Reliable Implementation of Real Number Algorithms: Theory and Practice

Dagstuhl Seminar 06021 – January 08-13, 2006

FMA Implementations of the Compensated Horner Scheme

Stef GRAILLAT, Philippe LANGLOIS and Nicolas LOUVET

University of Perpignan, France.

http://webdali.univ-perp.fr/

Outline

1. Motivations and previous results: the Compensated Horner Scheme (CHS)

2. How to benefit from the Fused Multiply and Add operator (FMA)?

   How to derive Error Free Transformations for polynomial evaluations and FMA?

   (a) From HornerFMA to CompensatedHorner3FMA

   (b) From CompensatedHorner to CompensatedHornerFMA

3. Theoretical and numerical results

   (a) The accuracy verifies a “twice the working precision” behavior

   (b) The running-time is twice faster than double-double Horner scheme.

Provide numerical algorithms and software being

- a few times more accurate than the result from IEEE-754 working precision:
  - ▷ the actual accuracy is proven to satisfy improved versions of the “classic rule of thumb”;
- efficient in term of running-time without too much portability sacrifice:
  - ▷ only working with IEEE-754 precisions: single, double;
- together with a residual error bound to control the accuracy of the computed result:
  - ▷ dynamic and validated error bound computable in IEEE-754 arithmetic.

Example for polynomial evalution with Horner scheme:

▷ the Compensated Horner Scheme[a]

[a] S. Graillat, Ph.L., N. Louvet. *Compensated Horner Scheme*. Research Report, DALI-LP2A, University of Perpignan, aug. 05 (submitted).

The general rule of thumb for accuracy of a backward stable algorithm

- IEEE-754 double precision, with rounding to the nearest: $\mathtt{u} = 2^{-53} \approx 1.1 \cdot 10^{-16} \approx 16$ digits.

- The general "rule of thumb" (RoT) for backward stable algorithms:

  $$\boxed{\text{solution accuracy} \approx \text{condition number of the problem} \times \mathtt{u}}$$

  where $\mathtt{u}$ is the computing precision

- Condition number for the evaluation of $p(x) = \sum_{i=0}^{n} a_i x^i$:

  $$\text{cond}(p, x) = \frac{\sum_{i=0}^{n} |a_i||x|^i}{|p(x)|}, \text{ always } \geq 1.$$

- If $\text{cond}(p, x) \approx 10^{10}$, with $\mathtt{u} \approx 1.1 \cdot 10^{-16}$,

  $$\text{solution accuracy} \approx 10^{-6} \approx \text{only 6 digits!}$$

The compensated rule of thumb

▷ "Compensated RoT" for twice the current working precision behavior:

$$\boxed{\text{accuracy of the computed solution} \lesssim \text{precision} + \text{condition number} \times \text{precision}^2.}$$

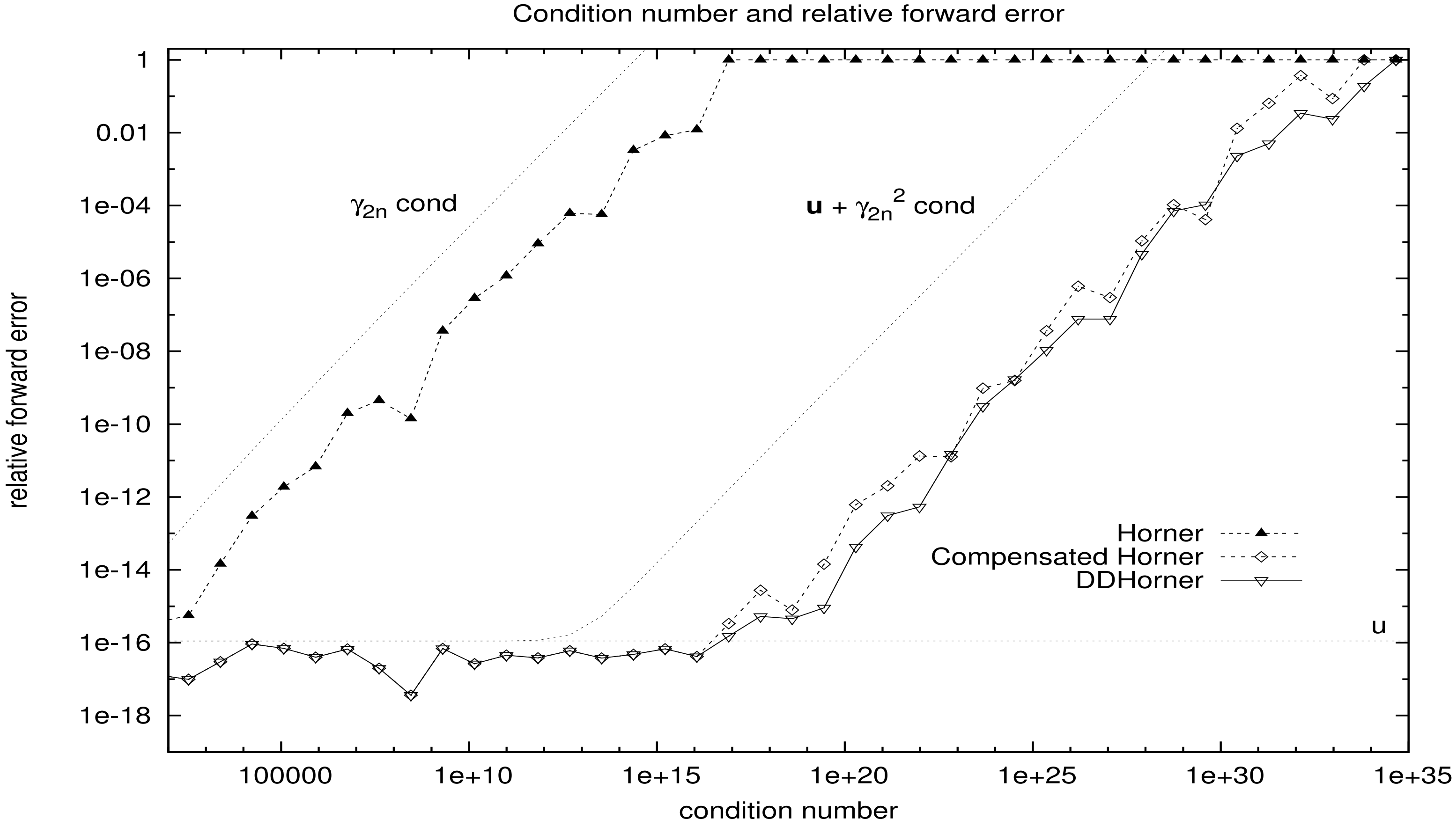
Theorem 1 *Given p a polynomial of degree n with floating point coefficients, and x a floating point value. If no underflow occurs,*

$$\frac{|\text{CompensatedHorner}\,(p,x) - p(x)|}{|p(x)|} \leq \mathtt{u} + \gamma_{2n}^{2} \textit{cond}(p,x),$$

with $\gamma_{2n} \approx n \times \mathtt{u}$.

- Proofs similar to Ogita-Rump-Oichi 2005 SISC paper.
- K-fold compensated "rule of thumb"

CHS: accuracy experiments exhibit a “twice the working precision behavior”



▷ CompensatedHorner and DDHorner (double-double) provide the same accuracy.

Compensated Horner Scheme: measured and theoretical ratios of the running-time

Pentium 4: 3.0GHz, 1024kB L2 cache - GCC 3.4.1				
ratio	minimum	mean	maximum	theoretical
CompensatedHorner/Horner	1.5	2.9	3.2	13
DDHorner/Horner	2.3	8.4	9.4	17

Theorem 2 *Given a polynomial p with floating point coefficients, and a floating point value x, we consider* $\mathsf{res} = \mathsf{CompensatedHorner}\,(p, x)$.
The absolute forward error affecting the evaluation is bounded according to

$$|\mathsf{CompensatedHorner}\,(p, x) - p(x)| \leq$$
$$\mathtt{fl}((\mathtt{u}|\mathsf{res}| + (\gamma_{4n+2}\mathsf{HornerSum}\,(|p_\pi|, |p_\sigma|, |x|) + 2\mathtt{u}^2|\mathsf{res}|))).$$

- The dynamic bound is computed in entire FPA with round to the nearest mode only
- The dynamic bound is less pessimistic than the a priori one.
- Proof uses EFT and the standard model of FPA (as in Ogita-Rump-Oishi paper)

Today

1. Motivations and previous results: the Compensated Horner Scheme (CHS)

2. How to benefit from the Fused Multiply and Add operator (FMA)?

   How to derive Error Free Transformations for polynomial evaluations and FMA?

   (a) From Horner to HornerFMA

   (b) From HornerFMA to CompensatedHorner3FMA

   (c) From CompensatedHorner to CompensatedHornerFMA

3. Theoretical and numerical results

   (a) The accuracy verifies a "twice the working precision" behavior

   (b) The running-time is twice faster than double-double Horner

Let us consider the availability of a FMA operator

- What is a Fused Multiply and Add FMA in floating point arithmetic?
  - ▷ Given a, b and c three floating point values, FMA (a, b, c) computes $a \times b + c$ rounded according to the current rounding mode.

    $\Rightarrow$ only one rounding error for two arithmetic operations!
  - ▷ The (FMA) is available on Intel Itanium, IBM Power PC...
- So, when we evaluate polynomials with the Horner scheme:
  - ▷ number of floating point operations (flops) $\Rightarrow$ divided by two,
  - ▷ number of rounding errors $\Rightarrow$ also divided by two!

For general polynomials and arguments, do we obtain a more accurate evaluation?

Algorithm 1 *Classic Horner scheme*

function $[s_0] = \text{Horner}(p, x)$

$s_n = a_n$

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

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

  $s_i = p_i \oplus a_i$

end

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

where $\gamma_{2n} = \frac{2n\mathbf{u}}{1-2n\mathbf{u}} \approx 2n\mathbf{u}$.

Algorithm 2 *Horner scheme with a FMA*

function $[s_0] = \text{Horner}(p, x)$

$s_n = a_n$

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

  $s_i = \text{FMA}\left(s_{i+1}, x, a_i\right)$

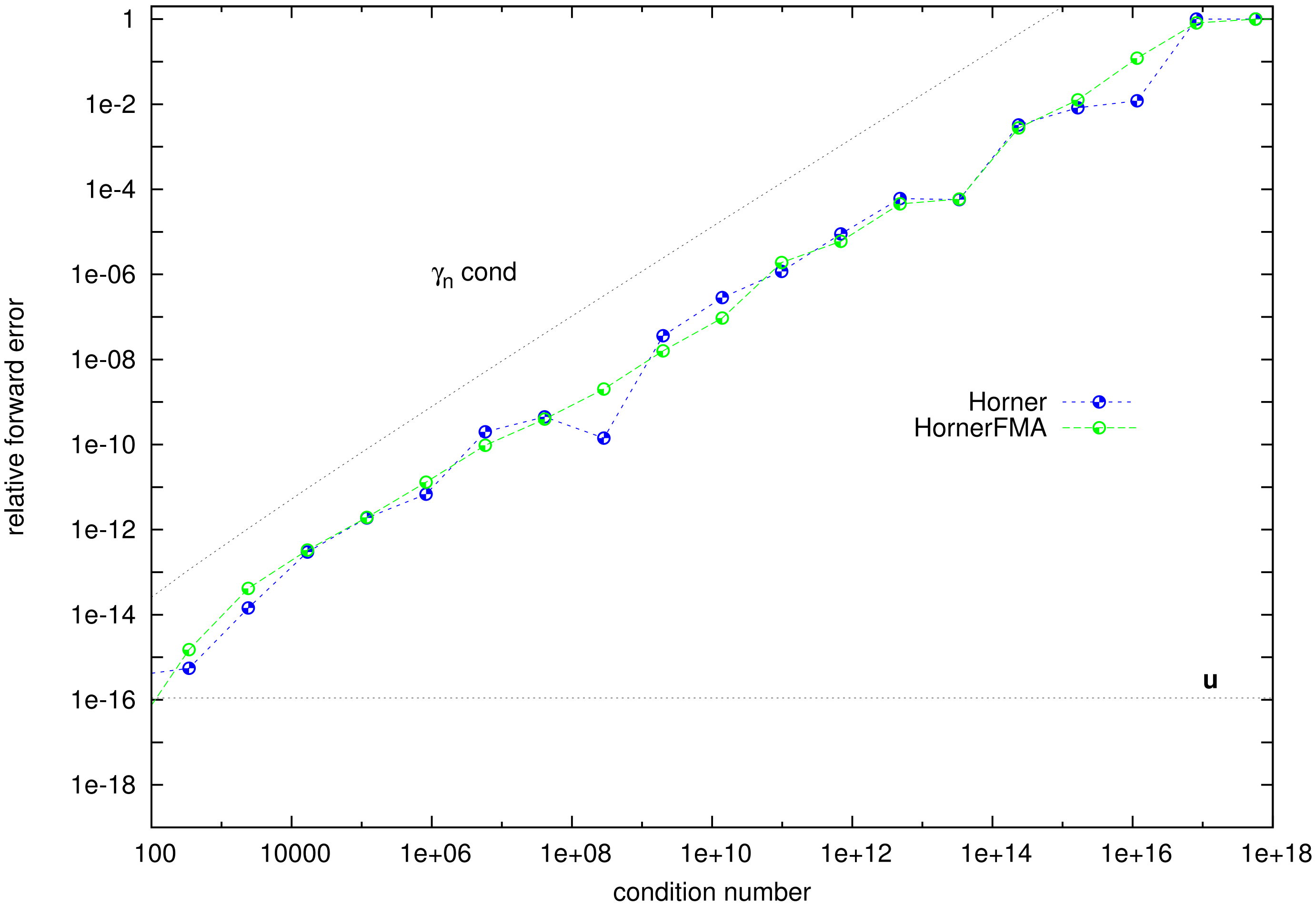
end

$$\boxed{\frac{|\text{HornerFMA}(p,x) - p(x)|}{|p(x)|} \leq \underbrace{\gamma_{n}}_{\approx n\mathbf{u}} \times \text{cond}(p, x)}$$

where $\gamma_{n} = \frac{n\mathbf{u}}{1-n\mathbf{u}} \approx n\mathbf{u}$.

Almost the same accuracy!

- $p_k(x) = (1-x)^k$, in expanded form, at $x = \mathtt{fl}(1.333)$
- k increases $\Rightarrow$ cond(p, x) increases!



More accuracy for ill-conditionned problems?

More accuracy for the Horner Scheme

More accuracy thanks to . . .

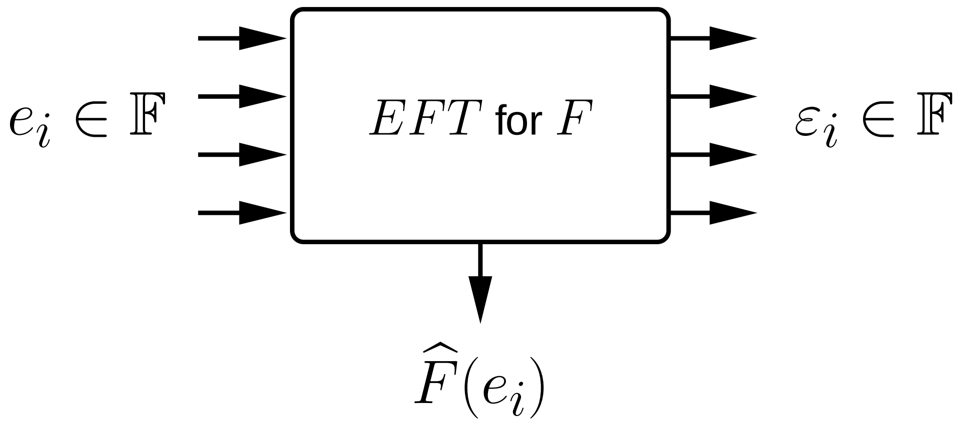
1. More bits
   - Many solutions: 80 bits register of x86, quadruple precision $\mathtt{u}^2$, MP, MPFR, Arprec/MPFUN.
   - Reference in our context is the fixed length floating point expansion, such as double-double ($\mathtt{u}^2$) and quad-double ($\mathtt{u}^4$).
2. Compensated algorithms
   - General idea: **correcting generated rounding errors**
   - Many examples: Kahan's compensated summation (65), . . . , Priest's doubly compensated summation (92), . . . , Ogita-Rump-Oishi in (SISC 05).
   - Sometimes, more bits (together with some extra-work) yield the definitive solution:

     accuracy $\approx \mathtt{u}$
   - The more general case:

     at a reasonable cost we can get a twice the current working precision behavior...

EFT for the summation

Error Free Transformations are properties and algorithms to compute the generated rounding errors at the current working precision.



$$F(e_i) = \widehat{F}(e_i) + G(\varepsilon_i)$$
$$\text{with } \operatorname{cond}(G, \varepsilon_i) < \operatorname{cond}(F, e_i)$$

Summation: Knuth (74)

▷ $(x, y) = \text{2Sum}\,(a, b)$ is such that $a + b = x + y$ and $x = a \oplus b$.

- Only in round to the nearest rounding mode: Bohlender *et al.* (91).
- 2Sum requires 6 flops.

Product: well known algorithm from Veltkamp and Dekker (71)

▷ $(x, y) = \text{2Prod}\,(a, b)$ is such that $a \times b = x + y$ and $x = a \otimes b$.

- In all the rounding modes: Bohlender *et al.* (91), Boldo and Daumas (03).
- Not valid in case of underflow.
- 2Prod requires 17 flops...

...but only two flops with a FMA !

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

Algorithm 3 *EFT for the product of two floating point numbers*

function $[x, y] = \text{2ProdFMA}\,(a, b)$
$x = a \otimes b$
$y = \text{FMA}\,(a, b, -x)$

FMA: Boldo and Muller, 2005

▷ $(x, y, z) = \text{3FMA}(a, b, c)$ is such that
$x = \text{FMA}(a, b, c)$ and $a \times b + c = x + y + z$.

- Three floating point values needed to represent the exact result.
- Only available in round to the nearest.

Algorithm 4 *EFT for the FMA operation*

function $[x, y, z] = \text{3FMA}(a, b, c)$
$x = \text{FMA}(a, b, c)$
$(u_1, u_2) = \text{2ProdFMA}(a, b)$
$(\alpha_1, z) = \text{2Sum}(b, u_2)$
$(\beta_1, \beta_2) = \text{2Sum}(u_1, \alpha_1)$
$y = (\beta_1 \ominus x) \oplus \beta_2$

- 3FMA requires 17 flops.

Two Compensated Horner Schema with FMA

1. Improve the EFT for multiplication in Horner with 2ProdFMA

   ▷ EFTHornerFMA and CompHornerFMA,

2. Apply the EFT for FMA in HornerFMA with 3FMA

   ▷ EFTHorner3FMA and CompHorner3FMA.

- We prove that the two corresponding algorithms verify the “twice the current working precision behavior”.

An EFT for the Horner scheme with 2ProdFMA for Horner

If no underflow occurs,

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

with $p_\pi(X) = \sum_{i=0}^{n-1} \pi_i X^i, \quad p_\sigma(X) = \sum_{i=0}^{n-1} \sigma_i X^i$, and π_i and σ_i in $\mathbb{F}$.

Algorithm 5 *Classic Horner scheme*

function $[h] = \mathsf{Horner}(p, x)$

$s_n = a_n$

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

  $p_i = s_{i+1} \otimes x$ % rounding error π_i

  $s_i = p_i \oplus a_i$ % rounding error σ_i

end

$h = s_0$

Algorithm 6 *EFT for the Horner scheme*

function $[h, p_\pi, p_\sigma] = \mathsf{EFTHornerFMA}(p, x)$

$s_n = a_n$

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

  $[p_i, \pi_i]$ = *2ProdFMA*(s_{i+1}, x)

  $[s_i, \sigma_i]$ = *2Sum*(p_i, a_i)

  Let π_i be the coefficient of degree i in p_π

  Let σ_i be the coefficient of degree i in p_σ

end

$h = s_0$

- Since $p(x) = \text{Horner}\,(p,x) + (p_\pi + p_\sigma)(x)$, we correct $\text{Horner}\,(p,x)$ by computing an approximate of $(p_\pi + p_\sigma)(x)$.

Algorithm 7 *Compensated Horner scheme*

function $[res] = \text{CompHornerFMA}\,(p,x)$
$[h, p_\pi, p_\sigma] = \text{EFTHornerFMA}\,(p,x)$
$c = \text{HornerFMA}\,(p_\pi \oplus p_\sigma, x)$
$res = h \oplus c$

- **Theorem 3** *Given p a polynomial with floating point coefficients, and x a floating point value,*

$$\frac{|\text{CompHornerFMA}\,(p,x) - p(x)|}{|p(x)|} \leq \mathbf{u} + (1+\mathbf{u})\gamma_n\gamma_{2n}\textit{cond}(p,x)$$

$$\leq \mathbf{u} + 2n^2\mathbf{u}^2\textit{cond}(p,x) + O(\mathbf{u}^3).$$

with $\quad \gamma_k = \frac{k\mathbf{u}}{1-k\mathbf{u}} \quad \Rightarrow \quad (1+\mathbf{u})\gamma_n\gamma_{2n} \approx 2n^2\mathbf{u}^2.$

Another EFT with 3FMA for HornerFMA

If no underflow occurs,

$$\boxed{p(x) = \text{HornerFMA}\,(p, x) + (p_\varepsilon + p_\varphi)(x),}$$

with $p_\varepsilon(X) = \sum_{i=0}^{n-1} \varepsilon_i X^i, \quad p_\varphi(X) = \sum_{i=0}^{n-1} \varphi_i X^i$ and ε_i and φ_i in $\mathbb{F}$.

Algorithm 8 *Horner scheme with a FMA*

function $[h] = \text{HornerFMA}\,(p, x)$

$u_n = a_n$

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

  u_i = *FMA*(u_{i+1}, x, a_i)

    % rounding error $\varepsilon_i + \varphi_i$

end

$h = u_0$

Algorithm 9 *EFT for the Horner scheme*

function $[h, p_\varepsilon, p_\varphi] = \text{EFTHornerFMA}(p, x)$

$s_n = a_n$

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

  $[u_i, \varepsilon_i, \varphi_i]$ = *3FMA*(u_{i+1}, x, a_i)

  Let ε_i be the coefficient of degree i in p_ε

  Let φ_i be the coefficient of degree i in p_φ

end

$h = u_0$

- Since $p(x) = \text{Horner}\,(p,x) + (p_\varepsilon + p_\varphi)(x)$, we correct $\text{Horner}\,(p,x)$ by computing an approximate of $(p_\varepsilon + p_\varphi)(x)$.

Algorithm 10 *Compensated Horner scheme with 3FMA*

function $[res] = \text{CompHorner3FMA}\,(p,x)$
$[h, p_\varepsilon, p_\varphi] = \text{EFTHorner3FMA}\,(p,x)$
$c = \text{HornerFMA}\,(p_\varepsilon \oplus p_\varphi, x)$
$res = h \oplus c$

- **Theorem 4** *Given p a polynomial with floating point coefficients, and x a floating point value,*

$$\begin{aligned}\frac{|\text{CompHorner3FMA}\,(p,x) - p(x)|}{|p(x)|} &\leq \mathbf{u} + (1+\mathbf{u})\gamma_n^2 \textit{cond}(p,x) \\ &\leq \mathbf{u} + n^2\mathbf{u}^2 \textit{cond}(p,x) + O(u^3).\end{aligned}$$

with $\gamma_n = \frac{n\mathbf{u}}{1-n\mathbf{u}} \quad \Rightarrow \quad (1+\mathbf{u})\gamma_n^2 \approx n^2\mathbf{u}^2$.

Summary of the results

Algorithm	EFT	Relative error bound	flops
CompHornerFMA	EFTHornerFMA (2ProdFMA)	$\mathtt{u} + 2n^2\mathtt{u}^2\text{cond}(p,x) + O(\mathtt{u}^3)$	$10n - 1$
CompHorner3FMA	EFTHorner3FMA (3FMA)	$\mathtt{u} + n^2\mathtt{u}^2\text{cond}(p,x) + O(\mathtt{u}^3)$	$19n$

Almost the same accuracy!

But EFTHornerFMA requires about 2 times less flops than EFTHorner3FMA.

Numerical experiments

- Experimental results exhibit the
  - ▷ actual accuracy: twice the current working precision behavior,
  - ▷ actual speed: about twice faster than the corresponding double-double subroutine.

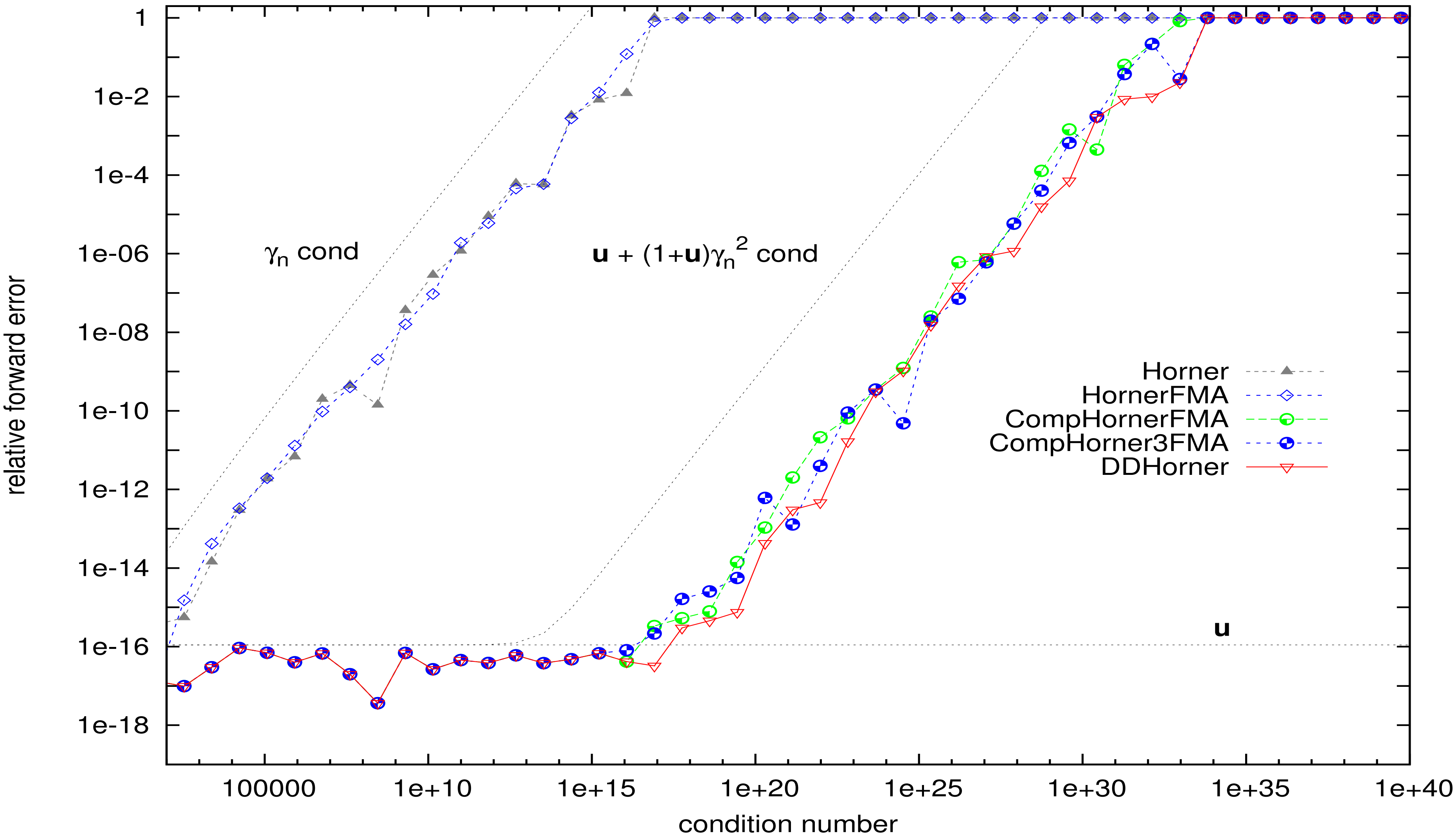
Experimented algorithms

- We compare
  - ▷ HornerFMA: IEEE-754 double precision Horner scheme + FMA.
  - ▷ CompHornerFMA: compensated Horner scheme with 2ProdFMA.
  - ▷ CompHorner3FMA: compensated HornerFMA scheme with 3FMA.
  - ▷ DDHorner: classical Horner scheme + internal double-double computation:

    $\longrightarrow$ based on double-double Bailey library,

    $\longrightarrow$ also benefits from the FMA.
- All computations are performed in C language and IEEE-754 double precision.

Accuracy experiments exhibit a twice the working precision behavior



CompHornerFMA, CompHorner3FMA and DDHorner provide the same accuracy.

Numerical experiments: testing the speed efficiency

- What is measured?

  The overhead to double the accuracy with the time ratios:

  - ▷ CompHornerFMA/HornerFMA,
  - ▷ CompHorner3FMA/HornerFMA,
  - ▷ DDHorner/HornerFMA

Speed efficiency: measured and theoretical ratios

environment	CompHornerFMA / HornerFMA mean	CompHornerFMA / HornerFMA theo.	CompHorner3FMA / HornerFMA mean	CompHorner3FMA / HornerFMA theo.	DDHorner / HornerFMA mean	DDHorner / HornerFMA theo.
Itanium 1, 733MHz, GCC 2.96	2.6	10	4.8	19	6.5	20
Itanium 2, 900MHz, GCC 3.3.5	2.5	10	4.4	19	7.9	20
Itanium 2, 1.6GHz, GCC 3.4.4	3.8	10	6.0	19	7.3	20

The corrected algorithm runs about twice faster than corresponding double-double implementation.

Conclusion

- For general polynomials, the FMA mainly improves the efficiency, but not the accuracy.
- To improves the accuracy, we have presented two compensated Horner schemes:
  - ▷ CompHorner3FMA,
  - ▷ CompHornerFMA.
- Both exhibit the “twice the current working precision” behavior.
- CompHornerFMA is much faster than CompHorner3FMA

  $\Rightarrow$ more interesting to correct $\otimes$ (with 2ProdFMA) than the FMA (with 3FMA).
- CompHornerFMA is about twice faster than DDHorner based on double-double.
- K-fold versions from EFT for polynomial evaluation
- Research Reports at `http://webdali.univ-perp.fr`

.

Speed efficiency: measured and theoretical ratios

ratio	minimum	mean	maximum	theoretical
Itanium 1, 733 MHz, GCC 2.9.6				
CompHornerFMA/HornerFMA	1.7	2.6	2.7	10
CompHorner3FMA/HornerFMA	2.3	4.8	5.1	19
DDHorner/HornerFMA	2.9	6.5	7.0	20
Itanium 2, 1.9GHz, GCC 3.3.3				
CompHornerFMA/HornerFMA	1.7	2.5	2.6	10
CompHorner3FMA/HornerFMA	2.4	4.4	4.6	19
DDHorner/HornerFMA	3.0	7.9	8.5	20
Itanium 2, 1.6GHz, GCC 3.4.4				
CompHornerFMA/HornerFMA	2.4	3.8	4.1	10
CompHorner3FMA/HornerFMA	3.4	6.0	6.3	19
DDHorner/HornerFMA	3.8	7.3	7.6	20