Faithful roundings of sum with nonnegative entries

Stef Graillat ステフ・グライヤ

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

JSIAM Meeting 2011

Doshisha University, Kyoto, Japan, September 14-16, 2011

Motivations

- Computing summation is a basic task in scientific computing
- Classic algorithm is recursive summation algorithm

Algorithm 1 (Recursive summation algorithm)

function $\texttt{res} = \texttt{Sum}(p)$
$s = 0$
for $i = 1 : n$
$s = \mathrm{fl}(s + p_i)$
$\texttt{res} = s$

- But due to rounding errors, the computed result can be far from the exact result

Outline of the talk

- Motivations
- Basic of floating-point arithmetic
- Faithful roundings of sum with nonnegative entries
- Faithful roundings of product of floating-point numbers
- Conclusion and future work

Floating-point numbers

Normalized floating-point numbers $\mathbb{F} \subset \mathbb{R}$:

$$x = \pm \underbrace{x_0.x_1 \dots x_{M-1}}_{mantissa} \times b^e, \quad 0 \le x_i \le b-1, \quad x_0 \neq 0$$

b : basis, M : precision, e : exponent such that $e_{\min} \le e \le e_{\max}$

Approximation of $\mathbb{R}$ by $\mathbb{F}$ with rounding $\mathrm{fl} : \mathbb{R} \to \mathbb{F}$.
Let $x \in \mathbb{R}$ then

$$\mathrm{fl}(x) = x(1+\delta), \quad |\delta| \le \mathbf{u}$$

Unit rounding $\mathbf{u} = b^{1-M}/2$ for rounding to nearest

Standard model of floating-point arithmetic

Let $x, y \in \mathbb{F}$ and $\circ \in \{+, -, \cdot, /\}$.

The result $x \circ y$ is not in general a floating-point number

$$\mathrm{fl}(x \circ y) = (x \circ y)(1 + \delta), \quad |\delta| \leq \mathbf{u}$$

IEEE 754 standard (1985 and 2008)

Type	Size	Mantissa	Exponent	Unit rounding	Interval
Single	32 bits	23+1 bits	8 bits	$\mathbf{u} = 2^{-24} \approx 5,86 \times 10^{-8}$	$\approx 10^{\pm 38}$
Double	64 bits	52+1 bits	11 bits	$\mathbf{u} = 2^{-53} \approx 1,11 \times 10^{-16}$	$\approx 10^{\pm 308}$

Error analysis for the recursive algorithm

Algorithm 2 (Recursive summation algorithm)

function $\texttt{res} = \texttt{Sum}(p)$
  $s = 0$
  for $i = 1 : n$
    $s = \text{fl}(s + p_i)$
  $\texttt{res} = s$

Lemma 1 (Higham)

Let $s = \sum_{i=1}^{n} p_i$. *Then we have*

$$|\texttt{res} - s| \leq \gamma_{n-1} \sum_{i=1}^{n} |p_i|$$

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

Verified computing

As

$$|\texttt{res} - s| \le \gamma_{n-1} \sum_{i=1}^{n} |p_i|$$

then getting a tight error bound needs to accurately evaluate

$$\sum_{i=1}^{n} |p_i|$$

→ need to accurately evaluate the sum of nonnegative numbers

Conditioning of summation

Condition numbers measure the sensitivity of the solution of a problem to perturbation in the data

$$\operatorname{cond}\left(\sum p_i\right) := \lim_{\varepsilon \to 0} \sup \left\{ \left| \frac{\sum (p_i + \tilde{p}_i) - \sum p_i}{\varepsilon \sum p_i} \right| : |\tilde{p}_i| \le \varepsilon |p| \right\}$$

It is well-known that

$$\operatorname{cond}\left(\sum p_i\right) = \frac{\sum |p_i|}{|\sum p_i|}$$

So for nonnegative numbers

$$\operatorname{cond}\left(\sum p_i\right) = 1$$

The problem is then well-conditioned

Recursive algorithm with nonnegative numbers

Algorithm 3 (Recursive summation algorithm)

function $\texttt{res} = \texttt{Sum}(p)$
  $s = 0$
  for $i = 1 : n$
    $s = \mathrm{fl}(s + p_i)$
  $\texttt{res} = s$

If $s := \sum p_i \neq 0$ then

$$\frac{|\texttt{res} - s|}{|s|} \leq \gamma_{n-1} \approx (n-1)\mathbf{u}$$

Good accuracy if n is small but possibly no accuracy at all if $n \approx 1/\mathbf{u}$

Getting more accuracy with compensated algorithms

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,

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

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)

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 4 (EFT of the sum of 2 floating point numbers with $|a| \geq |b|$)

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

Algorithm 5 (EFT of the sum of 2 floating point numbers)

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

Error bound for EFT of the sum

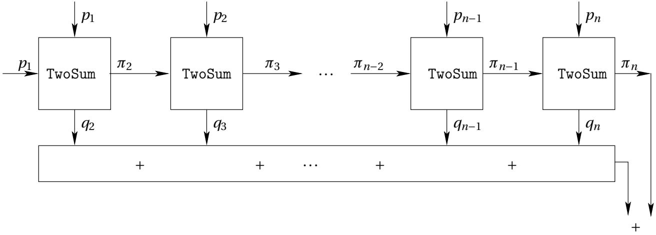
Theorem 1

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

$$a + b = x + y, \quad x = \mathrm{fl}(a + b), \quad |y| \le \mathbf{u}|x|, \quad |y| \le \mathbf{u}|a + b|.$$

The algorithm `TwoSum` *requires* 6 *flops.*

Compensated summation algorithm



Algorithm 6 (Ogita, Rump, Oishi (2005))

function $\texttt{res} = \texttt{CompSum}(p)$

$\pi_1 = p_1$; $\sigma_1 = 0$;

for $i = 2 : n$

$[\pi_i, q_i] = \texttt{TwoSum}(\pi_{i-1}, p_i)$

$\sigma_i = \mathrm{fl}(\sigma_{i-1} + q_i)$

$\texttt{res} = \mathrm{fl}(\pi_n + \sigma_n)$

Compensated summation algorithm

Algorithm 7 (Ogita, Rump, Oishi (2005))

function $\texttt{res} = \texttt{CompSum}(p)$

  $\pi_1 = p_1\ ; \sigma_1 = 0;$

  for $i = 2 : n$

    $[\pi_i, q_i] = \texttt{TwoSum}(\pi_{i-1}, p_i)$

    $\sigma_i = \mathrm{fl}(\sigma_{i-1} + q_i)$

  $\texttt{res} = \mathrm{fl}(\pi_n + \sigma_n)$

Let $s = \sum_{i=1}^{n} p_i$. Then one has (Ogita, Rump, Oishi 2005)

$$|\texttt{res} - s| \le \mathbf{u}|\sum_{i=1}^{n} p_i| + \gamma_{n-1}^2 \sum_{i=1}^{n} |p_i|$$

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

Faithful rounding (1/3)

Floating point predecessor and successor of a real number r satisfying $\min\{f : f \in \mathbb{R}\} < r < \max\{f : f \in \mathbb{F}\}$:

$$\mathrm{pred}(r) := \max\{f \in \mathbb{F} : f < r\} \quad \text{and} \quad \mathrm{succ}(r) := \min\{f \in \mathbb{F} : r < f\}.$$

Definition 1

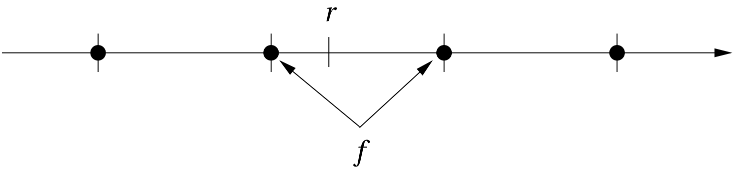
A floating point number $f \in \mathbb{F}$ is called a faithful rounding of a real number $r \in \mathbb{R}$ if

$$\mathrm{pred}(f) < r < \mathrm{succ}(f).$$

We denote this by $f \in \square(r)$. For $r \in \mathbb{F}$, this implies that $f = r$.

Faithful rounding means that the computed result is equal to the exact result if the latter is a floating point number and otherwise is one of the two adjacent floating point numbers of the exact result.

Faithful rounding (2/3)



Lemma 2 (Rump, Ogita and Oishi, 2005)

Let $r, \delta \in \mathbb{R}$ *and* $\tilde{r} := \mathrm{fl}(r)$. *Suppose that* $2|\delta| < \mathbf{u}|\tilde{r}|$. *Then* $\tilde{r} \in \Box(r+\delta)$, *that means* $\tilde{r}$ *is a faithful rounding of* $r+\delta$.

Faithful rounding (3/3)

Let `res` = `CompSum`(p)

Theorem 2 (Graillat 2011)

Suppose `CompSum` *algorithm is applied to nonnegative floating-point number* $p_i \in \mathbb{F}$, $1 \le i \le n$ *and that*

$$n < 1 + \frac{\sqrt{1-\mathbf{u}}}{\sqrt{2}\sqrt{1+\mathbf{u}} + \sqrt{1-\mathbf{u}}}\mathbf{u}^{-1/2}.$$

Then the result `res` *is a faithful rounding of* $s := \sum p_i \ge 0$.

If $n < \alpha \mathbf{u}^{-1/2}$ where $\alpha \approx 0.4$ then the result is faithfully rounded

In double precision where $\mathbf{u} = 2^{-53}$, if $n \lesssim 3 \cdot 10^7$, we get a faithfully rounded result

Classic method for computing product

The classic method for evaluating a product of n numbers $a = (a_1, a_2, \dots, a_n)$

$$p = \prod_{i=1}^{n} a_i$$

is the following algorithm.

Algorithm 8 (Product evaluation)

function $\texttt{res} = \texttt{Prod}(a)$
  $p_1 = a_1$
  for $i = 2 : n$
    $p_i = \mathrm{fl}(p_{i-1} \cdot a_i)$ % rounding error π_i
  end
  $\texttt{res} = p_n$

This algorithm requires $n - 1$ flops

Error analysis

$$\gamma_n := \frac{n\mathbf{u}}{1 - n\mathbf{u}} \quad \text{for } n \in \mathbb{N}.$$

A forward error bound is

$$|a_1 a_2 \cdots a_n - \mathtt{res}| \le \gamma_{n-1} |a_1 a_2 \cdots a_n|$$

A validated error bound is

$$|a_1 a_2 \cdots a_n - \mathtt{res}| \le \mathrm{fl}\left(\frac{\gamma_{n-1} |\mathtt{res}|}{1 - 2\mathbf{u}}\right)$$

EFT for the product (1/3)

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

Algorithm `TwoProduct` by Veltkamp and Dekker (1971)

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

Algorithm 9 (Error-free split of a floating point number into two parts)

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

EFT for the product (2/3)

Algorithm 10 (EFT of the product of 2 floating point numbers)

function $[x, y] = \texttt{TwoProduct}(a, b)$

$x = \mathrm{fl}(a \cdot b)$

$[a_1, a_2] = \texttt{Split}(a)$

$[b_1, b_2] = \texttt{Split}(b)$

$y = \mathrm{fl}(a_2 \cdot b_2 - (((x - a_1 \cdot b_1) - a_2 \cdot b_1) - a_1 \cdot b_2))$

Theorem 3

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

$$a \cdot b = x + y, \quad x = \mathrm{fl}(a \cdot b), \quad |y| \le \mathbf{u}|x|, \quad |y| \le \mathbf{u}|a \cdot b|,$$

The algorithm `TwoProduct` *requires* 17 *flops.*

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 11 (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.

Compensated method for computing product

Algorithm 12 (Product evaluation with a compensated scheme (Graillat 2008))

function $\texttt{res} = \texttt{CompProd}(a)$
  $p_1 = a_1$
  $e_1 = 0$
  for $i = 2 : n$
    $[p_i, \pi_i] = \texttt{TwoProduct}(p_{i-1}, a_i)$
    $e_i = \mathrm{fl}(e_{i-1} a_i + \pi_i)$
  end
  $\texttt{res} = \mathrm{fl}(p_n + e_n)$

This algorithm requires $19n - 18$ flops

Error analysis

Theorem 4 (Graillat 2008)

Suppose Algorithm `CompProd` *is applied to floating point number $a_i \in \mathbb{F}$, $1 \le i \le n$, and set $p = \prod_{i=1}^{n} a_i$. Then,*

$$|\mathtt{res} - p| \le \mathbf{u}|p| + \gamma_n \gamma_{2n} |p|$$

Condition number of the product evaluation:

$$\mathrm{cond}(a) = \lim_{\varepsilon \to 0} \sup \left\{ \frac{|(a_1 + \Delta a_1) \cdots (a_n + \Delta a_n) - a_1 \cdots a_n|}{\varepsilon |a_1 a_2 \cdots a_n|} : |\Delta a_i| \le \varepsilon |a_i| \right\}$$

A standard computation yields

$$\mathrm{cond}(a) = n$$

Validated error bound

Lemma 3 (Graillat 2008)

Suppose Algorithm `CompProd` *is applied to floating point numbers* $a_i \in \mathbb{F}$, $1 \le i \le n$ *and set* $p = \prod_{i=1}^{n} a_i$*. Then, the absolute forward error affecting the product is bounded according to*

$$|\mathtt{res} - p| \le \mathrm{fl}\left(\left(\mathbf{u}|\mathtt{res}| + \frac{\gamma_n \gamma_{2n} |a_1 a_2 \cdots a_n|}{1-(n+3)\mathbf{u}}\right) / \ (1-2\mathbf{u})\right).$$

Faithful rounding

Let $\texttt{res} = \texttt{CompProd}(p)$

Lemma 4

If $n < \dfrac{\sqrt{1-\mathbf{u}}}{\sqrt{2}\sqrt{2+\mathbf{u}}+2\sqrt{(1-\mathbf{u})\mathbf{u}}}\mathbf{u}^{-1/2}$ *then* $\texttt{res}$ *is a faithful rounding of* p*.*

If $n < \alpha\mathbf{u}^{-1/2}$ where $\alpha \approx 1/2$ then the result is faithfully rounded

In double precision where $\mathbf{u} = 2^{-53}$, if $n < 2^{25} \approx 5 \cdot 10^7$, we get a faithfully rounded result

Validated error bound and faithful rounding

If

$$\mathrm{fl}\left(2\frac{\gamma_n\gamma_{2n}|a_1a_2\cdots a_n|}{1-(n+3)\mathbf{u}}\right) < \mathrm{fl}(\mathbf{u}|\mathtt{res}|)$$

then we got a faitfully rounded result. This makes it possible to check *a posteriori* if the result is faithfully rounded.

Various results

- Similar results apply for other compensated algorithm for dot product or Horner scheme with nonnegative entries
- Compensated dot product : computing $x^T y$

Algorithm 13 (Ogita, Rump and Oishi 2005)

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
  $\texttt{res} = \mathrm{fl}(p + s)$

Compensated dot product

Algorithm 14 (Ogita, Rump and Oishi 2005)

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

Then one has (Ogita, Rump, Oishi 2005)

$$|\texttt{res} - x^T y| \le \mathbf{u}|x^T y| + \gamma_{2n}^2 |x|^T |y|$$

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

The Horner scheme

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

Algorithm 15 (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 π_i
    $s_i = \mathrm{fl}(p_i + a_i)$ % rounding error σ_i
  end
  $\texttt{res} = s_0$

$$\frac{|p(x) - \texttt{Horner}(p,x)|}{|p(x)|} \leq \underbrace{\gamma_{2n}}_{\approx 2n\mathbf{u}} \mathrm{cond}(p,x) \text{ with } \mathrm{cond}(p,x) := \frac{\sum_{i=0}^{n} |a_i||x|^i}{|\sum_{i=0}^{n} a_i x^i|}$$

Error-free transformation for the Horner scheme

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

Algorithm 16 (Error-free transformation for the Horner scheme (Graillat,Louvet,Langlois 2005))

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

$s_n = a_n$

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

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

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

  Let π_i be the coefficient of degree i of p_π

  Let σ_i be the coefficient of degree i of p_σ

end

$\texttt{Horner}(p,x) = s_0$

Compensated Horner scheme and its accuracy

Algorithm 17 (Compensated Horner scheme)

function $\texttt{res} = \texttt{CompHorner}(p, x)$
$[h, p_\pi, p_\sigma] = \texttt{EFTHorner}(p, x)$
$c = \texttt{Horner}(p_\pi + p_\sigma, x)$
$\texttt{res} = \mathrm{fl}(h + c)$

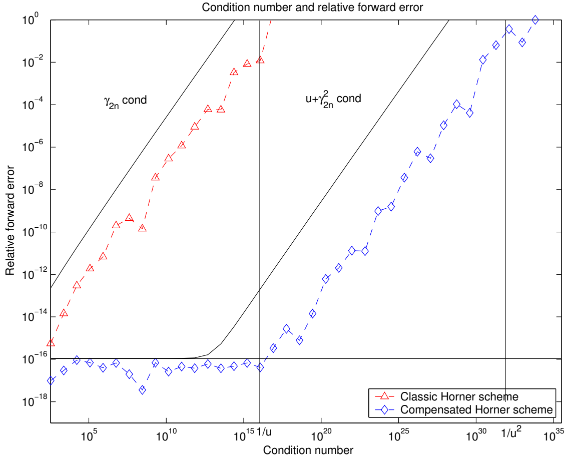
Theorem 5 (Graillat,Louvet,Langlois 2005)

Let p be a polynomial of degree n with floating point coefficients, and x be a floating point value. Then if no underflow occurs,

$$\frac{|\texttt{CompHorner}(p,x) - p(x)|}{|p(x)|} \leq \mathbf{u} + \underbrace{\gamma_{2n}^2}_{\approx 4n^2\mathbf{u}^2} \operatorname{cond}(p,x).$$

Numerical experiments: testing the accuracy

Evaluation of $p_n(x) = (x-1)^n$ for $x = \mathrm{fl}(1.333)$ and $n = 3, \ldots, 42$



Conclusion and future work

Conclusion

- Compensated algorithms make it possible to accurately compute with nonnegative entries
- It makes it possible to compute some accurate error bounds

Future work

- Computing accurately the 2-norm of a vector

Thank you for your attention