IBM SYSTEM/360
ASSEMBLY LANGUAGE
INTERVAL ARITHMETIC SOFTWARE

(NASA-TH-X-65971) IBM SYSTEM/360 ASSEMBLY
LANGUAGE INTERVAL ARITHMETIC SOFTWARE E.J.
Phillips (NASA) Apr. 1972 20 p CSCL 09B

APRIL 1972

GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND
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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENT</td>
<td>v</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>INTERVAL ARITHMETIC PACKAGE</td>
<td>2</td>
</tr>
<tr>
<td>I Language</td>
<td>2</td>
</tr>
<tr>
<td>II Purpose</td>
<td>2</td>
</tr>
<tr>
<td>III Method</td>
<td>2</td>
</tr>
<tr>
<td>IV Calling Sequence</td>
<td>2</td>
</tr>
<tr>
<td>V Mathematics Used in Package</td>
<td>7</td>
</tr>
<tr>
<td>VI Restrictions</td>
<td>9</td>
</tr>
<tr>
<td>VII Appendix (Sample problem and output)</td>
<td>10</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>16</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENT

The author wishes to gratefully acknowledge the assistance of Dr. Ronald Tost and other members of the Gesellschaft Für Mathematik 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. \( D = \text{DRE}(A) \) \( D = AR \) Right Corner of \( A \)

Ex. \( A = (2.5, 4.0) \)
\( D = \text{DRE}(A) = 4.0 \)

3. \( A = \text{IT}$(N) \) \( A = (N, N) \) Produce an interval from an integer

Ex. \( N = 5 \)
\( A = \text{IT}$(N) = (5.0, 5.0) \)

4. \( A = \text{DT}$(D) \) \( A = (D, D) \) Produce an interval from a real number

Ex. \( D = 2.52 \)
\( A = \text{DT}$(D) = (2.52, 2.52) \)
\( A = \text{DT}$(.5DO) = (0.5DO, 0.5DO) \)

5. \( A = \text{ADD} (B, C) \) \( A = B + C \) Addition

Ex. \( B = (3.0, 4.0) \)
\( C = (2.0, 2.0) \)
\( A = \text{ADD} (B, C) = (3.0 + 2.0, 4.0 + 2.0) \)
\( = (5.0, 6.0) \)
\( B = (-1.5, -0.5) \)
\( C = (2.5, 4.5) \)
\( A = \text{ADD} (B, C) = (-1.5 + 2.5, -0.5 + 4.5) \)
\( = (1.0, 4.0) \)

6. \( A = \text{SUB} (B, C) \) \( A = B - C \) Subtraction

Ex. \( B = (3.0, 4.0) \)
\( C = (2.0, 2.0) \)
\( A = \text{SUB} (B, C) = (3.0 - 2.0, 4.0 - 2.0) \)
\( = (1.0, 2.0) \)
\( B = (3.0, 4.0) \)
\( C = (-1.5, -2.5) \)
\( A = \text{SUB} (B, C) = (3.0 - (-1.5), 4.0 - (-2.5)) \)
\( = (4.5, 6.5) \)

7. \( A = \text{MUL} (B, C) \) \( A = B \times C \) Multiplication

Ex. \( B = (3.0, 4.0) \)
\( C = (2.0, 2.5) \)
\( A = \text{MUL} (B, C) = (\text{MIN} (6.0, 7.5, 8.0, 10.0), \text{MAX} (6.0, 7.5, 8.0, 10.0)) \)
\( = (6.0, 10.0) \)

8. \( A = \text{DIV} (B, C) \) \( A = B/C \) Division

Ex. \( B = (3.0, 4.0) \) if \( 0 \neq (B, C) \)
\( C = (1.0, 2.0) \)
\( A = \text{DIV} (B, C) = (3.0, 4.0) \times (0.5, 1.0) \)
\( = (\text{MIN} (1.5, 3.0, 2.0, 4.0), \text{MAX} (1.5, 3.0, 2.0, 4.0)) \)
\( = (1.5, 4.0) \)
9. \( A = \text{DOT}(B, N) \quad A = B^N \) Exponentiation
   Ex. \( B = (3.0, 4.0) \)
   \( N = 2 \)
   \( A = \text{DOT}(B, N) = (9.0, 16.0) \)

10. \( A = \text{MIN}(B) \quad A = -B \) Negative Interval
    \( B = (3.0, 4.0) \)
    \( A = \text{MIN}(B) = (-4.0, -3.0) \)

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

    Examples:

    (1) \( B = (4.0, 4.25) \)
        \( A = \text{ABS}(B) = (4.0, 4.25) \)

    (2) \( B = (-2.0, -1.0) \)
        \( A = \text{ABS}(B) = (1.0, 2.0) \)

    (3) \( B = (-1.5, 4.0) \)
        \( A = \text{ABS}(B) = (0, \text{MAX}(-1.5, 4.0)) \)
        \( = (0, 4.0) \)

12. \( A = \text{SCH}(B, C) \) Intersection
    \( B = (4.0, 5.0) \) \$SCH can produce illegal intervals. This can be
    \( C = (-1.5, 2.0) \) proved with ICOR.
    \( A = \text{SCH}(B, C) = (\text{MAX}(C_L, B_L), \text{MIN}(C_R, B_R)) \)
    \( = (4.0, 2.0) \) (- illegal interval)

13. \( A = \text{VER}(B, C) \) Union
    \( B = (1.5, 2.5) \)
    \( C = (1.0, 5.0) \)
    \( A = \text{VER}(B, C) = (\text{MIN}(C_L, B_L), \text{MAX}(C_R, B_R)) \)
    \( = (1.0, 5.0) \)

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:

^ : if AR < 0 (if AR < 0, 0 ∈ A)
^ ^ : if AL > 0 (if AL > 0, 0 ∈ A)
^ ^ : else 0 ∈ A (zero contained in A) null interval

15. IF (ICOR(A)) ^, ^ , ^ ^ Interval Test
   IF ICOR(A) < 0 GO TO^  
   IF ICOR(A) = 0 GO TO^ ^  
   IF ICOR(A) > 0 GO TO^ ^ ^

Test to see:

^ : if AL > AR (illegal interval)
^ ^ : if AL = AR
^ ^ ^ : if AL < AR

16. $ENT – Test on being contained in
   CALL $ENTC A, B, &^1, &^2, &^3, &^4)
   ^1: AL > BL and AR < BR (A is contained in B)
   ^2: AL = BL and AR = BR (A and B are equal)
   ^3: AL < BL and AR > BR (B is contained in A)
   ^4: 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.

Ex:

   CALL $SUM (A, I, J, K)  A = (0.0, 0.0)
   D = $MUL (B, C)         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:

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

Correspondent

   A = (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 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 ('10') → 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>

6
V. MATHEMATICS USED IN PACKAGE

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

\[[a, b] \ast [c, d] = \{x \ast y | a \leq x \leq b, c \leq y \leq d\}\]

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

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

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

and if 0 \(\not\in\) [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

<table>
<thead>
<tr>
<th>bit</th>
<th>0</th>
<th>1</th>
<th>−</th>
<th>7</th>
<th>8</th>
<th>63</th>
</tr>
</thead>
<tbody>
<tr>
<td>sign</td>
<td>characteristic</td>
<td>exponent</td>
<td>fraction (binary)</td>
<td>mantissa</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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. \[ \frac{0 | 1000000}{64 - 64} = 0 = 16 \]

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 \(16^{-65} \times (5.4 \times 10^{-79})\)
\[(1 - 16^{-14}) \times 16^{63} (7.2 \times 10^{75})\]

Example:

\[1 1000001 01010100 00000000 00000000 00000000 \text{ etc.}\]

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.

\[A = 436D1000 \quad 00000000 = \text{normalized number}\]
\[B = 4306D1000 \quad 00000000 = \text{unnormalized}\]
\[= 426D1000 \quad 0000000 \quad \text{after normalization.}\]
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,TD,XREF

ISN 0002  IMPLICIT COMPLEX*16(A-H,S),REAL*8(D-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=0.5DO
ISN 0007  A=(2.0DD$+5.0DD$)
ISN 0008  B=(1.0DD$+8.0DD$)
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)=$ITS(N)
ISN 0017  WRITE(6,108)
ISN 0018  R=DLI(D(1))
ISN 0019  S=DRE(D(1))
ISN 0020  WRITE(6,2002) I,R,S
ISN 0021  I=I+1
ISN 0022  D(2)=$DTS(P)
ISN 0023  WRITE(6,109)
ISN 0024  R=DLI(D(2))
ISN 0025  S=DRE(D(2))
ISN 0026  WRITE(6,2002) I,R,S
ISN 0027  I=I+1
ISN 0028  D(3)=$ADD(A,B)
ISN 0029  WRITE(6,102)
ISN 0030  R=DLI(D(1))
ISN 0031  S=DRE(D(1))
ISN 0032  WRITE(6,2002) I,R,S
ISN 0033  I=I+1
ISN 0034  D(4)=$SUB(A,B)
ISN 0035  WRITE(6,103)
ISN 0036  R=DLI(D(1))
ISN 0037  S=DRE(D(1))
ISN 0038  WRITE(6,2002) I,R,S
ISN 0039  I=I+1
ISN 0040  D(5)=$MUL(A,B)
ISN 0041  WRITE(6,104)
ISN 0042  R=DLI(D(1))
ISN 0043  S=DRE(D(1))
ISN 0044  WRITE(6,2002) I,R,S
ISN 0045  I=I+1
ISN 0046  D(6)=$DIV(A,B)
ISN 0047  WRITE(6,105)
ISN 0048  R=DLI(D(1))
ISN 0049  S=DRE(D(1))
ISN 0050  WRITE(6,2002) I,R,S
ISN 0051  WRITE(6,223)
ISN 0052  I=1
ISN 0053  C(1)=$POT(A,N)
ISN 0054  WRITE(6,110)
ISN 0055  R=DLI(C(1))
ISN 0056  S=DREC(C(I))
ISN 0057  WRITE(6,2002) I,R,S
ISN 0058  I=I+1
ISN 0059  C(2)=$DPT(B,N)
ISN 0060  WRITE(6,111)
ISN 0061  R=DLI(C(I))
ISN 0062  S=DREC(C(I))
ISN 0063  WRITE(6,2002) I,R,S
ISN 0064  I=I+1
ISN 0065  C(3)=$MIN(A)
ISN 0066  WRITE(6,112)
ISN 0067  R=DLI(C(I))
ISN 0068  S=DREC(C(I))
ISN 0069  WRITE(6,2002) I,R,S
ISN 0070  I=I+1
ISN 0071  C(4)=$MIN(B)
ISN 0072  WRITE(6,113)
ISN 0073  R=DLI(C(I))
ISN 0074  S=DREC(C(I))
ISN 0075  WRITE(6,2002) I,R,S
ISN 0076  I=I+1
ISN 0077  C(5)=$ABS(A)
ISN 0078  WRITE(6,114)
ISN 0079  R=DLI(C(I))
ISN 0080  S=DREC(C(I))
ISN 0081  WRITE(6,2002) I,R,S
ISN 0082  I=I+1
ISN 0083  C(6)=$ABS(B)
ISN 0084  WRITE(6,115)
ISN 0085  R=DLI(C(I))
ISN 0086  S=DREC(C(I))
ISN 0087  WRITE(6,2002) I,R,S
ISN 0088  I=I+1
ISN 0089  C(7)=$SCH(A,B)
ISN 0090  WRITE(6,107)
ISN 0091  R=DLI(C(I))
ISN 0092  S=DREC(C(I))
ISN 0093  WRITE(6,2002) I,R,S
ISN 0094  IF(1COR(C(7))) 10,20,20
ISN 0095  10 IF(R.GT.S) WRITE(6,190) R,S
ISN 0096  I=I+1
ISN 0097  20 C(8)=$VER(A,B)
ISN 0098  WRITE(6,106)
ISN 0099  R=DLI(C(I))
ISN 0100  S=DREC(C(I))
ISN 0101  WRITE(6,2002) I,R,S
ISN 0102  IF(IZER(I)) 28,30,29
ISN 0103  28 IF(S.LT.0) GO TO 35
ISN 0104  29 IF(R.GT.0) GO TO 35
ISN 0105  30 WRITE(6,210)
ISN 0106  C.TEST ON BEING_CONTAINED_IN
ISN 0107  35 WRITE(6,217)
ISN 0108  CALL SENT(A,0,636,637,638,639)
ISN 0109  36 WRITE(A,211)
ISN 0110  GO TO 40
ISN 0111  37 WRITE(6,212)
ISN 0114    38  GO TO 40
ISN 0115    38  WRITE(6,213)
ISN 0116    39  GO TO 40
ISN 0117    39  WRITE(6,214)
ISN 0118    40  R=(0.000,0.000)
ISN 0119    40  WRITE(6,101)
ISN 0120    40  T=DLI(A)
ISN 0121    40  (=DRE(B)
ISN 0122    41  WRITE(6,203) V,S,T,U,N,P
ISN 0123    41  WRITE(6,217)
ISN 0124    41  CALL SENT(A,B,641,642,643,644)
ISN 0125    41  WRITE(6,211)
ISN 0126    42  GO TO 44
ISN 0127    42  WRITE(6,212)
ISN 0128    43  GO TO 45
ISN 0129    43  WRITE(6,213)
ISN 0130    44  GO TO 45
ISN 0131    44  WRITE(6,214)
ISN 0132    45  CONTINUE
ISN 0133    45  CALL $SUM(B,I,1,5)
ISN 0134    45  F=$MUL(C(I),D(I))
ISN 0135    45  R=DLIF
ISN 0136    45  S=DRE(F)
ISN 0137    45  WRITE(6,218)
ISN 0138    45  WRITE(6,215),R,S.

C CONVERSION ROUTINE
ISN 0139    45  C(1)=$INP(  1,'3,1.56')
ISN 0140    45  C(2)=$INP(  -2,'4')
ISN 0141    45  C(3)=$INP(  '1')
ISN 0142    45  C(4)=$INP(  '1E45,23,2,5')
ISN 0143    45  C(5)=$DTS(5,500)
ISN 0144    45  C(6)=$IT$(9)
ISN 0145    45  WRITE(6,216)
ISN 0146    45  DD 9 I=1,6
ISN 0147    45  R=DLI(C(II))
ISN 0148    45  S=DRE(C(II))
ISN 0149    45  WRITE(6,2002),I,R,S.
ISN 0150    45  2002 FORMAT('X15','LEFT= ',D24.16,' RIGHT= ',D24.16)
ISN 0151    45  101 FORMAT('1 INTERVAL ARITHMETIC PACKAGE')
ISN 0152    45  102 FORMAT('OA+B')
ISN 0153    45  103 FORMAT('OA-A')
ISN 0154    45  104 FORMAT('OA*B')
ISN 0155    45  105 FORMAT('OA/B')
ISN 0156    45  106 FORMAT('OA UNION B')
ISN 0157    45  107 FORMAT('OA INTERSECT B')
ISN 0158    45  108 FORMAT('OSIT$(N')
ISN 0159    45  109 FORMAT('OSIT$(P')
ISN 0160    45  110 FORMAT('OA==N')
ISN 0161    45  111 FORMAT('OP==N')
ISN 0162    45  112 FORMAT('OSMIN(A')
ISN 0163    45  113 FORMAT('OSMIN(R')
ISN 0164    45  114 FORMAT('OSABS(A')
ISN 0165    45  115 FORMAT('OSABS(R')
ISN 0166    45  190 FORMAT(' THIS IS AN ILLEGAL INTERVAL ('D24,16,''),(D24,16,')')
ISN 0167    45  203 FORMAT('X1*='D24,16,'X1/D24,16/1X*='D24,16,'X1/D24,16/1X*='D24,16,
11X*='D24,16,'X1/'D24,16/'= 'D24,16,'X1/='D24,16,'X1/')

12
<table>
<thead>
<tr>
<th>ISN 0168</th>
<th>210</th>
<th>FORMAT(' ZERO IS CONTAINED IN THE FOLLOWING INTERVAL ('',G24.16,',')')</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISN 0169</td>
<td>211</td>
<td>FORMAT(' A IS CONTAINED IN B')</td>
</tr>
<tr>
<td>ISN 0170</td>
<td>212</td>
<td>FORMAT(' A AND B ARE EQUAL')</td>
</tr>
<tr>
<td>ISN 0171</td>
<td>213</td>
<td>FORMAT(' B IS CONTAINED IN A')</td>
</tr>
<tr>
<td>ISN 0172</td>
<td>214</td>
<td>FORMAT(' A AND B ARE NOT FULLY CONTAINED IN EACH OTHER')</td>
</tr>
<tr>
<td>ISN 0173</td>
<td>215</td>
<td>FORMAT(' LEFT= ',D24.16,2X,'RIGHT= ',D24.16)</td>
</tr>
<tr>
<td>ISN 0174</td>
<td>216</td>
<td>FORMAT('OS$INP CONVERSION ROUTINE')</td>
</tr>
<tr>
<td>ISN 0175</td>
<td>217</td>
<td>FORMAT('OS$ENT-TEST ON BEING CONTAINED IN')</td>
</tr>
<tr>
<td>ISN 0176</td>
<td>218</td>
<td>FORMAT('OSUM PRODUCED WITH SCALAR PRODUCT')</td>
</tr>
<tr>
<td>ISN 0177</td>
<td>222</td>
<td>FORMAT('OVALUES FOR ARRAY D(I)')</td>
</tr>
<tr>
<td>ISN 0178</td>
<td>223</td>
<td>FORMAT('OVALUES FOR ARRAY C(I)')</td>
</tr>
<tr>
<td>ISN 0179</td>
<td></td>
<td>STOP</td>
</tr>
<tr>
<td>ISN 0180</td>
<td></td>
<td>END</td>
</tr>
</tbody>
</table>

13
INTERVAL ARITHMETIC PACKAGE

A = 0.2000000000000000D 01, 0.8000000000000000D 01
B = 0.0
data

N = 3
P = 0.5000000000000000D 00

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

SUM PRODUCED WITH SCALAR PRODUCT
LEFT = -0.6850000000000000D 02 RIGHT = 0.8730000000000000D 03

$INP CONVERSION ROUTINE

1LEFT = 0.1300000000000000D 01 RIGHT = 0.1560000000000000D 01
2LEFT = -0.2000000000000000D 01 RIGHT = 0.4000000000000000D 01
3LEFT = -0.2000000000000000D 01 RIGHT = 0.4000000000000000D 01
4LEFT = 0.2500000000000000D 01 RIGHT = 0.1000000000000000D 46
5LEFT = 0.5500000000000000D 01 RIGHT = 0.5500000000000000D 01
6LEFT = 0.9000000000000000D 01 RIGHT = 0.9000000000000000D 01
INTERVAL ARITHMETIC PACKAGE

A = 0.20000000000000000 01, 0.50000000000000000 01
B = 0.10000000000000000 01, 0.80000000000000000 01
N = 3
P = 0.50000000000000000 01

VALUES FOR ARRAY D(J)

$ITS(N)
1LEFT = 0.30000000000000000 01 RIGHT = 0.30000000000000000 01
$ITS(B)
2LEFT = 0.50000000000000000 00 RIGHT = 0.50000000000000000 00

A+B
3LEFT = 0.30000000000000000 01 RIGHT = 0.13000000000000000 02

A-R
4LEFT = -0.60000000000000000 01 RIGHT = -0.40000000000000000 01

A*B
5LEFT = 0.20000000000000000 01 RIGHT = 0.40000000000000000 02

A/B
6LEFT = 0.25000000000000000 00 RIGHT = 0.50000000000000000 01

VALUES FOR ARRAY C(J)

A**N
1LEFT = 0.80000000000000000 01 RIGHT = 0.12500000000000000 03

B**N
2LEFT = 0.10000000000000000 01 RIGHT = 0.51200000000000000 03

$MIN(A)
3LEFT = -0.50000000000000000 01 RIGHT = -0.20000000000000000 01

$MIN(B)
4LEFT = -0.80000000000000000 01 RIGHT = -0.10000000000000000 01

$ARS(A)
5LEFT = 0.20000000000000000 01 RIGHT = 0.50000000000000000 01

$ARS(B)
6LEFT = 0.10000000000000000 01 RIGHT = 0.80000000000000000 01

A INTERSECT B
7LEFT = 0.20000000000000000 01 RIGHT = 0.50000000000000000 01

A UNION B
8LEFT = 0.10000000000000000 01 RIGHT = 0.80000000000000000 01

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

