UNPL

1074

Bell Laboratories **Cover Sheet for Technical Memorandum**

The information contained herein is for the use of employees of Bell Laboratories and is not for publication. (See GEI 13.9-3)

Title- **A Library of Reference Standard Mathematical Subroutines**

Date- **May 1, 1975**

TM- **75-1271-6**

Other Keywords- **Functions**
Computing

Author	Location	Extension	Charging Case- 39199
Robert Morris	**MH 2C-524**	**3878**	Filing Case- 39199-11

ABSTRACT

There exists on the UNIX time-sharing system a collection of arbitrary precision arithmetic routines and a convenient language BC to call on the routines. A set of mathematical library functions has been written to use the arbitrary accuracy. The functions written to date are

- exponential function
- logarithm
- sine
- cosine
- arctangent
- Bessel function J of integer order

The routines were written in conjunction with a complete error analysis which simplified the design process. The relevant error analysis and the subroutine listings are given.

Pages Text 7	Other 7	Total 14
No. Figures 0	No. Tables 0	No. Refs. 2

BELL TELEPHONE LABORATORIES, INC.

DISTRIBUTION
(REFER GEI 13.9-3)

TM-75-1271-6

COMPLETE MEMORANDUM TO

CORRESPONDENCE FILES

OFFICIAL FILE COPY
PLUS ONE COPY FOR
EACH ADDITIONAL FILING
CASE REFERENCED

DATE FILE COPY
(FORM E-1328)

10 REFERENCE COPIES

AAGESEN,JOHN
<AHO,A V
ARTHURS,EDWARD
ATAL,B S
<BECKER,R A
BIREN,MRS IRMA B
BLUE,J L
BLUM,MRS MARION
BOYCE,W M
BRACKETT,CHARLES A
BRANDENBURG,LANE H
BROWN,W STANLEY
CHADDHA,ROSHAN L
CHAO,MIN-TE
<CHEN,STEPHEN
<CHERRY,MS L L
DONOHUE,B P III
EPSTEIN,MARVIN P
FAULKNER,R A
FERIDUN,K K
<FLEISCHER,HERBERT I
FRANK,MISS A J
<FRASER,A G
FREEMAN,K GLENN
FRIEDMAN,KENNETH A
GAREY,MICHAEL R
<GOLDSTEIN,A JAY
GRAHAM,R L
GRATZER,FRANK J
HALFIN,SHLOMO
<HAMILTON,PATRICIA
HAMMING,R W
<HANNAY,N B
HASKELL,BARRY G
HIGHLAND,PATRICK A
HINDS,EROLD W
HOLTZMAN,JACK M
HUPKA,MRS FLORENCE
JAGERMAN,DAVID L
<JOHNSON,STEPHEN C
KAISER,J F
KATZ,STEVEN S
<KEESE,W M
<KERNIGHAN,BRIAN W
KORNEGAY,R L
KUO,YEN-LONG
LEWINE,ROBERT N
LIOU,MING L
LUDERER,GOTTFRIED W R
LUDWIG,R L

COMPLETE MEMORANDUM TO

MAC WILLIAMS,MRS F J
<MARANZANO,JOSEPH F
<MARKY,MISS G A
<MC GILL,ROBERT
MC ILROY,M DOUGLAS
+MCDONALD,H S
MORGAN,S P
MORRIS,ROBERT
NOE,NORMAN H
<OSSANNA,J F JR
PINSON,ELLIOT N
POLLAK,HENRY O
PRIESTER,R W
+PRIM,ROBERT C
RABINER,LAWRENCE R
<RALEIGH,THOMAS M
ROSENBERGER,JOHN R III
SACKS,ROBERT C
SATZ,L R
<SCHLEGEL,C T
SCHONFELD,TIBOR J
SEMMELMAN,C L
SO,HING C
STEIGERWALT,R A
SWANN,DALE W
TERRY,M E
<THOMSON,M-L
TUTELMAN,DAVID M
<WALFORD,ROBERT B
WARNER,D D
WATSON,D S
<WOLMAN,ERIC
<YAMIN,MRS E E
ZABARSKY,ALAN P
84 NAMES

COVER SHEET ONLY TO

CORRESPONDENCE FILES

4 COPIES PLUS ONE
COPY FOR EACH FILING
CASE

AARON,M R
ACKERMAN,L J
AMRON,I
ANDERSON,JANIS C
APRILLE,T J JR
ARREDONDO,GASTON A
BABAA,IHSAN M H
BADURA,DENNIS C
BAIRD,D R
BAKER,BRENDA S
BARNETT,WILLIAM T
BARTELT,J L
BERGHOLM,J O
BERGLAND,G DAVID
BOAKYE,K A
BOURNE,STEPHEN R

COVER SHEET ONLY TO

BRADFORD,C E
BRITT,WARREN D
BUCHSBAUM,S J
BULFER,ANDREW F
BULLEY,RAYMOND M
BURG,F M
BURNETT,D S
BUTZIEN,PAUL E
<CANADAY,RUDD H
CARRAN,J H
CARROLL,J DOUGLAS
CASTELLANO,MRS M A
CAVINESS,JOHN D
CERMAK,IVAN A
<CHAMBERS,J M
CHAMBERS,MRS B C
CHAWLA,BASANT R
CHICK,ARTHUR J
CHUNG,MRS FAN R K
COLE,LOUIS M
COULTER,J REGINALD
CUTLER,C CHAPIN
DAYEM,RIFAAT A
DE FAZIO,MICHAEL J
DE LESSIO,NOEL X
DEEM,GARY S
DENKMANN,W JOHN
DEUTSCH,DAVID N
DONNELLY,MISS MARY M
DOWELL,RICHARD I
DUTTWEILER,D L
EDELSON,D
EHLINGER,JAMES C
ELLIOTT,R J
ELLIOTT,WILLIAM H III
EVANS,MELVIN J
EVERHART,JOSEPH R
FELDMAN,STUART I
<FELS,ALLEN M
FONTENOT,MICHAEL L
FORD,GERARD A
FORYS,LEONARD J
FOSCHINI,GERARD J
FOWLKES,E B
FOX,PHYLLIS
FRANKS,RICHARD L
FREDERICKS,ALBERT A
FREEMAN,R DON
FRETWELL,LYMAN J
FRIEDENSON,ROBERT A
FROST,H BONNELL
<GATES,G W
GIBB,KENNETH R
GIBSON,ALLEN E
<GILLETTE,DEAN
GNANADESIKAN,R
GOPINATH,B
GORDON,BRIAN G
GORMAN,JAMES E
GRAU,T G
GREENBAUM,H J
GROFF,R H

COVER SHEET ONLY TO

GROSS,ALAN M
GROSS,ARTHUR G
GUANCIAL,EDWARD JR
GUMMEL,HERMANN K
GUST,V J
GUTHERY,SCOTT B
HAISCH,H F JR
HALE,A L
HALLINE,EDWIN G
HALL,ANDREW D JR
HALL,MILTON S JR
<HARKNESS,C J
HARPER,EDWARD Y
HARTMAN,WILLIAM H
HARTWELL,WALTER T
HARUTA,K
<HAWKINS,DONALD T
HAWKINS,RICHARD B
HEJNY,GEORGE J
HELLER,JAMES E
HILBORN,C GENE JR
HILL,DOUGLAS W
HINDERKS,L W
HOADLEY,A BRUCE
HORING,SHELDON
HO,TIEN-LIN
<HYMAN,B
<IVIE,EVAN L
JACKSON,PETER E
JACOBS,H S
JAIN,ARIDAMAN K
JENKINS,JOHN O
<JENSEN,PAUL D
JESSOP,WARREN H
JUDICE,CHARLES N
KANE,MRS ANNE B
KARAFIN,B J
KEILIN,JOSEPH E
KENNY,JOHN J
KNOLL,RONALD L
KNOWLTON,KENNETH
KOSMAN,ROBERT A
KOWAL,C R
KRAMBECK,ROBERT H
KRAMER,STUART A
KRUSKAL,JOSEPH B
LA PADULA,CHARLES A
LABIANCA,FRANK M
LAMBERT,MRS C A
LAMBERT,P F
LANDWEHR,JAMES M
LANZEROTTI,LOUIS J
LAX,M
LESK,MICHAEL E
LIEBESMAN,BURTON S
LINKE,WILLIAM F
LOGAN,JOHN
<LOGAN,MRS V G
LO,CHRISTOPHER C
LUCKY,R W
<LYCKLAMA,HEINZ
MALLOWS,COLIN L

COVER SHEET ONLY TO

MAMMEL,MRS WANDA L
MANDRONE,C D
MATHEWS,MAX V
MAUNSELL,HENRY I G
MAURER,R E
MAYO,J S
MAZIK,R F
<MC CULLOUGH,RICHARD H
MC DONALD,LAWRENCE J
MC DONALD,RICHARD A
MC MAHON,L E
<MC MILLAN,W F
<MC TIGUE,G E
MCDONALD,
MENIST,DAVID B
<MENNINGER,R E
METZ,ROBERT F
MEYER,JACQUES S
MILLER,J A
MILLER,S E
MILTON,JAMES L
MISAWA,T
MOORTHY,SUNDARAM C
MORGEN,DENNIS H
MOTTL,T O
MULLER,ERWIN E
<MUSA,J D
NEAL,SCOTTY R
NETRAVALI,ARUN N
<NINKE,WILLIAM H
NIPPERT,JAMES W
O NEIL,F J
OBENCHAIN,R L
OBERER,ERIC
OLIVE,JOSEPH P
OLSEN,RONALD G
OLSON,HILDING M
OPFERMAN,D C
<ORCHARD,R A
ORCUTT,S E
OWENS,MRS G L
PACK,CHARLES D
PALMER,JOHN F
PAPPAS,A L
PARKER,J C JR
PATEL,C K N
PEOPLES,J T
PERITSKY,MARTIN M
POLONSKY,IVAN P
PORADEK,JAMES R
<RAFSKY,L C
RAGO,L F
REED,S C
REHERT,ALLEN F
RICHMAN,P L
<RIDDLE,GUY G
RITCHIE,DENNIS M
<ROBERTS,CHARLES S
ROCA,RICHARD T
ROSENBLUM,M J
<ROSLER,LAWRENCE
ROVEGNO,MRS HELEN D
...
268 TOTAL

+ NAMED BY AUTHOR > CITED AS REFERENCE < REQUESTED BY READER (NAMES WITHOUT PREFIX WERE SELECTED USING THE AUTHOR'S SUBJECT OR ORGANIZATIONAL SPECIFICATION AS GIVEN BELOW)

ERCURY SPECIFICATION.....

OMPLETE MEMO TO:
127-SUP

IANMAE = MATHEMATICS/NUMERICAL/APPROXIMATION, EXPANSIONS, SERIES

VER SHEET TO:
12-DIR 13-DIR 127

ANM# = MATHEMATICS/NUMERICAL/SURVEY PAPERS ONLY

GET A COMPLETE COPY:

BE SURE YOUR CORRECT ADDRESS IS GIVEN ON THE OTHER SIDE.
FOLD THIS SHEET IN HALF WITH THIS SIDE OUT AND STAPLE.
CIRCLE THE ADDRESS AT RIGHT. USE NO ENVELOPE.

MORRIS,ROBERT
MH 2C524

TM-75-1271-6
TOTAL PAGES 14

PLEASE SEND A COMPLETE COPY TO THE ADDRESS SHOWN ON THE OTHER SIDE
NO ENVELOPE WILL BE NEEDED IF YOU SIMPLY STAPLE THIS COVER SHEET TO THE COMPLETE COPY.
IF COPIES ARE NO LONGER AVAILABLE PLEASE FORWARD THIS REQUEST TO THE CORRESPONDENCE FILES.

Bell Laboratories

Subject: A Library of Reference Standard Mathematical Subroutines
Case- 39199 -- File- 39199-11

date: May 1, 1975

from: Robert Morris

TM: 75-1271-6

MEMORANDUM FOR FILE

Introduction

There exists on the UNIX time sharing system a collection of arbitrary precision arithmetic routines and a convenient language BC to to call on the routines [1,2]. The existence of these programs provided a strong temptation to write a set of mathematical library functions to use the arbitrary accuracy.

The temptation was not resisted and there is now a small reference standard collection of library functions written in BC which is designed to serve

- for computation of constants for inclusion in library routines where, generally, the computations should be done to slightly greater accuracy than the target machine provides.
- for generating test values to validate mathematical library routines. These also must be computed to slightly greater accuracy than the target machine provides.
- for the rare application which needs extra precision.

The Library

The arithmetic routines that are available at the time of writing are:

Function	Name
exponential	e(x)
sine	s(x)
cosine	c(x)
arctangent	a(x)
logarithm	l(x)
Bessel's function	j(n,x)

Usage

The library is accessed by typing the UNIX command

```
bc -l
```

The library is then loaded with a default precision of twenty decimal places. This precision can be changed to *k* decimal places by typing

```
scale = k
```

where k must not exceed 99. Statements can then be typed in order to print or to use values

of the functions as described in [1].

Design Criteria

The purpose of the reference standard library is to provide reference results with predictable error characteristics. The intent was to supply the number of decimal places requested by the user and preferably see to it that they are all correct. Speed of execution was a secondary goal.

It is reasonable to supply function values either with a specified number of significant digits or with a specified number of decimal places. There are genuine advantages and disadvantages to either approach. Because BC is a scaled fixed-point facility, it turned out to be somewhat easier to supply a specified number of decimal places of accuracy. This accuracy can be maintained even when the number is very large.

In BC, there is an internal quantity called **scale** which determines the number of decimal places that are retained in multiplications, divisions, and in other operations that require discarding of digits. The value of **scale** is under control of the BC programmer and it is used by the library as a specification of the number of decimal places of accuracy required.

The details of the scaling operations in BC make it very easy to keep control over the accuracy of computations. In particular, additions and subtractions are never truncated, regardless of the value of **scale**. Therefore sums and differences are exact. The scale of a product is generally truncated if more decimal places would be generated than the value of **scale**. However at least as many decimal places are retained as there are in the operand with the larger number of decimal places. As many decimal places are retained in a square root as there are in the operand, unless the value of **scale** is larger. No arithmetic results are rounded by BC.

The crucial parts of this scaling doctrine are

- that no new errors are introduced by addition and
- that the accuracy of a multiplication can be controlled by controlling the scales of the operands.

Once the decision is made to specify the desired accuracy of a result in terms of a given number of decimal places, then it follows that we have to infer that the user has given the function argument to sufficient accuracy to warrant such accuracy in the result. For example, if we are greeted with

$scale = 20$

$a(1)$

which is a request to print the value of $\pi/4$ to twenty decimals, then we certainly can assume that the argument was intended to be taken as a 1 with lots of zeros after the decimal point (20 in fact).

Conversely, if the argument is given to more decimal places than required to produce the desired accuracy, we can safely throw away the extra digits. For example, the statements

$scale = 10$

$a(142857.142857)$

ask for a 10-decimal place arctangent, but in fact none of the digits after the decimal point are needed to provide the desired accuracy in the result.

The doctrine, then, is to ascribe an accuracy to the argument of just enough decimal places to produce the desired accuracy in the result.

Accuracy Problems

The most convenient general method for computing values of the basic mathematical functions is by summing a power series. The process of addition of series terms leads to truncation errors due to digits of the series terms being thrown away at every step. Besides, there is a series truncation error due to the terms in the series that are ignored and the error is equal to the sum of the ignored terms.

Some of the series expansions that one would like to use do not converge for all values of the argument. Examples are the power series expansions for the logarithm and for the arctangent. In these cases some sort of transformation of the argument is required before summation takes place. As a result, the series sum may need some manipulation to correspond to the initial transformation. Convergence is not the whole problem since, for example, the power series for sine and cosine converge for all arguments but it is much faster to compute these two functions after subtracting multiples of 2π.

To get the desired accuracy it is necessary to take a critical look at every transaction that takes place on the way from the argument to the result.

Fixed-Point Error Analysis

Here is an analysis of the effect of individual arithmetic operations on fixed-point accuracy expressed in terms of decimal places. This analysis is needed to be able to analyze the accuracy of a function as a whole.

The only characteristic of an arithmetic operation that is relevant to this analysis is the derivative of the function (or of an arithmetic operation when viewed as a function).

If the transformation $x \rightarrow f(x)$ has the derivative $f'(x)$, then uncertainties or errors in the value of x are multiplied by $f'(x)$ to produce corresponding uncertainties in the value of $f(x)$. This fact is an immediate consequence of the definition of a derivative.

Now if the number of decimal places in x is ν, then the uncertainty in its value is $10^{-\nu}$. Conversely, an uncertainty of ϵ corresponds to $-\log_{10}\epsilon$ decimal places of accuracy. An uncertainty of ϵ in the value of x produces an uncertainty of $\epsilon f'(x)$ in the value of $f(x)$ supposing ϵ to be small enough to justify ignoring higher order terms. When expressed in terms of places of accuracy, an accuracy of ν decimal places in x corresponds to (i.e. produces) an accuracy of $\nu - \log_{10} f'(x)$.

The analysis that is most appropriate for our application is a retrospective one; how many decimal places of accuracy are needed in an operand (or function argument) to produce given accuracy in a result. If the required accuracy is ν decimal places, then the accuracy required of the argument is

$$\nu + \log_{10} f'(x)$$

if everything else is exact. This requirement holds not only for a function treated as a whole but also for every individual operation that goes into the process of function evaluation.

If we are to estimate the error introduced by computing a function of k variables, we can distinguish two situations which are easy to analyze. If we can assume that all of the error in the result is due to uncertainty in just one of the variables, then the analysis above is appropriate. If on the other hand, we can assume that each of the variables contributes an equal share of error, then an accuracy of ν decimal places in x_1 produces an accuracy of

$$\nu - \log_{10} \frac{\partial}{\partial x_1} f(x_1, \cdots) - \log_{10} k$$

where k is the number of variables x_i.

Intermediate Results

Perhaps the most valuable characteristic of BC for computing mathematical functions is that intermediate results can become exceedingly large without harming the accuracy of the final result.

We must take care that the accuracy of intermediate results is large enough that the desired final accuracy can be obtained. If, for example, we were to compute the identity function $x \rightarrow x$ by first dividing by 10 and then multiplying by 10, then one decimal place of accuracy would be lost in the intermediate result that could not be retrieved.

There are two ways of interpreting these doctrines that are useful lines of thought.

First, if an intermediate result y is obtained for which the derivative dy/dx is less than the derivative $f'(x)$, then accuracy will be lost unless the value of the intermediate result is computed to higher accuracy. The increase of accuracy is the logarithm of the ratio of the two derivatives.

Second, if an intermediate result y is obtained for which the derivative $\partial f(x)/\partial y$ is greater than 1, then the intermediate result y must be known to more accuracy than is required for the result $f(x)$. The increase of accuracy is again the logarithm of the derivative.

The preceding two criteria are in fact identical.

Here are the quantities that we have to work with

(1) The scales of the arguments and all the intermediate results.

(2) The derivatives of the result with respect to all intermediate results.

(3) The scale of the result – this is the amount of information required at the far end of the process.

Multiplication

Suppose that, given x and y, we are to compute the value of $f(x,y) = xy$ to a given number ν of decimal places. To how many decimal places need we compute x and y to produce the required ν places in the result?

The values of the relevant derivatives are

$$\partial f/\partial x = y$$

$$\partial f/\partial y = x$$

The number of decimal places needed in x (if y is exact) is

$$\nu + \log_{10} y$$

and the number needed in y (if x is exact) is

$$\nu + \log_{10} x$$

The errors are additive. If a product of k terms is formed and it is assumed that each factor contributes an equal share of error, then each must be computed to $\log_{10} k$ places of additional accuracy. The same analysis applies to multiplication by a constant. Suppose that, given x, we are to compute the value of $\pi^2 x$ to a given number ν of decimal places. Then π^2 needs to be computed to $\nu + \log_{10} x$ decimal places under the assumption that x is given as accurate.

If x is an intermediate result, then we can obtain the same accuracy by computing π^2 to $\nu + \log_{10} x + \log_{10} 2$ places and x to $\nu + \log_{10} (\pi^2) + \log_{10} 2$ places.

Exponentiation

If the value of x^k is required to ν decimal places, then we can appeal directly to the partial derivative rule and find that x must be known to

$$\nu + \log_{10} k + (k-1)\log_{10} x$$

decimal places. The exponentiation may be computed by repeated multiplication by x and the same rule predicts how may decimal places of each partial result x^j need to be retained.

Addition

The derivatives with respect to x and y of the function $f(x) = x+y$ are both equal to 1, and the errors are purely additive. Just as in the case of multiplication, if we assume that each addend contributes equal error, it is sufficient that each summand be known to $\nu + \log_{10} k$ decimal places.

The Exponential Function

The most obvious route to computing values of the exponential function is to form appropriate partial sums of the power series

$$e^x = \sum_0^\infty \frac{x^k}{k!}$$

which converges for all values of x. It converges rapidly in the sense that eventually the ratio between successive terms approaches zero.

The following scheme will serve to perform the desired computation

(1) Form successive powers x^k of x and successive factorials $k!$

(2) At each step, perform the division $x^k/k!$

(3) Add the quotient from (2) into an accumulating sum.

(4) When we have taken enough terms, we can stop.

The following questions of accuracy arise.

- What accuracy is needed for each x^k?
- To what accuracy need the division be performed?
- To what accuracy should the addition be performed?
- What is the series truncation error?

The additional accuracy that must be ascribed to the argument x in order to obtain ν decimal places in e^x is $\log_{10} e^x$. Then x must be considered to have an accuracy

$$\nu_x = \nu + 0.435x$$

Each of the terms $x^k/k!$ in the summation must be evaluated to a precision greater than ν places and the extra precision depends on the total number k of terms summed.

The total number k of terms in the sum can be estimated by first observing that it requires no more than

$$k_1 = e\,|x|$$

terms for the term $x^{k_1}/k_1!$ to become less than 1 in magnitude (by Stirling's formula) and that from this point on, each successive term is multiplied by a factor smaller in magnitude than $1/e$ so that no more than

$$k_2 = \nu \log 10$$

more terms are needed to reach a term less than $10^{-\nu}$. The error caused by truncation cannot

exceed twice the value of the first neglected term. (The factor is actually $e/(e-1)$).

Therefore the total number of terms required in the series is bounded by

$$k = e\,|x| + \nu \log 10$$

This estimate is very accurate for large x and for small x it doesn't matter much.

We now have enough information to say how to perform the computation without exceeding the error requirement.

x must be treated as if it has $\nu + 0.435x$ places of accuracy and it is convenient (but not necessary) to form all the powers x^j to this accuracy. The division $x^j/j!$ must be performed to give results to $\nu + \log_{10} k$ places (k as defined above) and the summation must be performed to this accuracy.

The Logarithm

$$f(x) = \log(x)$$

The function can conveniently be evaluated by repeatedly taking the square root of the argument x until the result y is in the interval $0.5 < y < 2.0$. Then

$$y = x^{1/p}$$

where p is a power of 2. Then the transformation

$$u = (y-1)/(y+1)$$

reduces u to the range $-1/3 < u < 1/3$ and permits us to evaluate the logarithm by the series

$$t = \tfrac{1}{2} \log y = u + \frac{u^3}{3} + \frac{u^5}{5} + \cdots$$

and then

$$\log x = 2pt$$

In order to control the error in this computation we must consider the chain of computation

$$x \to y \to u \to t \to f(x)$$

The reader will please note that it was an attempt to analyze just this mess that led to the development of the explicit methods of this paper.

Len ν be the number of decimal places required in the answer. By evaluating the derivative at each step we derive the following results for the number of decimal places required in each intermediate result.

$$\nu(x) = \nu - \log_{10} x$$

$$\nu(y) = \nu + \log_{10} p - \log_{10} y$$

$$\nu(u) = \nu + \log_{10} p - \log_{10} y - \log_{10} 2 + 2\log_{10}(y+1)$$

$$\nu(t) = \nu + \log_{10} p + \log_{10} 2$$

By using the known bounds on the value of y, we can find very close bounds for the accuracy required at each step of the calculation. The result is that the entire computation can be done to $\nu + \log_{10} p + 1$ decimal places.

The Sine and Cosine

Since $f'(x) < 1$ for all x, we need only ν decimal places of the argument. We can subtract (or add) multiples of 2π to get the argument into the range $-\pi < x < \pi$ and the multiple must have ν places. So π must be computed (presumably from the arctangent routine) to $\nu + \log_{10} 2x/\pi$ decimal places. We can go further and add or subtract π (as appropriate) to reduce the range of the argument to the interval $-\pi/2 < x < \pi/2$.

Then the sine can be evaluated from the power series

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

The number of terms is bounded in just the same way as for the exponential function and the bound is

$$k = \frac{e\pi}{4} + \nu \log 10$$

Thus the division and the summation need to be performed to $\nu + \log_{10} k$ places and the evaluation of x^j need be done to only ν places.

The cosine is evaluated by

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

The Arctangent

The argument x can be reduced to the range $|x| < .5$ by two applications of the addition law for arctangents, namely, the transformation

$$x \leftarrow \frac{\sqrt{1+x^2} - 1}{x}$$

to form the new argument y. Then $\arctan x = 4 \arctan y$ and we can use the series

$$\arctan y = y - \frac{y^3}{3} + \frac{y^5}{5} - \frac{y^7}{7} + \cdots$$

which converges for $y < 1$. Since the derivative of arctan is not greater than 1, the argument x can be used to ν places. $\arctan y$ is needed to $\nu + \log_{10} 4$ places and therefore this is also enough places for y. In the series, the division and the summation must be done to $\nu + \log_{10} 4 + \nu \log_2 10$.

Bessel's Function J of Integer Order

The definition of the function is

$$J_n(x) = \left(\frac{x}{2}\right)^n \sum_0^\infty \frac{(-x^2/4)^k}{k!\,(n+k)!}$$

The derivative is never greater than 1 in magnitude, so the argument can be used to ν places. The number of terms needed in the summation can be estimated in much the same way as for the exponential function and an upper bound is

$$k = \frac{ex}{2} + \frac{\nu}{2} \log 10 - \frac{n}{2}$$

This expression for k can turn out to be negative and in such cases, the function value is zero to the required number of places.

MH-1271-RM
Atts.
References
Appendix

Robert Morris

References

[1] L. L. Cherry and R. Morris, *BC — An Arbitrary Precision Desk-Calculator Language*, TM 75-1271-5

[2] R. Morris and L. L. Cherry, *DC — An Interactive Desk Calculator*, TM 75-1271-8

APPENDIX

The subroutines whose listings appear in this appendix were designed primarily for accuracy and there was little attempt to make them run fast.

The routines were written at different times with different styles. In most cases the bounds used for the required scales of the computations were simplified.

There are some programming techniques and tricks worthy of mention:

The value of the expression $length(a) - scale(a)$ is $\log_{10}(a)$ rounded up to the next integer. This computation is used constantly to set the internal scale.

The lines

$scale = t$
$a = a/1$

serves to give to the quantity a the scale t.

```
define e(x){
        auto a, b, c, i, s, t
        t = scale

        s = x
        if(s<0) s = -s
        a = 2.72*s + 2.31*t
        a = length(a) - scale(a)

        s = t + .435*x + 1
        if(s<0) s = 0
        scale = s
        x = x/1

        scale = t + a
        a = 1
        b = 1
        s = 1
        for(i=1; 1==1; i++){
                a = a*x
                b = b*i
                c = a/b
                if(c == 0){scale=t; return(s/1)}
                s = s+c
        }
}
```

```
define l(x){
        auto a, b, c, d, e, p, g, u, s, t
        if(x <=0) return(1-10^scale)
        t = scale

        p=1
        scale = scale + scale(x) - length(x) + 1
        s=scale
        while(x > 2){
                s = s + (length(x)-scale(x))/2 + 1
                if(s>0) scale = s
                x = sqrt(x)
                p=p*2
        }
        while(x < .5){
                s = s + (length(x)-scale(x))/2 + 1
                if(s>0) scale = s
                x = sqrt(x)
                p=p*2
        }

        scale = t + length(p) - scale(p) + 1
        u = (x-1)/(x+1)

        scale = scale + 1.1*length(t) - 1.1*scale(t)
        s = u*u
        b = 2*p
        c = b
        d = 1
        e = 1
        for(a=3;1==1;a=a+2){
                b=b*s
                c=c*a+d*b
                d=d*a
                g=c/d
                if(g==e){
                        scale = t
                        return(u*c/d)
                }
                e=g
        }
}
```

```
define s(x){
        auto a, b, c, s, t, y, p, n, i
        t = scale
        y = x/.7853
        s = t + length(y) - scale(y)
        if(s<t) s=t
        scale = s
        p = a(1)

        scale = 0
        if(x>=0) n = (x/(2*p)+1)/2
        if(x<0) n = (x/(2*p)-1)/2
        x = x - 4*n*p
        if(n%2!=0) x = -x

        scale = t + length(1.2*t) - scale(1.2*t)
        y = -x*x
        a = x
        b = 1
        s = x
        for(i=3; 1==1; i=i+2){
                a = a*y
                b = b*i*(i-1)
                c = a/b
                if(c==0){scale=t; return(s/1)}
                s = s+c
        }
}

define c(x){
        auto t
        t = scale
        scale = scale+1
        x = s(x+2*a(1))
        scale = t
        return(x/1)
}
```

```
define a(x){
        auto a, b, c, d, e, f, g, s, t
        if(x==0) return(0)
        t = scale
        f=1
        while(x > .5){
                scale = scale + 1
                x= -(1-sqrt(1.+x*x))/x
                f=f*2
        }
        while(x < -.5){
                scale = scale + 1
                x = -(1-sqrt(1.+x*x))/x
                f=f*2
        }
        s = -x*x
        b = f
        c = f
        d = 1
        e = 1
        for(a=3;1==1;a=a+2){
                b=b*s
                c=c*a+d*b
                d=d*a
                g=c/d
                if(g==e){
                        scale = t
                        return(x*c/d)
                }
                e=g
        }
}
```

```
define j(n,x){
auto a,b,c,d,e,g,i,s,k,t

if(n<0){
        n= -n
        x= -x
        }

        a = x
        if(a<0) a = -a
        t = scale
        k = 1.36*a + 1.16*t - n/2
        k = length(k) - scale(k)
        if(k>0) scale = scale + k

s= -x*x/4
a=1
c=1
for(i=1;i<=n;i++){
        a=a*x
        c = c*2*i
        }
b=a
d=1
e=1
for(i=1;1;i++){
        a=a*s
        b=b*i*(n+i) + a
        c=c*i*(n+i)
        g=b/c
        if(g==e){
                scale = t
                return(g/1)
                }
        e=g
        }
}
```