X-552-72-147

PREPRINT

NASA TM X- 65971

IBM SYSTEM/360 ASSEMBLY LANGUAGE INTERVAL ARITHMETIC SOFTWARE

APRIL 1972

GSFC

GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND

X-552-72-147

IBM SYSTEM/360
ASSEMBLY LANGUAGE
INTERVAL ARITHMETIC SOFTWARE

E. J. Phillips
Computing and Software, Inc.
Data Systems Division

April 1972

Goddard Space Flight Center
Greenbelt, Maryland

CONTENTS

ACKNOWLEDGMENT

The author wishes to gratefully acknowledge the assistance of Dr. Ronald Tost and other members of the Gesellschaft Für Mathemetik Und Datenverarbeitung, Bonn, Germany, who prepared and helped implement the software package on the IBM 360 system.

IBM SYSTEM/360
ASSEMBLY LANGUAGE
INTERVAL ARITHMETIC SOFTWARE

INTRODUCTION

This document describes some computer software designed to perform interval arithmetic. An interval is defined as the set of all real numbers between two given numbers including or excluding one or both endpoints. Interval arithmetic consists of the various elementary arithmetic operations defined on the set of all intervals, such as interval addition, subtraction, union, etc. The theoretical aspects of interval arithmetic are described in reference (2).

One of the main applications of interval arithmetic is in the area of error analysis of computer calculations. For example, it has been used successfully to compute bounds on sounding errors in the solution of linear algebraic systems[2], error bounds in numerical solutions of ordinary differential equations,[1][3] as well as integral equations and boundary value problems. The described software should enable users to implement algorithms of the type described in these references efficiently on the IBM 360 system.

The subroutines in this package were supplied to Goddard Space Flight Center by Dr. Ronald Tost, Institute für Numerische Datenverarbeitung under an exchange agreement with Dr. C. E. Velez, GSFC Code 552. Although the subroutines were coded in IBM 360 assembly language, they were designed so that calls may be made from FORTRAN programs. The sample problem in Section VII and all of the examples in Sections I to VI are given in FORTRAN to illustrate how this package may be used in FORTRAN programs. The subroutines were tested under varying conditions by Computing and Software, Inc., personnel.

The following pages explain in detail the arithmetic operations performed by this package.

INTERVAL ARITHMETIC PACKAGE

I. Language

Assembler Language Code for IBM/360.

II. Purpose

This package is designed to perform arithmetic operations using floating point (64 bits) interval arithmetic.

III. Method

The interval members are defined as COMPLEX * 16, by the user. The basic arithmetic operations are performed along with some basic set theory applications. The arithmetic functions are: addition, subtraction, multiplication, division, exponentiation. The absolute value of an interval and a negative interval can be created. Real and integer numbers can be converted into interval numbers. The intersection and union between two sets of interval numbers can be found. An illegal interval can be tested for. There are tests to determine if zero is contained in (ϵ) an interval or if one interval is contained in another. The sum of two sets of interval numbers can be found by means of a scalar product. The above operations are accomplished by the following functions:

DLI, DRE, IT, DT, $ADD, $SUB, $MUL, $DIV, $POT, $MIN, $ABS, $SCH, $VER, IZER, ICOR, $ENT, $SUM, $INP.

IV. Calling Sequence

The interval numbers and functions are defined as COMPLEX * 16.

In the following set: A is an INTERVAL (AL, AR)
B is an INTERVAL (BL, BR)
C is an INTERVAL (CL, CR)
D: REAL * 8
N: INTEGER * 4

Functions:

1. D = DLI(A) D: = AL Left Corner of A
   Ex. A = (2.5, 4.0)
   D = DLI(A) = 2.5

2. D = DRE(A) D: = AR Right Corner of A
   Ex. A = (2.5, 4.0)
   D = DRE(A) = 4.0

3. A = IT(N) A: = (N, N) Produce an interval from an integer
   Ex. N = 5
   A = IT(N) = (5.0, 5.0)

4. A = DT(D) A: = (D, D) Produce an interval from a real number
   Ex. D = 2.52
   A = DT(D) = (2.52, 2.52)
   A = DT (.5DO) = (0.5D0, 0.5D0)

5. A = $ADD (B, C) A: = B + C Addition
   Ex. B = (3.0, 4.0)
   C = (2.0, 2.0)
   A = $ADD (B, C) = (3.0 + 2.0, 4.0 + 2.0)
   = (5.0, 6.0)
   B = (-1.5, -0.5)
   C = (2.5, 4.5)
   A = $ADD (B, C) = (-1.5 + 2.5, -0.5 + 4.5)
   = (1.0, 4.0)

6. A = $SUB (B, C) A: = B - C Subtraction
   Ex. B = (3.0, 4.0)
   C = (2.0, 2.0)
   A = $SUB (B, C) = (3.0 - 2.0, 4.0 - 2.0)
   = (1.0, 2.0)
   B = (3.0, 4.0)
   C = (-1.5, -2.5)
   A = $SUB (B, C) = (3.0 - (-1.5), 4.0-(-2.5))
   = (4.5, 6.5)

7. A = $MUL (B, C) A: = B * C Multiplication
   Ex. B = (3.0, 4.0)
   C = (2.0, 2.5)
   A = $MUL (B, C) = (MIN (6.0, 7.5, 8.0, 10.0), MAX (6.0, 7.5, 8.0, 10.0))
   = (6.0, 10.0)

8. A = $DIV (B, C) A: = B/C Division
   Ex. B = (3.0, 4.0) if 0 $\notin$ (B, C)
   C = (1.0, 2.0)
   A = $DIV (B, C) = (3.0, 4.0) * (0.5, 1.0)
   = (MIN (1.5, 3.0, 2.0, 4.0), MAX (1.5, 3.0, 2.0, 4.0))
   = (1.5, 4.0)

9. A = $DOT(B, N) A: = B**N Exponentiation
   Ex. B = (3.0, 4.0)
   N = 2
   A = $DOT (B, N) = (9.0, 16.0)

10. A = $MIN(B) A: = - B Negative Interval
   B = (3.0, 4.0)
   A = $MIN(B) = (-4.0, -3.0)

11. A = $ABS(B) A: = B if BL > 0 Absolute Value
   A: = - B if BR < 0 else
   A: = (0, MAX(BL, BR))

Examples:

(1) B = (4.0, 4.25)
   A = $ABS(B) = (4.0, 4.25)

(2) B = (-2.0, -1.0)
   A = $ABS(B) = (1.0, 2.0)

(3) B = (-1.5, 4.0)
   A = $ABS(B) = (0, MAX (-1.5, 4.0))
   = (0, 4.0)

12. A = $SCH(B, C) A: = (MAX((CL, BL), MIN(CR, BR)) Intersection
   Ex. B = (4.0, 5.0) $SCH can produce illegal intervals. This can be
   C = (-1.5, 2.0) proved with ICOR.
   A = $SCH(B, C) = (MAX (4.0, -1.5), MIN (5.0, 2.0)
   = (4.0, 2.0) (- illegal interval)
   B = (1.0, 2.0)
   C = (-1.5, 5.0)
   A = $SCH(B, C) = (1.0, 2.0)

13. A = $VER(B, C) A: = (MIN(CL, BL), MAX(CR, BR)) Union
   B = (1.5, 2.5)
   C = (1.0, 5.0)
   A = $VER(B, C) = (MIN(1.0, 1.5), MAX(5.0, 2.5))
   = (1.0, 5.0)

14. IF (IZER (A)) ^, ^^, ^^^ Test to see if zero (0) is contained in A
   If IZER(A) < 0 GO TO^
   If IZER(A) = 0 GO TO^^
   If IZER(A) > 0 GO TO^^^

Test to see:

ˆ : if AR < 0 (if AR < 0 , 0 $\notin$ A)
ˆˆˆ : if AL > 0 (if AL > 0 , 0 $\notin$ A)
ˆˆ : else 0 $\in$ A (zero contained in A) null interval

15. IF (ICOR(A)) ˆ, ˆˆ, ˆˆˆ Interval Test
IF ICOR(A) < 0 GO TOˆ
IF ICOR(A) = 0 GO TOˆˆ
IF ICOR(A) > 0 GO TOˆˆˆ

Test to see:

ˆ : if AL > AR (illegal interval)
ˆˆ : if AL = AR
ˆˆˆ : if AL < AR

16. $ENT – Test on being contained in
CALL $ENTC A, B, &ˆ1, &ˆ2, &ˆ3, &ˆ4)
ˆ1: AL > BL and AR < BR (A is contained in B)
ˆ2: AL = BL and AR = BR (A and B are equal)
ˆ3: AL < BL and AR > BR (B is contained in A)
ˆ4: A and B are not fully contained in one another or are not contained in each other at all.

17. $SUM – Build a sum with a scalar product.
CALL $SUM(A, I, J, K) where I = J, K

$SUM store A, I, J, K and sets up a loop made to perform addition.

Along with the call to $SUM a multiplication statement has to follow immediately.

```
CALL $SUM (A, I, J, K)
D = $MUL (B, C)
```

Ex:
A = (0.0, 0.0)
B(1) = (1.0, 2.0) C(1) = (2.0, 3.0)
B(2) = (3.0, 4.0) C(2) = (4.0, 5.0)
B(3) = (5.0, 6.0) C(3) = (6.0, 7.0)

Example:

J = 1, K = 3

```
CALL $SUM (A, I, 1, 3)
D = $MUL (B(I), C(I))
```

```
CALL $SUM (A, I, J, K)
  D = $MUL (B(I), C(I))
```

Correspondent

```
AKKU = A
DO 1  I = 1, 3
```

D = (0.0, 0.0) + (2.0, 6.0)
= (2.0, 6.0) + (12.0, 20.0)
= (14.0, 26.0) + (30.0, 42.0)
= (44.0, 68.0)

```
1  AKKU = AKKU + $MUL (B(I), C(I))
   D = AKKU
```

D, AKKU, and A are intervals; B, C are interval arrays. The sequence of $SUM and $MUL cannot contain interval functions.

18. $INP A = $INP ('⟨INPUT⟩') conversion routine

⟨INPUT⟩ is an interval number, that is produced from an alphanumeric field of input values surrounded by quotation marks. The routine creates an interval from the set of numbers in the alphanumeric field by choosing the smallest number for the left endpoint and the largest number for the right endpoint. The interval number is greater than or equal to 1 (≥ 1) in absolute value. The real or integer numbers are separated by commas but can contain no blanks. The beginning and end of the interval number must contain at least one blank. The left side of the output is the minimum of the real or integer numbers. The right side of the output contains the maximum of the real or integer numbers.

Example:

```
A = $INP ('␣ 1.3, 1.56␣')   → A = (1.3, 1.56)
A = $INP ('␣ -2., 4␣')      → A = (-2., 4.)
A = $INP ('1␣')             → No answer a blank doesn't preceed "1" on
                              left side
A = $INP ('␣ 1E45, 23, 2.5␣') → A = (2.5, 1.E45)
```

The arithmetic functions can produce exponent overflow and divide check interrupts. Certain interval functions can produce illegal intervals and only two of the subroutines in this package can determine if the output is illegal, IZER and ICOR.

In all the arithmetic interval functions the input is tested to determine if it is illegal. No error messages are generated just a return to the calling program.

DEFINITION OF ⟨INPUT⟩:

```
⟨INPUT⟩ : = ^ ⟨INTER⟩ ^
⟨INTER⟩ : = ⟨REAL⟩
        : = <INTER>, ⟨REAL⟩
 REAL   : = ⟨FLOAT⟩
        : = ⟨FLOAT⟩ ⟨EX⟩ ⟨INTEG⟩
```

⟨FLOAT⟩ : = ⟨INTEG⟩
: = ⟨INTEG⟩ . ⟨ZIFF⟩

⟨INTEG⟩ : = ⟨SIGN⟩ ⟨ZIFF⟩

⟨ZIFF⟩ : = ⟨DIGIT⟩
: = ⟨ZIFF⟩ ⟨DIGIT⟩

⟨DIGIT⟩ : = 1 2 3 4 5 6 7 8 9 Ø < >

⟨SIGN⟩ : = + - < >

⟨EX⟩ : = E
: = D

^ At least one blank.

< > The empty string.

V. MATHEMATICS USED IN PACKAGE

If * is one of the symbols +, -, ·, /, the arithmetic operations of intervals are defined by:

$$[a, b] * [c, d] = \{x * y \mid a \leq x \leq b, c \leq y \leq d\}$$

$$[a, b] + [c, d] = [a + c, b + d]$$

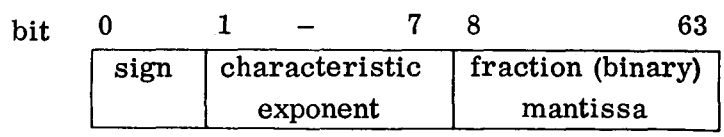
$$[a, b] - [c, d] = [a - d, b - c]$$

$$[a, b] \cdot [c, d] = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)]$$

and if $0 \notin [c, d]$ then

$$[a, b] / [c, d] = [a, b] \cdot \left[\frac{1}{d}, \frac{1}{c}\right]$$

Normalized floating point interval arithmetic is used in this package. Under IBM O/S 360 a floating point number is represented as



where bit = 0 sign of number (0-positive, 1 negative)

1 - 7 = the exponent indicating the power of 16 by which the fraction is multiplied. A zero in bit 1 indicates negative exponent, 1 indicates positive exponent. It is coded in excess 64-notation (decimal). That is, subtract decimal equivalent of characteristic by 64 to obtain the actual characteristic.
e.g. $\underline{0\,|\,1000000\,| = 64}$
$64 - 64 = 0 = 16$

8 - 63 = fraction or mantissa. The binary points of the fraction is just before bit position 8. That is to say that a 1 in bit position 8 represents 2^{-1}, a 1 in bit position 9 represents 2^{-2}, etc.

The magnitudes of a floating point number is 16^{-65} (5.4×10^{-79})
$(1 - 16^{-14}) \cdot 16^{63}$ (7.2×10^{75})

Example:

1 1000001 01010100 00000000 00000000 00000000 etc.

Sign Characteristic

(-1) 65

$65 - 64 = 1$

$$- 1 \times 16^1 \times (2^{-2} + 2^{-4} + 2^{-6}) = -2^4 (2^{-2} + 2^{-4} + 2^{-6})$$
$$= - (2^2 + 2^0 + 2^{-2})$$
$$= - (4 + 1 + 0.25) = -5.25$$

Normalized floating point numbers have a non-zero high order hexadecimal fraction digit. One or more high order hexadecimal fraction digits which are zero makes the number unnormalized. Normalization consists of shifting the fraction left until the high order hexadecimal digit is non-zero and reducing the characteristic by the number of hexadecimal digits shifted. For multiplication and division the operands are normalized before the operations are performed. For the remaining operands (+, -) the result is normalized.

A = 436D1000 00000000 = normalized number

B = 4306D100 00000000 = unnormalized

= 426D1000 0000000 after normalization.

VI. RESTRICTIONS

1. Only basic arithmetic operations and basic set theory applications are performed. No trigonometric operations are performed.

2. Uses long (64 bits) floating point interval arithmetic.

3. The arithmetic performed can produce illegal intervals.

4. The arithmetic performed can cause arithmetic overflow and divide check interrupts.

```
        COMPILER OPTIONS - NAME=  MAIN,OPT=01,LINECNT=58,SIZE=0000K,
                           SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
ISN 0002            IMPLICIT COMPLEX*16(A-H,$),REAL*8(O-Z),INTEGER(I-N)
ISN 0003            DIMENSION C(20),D(20),E(20)
ISN 0004            WRITE(6,101)
ISN 0005            I=1
ISN 0006            P=0.5D0
ISN 0007            A=(2.0D0,5.0D0)
ISN 0008            B=(1.0D0,8.0D0)
ISN 0009            N=3
ISN 0010            V=DLI(A)
ISN 0011            S=DRE(A)
ISN 0012            T=DLI(B)
ISN 0013            U=DRE(B)
ISN 0014            WRITE(6,203) V,S,T,U,N,P
ISN 0015            WRITE(6,222)
ISN 0016            D(1)=$IT$(N)
ISN 0017            WRITE(6,108)
ISN 0018            R=DLI(D(I))
ISN 0019            S=DRE(D(I))
ISN 0020            WRITE(6,2002) I,R,S
ISN 0021            I=I+1
ISN 0022            D(2)=$DT$(P)
ISN 0023            WRITE(6,109)
ISN 0024            R=DLI(D(I))
ISN 0025            S=DRE(D(I))
ISN 0026            WRITE(6,2002) I,R,S
ISN 0027            I=I+1
ISN 0028            D(3)=$ADD(A,B)
ISN 0029            WRITE(6,102)
ISN 0030            R=DLI(D(I))
ISN 0031            S=DRE(D(I))
ISN 0032            WRITE(6,2002) I,R,S
ISN 0033            I=I+1
ISN 0034            D(4)=$SUB(A,B)
ISN 0035            WRITE(6,103)
ISN 0036            R=DLI(D(I))
ISN 0037            S=DRE(D(I))
ISN 0038            WRITE(6,2002) I,R,S
ISN 0039            I=I+1
ISN 0040            D(5)=$MUL(A,B)
ISN 0041            WRITE(6,104)
ISN 0042            R=DLI(D(I))
ISN 0043            S=DRE(D(I))
ISN 0044            WRITE(6,2002) I,R,S
ISN 0045            I=I+1
ISN 0046            D(6)=$DIV(A,B)
ISN 0047            WRITE(6,105)
ISN 0048            R=DLI(D(I))
ISN 0049            S=DRE(D(I))
ISN 0050            WRITE(6,2002) I,R,S
ISN 0051            WRITE(6,223)
ISN 0052            I=1
ISN 0053            C(1)=$POT(A,N)
ISN 0054            WRITE(6,110)
ISN 0055            R=DLI(C(I))
```

```
ISN 0056              S=DRE(C(I))
ISN 0057              WRITE(6,2002) I,R,S
ISN 0058              I=I+1
ISN 0059              C(2)=$POT(B,N)
ISN 0060              WRITE(6,111)
ISN 0061              R=DLI(C(I))
ISN 0062              S=DRE(C(I))
ISN 0063              WRITE(6,2002) I,R,S
ISN 0064              I=I+1
ISN 0065              C(3)=$MIN(A)
ISN 0066              WRITE(6,112)
ISN 0067              R=DLI(C(I))
ISN 0068              S=DRE(C(I))
ISN 0069              WRITE(6,2002) I,R,S
ISN 0070              I=I+1
ISN 0071              C(4)=$MIN(B)
ISN 0072              WRITE(6,113)
ISN 0073              R=DLI(C(I))
ISN 0074              S=DRE(C(I))
ISN 0075              WRITE(6,2002) I,R,S
ISN 0076              I=I+1
ISN 0077              C(5)=$ABS(A)
ISN 0078              WRITE(6,114)
ISN 0079              R=DLI(C(I))
ISN 0080              S=DRE(C(I))
ISN 0081              WRITE(6,2002) I,R,S
ISN 0082              I=I+1
ISN 0083              C(6)=$ABS(B)
ISN 0084              WRITE(6,115)
ISN 0085              R=DLI(C(I))
ISN 0086              S=DRE(C(I))
ISN 0087              WRITE(6,2002) I,R,S
ISN 0088              I=I+1
ISN 0089              C(7)=$SCH(A,B)
ISN 0090              WRITE(6,107)
ISN 0091              R=DLI(C(I))
ISN 0092              S=DRE(C(I))
ISN 0093              WRITE(6,2002) I,R,S
ISN 0094              IF(ICOR(C(7))) 10,20,20
ISN 0095        10    IF(R.GT.S) WRITE(6,190) R,S
ISN 0097              I=I+1
ISN 0098        20    C(8)=$VER(A,B)
ISN 0099              WRITE(6,106)
ISN 0100              R=DLI(C(I))
ISN 0101              S=DRE(C(I))
ISN 0102              WRITE(6,2002) I,R,S
ISN 0103              IF(IZER(I)) 28,30,29
ISN 0104        28    IF (S.LT.0) GO TO 35
ISN 0106        29    IF(R.GT.0) GO TO 35
ISN 0108        30    WRITE(6,210)
                C TEST ON BEING CONTAINED IN
ISN 0109        35    WRITE(6,217)
ISN 0110              CALL $ENT(A,B,&36,&37,&38,&39)
ISN 0111        36    WRITE(6,211)
ISN 0112              GO TO 40
ISN 0113        37    WRITE(6,212)
```

```
ISN 0114              GO TO 40
ISN 0115        38    WRITE(6,213)
ISN 0116              GO TO 40
ISN 0117        39    WRITE(6,214)
ISN 0118        40    B=(0.0D0,0.0D0)
ISN 0119              WRITE(6,101)
ISN 0120              T=DLI(B)
ISN 0121              U=DRE(B)
ISN 0122              WRITE(6,203) V,S,T,U,N,P
ISN 0123              WRITE(6,217)
ISN 0124              CALL $ENT(A,B,&41,&42,&43,&44)
ISN 0125        41    WRITE(6,211)
ISN 0126              GO TO 44
ISN 0127        42    WRITE(6,212)
ISN 0128              GO TO 45
ISN 0129        43    WRITE(6,213)
ISN 0130              GO TO 45
ISN 0131        44    WRITE(6,214)
ISN 0132        45    CONTINUE
ISN 0133              CALL $SUM(B,I,1,5)
ISN 0134              F=$MUL(C(I),D(I))
ISN 0135              R=DLI(F)
ISN 0136              S=DRE(F)
ISN 0137              WRITE(6,218)
ISN 0138              WRITE(6,215) R,S
                C     CONVERSION ROUTINE
ISN 0139              C(1)=$INP( ' 1.3,1.56 ')
ISN 0140              C(2)=$INP( ' -2.,4 ' )
ISN 0141              C(3)=$INP('1 ')
ISN 0142              C(4)=$INP( ' 1E45,23,2.5 ')
ISN 0143              C(5)=$DT$(5.5D0)
ISN 0144              C(6)=$IT$(9)
ISN 0145              WRITE(6,216)
ISN 0146              DO 9 I=1,6
ISN 0147              R=DLI(C(I))
ISN 0148              S=DRE(C(I))
ISN 0149        9     WRITE(6,2002) I,R,S
ISN 0150      2002    FORMAT(1X,I5,'LEFT= ',D24.16,' RIGHT= ',D24.16)
ISN 0151       101    FORMAT('1INTERVAL ARITHMETIC PACKAGE')
ISN 0152       102    FORMAT('0A+B')
ISN 0153       103    FORMAT('0A-B')
ISN 0154       104    FORMAT('0A*B')
ISN 0155       105    FORMAT('0A/B')
ISN 0156       106    FORMAT('0A UNION B')
ISN 0157       107    FORMAT('0A INTERSECT B')
ISN 0158       108    FORMAT('0$IT$(N)')
ISN 0159       109    FORMAT('0$DT$(P)')
ISN 0160       110    FORMAT('0A**N')
ISN 0161       111    FORMAT('0B**N')
ISN 0162       112    FORMAT('0$MIN(A)')
ISN 0163       113    FORMAT('0$MIN(B)')
ISN 0164       114    FORMAT('0$ABS(A)')
ISN 0165       115    FORMAT('0$ABS(B)')
ISN 0166       190    FORMAT(' THIS IS AN ILLEGAL INTERVAL (',G24.16,',',G24.16,')')
ISN 0167       203    FORMAT(1X,'A= ',D24.16,',',D24.16/1X,'B= ',D24.16,',',D24.16/
                     1 1X,'N= ',I5/1X,'P= ',D24.16)
```

```
ISN 0168      210    FORMAT(' ZERO IS CONTAINED IN THE FOLLOWING INTERVAL (',G24.16,',' 
                    1G24.16,')')
ISN 0169      211    FORMAT('  A IS CONTAINED IN B')
ISN 0170      212    FORMAT('  A AND B ARE EQUAL')
ISN 0171      213    FORMAT('  B IS CONTAINED IN A')
ISN 0172      214    FORMAT('  A AND B ARE NOT FULLY CONTAINED IN EACH OTHER')
ISN 0173      215    FORMAT(' LEFT= ',D24.16,2X,'RIGHT= ',D24.16)
ISN 0174      216    FORMAT('0$INP CONVERSION ROUTINE')
ISN 0175      217    FORMAT('0$ENT-TEST ON BEING CONTAINED IN')
ISN 0176      218    FORMAT('0SUM PRODUCED WITH SCALAR PRODUCT')
ISN 0177      222    FORMAT('0VALUES FOR ARRAY D(I)')
ISN 0178      223    FORMAT('0VALUES FOR ARRAY C(I)')
ISN 0179             STOP
ISN 0180             END
```

```
INTERVAL ARITHMETIC PACKAGE
A=    0.20000000000000000D 01,   0.80000000000000000D 01
B=    0.0                    ,   0.0
N=      3
P=    0.50000000000000000D 00

$ENT-TEST ON BEING CONTAINED IN
 A AND B ARE NOT FULLY CONTAINED IN EACH OTHER

SUM PRODUCED WITH SCALAR PRODUCT
LEFT=  -0.68500000000000000D 02  RIGHT=    0.87300000000000000D 03

$INP CONVERSION ROUTINE
     1LEFT=    0.13000000000000000D 01 RIGHT=    0.15600000000000000D 01
     2LEFT=   -0.20000000000000000D 01 RIGHT=    0.40000000000000000D 01
     3LEFT=   -0.20000000000000000D 01 RIGHT=    0.40000000000000000D 01
     4LEFT=    0.25000000000000000D 01 RIGHT=    0.10000000000000000D 46
     5LEFT=    0.55000000000000000D 01 RIGHT=    0.55000000000000000D 01
     6LEFT=    0.90000000000000000D 01 RIGHT=    0.90000000000000000D 01
```

```
INTERVAL ARITHMETIC PACKAGE
A=    0.2000000000000000D 01,  0.5000000000000000D 01
B=    0.1000000000000000D 01,  0.8000000000000000D 01
N=     3
P=    0.5000000000000000D 00

VALUES FOR ARRAY D(I)

$IT$(N)
     1LEFT=    0.3000000000000000D 01 RIGHT=    0.3000000000000000D 01

$DT$(R)
     2LEFT=    0.5000000000000000D 00 RIGHT=    0.5000000000000000D 00

A+B
     3LEFT=    0.3000000000000000D 01 RIGHT=    0.1300000000000000D 02

A-B
     4LEFT=   -0.6000000000000000D 01 RIGHT=    0.4000000000000000D 01

A*B
     5LEFT=    0.2000000000000000D 01 RIGHT=    0.4000000000000000D 02

A/B
     6LEFT=    0.2500000000000000D 00 RIGHT=    0.5000000000000000D 01

VALUES FOR ARRAY C(I)

A**N
     1LEFT=    0.8000000000000000D 01 RIGHT=    0.1250000000000000D 03

B**N
     2LEFT=    0.1000000000000000D 01 RIGHT=    0.5120000000000000D 03

$MIN(A)
     3LEFT=   -0.5000000000000000D 01 RIGHT=   -0.2000000000000000D 01

$MIN(B)
     4LEFT=   -0.8000000000000000D 01 RIGHT=   -0.1000000000000000D 01

$ABS(A)
     5LEFT=    0.2000000000000000D 01 RIGHT=    0.5000000000000000D 01

$ABS(B)
     6LEFT=    0.1000000000000000D 01 RIGHT=    0.8000000000000000D 01

A INTERSECT B
     7LEFT=    0.2000000000000000D 01 RIGHT=    0.5000000000000000D 01

A UNION B
     8LEFT=    0.1000000000000000D 01 RIGHT=    0.8000000000000000D 01

$ENT-TEST ON BEING CONTAINED IN
 A IS CONTAINED IN B
```

References

1. Krückeberg, F. and Unger, H., "On the Numerical Integration of Ordinary Differential Equations and the Determination of Error Bounds," Symposium, Provisional Computation Center, Rome, 1966

2. Moore, R. E., "Interval Analysis," Prentice Hall, 1966.

3. Moore, R. E., "Interval Arithmetic and its Application to Error Estimation and Control," in "Error in Digital Computation" edited by L. B. Rall, Vol. 1, N.Y., 1965.