Accurate simple zeros of polynomials in floating point arithmetic

Stef Graillat

LIP6/PEQUAN - Université Pierre et Marie Curie (Paris 6)

Groupe de Travail Arénaire
ENS Lyon, LIP, April 23rd, 2009

Floating point number

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

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

b : basis, p : precision, e : exponent range s.t. $e_{\min} \le e \le e_{\max}$

Machine epsilon $\epsilon = b^{1-p}$, $|1^+ - 1| = \epsilon$

Approximation of $\mathbb{R}$ by $\mathbb{F}$, rounding $\mathrm{fl} : \mathbb{R} \to \mathbb{F}$
Let $x \in \mathbb{R}$ then

$$\mathrm{fl}(x) = x(1+\delta), \quad |\delta| \le \mathbf{u}.$$

Unit roundoff $\mathbf{u} = \epsilon/2$ for round-to-nearest

Standard model of floating point arithmetic

Let $x, y \in \mathbb{F}$,

$$\mathrm{fl}(x \circ y) = (x \circ y)(1 + \delta), \quad |\delta| \le \mathbf{u}, \quad \circ \in \{+, -, \cdot, /\}$$

IEEE 754 standard (1985)

Type	Size	Mantissa	Exponent	Unit roundoff	Range
Double	64 bits	52+1 bits	11 bits	$\mathbf{u} = 2^{-53} \approx 1,11 \times 10^{-16}$	$\approx 10^{\pm 308}$

Aim of the talk

- Use Newton's method to accurately compute the simple roots of a polynomial.
- This needs to accurately calculate the residual (*i.e.* to accurately evaluate a polynomial)

Outline of the talk

1. Accurate polynomial evaluation

2. Accurate Newton's method

Outline of the talk

1. Accurate polynomial evaluation

2. Accurate Newton's method

What are Error-Free Transformations (EFT) ?

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

Error free transformations are properties and algorithms to compute the generated elementary rounding errors,

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

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
- compensated algorithms (Kahan, Priest, Ogita-Rump-Oishi, Graillat-Langlois-Louvet)

EFT for the summation

$$x = \mathrm{fl}(a \pm b) \quad \Rightarrow \quad a \pm b = x + y \quad \text{with } y \in \mathbb{F},$$

Algorithms of Dekker (1971) and Knuth (1974)

Algorithm 1 (EFT of the sum of 2 floating point numbers with $|a| \geq |b|$)

function $[x, y] = \texttt{FastTwoSum}(a, b)$
  $x = \mathrm{fl}(a + b)$
  $y = \mathrm{fl}((a - x) + b)$

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

function $[x, y] = \texttt{TwoSum}(a, b)$
  $x = \mathrm{fl}(a + b)$
  $z = \mathrm{fl}(x - a)$
  $y = \mathrm{fl}((a - (x - z)) + (b - z))$

EFT for the product (1/2)

$$x = \mathrm{fl}(a \cdot b) \quad \Rightarrow \quad a \cdot b = x + y \quad \text{with } y \in \mathbb{F},$$

Algorithm `TwoProduct` by Veltkamp and Dekker (1971)

$$a = x + y \quad \text{and} \quad x \text{ and } y \text{ non overlapping with } |y| \le |x|.$$

Algorithm 3 (Error-free split of a floating point number into two parts)

function $[x, y] = \mathtt{Split}(a)$
  factor $= \mathrm{fl}(2^s + 1)$ % $\mathbf{u} = 2^{-p}$, $s = \lceil p/2 \rceil$
  $c = \mathrm{fl}(\mathtt{factor} \cdot a)$
  $x = \mathrm{fl}(c - (c - a))$
  $y = \mathrm{fl}(a - x)$

EFT for the product (2/2)

Algorithm 4 (EFT of the product of 2 floating point numbers)

function $[x, y] = \texttt{TwoProduct}(a, b)$
$\quad x = \mathrm{fl}(a \cdot b)$
$\quad [a_1, a_2] = \texttt{Split}(a)$
$\quad [b_1, b_2] = \texttt{Split}(b)$
$\quad y = \mathrm{fl}(a_2 \cdot b_2 - (((x - a_1 \cdot b_1) - a_2 \cdot b_1) - a_1 \cdot b_2))$

EFT for the product (3/3)

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

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

Algorithm 5 (EFT of the product of 2 floating point numbers)

function $[x, y] = \texttt{TwoProductFMA}(a, b)$
  $x = \mathrm{fl}(a \cdot b)$
  $y = \texttt{FMA}(a, b, -x)$

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

Summary

Theorem 1

Let $a, b \in \mathbb{F}$ *and let* $x, y \in \mathbb{F}$ *such that* $[x, y] = \texttt{TwoSum}(a, b)$*. Then,*

$$a + b = x + y, \quad x = \mathrm{fl}(a + b), \quad |y| \leq \mathbf{u}|x|, \quad |y| \leq \mathbf{u}|a + b|.$$

The algorithm `TwoSum` *requires* 6 *flops.*

Let $a, b \in \mathbb{F}$ *and let* $x, y \in \mathbb{F}$ *such that* $[x, y] = \texttt{TwoProduct}(a, b)$ *. Then,*

$$a \cdot b = x + y, \quad x = \mathrm{fl}(a \cdot b), \quad |y| \leq \mathbf{u}|x|, \quad |y| \leq \mathbf{u}|a \cdot b|,$$

The algorithm `TwoProduct` *requires* 17 *flops.*

The Horner scheme

Algorithm 6 (Horner scheme)

function $\texttt{res} = \texttt{Horner}(p, x)$

$s_n = a_n$

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

$p_i = \mathrm{fl}(s_{i+1} \cdot x)$ % rounding error π_i

$s_i = \mathrm{fl}(p_i + a_i)$ % rounding error σ_i

end

$\texttt{res} = s_0$

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

$$\frac{|p(x) - \texttt{Horner}(p, x)|}{|p(x)|} \leq \underbrace{\gamma_{2n}}_{\approx 2n\mathbf{u}} \mathrm{cond}(p, x)$$

Error-free transformation for the Horner scheme

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

Algorithm 7 (Error-free transformation for the Horner scheme)

function $[\texttt{Horner}(p,x), p_\pi, p_\sigma] = \texttt{EFTHorner}(p,x)$
$s_n = a_n$
for $i = n-1 : -1 : 0$
  $[p_i, \pi_i] = \texttt{TwoProduct}(s_{i+1}, x)$
  $[s_i, \sigma_i] = \texttt{TwoSum}(p_i, a_i)$
  Let π_i be the coefficient of degree i of p_π
  Let σ_i be the coefficient of degree i of p_σ
end
$\texttt{Horner}(p,x) = s_0$

Compensated Horner scheme[1] and its accuracy

Algorithm 8 (Compensated Horner scheme)

function $\texttt{res} = \texttt{CompHorner}(p, x)$
$[h, p_\pi, p_\sigma] = \texttt{EFTHorner}(p, x)$
$c = \texttt{Horner}(p_\pi + p_\sigma, x)$
$\texttt{res} = \mathrm{fl}(h + c)$

Theorem 2

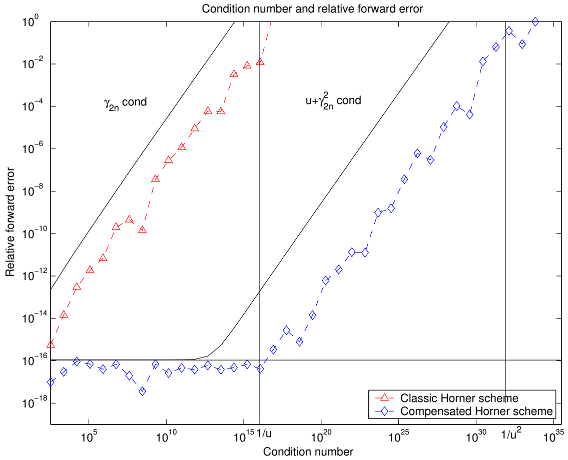
Let p be a polynomial of degree n with floating point coefficients, and x be a floating point value. Then if no underflow occurs,

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

[1] Compensated Horner Scheme, S.G., Philippe Langlois and Nicolas Louvet, Research Report RR2005-04, University of Perpignan, France, july 2005

Numerical experiments : testing the accuracy

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



Outline of the talk

1. Accurate polynomial evaluation
2. Accurate Newton's method

Condition number for root finding

Definition 1

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*

$$\mathrm{cond}(p, x) = \lim_{\varepsilon \to 0} \sup \left\{ \frac{|\Delta x|}{\varepsilon |x|} : |\Delta a_i| \leq \varepsilon |a_i| \right\}.$$

Theorem 3

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*

$$\mathrm{cond}(p, x) = \frac{\widetilde{p}(|x|)}{|x||p'(x)|},$$

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

Classic Newton's method

Algorithm 9 (Classic Newton's method)

$$x_0 = \xi$$
$$x_{i+1} = x_i - \frac{p(x_i)}{p'(x_i)}$$

$$\frac{|x_{i+1} - x|}{|x|} \approx \gamma_{2n} \operatorname{cond}(p, x)$$

Accurate Newton's method

Algorithm 10 (Accurate Newton's method)

$x_0 = \xi$

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

Using a theorem of F. Tisseur[2], one can show

Theorem 4

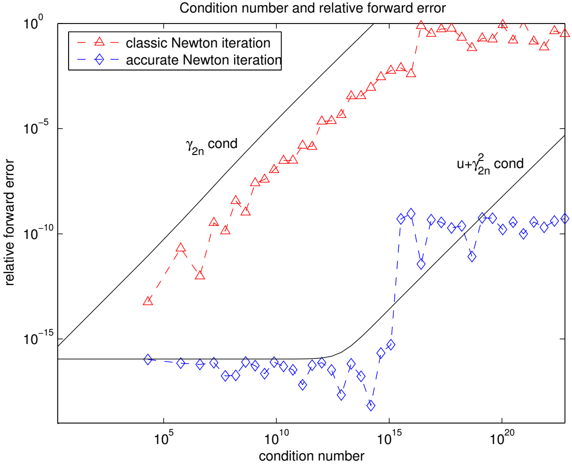
Assume that there is an x such that $p(x) = 0$ and $p'(x) \neq 0$ is not too small. Assume also that $\mathbf{u} \cdot \mathrm{cond}(p, x) \leq 1/8$ for all i.
Then, for all x_0 such that $\beta |p'(x)^{-1}| |x_0 - x| \leq 1/8$, Newton's method in floating point arithmetic generates a sequence of $\{x_i\}$ whose relative error decreases until the first i for which

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

[2] Newton's Method in Floating Point Arithmetic and Iterative Refinement of Generalized Eigenvalue Problems, *SIAM J. Matrix Anal. Appl.*, 22(4) : 1038-1057, 2001

Numerical experiments

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



Accuracy of the classic Newton iteration and of the accurate Newton iteration

What about multiple zeros?

Definition 2

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

$$\mathrm{cond}_m(p, x) = \lim_{\varepsilon \to 0} \sup \left\{ \frac{|\Delta x|}{\varepsilon^{1/m} |x|} : |\Delta a_i| \le \varepsilon |a_i| \right\}.$$

Theorem 5

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

$$\mathrm{cond}_m(p, x) = \frac{1}{|x|} \left(\frac{m!\, \widetilde{p}(|x|)}{|p^{(m)}(x)|} \right)^{1/m}.$$

What about multiple zeros ?

- If the root has multiplicity $m > 1$, one can use the modified Newton's iteration as follows.

Algorithm 11 (Modified Newton's method)

$x_0 = \xi$
$x_{i+1} = x_i - m\frac{p(x_i)}{p'(x_i)}$

- Using deflation : trying to find zeros of $p(x)/p'(x)$

We hope to achieve (in the classic case) the bound

$$\frac{|x_{i+1} - x|}{|x|} \approx \mathbf{u}^{1/m}\, \mathrm{cond}_m(p, x)$$

and for accurate case

$$\frac{|x_{i+1} - x|}{|x|} \approx \mathbf{u} + \mathbf{u}^{2/m}\, \mathrm{cond}_m(p, x)$$

Future work

The problem is to find the multiplicity : one can guess it using some kind of approximate gcd[3].

Work to be done :

- Deal with zeros with multiplicities *via* an accurate modified Newton's method
- Use of deflation to also deal with multiplicities

[3] Computing multiple roots of inexact polynomials, Z. Zeng, Mathematics of Computation, 74(2005), pp 869 - 903

Thank you for your attention