SWIM 08, Montpellier, France, June 19 – 20, 2008

Compensated algorithms and validated error bounds in floating point computation

Philippe Langlois
DALI, Université de Perpignan

Nicolas Louvet
Arénaire, ENS Lyon, INRIA

with contributions by

Claude-Pierre Jeannerod and **Guillaume Revy**
Arénaire, LIP, ENS Lyon, INRIA.

Acknowledgment

This work is partly funded by the ANR EVA-Flo Project, ANR-BLAN06-2-135670 2006.

Why this talk here today?

Interval arithmetic is an excellent tool to validated computed solutions since

- it controls both data uncertainties and finite precision computing errors,
- it provides a guaranteed localisation of solution sets,
- it can be applied to many problems with a reasonable difficulty
- but it sometimes suffers from too expensive running-time.

Why this talk here today?

Interval arithmetic is an excellent tool to validated computed solutions since

- it controls both data uncertainties and finite precision computing errors,
- it provides a guaranteed localisation of solution sets,
- it can be applied to many problems with a reasonable difficulty
- but it sometimes suffers from too expensive running-time.

Validated error bounds and compensated algorithms

- only focus finite precision computing errors,
- provide validated but only pointwise solution,
- are devoted to few algorithms (at least for the moment),
- but are actually fast.

Why this talk here today?

Interval arithmetic is an excellent tool to validated computed solutions since

- it controls both data uncertainties and finite precision computing errors,
- it provides a guaranteed localisation of solution sets,
- it can be applied to many problems with a reasonable difficulty
- but it sometimes suffers from too expensive running-time.

Validated error bounds and compensated algorithms

- only focus finite precision computing errors,
- provide validated but only pointwise solution,
- are devoted to few algorithms (at least for the moment),
- but are actually fast.

So the question I ask to experts in interval arithmetic is:

Are there parts of interval solving scenaris that can be speed-up by validated and compensated algorithms?

Scope and general motivation

Scope: finite precision computation

- IEEE-754 floating point arithmetic, rounding to the nearest, no overflow.
- Data uncertainty is not considered: floating point entries.

Scope and general motivation

Scope: finite precision computation

- IEEE-754 floating point arithmetic, rounding to the nearest, no overflow.
- Data uncertainty is not considered: floating point entries.

Motivations:

1. How to estimate the accuracy of a finite precision computation?
   - *A priori* error analysis — Wilkinson (1963) and sons
   - *A posteriori* or dynamic error analysis — Wilkinson (1971) and sons
2. How to compute validated error bounds?
   - Interval arithmetic — Moore-Tsunaga ($\approx$ 1960) and sons
   - **Validated error bounds** — Rump (2005) and sons
3. How to improve and validate the accuracy of the computed result?
   - **Compensated algorithms** — Kahan-Babuska (1965) and sons
   - ... with validated error bounds.

Menu du jour

1. Error bounds. . . illustrated with the Horner algorithm
2. Compensated algorithms to double (at least) the accuracy
3. A validated error bound to control the actual accuracy
4. Performance issues exhibit challenging overheads for running time
5. Conclusion

How to perform these rounding error analysis?

- Standard model of floating point computation:
  Let $a, b \in \mathbb{F}$, $\circ \in \{+, -, \times, /\}$ and $\mathrm{fl}(x \circ y)$ be the exact $x \circ y$ rounded to the nearest floating point value at precision u.

$$\mathrm{fl}(a \circ b) = (1 + \varepsilon_1)(a \circ b) = (a \circ b)/(1 + \varepsilon_2), \quad \text{with} \quad |\varepsilon_1|, |\varepsilon_2| \leq u.$$

How to perform these rounding error analysis?

- Standard model of floating point computation:
  Let $a, b \in \mathbb{F}$, $\circ \in \{+, -, \times, /\}$ and $\mathrm{fl}(x \circ y)$ be the exact $x \circ y$ rounded to the nearest floating point value at precision u.

$$\mathrm{fl}(a \circ b) = (1 + \varepsilon_1)(a \circ b) = (a \circ b)/(1 + \varepsilon_2), \quad \text{with} \quad |\varepsilon_1|, |\varepsilon_2| \leq u.$$

- From one local rounding error to a global error bound (while $nu < 1$):

$$\prod_{i=1}^{n} (1 + \varepsilon_i)^{\pm 1} = 1 + \theta_n, \text{ with } |\theta_n| \leq \gamma_n := \frac{nu}{1 - nu} \approx nu.$$

How to perform these rounding error analysis?

- Standard model of floating point computation:
  Let $a, b \in \mathbb{F}$, $\circ \in \{+, -, \times, /\}$ and $\mathrm{fl}(x \circ y)$ be the exact $x \circ y$ rounded to the nearest floating point value at precision u.

$$\mathrm{fl}(a \circ b) = (1 + \varepsilon_1)(a \circ b) = (a \circ b)/(1 + \varepsilon_2), \quad \text{with} \quad |\varepsilon_1|, |\varepsilon_2| \leq u.$$

- From one local rounding error to a global error bound (while $nu < 1$):

$$\prod_{i=1}^{n} (1 + \varepsilon_i)^{\pm 1} = 1 + \theta_n, \text{ with } |\theta_n| \leq \gamma_n := \frac{nu}{1 - nu} \approx nu.$$

- From an *a priori* bound to a computable bound (while $nu < 1$):
  For $\widehat{\gamma}_n = (n \times u) \oslash (1 \ominus n \times u)$, we have

$$\gamma_n \leq (1 + u)\,\widehat{\gamma}_n \leq (1 - u)\,\widehat{\gamma}_n,$$

$$(1 + u)^n |x| \leq |x| \oslash (1 \ominus (n \oplus 1) \otimes u).$$

Mise en bouche

1. Error bounds. . . illustrated with the Horner algorithm
   - A priori and not validated analysis
   - Towards validated *a priori* or dynamic error bounds
2. Compensated algorithms to double (at least) the accuracy
   - Introducing example and error-free transformations (EFT)
   - Compensated Horner algorithm
3. A validated error bound to control the actual accuracy
   - Validated dynamic bound for CompHorner
   - Application to a faithfully rounded polynomial evaluation
4. Performance issues exhibit challenging overheads for running time
   - Overhead to more accuracy
   - Overhead for a validated and more accurate result
5. Conclusion

Relative accuracy of the Horner algorithm

We consider the polynomial

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

with $a_i \in \mathbb{F}$, $x \in \mathbb{F}$

Algorithm

function $r_0 = \text{Horner}\,(p, x)$

$r_n = a_n$

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

$\quad r_i = r_{i+1} \otimes x \oplus a_i$

end

Relative accuracy of the Horner algorithm

We consider the polynomial

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

with $a_i \in \mathbb{F}$, $x \in \mathbb{F}$

Algorithm

function $r_0 = \text{Horner}\,(p, x)$
$r_n = a_n$
for $i = n-1 : -1 : 0$
$\quad r_i = r_{i+1} \otimes x \oplus a_i$
end

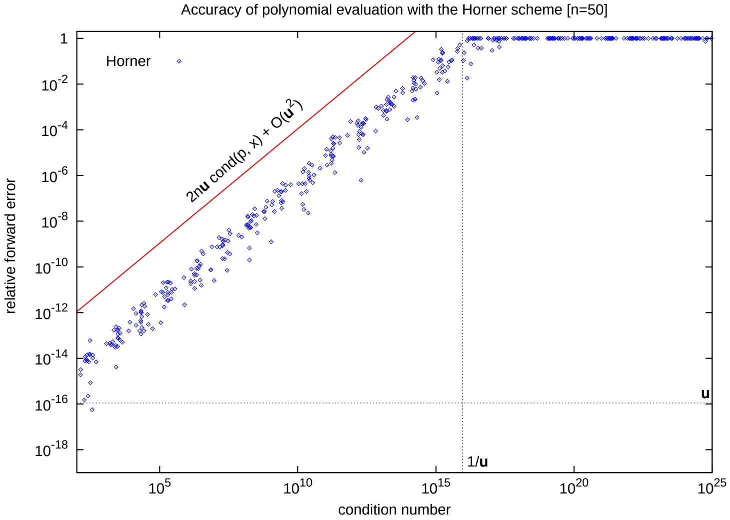
A priori bound of the relative accuracy of the Horner algorithm:

$$\frac{|\text{Horner}(p,x) - p(x)|}{|p(x)|} \leq \underbrace{\gamma_{2n}}_{\approx 2nu} \text{cond}(p,x).$$

$\text{cond}(p,x)$ denotes the condition number of the evaluation:

$$\text{cond}(p,x) = \frac{\sum |a_i x^i|}{|p(x)|} \geq 1.$$

Accuracy $\lesssim$ condition number of the problem $\times$ u



Bounds for the absolute error in Horner algorithm

Find a bound B or a computable $\widehat{B}$ such that $|\text{Horner}(p, x) - p(x)| \leq B$.

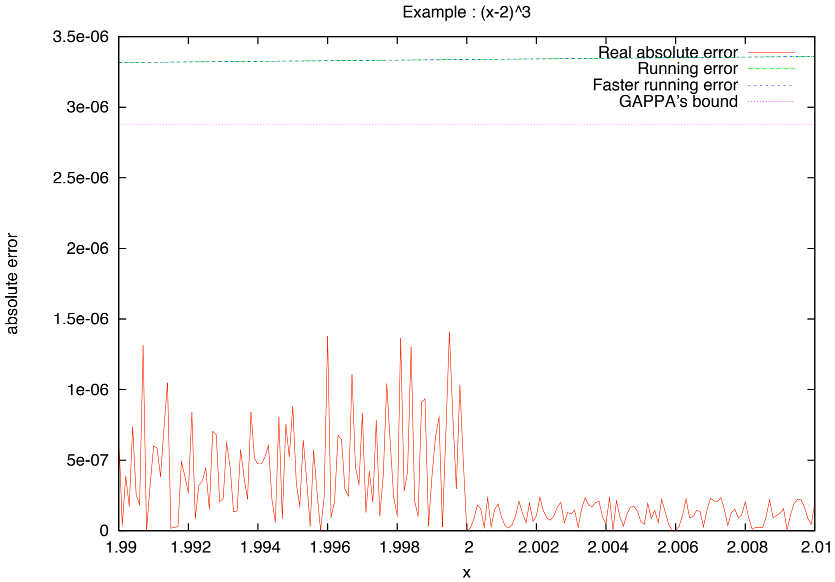
- Classic bounds are pessimistic but not validated ($b < B$)
  - *A priori* bound: $b_{AP} = \gamma_{2n} \sum |a_i x^i|$
  - Wilkinson's running error bound (N.J. Higham, ASNA, p.95)
    $\widehat{b}_{REA} = uE_0$, from $E_i = \left(E_{i+1} + |\widehat{r}_{i+1}|\right)|x| + |\widehat{r}_i|, (i = n-1 : 0)$ and $E_n = 0$.

Bounds for the absolute error in Horner algorithm

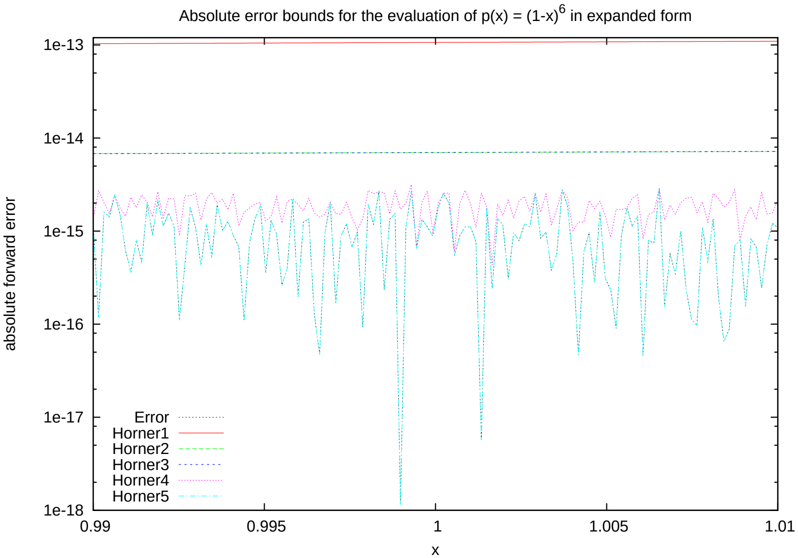
Find a bound B or a computable $\widehat{B}$ such that $|\text{Horner}(p, x) - p(x)| \leq B$.

- Classic bounds are pessimistic but not validated ($b < B$)
  - *A priori* bound: $b_{AP} = \gamma_{2n} \sum |a_i x^i|$
  - Wilkinson's running error bound (N.J. Higham, ASNA, p.95)
    $\widehat{b}_{REA} = uE_0$, from $E_i = \big(E_{i+1} + |\widehat{r}_{i+1}|\big)|x| + |\widehat{r}_i|, (i = n-1:0)$ and $E_n = 0$.
- Recent validated bounds when no underflow occurs
  - A validated *a priori* error bound: $\widehat{B}_{AP} = \text{fl}\left(\frac{\gamma_{2n}\text{Horner}(|p|,|x|)}{1-(2n+3)u}\right)$
  - A validated running error bound: $\widehat{B}_{REA} = \text{fl}\left(\frac{u}{1-(3n+1)u}\text{fl}(E_0)\right)$
- Other bounds (tighter, faster) and flop counts in a joint work in progress of Cl.-P. Jeannerod, Ph. L., N. Louvet and G.Revy.

Validated error bounds for Horner algorithm



Validated error bounds for Horner algorithm



Horner1: *a priori* Horner2 and Horner3: REA Horner4 and Horner5: EFT

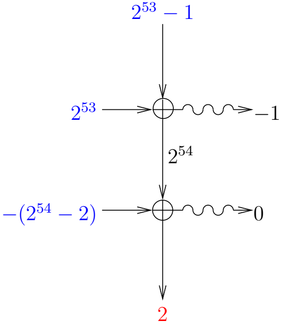
Entrées

Example: compensated summation

IEEE double precision numbers: $x_1 = 2^{53} - 1$, $x_2 = 2^{53}$ and $x_3 = -(2^{54} - 2)$.

Exact sum: $x_1 + x_2 + x_3 = 1$.

Classic summation



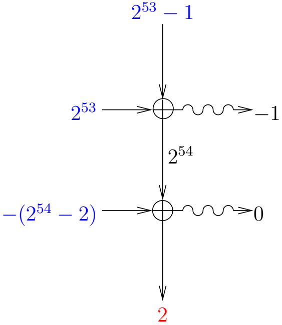
Relative error = 1

Example: compensated summation

IEEE double precision numbers: $x_1 = 2^{53} - 1$, $x_2 = 2^{53}$ and $x_3 = -(2^{54} - 2)$.

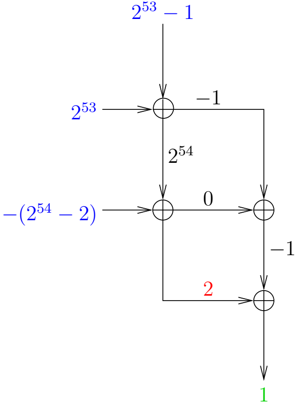
Exact sum: $x_1 + x_2 + x_3 = 1$.

Classic summation



Relative error = 1

Compensation of the rounding errors



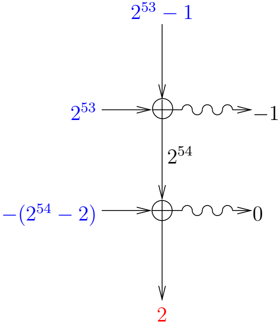
The exact result is computed

Example: compensated summation

IEEE double precision numbers: $x_1 = 2^{53} - 1$, $x_2 = 2^{53}$ and $x_3 = -(2^{54} - 2)$.

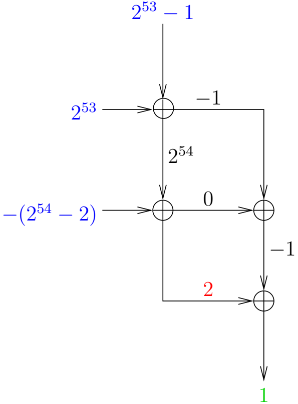
Exact sum: $x_1 + x_2 + x_3 = 1$.

Classic summation



Relative error = 1

Compensation of the rounding errors



The exact result is computed

The rounding errors are computed thanks to *error-free transformations*.

Error-free transformations (EFT)

Error-Free Transformations are algorithms to compute the rounding errors at the current working precision.

$+$	$(x, y) = 2\mathrm{Sum}(a, b)$ such that $x = a \oplus b$ and $a + b = x + y$	6 flop	Knuth (74)
$\times$	$(x, y) = 2\mathrm{Prod}(a, b)$ such that $x = a \otimes b$ and $a \times b = x + y$	17 flop	Dekker (71)

with $a, b, x, y \in \mathbb{F}$.

Algorithm (Knuth)

function [x,y] = 2Sum(a,b)

$x = a \oplus b$

$z = x \ominus a$

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

Error-free transformations (EFT)

Error-Free Transformations are algorithms to compute the rounding errors at the current working precision.

$+$	$(x, y) = \mathrm{2Sum}(a, b)$ such that $x = a \oplus b$ and $a + b = x + y$	6 flop	Knuth (74)
$\times$	$(x, y) = \mathrm{2Prod}(a, b)$ such that $x = a \otimes b$ and $a \times b = x + y$	17 flop	Dekker (71)

with $a, b, x, y \in \mathbb{F}$.

Algorithm (Knuth)

function [x,y] = 2Sum(a,b)

$x = a \oplus b$

$z = x \ominus a$

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

Compensated summation algorithms:

- Kahan, Møller (1965),
- Pichat (1972),
- Neumaier (1974),
- Priest (1992),
- Ogita-Rump-Oishi (2005).

EFT for the Horner algorithm

Consider $p(x) = \sum_{i=0}^{n} a_i x^i$ of degree n, $a_i, x \in \mathbb{F}$.

Algorithm (Horner)

function $r_0 = \text{Horner}\,(p, x)$

$r_n = a_n$

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

$p_i = r_{i+1} \otimes x$ % error $\pi_i \in \mathbb{F}$

$r_i = p_i \oplus a_i$ % error $\sigma_i \in \mathbb{F}$

end

Let us define two polynomials p_π and p_σ such that:

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

EFT for the Horner algorithm

Consider $p(x) = \sum_{i=0}^{n} a_i x^i$ of degree n, $a_i, x \in \mathbb{F}$.

Algorithm (Horner)

function $r_0 = \text{Horner}\,(p, x)$
  $r_n = a_n$
  for $i = n-1 : -1 : 0$
    $p_i = r_{i+1} \otimes x$ % error $\pi_i \in \mathbb{F}$
    $r_i = p_i \oplus a_i$ % error $\sigma_i \in \mathbb{F}$
  end

Let us define two polynomials p_π and p_σ such that:

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

Theorem (EFT for Horner algorithm)

$$\underbrace{p(x)}_{\textit{exact value}} = \underbrace{\text{Horner}\,(p, x)}_{\in \mathbb{F}} + \underbrace{(p_\pi + p_\sigma)(x)}_{\textit{forward error}}.$$

EFT for the Horner algorithm

Consider $p(x) = \sum_{i=0}^{n} a_i x^i$ of degree n, $a_i, x \in \mathbb{F}$.

Algorithm (Horner)

function $r_0 = \text{Horner}\,(p, x)$

$\quad r_n = a_n$

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

$\quad\quad p_i = r_{i+1} \otimes x \qquad$ % error $\pi_i \in \mathbb{F}$

$\quad\quad r_i = p_i \oplus a_i \qquad$ % error $\sigma_i \in \mathbb{F}$

$\quad$ end

Algorithm (EFT for Horner)

function $[r_0, p_\pi, p_\sigma] = \text{EFTHorner}\,(p, x)$

$\quad r_n = a_n$

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

$\quad\quad [p_i, \pi_i] = \text{2Prod}\,(r_{i+1}, x)$

$\quad\quad [r_i, \sigma_i] = \text{2Sum}\,(p_i, a_i)$

$\quad\quad p_\pi[i] = \pi_i \qquad p_\sigma[i] = \sigma_i$

$\quad$ end

Theorem (EFT for Horner algorithm)

$$\underbrace{p(x)}_{\textit{exact value}} = \underbrace{\text{Horner}\,(p, x)}_{\in \mathbb{F}} + \underbrace{(p_\pi + p_\sigma)(x)}_{\textit{forward error}}.$$

Compensated Horner algorithm

$(p_\pi + p_\sigma)(x)$ is exactly the forward error affecting Horner (p, x).

$\Rightarrow$ we compute an approximate of $(p_\pi + p_\sigma)(x)$ as a correcting term.

Algorithm (Compensated Horner algorithm)

function $\bar{r} =$ CompHorner (p, x)

$[\widehat{r}, p_\pi, p_\sigma] =$ EFTHorner (p, x) % $\widehat{r} =$ Horner (p, x)

$\widehat{c} =$ Horner $(p_\pi \oplus p_\sigma, x)$

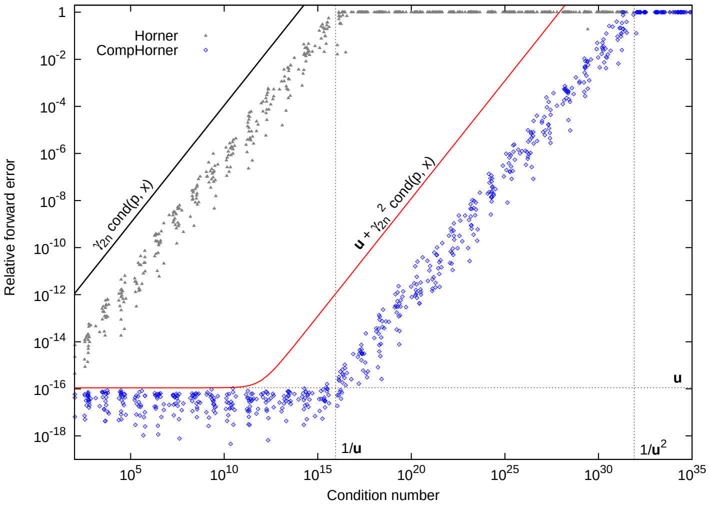
$\bar{r} = \widehat{r} \oplus \widehat{c}$

Theorem

Given p a polynomial with floating point coefficients, and $x \in \mathbb{F}$,

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

Accuracy of the result $\lesssim u +$ condition number $\times\ u^2$.



The compensated Horner algorithm is as accurate as the classic Horner algorithm performed in twice the working precision, with a final rounding.

Compensated algorithms

- Algorithms that correct the generated rounding errors.
- The rounding errors are computed at the current working precision thanks to error-free transformations.
- Compensation applies to summation, dot product, polynomial evaluation, triangular linear system,
- Existing examples: Kahan's compensated summation (65), Priest's doubly compensated summation (92), Ogita-Rump-Oishi (SISC 05), Langlois-Louvet (Arith 2007) ...
- Compensated algorithms run faster than challenger algorithms (to be presented later)
- More accuracy is available (EFT implies recursivity) and is still running fast up to ≈ 200 bits of precision

Plat

1. Error bounds. . . illustrated with the Horner algorithm
   - A priori and not validated analysis
   - Towards validated *a priori* or dynamic error bounds
2. Compensated algorithms to double (at least) the accuracy
   - Introducing example and error-free transformations (EFT)
   - Compensated Horner algorithm
3. A validated error bound to control the actual accuracy
   - Validated dynamic bound for CompHorner
   - Application to a faithfully rounded polynomial evaluation
4. Performance issues exhibit challenging overheads for running time
   - Overhead to more accuracy
   - Overhead for a validated and more accurate result
5. Conclusion

More accuracy without validated bound is useless

Consider a polynomial p of degree n with floating point coefficients, and $x \in \mathbb{F}$.

Algorithm

function $\overline{r} = \text{CompHorner}(p, x)$
$\quad [\widehat{r}, p_\pi, p_\sigma] = \text{EFTHorner}(p, x)$
$\quad \widehat{c} = \text{Horner}(p_\pi \oplus p_\sigma, x)$
$\quad \overline{r} = \widehat{r} \oplus \widehat{c}$

A priori error bound for the compensated evaluation:

$$|\text{CompHorner}(p,x) - p(x)| \le u|p(x)| + \underbrace{\gamma_{2n}^2}_{\approx (2nu)^2} \widetilde{p}(x).$$

Problem: This *a priori* error bound

- can not be computed at running time, as $|p(x)|$ is "unknown";
- is pessimistic compared to the actual error.

A dynamic and validated version of CompHorner

Consider a polynomial p of degree n with floating point coefficients, and $x \in \mathbb{F}$.

Algorithm

function $\bar{r} = \text{CompHorner}(p, x)$
$[\widehat{r}, p_\pi, p_\sigma] = \text{EFTHorner}(p, x)$
$\widehat{c} = \text{Horner}(p_\pi \oplus p_\sigma, x)$
$\bar{r} = \widehat{r} \oplus \widehat{c}$ % Rounding error $\delta = \widehat{r} + \widehat{c} - \bar{r} \in \mathbb{F}$.

Since EFTHorner is as error-free transformation, we have:

$$\underbrace{|\text{CompHorner}(p,x) - p(x)|}_{\text{error in the compensated result}} \leq |\delta| + \underbrace{|\widehat{c} - c|}_{\text{error in the correcting term}}.$$

A dynamic and validated version of CompHorner

Consider a polynomial p of degree n with floating point coefficients, and $x \in \mathbb{F}$.

Algorithm

function $[\overline{r}, \beta] = \text{CompHornerBound}(p, x)$

if $2(n+1)u \geq 1$, error('Validation impossible'), end

$[\widehat{r}, p_\pi, p_\sigma] = \text{EFTHorner}(p, x)$

$\widehat{c} = \text{Horner}(p_\pi \oplus p_\sigma, x)$

$[\overline{r}, \delta] = \text{2Sum}(\widehat{r}, \widehat{c})$ % Exact computation of δ

$\alpha = \left(\widehat{\gamma}_{2n-1} \otimes \text{Horner}(|p_\pi \oplus p_\sigma|, |x|)\right) \oslash (1 - 2(n+1)u)$

$\beta = (|\delta| \oplus \alpha) \oslash (1 - 2u)$

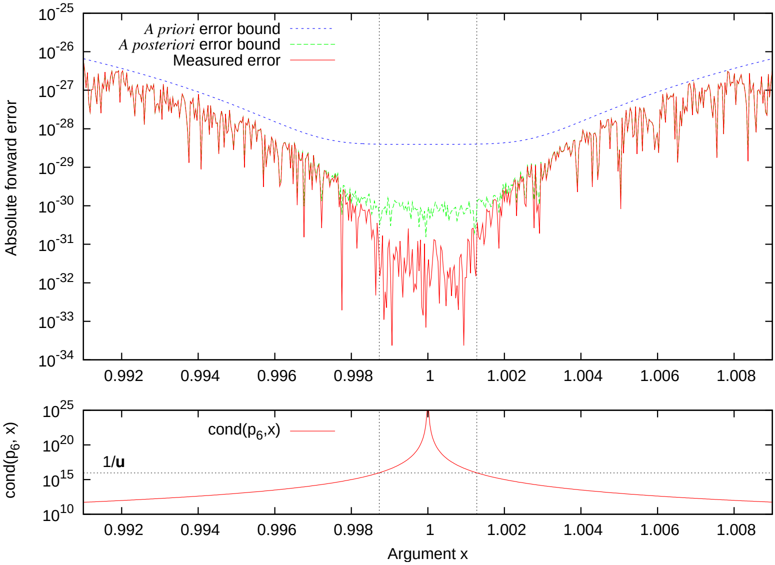
Theorem

Together with the compensated evaluation, CompHornerBound(p, x) *computes an a posteriori error bound* β *s.t.*

$$|\text{CompHorner}(p, x) - p(x)| \leq \beta.$$

Sharpness of the *a posteriori* error bound

Evaluation of $p_6(x) = (1-x)^6$ in expanded form in the neighborhood of $x = 1$.

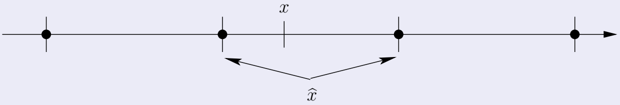


Towards a faithfully rounded polynomial evaluation

Definition

A floating point number $\widehat{x}$ is said to be a faithful rounding of a real number x if

- either $\widehat{x} = x$,
- or $\widehat{x}$ is one of the two floating point neighbours of x.



The worst case accuracy bound for CompHorner,

$$\frac{|\mathrm{CompHorner}\,(p,x) - p(x)|}{|p(x)|} \leq \underbrace{u + (2nu)^2 \mathrm{cond}(p,x) + \mathcal{O}(u^3)}_{>u}$$

is too large for reasoning about faithful rounding.

A sufficient condition for faithful rounding

We recall:

- $|\widehat{c} - c|$ is the error in the computed correcting term $\widehat{c} \in \mathbb{F}$
- $\overline{r} = \text{CompHorner}\,(p, x)$ is the compensated result.

Lemma

$$|\widehat{c} - c| < \frac{u}{2}|\overline{r}| \Rightarrow \overline{r} \textit{ is a faithful rounding of } p(x).$$

(see Lemma 2.5 in *Accurate floating point summation*, Rump, Ogita and Oishi, 2005)

Using this lemma, we present two results:

- an *a priori* upper bound on $\text{cond}(p, x)$ to ensure faithful rounding,
- an *a posteriori* (running time) test for faithful rounding.

An *a posteriori* test for faithful rounding

- A bound on the error $|\widehat{c} - c|$ in the computed correcting term $\widehat{c}$:

$$|c - \widehat{c}| \leq \mathrm{fl}\left(\frac{\widehat{\gamma}_{2n-1}\mathrm{Horner}\left(|p_\pi \oplus p_\sigma|, |x|\right)}{1 - 2(n+1)u}\right) =: \beta$$

  Bound satisfied when computed at running time in fp arithmetic.

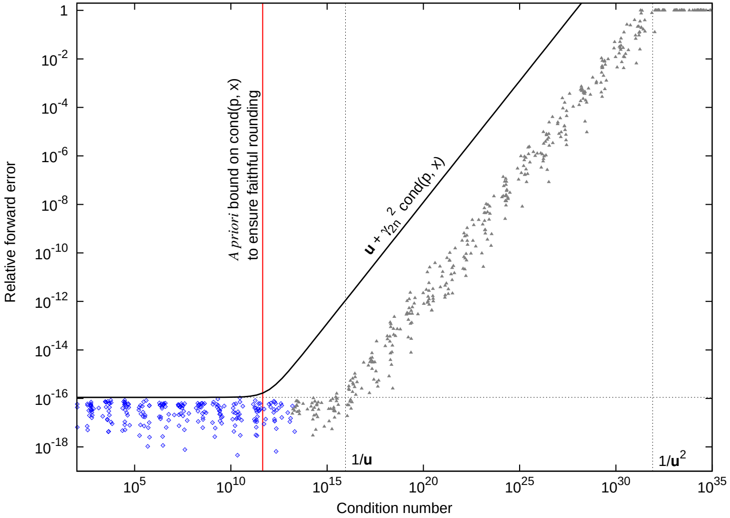
- Then,

$$\begin{aligned} \beta < \frac{u}{2}|\,\overline{r}| \quad &\Rightarrow \quad |c - \widehat{c}| < \frac{u}{2}|\,\overline{r}| \\ &\Rightarrow \quad \overline{r} \text{ is a faithful rounding of } p(x). \end{aligned}$$

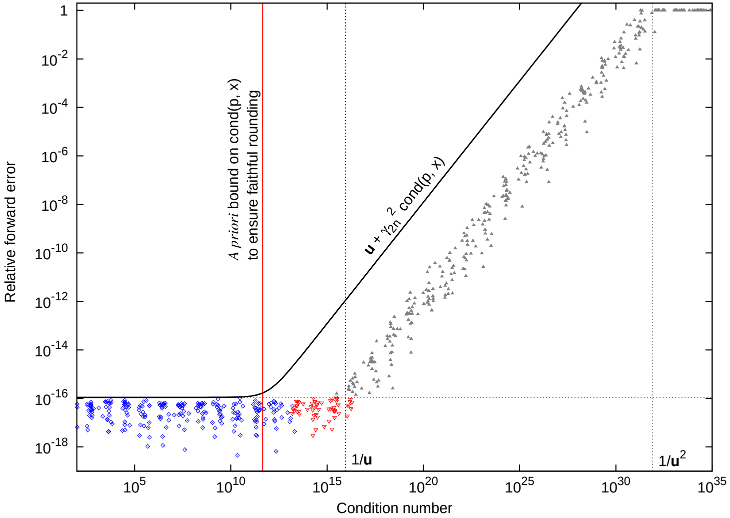
  This is again a sufficient condition :

  - if this test is satisfied, this ensure faithful rounding,
  - else, the compensated may be faithfully rounded or not.

An *a posteriori* condition (and an *a priori* one)



An *a posteriori* condition (and an *a priori* one)



Dessert

1. Error bounds... illustrated with the Horner algorithm
   - A priori and not validated analysis
   - Towards validated *a priori* or dynamic error bounds
2. Compensated algorithms to double (at least) the accuracy
   - Introducing example and error-free transformations (EFT)
   - Compensated Horner algorithm
3. A validated error bound to control the actual accuracy
   - Validated dynamic bound for CompHorner
   - Application to a faithfully rounded polynomial evaluation
4. Performance issues exhibit challenging overheads for running time
   - Overhead to more accuracy
   - Overhead for a validated and more accurate result
5. Conclusion

Overhead to double the precision

- We compare:
  - CompHorner = Compensated Horner algorithm
  - DDHorner = Horner algorithm + double-double (Bailey's library)

  Both provide the same output accuracy.
- Practical overheads compared to the classic Horner algorithm[1]:

		$\frac{\text{CompHorner}}{\text{Horner}}$	$\frac{\text{DDHorner}}{\text{Horner}}$	$\frac{\text{DDHorner}}{\text{CompHorner}}$
Pentium 4, 3.00 GHz	GCC 4.1.1	2.8	8.6	3.0
(x87 fp unit)	ICC 9.1	2.7	9.0	3.4
Athlon 64, 2.00 GHz	GCC 4.1.2	3.2	8.7	2.7
Itanium 2, 1.4 GHz	GCC 4.1.1	2.8	6.7	2.4
	ICC 9.1	1.5	5.9	3.9
		2 – 4	**6 – 9**	**2 – 4**

CompHorner runs a least two times faster than DDHorner.

[1]Average ratios for polynomials of degree 5 to 200; wp = IEEE-754 double precision

Overhead for a validated and more accurate result

Practical overheads compared to the classic Horner algorithm[1]:

		$\frac{\text{CompHorner}}{\text{Horner}}$	$\frac{\text{DDHorner}}{\text{Horner}}$	$\frac{\text{CompHornerIsFaith}}{\text{Horner}}$
Pentium 4, 3.00 GHz	GCC 4.1.1	3.42	10.6	4.41
(sse fp unit)	ICC 9.1	3.09	9.35	3.96
Athlon 64, 2.00 GHz	GCC 4.1.2	3.96	10.4	4.35
Itanium 2, 1.4 GHz	GCC 4.1.1	3.30	8.20	4.05
	ICC 9.1	1.93	9.68	2.26
		$\sim \mathbf{2-4}$	$\sim \mathbf{8-10}$	$\sim \mathbf{2-5}$

- CompHorner = Compensated Horner algorithm
- DDHorner = Horner algorithm + double-double (Bailey's library)
- CompHornerIsFaith = CompHorner + test for faithful rounding.

CompHorner runs a least two times faster than DDHorner.

[1] Average ratios for polynomials of degree 5 to 200.

Et pour finir

- Compensated Horner algorithm
  - as accurate as the Horner scheme performed in doubled working precision,
  - very efficient compared to the double-double alternative.
  - error bound computed using basic fp arithmetic, in RTN rounding mode;
  - underflow is considered in Louvet's PhD;
  - runs at most 1.5 times slower than the non validated algorithm.
- Other compensated algorithms and associated validated bounds
  - for summation, dot product, triangular linear system solution,
  - intrinsic excellent running time performances on superscalar machines (ILP)

Are there parts of interval solving scenaris that can be speed-up by validated and compensated algorithms?