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

Nicolas Brisebarre, Jean-Michel Muller, *Fellow, IEEE,* Joris Picot

***Abstract*—Assume we use a binary floating-point arithmetic and that RN is the round-to-nearest function. Also assume that c is a constant or a real function of one or more variables, and that we have at our disposal a correctly rounded implementation of c, say $\hat{c} = \mathrm{RN}(c)$. For evaluating $x \cdot c$ (resp. x/c or c/x), the natural way is to replace it by $\mathrm{RN}(x \cdot \hat{c})$ (resp. $\mathrm{RN}(x/\hat{c})$ or $\mathrm{RN}(\hat{c}/x)$), that is, to call function $\hat{c}$ and to perform a floating-point multiplication or division. This can be generalized to the approximation of n/d by $\mathrm{RN}(\hat{n}/\hat{d})$ and the approximation of $n \cdot d$ by $\mathrm{RN}(\hat{n} \cdot \hat{d})$, where $\hat{n} = \mathrm{RN}(n)$ and $\hat{d} = \mathrm{RN}(d)$, and n and d are functions for which we have at our disposal a correctly rounded implementation. We discuss tight error bounds in ulps of such approximations. From our results, one immediately obtains tight error bounds for calculations such as $\mathtt{x*pi}$, $\mathtt{ln(2)/x}$, $\mathtt{x/(y+z)}$, $\mathtt{(x+y)*z}$, $\mathtt{x/sqrt(y)}$, $\mathtt{sqrt(x)/y}$, $\mathtt{(x+y)(z+t)}$, $\mathtt{(x+y)/(z+t)}$, $\mathtt{(x+y)/(zt)}$, etc. in floating-point arithmetic.**

***Index Terms*—Floating-point arithmetic, ulp, numerical error, correct rounding, multiplication by a constant**

I. Introduction

A. Purpose of this paper and notation

MULTIPLYING a floating-point number by a real constant (such as π or $\ln(2)$), or multiplying or dividing it by a correctly rounded function of one or more variables (such as $\sqrt{x}$, $x+y$, or xy), or dividing a constant or correctly-rounded function by a floating-point number are very frequent operations in numerical computing. When last-bit accuracy is desired, one can use specifically-designed solutions (see for instance [1] for multiplication by a constant). However, here, we are interested by the error of the straightforward approach. For instance, when a programmer writes the statement

$$\mathtt{s = x * pi;}$$

he or she most probably wants to compute $x \cdot \pi$ as accurately as possible. However the variable pi is already a floating-point approximation of the real π, and, as a result, the error in this computation is larger than the mere approximation caused by the floating-point multiplication. We are also interested in calculations such as

$$\mathtt{s = x/sqrt(y);}$$

or

$$\mathtt{s = (x + y) * (z + t);}$$

A tight bound on the *relative error* of such operations is very easily obtained (see Section I-B). And yet, although relative errors are frequently easier to manipulate, for basic functions that can be regarded as "atomic", errors in ulps are frequently preferred, because they convey more information (for example, while correct rounding implies an error less than $0.5\,\mathrm{ulp}$ and a relative error less than u—see definition below—, the converse is *almost* true for the error in ulps,[1] and far from being true for the relative errors). The purpose of this paper is to give very tight bounds on the error in ulps on the multiplication or division of a floating-point number by a real constant or a correctly-rounded function, or on the product or quotient of two correctly-rounded functions. As is common in rounding error analysis of mathematical functions, we assume that the input floating-point numbers are exact (in practice, they often result from a previous calculation or from a measurement and therefore contain some error, but the impact of that error on the final result must be evaluated separately). See for instance [2].

In the following, we assume a binary, precision-p, floating-point (FP) arithmetic, with correctly rounded (to nearest) floating-point operations. A floating-point number is zero or a number of the form

$$x = M_x \cdot 2^{e_x - p + 1},$$

where M_x and e_x are integers, with $2^{p-1} \leqslant |M_x| \leqslant 2^p - 1$. Here, we do not assume bounds on the exponents e_x, which means that our results apply to real-life arithmetics such as the ones specified by the IEEE 754 Standard for Floating-Point arithmetic [3] whenever underflow and overflow do not occur.[2] In the following, if t is a real number then $\mathrm{RN}(t)$ is the floating-point number nearest to t (a tie-breaking choice is necessary if t is halfway between two consecutive floating-point numbers: our bounds are valid whatever the tie-breaking choice, provided that it satisfies $\mathrm{RN}(-x) = -\mathrm{RN}(x)$ and $\mathrm{RN}(2^k x) = 2^k\,\mathrm{RN}(x)$, however our examples use the ties-to-even rule, which is the default in IEEE-754 arithmetic). If $t \in \mathbb{R}, t \neq 0$ then

- $\mathrm{ulp}(t)$ (*unit in the last place* of t) is $2^{\lfloor \log_2 |t| \rfloor - p + 1}$, and
- $\mathrm{ufp}(t)$ (*unit in the first place* [4] *of* t) is $2^{\lfloor \log_2 |t| \rfloor}$.

Functions ufp and ulp are defined for *real numbers*, not only for FP numbers. This is why we cannot use the "exponent of t" in their definitions. The *unit round-off* [5] is the number

N. Brisebarre is with CNRS, Lab. LIP, ENS de Lyon, Lyon, France
J.-M. Muller is with CNRS, Lab. LIP, ENS de Lyon, Lyon, France
J. Picot is with ENS de Lyon, Lab. LIP, Lyon, France

[1] If we forget about tie-breaking rules, and if we are careful at powers of 2.

[2] In particular, we assume that there are no subnormal results.

$u = 2^{-p}$. Note that $\mathrm{ulp}(t)$ is the distance between two consecutive FP numbers in the neighborhood of t. If $1 \leqslant t < 2$ then $\mathrm{ulp}(t) = 2u$. In particular, the FP number preceding 2 is $2 - 2u$.

These notions are illustrated by Figure 1.

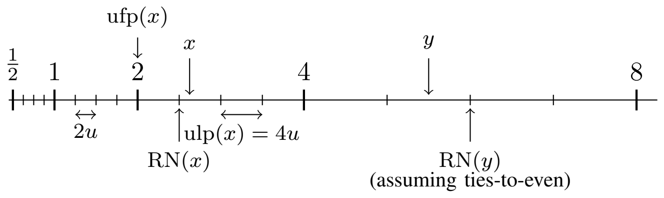

Fig. 1. The FP numbers between $1/2$ and 8 in the toy system $p = 3$ (*i.e.*, $u = 1/8$).

In the following, c is either a real constant or a real function of one or more floating-point variables. We assume that $\hat{c} = \mathrm{RN}(c)$ is available. If c is a constant, $\hat{c}$ is just the floating-point number nearest to c. If c is an arithmetic function (e.g., $c(x, y) = x + y$ or $c(x, y) = xy$), or the square root, then $\hat{c}$ is provided by the underlying IEEE-754 arithmetic. If c is a more complex function, such as $\exp(x)$, $\ln(x)$, etc., $\hat{c}$ can be provided by a correctly-rounded function such as the ones included in the CORE-MATH library [6].

We first obtain tight bounds on the error, expressed in ulps, of approximating the product $c \cdot x$ (resp. the quotient x/c or the quotient c/x), where x is a floating-point number, by

$$s = \mathrm{RN}(\hat{c} \cdot x) \quad (\text{resp. } \mathrm{RN}(x/\hat{c}) \text{ or } \mathrm{RN}(\hat{c}/x)).$$

This is done in Section II for $x \cdot c$, in Section III for x/c, and in Section IV for c/x.

When c is a constant, we assume that it is not a floating-point number (otherwise, the well-known bound 0.5 ulp applies and is optimal). Our results apply for instance to the computation of

$$x \cdot \cos\left(\frac{2k\pi}{N}\right),$$

that appears in discrete cosine transforms, assuming that either the values $\mathrm{RN}(\cos(2k\pi/N))$ are precomputed and stored, or that N is a power of 2, k is less than 2^p and a correctly-rounded function `cospi` is available (there is one for instance in the CORE-MATH library). Our results also apply to the calculation of expressions of the form

$$(x+y) \cdot z,\ z/(x+y),\ (x+y)/z,\ (x \cdot y) \cdot z,\ z/(x \cdot y),\ x \cdot \sqrt{y},\ x/\sqrt{y},\ \sqrt{y}/x,\ e^x \cdot y,\ x \cdot \ln(y),\ x/\pi,\ \pi/x, \text{ etc.},$$

where x, y, and z are floating-point numbers and assuming that the exponential and logarithm are correctly rounded. To the best of our knowledge, no tight error bounds in ulps for these functions have been published so far.

By "expressing the error bound in ulps" we mean that we want to find some real α, as small as possible, such that

$$|s - cx| \leqslant \alpha \cdot \mathrm{ulp}(cx).$$

It is wiser to measure errors in terms of ulps of the *exact result* instead of ulps of the *computed result*, because the latter choice could lead to dubious conclusions. The authors of [7, Section 2.5] illustrate this as follows:

> Assume for instance that the exact result is the real $x = 1 + u$ and consider two (quite poor) computed floating-point results: $a = 2 - 2u$ and $b = 2 + 4u$. Since $x < a < b$, it would make no sense to consider that b is a better approximation to x than a. And yet x is within $\left(2^{p-1} - 3/2\right) \mathrm{ulp}(a)$ from a, and within $\left(2^{p-2} + 3/4\right) \mathrm{ulp}(b)$ from b.

In Section V, we generalize the work of the preceding sections to the approximation of $m \cdot n$, where m and n are either real constants or correctly-rounded functions, by $s = \mathrm{RN}(\hat{m} \cdot \hat{n})$, where $\hat{m} = \mathrm{RN}(m)$ and $\hat{n} = \mathrm{RN}(n)$. The obtained results apply to functions such as $(x + y) \cdot (z + t)$, $\sqrt{x} \cdot (y \cdot z)$, $\ln(x) \cdot \ln(y)$ (assuming a correctly-rounded logarithm), etc. Finally, in Section VI, we present a similar study for the quotient of two correctly-rounded functions.

B. Just a few words on relative errors

Obtaining relative errors on the calculation of cx or x/c is straightforward. We just quickly address that question, for the sake of completeness and to show that the obtained results do not suffice for deducing a tight bound on the error in ulps. Without loss of generality, we assume $c > 0$ and $x > 0$. The relative error due to rounding to nearest a real number is bounded by $u/(1+u)$ and that bound is optimal [8, p. 232] [9, p. 74] and has been used to obtain tight relative error bounds on various operations (see e.g. [10], [11]). Therefore

$$c \cdot \left(1 - \frac{u}{1+u}\right) \leqslant \hat{c} \leqslant c \cdot \left(1 + \frac{u}{1+u}\right),$$

$$\hat{c}x \cdot \left(1 - \frac{u}{1+u}\right) \leqslant \mathrm{RN}\left(\hat{c}x\right) \leqslant \hat{c}x \cdot \left(1 + \frac{u}{1+u}\right),$$

and

$$\frac{x}{\hat{c}} \cdot \left(1 - \frac{u}{1+u}\right) \leqslant \mathrm{RN}\left(\frac{x}{\hat{c}}\right) \leqslant \frac{x}{\hat{c}} \cdot \left(1 + \frac{u}{1+u}\right).$$

From this, we easily deduce that the numbers $s_1 = \mathrm{RN}\left(\hat{c}x\right)$ and $s_2 = \mathrm{RN}\left(x/\hat{c}\right)$ satisfy

$$cx \cdot \left(1 - \frac{u}{1+u}\right)^2 \leqslant s_1 \leqslant cx \cdot \left(1 + \frac{u}{1+u}\right)^2,$$

and

$$\frac{x}{c} \cdot \frac{1 - \frac{u}{1+u}}{1 + \frac{u}{1+u}} \leqslant s_2 \leqslant \frac{x}{c} \cdot \frac{1 + \frac{u}{1+u}}{1 - \frac{u}{1+u}},$$

so that the relative error of the multiplication by c is bounded by

$$\begin{aligned} \left(1 + \frac{u}{1+u}\right)^2 - 1 &= \frac{2u + 3u^2}{1 + 2u + u^2} \\ &< 2u, \end{aligned}$$

and the relative error of the division by c is bounded by

$$\begin{aligned} \max\left\{1 - \frac{1 - \frac{u}{1+u}}{1 + \frac{u}{1+u}}; \frac{1 + \frac{u}{1+u}}{1 - \frac{u}{1+u}} - 1\right\} &= \frac{1 + \frac{u}{1+u}}{1 - \frac{u}{1+u}} - 1 \\ &= 2u. \end{aligned}$$

Very similarly, the relative error of the division of c by x is bounded by

$$\left(1+\frac{u}{1+u}\right)^2-1<2u,$$

the relative error due to the approximation of $m\cdot n$ by $\mathrm{RN}(\hat{m}\cdot\hat{n})$ is bounded by

$$\left(1+\frac{u}{1+u}\right)^3-1<3u,$$

and the relative error due to the approximation of n/d by $\mathrm{RN}(\hat{n}/\hat{d})$ is bounded by

$$\begin{array}{rcl}\dfrac{\left(1+\frac{u}{1+u}\right)^2}{1-\frac{u}{1+u}}-1 & = & \dfrac{3u+4u^2}{1+u}\\ & < & 3u+u^2\simeq 3u.\end{array}$$

Hence the relative error of the computation of cx, x/c and c/x is bounded by $2u$, and the relative error of the computation of $m\cdot n$ and n/d are bounded by around $3u$. These bounds are very tight (for instance, in binary32/single-precision arithmetic ($p=24$), error $1.99902u$ is attained for the multiplication of $c=16779263$ and $x=8392705$). From these bounds the best bounds on the error in ulps one can deduce (conversions between errors in ulps and relative errors are presented for instance in [12, Section 2.3.3]) are $2\,\mathrm{ulp}$ for the computation of cx, x/c and c/x (resp. around $3\,\mathrm{ulp}$ for the computation of $m\cdot n$ and n/d). And yet, we are going to show the significantly better bounds $1.5\,\mathrm{ulp}$ (resp. $2.5\,\mathrm{ulp}$).

II. Multiplication of a FP number by a constant or a correctly-rounded function

Let us first establish an error bound in ulps on the computation of $x\cdot c$, where x is a floating-point number and c is a constant or a correctly-rounded function. We want to bound the error of approximating $x\cdot c$ by

$$\mathrm{RN}\left(x\cdot\hat{c}\right).$$

We will first consider "general" bounds, applicable to any c. Then we will try to improve these bounds in the particular case where c is a constant.

A. *First steps*

Since $\mathrm{ulp}(2^k t)=2^k\,\mathrm{ulp}(t)$ and $\mathrm{RN}(2^k t)=2^k\,\mathrm{RN}(t)$ for any integer k and real t), without loss of generality, we can assume $1\leqslant c<2$ and $1\leqslant x<2$. Since $\hat{c}$ is c rounded to nearest, we have

$$|c-\hat{c}|\leqslant\frac{1}{2}\,\mathrm{ulp}(c),$$

and therefore

$$|cx-\hat{c}x|\leqslant\frac{x}{2}\,\mathrm{ulp}(c). \tag{1}$$

We also have

$$|s-\hat{c}x|\leqslant\frac{1}{2}\,\mathrm{ulp}(\hat{c}x), \tag{2}$$

so that, by the triangular inequality

$$|s-cx|\leqslant\frac{x}{2}\,\mathrm{ulp}(c)+\frac{1}{2}\,\mathrm{ulp}(\hat{c}x). \tag{3}$$

Equation (3) expresses the error in terms of $\mathrm{ulp}(c)$ and $\mathrm{ulp}(\hat{c}x)$, whereas we need to express it in terms of $\mathrm{ulp}(cx)$. Since $x\geqslant 1$ and ulp is an increasing function, $\mathrm{ulp}(c)$ can be bounded (although possibly non-optimally) by $\mathrm{ulp}(cx)$. The case of $\mathrm{ulp}(\hat{c}x)$ is more difficult to handle: $\hat{c}x$ and cx are very close values, between 1 and 4, so that we will *almost always* have $\mathrm{ulp}(\hat{c}x)=\mathrm{ulp}(cx)$. However, there may be some corner cases when one of $\hat{c}x$ and cx is less than 2 and the other one is larger than or equal to 2. In these cases $\mathrm{ulp}(\hat{c}x)$ will be either $\frac{1}{2}\,\mathrm{ulp}(cx)$ or $2\,\mathrm{ulp}(cx)$. Let us quickly eliminate the latter case, by showing that when $\mathrm{ulp}(\hat{c}x)>\mathrm{ulp}(cx)$ the computation is very accurate (in that case, the error will be significantly less than the general error bound we give later on).

B. *The special case* $\mathrm{ulp}(\hat{c}x)>\mathrm{ulp}(cx)$

For that case to happen, we must have

$$cx<2\leqslant\hat{c}x.$$

Since $\mathrm{ulp}(c)=2^{1-p}=2u$ and $\hat{c}=\mathrm{RN}(c)$, we have

$$c<\hat{c}\leqslant c+u,$$

and we therefore deduce

$$cx<2\leqslant\hat{c}x\leqslant cx+ux.$$

Since x is a floating-point number and $x<2$, we have $x\leqslant 2-2u$, and therefore

$$cx<2\leqslant\hat{c}x\leqslant cx+(2u-2u^2).$$

Hence, $2\leqslant\hat{c}x<2+2u-2u^2$. Since RN is an increasing function and $\mathrm{RN}(2)=\mathrm{RN}(2+2u-2u^2)=2$ we deduce that $s=\mathrm{RN}(\hat{c}x)=2$. Therefore,

$$\begin{array}{rcl}|s-cx| & = & |2-cx|\\ & \leqslant & 2u-2u^2\\ & = & (1-u)\cdot\mathrm{ulp}(cx).\end{array}$$

C. *The general case* $\mathrm{ulp}(\hat{c}x)\leqslant\mathrm{ulp}(cx)$

1) A bound that does not depend on c: In that case, (2) gives

$$|s-\hat{c}x|\leqslant\frac{1}{2}\,\mathrm{ulp}(cx), \tag{4}$$

and (3) implies

$$|s-cx|\leqslant\frac{x}{2}\,\mathrm{ulp}(c)+\frac{1}{2}\,\mathrm{ulp}(cx). \tag{5}$$

As $x\geqslant 1$, $\mathrm{ulp}(c)\leqslant\mathrm{ulp}(cx)$. Since $x\leqslant 2-2u$, we finally obtain

$$|s-cx|\leqslant\left(\frac{3}{2}-u\right)\cdot\mathrm{ulp}(cx). \tag{6}$$

Example 1 below shows that the bound (6) is asymptotically optimal (as $u\to 0$ or, equivalently, as $p\to+\infty$): one cannot improve it (at least at order 0 in u) without further assumptions on c. Making further assumptions on c makes no sense if c is the result of a previous operation or function (so that the bound (6) will not be improved at order 0 in u for calculations such as $(x+y)\cdot z$, $(x\cdot y)\cdot z$, $x\cdot\sqrt{y}$, $e^x\cdot y$, $x\cdot\ln(y)$, etc.).

However, if c is a real constant (such as π, $\ln(2)$, or one of the terms of the form $\cos(2k\pi/N)$ that appear in Fourier-related transforms), it makes sense to try to find a sharper bound that depends on c.

2) Improving the bound when c is a constant: We obtained (5) by adding (4) and (1). Now, we will keep (4)—which cannot be improved, unless $\hat{c}$ is a power of 2—unchanged, and try to improve on (1), by introducing a new bound on $|cx - \hat{c}x|$ that depends on c. We start from

$$|cx - \hat{c}x| = x \cdot |c - \hat{c}| .$$

Two cases may occur:

- if $x < 2/c$ then $\mathrm{ulp}(cx) = \mathrm{ulp}(c) = 2^{1-p} = 2u$, and therefore
$$|cx - \hat{c}x| < 2^p \cdot \left|\frac{c-\hat{c}}{c}\right| \cdot \mathrm{ulp}(cx); \quad (7)$$
- if $x \geqslant 2/c$ then $\mathrm{ulp}(cx) = 2 \ \mathrm{ulp}(c) = 2^{2-p}$, and therefore
$$|cx - \hat{c}x| < 2^{p-1} \cdot |c - \hat{c}| \cdot \mathrm{ulp}(cx). \quad (8)$$

Since c is between 1 and 2, the bound (8) is always less than the bound (7). Hence, the bound (7) holds in all cases.

We immediately deduce

$$|s - cx| \leqslant \left(\frac{1}{2} + 2^p \cdot \left|\frac{c-\hat{c}}{c}\right|\right) \cdot \mathrm{ulp}\,(cx). \quad (9)$$

Note that the bound (9) varies with p. One may obtain a bound that no longer depends on p, but is in general looser, by defining

$$\mathrm{mant}(c) = \frac{c}{\mathrm{ufp}(c)} = \frac{c}{2^{\lfloor \log_2(c) \rfloor}}$$

(it is the "infinite precision significand" of c—up to now, in our proofs, we used $1 \leqslant c < 2$, in which case $\mathrm{mant}(c) = c$, but to give a final result that is valid for all positive c, we need here to use function mant). If we note that $|c - \hat{c}| \leqslant u \cdot \mathrm{ufp}(c)$, we obtain

$$|s - cx| \leqslant \left(\frac{1}{2} + \frac{1}{\mathrm{mant}(c)}\right) \cdot \mathrm{ulp}\,(cx).$$

3) Summary: We finally obtain

Property 2.1: Assume radix-2, precision-p floating-point arithmetic. If c is a real constant or a real function of one or more variables, x is a floating-point number, and $\hat{c} = \mathrm{RN}(c)$ is the correctly-rounded (to nearest) implementation of c then, barring underflow and overflow, the FP number $s = \mathrm{RN}(\hat{c} \cdot x)$ satisfies

$$|s - cx| \leqslant \left(\frac{3}{2} - u\right) \cdot \mathrm{ulp}\,(cx) < \frac{3}{2}\,\mathrm{ulp}\,(cx). \qquad \square$$

Property 2.2: Assume radix-2, precision-p floating-point arithmetic. If c is a nonzero real constant, x is a floating-point number, and $\hat{c} = \mathrm{RN}(c)$ then, barring underflow and overflow, the FP number $s = \mathrm{RN}(\hat{c} \cdot x)$ satisfies

$$|s - cx| \leqslant \left(\frac{1}{2} + \frac{1}{\mathrm{mant}(c)}\right) \cdot \mathrm{ulp}\,(cx),$$
$$|s - cx| \leqslant \left(\frac{1}{2} + 2^p \cdot \left|\frac{c-\hat{c}}{c}\right|\right) \cdot \mathrm{ulp}\,(cx). \qquad \square$$

The bound of Property 2.1 is general. The first bound of Property 2.2 is tighter but requires c to be a constant and depends on its value. The second bound of Property 2.2 is even tighter but depends on the values of c and p.

The following "generic" (i.e., parameterized by the precision p) example shows that in the general case of an arbitrary constant c, the bound given by Property 2.1 is asymptotically optimal.

Example 1: Assume RN breaks ties to even. If p is even, the choice

$$\begin{aligned} x &= 2^p - 2^{p/2}, \\ c &= 1 + 2^{-p/2-1} - 2^{-p}, \end{aligned}$$

gives

$$\begin{aligned} \hat{c} &= 1 + 2^{-p/2-1}, \\ cx &= 2^p - 2^{p/2-1} - \tfrac{3}{2} + 2^{-p/2}, \\ s &= \mathrm{RN}(\hat{c}x) = 2^p - 2^{p/2-1}, \end{aligned}$$

resulting in an error

$$|s - cx| = \left(\frac{3}{2} - 2^{-p/2}\right) \mathrm{ulp}(cx).$$

If p is odd, the choice

$$\begin{aligned} x &= 2^p - 2^{(p-1)/2}, \\ c &= 1 + 2^{-(p+1)/2} - 2^{-p}, \end{aligned}$$

gives

$$\begin{aligned} \hat{c} &= 1 + 2^{-(p+1)/2}, \\ cx &= 2^p - \tfrac{3}{2} + 2^{(-p-1)/2}, \\ s &= \mathrm{RN}(\hat{c}x) = 2^p, \end{aligned}$$

resulting in an error

$$|s - cx| = \left(\frac{3}{2} - 2^{-(p+1)/2}\right) \mathrm{ulp}(cx).$$

For instance, if $p = 24$, our example gives

$$\begin{aligned} c &= 16779263/2^{24}, \\ x &= 16773120, \end{aligned}$$

and an error equal to $1.499756 \cdots \mathrm{ulp}(cx)$; if $p = 53$, it gives

$$\begin{aligned} c &= 9007199321849855/2^{53}, \\ x &= 9007199187632128, \end{aligned}$$

and an error $1.49999999254942 \cdots \mathrm{ulp}(cx)$; and if $p = 113$ it gives

$$\begin{aligned} c &= 10384593717069655329118586696368127/2^{113}, \\ x &= 10384593717069655185003398620512256, \end{aligned}$$

and an error

$$1.49999999999999999993061106 \cdots \mathrm{ulp}(cx).$$

Tables I, II, and III present the various bounds one obtains from Properties 2.1 and 2.2 in the cases $c = \pi$,

TABLE I

THE VARIOUS BOUNDS GIVEN BY PROPERTIES 2.1 AND 2.2 IN THE CASE $c = \pi$, COMPARED WITH THE ACTUAL LARGEST ERROR OBTAINED THROUGH EXHAUSTIVE TESTING (FOR SMALL VALUES OF p ONLY).

p	8	16	24	53	113
bound of Prop. 2.1	1.496093750	1.499984741	1.499999940	1.500000000	1.500000000
1st bound of Prop. 2.2	1.136619772	1.136619772	1.136619772	1.136619772	1.136619772
2nd bound of Prop. 2.2	.5788515082	.6858466083	.9668685680	.8511161042	.7866483180
actual largest error	.5176877776	.6825298419			

TABLE II

THE VARIOUS BOUNDS GIVEN BY PROPERTIES 2.1 AND 2.2 IN THE CASE $c = \cos(5\pi/32)$, COMPARED WITH THE ACTUAL LARGEST ERROR OBTAINED THROUGH EXHAUSTIVE TESTING (FOR SMALL VALUES OF p ONLY).

p	8	16	24	53	113
bound of Prop. 2.1	1.496093750	1.499984741	1.499999940	1.500000000	1.500000000
1st bound of Prop. 2.2	1.066944035	1.066944035	1.066944035	1.066944035	1.066944035
2nd bound of Prop. 2.2	.7587037370	.9626486317	1.013690470	.7026621871	.8537866473
actual largest error	.7004712694	.9585313311			

TABLE III

THE VARIOUS BOUNDS GIVEN BY PROPERTIES 2.1 AND 2.2 IN THE CASE $c = 263/256$, COMPARED WITH THE ACTUAL LARGEST ERROR OBTAINED THROUGH EXHAUSTIVE TESTING (FOR SMALL VALUES OF p ONLY).

p	8	16	24	53	113
bound of Prop. 2.1	1.496093750	1.499984741	1.499999940	1.500000000	1.500000000
1st bound of Prop. 2.2	1.473384030	1.473384030	1.473384030	1.473384030	1.473384030
2nd bound of Prop. 2.2	1.473384030	.5000000000	.5000000000	.5000000000	.5000000000
actual largest error	1.437500000	.5000000000			

$c = \cos(5\pi/32)$, and $c = 263/256$, and for various values of the precision p ranging from 8 to 113. For small values of p an exhaustive search for the actual largest error was possible: it shows that the second bound of Property 2.2 is very tight.

n	2nd bound of Prop. 2.2
4	1.0140506566
5 − 7	1.2984865212
8	1.3717040024
9 − 12	1.4501519783
13	1.4616997574
14	1.4761273962
15 − 18	1.4940991041
19 − 20	1.4970223192
⋯	⋯
32	1.4999503673

TABLE IV

FOR VARIOUS VALUES OF n, WE GIVE THE LARGEST VALUE OF THE 2ND BOUND OF PROP. 2.2 FOR ALL CONSTANTS $\cos(2k\pi/2^n)$ THAT APPEAR IN A FFT OR DCT OF SIZE 2^n, ASSUMING $p = 24$, WHICH CORRESPONDS TO BINARY32 ARITHMETIC.

Table IV presents, for various values of n, the largest value of the second bound of Property 2.2 for all constants $\cos(2k\pi/2^n)$ that appear in a FFT or DCT of size 2^n, assuming $p = 24$. For instance, from that table, someone wanting to do an error analysis of a DCT of 2^{12} elements in binary32 arithmetic can assume that all multiplications by the terms $\cos(2k\pi/2^n)$ are performed with an error less than 1.4502 ulp.

Example 1 establishes the asymptotical optimality of the bound of Property 2.1 when c is a constant, but it also gives information for some particular functions:

- since the values of c used in Example 1 can straightforwardly be obtained by adding two precision-p floating-point numbers, we also deduce the asymptotic optimality of the bound of Property 2.1 for the calculation of $\mathtt{z} * (\mathtt{x} + \mathtt{y})$;
- in the particular cases (but they are the ones that matter the most in practice, since they correspond to binary32/single-precision, binary64/double-precision and binary128/quad-precision arithmetics) $p = 24$, $p = 53$ and $p = 113$, factorizations of the particular values of c given in the example show that these values are the product of two floating-point numbers. This immediately shows that errors $1.499756\cdots \mathrm{ulp}(cx)$ (in binary32 arithmetic), $1.49999999254942\cdots \mathrm{ulp}(cx)$ (in binary64 arithmetic), and $1.4999999999999999993061106\cdots \mathrm{ulp}(cx)$ (in binary128 arithmetic) can be attained when calculating $\mathtt{z} * (\mathtt{x} * \mathtt{y})$, showing that for that function, the bound of Property 2.1 is very tight.

For various other functions, we have built examples that show the sharpness of the bound of Property 2.1. For instance, in binary64 arithmetic, with

$$x = 9007197761440759$$

and

$$y = 4503599630388691/2^{52}$$

the error made when computing $x\sqrt{y}$ is $1.4991\,\mathrm{ulp}(x\sqrt{y})$.

III. Division of a FP number by a correctly-rounded function

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

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

where, as previously, $\hat{c} = \mathrm{RN}(c)$. Here, the case where c is a constant is of little interest, since one seldom divides by a constant (multiplying by the reciprocal of that constant c is in general a much better option, unless c is much closer to a FP number than $1/c$, in which case the division, although in general significantly slower, will be more accurate than the multiplication). However the other cases are important: the results we give below cover computations such as $x/(y+z)$, $x/\sqrt{y}$, $x/(y\cdot z)$, $x/\cos(y)$, etc., where x, y, and z are floating-point numbers and assuming function cos is correctly rounded.

Without loss of generality, we assume $1 \leqslant x < 2$ (so that, since x is a FP number, $1 \leqslant x \leqslant 2-2u$), and $1 \leqslant c < 2$ (which implies $1 \leqslant \hat{c} \leqslant 2$). We have

$$\left|\frac{1}{c} - \frac{1}{\hat{c}}\right| = \frac{|\hat{c}-c|}{\hat{c}c} \leqslant \frac{u}{\hat{c}c} \leqslant u. \tag{10}$$

Let us first eliminate a few particular cases

- if $x = 1$ and $\hat{c} = 1$ then

$$\begin{array}{rcl}\left|s - \dfrac{x}{c}\right| & \leqslant & \left|\mathrm{RN}\left(\dfrac{1}{\hat{c}}\right) - \dfrac{1}{\hat{c}}\right| + \left|\dfrac{1}{c} - \dfrac{1}{\hat{c}}\right| \\ & \leqslant & 0 + u,\end{array}$$

  and that bound u is equal to $\mathrm{ulp}(1/c)$ if $c > 1$ (and to $(1/2)\,\mathrm{ulp}(1/c)$ if $c = 1$ but in that straightforward case, the error is obviously zero);
- $x = 1$ and $\hat{c} > 1$ (which implies $\hat{c} \geqslant 1+2u$ and $c \geqslant 1+u$) then, using (10),

$$\begin{aligned}\left|s - \frac{x}{c}\right| &\leqslant \left|\mathrm{RN}\left(\frac{1}{\hat{c}}\right) - \frac{1}{\hat{c}}\right| + \left|\frac{1}{c} - \frac{1}{\hat{c}}\right| \\ &\leqslant \frac{u}{2} + \frac{u}{(1+2u)(1+u)} \\ &= \frac{3}{2}u - \frac{3u^2+2u^3}{1+3u+2u^2} \\ &= \left(\frac{3}{2} - \frac{3u+2u^2}{1+3u+2u^2}\right)\mathrm{ulp}\left(\frac{1}{c}\right).\end{aligned}$$

  and that bound is less than

$$\left(\frac{3}{2} - 3u + 7u^2\right)\mathrm{ulp}\left(\frac{1}{c}\right);$$

- if $\hat{c} = 1$ and $x \neq 1$ (so that $x \geqslant 1+2u$) then $s = \mathrm{RN}(x/\hat{c}) = x/\hat{c}$, so that

$$\begin{array}{rcl}\left|s - \dfrac{x}{c}\right| & = & x\cdot\left|\dfrac{1}{c} - \dfrac{1}{\hat{c}}\right| \\ & \leqslant & x\cdot\dfrac{u}{\hat{c}c} \\ & \leqslant & 2u - 2u^2,\end{array}$$

  and $\hat{c} = 1$ implies $1 \leqslant c \leqslant 1+u$, so that

$$\frac{x}{c} \geqslant \frac{1+2u}{1+u} > 1,$$

  hence $\mathrm{ulp}(x/c) = 2u$ and therefore

$$\left|s - \frac{x}{c}\right| \leqslant (1-u)\,\mathrm{ulp}\left(\frac{x}{c}\right).$$

We can now assume $1+2u \leqslant x \leqslant 2-2u$ and $1+2u \leqslant \hat{c} \leqslant 2$ (so that $1+u \leqslant c \leqslant 2$). We have

$$\left|s - \frac{x}{c}\right| \leqslant \left|\mathrm{RN}\left(\frac{x}{\hat{c}}\right) - \frac{x}{\hat{c}}\right| + \left|\frac{x}{c} - \frac{x}{\hat{c}}\right|.$$

- If $c \leqslant x$ then $\hat{c} \leqslant x$ (since RN is an increasing function and $\mathrm{RN}(x) = x$) and therefore $\mathrm{ulp}(x/c) = \mathrm{ulp}(x/\hat{c})$. Also,

$$\left|\mathrm{RN}\left(\frac{x}{\hat{c}}\right) - \frac{x}{\hat{c}}\right| \leqslant u,$$

  and

$$\left|\frac{x}{c} - \frac{x}{\hat{c}}\right| \leqslant x\cdot\frac{u}{\hat{c}c} \leqslant (2-2u)\cdot\frac{u}{(1+2u)(1+u)},$$

  so that

$$\begin{aligned}\left|s - \frac{x}{c}\right| &\leqslant u\cdot\left(1 + \frac{2-2u}{(1+2u)(1+u)}\right) \\ &= 3u - \frac{4u^2(2+u)}{1+3u+2u^2},\end{aligned}$$

  and $\mathrm{ulp}(x/c) = 2u$, so that

$$\begin{aligned}\left|s - \frac{x}{c}\right| &\leqslant \left(\frac{3}{2} - \frac{4u+2u^2}{1+3u+2u^2}\right)\mathrm{ulp}\left(\frac{x}{c}\right) \\ &\leqslant \left(\frac{3}{2} - 4u + 10u^2\right)\mathrm{ulp}\left(\frac{x}{c}\right).\end{aligned}$$

- If $c > x$ then $\hat{c} \geqslant x$ (since RN is an increasing function). We have

$$\left|\mathrm{RN}\left(\frac{x}{\hat{c}}\right) - \frac{x}{\hat{c}}\right| \leqslant \frac{u}{2},$$

  and

$$\begin{array}{rcl}\left|\dfrac{x}{c} - \dfrac{x}{\hat{c}}\right| & \leqslant & x\cdot\dfrac{u}{\hat{c}c} \\ & = & \left(\dfrac{x}{c}\right)\cdot\dfrac{u}{\hat{c}} \\ & < & 1\cdot\dfrac{u}{1+2u},\end{array}$$

  and $\mathrm{ulp}(x/c) = u$, so that

$$\begin{aligned}\left|s - \frac{x}{c}\right| &\leqslant \left(\frac{1}{2} + \frac{1}{1+2u}\right)\mathrm{ulp}\left(\frac{x}{c}\right) \\ &= \left(\frac{3}{2} - \frac{2u}{1+2u}\right)\mathrm{ulp}\left(\frac{x}{c}\right) \\ &\leqslant \left(\frac{3}{2} - 2u + 4u^2\right)\mathrm{ulp}\left(\frac{x}{c}\right).\end{aligned}$$

Putting all this together, we obtain

Property 3.1: Assume radix-2, precision-p floating-point arithmetic. If c is a real constant or a real function of one or more variables, x is a floating-point number, and $\hat{c} = \mathrm{RN}(c)$ is the correctly-rounded (to nearest) implementation of c then, barring underflow and overflow, the FP number $s = \mathrm{RN}(x/\hat{c})$ satisfies

$$\begin{aligned}\left|s - \frac{x}{c}\right| &\leqslant \left(\frac{3}{2} - \frac{2u}{1+2u}\right) \mathrm{ulp}\left(\frac{x}{c}\right) \\ &\leqslant \left(\frac{3}{2} - 2u + 4u^2\right) \mathrm{ulp}\left(\frac{x}{c}\right). \qquad \square\end{aligned}$$

The following example shows that when c is an arbitrary real constant, the bound of Property 3.1 is asymptotically optimal.

Example 2: If p is even and larger than 4, the choice

$$\left\{\begin{array}{lcl} x &=& 2 - 2^{-p/2}, \\ c &=& 1 + 2^{-p/2-1} - 2^{-p}, \end{array}\right.$$

gives

$$\begin{gathered}\hat{c} = 1 + 2^{-p/2-1}, \\ x/c = 2 - 2^{-p/2+1} + 3 \cdot 2^{-p} + O\left(2^{-3p/2}\right), \\ \mathrm{RN}(x/\hat{c}) = 2 - 2^{-p/2+1},\end{gathered}$$

resulting in an error

$$\left|\frac{x}{c} - \mathrm{RN}\left(\frac{x}{\hat{c}}\right)\right| = \left(\frac{3}{2} + O\left(2^{-p/2}\right)\right) \mathrm{ulp}\left(\frac{x}{c}\right).$$

If p is odd and larger than 5, the choice

$$\left\{\begin{array}{lcl} x &=& 1, \\ c &=& 1 + 2^{-(p+1)/2} - 2^{-p}, \end{array}\right.$$

gives

$$\begin{gathered}\hat{c} = 1 + 2^{-(p+1)/2}, \\ x/c = 1 - 2^{-(p+1)/2} + 3 \cdot 2^{-p-1} + O\left(2^{-3p/2}\right), \\ \mathrm{RN}(x/\hat{c}) = 1 - 2^{-(p+1)/2},\end{gathered}$$

resulting again in an error

$$\left|\frac{x}{c} - \mathrm{RN}\left(\frac{x}{\hat{c}}\right)\right| = \left(\frac{3}{2} + O\left(2^{-p/2}\right)\right) \mathrm{ulp}\left(\frac{x}{c}\right).$$

For instance, if $p = 24$, our example gives

$$\begin{array}{lcl} c &=& 16779263/2^{24}, \\ x &=& 8191/4096, \end{array}$$

and an error equal to $1.49957\cdots \mathrm{ulp}(x/c)$; if $p = 53$, it gives

$$\begin{array}{lcl} c &=& 9007199321849855/2^{53}, \\ x &=& 1, \end{array}$$

and an error $1.49999998137\cdots \mathrm{ulp}(x/c)$; and if $p = 113$ it gives

$$\begin{array}{lcl} c &=& 10384593717069655329118586696368127/2^{113} \\ x &=& 1, \end{array}$$

and an error $1.49999999999999998265\cdots \mathrm{ulp}(x/c)$.

Example 2 establishes the asymptotical optimality when c is a constant, but also gives information for some particular functions. It turns out that the values of c in Example 2 are the same as those in Example 1, so the reasoning is the same:

- since the values of c used in Example 2 can straightforwardly be obtained by adding two FP numbers, we also deduce the asymptotic optimality of the bound of Property 3.1 for the calculation of $\mathtt{z}/(\mathtt{x}+\mathtt{y})$.
- in the particular cases $p = 24$ (binary32 arithmetic), $p = 53$ (binary64) and $p = 113$ (binary128), factorizations of the particular values of c given in the example show that these values can be obtained by multiplying FP numbers. This immediately shows that errors $1.49957\cdots \mathrm{ulp}(x/c)$ (in single precision), $1.49999998137\cdots \mathrm{ulp}(x/c)$ (in double precision), and $1.49999999999999998265\cdots \mathrm{ulp}(x/c)$ (in quad precision) can be attained when calculating $\mathtt{z}/(\mathtt{x} * \mathtt{y})$, showing that for that function, the bound of Property 3.1 is very tight.

For other functions, testings in small precisions show that the bounds are tight. For instance, if $p = 24$ (which corresponds to binary32/single-precision arithmetic), error $1.4959\,\mathrm{ulp}(x/\sqrt{y})$ is attained when computing $x/\sqrt{y}$ for

$$x = 16763899$$

and

$$y = 8396805/2.$$

For the same function in binary64 arithmetic, error $1.49906\,\mathrm{ulp}(x/\sqrt{y})$ is attained for

$$x = 9007198105271337$$

and

$$y = 4503599631275935/2^{52}.$$

IV. 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

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

where, as previously, $\hat{c} = \mathrm{RN}(c)$. Since the reasoning is highly similar to that of Section III, we only outline the major points. The cases where c or x are a power of 2 are straightforward, so without loss of generality, we assume $1 < x < 2$ (so that, since x is a FP number, $1 + 2u \leqslant x \leqslant 2 - 2u$), and $1 < c < 2$ (which implies $1 \leqslant \hat{c} \leqslant 2$).

- if $c < x$ then $\hat{c} \leqslant x$, and therefore
$$\left|\frac{\hat{c}}{x} - \mathrm{RN}\left(\frac{\hat{c}}{x}\right)\right| \leqslant \frac{u}{2},$$
and
$$\begin{aligned}\left|\frac{\hat{c}}{x} - \frac{c}{x}\right| &= \frac{1}{x} \cdot |\hat{c} - c| \\ &\leqslant \frac{1}{1+2u} \cdot u,\end{aligned}$$
so that
$$\begin{aligned}\left|s - \frac{c}{x}\right| &\leqslant \frac{3u + 2u^2}{2+4u} \\ &= \frac{3+2u}{2+4u} \cdot \mathrm{ulp}\left(\frac{c}{x}\right).\end{aligned}$$
- if $c \geqslant x$ then $\hat{c} \geqslant x$, and therefore
$$\left|\frac{\hat{c}}{x} - \mathrm{RN}\left(\frac{\hat{c}}{x}\right)\right| \leqslant u,$$

and

$$\left|\frac{\hat{c}}{x}-\frac{c}{x}\right| = \frac{1}{x}\cdot|\hat{c}-c| \leqslant \frac{1}{1+2u}\cdot u < u,$$

and therefore

$$\left|s-\frac{c}{x}\right| < 2u = \mathrm{ulp}\left(\frac{c}{x}\right).$$

Putting all this together, we conclude

Property 4.1: Assume radix-2, precision-p floating-point arithmetic. If c is a real constant or a real function of one or more variables, x is a floating-point number, and $\hat{c} = \mathrm{RN}(c)$ is the correctly-rounded (to nearest) implementation of c then, barring underflow and overflow, the FP number $s = \mathrm{RN}(\hat{c}/x)$ satisfies

$$\begin{aligned}\left|s-\frac{c}{x}\right| &\leqslant \frac{3+2u}{2+4u}\cdot\mathrm{ulp}\left(\frac{c}{x}\right)\\ &\leqslant \left(\frac{3}{2}-2u+4u^2\right)\mathrm{ulp}\left(\frac{c}{x}\right). \qquad \square\end{aligned}$$

Property 4.1 covers calculations such as $\ln(2)/x$, $\sqrt{x}/y$, $(x+y)/z$, ... In the general case, the bound given by Property 4.1 is very tight. For instance, if $p = 53$, $c = 2^{53}+1$, and $x = 2^{52}+2^{25}$, assuming RN breaks ties to even, we obtain $\hat{c} = 2^{53}$ and $s = 134217727/2^{26}$, resulting in an error

$$1.4999999888241291\cdots\mathrm{ulp}(c/x).$$

That example can be transformed into a "generic" example, that shows that the bound of Property 4.1 is asymptotically optimal for odd values of p (at least assuming that RN breaks ties to even which is the default in IEEE 754 arithmetic). This is done as follows. Let p be an odd integer, let $c = 2^p+1$ (hence $\hat{c} = 2^p$) and

$$x = 2^{p-1}+2^{(p-3)/2}.$$

We have

$$\begin{aligned}\frac{\hat{c}}{x} = \frac{2^p}{2^{p-1}+2^{(p-3)/2}} &= 2\frac{1}{1+2^{-(p+1)/2}}\\ &= 2(1-2^{-(p+1)/2}+\varepsilon_p),\end{aligned}$$

with $|\varepsilon_p| < 2^{-(p+1)}$. Therefore,

$$\mathrm{RN}(\hat{c}/x) = 2(1-2^{-(p+1)/2}),$$

and

$$\begin{aligned}\mathrm{RN}\left(\frac{\hat{c}}{x}\right)-\frac{c}{x} &= 2(1-2^{-(p+1)/2}) - \frac{2+2^{-(p-1)}}{1+2^{-(p+1)/2}}\\ &= 2(1-2^{-(p+1)/2})\\ &\quad -(2+2^{-(p-1)})(1-2^{-(p+1)/2}+2^{-(p+1)}+\eta_p),\end{aligned}$$

where $|\eta_p| < 2^{-3(p+1)/2}$. It then follows

$$\begin{aligned}&\left|\mathrm{RN}\left(\frac{\hat{c}}{x}\right)-\frac{c}{x}\right|\\ &= \left(\frac{3}{2}-2^{-(p+1)/2}+2^{-(p+1)}+\eta_p(1+2^p)\right)\mathrm{ulp}\left(\frac{c}{x}\right)\\ &= \left(\frac{3}{2}+2^{-(p+1)/2}\gamma_p\right)\mathrm{ulp}\left(\frac{c}{x}\right),\end{aligned}$$

where $|\gamma_p| < 3$.

V. Product of two correctly-rounded functions

We now consider the approximation of $m\cdot n$, where m and n are either real constants or correctly-rounded functions, by

$$s = \mathrm{RN}\left(\hat{m}\cdot\hat{n}\right),$$

where $\hat{m} = \mathrm{RN}(m)$ and $\hat{n} = \mathrm{RN}(n)$ (the case where *both* m and n are constants is of course of little interest). To deal with this case, we need the following lemma.

Lemma 5.1: Let x and y be real numbers satisfying

$$1 \leqslant x, y \leqslant 2.$$

If $xy \leqslant 2$ then

$$x+y \leqslant 3. \qquad \square$$

The proof of Lemma 5.1 is simple: $xy \leqslant 2$ implies $x \leqslant 2/y$, therefore $x+y \leqslant 2/y+y$. Since $y > 0$, the number $2/y+y-3$ has the same sign as

$$P(y) = y^2-3y+2.$$

The roots of polynomial P are 1 and 2 and we immediately deduce that $P(y) \leqslant 0$ for y between 1 and 2.

Let us now bound the error $|s-mn|$. Without loss of generality, we assume $1 \leqslant m < 2$ and $1 \leqslant n < 2$. We have

$$\begin{aligned}|s-mn| &= \left|\mathrm{RN}\left(\hat{m}\cdot\hat{n}\right)-mn\right|\\ &\leqslant \left|\mathrm{RN}\left(\hat{m}\cdot\hat{n}\right)-\hat{m}\cdot\hat{n}\right| + \left|\hat{m}\cdot\hat{n}-mn\right|,\end{aligned}$$

and

$$\begin{aligned}|\hat{m}\cdot\hat{n}-mn| &\leqslant \hat{m}\cdot|\hat{n}-n| + n\cdot|\hat{m}-m|\\ &\leqslant (\hat{m}+n)\cdot u.\end{aligned}$$

- if $mn \leqslant 2$ and $\hat{m}\hat{n} \leqslant 2$ then $\left|\mathrm{RN}\left(\hat{m}\cdot\hat{n}\right)-\hat{m}\cdot\hat{n}\right| \leqslant u$. Furthermore, Lemma 5.1 implies $m+n \leqslant 3$ and therefore $\hat{m}+n \leqslant 3+u$. All this gives $|s-mn| \leqslant 4u+u^2$, and since $\mathrm{ulp}(mn) \geqslant 2u$, we find
$$|s-mn| \leqslant (2+u/2)\,\mathrm{ulp}(mn);$$
- if $mn > 2$ then $\left|\mathrm{RN}\left(\hat{m}\cdot\hat{n}\right)-\hat{m}\cdot\hat{n}\right| \leqslant 2u$ and $\hat{m}+n \leqslant 4$, so that
$$|s-mn| \leqslant 6u = \frac{3}{2}\,\mathrm{ulp}(mn);$$
- if $mn \leqslant 2$ and $\hat{m}\hat{n} > 2$ then $\left|\mathrm{RN}\left(\hat{m}\cdot\hat{n}\right)-\hat{m}\cdot\hat{n}\right| \leqslant 2u$ and (from Lemma 5.1) $\hat{m}+n \leqslant 3+u$, so that
$$|s-mn| \leqslant 5u+u^2,$$
and this last bound is
$$\left(\frac{5}{2}+\frac{u}{2}\right)\cdot\mathrm{ulp}(mn)$$
when $mn < 2$, and
$$\left(\frac{5}{4}+\frac{u}{4}\right)\cdot\mathrm{ulp}(mn)$$
when $mn = 2$.

We finally obtain

Property 5.2: Assume radix-2, precision-p floating-point arithmetic. If m and n are real constants or real functions of one or more variables, $\hat{m} = \mathrm{RN}(m)$ and $\hat{n} = \mathrm{RN}(n)$ then, barring underflow and overflow, the floating-point number $s = \mathrm{RN}(\hat{m} \cdot \hat{m})$ satisfies

$$|s - mn| \leqslant \left(\frac{5}{2} + \frac{u}{2}\right) \mathrm{ulp}\,(mn)\,. \qquad \square$$

The bound given by Property 5.2 is very tight. It is asymptotically optimal for even values of p . This is shown when RN breaks ties to even by considering

$$\begin{cases} m &= 2^p + 2^{p/2} - 1, \text{ and} \\ n &= 2^{p+1} - 2^{p/2+1} + 3, \end{cases}$$

for which we obtain

$$\begin{aligned} \hat{m} &= 2^p + 2^{p/2} \\ \hat{n} &= 2^{p+1} - 2^{p/2+1} + 4 \\ mn &= 2^{2p+1} - 2^p + 5 \cdot 2^{p/2} - 3 \\ s &= 2^{2p+1} + 2^{p+2}. \end{aligned}$$

With another tie-breaking rule, adding a very small $\epsilon > 0$ to m and n gives the same $\hat{m}$ and $\hat{n}$ and the same s. We conjecture that the bound is also asymptotically optimal for odd values of the precision p but we have no proof. For instance, in binary32 arithmetic ($p = 24$), the error committed when computing $(x + y) \cdot (z + t)$, where x, y, z, and t are the 4 FP numbers defined as follows:

$$\begin{cases} x &= 2^{24} = 16777216; \\ y &= 2^{12} - 1 = 4095; \\ z &= 2^{25} - 2^{13} = 33546240; \\ t &= 3; \end{cases}$$

is equal to

$$2.4993897 \cdots \mathrm{ulp}\big((x + y) \cdot (z + t)\big)$$

(this corresponds to Property 5.2, with $m = x + y$ and $n = z + t$). Even if we have no proof of asymptotic optimality for odd p, we have found cases for which the error is very close to the bound. For instance if $p = 53$, with

$$\begin{aligned} m &= 9007199877083003, \text{ and} \\ n &= 18014397264798047, \end{aligned}$$

the error is $2.4999982\,\mathrm{ulp}(mn)$. As m is the product of the two binary64 numbers

$$e = 290554834744613$$

and

$$f = 31,$$

and n is the product of the two binary64 numbers

$$g = 29$$

and

$$h = 621186112579243,$$

error $2.4999982\,\mathrm{ulp}(efgh)$ is attained when computing $(\texttt{e} * \texttt{f}) * (\texttt{g} * \texttt{h})$ in binary64/double-precision arithmetic, which is very close to the bound of Property 5.2. Property 5.2 covers calculations such as $\pi \cdot \sqrt{x}$, $(x + y) \cdot (z + t)$, etc. If an FMA (*fused multiply-add*) 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.

VI. Quotient of two correctly-rounded functions

We finally consider the approximation of n/d, where n and d are either real constants or correctly-rounded functions, by

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

where $\hat{n} = \mathrm{RN}(n)$ and $\hat{d} = \mathrm{RN}(d)$ (the case where *both* n and d are constants is of course of little interest). We assume that n and d are not FP numbers (that case is covered by Sections III and IV), so that without loss of generality, we can assume $1 < n < 2$ and $1 < d < 2$, which implies $|\hat{n} - n| \leqslant u$ and $|\hat{d} - d| \leqslant u$. We have

$$\begin{aligned} \left|\frac{\hat{n}}{\hat{d}} - \frac{n}{d}\right| &\leqslant \left|\frac{\hat{n}}{\hat{d}} - \frac{n}{\hat{d}}\right| + \left|\frac{n}{\hat{d}} - \frac{n}{d}\right| \\ &= \frac{1}{\hat{d}} \cdot |\hat{n} - n| + \frac{n}{d} \cdot \left|\frac{\hat{d} - d}{\hat{d}}\right| \\ &\leqslant \left(1 + \frac{n}{d}\right) \cdot \frac{u}{\hat{d}}. \end{aligned}$$

If $\hat{n} = \hat{d}$ or $\hat{n} = 2\hat{d}$ or $\hat{n} = \hat{d}/2$ then $s = \hat{n}/\hat{d}$, and otherwise $n < d \Rightarrow \hat{d}/2 < \hat{n} < \hat{d}$ and $n > d \Rightarrow \hat{n} > \hat{d} > \hat{n}/2$ so that $\mathrm{ulp}(\hat{n}/\hat{d}) = \mathrm{ulp}(n/d)$. Therefore, in all cases,

$$\left|s - \frac{\hat{n}}{\hat{d}}\right| \leqslant \frac{1}{2}\,\mathrm{ulp}\left(\frac{n}{d}\right).$$

Putting all this together, we obtain,

$$\left|s - \frac{n}{d}\right| \leqslant \frac{1}{2}\,\mathrm{ulp}\left(\frac{n}{d}\right) + \left(1 + \frac{n}{d}\right) \cdot \frac{u}{\hat{d}},$$

which immediately gives:

- if $n \geqslant d$ then $\mathrm{ulp}(n/d) = 2u$ and
$$\left(1 + \frac{n}{d}\right) \cdot \frac{u}{\hat{d}} \leqslant 3u,$$
so that
$$\left|s - \frac{n}{d}\right| \leqslant 2\,\mathrm{ulp}\left(\frac{n}{d}\right);$$
- if $n < d$ then $\mathrm{ulp}(n/d) = u$ and
$$\left(1 + \frac{n}{d}\right) \cdot \frac{u}{\hat{d}} \leqslant 2u,$$
so that
$$\left|s - \frac{n}{d}\right| \leqslant \frac{5}{2}\,\mathrm{ulp}\left(\frac{n}{d}\right).$$

We therefore obtain

Property 6.1: Assume radix-2, precision-p floating-point arithmetic. If n and d are real constants or real functions of one or more variables, $\hat{n} = \mathrm{RN}(n)$ and $\hat{d} = \mathrm{RN}(d)$ then, barring underflow and overflow, the floating-point number $s = \mathrm{RN}(\hat{n}/\hat{d})$ satisfies

$$\left|s - \frac{n}{d}\right| \leqslant \frac{5}{2}\,\mathrm{ulp}\left(\frac{n}{d}\right). \qquad \square$$

Property 6.1 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.

The bound given by Property 6.1 is very tight. In the general case, assuming RN breaks ties to even, it is asymptotically optimal, as one may check using the generic values

- if p is odd: $n = 2^p + 1$ and $d = 2^p + 2^{(p-1)/2} - 1$;
- if p is even: $n = 2^p + 2^{p/2-1} + 1$ and $d = 2^p + 2^{p/2} - 1$.

Let us quickly detail the case p odd: let $v = 2^{(-p+1)/2}$ so that $v^2 = 2^{-p+1} = 2u$. We have

$$\frac{n}{d} = \frac{1+2^{-p}}{1+2^{(-p-1)/2}-2^{-p}} = \frac{1+v^2/2}{1+v/2-v^2/2}.$$

From $\hat{n} = 2^p$ and $\hat{d} = 2^p + 2^{(p-1)/2}$ we obtain

$$\frac{\hat{n}}{\hat{d}} = \frac{1}{1+v/2} = 1 - v/2 + v^2/4 - v^8/8 + \cdots,$$

from which we obtain $\mathrm{RN}(\hat{n}/\hat{d}) = 1 - v/2$. Hence the error is

$$\begin{aligned}\left|\left(1-\frac{v}{2}\right) - \left(\frac{1+\frac{v^2}{2}}{1+\frac{v}{2}-\frac{v^2}{2}}\right)\right| &= \frac{5}{4}v^2 + O(v^3) \\ &= \left(\frac{5}{2} + O(u^{3/2})\right)\mathrm{ulp}\left(\frac{n}{d}\right).\end{aligned}$$

For instance, consider the computation of $(x+y)/(z+t)$ in binary64 arithmetic, where x, y, z and t are the four following binary64 numbers:

$$\begin{cases} x &= 2^{53}, \\ y &= 1, \\ z &= 2^{53}, \\ t &= 2^{26} - 1. \end{cases}$$

This corresponds to Property 6.1, with $n = x+y$ and $d = z+t$. We have

$$\begin{aligned}\frac{x+y}{z+t} &= \frac{9007199254740993}{9007199321849855} \\ &= 0.999999992549419680631925\cdots\end{aligned}$$

and the computed result is

$$\frac{134217727}{134217728} = 0.99999999254941940307 6\cdots,$$

resulting in an error

$$2.49999997392\cdots\mathrm{ulp}\left(\frac{x+y}{z+t}\right).$$

Interestingly enough, these values of $n = 2^{53} + 1$ and $d = 2^{53} + 2^{26} - 1$ can be written as the product of two binary64 FP numbers, since

$$n = 321. \times 28059810762433.,$$

and

$$d = 34292630015. \times 262657.$$

As a consequence, the same error $2.49999997392\cdots$ ulp is also attained for computations of the form $(xy)/(z+t)$; $(x+y)/(zt)$; and $(xy)/(zt)$ in binary64 arithmetic. This can be generalized to odd values of the precision p, as d can be factored $(2^{(p-1)/2}+1)(2^{1+(p-1)/2}-1)$ and n is a multiple of 3. For other functions, we also found cases where the actual error is very close to the bound 2.5 ulp. For instance, in binary64 floating-point arithmetic ($p = 53$), error 2.4994 ulp is attained when computing

$$\frac{x_1+y_1}{\sqrt{z}}$$

or

$$\frac{x_2y_2}{\sqrt{z}},$$

with

$$\begin{cases} x_1 &= 9007199312857556, \\ y_1 &= 1, \\ x_2 &= 1870953, \\ y_2 &= 4814230669, \\ z &= 4503599859833552, \end{cases}$$

These two cases correspond to Property 6.1 with

$$\begin{aligned} n &= x_1 + y_1 \\ &= x_2y_2 \end{aligned}$$

and

$$d = \sqrt{z}.$$

Conclusion

We have given sharp error bounds in ulps for computations in binary floating-point arithmetic of the form $x \cdot c$, x/c, c/x, $m \cdot n$ and n/d, where x is a floating-point 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

$$\mathtt{x*pi},\ \mathtt{ln(2)/x},\ \mathtt{x/(y+z)},\ \mathtt{(x+y)*z},\ \mathtt{x/sqrt(y)},\ \mathtt{sqrt(x)/y},\ \mathtt{(x+y)(z+t)},\ \mathtt{(x+y)/(z+t)},\ \mathtt{(x+y)/(zt)},$$
etc.

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

Acknowledgement

This work is partly sponsored by the ANR Nuscap (ANR-20-CE48-0014) project of the French Agence Nationale de la Recherche.

References

[1] N. Brisebarre and J.-M. Muller, "Correctly rounded multiplication by arbitrary precision constants," *IEEE Trans. Comput.*, vol. 57, no. 2, pp. 165–174, Feb. 2008.

[2] W. J. Cody, "Implementation and testing of function software," in *Problems and Methodologies in Mathematical Software Production, International Seminar*. Berlin, Heidelberg: Springer-Verlag, 1980, p. 24–47.

[3] IEEE, *IEEE Standard for Floating-Point Arithmetic (IEEE Std 754-2019)*, Jul. 2019. [Online]. Available: https://ieeexplore.ieee.org/servlet/opac?punumber=8766227

[4] S. M. Rump, T. Ogita, and S. Oishi, "Accurate floating-point summation, Part I: Faithful rounding," *SIAM J. Sci. Comput.*, vol. 31, no. 1, pp. 189–224, 2008. [Online]. Available: https://doi.org/10.1137/050645671

[5] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. SIAM, Philadelphia, PA, 2002.

[6] A. Sibidanov, P. Zimmermann, and S. Glondu, "The CORE-MATH project," in *29th IEEE Symposium on Computer Arithmetic*, 2022, preprint available at https://hal.inria.fr/hal-03721525.

[7] S. Boldo, C.-P. Jeannerod, G. Melquiond, and J.-M. Muller, "Floating-point arithmetic," *Acta Numer.*, vol. 32, p. 203–290, 2023.

[8] D. E. Knuth, *The Art of Computer Programming*, 3rd ed. Addison-Wesley, Reading, MA, 1998, vol. 2.

[9] P. H. Sterbenz, *Floating-Point Computation*. Englewood Cliffs, NJ: Prentice-Hall, 1974.

[10] C.-P. Jeannerod and S. M. Rump, "On relative errors of floating-point operations: optimal bounds and applications," *Math. Comp.*, vol. 87, pp. 803–819, 2018.

[11] S. M. Rump, "Error bounds for computer arithmetics," in *26th IEEE Symposium on Computer Arithmetic*, 2019, pp. 1–14.

[12] J.-M. Muller, N. Brunie, F. de Dinechin, C.-P. Jeannerod, M. Joldes, V. Lefèvre, G. Melquiond, N. Revol, and S. Torres, *Handbook of Floating-Point Arithmetic, 2nd edition*. Birkhäuser Boston, 2018, ACM G.1.0; G.1.2; G.4; B.2.0; B.2.4; F.2.1., ISBN 978-3-319-76525-9.

Nicolas Brisebarre was born in Bordeaux, France, in 1971. He received his Ph.D. in pure mathematics from the Université Bordeaux I, France, in 1998. He is *Chargé de recherche* (junior researcher) at CNRS, France, and he is a member of the LIP laboratory (LIP is a joint computer science laboratory of CNRS, École Normale Supérieure de Lyon, INRIA, and Université Claude Bernard Lyon 1). His research interests are in computer arithmetic, number theory, validated computing and computer algebra.

Jean-Michel Muller was born in Grenoble, France, in 1961. He received his Ph.D. degree in 1985 from the Institut National Polytechnique de Grenoble. He is Directeur de Recherches (senior researcher) at CNRS, France, and he is the co-head of GDR-IM. His research interests are in Computer Arithmetic. He is the author of several books, including "Elementary Functions, Algorithms and Implementation" (3rd edition, Birkhauser, 2016), and he coordinated the writing of the "Handbook of Floating-Point Arithmetic" (2nd edition Birkhäuser, 2018). He is currently associate editor in chief of the IEEE Transactions on Emerging Topics in Computing. He is a fellow of the IEEE.

Joris Picot was born in Troyes, France, in 1985. He received his Ph.D. degree in 2013 from the Université de Toulouse. He spent seven years in the developement of computational fluid dynamics software, and he is Research Engineer at École Normale Supérieure de Lyon, France. His research interests are in Rigorous Computing.