Floating-point arithmetic and error analysis (AFAE)

Multiprecision and Interval Arithmetic

Stef Graillat

LIP6/PEQUAN - Sorbonne University

Lecture Master 2 SFPN - MAIN5

Outline

Outline

Bibliography I

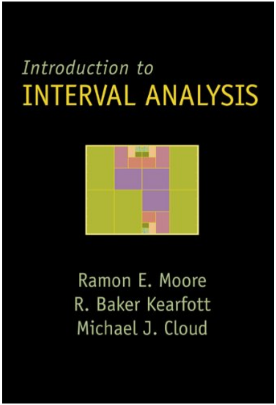
Ramon E. Moore.
Interval analysis.
Prentice-Hall Inc., Englewood Cliffs, N.J., 1966.

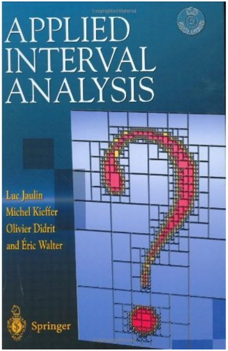
Luc Jaulin, Michel Kieffer, Olivier Didrit et Éric Walter.
Applied interval analysis.
Springer-Verlag London Ltd., London, 2001.

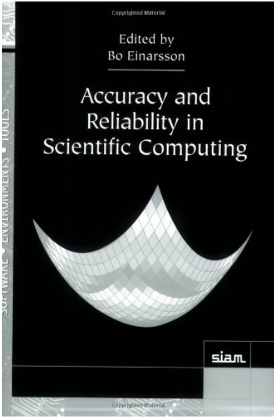
R. Moore, R. Kearfott et M. Cloud
Introduction to Interval Analysis.
SIAM, 2009

Arnold Neumaier.
Interval methods for systems of equations.
Cambridge University Press, Cambridge, 1990.

References







Verification methods: Rigorous results using floating-point arithmetic, Siegfried M. Rump, Acta Numerica (2010), pp. 287–449

Historical remarks

Who invented Interval Arithmetic?

- Ramon Moore in 1962 - 1966 ?
- T. Sunaga in 1958 ?
- Rosalind Cecil Young in 1931 ?

Cf. `http://www.cs.utep.edu/interval-comp/`

Popularization in the 1980, German school (U. Kulisch).

IEEE-754 standard for floating-point arithmetic in 1985: directed roundings are standardized and available.

Since the nineties: interval algorithms.

Directed roundings

Let

$$x_1 = \nabla(1/3), \quad x_2 = \Delta(1/3)$$

Then we mathematically have

$$x_1 \leq 1/3 \leq x_2 \quad \text{with} \quad x_1, x_2 \in \mathbb{F}$$

More general $a, b \in \mathbb{F}$, we have :

$$\nabla(a \circ b) \leq a \circ b \leq \Delta(a \circ b)$$

for $\circ \in \{+, -, \times, /\}$

Directed roundings (cont'd)

With INTLAB

`setround(-1)`	rounding downwards
`setround(1)`	rounding upwards
`setround(0)`	rounding to nearest

Example:

```
setround(-1)
x = 1/3
setround(1)
y = 1/3
```

Then we have the mathematical inequality

$$x \leq 1/3 \leq y$$

Interval arithmetic

- Interval arithmetic: replace numbers by intervals and compute.
- Fundamental theorem of interval arithmetic: the exact result belongs to the computed interval.
- No result is lost, the computed interval is guaranteed to contain every possible result.

Interval arithmetic

Interval Arithmetic and validated scientific computing: two directions

1. replace floating-point arithmetic by interval arithmetic to bound from above roundoff errors;
2. replace floating-point arithmetic and algorithms by interval ones to compute guaranteed enclosures.

Interval arithmetic

Interval arithmetic: replace numbers by intervals and compute.

Initially introduced to take into account roundoff errors (Moore 1966) and also uncertainties (on the physical data, etc), then computations with sets.

Interval analysis: develop algorithms for reliable (or verified, or guaranteed) computing, that are suited for interval arithmetic,
i.e. different from the algorithms from classical numerical analysis.

Examples of applications

- control the roundoff errors, cf. computational geometry
- solve several problems with verified solutions: linear and nonlinear systems of equations and inequations, constraints satisfaction, (non/convex, un/constrained) global optimization, etc.
- mathematical proofs: cf. Hales' proof of the Kepler's conjecture

Definitions and notation

Objects

- interval of real numbers: closed connected sets of $\mathbb{R}$
  - interval for π: $[3.14159, 3.14160]$
  - data d known with absolute uncertainty of ε: $[d-\varepsilon, d+\varepsilon]$
- interval vector

$$\mathbf{v} = \begin{pmatrix} [1,2] \\ [2,4] \end{pmatrix}$$

- interval matrix

$$\mathbf{A} = \begin{pmatrix} [1,3] & [3,4] \\ [2,5] & [1,2] \end{pmatrix}$$

Representation inf-sup of intervals

$$\mathbf{x} = [\underline{x}; \overline{x}] = \{x \in \mathbb{R} : \underline{x} \le x \le \overline{x}\}.$$

The set of interval of $\mathbb{R}$ is denoted $\mathbb{IR}$.

Definitions (2/3)

- Representation inf-sup of intervals

$$\mathbf{x} = [\underline{x}; \overline{x}] = \{x \in \mathbb{R} : \underline{x} \leq x \leq \overline{x}\}.$$

- A real number is represented by $[x; x]$.
- Interval quantities are generally represented in boldface.
- The set of interval of $\mathbb{R}$ is denoted $\mathbb{IR}$.
- We denote by $\mathrm{mid}(\mathbf{x})$ the midpoint of an interval $\mathbf{x} = [\underline{x}; \overline{x}]$,

$$\mathrm{mid}(\mathbf{x}) = (\overline{x} + \underline{x})/2,$$

and by $w(\mathbf{x})$ the width of $\mathbf{x}$,

$$w(\mathbf{x}) = \overline{x} - \underline{x}.$$

Operations on intervals

Given two intervals $\mathbf{x}$, $\mathbf{y}$ and $\diamond \in \{+, -, \times, /\}$, one defines

$$\mathbf{x} \diamond \mathbf{y} = \{x \diamond y : x \in \mathbf{x}, y \in \mathbf{y}\}.$$

On can implement these operations as :

$$\begin{aligned}
\mathbf{x}+\mathbf{y} &= [\underline{x}+\underline{y}; \overline{x}+\overline{y}],\\
\mathbf{x}-\mathbf{y} &= [\underline{x}-\overline{y}; \overline{x}-\underline{y}],\\
\mathbf{x}\times\mathbf{y} &= [\min\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}; \max\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}],\\
\mathbf{x}^2 &= [\min(\underline{x}^2, \overline{x}^2),\ \max(\underline{x}^2, \overline{x}^2)] \text{ if } 0 \notin [\underline{x},\ \overline{x}],\\
& \quad [0, \max(\underline{x}^2, \overline{x}^2)] \text{ otherwise},\\
1/\mathbf{x} &= [1/\overline{x}; 1/\underline{x}] \text{ if } 0 \notin [\underline{x},\ \overline{x}],\\
\mathbf{x}/\mathbf{y} &= \mathbf{x} \times 1/\mathbf{y} \text{ if } 0 \notin [\underline{y},\ \overline{y}],\\
\sqrt{\mathbf{x}} &= [\sqrt{\underline{x}},\ \sqrt{\overline{x}}] \text{ if } 0 \le \underline{x},\\
& \quad [0,\ \sqrt{\overline{x}}] \text{ otherwise}.
\end{aligned}$$

Operations on intervals

In floating-point arithmetic, if one wants validated results, one need to take into account rounding errors !

$$\begin{aligned} \mathbf{x} + \mathbf{y} &= [\nabla(\underline{x} + \underline{y}), \Delta(\overline{x} + \overline{y})] \supseteq \{x + y | x \in \mathbf{x}, y \in \mathbf{y}\} \\ \mathbf{x} - \mathbf{y} &= [\nabla(\underline{x} - \overline{y}), \Delta(\overline{x} - \underline{y})] \supseteq \{x - y | x \in \mathbf{x}, y \in \mathbf{y}\} \end{aligned}$$

where ∇ (resp. Δ) representes rounding toward $-\infty$ (resp. rounding toward $+\infty$).

Operations on intervals (cont'd)

Algebraic properties : associativity and commutativity still hold

But other properties have been lost :

- the subtraction is not the inverse of addition : $\mathbf{x} - \mathbf{x} \neq [0]$
- the division is not the inverse of multiplication
- ...

Intervals and functions

Definition : an interval extension $\mathbf{f}$ of f must satisfy

$$\forall \mathbf{x}, f(\mathbf{x}) \subseteq \mathbf{f}(\mathbf{x}) \text{ et } \forall x, f(\{x\}) = \mathbf{f}(\{x\})$$

Elementary functions :

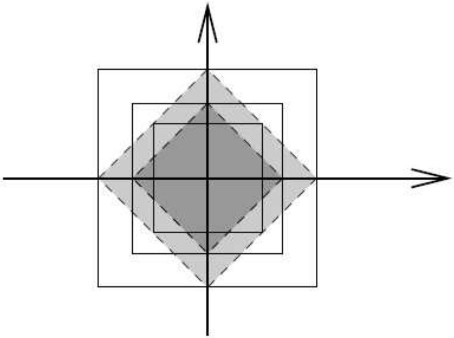
$$\begin{aligned} \exp \mathbf{x} &= [\exp \underline{x}, \exp \overline{x}] \\ \sin[\pi/6, 2\pi/3] &= [1/2, 1] \end{aligned}$$

Problem : data dependency

$$\mathbf{x} - \mathbf{x} = \{x - y : x \in \mathbf{x}, \quad y \in \mathbf{x}\} \neq \{0\}$$

Problem : wrapping effect

Effect of 2 rotations of $\pi/4$



Libraries

Libraries for scientific software:

- INTLAB for MATLAB/Octave;
- Mathematica;
- intpakX for Maple.

Well-know libraries integrating interval arithmetic for classic languages:

- XSC (eXtended Scientific Computing);
- “range” arithmetic;
- MPFI (Multiple Precision Floating-point Interval arithmetics library).

Interval Newton iteration

Computation of $\sqrt{2}$ by Newton iteration with solving the equation $f(x) = 0$ with $f(x) = x^2 - 2$.

Newton iteration: $x_{k+1} = x_k - (x_k^2 - 2)/(2x_k)$

```
>> x=1; for i=1:5, x = x - (x^2-2)/(2*x), end
     x = 1.500000000000000
     x = 1.416666666666667
     x = 1.414215686274510
     x = 1.414213562374690
     x = 1.414213562373095
```

Interval Newton iteration

Naive interval Newton iteration starting with $[1.4, 1.5]$

```
>> X=infsup(1.4,1.5);
>> for i=1:5, X = X - (X^2-2)/(2*X), end
intval X =
[    1.3107,    1.5143]
intval X =
[    1.1989,    1.6218]
intval X =
[    0.9359,    1.8565]
intval X =
[    0.1632,    2.4569]
intval X =
[  -12.2002,    8.5014]
```

Interval Newton iteration

Problem : data dependency

Instead of an inclusion of

$$\{x - (x^2 - 2)/(2x) : x \in \mathbf{X}_k\},$$

naive interval arithmetic computes an inclusion of

$$\{\xi_1 - (\xi_2^2 - 2)/(2\xi_3) : \xi_1, \xi_2, \xi_3 \in \mathbf{X}_k\}$$

Interval Newton iteration

Theorem

Let a differentiable function $f : \mathbb{R} \to \mathbb{R}$, $\mathbf{X} = [x_1, x_2] \in \mathbb{IR}$ and $\widetilde{x} \in \mathbf{X}$ be given, and suppose $0 \notin f'(\mathbf{X})$. Using interval operations, define

$$N(\widetilde{x}, \mathbf{X}) := \widetilde{x} - f(\widetilde{x})/f'(\mathbf{X}).$$

If $N(\widetilde{x}, \mathbf{X}) \subset \mathbf{X}$, then $\mathbf{X}$ contains a unique root of f. If $N(\widetilde{x}, \mathbf{X}) \cap \mathbf{X} = \emptyset$ then $f(x) \neq 0$ for all $x \in \mathbf{X}$.

Proof :

If $N(\widetilde{x}, \mathbf{X}) \subset \mathbf{X}$ then $x_1 \leq \widetilde{x} - f(\widetilde{x})/f'(\xi) \leq x_2$ for all $\xi \in \mathbf{X}$.
Therefore $0 \notin f'(\mathbf{X})$ implies

$$(f(\widetilde{x}) + f'(\xi_1)(x_1 - \widetilde{x})) \cdot (f(\widetilde{x}) + f'(\xi_2)(x_2 - \widetilde{x})) \leq 0$$

for all $\xi_1, \xi_2 \in \mathbf{X}$ and in particular $f(x_1) \cdot f(x_2) \leq 0$.

Interval Newton iteration

So there is a root of f in $\mathbf{X}$, which is unique because $0 \notin f'(\mathbf{X})$.

Suppose $\widehat{x} \in \mathbf{X}$ is a root f.

By the Mean Value Theorem, there exists $\xi \in \mathbf{X}$ with $f(\widetilde{x}) = f'(\xi)(\widetilde{x} - \widehat{x})$.

So $\widehat{x} = \widetilde{x} - f(\widetilde{x})/f'(\xi)$.

Interval Newton iteration

The assumptions of the Theorem are verified as follows.

1. Let $\mathbf{F}$ and $\mathbf{Fs}$ be interval extensions of f and f', respectively.
2. If $\mathbf{Fs}(\mathbf{X})$ does not contain zero 0, then $f'(x) \neq 0$ for $x \in \mathbf{X}$.
3. If $\widetilde{x} - \mathbf{F}(\widetilde{x})/\mathbf{Fs}(\mathbf{X}) \subset \mathbf{X}$ then $\mathbf{X}$ contains a unique roots of f.
4. If $\{\widetilde{x} - \mathbf{F}(\widetilde{x})/\mathbf{Fs}(\mathbf{X})\} \cap \mathbf{X} = \emptyset$, then $\mathbf{X}$ contains no roots of f.

Interval Newton iteration

```
>> X=infsup(1,2);
   for i=1:4, xs=intval(mid(X)); X = xs - (xs^2-2)/(2*X), end
intval X =
[   1.37499999999999,   1.43750000000001]
intval X =
[   1.41406249999999,   1.41441761363637]
intval X =
[   1.41421355929452,   1.41421356594718]
intval X =
[   1.41421356237309,   1.41421356237310]
```

Proving that a matrix is nonsingular

Theorem

Let A be a matrix and R another matrix such that $\|I - RA\| < 1$. Then A is nonsingular

Proof.

By contrapositive, if A is singular, there exists $x \neq 0$ such that $Ax = 0$. Then $(I - RA)x = x$ and so $\|I - RA\| \geq 1$. $\square$

On a computer, choose for $R \approx A^{-1}$ and then compute $\|I - RA\|$ with interval arithmetic.

Proving that a matrix is nonsingular with INTLAB

Let A be a matrix of dimension n

```
R = inv(A)
C = eye(n) - R*intval(A)
nonsingular = ( norm(C,1) < 1 )
```

If nonsingular = 1, then A is nonsingular.
If nonsingular = 0, then we can say nothing

A simple approach

Let $f : \mathbb{R}^n \to \mathbb{R}^n$ and $\widehat{x} \in \mathbb{R}^n$ unknown such that $f(\widehat{x}) = 0$

Let $\widetilde{x} \approx \widehat{x}$ such that $f(\widetilde{x}) \approx 0$

Find a bound for $\widetilde{x}$: an interval X such that $\widehat{x} \in X$

We have

$$f(x) = 0 \qquad \Leftrightarrow \qquad g(x) = x$$

with $g(x) := x - Rf(x)$ with $\det(R) \neq 0$.

Theorem (Brouwer, 1912)

Every continuous function from a closed ball of a Euclidean space to itself has a fixed point.

A simple approach (cont'd)

By Brouwer fixed point theorem,

$$X \in \mathbb{IR}^n, \quad g(X) \subseteq X \quad \Rightarrow \quad \exists \widehat{x} \in X, \quad g(\widehat{x}) = \widehat{x} \quad \Rightarrow \quad f(\widehat{x}) = 0$$

We just have to check $g(X) \subseteq X$ and prove $\det(R) \neq 0$.

But naive approach fails:

$$g(X) \subseteq X - Rf(X) \not\subseteq X$$

Bounds for the solution of nonlinear systems

Mean Value Theorem:

$$\text{if } f \in \mathcal{C}^1 \text{ then } f(x) = f(\widetilde{x}) + M(x - \widetilde{x}) \text{ with } M = \left(\frac{\partial f}{\partial x}(\xi_i)\right)_i$$

Let $Y := X - \widetilde{x}$ and

$$\begin{aligned} x \in X \quad \Rightarrow \quad g(x) - \widetilde{x} &= x - \widetilde{x} - Rf(x) \\ &= -Rf(\widetilde{x}) + (I - RM)(x - \widetilde{x}) \\ &\in -Rf(\widetilde{x}) + (I - RM)Y \end{aligned}$$

As a consequence

$$-Rf(\widetilde{x}) + (I - RM)Y \subseteq Y \quad \Rightarrow \quad g(X) - \widetilde{x} \subseteq Y \quad \Rightarrow \quad g(X) \subseteq X$$

Bounds for the solution of nonlinear systems

Theorem

Let $f : \mathbb{R}^n \to \mathbb{R}^n$ with $f = (f_1, \ldots, f_n) \in \mathcal{C}^1$, $\tilde{x} \in \mathbb{R}^n$, $X \in \mathbb{IR}^n$ with $0 \in X$ and $R \in \mathbb{R}^{n\times n}$ be given. Let $M \in \mathbb{IR}^{n\times n}$ be given such that

$$\{\nabla f_i(\zeta) : \zeta \in \tilde{x} + X\} \subseteq M_{i,:} \ .$$

Assume

$$-Rf(\tilde{x}) + (I - RM)X \subseteq \mathrm{int}(X).$$

Then there is a unique $\hat{x} \in \tilde{x} + X$ with $f(\hat{x}) = 0$. Moreover, every matrix $\tilde{M} \in M$ is nonsingular. In particular, the Jacobian $J_f(\hat{x}) = \frac{\partial f}{\partial x}(\hat{x})$ is nonsingular.

Verification of multiple roots

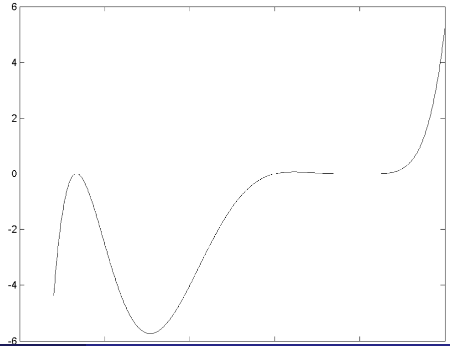
- Verification method for computing guaranteed (real or complex) error bounds for double roots of systems of nonlinear equations.
- To circumvent the problem of ill-posedness we prove that a slightly perturbed system of nonlinear equations has a double root.
- For example, for a given univariate function $f : \mathbb{R} \to \mathbb{R}$ we compute two intervals $X, E \subseteq \mathbb{R}$ with the property that there exists $\widehat{x} \in X$ and $\widehat{e} \in E$ such that $\widehat{x}$ is a double root of $\overline{f}(x) := f(x) - \widehat{e}$.

Verification of multiple roots

The typical scenario in the univariate case is a function $f : \mathbb{R} \to \mathbb{R}$ with a double root $\widehat{x}$, i.e. $f(\widehat{x}) = f'(\widehat{x}) = 0$ and $f''(\widehat{x}) \neq 0$.

Consider, for example,

$$\begin{aligned} f(x) &= 18x^7 - 183x^6 + 764x^5 - 1675x^4 + 2040x^3 - 1336x^2 + 416x - 48 \\ &= (3x-1)^2(2x-3)(x-2)^4 \end{aligned}$$



Verification of multiple roots

- Verification methods for multiple roots of polynomials already exist. A set containing k roots of a polynomial is computed, but no information on the true multiplicity can be given.

- Algorithm `verifypoly` in INTLAB. Computing inclusions `X1`, `X2` and `X3` of the simple root $x_1 = 1.5$, the double root $x_2 = 1/3$ and the quadruple root $x_3 = 2$ of f:

```
>> X1 = verifypoly(f,1.3), X2 = verifypoly(f,.3), X3 = verifypoly(f,2.1)
intval X1 =
[   1.49999999999904,   1.50000000000078]
intval X2 =
[   0.33333316656015,   0.33333343640539]
intval X3 =
[    1.99741678159164,   2.00363593397305]
```

Verification of multiple roots (cont'd)

- The accuracy of the inclusion of the double root $x_2 = 1/3$ is much less than that of the simple root $x_1 = 1.5$, and this is typical.
- Perturb f into $\widetilde{f}(x) := f(x) - \varepsilon$ for some small real constant ε and look at a perturbed root $\widetilde{f}(\widehat{x} + h)$ of $\widetilde{f}$, then

$$0 = \widetilde{f}(\widehat{x} + h) = -\varepsilon + \frac{1}{2} f''(\widehat{x}) h^2 + \mathcal{O}(h^3)$$

implies

$$h \sim \sqrt{2\varepsilon / f''(\widehat{x})}.$$

- A relative error of size $\varepsilon \approx 10^{-16}$ implies a relative accuracy of $\sqrt{\varepsilon} \approx 10^{-8}$.

Dealing with double roots

- We consider for a double root the nonlinear system $G : \mathbb{R}^2 \to \mathbb{R}$ with

$$G(x,e) = \begin{pmatrix} f(x) - e \\ f'(x) \end{pmatrix} = 0$$

  in the two unknowns x and e.

- The Jacobian of this system is

$$J_G(x,e) = \begin{pmatrix} f'(x) & -1 \\ f''(x) & 0 \end{pmatrix},$$

  so that the nonlinear system is well-conditioned for the double root $x_2 = 1/3$ of f.

Dealing with double roots (cont'd)

- We provide an algorithm `verifynlss` in INTLAB.

```
>> Y2 = verifynlss(G,[.3;0])
intval Y2 =
[  3.33333333333328e-001,  3.33333333333337e-001]
[ -2.13162820728424e-014,  2.13162820728420e-014]
```

- This proves that there is a constant ε with $|\varepsilon| \leq 2.14 \cdot 10^{-14}$ such that the nonlinear equation $f(x) - \varepsilon = 0$ has a double root $\widehat{x}$ with $0.333333333333328 \leq \widehat{x} \leq 0.333333333333337$.

Outline

The number of correct digits of a computed result

"Rule of thumb" :

$$\boxed{\text{forward error} \approx \text{condition number} \times \text{backward error}}$$

number of correct digits

$$\text{Number of correct digits} = -\log_{10}(\text{forward error})$$

The number of correct digits of a computed result

Problem : determination of a multiple root x_* of multiplicity m of a polynomial $p(x) = \sum_{i=0}^{n} a_i x^i$.

A relative perturbation of u on a_i leads to a perturbation

$$x_*(u) - x_* = u^{1/m} \left[-\frac{m! a_i x_*^i}{p^{(m)}(x_*)} \right]^{1/m}$$

Multiple roots : always ill-conditioned, forward error of order $u^{1/m}$

Multiple precision vs arbitrary precision

Arbitrary precision
used for integer or rational arithmetic, where the representation sizes of the operands vary arbitrarily and can be arbitrarily large.

Multiple precision
used for floating-point arithmetic, where the lengths of the mantissas and exponents are fixed but can be arbitrarily large.

Applications

Either a bit more accuracy than floating-point computations, and thus a bit more computing precision (several hundreds of bits)

or extreme computations, such as

- the computation of the largest number of digits of π,
- checking some special cases to prove theorems,
- determining a counter-example to a conjecture.

Representation : with integers

A multiple-precision floating-point number is a number of the form

$$s.m.\beta^e.$$

The mantissa (of arbitrary length) is represented as an exact integer.

Exact integers may be represented as a sequence of machine integers (cf. GMP):

$$m = \sum_{i=0}^{n} m_i B^i,$$

where m_i are machine integers and B is the length of a machine word.

Representation : expansions

Representation using floating-point numbers: non-evaluated sum of floating-point numbers

$$\sum_{i=0}^{n} f_i$$

where the f_i are floating-point numbers, if possible with exponents sufficiently wide apart so that the mantissas do not overlap.

Multiple precision libraries

- Multiple precision libraries with multiple-digit format : a number is expressed as a sequence of digits coupled with a single exponent

  MPFR : `www.mpfr.org/`
  ARPREC : `crd.lbl.gov/~dhbailey/mpdist/`
  GMP : `gmplib.org/`

- Fixed precision libraries using a multiple-component format with a limited numbers of components
  Classic example is double-double (a double-double is a non-evaluated sum of 2 doubles)

  DD : `crd.lbl.gov/~dhbailey/mpdist/`
  QD : `crd.lbl.gov/~dhbailey/mpdist/`

Multiple precision libraries

- Multiple precision libraries using a multiple-component format where a number is expressed as unevaluated sums of ordinary floating-point words.

  Shewchuk : `http://www.cs.cmu.edu/~quake/robust.html`
  Priest

double-double library

A double-double number is a non-evaluated pair (a_h, a_l) of IEEE 754 floating-point numbers satisfying $a = a_h + a_l$ et $|a_l| \leq \mathbf{u}|a_h|$.

Algorithm (Addition of a double b and a double-double (a_h, a_l))

function $[c_h, c_l] = \mathtt{add\_dd\_d}(a_h, a_l, b)$
  $[t_h, t_l]$ = $\mathtt{TwoSum}(a_h, b)$
  $[c_h, c_l]$ = $\mathtt{FastTwoSum}(t_h, (t_l \oplus a_l))$

double-double library

Algorithm (Product of a double-double (a_h, a_l) by a double b)

function $[c_h, c_l]$ = $\mathtt{prod\_dd\_d}(a_h, a_l, b)$

$[s_h, s_l] = \mathtt{TwoProduct}(a_h, b)$

$[t_h, t_l] = \mathtt{FastTwoSum}(s_h, (a_l \otimes b))$

$[c_h, c_l] = \mathtt{FastTwoSum}(t_h, (t_l \oplus s_l))$

double-double library

Algorithm (Addition of a double-double (a_h, a_l) with a double-double (b_h, b_l))

function $[c_h, c_l] = \mathtt{add\_dd\_dd}(a_h, a_l, b_h, b_l)$

$[s_h, s_l]$ = $\mathtt{TwoSum}(a_h, b_h)$

$[t_h, t_l]$ = $\mathtt{TwoSum}(a_l, b_l)$

$[t_h, s_l] = \mathtt{FastTwoSum}(s_h, (s_l \oplus t_h))$

$[c_h, c_l] = \mathtt{FastTwoSum}(t_h, (t_l \oplus s_l))$

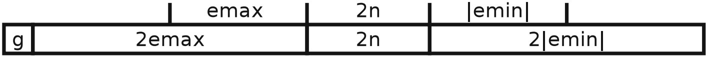
If a, b are double-double and $\circledcirc \in \{+, \times\}$, then we have

$$\mathrm{fl}(a \circledcirc b) = (1 + \delta)(a \circledcirc b),$$

with $|\delta| \leq 4 \cdot 2^{-106}$.

Kulisch accumulator

Computing without error due to the limited range of floating-point numbers



In double precision, $n = 53$ bits, $emin = -1022$, $emax = 1023$ and $k = 92$ bits

A register of length $L = k + 2emax + 2|emin| + 2n = 4288$ bits is sufficient (67 words of 64 bits)

Kulisch accumulator