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.

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 (≈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.
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
   - *A posteriori* or dynamic error analysis

2. How to compute validated error bounds?
   - Interval arithmetic
   - Validated error bounds

3. How to improve and validate the accuracy of the computed result?
   - Compensated algorithms
   - ... 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 $\text{fl}(x \circ y)$ be the exact $x \circ y$ rounded to the nearest floating point value at precision $u$.

  $$\text{fl}(a \circ b) = (1 + \varepsilon_1)(a \circ b) = (a \circ b)/(1 + \varepsilon_2), \quad \text{with} \quad |\varepsilon_1|, |\varepsilon_2| \leq u.$$
How to perform these rounding error analysis?

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

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

- **From one local rounding error to a global error bound (while \( n u < 1 \)):**
  \[
  \prod_{i=1}^{n} (1 + \varepsilon_i)^{\pm 1} = 1 + \theta_n, \quad \text{with} \quad |\theta_n| \leq \gamma_n := \frac{n u}{1 - n u} \approx n u.
  \]
How to perform these rounding error analysis?

- Standard model of floating point computation:
  Let $a, b \in \mathbb{F}$, $\circ \in \{+,-,\times,\div\}$ and $\text{fl}(x \circ y)$ be the exact $x \circ y$ rounded to the nearest floating point value at precision $u$.
  \[
  \text{fl}(a \circ b) = (1 + \varepsilon_1)(a \circ b) = (a \circ b)/(1 + \varepsilon_2), \quad \text{with} \quad |\varepsilon_1|, |\varepsilon_2| \leq u.
  \]

- From one local rounding error to a global error bound (while $nu < 1$):
  \[
  \prod_{i=1}^{n} (1 + \varepsilon_i)^{\pm 1} = 1 + \theta_n, \quad \text{with} \quad |\theta_n| \leq \gamma_n := \frac{nu}{1 - nu} \approx nu.
  \]

- From an \textit{a priori} bound to a computable bound (while $nu < 1$):
  For $\widehat{\gamma}_n = (n \times u) \triangleleft (1 \ominus n \times u)$, we have
  \[
  \gamma_n \leq (1 + u)\widehat{\gamma}_n \leq (1 - u)\widehat{\gamma}_n,
  \]
  \[
  (1 + u)^n|x| \leq |x| \ominus (n \oplus 1) \otimes u.
  \]
Error bounds... illustrated with the Horner algorithm

- A priori and not validated analysis
- Towards validated *a priori* or dynamic error bounds

Compensated algorithms to double (at least) the accuracy

- Introducing example and error-free transformations (EFT)
- Compensated Horner algorithm

A validated error bound to control the actual accuracy

- Validated dynamic bound for CompHorner
- Application to a faithfully rounded polynomial evaluation

Performance issues exhibit challenging overheads for running time

- Overhead to more accuracy
- Overhead for a validated and more accurate result

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

\[ r_i = r_{i+1} \otimes x \oplus a_i \]

end
Relative accuracy of the Horner algorithm

We consider the polynomial

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

with \( a_i \in \mathbb{F}, \ x \in \mathbb{F} \)

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

\[
\frac{|\text{Horner}(p, x) - p(x)|}{|p(x)|} \leq \gamma 2^n \text{cond}(p, x).
\]

\( \approx 2nu \)

\( \text{cond}(p, x) \) 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 $\times u$

Accuracy of polynomial evaluation with the Horner scheme $[n=50]$

Relative forward error $u^{-1/n} u_{\text{cond}(p, x)} + O(u^2)$

Horner
Find a bound $B$ or a computable $\hat{B}$ 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) 
    $\hat{b}_{REA} = uE_0$, from $E_i = (E_{i+1} + |\hat{r}_{i+1}|)|x| + |\hat{r}_i|, (i = n - 1 : 0)$ and $E_n = 0$.  

Bounds for the absolute error in Horner algorithm

Find a bound $B$ or a computable $\hat{B}$ 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)
    $\hat{b}_{REA} = u E_0$, from $E_i = (E_{i+1} + |\hat{r}_{i+1}|)|x| + |\hat{r}_i|, (i = n - 1 : 0)$ and $E_n = 0$.

- Recent validated bounds when no underflow occurs
  - A validated a priori error bound: $\hat{B}_{AP} = \text{fl} \left( \frac{\gamma_{2n} \text{Horner}(|p|, |x|)}{1 - (2n+3)u} \right)$
  - A validated running error bound: $\hat{B}_{REA} = \text{fl} \left( \frac{u}{1 - (3n+1)u} \text{fl}(E_0) \right)$

- 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

Example: \((x-2)^3\)

- **Real absolute error**
- **Running error**
- **Faster running error**
- **GAPPA's bound**

The graph shows the comparison of various bounds with the real absolute error for the function \((x-2)^3\).
Validated error bounds for the Horner algorithm

Absolute error bounds for the evaluation of $p(x) = (1-x)^6$ in expanded form

- Error
- Horner1: a priori
- Horner2 and Horner3: REA
- Horner4 and Horner5: EFT

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

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
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

\[
\begin{align*}
2^{53} & \rightarrow 2^{53} - 1 \\
\rightarrow 2^{54} & \rightarrow -1 \\
\rightarrow -(2^{54} - 2) & \rightarrow 0 \\
\rightarrow 2 & \rightarrow 1
\end{align*}
\]

Relative error = 1

Compensation of the rounding errors

\[
\begin{align*}
2^{53} & \rightarrow 2^{53} - 1 \\
\rightarrow 2^{54} & \rightarrow -1 \\
\rightarrow -(2^{54} - 2) & \rightarrow 0 \\
\rightarrow 2 & \rightarrow 1
\end{align*}
\]

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

```
2^{53} \rightarrow 2^{54} \rightarrow -(2^{54} - 2) \rightarrow 1
```

Compensation of the rounding errors

```
2^{53} \rightarrow -(2^{54} - 2) \rightarrow 0
```

Relative error = 1

```
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.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Transformation</th>
<th>Flop</th>
<th>Author (Year)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>$(x, y) = 2\text{Sum}(a, b)$</td>
<td>6</td>
<td>Knuth (74)</td>
</tr>
<tr>
<td></td>
<td>such that $x = a \oplus b$ and $a + b = x + y$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>×</td>
<td>$(x, y) = 2\text{Prod}(a, b)$</td>
<td>17</td>
<td>Dekker (71)</td>
</tr>
<tr>
<td></td>
<td>such that $x = a \otimes b$ and $a \times b = x + y$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Algorithm (Knuth)**

```latex
function [x,y] = 2\text{Sum}(a,b)
    x = a \oplus b
    z = x \ominus a
    y = (a \ominus (x \ominus z)) \oplus (b \ominus z)
```
Error-Free Transformations are algorithms to compute the rounding errors at the current working precision.

\[
\begin{align*}
+ & \quad (x, y) = 2\text{Sum}(a, b) \\
& \text{such that } x = a \oplus b \text{ and } a + b = x + y \\
\times & \quad (x, y) = 2\text{Prod}(a, b) \\
& \text{such that } x = a \otimes b \text{ and } a \times b = x + y
\end{align*}
\]

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

**Algorithm (Knuth)**

function \([x, y] = 2\text{Sum}(a, b)\)

\[
\begin{align*}
x &= a \oplus b \\
z &= x \ominus a \\
y &= (a \ominus (x \ominus z)) \oplus (b \ominus z)
\end{align*}
\]

Compensated summation algorithms:

- Kahan, Møller (1965),
- Pichat (1972),
- Neumaier (1974),
- Priest (1992),
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 \( r_0 = \text{Horner}(p, x) \)

\[
\begin{align*}
    r_n &= a_n \\
    \text{for } i &= n - 1 : -1 : 0 \\
    p_i &= r_{i+1} \odot x \quad \text{\% error } \pi_i \in \mathbb{F} \\
    r_i &= p_i \oplus a_i \quad \text{\% error } \sigma_i \in \mathbb{F}
\end{align*}
\]

end

Let us define two polynomials \( p_\pi \) and \( p_\sigma \) 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 \( r_0 = \text{Horner} (p, x) \)

\[
\begin{align*}
    r_n &= a_n \\
    \text{for } i &= n - 1 : -1 : 0 \\
    p_i &= r_{i+1} \otimes x \quad \text{\% error } \pi_i \in \mathbb{F} \\
    r_i &= p_i \oplus a_i \quad \text{\% error } \sigma_i \in \mathbb{F} \\
\end{align*}
\]

end

Let us define two polynomials \( p_\pi \) and \( p_\sigma \) 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)**

\[
p(x) = \text{Horner} (p, x) + (p_\pi + p_\sigma)(x) \in \mathbb{F}
\]

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

Ph. Langlois (University of Perpignan, France)
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)

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

Algorithm (EFT for Horner)

\[
\begin{align*}
function \ [r_0, p_\pi, p_\sigma] &= \text{EFTHorner} (p, x) \\
&& \\
& \quad r_n = a_n \\
& \quad \text{for } i = n - 1 : -1 : 0 \\
& \quad \quad [p_i, \pi_i] = 2\text{Prod} (r_{i+1}, x) \\
& \quad \quad [r_i, \sigma_i] = 2\text{Sum} (p_i, a_i) \\
& \quad \quad p_\pi [i] = \pi_i \quad p_\sigma [i] = \sigma_i \\
& \quad \text{end}
\end{align*}
\]

Theorem (EFT for Horner algorithm)

\[
\begin{align*}
p(x) &= \text{Horner} (p, x) + (p_\pi + p_\sigma)(x). \\
& \quad \quad \text{exact value} \quad \in \mathbb{F} \quad \text{forward error}
\end{align*}
\]
Compensated Horner algorithm

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

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

Algorithm (Compensated Horner algorithm)

function \(\hat{r} = \text{CompHorner} (p, x)\)

\([\hat{r}, p_\pi, p_\sigma] = \text{EFTHorner} (p, x) \quad \% \quad \hat{r} = \text{Horner} (p, x)\)

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

\(\bar{r} = \hat{r} \oplus \hat{c}\)

Theorem

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

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

\(\approx (2nu)^2\)
Accuracy of the result $\lesssim u + \text{condition number} \times u^2$.

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 \( \approx 200 \) bits of precision
Error bounds... illustrated with the Horner algorithm
- A priori and not validated analysis
- Towards validated *a priori* or dynamic error bounds

Compensated algorithms to double (at least) the accuracy
- Introducing example and error-free transformations (EFT)
- Compensated Horner algorithm

A validated error bound to control the actual accuracy
- Validated dynamic bound for CompHorner
- Application to a faithfully rounded polynomial evaluation

Performance issues exhibit challenging overheads for running time
- Overhead to more accuracy
- Overhead for a validated and more accurate result

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

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

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

\( \bar{r} = \hat{r} \oplus \hat{c} \)
```

A priori error bound for the compensated evaluation:

\[
|\text{CompHorner}(p, x) - p(x)| \leq u|p(x)| + \gamma_{2n}^2 \tilde{p}(x) \approx (2nu)^2
\]

**Problem:** This a priori error bound

- cannot 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 $\tilde{r} = \text{CompHorner}(p, x)$

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

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

$\tilde{c} = \hat{c} \oplus \tilde{r}$ \% Rounding error $\delta = \hat{r} + \hat{c} - \tilde{r} \in \mathbb{F}$.

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

$$|\text{CompHorner}(p, x) - p(x)| \leq |\delta| + |\hat{c} - c|.$$
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 $[\bar{r}, \beta] = \text{CompHornerBound}(p, x)$

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

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

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

$[\bar{r}, \delta] = 2\text{Sum}(\hat{r}, \hat{c})$ \hspace{1cm} % Exact computation of $\delta$

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

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

**Theorem**

*Together with the compensated evaluation, CompHornerBound($p, x$) computes an a posteriori error bound $\beta$ 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 $x = 1$. 

![Graph showing a posteriori error bound and measured error](image-url)
Towards a faithfully rounded polynomial evaluation

**Definition**

A floating point number \( \hat{x} \) is said to be a faithful rounding of a real number \( x \) if

- either \( \hat{x} = x \),
- or \( \hat{x} \) is one of the two floating point neighbours of \( x \).

The worst case accuracy bound for CompHorner,

\[
\frac{|\text{CompHorner}(p, x) - p(x)|}{|p(x)|} \leq u + (2nu)^2 \text{cond}(p, x) + \mathcal{O}(u^3)
\]

is too large for reasoning about faithful rounding.
A sufficient condition for faithful rounding

We recall:

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

**Lemma**

\[ |\hat{c} - c| < \frac{u}{2} |\bar{r}| \Rightarrow \bar{r} \text{ is a faithful rounding of } p(x). \]

(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 $|\hat{c} - c|$ in the computed correcting term $\hat{c}$:

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

Bound satisfied when computed at running time in fp arithmetic.

Then,

$$\beta < \frac{\mu}{2} |\bar{r}| \Rightarrow |c - \hat{c}| < \frac{\mu}{2} |\bar{r}|$$

$$\Rightarrow \bar{r} \text{ is a faithful rounding of } p(x).$$

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)

\[
\frac{1}{u} + \frac{1}{2n} \text{ cond}(p, x)
\]

*A priori* bound on \text{cond}(p, x) to ensure faithful rounding
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\(^\text{1}\):

<table>
<thead>
<tr>
<th>System</th>
<th>GCC</th>
<th>ICC 9.1</th>
<th>ICC 9.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pentium 4, 3.00 GHz</td>
<td>2.8</td>
<td>8.6</td>
<td>3.0</td>
</tr>
<tr>
<td>(x87 fp unit)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICC 9.1</td>
<td>2.7</td>
<td>9.0</td>
<td>3.4</td>
</tr>
<tr>
<td>Athlon 64, 2.00 GHz</td>
<td>3.2</td>
<td>8.7</td>
<td>2.7</td>
</tr>
<tr>
<td>GCC 4.1.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Itanium 2, 1.4 GHz</td>
<td>2.8</td>
<td>6.7</td>
<td>2.4</td>
</tr>
<tr>
<td>GCC 4.1.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICC 9.1</td>
<td>1.5</td>
<td>5.9</td>
<td>3.9</td>
</tr>
</tbody>
</table>

\( \begin{array}{|c|c|c|}
\hline
\text{CompHorner} & \text{DDHorner} & \text{DDHorner} \\
\text{Horner} & \text{Horner} & \text{CompHorner} \\
\hline
2 - 4 & 6 - 9 & 2 - 4 \\
\hline
\end{array} \)

**CompHorner** runs at 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\(^1\):

<table>
<thead>
<tr>
<th>Platform</th>
<th>Compiler</th>
<th>CompHorner Horner</th>
<th>DDHorner Horner</th>
<th>CompHornerIsFaith Horner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pentium 4, 3.00 GHz</td>
<td>GCC 4.1.1</td>
<td>3.42</td>
<td>10.6</td>
<td>4.41</td>
</tr>
<tr>
<td>(sse fp unit)</td>
<td>ICC 9.1</td>
<td>3.09</td>
<td>9.35</td>
<td>3.96</td>
</tr>
<tr>
<td>Athlon 64, 2.00 GHz</td>
<td>GCC 4.1.2</td>
<td>3.96</td>
<td>10.4</td>
<td>4.35</td>
</tr>
<tr>
<td>Itanium 2, 1.4 GHz</td>
<td>GCC 4.1.1</td>
<td>3.30</td>
<td>8.20</td>
<td>4.05</td>
</tr>
<tr>
<td></td>
<td>ICC 9.1</td>
<td>1.93</td>
<td>9.68</td>
<td>2.26</td>
</tr>
</tbody>
</table>

\(~ 2 – 4\) \(~ 8 – 10\) \(~ 2 – 5\)

- **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.
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?