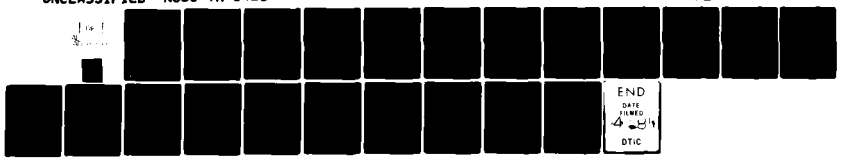
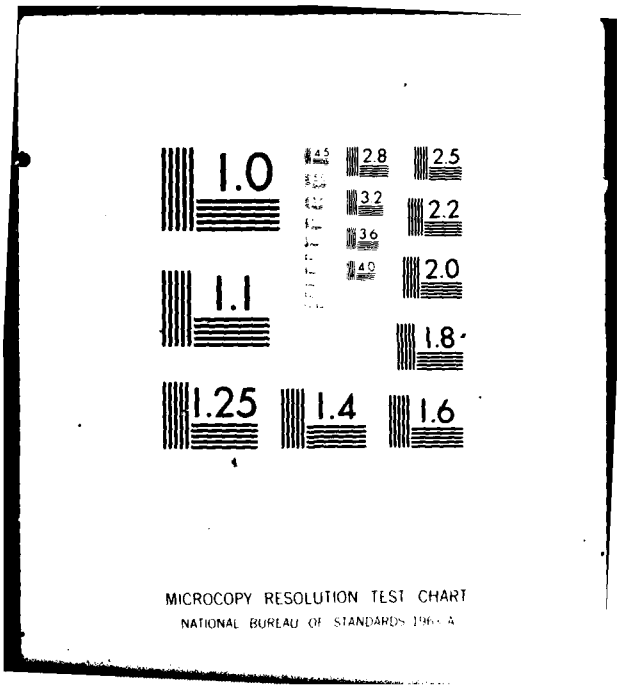
AD-A096 813 NAVAL UNDERWATER SYSTEMS CENTER NEW LONDON CT NEW LO--ETC F/G 9/2
A FUNCTIONALLY EXPANDED COMPUTER PROGRAM FOR COMPUTING MACHINE---ETC(U)
FEB 81 M J GOLDSTEIN

UNCLASSIFIED NUSC-TR-6421 NL



END
DATE FILMED
4-81
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

NUSC Technical Report 6421
12 February 1981

LEVEL II (12)

A Functionally Expanded Computer Program for Computing Machine-Dependent Constants

M.J. Goldstein
Information Services Department

Naval Underwater Systems Center
Newport, Rhode Island / New London, Connecticut

Preface

This report was prepared under NUSC Project No. 771Y00 for Special Projects and Studies.

The Technical Reviewer for this report was Richard Johnson (Code 711).

Acknowledgment

The author is grateful to James Ferrie (Code 325), Rosemary Molino (Code 325), and Nancy Sulinski (Code 7122) for running the computer program contained in Appendix B on the SEL 32/55, ITEL AS/5, and the VAX 11/780. Furthermore, appreciation is due to John Lawson (Code 7122) for developing the algorithm for page length computation.

Reviewed and Approved: 12 February 1981

T. G. Galib

T. Galib
Information Services Department

The author of this report is located at the New London Laboratory, Naval Underwater Systems Center, New London, Connecticut 06320.

(14)

NUSC

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
TR-6421	AD-A096813	

4. TITLE (and Subtitle)

A FUNCTIONALLY EXPANDED COMPUTER PROGRAM FOR COMPUTING MACHINE-DEPENDENT CONSTANTS.

(16)

5. TYPE OF REPORT & PERIOD COVERED

6. PERFORMING ORG. REPORT NUMBER

(10)

7. AUTHOR(s)

M. J. Goldstein

8. CONTRACT OR GRANT NUMBER(s)

Technical rept.

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Naval Underwater Systems Center
New London Laboratory
New London, CT 06320

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

771Y00

11. CONTROLLING OFFICE NAME AND ADDRESS

Naval Underwater Systems Center
Newport, RI 02840

(11)

12. REPORT DATE

12 Feb~~ruary~~ 1981

13. NUMBER OF PAGES

18

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

(12) 31

15. SECURITY CLASS. (of this report)

UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

machine-dependent constants
floating-point number
unit roundoff error
base
mantissa
number of base digits
number of bits in a digit
effective bit precision
page length
smallest positive computer number
largest computer number
program transportability

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In this report we present an updated version of an earlier FORTRAN program that computed machine-dependent constants. The earlier program has been expanded to compute (1) the base (radix) of the computer number system from the system's unit roundoff, (2) the number of digits in the fraction of a floating-point number, and (3) the number of bits in a digit. Furthermore, the program has been expanded to compute the page length of virtual memory machines like the VAX 11/780.

DD FORM 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

405918

20. (Continued)

An illustration is given of how the updated program can be used as an aid in transporting from one computer to another mathematical software that contains machine-dependent floating-point arithmetic parameters.

Table of Contents

A Functionally Expanded Computer Program For Computing Machine-Dependent Constants

Introduction

In a previous paper [1], a computer program was discussed that can be used (independently of any computer vendor claims) as a benchmark to reveal floating-point arithmetic characteristics of a binary device computer having a power of 2 base, B(= 2^k, k a positive integer), and in transporting mathematical computer software that depend on the machine's relative accuracy (computer unit roundoff error) and/or arithmetic range from one computer to another. Furthermore, since the relative accuracy depends on the minimum (effective) number of significant bits in the mantissa (fraction) of a floating-point number and the roundoff property of the computer arithmetic, the program also computed the effective number of bits in the mantissa and whether the arithmetic rounds or chops. However, the computer arithmetic's base B and the number of base B digits in the mantissa of a floating-point number were not computed.

In this report, we present an updated program that computes the base B from the unit roundoff error. The new program also computes the number of base B digits in the mantissa of a floating-point number and the number of bits, namely k, in a base B digit. The updated program calculates the page length of virtual memory computer systems that organize data in virtual memory as the VAX 11/780 does. In order to enhance the program's transportability between computer systems, adjustments have been made to the program code and the algorithm for computing the largest floating-point base B computer number.

Theory and Algorithms

Preliminaries

If s is a non-zero, base B, floating-point, t-digit computer number, we assume its representation is

$$s = \pm B^e f, \qquad -p \leqslant e \leqslant P$$

$$f = .b_1 b_2 \ldots b_t, \quad b_i \in \{0, 1, 2, \ldots, B-1\}, \quad b_1 \neq 0, \tag{1}$$

where B is a positive integer power of 2, namely 2^k, e is in a certain integer range with both p and P positive integers greater than t, and the value of f is given by

$$f = \sum_{i=1}^{t} b_i / B^i . \tag{2}$$

In other words, s is a real number that can be expressed as a normalized ($b_1 \neq 0$) base B number using at most t digits. In particular, if

$$b_i = \begin{cases} 1 & \text{if } i = 1, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

then f assumes its smallest value:

$$f = B^{-1}. \tag{4}$$

If $b_i = B-1$ for $i = 1, 2, \ldots, t$, then f assumes its largest value:

$$f = 1 - B^{-t}. \tag{5}$$

We assume that the internal computer representation of each base B digit b_i is given by k two state devices d_j, where each device takes on the value 0 or 1; that is,

$$b_i = \left(d_k^{(i)} d_{k-1}^{(i)} \ldots d_1^{(i)}\right)_2$$

$$= \sum_{j=1}^{k} d_j^{(i)} 2^{j-1}, \quad d_j^{(i)} \in \{0, 1\}. \tag{6}$$

In particular, if $f = 2^{-r}$ for $r = 1, 2, \ldots, k$, then

$$b_1 = 2^{k-r} = (\overbrace{0 \ldots 0}^{r-1} 1 \overbrace{0 \ldots 0}^{k-r})_2$$

$$b_i = 0 = (\overbrace{0 \ldots 0}^{k})_2, \qquad i = 2, \ldots, t. \tag{7}$$

Therefore, the number of bits in the fraction (mantissa) f is kt. Thus, for a floating-point word of fixed length, say N bits with kt bits in the mantissa, N-kt-1 bits are reserved for the exponent e (stored as the excess quantity $e + a$, $a \geq 0$) and the remaining bit is reserved for the sign of f. Furthermore, since $b_1 \neq 0$, there can be as few as $kt-k+1$ significant (effective) bits in the fraction f.

By (1) and (4) the smallest positive real number in the computer number set is

$$m = B^{-p} B^{-1}, \tag{8}$$

whereas the largest positive real number is

$$M = B^{p} (1-B^{-t}) \tag{9}$$

by (1) and (5). In a computation, the magnitude of results smaller than m produce underflow (usually returning a base B, t-digit representation for zero); the magnitude of results larger than M produce overflow.

The computer unit roundoff error u, which measures how closely a real number y can be approximated by a computer number s, is related to the error in s by the inequality

$$|y-s| \leq |s| u , \tag{10}$$

where

$$u = \begin{cases} B^{1-t} & \text{(chops)} \\ 0.5B^{1-t} & \text{(rounds).} \end{cases} \tag{11}$$

The computer arithmetic chops if all of the mantissa's low order digits b_i beyond the t-th are simply discarded in the normalized base B, infinite digit representation of y; rounds if b_t is incremented by 1 whenever $b_{t+1} \geq 2^{k-1}$, before discarding the low order digits.

Computation of the Base

In the earlier program, we computed the unit roundoff error when $B\ (=2^k)$ and t are unknown. (The theory and algorithm for this is in Appendix A.) The following Theorem and Corollaries lead to an algorithm for computing the base B from the unit roundoff error.

Theorem. In the interval $[B^{e-1}, B^e]$, where $-p \leq e \leq P$, the floating-point numbers are uniformly spaced with spacing B^{e-t}.

Proof. Let $B^e f$ be a floating-point number that is in the interval $[B^{e-1}, B^e]$, such that

$$f = .\ b_1 \ldots b_t, \quad B^{-1} \leq f < 1. \tag{12}$$

To obtain the next floating-point number in the interval, we add the base B digit one to the last digit b_t of f; that is, the next floating-point, base B, t-digit number in $[B^{e-1}, B^e]$ is

$$B^e (f + .\ b_1' \ldots b_t'), \tag{13}$$

where

$$b_i' = \begin{cases} \overbrace{(0 \ldots 0)_2}^{k} & \text{if } i = 1,2, \ldots, t-1, \\ \overbrace{(0 \ldots 0\ 1)_2}^{k-1} & \text{if } i = t, \end{cases}$$

and

$$.\ b_1' \ldots b_t' = \sum_{i=1}^{t} b_i'/(2^k)^i = (2^k)^{-t} = B^{-t}. \tag{14}$$

This completes the proof.

Corollary 1. The floating-point, base B, t-digit numbers in the interval $[B^{t-1}, B^t]$ are uniformly spaced with spacing $B^0 = 1$.

Proof. Immediate.

But the value of every floating-point, base B, t-digit number in $[B^{t-1}, B^t]$ is an integer. Therefore, every integer in $[B^{t-1}, B^t]$ can be represented exactly as a floating-point, base B, t-digit number, since B^{t-1} can be. In particular, the integers $2^L B^{t-1}$ for $L = 0, 1, 2, \ldots, k$, have exact floating-point, base B, t-digit representations.

Corollary 2. The floating-point, base B, t-digit numbers in the interval $[B^t, B^{t+1}]$ are uniformly spaced with spacing B.

Proof. Since $B^t = B^{(t+1)-1}$, set $e = t + 1$ in the Theorem. This completes the proof.

Thus, although B^t can be represented exactly as a floating-point, base B, t-digit number $B^t + 1$ cannot be. Then how is $B^t + 1$ approximated in the computer number set?

Consider that

$$\begin{aligned} B^t + 1 &= B^{t+1} B^{-1} + B B^{-1} \\ &= B^{t+1} B^{-1} + B^{t+1} (B^{-t} B^{-1}) \\ &= B^{t+1} (B^{-1} + B^{-t-1}) \\ &= B^{t+1} (.\, b_1 \ldots b_t\, b_{t+1}), \end{aligned} \tag{15}$$

where

$$b_i = \begin{cases} 1 & \text{for } i = 1, t+1 \\ 0 & \text{for } i = 2, \ldots, t\,. \end{cases}$$

Therefore, if the computer arithmetic rounds, the base B, t-digit, floating-point approximation to $B^t + 1$ is given by

$$B^t \oplus 1 = B^{t+1} .\, b_1 \ldots b_{t-1}\, b_t^* \tag{16}$$

with

$$b_t^* = \begin{cases} 1 & \text{if } B = 2, \\ 0 & \text{if } B = 2^k \text{ for } k \geq 2\,. \end{cases}$$

On the other hand, if the computer arithmetic chops, then for $B = 2^k$ ($k = 1, 2, \ldots$) the base B, t-digit, floating-point approximation to the sum $B^t + 1$ is given by (16) with $b_t^* = 0$.

Hence, the computer arithmetic will approximate the real sum $B^t + 1$ by the computer number B^t, when the computer arithmetic rounds if $B = 2^k$ for $k \geq 2$; if $B = 2$, then the sum is approximated by $B^t + B$. On the other hand, if the computer arithmetic chops, it approximates the sum $B^t + 1$ by B^t for all values of B that are positive integer powers of two.

Therefore, given the unit roundoff error u in (11), since its reciprocal u^{-1} is an integer in the interval $\{B^{t-1}, B^t\}$, and

$$2^L u^{-1} = B^t$$

when

$$L = \begin{cases} k & \text{(chops)} \\ k-1 & \text{(rounds)}, \end{cases} \tag{17}$$

we have the following algorithm.

Algorithm B

Given the unit roundoff u, the effective precision, say NBIT, and whether the computer arithmetic rounds or chops, the algorithm for computing B, the binary length of a base B digit (namely k) and the number of base B digits in the mantissa (namely t) is:

B1. Set k to zero (written symbolically as k←0).

$s \leftarrow u^{-1}$

$s_2 \leftarrow s$

B2. Do while $(s_2 + 1 \neq s_2)$.

$s_2 \leftarrow 2s_2$

$k \leftarrow k + 1$

End Do while

B3. $B \leftarrow s_2/s$

B4. If (machine rounds and B>2)

$B \leftarrow 2B$

$k \leftarrow k + 1$

End if

B5. $t \leftarrow (NBIT + k - 1)/k$

Largest Computer Number

In the earlier program [1] in order to compute the largest computer number

$$M = B^P (1 - B^{-t}), \tag{18}$$

we begin by taking the reciprocal of the smallest positive computer number

$$m = B^{-p-1}. \tag{19}$$

If the reciprocal does not cause overflow, the program outputs an adjusted value of the reciprocal as an approximation to M. Otherwise, we find the reciprocal of the product of m by the smallest power of two that does not cause overflow. This gives

$$M = B^{P}(0.5). \tag{20}$$

Then we output an adjusted value of this result as an approximation to M.

The problem with this algorithm is that it requires invoking a routine (not available on all systems) to test whether an overflow occurs. We have corrected this in the following way.

Assume that the relation between the smallest and largest exponents of B is given by

$$p = P + 1. \tag{21}$$

This is so on the UNIVAC 1108. On some machines the relation is p = P; e.g., VAX 11/780. Then with $p = P + 1$

$$m = B^{-p-1} = B^{-P-2} \tag{22}$$

and

$$m^{-1} = B^{P+2} = B^{P+3}B^{-1}, \tag{23}$$

so that

$$(B^3m)^{-1}(1-B^{1-t})B = B^{P}(1-B^{1-t}) \tag{24}$$

approximates M without computer arithmetic overflow, provided the order of operation is performed from left to right as indicated in (24). For if one were to compute $(B^3m)^{-1}B$ when $p = P + 1$, the result B^P would cause computer overflow.

Similarly, if p = P, then

$$(B^3m)^{-1}(1-B^{1-t})B = B^{P-1}(1-B^{1-t}) \tag{25}$$

so that the approximation is off by B.

Algorithm M

Thus, given m, B and t, the algorithm for approximating M without computer overflow is

M1. Compute B^{-m} and store it in M.

M2. Compute the reciprocal of M and store the result in M.

M3. Compute and store the product $M(1-B^{1-t})$ in M.

M4. Compute and store the product MB in M. Output M.

Page Length Computation

The idea behind the algorithm for determining page length is based on the fact that local data and program code are stored by the VAX 11/780 on separate pages on secondary computer memory, with program code following local data.* Therefore, if the number of computer words, e.g., I, required by local data occupies less than a page of secondary memory, then the first non-zero location following the Ith local data word is a program instruction.

Algorithm PL

Thus, dimensioning an integer array, e.g., IPGLN, as having length 1, and assuming the number of local data items is I, the algorithm for the page length is

PL1. NMBDT ← I + 1

PL2. J ← NMBDT

PL3. Do while (IPGLN (J) = 0)

J ← J + 1

End Do while

PL4. J ← J-1

The final value of J is the page length.

Program Description

The updated single precision program that incorporates Algorithms B, M and PL is given in Appendix B. Algorithm u in Appendix A for computing the unit roundoff error, which is encoded in the earlier program [1], is implemented in the new program without any logical changes.

Algorithm PL for computing page length is implemented in the program for systems like the VAX 11/780 with page length as great as 2048 words, where the local data occupies no more than 17 single precision words for the single precision

*Local data refers to data only within the program module that defines them; e.g., data items in DATA type statements and intermediate results calculated and stored by the program module.

version of the program, and no more than 29 single precision words for the double precision version of the program.

Unlike the earlier program in [1], the new program is written in ANSI 66 FORTRAN [2] (with the exception of the PRINT statement that is ANS 77 [3]) to promote compilability on computers that have different FORTRAN compilers. Furthermore, the program is designed so that it can be converted easily to double precision — simply remove the comment flag C from column 1 of comment lines 5, 6, 7, 86 and 87 in the program, and convert lines 8, 84 and 85 to comment lines.

Since the program requires no input data, it is convenient to use in determining the values of the machine-dependent parameters discussed in this report.

The program in Appendix B has compiled and executed successfully on the UNIVAC 1108, VAX 11/780, SEL 32/55 and ITEL AS/5.* The output for the double precision version of the program on the UNIVAC 1108 and VAX 11/780 appear in tables 1 and 2 below, where NBIT is the effective bit precision of a floating-point computer number; BASE is the computer arithmetic's base B; NBDGT is the number of base B digits in the fraction of a floating-point computer number; LNGH is the number of bits in a base B digit; and RNCHOP is the arithmetic's rounding property — 0 if the arithmetic chops results, 1 if it rounds. The number of bits in the mantissa of a floating-point computer number is the product of NBDGT by LNGH.

Table 1. UNIVAC 1108 Double Precision

```
UNIT ROUNDOFF ERROR =      .173472348-17      NBIT =  60      RNCHOP = 0.

BASE =   2.  NBDGT =   60  LNGH =    1

SMALLEST F. P. NUMBER =     .278134232313-308
APPROXIMATE LARGEST F. P. NUMBER =     .898846567431+308
PAGE LENGTH =   29
```

Table 2. VAX 11/780 Double Precision

```
UNIT ROUNDOFF ERROR =   0.13877787E-16      NBIT =  56      RNCHOP = 1.

BASE =   2.  NBDGT =   56  LNGH =    1

SMALLEST F. P. NUMBER =   0.29387358770E-38
APPROXIMATE LARGEST F. P. NUMBER =   0.85070591730D+38
PAGE LENGTH =  126
```

*The page length computation does not work on the ITEL AS/5, since the ITEL system does not organize local data in virtual memory as the VAX does.

An Application in Program Transportability

Consider transporting a computer program from one computer to another, whose execution depends on the values of machine-dependent floating-point arithmetic parameters in the program. In general, if the parameters are not reset to the new machine's values, the transported program will not execute properly. The program in Appendix B, or its double precision version, may be used to determine the new parameter values.

For example, consider the segment of a double precision program [4] in Appendix C for calculating Bessel functions $J_\nu(x)$ and $I_\nu(x)$ of real argument and integer order, which is available on NUSC's UNIVAC 1108. The program contains several machine-dependent parameters that are explained in the program's commentary and are set to the double precision values of the UNIVAC 1108 in the DATA statement at line 75 of the program. Trying to execute this program on the VAX will fail as a result of computer arithmetic overflow if the DATA statement values of the parameters have not been modified to reflect the VAX's double precision arithmetic. These parameter values can be changed to reflect the VAX's values with the aid of the program in Appendix B. First, run the double precision version of the program in Appendix B on the VAX. Then from its output (in table 2) and the explanation of machine-dependent constants in the Bessel function subroutine in Appendix C, one obtains the following DATA statement for the VAX:

```
DATA NSIG,NTEN,LARGEX,EXPARG/17,37,10000,87.D0/
```

where EXPARG is the natural logarithm of the approximation to the largest floating-point number in table 2, rounded to the nearest integer.

Conclusion

The program for computing machine-dependent constants can be used as a benchmark for revealing features of floating-point number systems on binary device computers, where the base of the number system is a positive integer power of two, the mantissa is a normalized fraction and the computer arithmetic has at least one guard digit. Furthermore, the program can be used as an aid in transporting mathematical software containing machine-dependent parameters from one computer to another. In fact, it is possible to facilitate the transportation of such software between many different computers by incorporating in the software program instructions that automatically compute the appropriate machine values for the parameters.

References

1. Marvin J. Goldstein and Richard Johnson, "A FORTRAN Program for Determining Computer Arithmetic Characteristics," NUSC TM No. 781074, April 1978.

2. *American Standard FORTRAN X3.9 1966*, Business Equipment Manufacturers Association, March, 1966.

3. *American National Standard Programming Language FORTRAN, ANSI X3.9 1977*, American National Standards Institute, New York, NY, 1978.

4. David J. Sookne, "Bessel Functions of Real Argument and Integer Order," *J. Res. Nat. Bur. Stand. (U.S.) 77B (Mat. Sci)*, Nos. 3 & 4, 124-132 (July/December 1973).

Appendix A

Theory and Algorithm for Unit Roundoff: Base B and t Unknown

By Theorem 1, the spacing of floating-point, base B, t-digit computer numbers in $[B^0, B]$ is B^{1-t}. Therefore, $B^0 + B^{1-t}$ is in the computer number set; furthermore, the real numbers

$$B^0 + vB^{1-t}, \qquad v = 1,2, \ldots, B^{t-1}(B-1)$$

are in $[B^0, B]$ and the base B, computer number set; and, therefore, so are the numbers

$$B^0 + 2^wB^{1-t}, \qquad w = 0,1,2, \ldots, k(t-1)-1,$$

where

$$2^wB^{1-t} = 2^{-L}, \qquad L = k(t-1)-w,$$

are negative powers of two that are in the computer number set, by (7).

However,

$$\begin{aligned} y = B^0 + 2^{-1}B^{1-t} &= BB^{-1} + B^{1-t}(2^{k-1}/B) \\ &= B(B^{-1} + 2^{k-1}/B^{t+1}) \\ &= B(.b_1 \ldots b_t b_{t+1}). \end{aligned}$$

with

$$b_i = \begin{cases} \overbrace{(0 \ldots 0}^{k-1}\ 1)_2 = 1 & \text{if } i = 1, \\ \overbrace{(0 \ldots 0}^{k})_2 = 0 & \text{if } i = 2, \ldots, t, \\ (1\ \overbrace{0 \ldots 0}^{k-1})_2 = 2^{k-1} & \text{if } i = t+1, \end{cases}$$

is a real number that is not in the computer number set although $2^{-1}B^{1-t}$ is. The real number y is approximated by a computer number s, namely

$$y \doteq s = \begin{cases} B(.b_1 \ldots b_t) & \text{(chops)} \\ B(.b_1 \ldots b_t') & \text{(rounds)} \end{cases}$$

with

$$b_i^{-} = (\overbrace{0 \ldots 0}^{k-1} 1)_2$$

provided the computer has a guard digit in its arithmetic hardware registers for b_{i-1}. Hence

$$s = \begin{cases} 1 & \text{(chopped arithmetic)} \\ 1 + B^{1-t} & \text{(rounded arithmetic)}. \end{cases}$$

Thus, the algorithm for computing the computer arithmetic's relative accuracy u, effective precision, and rounding property is

Algorithm u

u1. Compute $1 \oplus \varepsilon (\varepsilon = 2^{-L})$ for
$L = 1,2 \ldots$, until either $1 \oplus \varepsilon = 1$ or $(1 \oplus \varepsilon) \ominus \varepsilon \neq 1$

u2. If $(1 \oplus \varepsilon) \ominus \varepsilon \neq 1$, then $u = \varepsilon$ and the machine rounds; otherwise $u = 2\varepsilon$ and the machine chops.

u3. The minimum number of significant bits in the mantissa is the last value of L computed in u1, namely $k(t-1) + 1$.

Appendix B

Computer Program for Calculating Machine-Dependent Constants

```
M   [illegible]UNITFF
C      ****************************************************************
C        REVEALING F. P. ARITH FEATURES BY M. [illegible] GOLDSTEIN
C      ****************************************************************
         DIMENSION IB[illegible]
         DOUBLE PRECISION ZERO,HALF,ONE,TWO,UNIT,UNITA,DIFF,RNCHOP,
C       * SAVE,SAVE2,BASE
C        DATA ZERO, HALF, ONE, TWO, NMBDT 0.0D0,.5D0,1.0D0,2.0D0,30.
         DATA ZERO, HALF, ONE, TWO, NMBDT 0.0,0.5,1.0,2.0,18.
         UNIT = ONE
         NBIT = ZERO
C
C     ***DETERMINE UNIT ROUND OFF, EFFECTIVE NO. OF BITS, ROUNDING PROP.***
C
C        DO UNTIL((UNITA.LE.ONE).OR.((UNITA - UNIT).NE.ONE))
       5     UNIT = HALF*UNIT
             UNITA = UNIT + ONE
             NBIT = NBIT + 1
             DIFF = UNITA - UNIT
             IF((UNITA.GT.ONE).AND.DIFF.EQ.ONE)GO TO 5
C        CONTINUE
         IF(DIFF.EQ.ONE)GO TO 6
             GO TO 8
C        THEN MACHINE CHOPS
       6     UNIT = TWO*UNIT
             RNCHOP = ZERO
             GO TO 9
C        ELSE MACHINE ROUNDS
       8     RNCHOP = ONE
       9 CONTINUE
         PRINT 50, UNIT, NBIT, RNCHOP
C
C     ***DETERMINE BASE, NO. OF BASE DIGITS IN MANTISSA, DIGIT LNGH***
C
         LNGH = 0
         SAVE = ONE UNIT
         SAVE2 = SAVE
C        DO WHILE((SAVE2 + ONE).NE.SAVE2)
      10     IF((SAVE2 + ONE).EQ.SAVE2)GO TO 12
             SAVE2 = TWO*SAVE2
             LNGH = LNGH + 1
             GO TO 10
C        CONTINUE
      12 BASE = SAVE2/SAVE
         IF((RNCHOP.EQ.ONE).AND.(BASE.GT.TWO))GO TO 13
             GO TO 14
C        THEN
      13     BASE = TWO*BASE
             LNGH = LNGH + 1
C        CONTINUE
      14 NBDGT = (NBIT + LNGH - 1)/LNGH
         PRINT 52, BASE, NBDGT, LNGH

C     ***COMPUTE SMALLEST F. P. NO.***
C
      15 IF(UNIT.NE.ZERO.AND.UNITA.NE.UNIT)GO TO 20
             GO TO 25
C        THEN SMALLEST F. P. NO. NOT REACHED
      20     UNITA = UNIT
             UNIT = HALF*UNIT
             GO TO 15
C        ELSE PRINT SMALLEST F. P. NO.
      25     PRINT 55, UNITA
         CONTINUE
C
C     ***APPROXIMATE LARGEST F. P. NO.***
```

```
      C
         30 UNIT = UNITA*BASE**3
            UNIT = ONE/UNIT
            UNIT = UNIT*(ONE - BASE**(1 - NBDGT))
            UNIT = UNIT*BASE
      C     CONTINUE
            PRINT 60, UNIT
      C
      C  ***DETERMINE PAGE LENGTH***
      C
            DO 35 J = NMBDT, 2050
                IF(IPGLN(J).NE.0)GO TO 40
         35 CONTINUE
         40 J = J - 1
            IF(J.LT.2049)PRINT 70, J
         50 FORMAT(1H ,22HUNIT ROUNDOFF ERROR = ,E17.9,5X,7HNBIT = ,I3,
           X 5X,9HRNCHOP = ,F2.0 //)
         52 FORMAT(1H ,7HBASE = ,F4.0,10H  NBDGT = ,I4,9H  LNGH = ,I4//)
         55 FORMAT(1H ,24HSMALLEST F. P. NUMBER = ,E17.9)
         60 FORMAT(1H ,35HAPPROXIMATE LARGEST F. P. NUMBER = ,E17.9)
      C  55 FORMAT(1H ,24HSMALLEST F. P. NUMBER = ,D20.12)
      C  60 FORMAT(1H ,35HAPPROXIMATE LARGEST F. P. NUMBER = ,D20.12)
         70 FORMAT(1H ,14HPAGE LENGTH = ,I4)
            STOP
            END
```

Appendix C

Double Precision Program for Calculating Bessel Function $J_\nu(x)$ and $I_\nu(x)$

```
M.UG*STKAMC(1).BESSEL
      1            SUBROUTINE BESSEL(X, NB, IZE, B, NCALC)
      2      C  THIS ROUTINE CALCULATES BESSEL FUNCTIONS I AND J OF REAL
      3      C  ARGUMENT AND INTEGER ORDER.
      4      C
      5      C
      6      C     EXPLANATION OF VARIABLES IN THE CALLING SEQUENCE
      7      C
      8      C  X  DOUBLE PRECISION REAL ARGUMENT FOR WHICH I*S OR J*S
      9      C     ARE TO BE CALCULATED.  IF I*S ARE TO BE CALCULATED,
     10      C     ABS(X) MUST BE LESS THAN EXPARG (WHICH SEE BELOW).
     11      C NB  INTEGER TYPE.  1 + HIGHEST ORDER TO BE CALCULATED.
     12      C     IT MUST BE POSITIVE.
     13      C IZE INTEGER TYPE.  ZERO IF J*S ARE TO BE CALCULATED, 1
     14      C     IF I*S ARE TO BE CALCULATED.
     15      C B   DOUBLE PRECISION VECTOR OF LENGTH NB, NEED NOT BE
     16      C     INITIALIZED BY USER.  IF THE ROUTINE TERMINATES
     17      C     NORMALLY (NCALC = NB), IT RETURNS J(OR I)-SUB-ZERO
     18      C     THROUGH J(OR I)-SUB-NB-MINUS-ONE OF X IN THIS
     19      C     VECTOR
     20      C NCALC INTEGER TYPE, NEED NOT BE INITIALIZED BY USER.
     21      C     BEFORE USING THE RESULTS, THE USER SHOULD CHECK THAT
     22      C     NCALC=NB, I.E. ALL ORDERS HAVE BEEN CALCULATED TO
     23      C     THE DESIRED ACCURACY.  SEE ERROR RETURNS BELOW.
     24      C
     25      C
     26      C     EXPLANATION OF MACHINE DEPENDENT CONSTANTS
     27      C
     28      C NSIG  DECIMAL SIGNIFICANCE DESIRED.  SHOULD BE SET TO
     29      C       IFIX(ALOG10(2)*NBIT + 1), WHERE NBIT IS THE NUMBER OF
     30      C       BITS IN THE MANTISSA OF A DOUBLE PRECISION VARIABLE.
     31      C       SETTING NSIG LOWER WILL RESULT IN DECREASED ACCURACY
     32      C       WHILE SETTING NSIG HIGHER WILL INCREASE CPU TIME
     33      C       WITHOUT INCREASING ACCURACY.  THE TRUNCATION ERROR
     34      C       IS LIMITED TO T=.5*10**-NSIG FOR J*S OF ORDER LESS
     35      C       THAN ARGUMENT, AND TO A RELATIVE ERROR OF T FOR I*S
     36      C       AND THE OTHER J*S
     37      C NTEN  LARGEST INTEGER K SUCH THAT 10**K IS MACHINE-
     38      C       REPRESENTABLE IN DOUBLE PRECISION.
     39      C LARGEX UPPER LIMIT ON THE MAGNITUDE OF X.  BEAR IN MIND
     40      C       THAT IF ABS(X)=N, THEN AT LEAST N ITERATIONS OF THE
     41      C       BACKWARD RECURSION WILL BE EXECUTED.
     42      C EXPARG LARGEST DOUBLE PRECISION ARGUMENT THAT THE LIBRARY
     43      C       DEXP ROUTINE CAN HANDLE.
     44      C
     45      C
     46      C                ERROR RETURNS
     47      C
     48      C
     49      C       LET G DENOTE EITHER I OR J.
     50      C       IN CASE OF AN ERROR, NCALC.NE.NB, AND NOT ALL G*S
     51      C    ARE CALCULATED TO THE DESIRED ACCURACY.
     52      C       IF NCALC.LT.0, AN ARGUMENT IS OUT OF RANGE.  NB.LE.0
     53      C    OR IZE IS NEITHER 0 OR 1 OR IZE=1 AND ABS(X).GE.EXPARG.
     54      C    IN THIS CASE, THE B VECTOR IS NOT CALCULATED, AND NCALC
     55      C    IS SET TO MIN0(NB,0)-1 SO NCALC.NE.NB.
     56      C       NB.GT.NCALC.GT.0 WILL OCCUR IF NB.GT.MAGX AND ABS(G-
     57      C    SUB-NB-OF-X/G-SUB-MAGX OF-X).LT.10**-NTEN/2), I.E. NB
     58      C    IS MUCH GREATER THAN MAGX.  IN THIS CASE, B(N) IS CALCU-
     59      C    LATED TO THE DESIRED ACCURACY FOR N.LE.NCALC, BUT FOR
     60      C    NCALC.LT.N.LE.NB, PRECISION IS LOST.  IF N.GT.NCALC AND
     61      C    ABS(B(NCALC)/B(N)).EQ.10**-L, THEN ONLY THE FIRST NSIG-N
     62      C    SIGNIFICANT FIGURES OF B(N) MAY BE TRUSTED.  IF THE USER
     63      C    WISHES TO CALCULATE B(N) TO HIGHER ACCURACY, HE SHOULD USE
     64      C    AN ASYMPTOTIC FORMULA FOR HIGH ORDER.
     65      C
```

```
      C
            DOUBLE PRECISION
           1 X,B,F,TEST,TEMPA,TEMPB,TEMPC,EXPARG,SIGN,SUM,TOVER,
           2 PLAST,POLD,PSAVE,PSAVEL,ZERO,TENTH,HALF,QUART,ONE,TWO,TEN
           3,SROOT,DUM
      C    4,UNIT,KNCHOF,SMALL,BIG,CMLOG
            DIMENSION B(NB)
            DATA ZERO,TENTH,QUART,HALF,ONE,TWO,TEN/0.0D0,.1D0,.25D0,.5D0,
           1 2.D0,1.D1/
            DATA NSIG,NTEN,LARGEX,EXPARG/19,307,100000,7.D2/
```

Initial Distribution List

ADDRESSEE	NO. OF COPIES
Dep. USDR&E (Res. & Adv. Tech.) (R. M. Davis)	1
Dep. USDR&E (Dir. Elect. & Phys. Sc.) (L. Wiseberg)	1
OASN, Dept. Assit. Sec. (Res. & Adv. Tech.) (Dr. R. Hoglund)	1
ONR (ONR-200, -102, -212, -222, -230)	5
CNO (OP-02, -03-EG, -090, -098, -224, -35, -902, -951, -96, -961, -981G1, -981H, -982F, -983	14
CNM (MAT-08T2, -08T21, -08T24, SP-20, ASW-122, ASW-124)	6
DIA (DT-2C)	1
NAVSURFWEAPCTR, White Oak	1
DWTNSRDC ANNA	1
DWTNSRDC CARD	1
DTNSRDC BETHESDA	1
NRL	1
NRL, USRD	1
NRL, AESD	1
NORDA (Code 110, Dr. R. Goodman)	1
USOC (Code 241, 240)	2
NAVSUBASE, New London	1
NABSUBSUPACNLON	1
NOO (Code 02)	1
NAVELECSYSCOM (ELEX 00, 03, 304, 310, 5101, PME-108, -117)	7
NAVSEASYSCOM (SEA-003, -05R, -511, -06, -06D, -06R, -06V, -06Z, -61, -61R, -62, -62R, -63, -63R, -63R-1, -63R-13, -63X, -631X, -631Y, -632X, -902)	21
NAVSEASYSDET, Norfolk	1
NASC (AIR 610)	1
NAVAIRDEVCEN (Code 00, 2052)	2
NOSC (Code 00, Code 6565)	2
NAVWPNSCEN	1
NAVCOASTSYSLAB	1
CIVENGRLAB	1
NAVSURFWPNCEN	1
CHESNAVFACENGCOM (FPO-1P3)	1
APL/UW, Seattle	1
ARL/Penn State	1
CENTER FOR NAVAL ANALYSES	1
DTIC	12
DARPA	1
NOAA/ERL	1
NATIONAL RESEARCH COUNCIL	1
WEAPON SYS EVAL GROUP	1
WOODS HOLE OCEANOGRAPHIC INST.	1
ENGINEERING SOCIETIES LIBRARY	1
MARINE PHYSICAL LAB, SCRIPPS	1


LMED
8