Correctly rounded multiplication by arbitrary precision constants

J.-M. Muller
Arénaire, LIP, É.N.S. Lyon
N. Brisebarre
LArAl, Univ. St-Étienne et Arénaire, LIP, É.N.S. Lyon

Arith'17
June 27, 2005

CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

INRIA
RHÔNE-ALPES

Multiplications by constants

Many numerical algorithms : multiplications by constants that are not exactly representable in floating-point (FP) arithmetic.

Typical constants that are used : π, $1/\pi$, $\ln(2)$, e, $B_k/k!$ (Euler-McLaurin summation), $\cos(k\pi/N)$ and $\sin(k\pi/N)$ (Fast Fourier Transforms). Some numerical integration formulas such as :

$$\int_{x_0}^{x_1} f(x)\mathrm{d}x \approx h\left(\frac{55}{24}f(x_1) - \frac{59}{24}f(x_2) + \frac{37}{24}f(x_3) - \frac{9}{24}f(x_4)\right)$$

also naturally involve multiplications by constants.

Correctly rounded Multiplications by constants

For approximating Cx, where C is an infinite-precision constant and x is a FP number, desirable result $= \circ(Cx)$, where $\circ(u)$ is u rounded to the nearest FP number.

Our goal : We want to compute at low cost $\circ(Cx)$ for all input FP numbers x (provided no overflow or underflow occur).

Naive idea : let C_h be the FP number that is closest to C, we actually compute $\circ(C_h x)$. The obtained result is frequently different from $\circ(Cx)$.

Some statistics

Let $n =$ number of mantissa bits of the binary FP format.

Comparison of $\circ(C_h x)$ and $\circ(Cx)$ for all possible values of the mantissa of x.

n	Proportion of correctly rounded results
4	0.62500
5	0.93750
6	0.78125
7	0.59375
$\cdots$	$\cdots$
16	0.86765
17	0.73558
$\cdots$	$\cdots$
24	0.66805

TAB. 1: *Proportion of input values x for which $\circ(C_h x) = \circ(Cx)$ for $C = \pi$ and various values of the number n of mantissa bits.*

Correctly rounded Multiplications by constants

Our goal – at least for some constants and some FP formats – is to return $\circ(Cx)$ for all input FP numbers x (provided no overflow or underflow occur), and at a low cost.

To do that, we will use *fused multiply and add* (`fma`) instructions.

`fma` : computes correct rounding of $ab + c$ where a, b and c are FP numbers.

We assume binary FP arithmetic.

The algorithm

- We want Cx with correct rounding (assuming rounding to nearest even).
- C is not an FP number.
- We assume that a `fma` instruction is available. Operands stored in a binary FP format with n-bit mantissas.
- We assume that the two following FP numbers are pre-computed :

$$\left\{ \begin{array}{rcl} C_h & = & \circ(C), \\ C_\ell & = & \circ(C - C_h), \end{array} \right.$$

where $\circ(t)$ stands for t rounded to the nearest FP number.

Algorithm. ***(Multiplication by C with a multiplication and a `fma`).*** *From x, compute*

$$\left\{ \begin{array}{rcl} u_1 & = & \circ(C_\ell x), \\ u_2 & = & \circ(C_h x + u_1). \end{array} \right.$$

The result to be returned is u_2.

Algorithm. ***(Multiplication by C with a multiplication and a `fma`).*** *From x, compute*

$$\begin{cases} u_1 & = & \circ(C_\ell x), \\ u_2 & = & \circ(C_h x + u_1). \end{cases}$$

The result to be returned is u_2.

Without l.o.g., we assume that $1 < x < 2$ and $1 < C < 2$, that C is not exactly representable, and that $C - C_h$ is not a power of 2.

Warning ! There exist C and x s.t. $u_2 \neq \circ(Cx)$.

We give 3 methods for checking if $\forall x, u_2 = \circ(Cx)$.

Algorithm. ***(Multiplication by C with a multiplication and a `fma`).*** *From x, compute*

$$\left\{\begin{array}{lcl} u_1 & = & \circ(C_\ell x), \\ u_2 & = & \circ(C_h x + u_1). \end{array}\right.$$

The result to be returned is u_2.

3 methods for checking if $\forall x, u_2 = \circ(Cx)$.

Methods 1 and 2 are simple but do not always give a complete answer :

- they either certify that our algorithm always returns a correctly rounded result,
- or give a “bad case”, i.e. an FP number x s.t. $u_2 \neq \circ(Cx)$.

Method 3 is a bit more complicated but gives a complete answer :

- it gives all “bad cases”,
- or certify that there are none, i.e. that our algorithm always gives the correct result.

Analyzing the algorithm

We will use the following property, that bounds the maximum possible distance between u_2 and Cx in the algorithm.

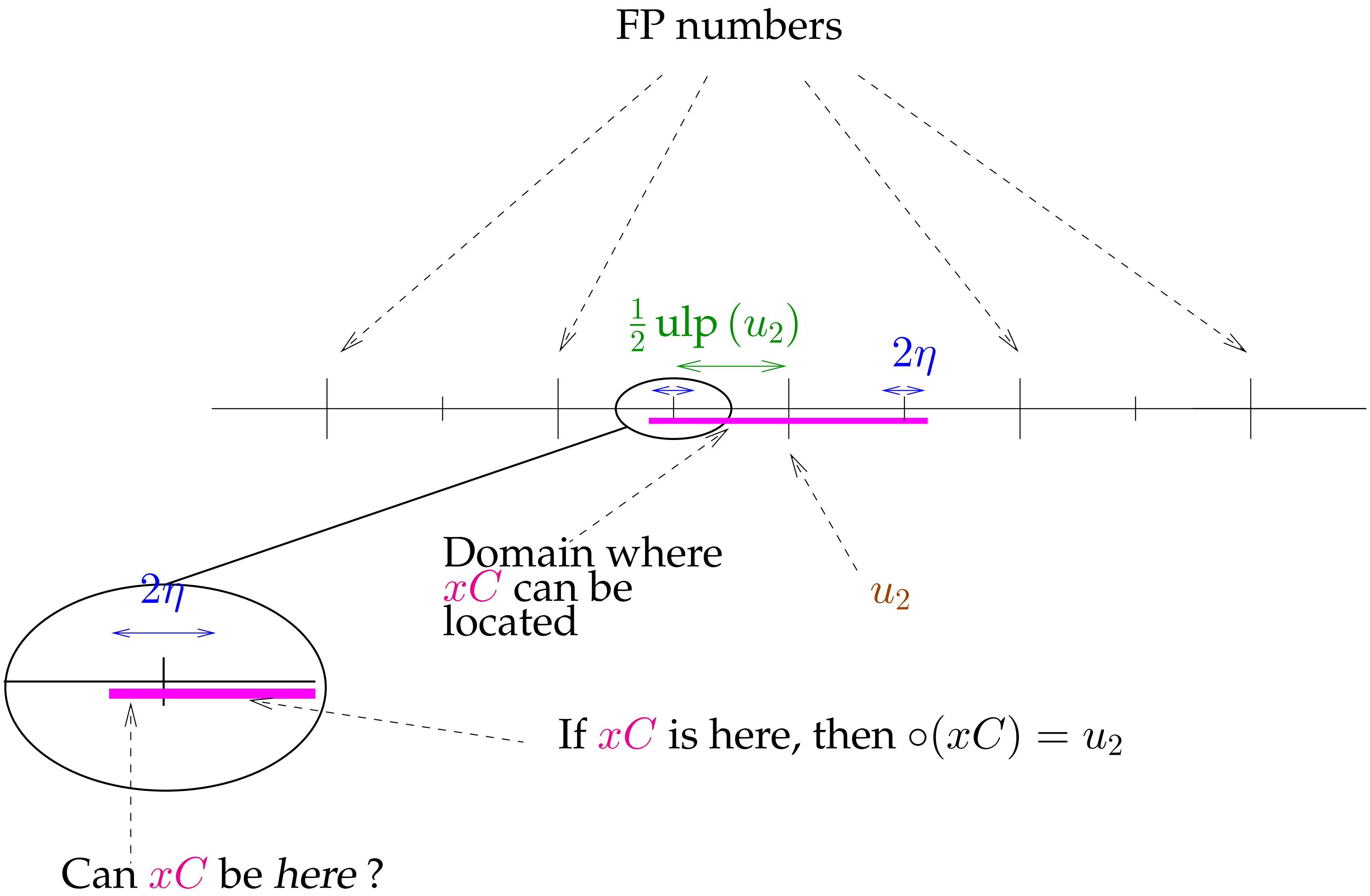
Property 1.
For all FP number x, we have

$$|u_2 - Cx| < \frac{1}{2}\,\mathit{ulp}\,(u_2) + 2\,\mathit{ulp}\,(C_\ell).$$

[Remember that $C_h = \circ(C)$, $C_\ell{=}\circ(C - C_h)$, $u_1 = \circ(C_\ell x)$, $u_2 = \circ(C_h x + u_1)$.]

Analyzing the algorithm

Recall : we have $|u_2 - Cx| < 1/2\,\text{ulp}\,(u_2) + \eta$ with $\eta := 2\,\text{ulp}\,(C_\ell)$.



Analyzing the algorithm

Remark . *We know that* xC *is within* $1/2\,\text{ulp}\,(u_2) + 2\,\text{ulp}\,(C_\ell)$ *from the FP number* u_2*. If we prove that* xC *cannot be at a distance* $\leq 2\,\text{ulp}\,(C_\ell)$ *from the middle of two consecutive FP numbers, then* u_2 *will be the FP number that is closest to* xC*.*

A reminder on continued fractions

Let $\beta \in \mathbb{R}$. From β, two sequences (a_i) and (r_i) defined by :

$$\begin{cases} r_0 & = & \beta, \\ a_i & = & \lfloor r_i \rfloor, \\ r_{i+1} & = & 1/(r_i - a_i). \end{cases}$$

If $\beta \notin \mathbb{Q}$, these sequences are defined $\forall i$, and the rational number

$$\frac{p_i}{q_i} = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\ddots + \cfrac{1}{a_i}}}}}$$

is the ith *convergent* to β. If $\beta \in \mathbb{Q}$, these sequences terminate for some i, and $p_i/q_i = \beta$ exactly.

We will use the following two results :

Theorem 2. *Let* $(p_j/q_j)_{j\geq 1}$ *be the convergents of* β*. For any* (p,q)*, with* $0 \leq q < q_{n+1}$*, we have*

$$|p - \beta q| \geq |p_n - \beta q_n|.$$

Theorem 3. *Let* p, q *be nonzero integers, with* $\gcd(p,q) = 1$*. If*

$$\left|\frac{p}{q} - \beta\right| < \frac{1}{2q^2}$$

then p/q *is a convergent of* β*.*

Method 3

Assume $x > x_{\text{cut}} := 2/C$ (the case $x < x_{\text{cut}} = 2/C$ is similar).

Let $X_{\text{cut}} := \lfloor 2^{n-1} x_{\text{cut}} \rfloor$.

We recall the notations : $C_h = \circ(C), C_\ell = \circ(C - C_h)$, $u_1 = \circ(C_\ell x)$, $u_2 = \circ(C_h x + u_1)$.

We want to determine the integers $X,\ X_{\text{cut}} \leq X \leq 2^n - 1$ that satisfy

$$\left| u_2 - C \frac{X}{2^{n-1}} \right| < \frac{1}{2} \, \text{ulp}\,(u_2) + 2\, \text{ulp}\,(C_\ell),$$

or equivalently, the integers $X,\ X_{\text{cut}} \leq X \leq 2^n - 1$ s.t. there exists an integer A with

$$\left| C \frac{X}{2^{n-1}} - \frac{2A+1}{2^{n-1}} \right| \leq 2\, \text{ulp}\,(C_\ell).$$

Once we know the X candidate, we compute u_2 and $\circ(Cx)$ to check if they coincide or not.

Method 3

We search for the $x = X/2^{n-1},\ X_{\text{cut}} \le X \le 2^n - 1$ s.t. there exits an integer A with

$$\left| C\frac{X}{2^{n-1}} - \frac{2A+1}{2^{n-1}} \right| \le 2\,\text{ulp}\,(C_\ell).$$

We know that $\text{ulp}\,(C_\ell) \le 2^{-2n}$.

We distinguish the cases $\text{ulp}\,(C_\ell) \le 2^{-2n-1}$ and $\text{ulp}\,(C_\ell) = 2^{-2n}$.

Method 3

First, we assume ulp $(C_\ell) \leq 2^{-2n-1}$.

In that case, the integers $x = X/2^{n-1},\ X_{\text{cut}} \leq X \leq 2^n - 1$ satisfy

$$\left| 2C - \frac{2A+1}{X} \right| < \frac{1}{2X^2} :$$

$(2A+1)/X$ is a convergent of $2C$ from Theorem 3. It suffices then to check the convergents of $2C$ of denominator less or equal to $2^n - 1$.

Method 3

Now, assume $\text{ulp}\,(C_\ell) = 2^{-2n}$.

Careful computations lead to the following problem : determine the $X,\ X_{\text{cut}} \leq X \leq 2^n - 1$ s.t.

$$\{X(C_h + C_\ell) + \frac{1}{2^{n+1}}\} \leq \frac{1}{2^n},$$

where $\{y\}$ is the fractional part of y : $\{y\} = y - \lfloor y \rfloor$.

We use an efficient algorithm due to V. Lefèvre to determine all the integers $X, X_{\text{cut}} \leq X \leq 2^n - 1$ solution of this inequality.

Two other methods

- See the paper for details.

- Methods 1 and 2 are simpler : they each give a criterion, easy to check, that guarantee that the algorithm always returns a correctly rounded result. They also may give some values of x such that $u_2 \neq \circ(Cx)$.

- Method 1 uses Theorem 2, Method 2 uses Theorem 3. We may need the examination of all convergents to $2C$ or C.

Two examples

Method 1 allows to prove

Theorem 4. [Correctly rounded multiplication by π] *The algorithm always returns a correctly rounded result in double precision with* $C = 2^j\pi$*, where* j *is any integer, provided no under/overflow occur.*

With $\ln(2)$, needs more work (uses Method 2 and examination of all convergents)

Theorem 5. [Correctly rounded multiplication by $\ln(2)$] *The algorithm always returns a correctly rounded result in double precision with* $C = 2^j \ln(2)$*, where* j *is any integer, provided no under/overflow occur.*

Example 3 : multiplication by $1/\pi$ in double precision

Consider the case $C = 4/\pi$ and $n = 53$, and assume we use Method 1. We find a counterexample : $x = 6081371451248382 \times 2^{\pm k}$.

Method 3 certifies that $x = 6081371451248382 \times 2^{\pm k}$ are the *only* FP values for which our algorithm fails.

Implementation

We have written Maple programs that implement Methods 1, 2 and 3, and a GP/PARI program that implements Method 3.

These programs can be downloaded from the url

`http://perso.ens-lyon.fr/jean-michel.muller/MultConstant.html`

Some results

C	n	Method 1	Method 2	Method 3
π	8	Does not work for 226	Does not work for 226	AW (c) unless $X =$ 226
π	24	unable	unable	AW
π	53	AW	unable	AW
π	64	unable	AW	AW (c)
π	113	AW	AW	AW (c)

TAB. 2: *Some results obtained using Methods 1, 2 and 3. The results given for constant C hold for all values $2^{\pm j}C$. “AW” means “always works” and “unable” means “the method is unable to conclude”. For Method 3, “(c)” means that we have needed to check the convergents.*

C	n	Method 1	Method 2	Method 3
$1/\pi$	24	unable	unable	AW
$1/\pi$	53	Does not work for 6081371451248382	unable	AW unless $X =$ 6081371451248382
$1/\pi$	64	AW	AW	AW (c)
$1/\pi$	113	unable	unable	AW
$\ln 2$	24	AW	AW	AW (c)
$\ln 2$	53	AW	unable	AW (c)
$\ln 2$	64	AW	unable	AW (c)
$\ln 2$	113	AW	AW	AW (c)

TAB. 3: *Some results obtained using Methods 1, 2 and 3. The results given for constant C hold for all values $2^{\pm j}C$. "AW" means "always works" and "unable" means "the method is unable to conclude". For Method 3, "(c)" means that we have needed to check the convergents.*

C	n	Method 1	Method 2	Method 3
$\frac{1}{\ln 2}$	24	unable	AW	AW (c)
$\frac{1}{\ln 2}$	53	AW	AW	AW (c)
$\frac{1}{\ln 2}$	64	unable	unable	AW
$\frac{1}{\ln 2}$	113	unable	unable	AW
$\ln 10$	24	unable	AW	AW (c)
$\ln 10$	53	unable	unable	AW
$\ln 10$	64	unable	AW	AW (c)
$\ln 10$	113	AW	AW	AW (c)

TAB. 4: *Some results obtained using Methods 1, 2 and 3. The results given for constant C hold for all values $2^{\pm j}C$. "AW" means "always works" and "unable" means "the method is unable to conclude". For Method 3, "(c)" means that we have needed to check the convergents.*

C	n	Method 1	Method 2	Method 3
$\frac{2^j}{\ln 10}$	24	unable	unable	AW
$\frac{2^j}{\ln 10}$	53	unable	AW	AW (c)
$\frac{2^j}{\ln 10}$	64	unable	AW	AW (c)
$\frac{2^j}{\ln 10}$	113	unable	unable	AW
$\cos\frac{\pi}{8}$	24	unable	unable	AW
$\cos\frac{\pi}{8}$	53	AW	AW	AW (c)
$\cos\frac{\pi}{8}$	64	AW	unable	AW
$\cos\frac{\pi}{8}$	113	unable	AW	AW (c)

TAB. 5: *Some results obtained using Methods 1, 2 and 3. The results given for constant C hold for all values $2^{\pm j}C$. "AW" means "always works" and "unable" means "the method is unable to conclude". For Method 3, "(c)" means that we have needed to check the convergents.*

Conclusion

The three methods we have proposed allow to check whether correctly rounded multiplication by an “infinite precision” constant C is feasible at a low cost (one multiplication and one `fma`).