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 \( \mathbb{F} \subset \mathbb{R} \):

\[
x = \pm x_0 \cdot x_1 \ldots x_{p-1} \times b^e, \quad 0 \leq x_i \leq b - 1, \quad x_0 \neq 0
\]

\( b \) : basis, \( p \) : precision, \( e \) : exponent range s.t. \( e_{\text{min}} \leq e \leq e_{\text{max}} \)

Machine epsilon \( \epsilon = b^{1-p}, \quad |1^+ - 1| = \epsilon \)

Approximation of \( \mathbb{R} \) by \( \mathbb{F} \), rounding \( \text{fl} : \mathbb{R} \to \mathbb{F} \)

Let \( x \in \mathbb{R} \) then

\[
\text{fl}(x) = x(1 + \delta), \quad |\delta| \leq u.
\]

Unit roundoff \( u = \epsilon/2 \) for round-to-nearest
Standard model of floating point arithmetic

Let \( x, y \in \mathbb{F} \),

\[
\text{fl}(x \circ y) = (x \circ y)(1 + \delta), \quad |\delta| \leq u, \quad \circ \in \{+,-,\cdot,\}/
\]

IEEE 754 standard (1985)

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Mantissa</th>
<th>Exponent</th>
<th>Unit roundoff</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double</td>
<td>64 bits</td>
<td>52+1 bits</td>
<td>11 bits</td>
<td>( u = 2^{-53} \approx 1, 11 \times 10^{-16} )</td>
<td>( \approx 10^{\pm 308} )</td>
</tr>
</tbody>
</table>
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 $u$ (no underflow nor overflow)

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

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

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

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


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

```plaintext
function \([x, y] = \text{FastTwoSum}(a, b)\)
   x = \text{fl}(a + b)
   y = \text{fl}((a - x) + b)
```

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

```plaintext
function \([x, y] = \text{TwoSum}(a, b)\)
   x = \text{fl}(a + b)
   z = \text{fl}(x - a)
   y = \text{fl}((a - (x - z)) + (b - z))
```
EFT for the product (1/2)

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

Algorithm TwoProduct by Veltkamp and Dekker (1971)

\[ a = x + y \quad \text{and} \quad x \text{ and } y \text{ non overlapping with } |y| \leq |x|. \]

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

function \([x, y] = \text{Split}(a)\)

\[
\begin{align*}
\text{factor} &= \text{fl}(2^s + 1) \\
\text{c} &= \text{fl}(\text{factor} \cdot a) \\
x &= \text{fl}(\text{c} - (\text{c} - a)) \\
y &= \text{fl}(a - x)
\end{align*}
\]

\% \ u = 2^{-p} , \ s = \lceil p/2 \rceil
Algorithm 4 (EFT of the product of 2 floating point numbers)

function \([x, y] = \text{TwoProduct}(a, b)\)
\[
x = \text{fl}(a \cdot b)
\]
\[
[a_1, a_2] = \text{Split}(a)
\]
\[
[b_1, b_2] = \text{Split}(b)
\]
\[
y = \text{fl}(a_2 \cdot b_2 - (((x - a_1 \cdot b_1) - a_2 \cdot b_1) - a_1 \cdot b_2))
\]
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 5 (EFT of the product of 2 floating point numbers)

```plaintext
function \([x, y] = \text{TwoProductFMA}(a, b)\)
\[
x = \text{fl}(a \cdot b)
\]
\[
y = \text{FMA}(a, b, -x)
\]
```

The FMA is available for example on PowerPC, Itanium, Cell processors.
Theorem 1

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

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

The algorithm TwoSum requires 6 flops.

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

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

The algorithm TwoProduct requires 17 flops.
The Horner scheme

Algorithm 6 (Horner scheme)

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

\[
\begin{align*}
  s_n &= a_n \\
  \text{for } i &= n - 1 : -1 : 0 \\
  p_i &= \text{fl}(s_{i+1} \cdot x) \quad \% \text{ rounding error } \pi_i \\
  s_i &= \text{fl}(p_i + a_i) \quad \% \text{ rounding error } \sigma_i \\
  \text{end} \\
  \text{res} &= s_0
\end{align*}
\]

\[\gamma_n = nu/(1 - nu) \approx nu\]

\[
\frac{|p(x) - \text{Horner}(p, x)|}{|p(x)|} \leq \gamma_{2n} \cdot \text{cond}(p, x) \approx 2nu
\]
Algorithm 7 (Error-free transformation for the Horner scheme)

function \[\text{Horner}(p, x), p_{\pi}, p_{\sigma}\] = EFTHorner(p, x)

\[s_n = a_n\]

for \(i = n - 1 : -1 : 0\)

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

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

Let \(\pi_i\) be the coefficient of degree \(i\) of \(p_{\pi}\)

Let \(\sigma_i\) be the coefficient of degree \(i\) of \(p_{\sigma}\)

end

\[\text{Horner}(p, x) = s_0\]
Compensated Horner scheme\(^1\) and its accuracy

### Algorithm 8 (Compensated Horner scheme)

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

\[
\left| \frac{\text{CompHorner}(p, x) - p(x)}{|p(x)|} \right| \leq u + \gamma_{2n}^2 \text{cond}(p, x).
\]

\[\approx 4n^2u^2\]

\(^1\)Compensated Horner Scheme, S.G., Philippe Langlois and Nicolas Louvet, Research Report RR2005-04, University of Perpignan, France, july 2005
Evaluation of $p_n(x) = (x - 1)^n$ for $x = \text{fl}(1.333)$ and $n = 3, \ldots, 42$
Outline of the talk

1. Accurate polynomial evaluation
2. Accurate Newton’s method
**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

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

**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

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

with $\tilde{p}(x) = \sum_{i=0}^{n} |a_i| z^i$. 
Algorithm 9 (Classic Newton’s method)

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

\[ \frac{|x_{i+1} - x|}{|x|} \approx \gamma_{2n} \text{cond}(p, x) \]
Accurate Newton’s method

Algorithm 10 (Accurate Newton’s method)

\[
x_0 = \xi \\
x_{i+1} = x_i - \frac{\text{CompHorner}(p, x_i)}{p'(x_i)}
\]

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 \( u \cdot \text{cond}(p, x) \leq 1/8 \) 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

\[
\frac{|x_{i+1} - x|}{|x|} \approx u + \gamma^2_{2n} \text{cond} (p, x).
\]


S. Graillat (Univ. Paris 6)
Numerical experiments

Test with \( p_n(x) = (x - 1)^n - 10^{-8} \) and \( x = 1 + 10^{-8}/n \) for \( n = 1 : 40 \)

\[ \text{cond}(p_n, x) \text{ varies from } 10^4 \text{ to } 10^{22} \]

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

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

**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

\[
\text{cond}_m(p, x) = \frac{1}{|x|} \left( \frac{m! \tilde{p}(|x|)}{|p^{(m)}(x)|} \right)^{1/m}.
\]
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)**

\[
\begin{align*}
x_0 &= \xi \\
x_{i+1} &= x_i - m \frac{p(x_i)}{p'(x_i)}
\end{align*}
\]

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

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

\[
\frac{|x_{i+1} - x|}{|x|} \approx u^{1/m} \text{cond}_m(p, x)
\]

and for accurate case

\[
\frac{|x_{i+1} - x|}{|x|} \approx u + u^{2/m} \text{cond}_m(p, x)
\]
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