\cos(x+iy) = [\cos(x)][\cosh(y)] - [i][\sin(x)][\sinh(y)]$

2. Subroutines SCN and SCNH are used by this subroutine.

Effect of an Argument Error

Combine the effects of the sine and cosine functions of the SCN subroutine and the sinh and cosh functions of the SCNH subroutine according to the algorithm in item 1 above.

Performance Statistics

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$\lvert x_1 \rvert \leq 10$, $\lvert x_2 \rvert \leq 1$	1.68×10^{-8}	(4.15×10^{-8})*	5548	2129

* The maximum relative error cited here is based on a set of 2500 random samples uniformly distributed over the range. In the immediate neighborhood of the points $n\pi + 0i$ (where $n = \pm 1, \pm 2, \ldots$,) the relative error can be quite high in spite of a small absolute error at these points. As x_2 remains constant and $\lvert x_1 \rvert$ increases, the relative error increases; as x_1 remains constant and $\lvert x_2 \rvert$ increases, the relative error remains substantially the same.

Figure 43. FCSC, Complex Sine Function

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$\lvert x_1 \rvert \leq 10$ $\lvert x_2 \rvert \leqq 1$	1.70×10^{-8}	(4.12×10^{-8})*	5498	2119

* The maximum relative error cited here is based on a set of 2500 random samples uniformly distributed over the range. In the immediate neighborhood of the points $(n + \frac{1}{2})\pi + 0i$ (where $n = 0, \pm 1, \pm 2, \ldots$,) the relative error can be quite high in spite of a small absolute error at these points. As x_2 remains constant and $\lvert x_1 \rvert$ increases, the relative error increases; as x_1 remains constant and $\lvert x_2 \rvert$ increases, the relative error remains substantially the same.

● Figure 44. FCSC, Complex Cosine Function

Complex Absolute Value—FCAB

Algorithm

1. If $\lvert x \rvert \geqq \lvert y \rvert$, $\lvert x + yi \rvert = \lvert x \rvert \sqrt{1 + (y/x)^2} + 0i$
2. If $\lvert y \rvert > \lvert x \rvert$, $\lvert x + yi \rvert = \lvert y \rvert \sqrt{1 + (x/y)^2} + 0i$
3. The SQR subroutine is used by this subroutine.

NOTE: If the result is greater than Ω, then floating-point overflow occurs.

Performance Statistics

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
$10^{-30} \leqq \lvert x_1 + ix_2 \rvert \leqq 10^{37}$	6.71×10^{-9}	2.29×10^{-8}	1212	436

Note: the distribution of sample arguments upon which the above statistics are based is exponential radially and uniform around the origin.

Figure 45. FCAB, Complex Absolute Value Function

Complex Arithmetic—FCA

Algorithm

1. $(a + bi) \pm (c + di) = (a \pm c) + (b \pm d)i$
2. $(a + bi)(c + di) = (ac - bd) + (ad + bc)i$
3. If $\lvert c \rvert \geqq \lvert d \rvert$,

$$\frac{(a+bi)}{(c+di)} = \frac{a/c + bd/c^2}{1 + (d/c)^2} + \frac{-ad/c^2 + b/c}{1 + (d/c)^2} \text{(i)}$$

If $\lvert d \rvert > \lvert c \rvert$, then reduce to the above case by transforming

$$\frac{(a+bi)}{(c+di)} = \frac{(b-ai)}{(d-ci)}$$

NOTE: When the magnitude of the result is close to Ω, a floating-point overflow is possible. When the magnitude of the result is close to the underflow threshold, the accuracy of the answer diminishes.

Performance Statistics

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
For pairs of operands that are away from the overflow or underflow thresholds	1.09×10^{-8}	2.62×10^{-8}	883	305

Figure 46. FCA, Complex Multiplication Function

Argument Range	Root-Mean-Square Relative Error $\sigma(\epsilon)$	Maximum Relative Error $M(\epsilon)$	Average Speed in Microseconds	
			7040	7044
For pairs of operands away from the overflow or underflow thresholds	1.3×10^{-8}	3.23×10^{-8}	1444	554

Figure 47. FCA, Complex Division Function

Note: The distribution of sample operands upon which the statistics in Figures 46 and 47 are based is exponential radially and uniform around the origin.

MTN Subroutine

Purpose

The purpose of the MTN subroutine is to modify the error control of the TNCT (single-precision tangent/cotangent)subroutine.

Usage and Effect

The calling sequence for the MTN subroutine is

```
CALL MTAN(k)
```

When this calling sequence is used with k = 0, 1, 2, 3, 4, or 5, the TNCT subroutine is modified to give a minimum relative accuracy guarantee of $1/(2^{k+2}-1)$ for correctly rounded arguments near the singularities. If this calling sequence is used with k greater than 5, TNCT is modified to suspend the above accuracy guarantee feature completely. In this case, any argument less than 2^{20} in magnitude (and greater than 2^{-126} for the cotangent function) will be accepted.

Modifications remain in effect until MTN is called again.

NOTE: A MAP programmer need not use the MTN subroutine to reset the accuracy guarantee. He can accomplish the same effect by including the following coding in his program.

1. To modify the guarantee

```
CLA  = 1
ALS  k
STO  CRIT.
```

   where

   k = 0,1,...,5

2. To eliminate the guarantee

```
STZ  CRIT.
```

Appendix B: Storage Requirements

The following chart shows the octal and decimal storage requirements for the FORTRAN IV Mathematical Subroutines.

SUBROUTINE	STORAGE REQUIREMENTS OCTAL	DECIMAL
ARSCN	117	79
ATN	230	152
ERF	136	94
FCA	131	89
FCAB	44	36
FCLG	45	37
FCSC	110	72
FCSQ	55	45
FCXP	63	51
FDAT	231	153
FDLG	166	118
FDMD	60	48
FDSC	215	141
FDSQ	75	61
FDX1	210	136
FDX2	72	58
FDXP	174	124
GAMA	245	165
LOG	145	101
MTN	20	16
SCN	157	111
SCNH	100	64
SQR	100	64
TNCT	171	121
TNH	76	62
XP1	73	59
XP2	67	55
XP3	51	41
XPN	106	70

Appendix C: Error Messages

Listed below are the off-line error messages corresponding to the FORTRAN IV Mathematical Subroutine Library. The last two digits of the number at the left of each message is the error code. The error code identifies the type of optional return that was taken when the error was detected. For example, error message 10709 signifies that the optional return identified by error code 9 was taken. For further information on error codes and optional returns see the section "Role of the Execution Error Monitor."

```
10701   0**0 INVALID ARGUMENTS
10702   0**-X INVALID ARGUMENTS
10703   TAN-COT(X) WHERE |X| GT OR EQ TO 2**20
10704   TAN-COT(X) WHERE X TOO CLOSE TO A SINGULARITY
10705   SINH-COSH(X) WHERE |X| GREATER THAN 88.029692
10707   -B**C WHERE C IS REAL
10708   EXP(X) WHERE X GREATER THAN 88.029692
10709   ATAN2(0,0) INVALID ARGUMENTS
10710   LOG(0) INVALID ARGUMENT
10711   LOG(-X) INVALID ARGUMENT
10712   SIN-COS(X) WHERE |X| GT OR EQ TO 2**25
10713   ARSIN-ARCOS(X) WHERE |X| GREATER THAN 1
10714   SQRT(-X) INVALID ARGUMENT
10715   CEXP(Z) WHERE Z REAL PART GREATER THAN 88.029692
10716   CEXP(Z) WHERE |Z| IMAG PART GT OR EQ TO 2**25
10717   CLOG(0) INVALID ARGUMENT
10718   CSIN-CCOS(Z) WHERE |Z| IMAG PART GT OR EQ TO 88.029692
10719   CSIN-CCOS(Z) WHERE |Z| REAL PART GT OR EQ TO 2**25
10720   GAMMA(X) WHERE X IS -, NEAR 0, OR GT OR EQ TO 34.843
10721   ALGAMA(X) WHERE X IS NON-POSITIVE OR GREATER THAN 1.54926*2**120
10722   DATAN2(0,0) INVALID ARGUMENTS
10724   DEXP(X) WHERE X GREATER THAN 88.029692
10725   DLOG(0) INVALID ARGUMENT
10726   DLOG(-X) INVALID ARGUMENT
10727   DSIN-DCOS(X) WHERE |X| GT OR EQ TO (2**50)PI
10728   DSQRT(-X) INVALID ARGUMENT
```

Index

CUT ALONG LINE

COMMENT SHEET

IBM 7040/7044 OPERATING SYSTEM (16/32K) SUBROUTINE LIBRARY
FORTRAN IV MATHEMATICAL SUBROUTINES

FORM C28-6806-1

FROM

NAME _____

OFFICE NO. _____

FOLD

CHECK ONE OF THE COMMENTS AND EXPLAIN IN THE SPACE PROVIDED

FOLD

☐ SUGGESTED ADDITION (PAGE)

☐ SUGGESTED DELETION (PAGE)

☐ ERROR (PAGE)

EXPLANATION

FOLD

FOLD

NO POSTAGE NECESSARY IF MAILED IN U.S.A.
FOLD ON TWO LINES, STAPLE, AND MAIL

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 33504

NEW YORK, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
1271 AVENUE OF THE AMERICAS
NEW YORK, N.Y. 10020

ATTN: PROGRAMMING SYSTEMS PUBLICATIONS,
DEPARTMENT D39

FOLD

FOLD

CUT ALONG LINE

STAPLE

STAPLE

Printed in U.S.A. C28-6806-1

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601