Compensated Horner scheme in complex floating point arithmetic

Stef Graillat and Valérie Ménissier-Morain

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

8th Conference on Real Numbers and Computers
Santiago de Compostela, Spain, July 7-9, 2008

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)

Real floating point arithmetic

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

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

Algorithm `TwoProduct` by Veltkamp and Dekker (1971)

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

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

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

EFT for the product (2/3)

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

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

EFT for the product (3/3)

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

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

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.*

Accurate sum and dot product

Compensated algorithms

Algorithm 6 (Ogita, Rump and Oishi 2005)

Summation in twice the working precision

function $\texttt{res} = \texttt{Sum2}(p)$
  $\pi_1 = p_1\,;\ \sigma_1 = 0\,;$
  for $i = 2 : n$
    $[\pi_i, q_i] = \texttt{TwoSum}(\pi_{i-1}, p_i)$
    $\sigma_i = \mathrm{fl}(\sigma_{i-1} + q_i)$
  $\texttt{res} = \mathrm{fl}(\pi_n + \sigma_n)$

Algorithm 7 (Ogita, Rump and Oishi 2005)

Dot product in twice the working precision

function $\texttt{res} = \texttt{Dot2}(x, y)$
  $[p, s] = \texttt{TwoProduct}(x_1, y_1)$
  for $i = 2 : n$
    $[h, r] = \texttt{TwoProduct}(x_i, y_i)$
    $[p, q] = \texttt{TwoSum}(p, h)$
    $s = \mathrm{fl}(s + (q + r))$
  end
  $\texttt{res} = \mathrm{fl}(p + s)$

Accurate sum and dot product

Proposition 1 (Ogita, Rump and Oishi 2005)

Suppose Algorithm Sum2 *is applied to floating point numbers* $p_i \in \mathbb{F}$, $1 \leq i \leq n$. *Let* $s := \sum p_i$, $S := \sum |p_i|$. *Then, we have*

$$|\mathtt{res} - s| \leq \mathbf{u}|s| + \gamma_{n-1}^2 S \approx \mathbf{u}|s| + n^2\mathbf{u}^2 S.$$

Proposition 2 (Ogita, Rump and Oishi 2005)

Let floating point numbers $x_i, y_i \in \mathbb{F}, 1 \leq i \leq n$, *be given and denote by* $\mathtt{res} \in \mathbb{F}$ *the result computed by Algorithm* Dot2. *Then occurs,*

$$|\mathtt{res} - x^T y| \leq \mathbf{u}|x^T y| + \gamma_n^2 |x^T||y| \approx \mathbf{u}|x^T y| + n^2\mathbf{u}^2 |x^T||y|.$$

$$\gamma_n = \frac{n\mathbf{u}}{1 - n\mathbf{u}} \approx n\mathbf{u}$$

Complex floating point arithmetic

What about complex numbers?

Splitting between real and imaginary part

- Summation (`Sum2cplx`)
  $s = \sum_{j=1}^{n} p_j$ with $p_j = a_j + ib_j$

$$\rightarrow \quad s = \underbrace{\sum_{j=1}^{n} a_j}_{\texttt{Sum2}} + i \underbrace{\sum_{j=1}^{n} b_j}_{\texttt{Sum2}}$$

- Dot product (`Dot2cplx`)
  $x = (x_j)$ with $x_j = a_j + ib_j$ and $y = (y_j)$ with $y_j = c_j + id_j$, $p = x^* y$

$$\rightarrow \quad p = \underbrace{\begin{bmatrix} \mathrm{Re}(x) \\ \mathrm{Im}(x) \end{bmatrix}^T \begin{bmatrix} \mathrm{Re}(y) \\ \mathrm{Im}(y) \end{bmatrix}}_{\texttt{Dot2}} + i \underbrace{\begin{bmatrix} \mathrm{Re}(x) \\ \mathrm{Im}(x) \end{bmatrix}^T \begin{bmatrix} \mathrm{Im}(y) \\ -\mathrm{Re}(y) \end{bmatrix}}_{\texttt{Dot2}}$$

Error bounds

Proposition 3

Suppose Algorithm `Sum2cplx` *is applied to floating point numbers* $p_j = a_j + ib_j \in \mathbb{F} + i\mathbb{F}$, $1 \leq j \leq n$. *Let* $s := \sum p_j$, $S := \sum |p_j|$. *Then, we have*

$$|\mathtt{res} - s| \leq \sqrt{2}\mathbf{u}|s| + 2\gamma_{n-1}^2 S.$$

Proposition 4

Let floating point numbers $x = (x_j)$ *with* $x_j = a_j + ib_j$ *and* $y = (y_j)$ *with* $y_j = c_j + id_j$ *be given and denote by* $\mathtt{res} \in \mathbb{F} + i\mathbb{F}$ *the result computed by Algorithm* `Dot2cplx`. *Then occurs,*

$$|\mathtt{res} - x^* y| \leq \sqrt{2}\mathbf{u}|x^* y| + 2\gamma_{2n}^2 |x|^T |y|.$$

More difficult for polynomial evaluation

Compensated Horner scheme (Graillat, Langlois and Louvet 2005)

$$p(z) = \sum_{j=0}^{n} a_j z^j, \qquad a_j \in \mathbb{C}, z = x + iy \in \mathbb{C}$$

$\longrightarrow$ Write $p(z) = p_r(x, y) + i p_i(x, y)$ with p_r and p_i with real coefficients and evaluate p_r and p_i with Horner scheme

Problem : need formal manipulations

$\Rightarrow$ need new EFT for complex floating point arithmetic

Complex EFT (1/2)

Given $x, y \in \mathbb{F} + i\mathbb{F}$,

$$\mathrm{fl}(x \circ y) = (x \circ y)(1 + \varepsilon_1), \text{ for } \circ \in \{+, -\} \text{ and } |\varepsilon_\nu| \leq \mathbf{u},$$

and

$$\mathrm{fl}(x \cdot y) = (x \cdot y)(1 + \varepsilon_1), |\varepsilon_1| \leq \sqrt{2}\gamma_2.$$

Algorithm 8 (EFT of the sum of 2 complex floating point numbers $x = a + ib$ and $y = c + id$)

function $[s, e] = \texttt{TwoSumCplx}(x, y)$
  $[s_1, e_1] = \texttt{TwoSum}(a, c)$
  $[s_2, e_2] = \texttt{TwoSum}(b, d)$
  $s = s_1 + is_2$
  $e = e_1 + ie_2$

Complex EFT (2/2)

Algorithm 9 (EFT of the product of two complex floating point numbers $x = a + ib$ and $y = c + id$)

function $[p, e, f, g] = \texttt{TwoProductCplx}(x, y)$
  $[z_1, h_1] = \texttt{TwoProduct}(a, c)$
  $[z_2, h_2] = \texttt{TwoProduct}(b, d)$
  $[z_3, h_3] = \texttt{TwoProduct}(a, d)$
  $[z_4, h_4] = \texttt{TwoProduct}(b, c)$
  $[z_5, h_5] = \texttt{TwoSum}(z_1, -z_2)$
  $[z_6, h_6] = \texttt{TwoSum}(z_3, z_4)$
  $p = z_5 + iz_6$
  $e = h_1 + ih_3$
  $f = -h_2 + ih_4$
  $g = h_5 + ih_6$

Summary

Theorem 2

Let $x, y \in \mathbb{F} + i\mathbb{F}$ *and let* $s, e \in \mathbb{F} + i\mathbb{F}$ *such that* $[s, e] = \texttt{TwoSumCplx}(x, y)$. *Then,*

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

The algorithm `TwoSumCplx` *requires* 12 *flops.*

Theorem 3

Let $x, y \in \mathbb{F} + i\mathbb{F}$ *and let* $p, e, f, g \in \mathbb{F} + i\mathbb{F}$ *such that* $[p, e, f, g] = \texttt{TwoProductCplx}(x, y)$. *Then,*

$$x \cdot y = p + e + f + g \quad p = \mathrm{fl}(x \cdot y), \quad |e + f + g| \leq \sqrt{2}\gamma_2 |x \cdot y|,$$

The algorithm `TwoProductCplx` *requires* 80 *flops.*

`TwoProductCplx` requires 20 flops if one uses `TwoProductFMA`.

The Horner scheme

Algorithm 10 (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

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

end

$\texttt{res} = s_0$

EFT for the polynomial evaluation

We now propose an EFT for the polynomial evaluation with the Horner scheme.

Algorithm 11 (EFT for the Horner scheme)

function $[h, p_\pi, p_\mu, p_\nu, p_\sigma] = \texttt{EFTHornerCplx}(p, x)$
  $s_n = a_n$
  for $i = n-1 : -1 : 0$
    $[p_i, \pi_i, \mu_i, \nu_i] = \texttt{TwoProductCplx}(s_{i+1}, x)$
    $[s_i, \sigma_i] = \texttt{TwoSumCplx}(p_i, a_i)$
    Let π_i be the coefficient of degree i in p_π
    Let μ_i be the coefficient of degree i in p_μ
    Let ν_i be the coefficient of degree i in p_ν
    Let σ_i be the coefficient of degree i in p_σ
  end
  $h = s_0$

Complex compensated Horner scheme

$$\boxed{p(x) = h + (p_\sigma + p_\pi + p_\mu + p_\nu)(x)}$$

Algorithm 12 (Evaluation of the sum of four polynomials)

function $\texttt{res} = \texttt{HornerSumAcc}(p, q, r, s, x)$
  $r_n = \texttt{Accsum}(a_n + b_n + c_n + d_n)$
  for $i = n-1 : -1 : 0$
    $r_i = \mathrm{fl}(r_{i+1} \cdot x + \texttt{Accsum}(a_i + b_i + c_i + d_i))$
  end
  $\texttt{res} = r_0$

Algorithm 13 (Complex compensated Horner scheme)

function $\texttt{res} = \texttt{CompHornerCplx}(p, x)$
  $[h, p_\pi, p_\mu, p_\nu, p_\sigma] = \texttt{EFTHornerCplx}(p, x)$
  $c = \texttt{HornerSumAcc}(p_\pi, p_\mu, p_\nu, p_\sigma, x)$
  $\texttt{res} = \mathrm{fl}(h + c)$

Complex compensated Horner scheme

Theorem 4

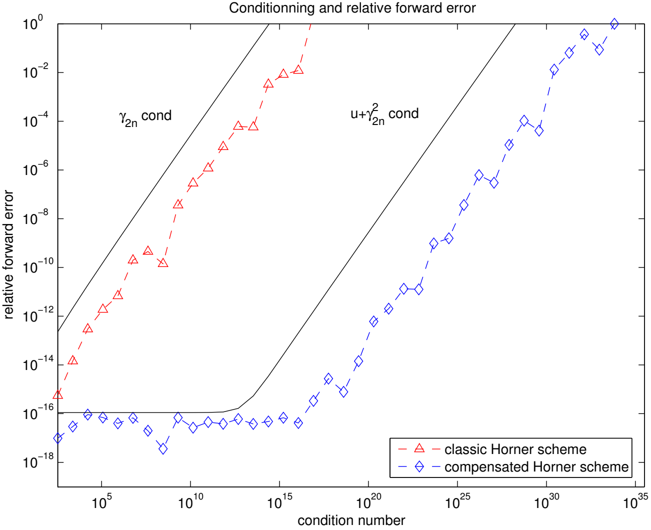
Given a polynomial $p = \sum_{i=0}^{n} a_i x^i$ of degree n with floating point coefficients, and x a floating point value. We consider the result $\texttt{CompHorner}(p, x)$ *computed by* $\texttt{CompHorner}$. *Then,*

$$|\texttt{CompHorner}(p, x) - p(x)| \leq \mathbf{u}|p(x)| + \widetilde{\gamma}_{2n}^2 \widetilde{p}(x).$$

$$\widetilde{\gamma}_n := \frac{n\sqrt{2}\gamma_2}{1 - n\sqrt{2}\gamma_2}.$$

Numerical experiment

$p(x) = (x - (1 + i))^n$ evaluated at $x = \mathrm{fl}(1.333 + 1.333i)$ and $n = 3 : 42$



Conclusion

- Compensated algorithms in complex floating point arithmetic :
  - use of real EFT when possible
  - use of complex EFT otherwise
  - complex version of the Compensated Horner Scheme
- Future work
  - validation in complex floating point arithmetic

Thank you for your attention