

Calcul précis et certifié de racines de polynômes : le cas des racines simples et multiples

Stef Graillat

LIP6/PEQUAN - Université Pierre et Marie Curie (Paris 6) - CNRS
en délégation CNRS au LIP/AriC - ENS Lyon - Inria - CNRS - UCBL

Séminaire Calcul Formel, laboratoire XLIM

Limoges, 27 mars 2014



A famous failure: Patriot missile

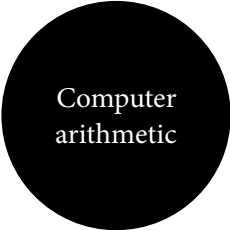
- $1/10$ is not representable by a finite number of digits in basis 2
 $1/10 = 0.00011001100110011001100110011001100\dots$
- On a 24 bit fixed-point register
error = $1.1001100\dots \times 2^{-24} \approx 0.000000095$ in decimal

A famous failure: Patriot missile

- $1/10$ is not representable by a finite number of digits in basis 2
 $1/10 = 0.00011001100110011001100110011001100\dots$
- On a 24 bit fixed-point register
error = $1.1001100\dots \times 2^{-24} \approx 0.000000095$ in decimal

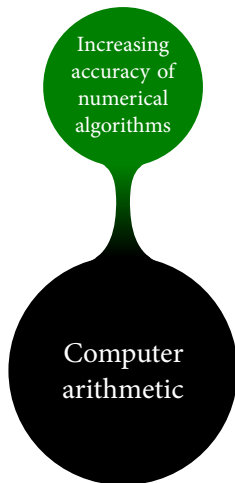


- First Gulf War in 1991: an american Patriot missile battery failed to intercept an Iraqi Scud missile. The Scud killed 28 soldiers.
- After 100 hours, the error is about 0.34 s: a Scud travels at about 1500 m/s, it makes 500 m

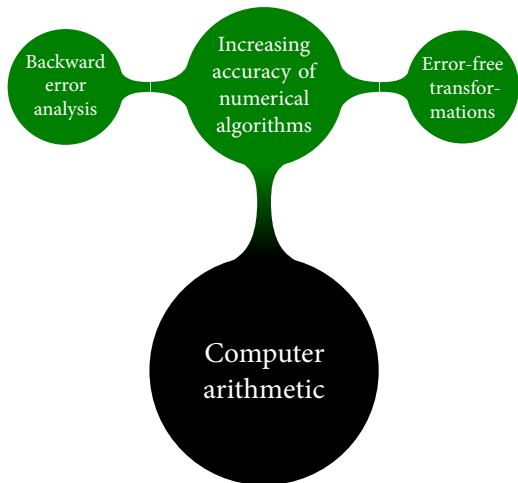


Computer
arithmetic

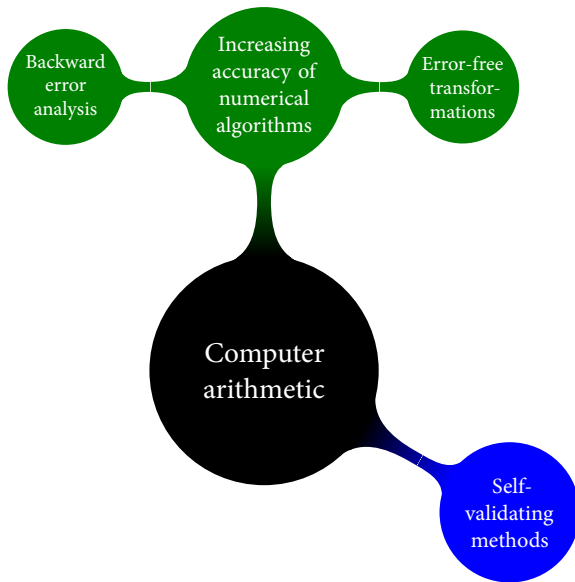
Research topic: compute *fast* and *accurately*



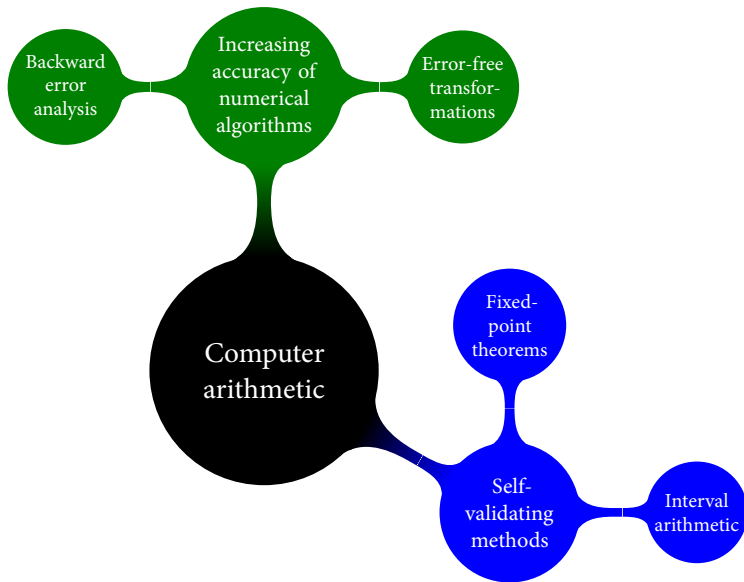
Research topic: compute *fast* and *accurately*



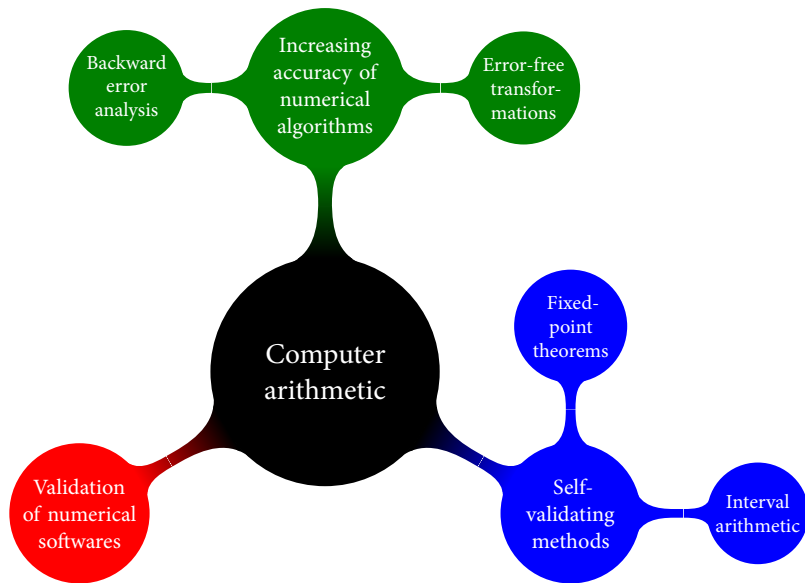
Research topic: compute *fast* and *accurately*



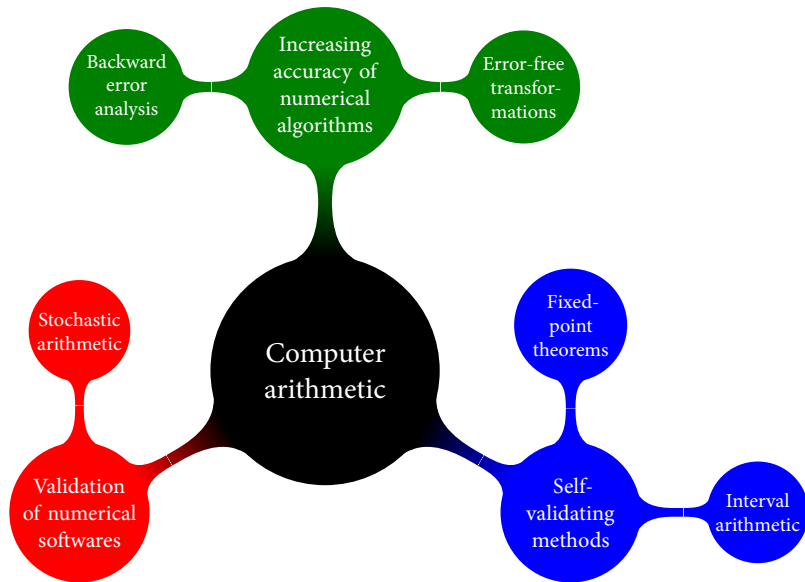
Research topic: compute *fast* and *accurately*



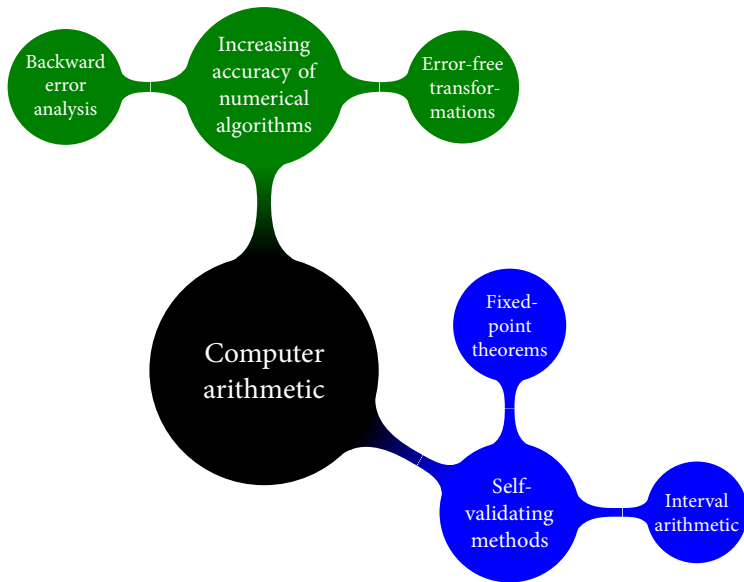
Research topic: compute *fast* and *accurately*



Research topic: compute *fast* and *accurately*



Research topic: compute *fast* and *accurately*



Outline

- 1 Introduction
- 2 Floating-point arithmetic
- 3 Error analysis and increase of accuracy
- 4 Interval analysis and self-validating methods
- 5 Conclusion and perspectives

Outline

- 1 Introduction
- 2 Floating-point arithmetic
- 3 Error analysis and increase of accuracy
- 4 Interval analysis and self-validating methods
- 5 Conclusion and perspectives

Standard model of floating-point arithmetic

Assume floating-point arithmetic adhering IEEE 754 with **rounding to nearest** with unit roundoff \mathbf{u} (no underflow nor overflow)

Floating point system $\mathbb{F} \subset \mathbb{R}$:

$$x = \pm x_0.x_1 \dots x_{p-1} \times b^e, \quad 0 \leq x_i \leq b-1, \quad x_0 \neq 0$$

With $x, y \in \mathbb{F}$ and $\circ \in \{+, -, \times, /\}$, $x \circ y$ is not in general a floating-point number

$$\text{fl}(x \circ y) = (x \circ y)(1 + \delta), \quad |\delta| \leq \mathbf{u}$$

IEEE 754 standard (1985,2008)

Type	Size	Significand	Exponent	Unit roundoff	Range
binary32	32 bits	23+1 bits	8 bits	$\mathbf{u} = 2^{-24} \approx 5.96 \times 10^{-8}$	$\approx 10^{\pm 38}$
binary64	64 bits	52+1 bits	11 bits	$\mathbf{u} = 2^{-53} \approx 1.11 \times 10^{-16}$	$\approx 10^{\pm 308}$

We denote: $\gamma_n := n\mathbf{u}/(1 - n\mathbf{u}) \approx n\mathbf{u}$

Understanding the difficulties when computing with finite precision

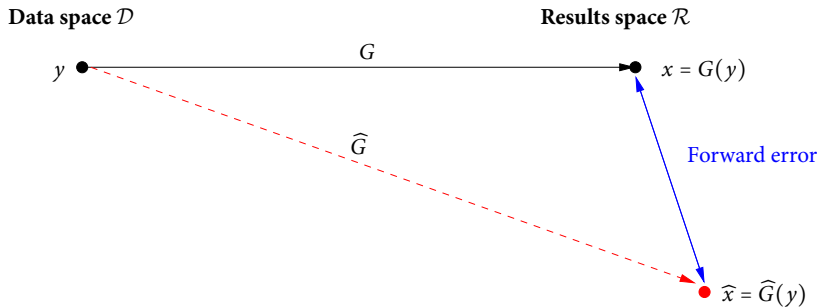
- Controlling the effects of finite precision:
 - How to measure the **difficulty of solving** the problem?
 - How to characterize the **reliability of the algorithm**?
 - How to estimate the **accuracy of the computed solution**?
- Limiting the effects of finite precision
 - How to **improve the accuracy of the solution**?

How to answer these questions?

Outline

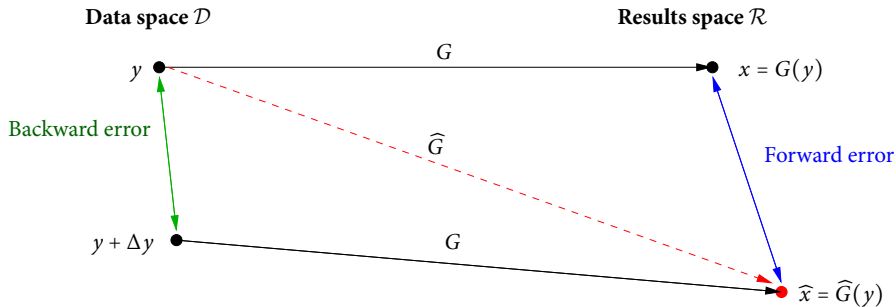
- 1 Introduction
- 2 Floating-point arithmetic
- 3 Error analysis and increase of accuracy**
- 4 Interval analysis and self-validating methods
- 5 Conclusion and perspectives

Error analysis (Wilkinson, Higham)



- Forward error analysis

Error analysis (Wilkinson, Higham)



- Forward error analysis
- Backward error analysis

Identify \hat{x} as the solution of a perturbed problem: $\hat{x} = G(y + \Delta y)$.

Advantages of backward error analysis

- **How to measure the difficulty of solving the problem ?**

Condition number measures the sensitivity of the solution to perturbation in the data

Condition number : $\text{cond}(P, y) := \lim_{\varepsilon \rightarrow 0} \sup_{\|\Delta y\| \leq \varepsilon} \left\{ \frac{\|\Delta x\|_{\mathcal{R}}}{\|\Delta y\|_{\mathcal{D}}} \right\}$

Advantages of backward error analysis

- **How to measure the difficulty of solving the problem ?**

Condition number measures the sensitivity of the solution to perturbation in the data

Condition number : $\text{cond}(P, y) := \lim_{\varepsilon \rightarrow 0} \sup_{\|\Delta y\| \leq \varepsilon} \left\{ \frac{\|\Delta x\|_{\mathcal{R}}}{\|\Delta y\|_{\mathcal{D}}} \right\}$

- **How to appreciate the reliability of the algorithm?**

Backward error measures the distance between the problem we solved and the initial problem.

Backward error : $\eta(\widehat{x}) = \min_{\Delta y \in \mathcal{D}} \{ \|\Delta y\|_{\mathcal{D}} : \widehat{x} = G(y + \Delta y) \}$

Advantages of backward error analysis

- **How to measure the difficulty of solving the problem ?**

Condition number measures the sensitivity of the solution to perturbation in the data

$$\text{Condition number} : \text{cond}(P, y) := \lim_{\varepsilon \rightarrow 0} \sup_{\|\Delta y\| \leq \varepsilon} \left\{ \frac{\|\Delta x\|_{\mathcal{R}}}{\|\Delta y\|_{\mathcal{D}}} \right\}$$

- **How to appreciate the reliability of the algorithm?**

Backward error measures the distance between the problem we solved and the initial problem.

$$\text{Backward error} : \eta(\widehat{x}) = \min_{\Delta y \in \mathcal{D}} \{ \|\Delta y\|_{\mathcal{D}} : \widehat{x} = G(y + \Delta y) \}$$

- **How to estimate the accuracy of the computed solution?**

At first order, the rule of thumb:

$$\text{forward error} \lesssim \text{condition number} \times \text{backward error}.$$

Advantages of backward error analysis

- **How to measure the difficulty of solving the problem ?**

Condition number measures the sensitivity of the solution to perturbation in the data

$$\text{Condition number} : \text{cond}(P, y) := \lim_{\varepsilon \rightarrow 0} \sup_{\|\Delta y\| \leq \varepsilon} \left\{ \frac{\|\Delta x\|_{\mathcal{R}}}{\|\Delta y\|_{\mathcal{D}}} \right\}$$

- **How to appreciate the reliability of the algorithm?**

Backward error measures the distance between the problem we solved and the initial problem.

$$\text{Backward error} : \eta(\widehat{x}) = \mathbf{u} \longrightarrow \text{backward stable}$$

- **How to estimate the accuracy of the computed solution?**

At first order, the rule of thumb:

$$\text{forward error} \lesssim \text{condition number} \times \text{backward error}.$$

Advantages of backward error analysis

- **How to measure the difficulty of solving the problem ?**

Condition number measures the sensitivity of the solution to perturbation in the data

$$\text{Condition number} : \text{cond}(P, y) := \lim_{\varepsilon \rightarrow 0} \sup_{\|\Delta y\| \leq \varepsilon} \left\{ \frac{\|\Delta x\|_{\mathcal{R}}}{\|\Delta y\|_{\mathcal{D}}} \right\}$$

- **How to appreciate the reliability of the algorithm?**

Backward error measures the distance between the problem we solved and the initial problem.

$$\text{Backward error} : \eta(\widehat{x}) = \mathbf{u} \longrightarrow \text{backward stable}$$

- **How to estimate the accuracy of the computed solution?**

At first order, the rule of thumb:

$$\text{forward error} \lesssim \text{condition number} \times \mathbf{u}.$$

Achieving more accuracy with compensated algorithms

Key tools for accurate computation

- fixed length expansions libraries: double-double (Briggs, Bailey, Hida, Li), quad-double (Bailey, Hida, Li)
- arbitrary length expansions libraries: Priest, Shewchuk
- arbitrary multiprecision libraries: MP, MPFUN/ARPREC, MPFR
- compensated algorithms (e.g. Kahan, Priest, Ogita-Rump-Oishi)

Error-free transformations (EFT) (Dekker, Knuth) are properties and algorithms to compute the elementary rounding errors,

$$a, b \in \mathbb{F}, \quad a \circ b = \text{fl}(a \circ b) + e, \text{ and } e \in \mathbb{F}$$

EFT for addition

$$x = a \oplus b \Rightarrow a + b = x + y \quad \text{with } y \in \mathbb{F},$$

Algorithm of Dekker (1971) and Knuth (1974)

Algorithm (EFT of the sum of 2 floating-point numbers)

```
function [x, y] = TwoSum(a, b)
```

$$x = a \oplus b$$

$$z = x \ominus a$$

$$y = (a \ominus (x \ominus z)) \oplus (b \ominus z)$$

EFT for multiplication

$$x = a \otimes b \Rightarrow a \times b = x + y \quad \text{with } y \in \mathbb{F},$$

Given $a, b, c \in \mathbb{F}$,

- $\text{FMA}(a, b, c)$ is the nearest floating-point number $a \cdot b + c \in \mathbb{F}$

Algorithm (EFT of the product of 2 floating-point numbers)

```
function  $[x, y] = \text{TwoProduct}(a, b)$ 
```

$$x = a \otimes b$$

$$y = \text{FMA}(a, b, -x)$$

The FMA is available for example on PowerPC, Itanium, Cell, Xeon Phi, Haswell processors.

Horner scheme

Algorithm

function res = Horner(p, x) % $p(x) = \sum_{i=0}^n a_i x^i$

$s_n = a_n$

 for $i = n - 1 : -1 : 0$

$p_i = s_{i+1} \otimes x$

$s_i = p_i \oplus a_i$

 end

 res = s_0

Condition number for the evaluation of $p(x)$:

$$\text{cond}(p, x) = \frac{\sum_{i=0}^n |a_i| |x|^i}{|\sum_{i=0}^n a_i x^i|} = \frac{\tilde{p}(|x|)}{|p(x)|}$$

Relative error bound: $\frac{|p(x) - \text{Horner}(p, x)|}{|p(x)|} \leq \underbrace{\gamma_{2n}}_{\approx 2n\mathbf{u}} \text{cond}(p, x)$

Horner scheme

Algorithm

```
function res = Horner(p, x)      %  $p(x) = \sum_{i=0}^n a_i x^i$   
     $s_n = a_n$   
    for  $i = n - 1 : -1 : 0$   
         $p_i = s_{i+1} \otimes x$       % rounding error  $\pi_i$   
         $s_i = p_i \oplus a_i$       % rounding error  $\sigma_i$   
    end  
    res =  $s_0$ 
```

Condition number for the evaluation of $p(x)$:

$$\text{cond}(p, x) = \frac{\sum_{i=0}^n |a_i| |x|^i}{|\sum_{i=0}^n a_i x^i|} = \frac{\tilde{p}(|x|)}{|p(x)|}$$

$$\text{Relative error bound: } \frac{|p(x) - \text{Horner}(p, x)|}{|p(x)|} \leq \underbrace{\gamma_{2n}}_{\approx 2n\mathbf{u}} \text{cond}(p, x)$$

EFT for Horner scheme

Algorithm (with Langlois, Louvet, JJIAM 2008)

function [Horner(p, x), p_π, p_σ] = EFTHorner(p, x)

$s_n = a_n$

for $i = n - 1 : -1 : 0$

$[p_i, \pi_i] = \text{TwoProduct}(s_{i+1}, x)$

$[s_i, \sigma_i] = \text{TwoSum}(p_i, a_i)$

end

Horner(p, x) = s_0

$$p_\pi(x) = \sum_{i=0}^{n-1} \pi_i x^i, \quad p_\sigma(x) = \sum_{i=0}^{n-1} \sigma_i x^i$$

$$p(x) = \text{Horner}(p, x) + (p_\pi + p_\sigma)(x)$$

Compensated Horner scheme (CHS) and its accuracy

Algorithm (CHS, with Langlois, Louvet, JJIAM 2008)

```
function res = CompHorner(p, x)
[h, pπ, pσ] = EFTHorner(p, x)
c = Horner(pπ ⊕ pσ, x)
res = h ⊕ c
```

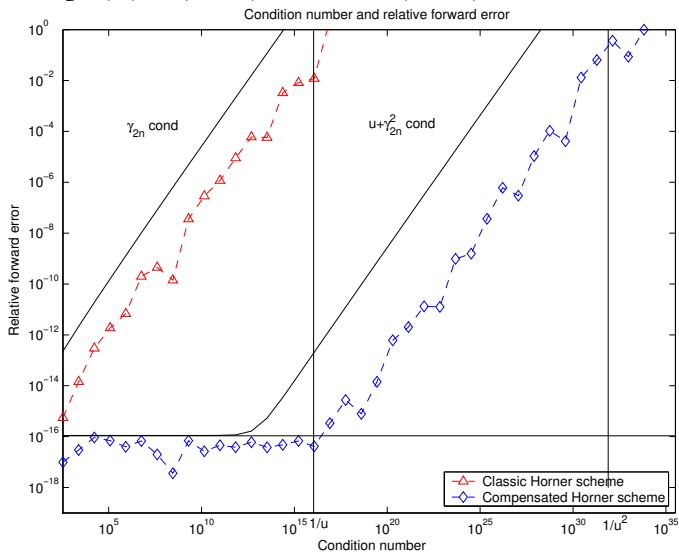
Theorem (with Langlois, Louvet, JJIAM 2008)

Let p be a polynomial of degree n with floating point coefficients, and x be a floating point value. Then if no underflow occurs,

$$\frac{|\text{CompHorner}(p, x) - p(x)|}{|p(x)|} \leq \mathbf{u} + \underbrace{\gamma_{2n}^2}_{\approx 4n^2 \mathbf{u}^2} \text{cond}(p, x).$$

Numerical experiments: testing the accuracy

Evaluation of $p_n(x) = (x - 1)^n$ for $x = \text{fl}(1.333)$ and $n = 3, \dots, 42$



Numerical experiments: testing the speed efficiency

We compare

- **Horner**: IEEE 754 double precision Horner scheme
- **CompHorner**: our Compensated Horner scheme
- **DDHorner**: Horner scheme with internal double-double computation

All computations are performed in C and IEEE 754 double precision

ratio	minimum	mean	maximum	theoretical
CompHorner/Horner	1.5	2.9	3.2	13
DDHorner/Horner	2.3	8.4	9.4	17

Compensated Horner Derivative algorithm

The Horner Derivative (HD) algorithm is the **classic method** for the evaluation of the k -derivative of a polynomial $p(x)$

Algorithm (HD)

```
function res=HD(p, x, k)
   $y_i^j = 0$  for  $i = 0 : 1 : k$  and  $j = n + 1 : -1 : 0$ 
   $y_{-1}^{j+1} = a_j$  for  $j = n : -1 : 0$ 

  for  $j = n : -1 : 0$ 
    for  $i = \min(k, n - j) : -1 : \max(0, k - j)$ 
       $y_i^j = x \otimes y_i^{j+1} \oplus y_{i-1}^{j+1}$ 
    end
  end
  res =  $k! \otimes y_k^0$ 
```

Algorithm (CHD)

```
function res=CompHD(p, x, k)
   $y_i^j = 0, \widehat{\epsilon}y_i^j = 0$ , for  $i = 0 : 1 : k$ , and  $j = n + 1 : -1 : 0$ 
   $y_{-1}^{j+1} = a_j, \widehat{\epsilon}y_{-1}^{j+1} = 0$ , for  $j = n : -1 : 0$ 
  for  $j = n : -1 : 0$ 
    for  $i = \min(k, n - j) : -1 : \max(0, k - j)$ 
       $[s, \pi_i^j] = \text{TwoProd}(x, \widehat{y}_i^{j+1})$ 
       $[\widehat{y}_i^j, \sigma_i^j] = \text{TwoSum}(s, \widehat{y}_{i-1}^{j+1})$ 
       $\widehat{\epsilon}y_i^j = x \otimes \widehat{\epsilon}y_i^{j+1} \oplus \widehat{\epsilon}y_{i-1}^{j+1} \oplus (\pi_i^j \oplus \sigma_i^j)$ 
    end
  end
  res =  $(\widehat{y}_k^0 \oplus \widehat{\epsilon}y_k^0) \otimes k!$ 
```

(with Jiang and al., JCAM 2013)

Rounding error analysis of CHD algorithm

Theorem (with Jiang and al., JCAM 2013)

Let $p(x) = \sum_{i=0}^n a_i x^i$ be a polynomial of degree n with floating-point coefficients, and x a floating-point value (with $p^{(k)}(x) \neq 0$). The relative forward error bound in CHD algorithm is such that

$$\frac{|\text{CompHD}(p, x, k) - p^{(k)}(x)|}{|p^{(k)}(x)|} \leq 2\mathbf{u} + (k+1) \underbrace{\gamma_{2n} \gamma_{3n}}_{\approx 6n^2 \mathbf{u}^2} \text{cond}(p, x, k).$$

The **condition number for the k -th derivative evaluation** of a polynomial $p(x) = \sum_{i=0}^n a_i x^i$ at entry x is given by

$$\text{cond}(p, x, k) = \frac{k! \sum_{m=k}^n \binom{m}{k} |x|^{m-k} |a_m|}{|k! \sum_{m=k}^n \binom{m}{k} x^{m-k} a_m|} = \frac{\tilde{p}^{(k)}(x)}{|p^{(k)}(x)|},$$

Condition number for root finding

Definition (Chatelin et al., 1996)

Let $p(z) = \sum_{i=0}^n a_i z^i$ be a polynomial of degree n and x be a simple zero of p . The condition number of x is defined by

$$\text{cond}_{\text{root}}(p, x) = \limsup_{\varepsilon \rightarrow 0} \left\{ \frac{|\Delta x|}{\varepsilon |x|} : |\Delta a_i| \leq \varepsilon |a_i| \right\}.$$

Theorem (Chatelin et al., 1996)

Let $p(z) = \sum_{i=0}^n a_i z^i$ be a polynomial of degree n and x be a simple zero of p . The condition number of x is given by

$$\text{cond}_{\text{root}}(p, x) = \frac{\tilde{p}(|x|)}{|x| |p'(x)|},$$

with $\tilde{p}(x) = \sum_{i=0}^n |a_i| x^i$.

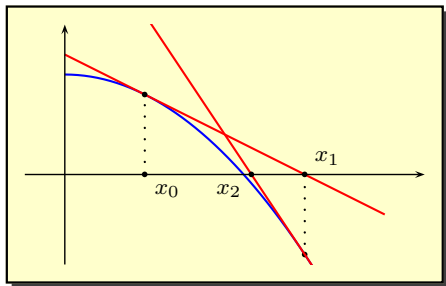
Application to Newton's method

Algorithm (The classic Newton's method)

$$x_0 = \xi$$

$$x_{i+1} = x_i - \frac{\text{Horner}(p, x_i)}{\text{HD}(p, x_i, 1)}$$

$$\frac{|x_{i+1} - x|}{|x|} \approx \gamma_{2n} \text{cond}_{\text{root}}(p, x) \quad [\text{Higham, 1996}]$$



Application to Newton's method

Algorithm (The accurate Newton's method)

$$x_0 = \xi$$
$$x_{i+1} = x_i - \frac{\text{CompHorner}(p, x_i)}{\text{HD}(p, x_i, 1)}$$

Theorem (Graillat, CMWA 2008)

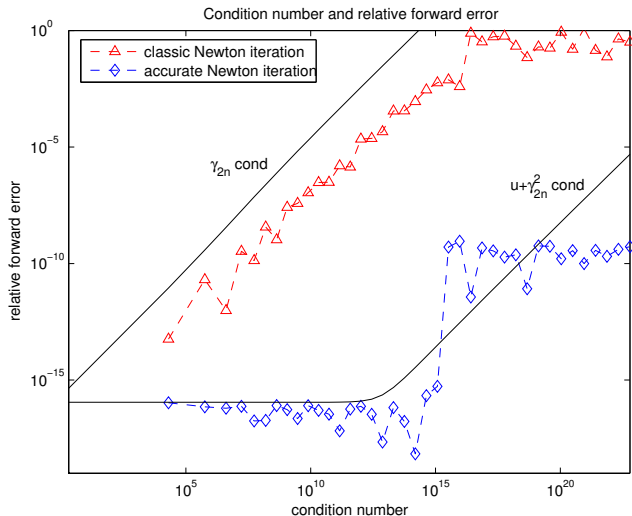
Assume that there is an x such that $p(x) = 0$ and $p'(x) \neq 0$ is not too small. Assume also that $\mathbf{u} \cdot \text{cond}_{\text{root}}(p, x) \leq 1/8$.

Then, for all x_0 such that $\beta |p'(x)|^{-1} |x_0 - x| \leq 1/8$, Newton's method in floating point arithmetic generates a sequence of $\{x_i\}$ whose relative error decreases until the first i for which

$$\frac{|x_{i+1} - x|}{|x|} \approx \mathbf{u} + \gamma_{2n}^2 \text{cond}_{\text{root}}(p, x).$$

Application to Newton's method

Test with $p_n(x) = (x - 1)^n - 10^{-8}$ and $x = 1 + 10^{-8/n}$ for $n = 1 : 40$
 $\text{cond}(p_n, x)$ varies from 10^4 to 10^{22}



Algorithm (The new accurate Newton's method)

$$x_0 = \xi$$

$$x_{i+1} = x_i - \frac{\text{CompHorner}(p, x_i)}{\text{CompHD}(p, x_i, 1)}$$

We proved that

- that the **convergence** of iterations strongly depends on the **accuracy of the derivative's evaluation** when the problem of finding simple root is too ill-conditioned, and
- that the **accuracy** of the final iteration result depends on the **accuracy with which the residual is computed**.

Application to Newton's method

We have shown (with Jiang and al., JCAM 2013) that

In case of classic Newton's algorithm:

$$\left| \frac{x_i - x}{x} \right| < C \gamma_{2n} \text{cond}_{\text{root}}(p, x).$$

In case of accurate Newton's algorithms:

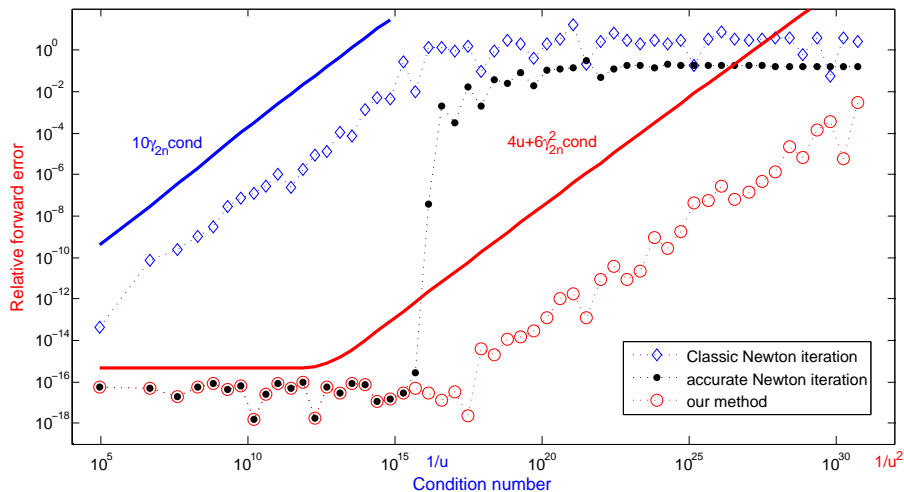
$$\left| \frac{x_i - x}{x} \right| < Ku + D \gamma_{2n}^2 \text{cond}_{\text{root}}(p, x).$$

where C , K and D are small factors.

Numerical experiments

Simple real zero of the expanded form of the polynomial

$$p_n(x) = (x - 1)^n - 2^{-31}, \text{ for } n = 2 : 55$$



Outline

- 1 Introduction
- 2 Floating-point arithmetic
- 3 Error analysis and increase of accuracy
- 4 Interval analysis and self-validating methods**
- 5 Conclusion and perspectives

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.

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

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

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

Operations on intervals

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

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

- One can show for example that :

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

$$\mathbf{x} \times \mathbf{y} = [\min(\underline{x}\underline{y}, \underline{x}\bar{y}, \bar{x}\underline{y}, \bar{x}\bar{y}), \max(\underline{x}\underline{y}, \underline{x}\bar{y}, \bar{x}\underline{y}, \bar{x}\bar{y})],$$

- In **floating-point arithmetic**, if one wants validated results, one needs to take into account rounding errors:

$$\mathbf{x} + \mathbf{y} = [\nabla(\underline{x} + \underline{y}), \Delta(\bar{x} + \bar{y})] \supseteq \{x + y | x \in \mathbf{x}, y \in \mathbf{y}\}$$

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

Proving that a matrix is nonsingular

Theorem (classic)

Let $A \in \mathbb{R}^{n \times n}$ be a matrix and $R \in \mathbb{R}^{n \times n}$ another matrix such that $\|I - RA\| < 1$. Then A is nonsingular

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

```
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 $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ 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

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

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,

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

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

Theorem (Rump, 1983)

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ with $f = (f_1, \dots, f_n) \in \mathcal{C}^1$, $\tilde{x} \in \mathbb{R}^n$, $X \in \mathbb{IR}^n$ 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

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

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

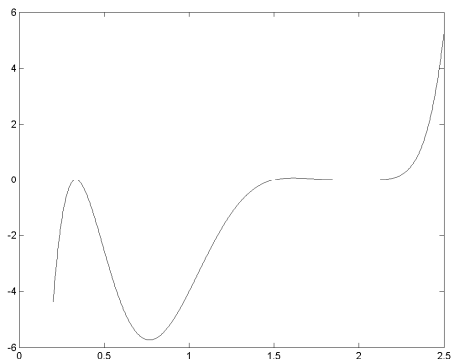
- Verification method for computing guaranteed (real or complex) error bounds for double roots of systems of nonlinear equations.
- To circumvent the problem of ill-posedness we prove that a slightly perturbed system of nonlinear equations has a double root.
- For example, for a given univariate function $f : \mathbb{R} \rightarrow \mathbb{R}$ 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 $\bar{f}(x) := f(x) - \widehat{e}$.

Verification of multiple roots

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

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 (Rump, 2003). 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 X_1 , X_2 and X_3 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 =
[ 1.499999999999904, 1.500000000000078]
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 $\tilde{f}(x) := f(x) - \varepsilon$ for some small real constant ε and look at a perturbed root $\tilde{f}(\hat{x} + h)$ of \tilde{f} , then

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

implies

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

- 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 $G : \mathbb{R}^2 \rightarrow \mathbb{R}$ 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 (with Rump, NA 2010) an algorithm `verifynlss` in INTLAB.

```
>> Y2 = verifynlss(G, [.3;0])
intval Y2 =
[ 3.3333333333333328e-001, 3.3333333333333337e-001]
[ -2.131628207280424e-014, 2.131628207280420e-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 \hat{x} with $0.3333333333333328 \leq \hat{x} \leq 0.3333333333333337$.

Outline

- 1 Introduction
- 2 Floating-point arithmetic
- 3 Error analysis and increase of accuracy
- 4 Interval analysis and self-validating methods
- 5 Conclusion and perspectives**

Conclusion

- **Compensated methods** are a fast way to get accurate results
They can be used to accurately evaluate **residuals**
- Self-validating methods can be an **efficient and fast** alternatives to obtain exact/certified results with finite precision arithmetic

- Increasing the accuracy:
 - a faithfully/correctly rounded Horner scheme
 - a correctly rounded 2-norm for vectors
 - “optimal” error bounds: like $|\text{fl}(x^n) - x^n| \leq (n - 1)\mathbf{u}x^n$
- Reproducibility and accuracy for exascale computing:
 - implementation of Kulisch accumulator on GPU and Xeon Phi
 - efficient exact dot product for multiprecision interval arithmetic
 - efficient reproducible BLAS libraries on GPU and Xeon Phi
- Numerical validation, certification and symbolic-numeric computation:
 - self-validating methods to detect singularities (approximate gcd, singular matrix, etc.)
 - automatic tool using stochastic arithmetic and delta-debugging for results with given accuracy