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 \( |M| \leq 2^p - 1 \):
  \[ x = M \cdot 2^{e-p+1}. \]

- \( \text{RN}(t) \) is \( t \) rounded to nearest, ties-to-even (\( \rightarrow \text{RN}(a^2) \) is the result of the FP multiplication \( a \cdot a \), assuming the round-to-nearest mode)

- \( \text{RD}(t) \) is \( t \) rounded towards \(-\infty\),

- \( u = 2^{-p} \) is the “unit roundoff.”

- we have \( \text{RN}(t) = t(1 + \epsilon) \) with \( |\epsilon| \leq \frac{u}{1+u} < u \).
Relative error due to rounding (Knuth)

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

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

→ the maximum relative error due to rounding is bounded by

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

attained → no further “general” improvement.

$2^e \cdot (1 + u)$

$2^{e+1}$

$2^e$
“Wobbling” relative error

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

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

We have,

**Lemma 1**

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

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

*(in other words, if $1 \leq w \leq t/ufp(t))$ then*

$$\left| \frac{\text{RN}(t) - t}{t} \right| \leq \frac{u}{w}.$$
Figure 1: If we know that $w \leq t/\text{ufp}(t) = t/2^e$, then $|\text{RN}(t) - t|/t \leq u/w$. 

→ the bound on the relative error of rounding $t$ is largest when $t$ is just above a power of 2.
Figure 2: Relative error $|RN(t) - t|/t$ due to rounding for $\frac{1}{5} \leq t \leq 8$, 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 $p \geq p_0$, 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, equivalently, $u \to 0$), 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.

\[
\begin{align*}
& s_a \leftarrow \text{RN}(a^2) \\
& s_b \leftarrow \text{RN}(b^2) \\
& s \leftarrow \text{RN}(s_a + s_b) \\
& \rho \leftarrow \text{RN}(\sqrt{s}) \\
\text{return} & \quad \rho
\end{align*}
\]

**Algorithm 2** With FMA.

\[
\begin{align*}
& s_b \leftarrow \text{RN}(b^2) \\
& s \leftarrow \text{RN}(a^2 + s_b) \\
& \rho \leftarrow \text{RN}(\sqrt{s}) \\
\text{return} & \quad \rho
\end{align*}
\]

- classical result: relative error of both algorithms $\leq 2u + O(u^2)$
- Jeannerod & Rump (2016): relative error of Algorithm 1 $\leq 2u$.
- **tight bounds**: in binary64 arithmetic, with
  \[a = 1723452922282957/2^{64} \text{ and } b = 4503599674823629/2^{52}\], both
  algorithms have relative error $1.99999993022\ldots u$.
- $\rightarrow$ 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.

→ Algorithm 2 wins, but not by a big margin.
Our main result for $\sqrt{a^2 + b^2}$

**Theorem 2**

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

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

**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, $(1/2)\text{ulp}(b^2)$—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 → 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;
- $b = 1 + \left\lfloor \sqrt{2} \cdot 2^{\frac{p-3}{2}} \right\rfloor \cdot 2^{-p+1}$ if $p$ is odd.

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

$$b^2 = 1 + 2^{-p/2+1} + 2^{-p} \rightarrow \text{RN}(b^2) = 1 + 2^{-p/2+1}.$$
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 $a = \text{square root of that value, adequately rounded.}$

We would like this part to maximize the error of the computation of $\sqrt{s}$. We would like that part to be of the form $01111\cdots$ or $10000\cdots$ to maximize the error of the computation of $s$.

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

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

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

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

(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.

<table>
<thead>
<tr>
<th>( p )</th>
<th>relative error</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1.97519352187392 \ldots u</td>
</tr>
<tr>
<td>20</td>
<td>1.99418559548869 \ldots u</td>
</tr>
<tr>
<td>24</td>
<td>1.99873332158282 \ldots u</td>
</tr>
<tr>
<td>28</td>
<td>1.99967582969338 \ldots u</td>
</tr>
<tr>
<td>32</td>
<td>1.99990783760560 \ldots u</td>
</tr>
<tr>
<td>36</td>
<td>1.99997442258505 \ldots u</td>
</tr>
<tr>
<td>40</td>
<td>1.99999449547633 \ldots u</td>
</tr>
<tr>
<td>44</td>
<td>1.99999835799502 \ldots u</td>
</tr>
<tr>
<td>48</td>
<td>1.99999967444005 \ldots u</td>
</tr>
<tr>
<td>52</td>
<td>1.99999989989669 \ldots u</td>
</tr>
<tr>
<td>56</td>
<td>1.99999997847972 \ldots u</td>
</tr>
</tbody>
</table>
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

$$a = \text{RN}(\sqrt{H}),$$

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.

<table>
<thead>
<tr>
<th>$p$</th>
<th>relative error</th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
<td>1.99999999188175005308...u</td>
</tr>
<tr>
<td>57</td>
<td>1.99999999764537355319...u</td>
</tr>
<tr>
<td>61</td>
<td>1.9999999949811629228...u</td>
</tr>
<tr>
<td>65</td>
<td>1.9999999988096732861...u</td>
</tr>
<tr>
<td>69</td>
<td>1.999999997055095283...u</td>
</tr>
<tr>
<td>73</td>
<td>1.99999999181918151...u</td>
</tr>
<tr>
<td>77</td>
<td>1.99999999800815518...u</td>
</tr>
<tr>
<td>81</td>
<td>1.9999999954499727...u</td>
</tr>
<tr>
<td>101</td>
<td>1.9999999949423...u</td>
</tr>
<tr>
<td>105</td>
<td>1.9999999986669...u</td>
</tr>
<tr>
<td>109</td>
<td>1.999999996677...u</td>
</tr>
<tr>
<td>113</td>
<td>1.99999999175...u</td>
</tr>
</tbody>
</table>
The case of $c/\sqrt{a^2 + b^2}$

**Algorithm 3** Without FMA.

\[
\begin{align*}
    s_a & \leftarrow \text{RN}(a^2) \\
    s_b & \leftarrow \text{RN}(b^2) \\
    s & \leftarrow \text{RN}(s_a + s_b) \\
    \rho & \leftarrow \text{RN}(\sqrt{s}) \\
    g & \leftarrow \text{RN}(c/\rho) \\
    \text{return } & g
\end{align*}
\]

**Algorithm 4** With FMA.

\[
\begin{align*}
    s_b & \leftarrow \text{RN}(b^2) \\
    s & \leftarrow \text{RN}(a^2 + s_b) \\
    \rho & \leftarrow \text{RN}(\sqrt{s}) \\
    g & \leftarrow \text{RN}(c/\rho) \\
    \text{return } & g
\end{align*}
\]

Straightforward error analysis: relative error $3u + O(u^2)$.

**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 $|\epsilon_1|, |\epsilon_2| \leq \frac{u}{1+3u} =: u_3$;
- $\exists \epsilon_3$ and $\epsilon_4$ such that $|\epsilon_3|, |\epsilon_4| \leq \frac{u}{1+u} =: u_1$ 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:

\[
(a^2 + b^2)(1 - u_1)(1 - u_3) \leq s \leq (a^2 + b^2)(1 + u_1)(1 + u_3).
\]

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

Sketch of the proof

and then

\[ \frac{c}{\sqrt{s}} \cdot \frac{1 - u + 2u^2}{1 + u_1} \leq g \leq \frac{c}{\sqrt{s}} \cdot \frac{1 + u - 2u^2}{1 - u_1}. \]

Consequently,

\[ \zeta \frac{c}{\sqrt{a^2 + b^2}} \leq g \leq \zeta' \frac{c}{\sqrt{a^2 + b^2}} \]

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 \( u \leq 1/2 \).
Asymptotic optimality of the bound for $c/\sqrt{a^2 + b^2}$

**Theorem 4**

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

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

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

$$c = \begin{cases} 
1 + 2^{-p+1} \cdot \left[ 3\sqrt{2} \cdot 2^{p/2-2} \right] & \text{(even } p), \\
1 + 3 \cdot 2^{-\frac{p-1}{2}} + 2^{-p+1} & \text{(odd } p). 
\end{cases}$$
Table 3: Relative errors obtained, for various precisions $p$, when running Algorithm 3 or Algorithm 4 with our generic values $a$, $b$, and $c$.

<table>
<thead>
<tr>
<th>$p$</th>
<th>relative error</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>2.998002589136762596763498...$u$</td>
</tr>
<tr>
<td>53</td>
<td>2.999999896465758351542169...$u$</td>
</tr>
<tr>
<td>64</td>
<td>2.99999997359196820010396...$u$</td>
</tr>
<tr>
<td>113</td>
<td>2.9999999999999999896692295...$u$</td>
</tr>
<tr>
<td>128</td>
<td>2.99999999999999999566038...$u$</td>
</tr>
</tbody>
</table>
we have reminded the relative error bound $2u$ for $\sqrt{a^2 + b^2}$, slightly improved the bound $(3u + O(u^2) \rightarrow 3u)$ for $c/\sqrt{a^2 + b^2}$, and considered variants that take advantage of the possible availability of an FMA,

- asymptotically optimal bounds \( \rightarrow \) trying to significantly refine them further is hopeless.

- Unbounded exponent range \( \rightarrow \) 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.