Optimizing the Representation of Intervals

Javier D. Bruguera

University of Santiago de Compostela, Spain

Numerical Sofware: Design, Analysis and Verification
Santander, Spain, July 4-6 2012

Contents

Contents

Abstract

Summary

- Representation of intervals that, instead of both end points, uses the low point and the width of the interval
  - More efficient
  - Width of the interval is represented with a smaller number of bits than the endpoint
  - Better utilization of the number of bits available
- The number of bits of the low point and of the width is determined so that the rounding error is minimized
- The representation is evaluated with several examples
  - Narrower than those obtained with the traditional representation

Contents

Interval arithmetic

What is interval arithmetic?

- Each value belongs to an interval such as the true value lies in the interval
- Used to
  - Bound roundoff errors in numerical computations
  - Evaluate the effects of approximation errors
  - Evaluate the effects of inaccurate inputs
- Disadvantage: produces large intervals
- Special algorithms to avoid large intervals

Interval arithmetic as an error monitoring method

Rounding error in floating–point computations

- Rounding introduces an error every FP operation
  - rounding error $\leq$ 0.5 *ulp* if exact rounding
  - rounding error $\leq$ 1 *ulp* if faithful rounding
- Errors can propagate and can be amplified (cancellations, normalizations)
- Errors can produce inaccurate results for some computations
- Accumulation of rounding errors, wider intervals

Other solutions for error monitoring

Hardware methods

- Significance arithmetic
  - Specification of the number of correct bits of the result
  - Number of correct bits updated only when there is a cancellation in efective subtraction
  - Does not include rounding errors
- Error estimate
  - Estimation of the rounding errors including propagation, amplification and cancellation of errors
  - Concurrently with the program execution
  - The estimate is not exact and can be inaccurate
- FP double–double and quad–double arithmetic
  - Results as non–evaluated sum of two or four DP–FP numbers
  - Rounding errors accumulate in the least significant part
  - Slow, several operations to determine the errors

Other solutions for error monitoring

Software methods: much slower implementation and modification of the program

- Running error
  - Error bound during the execution of a program
- Error computation
  - Based on automatic partial differentation
- Stochastic arithmetic
  - Several executions of the program with different rounding error approximations

Interval arithmetic

Traditional representation

- Traditional representation: lower and upper end points
- **LU representation**
  - Two floating–point numbers
  - Not efficient

  interval of size $2^{-j} \Rightarrow$ the j MS bits of L and U are the same

Implementation

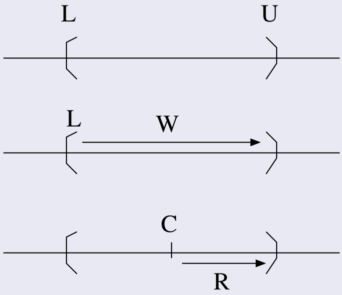
- Hardware
  - Enhancements of the ISA of a processor
  - Special functional units
  - Variable precision to avoid wide intervals
- Mainly used in software
  - Rounding modes of the IEEE standard

Contents

Alternative interval representations

Interval representations

- Lower and upper points (**LU representation**)
- Lower point and width of the interval (**LW representation**)
- Center point and radius (**CR representation**)
- Width (or radius) can be represented with less bits than low point (or center)



Contents

LW representation

Enclosing the real number x

- LU: x_l (low) and x_u (up) , such as

$$x_l \leq x \leq x_u$$

- LW: x_l (low) and x_w (width), such as

$$x_l \leq x \leq x_l + x_w$$

- $x_l, x_u, x_l + x_w$ standard FP numbers
- x_w FP number with a smaller precision and different range
  - Low point: $x_l = (-1)^{l_s} \times (1.l_f) \times 2^{l_e}$
  - Width: $x_w = (1.w_f) \times 2^{w_e}$

Number of bits for the low point and the interval width

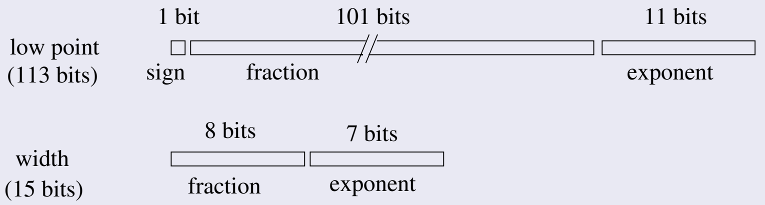
Variable or fixed number of bits?

1. Low is a floating–point number (double precision), width a floating–point number with a reduced precision
   - Number of bits of low same as for LU representation
   - Add the representation of width
   - Efficiency: reduction in number of bits with respect to LU
2. Total number of bits is fixed (2 doubles) and partitioned between the low and width
   - Same total number of bits as for LU representation
   - Efficiency: reduction in interval size for given number of bits

- The second approach seems more appropiate: $t = f + m$
  t: total number of bits,
  f, m: number of bits of the low point and the width

Number of bits for the low point and the interval width

Fixed number of bits



Total number of bits: 128

Fixed number of bits

Optimal width size

- $z = x \ op \ y$
- z_w is the sum of the *propagated* width and the *generated* width
  - The *propagated* width (w_p) depends on the operation
  - The *generated* width (w_g) is due to the roundoff of z_l and z_w

$$w_g < 2^{e_l - f} + 2^{e_w - m}$$

  - Considering $j = e_l - e_w$, $f = t - m$

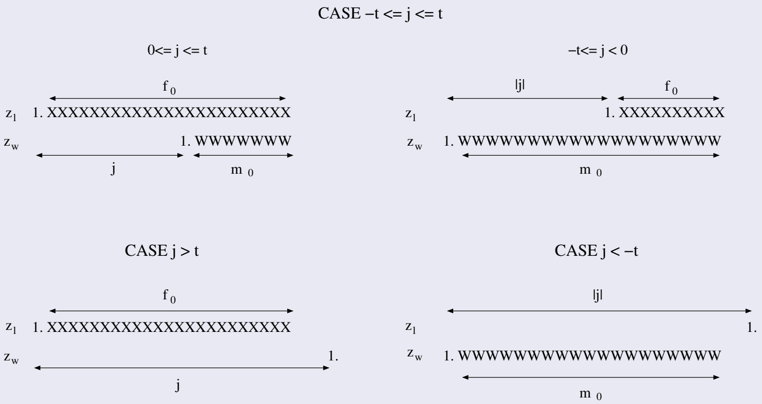
$$w_g < (2^{-(t-m)} + 2^{-(j+m)})2^{e_l}$$

  - Mimimum width is

$$\begin{aligned} m_{mim} &= (t-j)/2 \text{ and} \\ f &= t - m_{mim} = (t+j)/2 \end{aligned}$$

Optimal width size

Optimal width internal



Optimal width size

A numerical example

- $z_l = 1.3 \times 2^{-3}, \; z_w = 1.27 \times 2^{-25}$
- Number of total bits is $t = 32$
- LU: 16 bits each end point (most of the bits are equal)

$$z_{l16} = 1.0100110011001100 \times 2^{-3}$$

$$z_{u16} = 1.0100110011001101 \times 2^{-3}$$

- LW: $j = e_l - e_w$, $m = (t - j)/2$

$$j = 22, \Rightarrow m = 5, \; f = 27$$

- Interval width is

$$z_w = 1.01010 * 2^{-25}$$

Optimal width size

A numerical example

m	f	z_w
0	32	$1 * 2^{-24}$
2	30	$1.10 * 2^{-25}$
4	28	$1.0101 * 2^{-25}$
5	27	$1.01010 * 2^{-25}$
6	26	$1.010110 * 2^{-25}$
8	24	$1.10000110 * 2^{-25}$
10	22	$1.001000101 * 2^{-24}$
12	20	$1.01010001010 * 2^{-23}$
14	18	$1.0001010001010 * 2^{-21}$
16	16	$1.000001010001010 * 2^{-19}$

Table: Widths of interval for different values of m

Optimal width size

The best partition depends on m

- The best partition depends on the relative value of m
- The width varies, then the best partition is variable
  - Depends on the specific computation
  - Depends on the stage of the computation
- The variable width is difficult to implement
  - Use a fixed m which would not be optimal
- The utilization of the 32 bits is better in LW than in the LU

The operations

Addition/subtraction

- Addition

$$z_l = fpd(x_l + y_l)$$
$$z_w = fpu(x_w + y_w + ulp(z_l))$$

- Subtraction

$$z_l = fpd(x_l - (y_l + y_w))$$
$$z_w = fpu(x_w + y_w + ulp(z_l))$$

The operations

Multiplication

x_l	y_l	$x_l + x_w$	$y_l + y_w$	z_w
≥ 0	≥ 0	–	–	$(x_l + x_w) \times y_w + x_w \times y_l$
< 0	≥ 0	< 0	–	$x_w \times y_l + \lvert x_l \rvert \times y_w$
< 0	≥ 0	≥ 0	–	$x_w \times (y_l + y_w)$
≥ 0	< 0	–	< 0	$x_l \times y_w + x_w \times \lvert y_l \rvert$
≥ 0	< 0	–	≥ 0	$(x_l + x_w) \times y_w$
< 0	< 0	< 0	< 0	$\lvert x_l + x_w \rvert \times y_w + x_w \times \lvert y_l \rvert$
< 0	< 0	≥ 0	< 0	$x_w \times \lvert y_l \rvert$
< 0	< 0	< 0	≥ 0	$\lvert x_l \rvert \times y_w$
< 0	< 0	≥ 0	≥ 0	$\max(\lvert x_l \rvert \times y_w, x_w \times \lvert y_l \rvert, (x_l + x_w) \times y_w, x_w \times (y_l + y_w))$

Propagated interval width for multiplication

The operations

Division

x_l	y_l	$x_l + x_w$	$y_l + y_w$	z_w
≥ 0	> 0	–	–	$((x_l + x_w) \times y_w + x_w \times y_l)/(y_l \times (y_l + y_w))$
< 0	> 0	< 0	–	$(x_w \times y_l + \|x_l\| \times y_w)/(y_l \times (y_l + y_w))$
< 0	> 0	≥ 0	–	x_w / y_l
≥ 0	< 0	–	< 0	$(x_w \times \|y_l\| + x_l \times y_w)/(\|y_l\| \times \|y_l + y_w\|)$
< 0	< 0	< 0	< 0	$(\|x_l + x_w\| \times y_w + x_w \times \|y_l\|)/(\|y_l\| \times \|y_l + y_w\|)$
< 0	< 0	≥ 0	< 0	$x_w / \|y_l + y_w\|$
–	≤ 0	–	≥ 0	divisor interval includes 0

Propagated interval width for division

The operations

Implementation issues

- LW representation is intended for a processor with interval instructions and hardware implementation of the functional units
- Fixed number of bits for the width (implementation with a variable number of bits for the width might be impractical)
- Rounding of z_l and z_w
  - Towards minus infinity for z_l, towards plus infinity for z_w
  - Guarantees the enclosure of the real value
- Operations to compute the width
  - Can be performed in the precision of the width, narrow datapath
  - Every operation introduces a rounding error, width might be somewhat larger

Contents

Some examples

Objective: comparison with LU representation

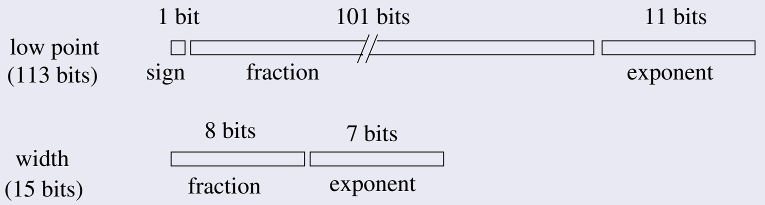
- Effect of LW representation on interval width
- Effect of precision of width m for fixed m
- Effect of having a variable m
- Tightness of the enclosure (error of the floating-point computation).

Parameters

- Relative width to reflect the accuracy of the result
- LU: endpoints are DP FP numbers ($64 + 64$ bits)
- LW: 128 bits, 109 bits for significands of L and W ($109 - m$ bits and m bits, respectively)
- The exact error is the ratio between the FP value and high precision result obtained with Maple

Number of bits for the low point and the interval width

Fixed number of bits



Total number of bits: 128

Some examples

Simple examples

- Evaluation of a polynomial
  - Relative error large when x is close to root value
- Inner product computation
  - Large errors if generated errors are accumulated
  - Generated errors can cancel
  - Final error large if there is a massive cancellation
- Logistic iteration

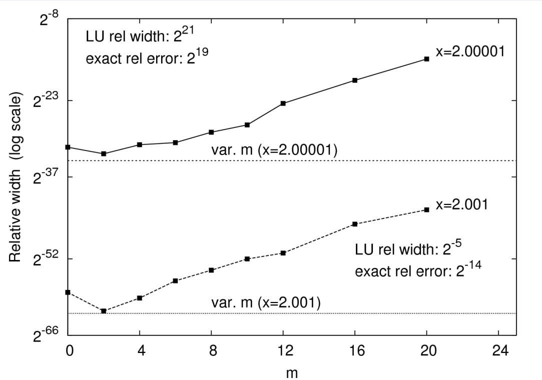
$$x_{n+1} = a \times x_n(1 - x_n), \quad 0 < a < 4, \quad 0 < x_0 < 1$$

  - $a < 3$. Converges to a fixed point, whatever x_0.
  - $3.0 \leq a \leq 3.57$. Periodic and the periodicity depends on a.
  - $a > 3.57$. Chaotic, with an unpredictable trajectory.

Simple examples

Evaluation of a polynomial

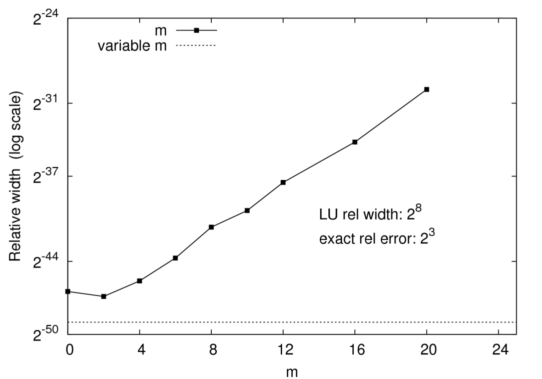
$$p(x) = x^4 - 8x^3 + 24x^2 - 32x + 16 \qquad \textit{root is } x = 2$$



Simple examples

Inner product with cancellation

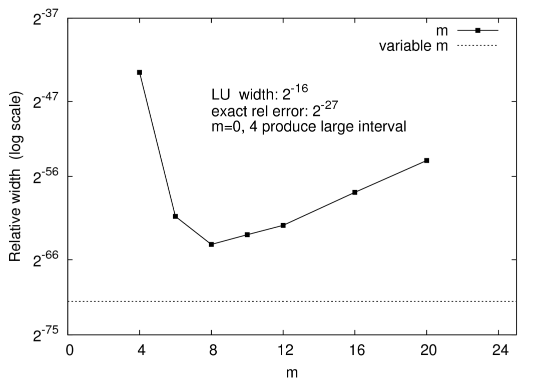
$$z = \sum_{i=1}^{5} x_i \times y_i$$



Simple examples

Logistic iteration

$$x_{n+1} = a \times x_n(1 - x_n), \quad a = 3.59, \quad 0 < x_0 < 1, \quad 150 \textit{ iterations}$$



Simple examples

Results

- Interval width of LW much smaller than that for LU
  - Larger number of bits used in the low point (same number of total bits)
- Width for the LW varies with m.
  - Best m depends on the computation, m between 6 and 8
  - Variable m produces a smaller width than the best fixed m
- Interval width vs FP error
  - Width in LU significantly larger than the FP error: rounding errors are compensated by the rounding-to-nearest scheme, which is not the case for the enclosure of LU.
  - Width in LW much smaller than the FP error: higher precision of low point.

Some examples

Gaussian elimination (GE)

- Solution of linear system $A \times x = b$
- GE with partial pivoting can lead to inaccurate results due
  - Accumulation of rounding errors
  - Cancellations
  - Bad selection of the pivots
- We have simulated GE for several matrices and dimension: LW representation produces narrower intervals.

Gaussian Elimination ($A \times x = b$)

$$A = \begin{pmatrix} 10 & 45 & 120 & 210 & 252 & 210 & 120 & 45 & 10 & 1 \\ 55 & 330 & 990 & 1848 & 2310 & 1980 & 1155 & 440 & 99 & 10 \\ 220 & 1485 & 4752 & 9240 & 11880 & 10395 & 6160 & 2376 & 540 & 55 \\ 715 & 5148 & 17160 & 34320 & 45045 & 40040 & 24024 & 9360 & 2145 & 220 \\ 2002 & 15015 & 51480 & 105105 & 140140 & 126126 & 76440 & 30030 & 6930 & 715 \\ 5005 & 38610 & 135135 & 280280 & 378378 & 343980 & 210210 & 83160 & 19305 & 2002 \\ 11440 & 90090 & 320320 & 672672 & 917280 & 840840 & 517440 & 205920 & 48048 & 5005 \\ 24310 & 194480 & 700128 & 1485120 & 2042040 & 1884960 & 1166880 & 466752 & 109395 & 11440 \\ 48620 & 393822 & 1432080 & 3063060 & 4241160 & 3938220 & 2450448 & 984555 & 231660 & 24310 \\ 92378 & 755820 & 2771340 & 5969040 & 8314020 & 7759752 & 4849845 & 1956240 & 461890 & 48620 \end{pmatrix}$$

$b = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$,

Result and error

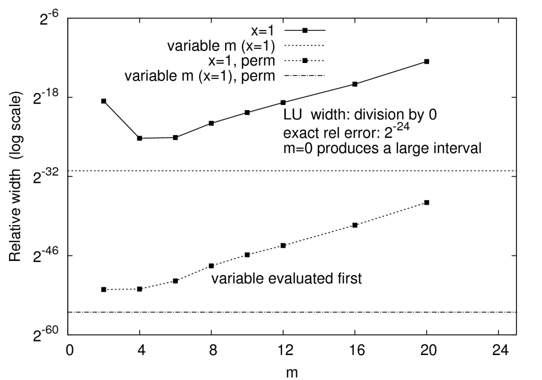
- Exact result: $x = (0, 1, -2, 3, -4, 5, -6, 7, -8, 9)$
- DP FP result:

$$\begin{aligned} fpx \quad = \quad & (9.8407492921 \times 10^{-9}, 0.9999999055, -1.9999994966, 2.9999980387, -3.9999937636, \\ & 4.9999828578, -5.9999578189, 6.9999049037, -7.9998003422, 8.9996049158) \end{aligned}$$

- Absolute error: from 2^{-12} (for x_9) to 2^{-27} (for x_0).
- For the LU representation the interval of divisors include 0

Gaussian Elimination ($A \times x = b$)

Interval width for x_1



Contents

Conclusions

Main conclusions

- LW representation as a more efficient alternative to the traditional LU representation of intervals
- LW representation and a fixed total number of bits,
  - Partition among the bits for the low point and the width
  - The rounding error is minimized.
  - Variable partition is optimal but difficult to implement.
  - Fixed partition can produce good results.
- The examples show that
  - LW representation results in a substantial reduction in the width of the interval with respect to the LU with the same number of bits (128 bits).
  - This reduction is mainly due to the increased precision of the low point, possible by the small number of bits required for the width.