Interval Arithmetic in Fortran – Some Specifics

by
R. Baker Kearfott

Department of Mathematics
University of Southwestern Louisiana
U.S.L. Box 4-1010
Lafayette, LA 70504-1010 U.S.A.

Outline of Presentation

- Overview – Major Points
- Overview – Some Features
- Highlights of Proposed Edits
- A Module?
- Example Code

Overview – Major Points

- *"Thou shalt contain."*
  - Computed results contain the exact mathematical results.
- *"Thou shalt be unrestrictive."*
  - easy to understand
  - implementable on all platforms
  - allow optimization
- *"Thou shalt blend in."*
  - Style of standard is used.
  - `COMPLEX` data type is used as a template.

Overview – Some Features

- An interval numeric data type, obeying, overall, the same syntax as the other numeric data types.
- Several new infix operators and intrinsics.
- Interval data in those intrinsics that accept real data.
- Neither support nor prohibit a complex interval data type or extended interval arithmetic.
- I/O as a natural but useful extension of Fortran standard I/O.
- *Results contain the exact mathematical result.*

The `INTERVAL` Data Type

- has two `REAL` components, a lower bound and upper bound;
- has arithmetic operations defined in terms of arithmetic operations on `REAL`, with *outward roundings*.
- Outward roundings can be defined with IEEE arithmetic, or otherwise.

The INTERVAL Data Type

Operational Definitions

$$\mathbf{x} + \mathbf{y} = [\underline{x} + \underline{y}, \overline{x} + \overline{y}],$$

$$\mathbf{x} - \mathbf{y} = [\underline{x} - \overline{y}, \overline{x} - \underline{y}],$$

$$\mathbf{x} \times \mathbf{y} = [\min\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}, \max\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}]$$

$$\frac{1}{\mathbf{x}} = \begin{cases} [1/\overline{x}, 1/\underline{x}] & \text{if } \underline{x} > 0 \\ [1/\underline{x}, 1/\overline{x}] & \text{if } \overline{x} < 0 \end{cases}$$

$$\mathbf{x} \div \mathbf{y} = \mathbf{x} \times 1/\mathbf{y}$$

Above, $\mathbf{x} = [\underline{x}, \overline{x}]$, $\mathbf{y} = [\underline{y}, \overline{y}]$, and $\underline{x} \leq \overline{x}$, $\underline{y} \leq \overline{y}$.

New Infix Operators

Syntax	function
`Z = X.IS.Y`	$\mathbf{z} \leftarrow \mathbf{x} \cap \mathbf{y}$
`Z = X.CH.Y`	$\mathbf{z} \leftarrow [\min\{\underline{x}, \underline{y}\}, \max\{\overline{x}, \overline{y}\}]$
`X.SB.Y`	`.TRUE.` if $\mathbf{x} \subseteq \mathbf{y}$
`X.SP.Y`	`.TRUE.` if $\mathbf{x} \supseteq \mathbf{y}$
`X.DJ.Y`	`.TRUE.` if $\mathbf{x} \cap \mathbf{y} = \emptyset$
`R.IN.X`	`.TRUE.` if $r \in \mathbf{x}$

Interval Versions of Relational Operators

Syntax	function
`Y.LT.X`	`.TRUE.` if $\overline{y} < \underline{x}$
`Y.GT.X`	`.TRUE.` if $\underline{y} > \overline{x}$
`Y.LE.X`	`.TRUE.` if $\overline{y} \geq \underline{x}$
`Y.GE.X`	`.TRUE.` if $\underline{y} \geq \overline{x}$
`Y.NE.X`	`.TRUE.` if $y \neq x$ for some $x \in \mathbf{x}$ and $y \in \mathbf{y}$
`Y.EQ.X`	`.TRUE.` if $\underline{x} = \overline{x} = \underline{y} = \overline{y}$

Special Interval Functions

Syntax	function
`R = INF(X)`	Lower bound of `X`
`R = SUP(X)`	Upper bound of `X`
`R = MID(X)`	Midpoint of `X`
`R = WID(X)`	$r \leftarrow \lvert\overline{x} - \underline{x}\rvert$
`R = MAG(X)`	$r \leftarrow \max\{\lvert\underline{x}\rvert, \lvert\overline{x}\rvert\}$
`R = MIG(X)`	"mignitude:" $r \leftarrow \min\{\lvert\underline{x}\rvert, \lvert\overline{x}\rvert\}$ if $0 \notin \mathbf{x}$, and $r \leftarrow 0$ otherwise
`Z = ABS(X)`	$\mathbf{z} \leftarrow \{\lvert x\rvert, x \in \mathbf{x}\}$
`Z = MAX(X,Y)`	$\mathbf{z} \leftarrow [\max\{\underline{x}, \underline{y}\}, \max\{\overline{x}, \overline{y}\}]$
`N = NDIGITS(X)`	Number of leading decimal digits that are the same in $\underline{x}$ and $\overline{x}$.

Generics that Allow Intervals

1. *All but the conversion functions return enclosures on the range.*
2. *The sharpness of the enclosures is not specified.*
3. *Mixed mode interval/complex is not specified, but mixed mode interval/real is.*

- **Numeric, Mathematical, and Vector:** All functions that accept real arguments should accept interval arguments.
- **Conversion from interval:** (return the midpoint) `DBLE`, `INT`, `REAL`.
- **Conversion to interval:** `INTERVAL`: Elemental function; allows one or two arguments.

Interval I/O

1. Intervals are represented as pairs of reals, beginning with "`(<`", separated by "`,`", and ending in "`>)`".
2. On input, the stored interval shall contain the interval represented by the character string.
3. On output, the printed interval shall contain the interval represented internally.
4. The `VE` and `VF` formats are provided for outwardly rounded interval I/O. Intervals can also be input and output with `NAMELIST` and list-directed I/O.
5. Intervals may also be input/output with two `E` or two `F` formats. Then, the default rounding for these formats (and not outward rounding) occurs.

Interval I/O

An Example

Suppose an interval variable `X` is defined in a program, suppose the program contained the statement

```
WRITE(*,'(1X,VE12.5E1)') X
```

and suppose the internal representation of `X` is that of the interval

```
(<1.9921875,2.9921875>)
```

Then a valid output produced by the `WRITE` statement is

```
(< +0.19921E+1, +0.29922E+1>)
```

Optimization of Interval Expressions

An Example

$$[0,1]^2 - [0,1]$$

evaluates to

$$[0,1] - [0,1] = [-1,1],$$

while

$$[0,1] \cdot ([0,1] - [1,1])$$

evaluates to

$$[0,1] \cdot [-1,0] = [-1,0].$$

However, both $[-1,1]$ and $[-1,0]$ are bounds on the range of

$$x^2 - x$$

over $[0,1]$. We would like to transform $x^2 - x$ to $x(x-1)$ in this case, or else compute the expression *both* ways and take the intersection.

A Module?

- How would the I/O be implemented in a user-friendly fashion?
- The interval infix operators have a natural precedence within the set of logical and arithmetic operators that is presently standard. How would this precedence be defined by a module?
- How would optimization of interval expressions be done?

Some Example Code

(From a module without interval I/O.)

```
PROGRAM INTERVAL_NEWTON_ITERATION_1_D
  USE INTLIB_ARITHMETIC
  IMPLICIT NONE

  REAL(KIND=KIND(0.0D0)) :: XP
  TYPE(INTERVAL) :: X, X_IMAGE
  INTEGER K
  REAL(KIND=KIND(0.0D0)) :: WIDTH_OF_X, OLD_WIDTH

  CALL SIMINI
     WIDTH_OF_X = 4
     X = INTERVAL(1,2)
     DO K = 1,10000
        OLD_WIDTH = WIDTH_OF_X
        WIDTH_OF_X = IWID(X)
        XP = DBLE(X)
        WRITE(6,*) K, X, XP, WIDTH_OF_X, &
                   WIDTH_OF_X/OLD_WIDTH**2
        X_IMAGE = XP - &
         ( INTERVAL(XP)**2-INTERVAL(4) ) / (2*X)
        IF(WIDTH_OF_X.LT.1D-11) EXIT
        X = X_IMAGE
     END DO
END PROGRAM INTERVAL_NEWTON_ITERATION_1_D
```

Some Example Code

Actual Output

Columns 1 through 4:

```
 1   0.9999999999999998   2.0000000000000004   1.5000000000000000
 2   1.9374999999999991   2.3750000000000018   2.1562500000000004
 3   1.9886592741935472   2.0195312500000009   2.0040952620967740
 4   1.9999724292486507   2.0000354537727549   2.0000039415107027
 5   1.9999999999417792   2.0000000000659868   2.0000000000038831
 6   1.9999999999999989   2.0000000000000009   2.0000000000000000
```

Columns 5 and 6:

```
1.0000000000000009   6.2500000000000056E-02
0.4375000000000028   0.4375000000000020
3.0871975806453740E-02   0.1612903225806542
6.3024524104227111E-05   6.6127289936492972E-02
1.2420753314756896E-10   3.1270065174661590E-02
1.9984014443252822E-15   1.2953492022672087E+05
```

Some Example Code

Output with Interval I/O

If the line:

```
WRITE(6,*) K, X, XP, WIDTH_OF_X, &
           WIDTH_OF_X/OLD_WIDTH**2
```

were replaced by:

```
WRITE(6,'(1X,I2,1X,VE10.4E1,3(1X,E12.4E2))') &
  K, X, XP, WIDTH_OF_X, &
  WIDTH_OF_X/OLD_WIDTH**2
```

the output would be:

```
 1 (< 0.9999E+0, 0.2001E+1>)  0.1500E+01  0.1000E+01  0.6250E-01
 2 (< 0.1937E+1, 0.2376E+1>)  0.2156E+01  0.4375E+00  0.4375E+00
 3 (< 0.1987E+1, 0.2020E+1>)  0.2004E+01  0.3087E-01  0.1612E+00
 4 (< 0.1999E+1, 0.2001E+1>)  0.2000E+01  0.6302E-04  0.6612E-01
 5 (< 0.1999E+1, 0.2001E+1>)  0.2000E+01  0.1242E-09  0.3127E-01
 6 (< 0.1999E+1, 0.2001E+1>)  0.2000E+01  0.1998E-14  0.1295E+05
```

Some Example Code

Single-Number Interval Output

SE and SF formats are also envisioned for output of intervals as single numbers. (See the notes.) If the write statement were:

```
WRITE(6,'(1X,I2,1X,SF18.15,3(1X,E12.4E2))') &
  K, X, XP, WIDTH_OF_X, &
  WIDTH_OF_X/OLD_WIDTH**2
```

the output would be:

```
1 (<******************>)  0.1500E+01  0.1000E+01  0.6250E-01
2 (< 2.                >)  0.2156E+01  0.4375E+00  0.4375E+00
3 (< 2.0               >)  0.2004E+01  0.3087E-01  0.1612E+00
4 (< 2.0000            >)  0.2000E+01  0.6302E-04  0.6612E-01
5 (< 2.000000000       >)  0.2000E+01  0.1242E-09  0.3127E-01
6 (< 2.00000000000000 >)  0.2000E+01  0.1998E-14  0.1295E+05
```

The number of digits printed is the number of digits *known to be correct*, assuming the actual number, displayed as an infinite decimal sequence, has been rounded into the displayed digits by rounding to nearest.

On Actual Edits

Certain features are not yet fully clear, but could be useful to have later:

Complex interval arithmetic: possibly to hard to implement in relation to utility.

Extended interval arithmetic: involves intervals with lower end points larger than upper end points. There are at least two different systems, with different meanings.

Should there be the following types of notes?

1. To explain and illustrate requirements of the standard.
2. To suggest and illustrate possible extensions.
3. To give guidance concerning implementation.