Error in ulps of the multiplication or division by a correctly-rounded function or constant in binary floating-point arithmetic

N. Brisebarre  J.-M. Muller  J. Picot
CNRS and ENS de Lyon, Laboratoire LIP
Lyon, France

30th IEEE Symposium on Computer Arithmetic
Portland, OR, September 2023
Goal: Tight (optimal or near optimal) error bounds in ulps for many usual functions:

\[ x \times \pi, \quad \ln(2)/x, \quad x/(y + z), \quad (x + y) \times z, \quad x/\sqrt{y}, \]
\[ \sqrt{x}/y, \quad (x + y)(z + t), \quad (x + y)/(z + t), \quad (x + y)/(zt), \]
\[ (ax + b)/(cy + d), \quad (x \times y)/\sqrt{z}, \quad \text{etc.} \]

Context:

- radix-2, precision-\( p \) floating-point arithmetic, assuming round to nearest (any tie-breaking rule in the proofs, ties-to-even in the examples);

- a FP number is zero or a number of the form \( x = M_x \cdot 2^{e_x - p + 1} \), where \( M_x, e_x \in \mathbb{Z} \), with \( 2^{p-1} \leq |M_x| \leq 2^p - 1 \) (we assume no underflow or overflow);

- rounding function RN:

  \[
  \text{program line } z = x + y \Rightarrow \text{obtained result } z = \text{RN}(x + y).
  \]
Link between all these functions?

\[ x \cdot \pi, \; \ln(2)/x, \; x/(y + z), \; (x + y) \cdot z, \; x/\sqrt{y}, \]
\[ \sqrt{x}/y, \; (x + y)(z + t), \; (x + y)/(z + t), \; (x + y)/(zt), \]
\[ (ax + b)/(cy + d), \; (x \cdot y)/\sqrt{z}, \text{ etc.} \]

They are of the form

\[ x \cdot c, \; x/c, \; c/x, \; m \cdot n, \; \text{or } n/d, \]

where

- \( x \) is a FPnumber, and
- \( c, n, m \) and \( d \) are either real constants or correctly-rounded functions of one or more variables.

**Examples:** \( c = \pi \), or \( c = \sqrt{y} \) where \( y \) is a FP number and \( \sqrt{y} \) is obtained through the (correctly rounded) \( \sqrt{\cdot} \) instruction, or \( c = y + z \) obtained through FPADD.
program line

\[ t = (x \times y) / \sqrt{z} \]

real function

\[ t = \frac{x \cdot y}{\sqrt{z}} \]

computed result

\[ \hat{t} = \text{RN} \left( \frac{\text{RN}(x \cdot y)}{\text{RN} (\sqrt{z})} \right) \]

We show that:

\[ |t - \hat{t}| \leq \frac{5}{2} \text{ulp}(t). \]

Very tight: \(2.4994 \text{ ulp}(t)\) attained in binary64 arithmetic.
Error in ulps vs. relative error

- numerical errors usually expressed as error in ulps or as relative errors.
- \(\text{ulp}(t)\) (unit in the last place of \(t\)) is \(2^{\lfloor \log_2 |t| \rfloor - p + 1}\),
- if \(t \neq 0\) is the exact result and \(\hat{t}\) is the computed approximation:
  - the relative error is \(\frac{|t - \hat{t}|}{|t|}\),
  - the error in ulps is \(\frac{|t - \hat{t}|}{\text{ulp}(t)}\).
Error in ulps vs. relative error

- **ulps** preferred for “atomic” calculations (they convey more information: correct rounding almost equivalent to error $\leq 0.5$ ulp);

- **relative errors** easier to manipulate for “large” calculations (e.g. from relative error on $f$ and $g$, obtaining relative error on $f \times g$ is straightforward);

- easy conversion between both but at the cost of information loss:
  - define $u = 2^{-p}$ (unit roundoff);
  - we approximate an exact result $t$ by a computed result $\hat{t}$:

\[
\text{error} \leq \alpha \text{ulp}(t) \quad \Rightarrow \quad \text{relative error} \leq 2\alpha u \quad \Rightarrow \quad \text{error} \leq 2\alpha \text{ulp}(t).
\]

→ we have lost a factor 2 in the round trip conversion.
The FP numbers between $1/2$ and $8$ in the toy system $p = 3$

\[ \frac{1}{2} \quad 1 \quad 2 \quad x \quad 4 \quad y \quad 8 \]

- $2u$
- $\text{ulp}(x) = 4u$
- $\text{RN}(x)$
- $\text{RN}(y)$ (assuming ties-to-even)
Multiplication of a FP number by a constant or a correctly-rounded function

Error bound in ulps on the computation of $x \cdot c$, where

- $x$ is a FP number (assumed exact!), and
- $c$ is a real constant or a correctly-rounded function (can be $\sqrt{y}$, $\pi$, $y + z$, $y \cdot z$, etc.).

We want to bound the error of approximating $x \cdot c$ by

$$\text{RN}(x \cdot \hat{c}),$$

where $\hat{c} = \text{RN}(c)$. Here, we consider “general” bounds, applicable to any $c$.

[In the TETC paper we also try to improve these bounds in the particular case where $c$ is a constant.]
Multiplication of a FP number by a constant or a correctly-rounded function

Property 1

Barring underflow and overflow, the FP number \( s = \RN(\hat{c} \cdot x) \) satisfies

\[
|s - cx| \leq \left( \frac{3}{2} - u \right) \cdot \text{ulp}(cx) < \frac{3}{2} \text{ulp}(cx).
\]

In the general case (arbitrary constant \( c \)) the bound is asymptotically optimal. Shown with the following generic example (assuming \( \RN \) breaks ties to even):

If \( p \) is even, choose

\[
x = 2^p - 2^{p/2}, \\
c = 1 + 2^{-p/2-1} - 2^{-p},
\]

If \( p \) is odd, choose

\[
x = 2^p - 2^{(p-1)/2}, \\
c = 1 + 2^{-(p+1)/2} - 2^{-p}.
\]
Some examples

▶ previous example: \( c \) can be expressed as sum of two FPNs
→ asymptotic optimality of the bound \( 1.5 \text{ ulp} \) of Property 1 for the calculation of \( z^* (x + y) \);

▶ errors

\[
1.499756 \ldots \text{ulp}(cx) \quad (\text{binary32 arithmetic}),
1.499999992549 \ldots \text{ulp}(cx) \quad (\text{binary64 arithmetic})
1.499999999999999993061 \ldots \text{ulp}(cx) \quad (\text{binary128 arithmetic})
\]

can be attained when calculating \( z^* (x \ast y) \), showing that for that function, the bound is very tight;

▶ in binary64 arithmetic, with \( x = 9007197761440759 \) and \( y = 4503599630388691/2^{52} \) error when computing \( x\sqrt{y} \) is \( 1.4991 \text{ ulp}(x\sqrt{y}) \).
We approximate $x/c$, where $x$ is a FP number and $c$ is either a real constant or a real function of one or more FP variables, by

$$s = \text{RN}(x/\hat{c}),$$

where, as previously, $\hat{c} = \text{RN}(c)$.

**Property 2**

*Barring underflow and overflow, the FP number $s = \text{RN}(x/\hat{c})$ satisfies*

$$\left| s - \frac{x}{c} \right| \leq \left( \frac{3}{2} - \frac{2u}{1 + 2u} \right) \text{ulp} \left( \frac{x}{c} \right)$$

$$\leq \left( \frac{3}{2} - 2u + 4u^2 \right) \text{ulp} \left( \frac{x}{c} \right)$$

$$< \frac{3}{2} \text{ulp} \left( \frac{x}{c} \right).$$

As for the product, “generic” example for a general constant $c$ that shows asymptotic optimality.
Tightness?

- asymptotic optimality of the bound $1.5\,\text{ulp}$ for the calculation of $z/(x + y)$.

- errors

  \[1.49957 \cdot \cdot \cdot \text{ulp}(x/c)\] (binary32),

  \[1.49999998137 \cdot \cdot \cdot \text{ulp}(x/c)\] (binary64), can be

  \[1.499999999999998265 \cdot \cdot \cdot \text{ulp}(x/c)\] (binary128)

attained when calculating $z/(x \times y)$, showing that for that function, the bound is very tight;

- binary64 arithmetic, error $1.49906\,\text{ulp}(x/\sqrt{y})$ attained for $x = 9007198105271337$ and $y = 4503599631275935/2^{52}$ when calculating $x/\sqrt{y}$.
Dividing a correctly-rounded function by a FP number

Now we consider approximating \( c/x \), where \( x \) is a FP number and \( c \) is either a real constant or a real function of one or more FP variables, by

\[
s = \text{RN}(\hat{c}/x),
\]

where, as previously, \( \hat{c} = \text{RN}(c) \).

Property 3

*Barring underflow and overflow, the FP number \( s = \text{RN}(\hat{c}/x) \) satisfies*

\[
|s - \frac{c}{x}| \leq \frac{3 + 2u}{2 + 4u} \cdot \text{ulp} \left( \frac{c}{x} \right)
\]

\[
\leq \left( \frac{3}{2} - 2u + 4u^2 \right) \text{ulp} \left( \frac{c}{x} \right).
\]

Similar examples of asymptotic optimality or tightness. Covers functions such as \( \ln(2)/x \), \( \sqrt{x}/y \), \( (x + y)/z \), ...
Product of two correctly-rounded functions

Approximation of $m \cdot n$, where $m$ and $n$ are either real constants or correctly-rounded functions, by

$$s = \text{RN} (\hat{m} \cdot \hat{n}) ,$$

where $\hat{m} = \text{RN}(m)$ and $\hat{n} = \text{RN}(n)$ (of course nobody multiplies 2 constants)

**Property 4**

*Barring underflow and overflow, the FP number $s = \text{RN}(\hat{m} \cdot \hat{m})$ satisfies*

$$|s - mn| \leq \left( \frac{5}{2} + \frac{u}{2} \right) \text{ulp} (mn) .$$

In the general case, the bound is asymptotically optimal for even values of $p$ (it probably is for odd values too but no proof).
error $2.4999982\ u\text{ulp}(efgh)$ is attained when computing
$(e \ast f) \ast (g \ast h)$ in binary64/double-precision arithmetic,

the property applies to calculations such as $\pi \cdot \sqrt{x}$, $(x + y) \cdot (z + t)$,
$(x \cdot y) \cdot \sqrt{z}$, $e^x \cos(y)$ (with correctly rounded functions), etc. If an
FMA instruction is available, it also covers computations of the form

$$(ax + b)(cy + d),$$

where $a$, $b$, $c$, $d$, $x$, and $y$ are FP numbers.
Approximation of $n/d$, where $n$ and $d$ are either real constants or correctly-rounded functions, by

$$s = \text{RN} \left( \frac{\hat{n}}{\hat{d}} \right),$$

where $\hat{n} = \text{RN}(n)$ and $\hat{d} = \text{RN}(d)$.

**Property 5**

*Barring underflow and overflow, the floating-point number $s = \text{RN}(\hat{n}/\hat{d})$ satisfies*

$$\left| s - \frac{n}{d} \right| \leq \frac{5}{2} \text{ulp} \left( \frac{n}{d} \right).$$

covers calculations such as $\pi/\sqrt{x}$, $(x + y)/(z + t)$, $(xy)/(z + t)$, etc. If an FMA instruction is available, it also covers computations of the form

$$\frac{ax + b}{cy + d},$$

where $a$, $b$, $c$, $d$, $x$, and $y$ are FP numbers.
Tightness?

- binary64, error $2.49999997392 \cdots \text{ulp}$ attained when computing $(x + y)/(z + t)$, $(xy)/(z + t)$; $(x + y)/(zt)$, and $(xy)/(zt)$ with well chosen values (see TETC paper);

- binary64, error $2.4994 \text{ulp}$ attained when computing

  $\frac{x + y}{\sqrt{z}}$

  or

  $\frac{xy}{\sqrt{z}}$

  with well chosen values.
Conclusion

- sharp error bounds in ulps for computations in binary FP arithmetic of the form $x \cdot c$, $x/c$, $c/x$, $m \cdot n$ and $n/d$, where $x$ is a FP number and $c$, $n$, $m$ and $d$ are either real constants or correctly-rounded functions of one or more variables;

- examples of functions for which our work gives tight bounds are $x \cdot \pi$, $\ln(2)/x$, $x/(y+z)$, $(x+y)\cdot z$, $x/\sqrt{y}$, $\sqrt{x}/y$, $(x+y)\cdot (z+t)$, $(x+y)/(z+t)$, $(x+y)/(zt)$, $(ax+b)/(cy+d)$, $(x \cdot y) \cdot \sqrt{z}$, etc.

- In several cases, we have been able to show that our bounds are asymptotically optimal.

Thank you!