# The Classical Relative Error Bounds for Computing $\sqrt{a^2+b^2}$ and $c/\sqrt{a^2+b^2}$ in Binary Floating-Point Arithmetic are Asymptotically Optimal

Claude-Pierre Jeannerod Jean-Michel Muller Antoine Plet

Inria, CNRS, ENS Lyon, Université de Lyon,
Lyon, France

ARITH 24, London, July 2017

# $\sqrt{a^2+b^2}$ and $c/\sqrt{a^2+b^2}$

- basic building blocks of numerical computing: computation of 2D-norms, Givens rotations, etc.;
- radix-2, precision-$p$, FP arithmetic, round-to-nearest, unbounded exponent range;
- Classical analyses: relative error bounded by $2u$ for $\sqrt{a^2+b^2}$, and by $3u+\mathcal{O}(u^2)$ for $c/\sqrt{a^2+b^2}$, where $u=2^{-p}$ is the unit roundoff.
- main results:
  - the $\mathcal{O}(u^2)$ term is not needed;
  - these error bounds are asymptotically optimal;
  - the bounds and their asymptotic optimality remain valid when an FMA is used to evaluate $a^2+b^2$.

# Introduction and notation

- radix-2, precision-$p$ FP number of exponent $e$ and integral significand $|M| \leqslant 2^p - 1$:
$$x = M \cdot 2^{e-p+1}.$$
- $\mathrm{RN}(t)$ is $t$ rounded to nearest, ties-to-even ($\rightarrow \mathrm{RN}(a^2)$ is the result of the FP multiplication `a*a`, assuming the round-to-nearest mode)
- $\mathrm{RD}(t)$ is $t$ rounded towards $-\infty$,
- $u = 2^{-p}$ is the "unit roundoff."
- we have $\mathrm{RN}(t) = t(1+\epsilon)$ with $|\epsilon| \leqslant \frac{u}{1+u} < u$.

# Relative error due to rounding (Knuth)

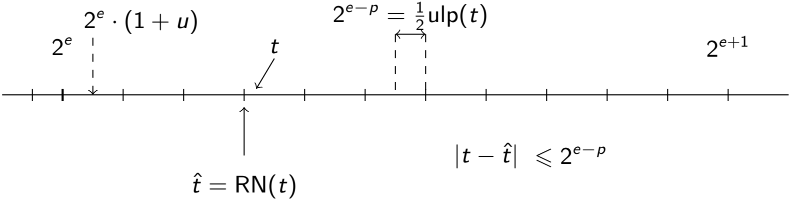
if $2^e \leqslant t < 2^{e+1}$, then $|t - \mathrm{RN}(t)| \leqslant 2^{e-p} = u \cdot 2^e$, and

- if $t \geqslant 2^e \cdot (1+u)$, then $|t - \mathrm{RN}(t)|/t \leqslant u/(1+u)$;
- if $t = 2^e \cdot (1 + \tau \cdot u)$ with $\tau \in [0, 1)$, then $|t - \mathrm{RN}(t)|/t = \tau \cdot u/(1 + \tau \cdot u) < u/(1+u)$,

$\rightarrow$ the maximum relative error due to rounding is bounded by

$$\frac{u}{1+u}.$$

attained $\rightarrow$ no further "general" improvement.



# "Wobbling" relative error

For $t \neq 0$, define (Rump's ufp function)

$$\mathrm{ufp}(t) = 2^{\lfloor \log_2 |t| \rfloor}.$$

We have,

**Lemma 1**

*Let $t \in \mathbb{R}$. If*

$$2^e \leqslant w \cdot 2^e \leqslant |t| < 2^{e+1}, e = \log_2 \mathrm{ufp}(p) \in \mathbb{Z} \tag{1}$$

*(in other words, if $1 \leqslant w \leqslant t/\mathrm{ufp}(t)$) then*

$$\left| \frac{\mathrm{RN}(t) - t}{t} \right| \leqslant \frac{u}{w}.$$

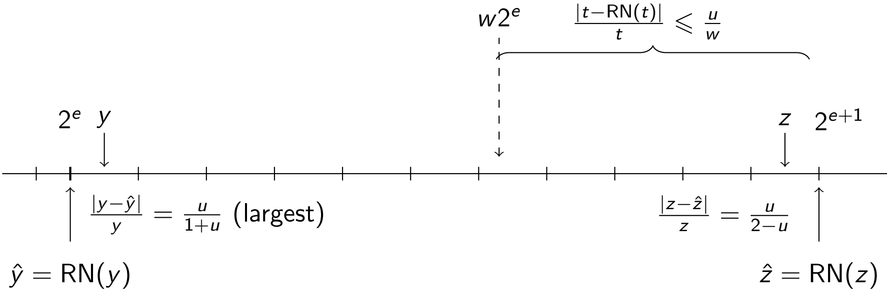



Figure 1: *If we know that* $w \leqslant t/\mathrm{ufp}(t) = t/2^e$, *then* $|\mathrm{RN}(t) - t|/t \leqslant u/w$.

$\rightarrow$ the bound on the relative error of rounding $t$ is largest when $t$ is just above a power of 2.

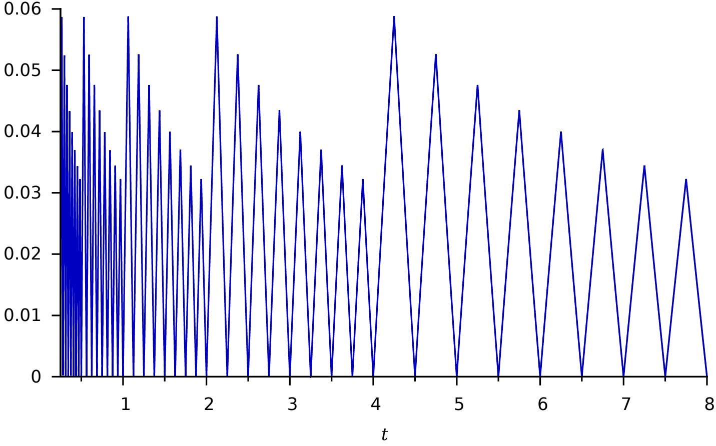



Figure 2: Relative error $|\mathrm{RN}(t) - t|/t$ due to rounding for $\frac{1}{5} \leqslant t \leqslant 8$, and $p = 4$.

# On the quality of error bounds

When giving for some algorithm a relative error bound that is a function $B(p)$ of the precision $p$ (or, equivalently, of $u = 2^{-p}$),

- if there exist FP inputs parameterized by $p$ for which the bound is attained for every $p \geqslant p_0$ , the bound is optimal;
- if there exist some FP inputs parameterized by $p$ and for which the relative error $E(p)$ satisfies $E(p)/B(p) \to 1$ as $p \to \infty$ (or, equivalenty, $u \to 0$), the bound is asymptotically optimal.

If a bound is asymptotically optimal: no need to try to obtain a substantially better bound.

## Computation of $\sqrt{a^2 + b^2}$

**Algorithm 1** Without FMA.

$s_a \leftarrow \mathrm{RN}(a^2)$
$s_b \leftarrow \mathrm{RN}(b^2)$
$s \leftarrow \mathrm{RN}(s_a + s_b)$
$\rho \leftarrow \mathrm{RN}(\sqrt{s})$
**return** $\rho$

**Algorithm 2** With FMA.

$s_b \leftarrow \mathrm{RN}(b^2)$
$s \leftarrow \mathrm{RN}(a^2 + s_b)$
$\rho \leftarrow \mathrm{RN}(\sqrt{s})$
**return** $\rho$

- classical result: relative error of both algorithms $\leqslant 2u + \mathcal{O}(u^2)$
- Jeannerod & Rump (2016): relative error of Algorithm 1 $\leqslant 2u$.
- tight bounds: in binary64 arithmetic, with $a = 1723452922282957/2^{64}$ and $b = 4503599674823629/2^{52}$, both algorithms have relative error $\mathbf{1.9999999}3022\ldots u$.

$\rightarrow$ both algorithms rather equivalent in terms of worst case error;

# Comparing both algorithms ?

- both algorithms rather equivalent in terms of worst case error;
- for $1,000,000$ randomly chosen pairs $(a, b)$ of binary64 numbers with the same exponent, same result in 90.08% of cases; Algorithm 2 (FMA) is more accurate in 6.26% of cases; Algorithm 1 is more accurate in 3.65% of cases;
- for $100,000$ randomly chosen pairs $(a, b)$ of binary64 numbers with exponents satisfying $e_a - e_b = -26$, same result in 83.90% of cases; Algorithm 2 (FMA) is more accurate in 13.79% of cases; Algorithm 1 is more accurate in 2.32% of cases.

$\rightarrow$ Algorithm 2 wins, but not by a big margin.

# Our main result for $\sqrt{a^2+b^2}$

Theorem 2

*For $p \geqslant 12$, there exist floating-point inputs $a$ and $b$ for which the result $\rho$ of Algorithm 1 or Algorithm 2 satisfies*

$$\left|\frac{\rho-\sqrt{a^2+b^2}}{\sqrt{a^2+b^2}}\right| = 2u-\epsilon, \quad |\epsilon| = \mathcal{O}(u^{3/2}).$$

**Consequence:** asymptotic optimality of the relative error bounds.

# Building the “generic” input values $a$ and $b$

(generic: they are given as a function of $p$)

1. We restrict to $a$ and $b$ such that $0 < a < b$.
2. $b$ such that the largest possible absolute error—that is, $(1/2)\mathrm{ulp}(b^2)$—is committed when computing $b^2$. To maximize the relative error, $b^2$ must be slightly above an **even** power of 2.
3. $a$ small enough $\rightarrow$ the computed approximation to $a^2 + b^2$ is slightly above the same power of 2;

   We choose

   - $b = 1 + 2^{-p/2}$ if $p$ is even;
   - $b = 1 + \left\lceil \sqrt{2} \cdot 2^{\frac{p-3}{2}} \right\rceil \cdot 2^{-p+1}$ if $p$ is odd.

Example ($p$ even): $b = 1 + 2^{-p/2}$ gives

$$b^2 = 1 + 2^{-p/2+1} + 2^{-p} \rightarrow \mathrm{RN}(b^2) = 1 + 2^{-p/2+1}.$$

# Building the “generic” input values $a$ and $b$

4. In Algorithm 1, when computing $s_a + s_b$, the significand of $s_a$ is right-shifted by a number of positions equal to the difference of their exponents. Gives the form $s_a$ should have to produce a large relative error.
5. We choose $a =$ square root of that value, adequately rounded.

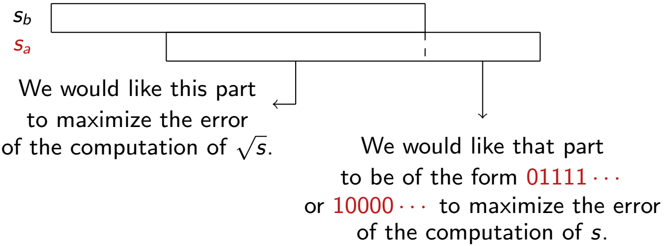



Figure 3: Constructing suitable generic inputs to Algorithms 1 and 2.

# Generic values for $\sqrt{a^2 + b^2}$, for even $p$

$$b = 1 + 2^{-p/2},$$

and

$$a = \mathrm{RD}\left(2^{-\frac{3p}{4}}\sqrt{G}\right),$$

where

$$G = \left\lceil 2^{\frac{p}{2}}\left(\sqrt{2} - 1\right) + \delta \right\rceil \cdot 2^{\frac{p}{2}+1} + 2^{\frac{p}{2}}$$

with

$$\delta = \begin{cases} 1 & \text{if } \left\lceil 2^{\frac{p}{2}}\sqrt{2} \right\rceil \text{ is odd,} \\ 2 & \text{otherwise,} \end{cases} \qquad (2)$$

Table 1: Relative errors of Algorithm 1 or Algorithm 2 for our generic values $a$ and $b$ for various even values of $p$ between 16 and 56.

| $p$ | relative error |
|---|---|
| 16 | $1.\mathbf{9}7519352187392\ldots u$ |
| 20 | $1.\mathbf{99}418559548869\ldots u$ |
| 24 | $1.\mathbf{99}873332158282\ldots u$ |
| 28 | $1.\mathbf{999}67582969338\ldots u$ |
| 32 | $1.\mathbf{9999}0783760560\ldots u$ |
| 36 | $1.\mathbf{9999}7442258505\ldots u$ |
| 40 | $1.\mathbf{99999}449547633\ldots u$ |
| 44 | $1.\mathbf{99999}835799502\ldots u$ |
| 48 | $1.\mathbf{999999}67444005\ldots u$ |
| 52 | $1.\mathbf{999999}89989669\ldots u$ |
| 56 | $1.\mathbf{9999999}7847972\ldots u$ |

# Generic values for $\sqrt{a^2+b^2}$, for odd $p$

We choose

$$b = 1 + \eta,$$

with

$$\eta = \left\lceil \sqrt{2} \cdot 2^{\frac{p-3}{2}} \right\rceil \cdot 2^{-p+1},$$

and

$$a = \mathrm{RN}\left(\sqrt{H}\right),$$

with

$$H = 2^{\frac{-p+3}{2}} - 2\eta - 3 \cdot 2^{-p} + 2^{\frac{-3p+3}{2}}.$$

Table 2: Relative errors of Algorithm 1 or Algorithm 2 for our generic values $a$ and $b$ and for various odd values of $p$ between 53 and 113.

| $p$ | relative error |
|---|---|
| 53 | $1.\mathbf{9999999}188175005308\ldots u$ |
| 57 | $1.\mathbf{9999999}764537355319\ldots u$ |
| 61 | $1.\mathbf{99999999}49811629228\ldots u$ |
| 65 | $1.\mathbf{99999999}88096732861\ldots u$ |
| 69 | $1.\mathbf{999999999}7055095283\ldots u$ |
| 73 | $1.\mathbf{9999999999}181918151\ldots u$ |
| 77 | $1.\mathbf{9999999999}800815518\ldots u$ |
| 81 | $1.\mathbf{99999999999}54499727\ldots u$ |
| 101 | $1.\mathbf{99999999999999}49423\ldots u$ |
| 105 | $1.\mathbf{99999999999999}86669\ldots u$ |
| 109 | $1.\mathbf{999999999999999}6677\ldots u$ |
| 113 | $1.\mathbf{9999999999999999}175\ldots u$ |

# The case of $c/\sqrt{a^2+b^2}$

**Algorithm 3** Without FMA.

$s_a \leftarrow \mathrm{RN}(a^2)$
$s_b \leftarrow \mathrm{RN}(b^2)$
$s \leftarrow \mathrm{RN}(s_a + s_b)$
$\rho \leftarrow \mathrm{RN}(\sqrt{s})$
$g \leftarrow \mathrm{RN}(c/\rho)$
**return** $g$

**Algorithm 4** With FMA.

$s_b \leftarrow \mathrm{RN}(b^2)$
$s \leftarrow \mathrm{RN}(a^2 + s_b)$
$\rho \leftarrow \mathrm{RN}(\sqrt{s})$
$g \leftarrow \mathrm{RN}(c/\rho)$
**return** $g$

Straightforward error analysis: relative error $3u + \mathcal{O}(u^2)$.

Theorem 3

*If $p \neq 3$, then the relative error committed when approximating $c/\sqrt{a^2+b^2}$ by the result $g$ of Algorithm 3 or 4 is less than $3u$.*

# Sketch of the proof

- Previous result on the computation of squares $\rightarrow$ if $p \neq 3$, then $s_a = a^2(1+\epsilon_1)$ and $s_b = b^2(1+\epsilon_2)$ with $|\epsilon_1|, |\epsilon_2| \leqslant \frac{u}{1+3u} =: u_3$;
- $\exists \epsilon_3$ and $\epsilon_4$ such that $|\epsilon_3|, |\epsilon_4| \leqslant \frac{u}{1+u} =: u_1$ and

$$s = \begin{cases} (s_a + s_b)(1+\epsilon_3) & \text{for Algorithm 3,} \\ (a^2 + s_b)(1+\epsilon_4) & \text{for Algorithm 4.} \end{cases}$$

$\rightarrow$ in both cases:

$$(a^2+b^2)(1-u_1)(1-u_3) \leqslant s \leqslant (a^2+b^2)(1+u_1)(1+u_3).$$

- the relative error of division in radix-2 FP arithmetic is at most $u - 2u^2$ (Jeannerod/Rump, 2016), hence

$$g = \frac{c}{\sqrt{s}\,(1+\epsilon_5)}(1+\epsilon_6)$$

with $|\epsilon_5| \leqslant u_1$ and $|\epsilon_6| \leqslant u - 2u^2$.

# Sketch of the proof

- and then

$$\frac{c}{\sqrt{s}} \cdot \frac{1-u+2u^2}{1+u_1} \leqslant g \leqslant \frac{c}{\sqrt{s}} \cdot \frac{1+u-2u^2}{1-u_1}.$$

- Consequently,

$$\zeta \frac{c}{\sqrt{a^2+b^2}} \leqslant g \leqslant \zeta' \frac{c}{\sqrt{a^2+b^2}}$$

with

$$\zeta := \frac{1}{\sqrt{(1+u_1)(1+u_3)}} \cdot \frac{1-u+2u^2}{1+u_1}$$

and

$$\zeta' := \frac{1}{\sqrt{(1-u_1)(1-u_3)}} \cdot \frac{1+u-2u^2}{1-u_1}.$$

To conclude, we check that $1-3u < \zeta$ and $\zeta' < 1+3u$ for all $u \leqslant 1/2$.

# Asymptotic optimality of the bound for $c/\sqrt{a^2+b^2}$

**Theorem 4**

*For $p \geqslant 12$, there exist floating-point inputs $a$, $b$, and $c$ for which the result $g$ returned by Algorithm 3 or Algorithm 4 satisfies*

$$\left| \frac{g - \frac{c}{\sqrt{a^2+b^2}}}{\frac{c}{\sqrt{a^2+b^2}}} \right| = 3u - \epsilon, \quad |\epsilon| = \mathcal{O}(u^{3/2}).$$

The "generic" values of $a$ and $b$ used to prove Theorem 4 are the same as the ones we have chosen for $\sqrt{a^2+b^2}$, and we use

$$c = \begin{cases} 1 + 2^{-p+1} \cdot \left\lfloor 3\sqrt{2} \cdot 2^{p/2-2} \right\rceil & \text{(even } p\text{)}, \\ 1 + 3 \cdot 2^{\frac{-p-1}{2}} + 2^{-p+1} & \text{(odd } p\text{)}. \end{cases}$$

Table 3: Relative errors obtained, for various precisions $p$, when running Algorithm 3 or Algorithm 4 with our generic values $a$, $b$, and $c$.

| $p$ | relative error |
| ---: | --- |
| 24 | $2.\mathbf{99}800258913676259676349 8\ldots u$ |
| 53 | $2.\mathbf{999999}896465758351542169\ldots u$ |
| 64 | $2.\mathbf{99999999}7359196820010396\ldots u$ |
| 113 | $2.\mathbf{9999999999999999}896692295\ldots u$ |
| 128 | $2.\mathbf{9999999999999999999}566038\ldots u$ |

# Conclusion

- we have reminded the relative error bound $2u$ for $\sqrt{a^2+b^2}$, slightly improved the bound ($3u+\mathcal{O}(u^2) \to 3u$) for $c/\sqrt{a^2+b^2}$, and considered variants that take advantage of the possible availability of an FMA,
- asymptotically optimal bounds $\to$ trying to significantly refine them further is hopeless.
- Unbounded exponent range $\to$ our results hold provided that no underflow or overflow occurs.
- handling "spurious" overflows and underflows: using an exception handler and/or scaling the input values:
  - if the scaling introduces rounding errors, then our bounds may not hold anymore;
  - if $a$ and $b$ (and $c$) are scaled by a power of 2, our analyses still apply.