Error bounds that are certain, sharp. . . and whose proof is trustable: the curse of long and boring proofs

Jean-Michel Muller
with contributions from M. Joldes, V. Popescu, L. Rideau, B. Salvy

CNRS - Laboratoire LIP

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

What's the point in all this?

We wish to prove error bounds of medium-size, "atomic" algorithms in FP arithmetic, but. . .

- error bounds. . . *for what purpose?*
- proofs. . . *for what purpose?*

Context: base 2, precision-p FP arithmetic

A Floating-Point number (FPN) x is represented by two integers:

- Floating-Point number:

$$x = \left(\frac{M}{2^{p-1}}\right) \cdot 2^e = m_0.m_1 m_2 \cdots m_{p-1} \cdot 2^e$$

  where $M, e \in \mathbb{Z}$, with $|M| \leq 2^p - 1$ and $e_{\min} \leq e \leq e_{\max}$. Additional requirement: e smallest under these constraints.
- x is normal if $|x| \geq 2^{e_{\min}}$ (implies $|M| \geq 2^{p-1}$, i.e., $m_0 = 1$);
- x is subnormal otherwise ($m_0 = 0$);
- largest finite FPN $\Omega = 2^{e_{\max}+1} - 2^{e_{\max}-p+1}$;
- rounding unit: $u = 2^{-p}$.

Error bounds. . .

- FP system parametered by precision p or unit round-off $u = 2^{-p}$;
- for a given algorithm, we consider an (absolute or relative) error bound $\mathcal{B}(u)$;
- the (most likely unknown) worst case error is $\mathcal{W}(u)$.

The bound $\mathcal{B}$ is

- certain (for $u \leq u_0$) if $\mathcal{W}(u) \leq \mathcal{B}(u)$ for $u \leq u_0$;
- asymptotically optimal if $\mathcal{W}(u)/\mathcal{B}(u) \rightarrow 1$ as $u \rightarrow 0$;
- tight (for $u \leq u_0$) if $\mathcal{W}(u)$ is close to $\mathcal{B}(u)$ for $u \leq u_0$.

Example: bounds of the form $\alpha u + \mathcal{O}(u^2)$, frequent in numerical analysis, are not certain, they do not allow to guarantee that the error is less than some clearly given value (and with 8-bit FP formats, with $p = 2$ or 3, high-order terms are not negligible!)

Error bounds (in FP arithmetic)... for what purpose ?

- choice between different algorithms:
    - an informed choice of the algorithm that has the best balance performance/accuracy requires tight bounds;
    - certainty not that important;
- guaranteeing the behavior of a possibly critical software:
    - need to prove that the error is not $\geq$ some threshold

    $\rightarrow$ certainty important,
    - tightness not always needed;

Error bounds (in FP arithmetic)... for what purpose ?

- careful implementation optimization: arithmetic in "small" FP formats faster than in "large" formats $\rightarrow$ use small formats whenever possible.
  - small formats $\rightarrow$ much larger rounding errors $\rightarrow$ careful analysis needed. Hence Tight error bounds are preferable,
  - however, very small underestimation is rarely a problem;
- fully validated set of "atomic" algorithms:
  - the most common transcendental functions such as exp, ln;
  - simple algebraic functions such as $1/\sqrt{x}$, $\mathrm{hypot}(x, y) = \sqrt{x^2 + y^2}$

  are "basic building blocks" of numerical computing : users expect same behavior as for $+$, $-$, $\times$, $\div$, $\sqrt{\cdot}$.

$\rightarrow$ having bounds that are both certain and tight is desirable.

Error bounds... an example with $\sqrt{x^2+y^2}$

if $|x| < |y|$ **then**
  swap (x, y)
end if
$r \leftarrow \mathrm{RN}\,(y/x)$
$t \leftarrow \mathrm{RN}\,(1 + r^2)$
$s \leftarrow \mathrm{RN}\,(\sqrt{t})$
$\rho_2 = \mathrm{RN}\,(|x| \cdot s)$

Bounds (assuming $u \leq 1/4$):

- straightforward bound: $\mathcal{B}_0(u) = \frac{7}{2}u + \mathcal{O}(u^2)$;
- easily obtained relative error bound: $\mathcal{B}_1(u) = 3u$;
- with more care: $\mathcal{B}_2(u) = \frac{5}{2}u + \mathcal{O}(u^2)$;
- with much more care: $\mathcal{B}_3(u) = \frac{5}{2}u + \frac{3}{8}u^2$.

$\mathcal{B}_1$ is certain but not sharp, $\mathcal{B}_2$ is asymptotically optimal but not certain, $\mathcal{B}_3$ is asymptotically optimal & certain.

The proof of bound $\mathcal{B}_3$ is muuuuuch longer than the proof of bound $\mathcal{B}_1$.
This seems general: tightness has a cost.

Proofs... for what purpose ?

1. to check, by following the proof step by step, that the claimed property holds;
2. to have a deep understanding of what is behind the claimed property.

Rather antagonistic goals:

- goal 1 requires many details,
- goal 2 needs a focus on the "big things" (hence many "*without loss of generality...*" or "*the second case is similar*").

In general our "paper proofs" are in between: is this the right solution?

The two examples considered in this talk

- “double word” arithmetic: formal proofs helped to
  - strengthen claimed results,
  - improve them,
  - find (hmmm. . . embarassing) bugs.
- hypotenuse function $\sqrt{x^2 + y^2}$: computer algebra helped to
  - obtain tight bounds,
  - explore several variants.

Before presenting that: additional notions on FP arithmetic (roundings, error-free transforms, double-word arithmetic).

Correct rounding

- the sum, product, ... of two FP numbers is not, in general, a FP number $\rightarrow$ must be rounded;
- the IEEE 754 Std for FP arithmetic specifies several rounding functions;
- the default function is RN ties to even.

Correctly rounded operation: returns what we would get by exact operation followed by rounding.

- correctly rounded $+$, $-$, $\times$, $\div$, $\sqrt{\cdot}$ are required;
- correctly rounded sin, cos, exp, ln, $1/\sqrt{x}$, $\sqrt{x^2+y^2}$, etc. only recommended (not mandatory).

$\rightarrow$ when `c = a + b` appears in a program, we get $c = \mathrm{RN}(a+b)$.

ulp ("unit in last place") and "absolute error" due to rounding

Definition (ulp function)

If $|x| \in [2^e, 2^{e+1})$, then $\operatorname{ulp}(x) = 2^{\max\{e, e_{\min}\}-p+1}$.

It is the distance between consecutive FP numbers in the neighborhood of x.

Properties:

- $|x - \operatorname{RN}(x)| \leq \frac{1}{2}\operatorname{ulp}(x)$;
- if x is a FP number then it is an integer multiple of $\operatorname{ulp}(x)$.

Frequently used for expressing errors of atomic functions.

Relative error due to rounding

- if x is in the normal range (i.e., $2^{e_{\min}} \leq |x| \leq \Omega$), then

$$|x - \mathrm{RN}(x)| \leq \frac{1}{2} \mathrm{ulp}(x) = 2^{\lfloor \log_2 |x| \rfloor - p},$$

therefore,

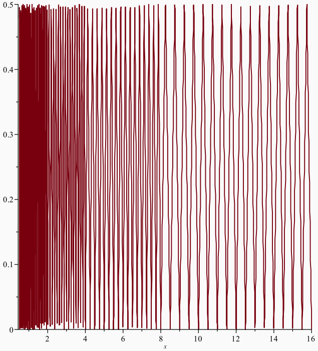
$$|x - \mathrm{RN}(x)| \leq u \cdot |x|, \tag{1}$$

with $u = 2^{-p} = \frac{1}{2} \mathrm{ulp}(1)$. Hence the relative error

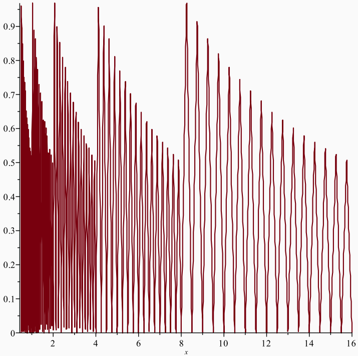
$$\frac{|x - \mathrm{RN}(x)|}{|x|}$$

(for $x \neq 0$) is $\leq u$.

- u, called unit round-off is frequently used for expressing errors.
- (1): u can be replaced by $\frac{u}{1+u}$ (attained for $x = 1 + u$).



Absolute error (in ulps) of rounding to nearest a real number $x \in [1/2, 16]$, assuming a binary FP "toy" system with $p = 5$.



Relative error (in multiples of $u = 2^{-p}$) of rounding to nearest a real number $x \in [1/2, 16]$, assuming a binary FP "toy" system with $p = 5$.

The relative error bound u is tight only slightly above a power of 2.

Error-free transforms and double-word arithmetic

2Sum(a, b)

$s \leftarrow \mathrm{RN}\,(a+b)$
$a' \leftarrow \mathrm{RN}\,(s-b)$
$b' \leftarrow \mathrm{RN}\,(s-a')$
$\delta_a \leftarrow \mathrm{RN}\,(a-a')$
$\delta_b \leftarrow \mathrm{RN}\,(b-b')$
$t \leftarrow \mathrm{RN}\,(\delta_a+\delta_b)$
return (s,t)

Fast2Sum(a, b)

$s \leftarrow \mathrm{RN}\,(a+b)$
$z \leftarrow \mathrm{RN}\,(s-a)$
$t \leftarrow \mathrm{RN}\,(b-z)$
return (s,t)

Barring overflow:

- the pair (s,t) returned by 2Sum satisfies $s = \mathrm{RN}\,(a+b)$ and $t = (a+b)-s$;
- if $|a| \geq |b|$ then the pair (s,t) returned by Fast2Sum satisfies $s = \mathrm{RN}\,(a+b)$ and $t = (a+b)-s$.

Such algorithms: Error-free transforms.

Error-free transforms and double-word arithmetic

2Prod(a, b)

$\pi \leftarrow \mathrm{RN}(ab)$
$\rho \leftarrow \mathrm{RN}(ab - \pi)$
return (π, ρ)

Barring overflow, if the exponents e_a and e_b of a and b satisfy $e_a + e_b \geq e_{\min} + p - 1$ then then the pair (π, ρ) returned by Fast2Sum satisfies $\pi = \mathrm{RN}(ab)$ and $\rho = (ab) - \pi$.

- Fast2Sum, 2Sum and 2Prod: return x represented by a pair (x_h, x_ℓ) of FPN such that $x_h = \mathrm{RN}(x)$ and $x = x_h + x_\ell$;
- Such pairs: double-word numbers (DW).

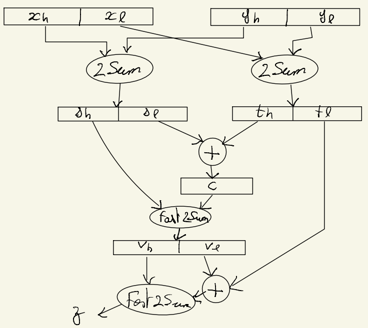
Algorithms for manipulating DW suggested by various authors since 1971.

DW+DW: “accurate version”

Sum of two DW numbers. There also exists a “quick & dirty” algorithm, but its relative error is unbounded.

DWPlusDW

1: $(s_h, s_\ell) \leftarrow \text{2Sum}(x_h, y_h)$
2: $(t_h, t_\ell) \leftarrow \text{2Sum}(x_\ell, y_\ell)$
3: $c \leftarrow \text{RN}\,(s_\ell + t_h)$
4: $(v_h, v_\ell) \leftarrow \text{Fast2Sum}(s_h, c)$
5: $w \leftarrow \text{RN}\,(t_\ell + v_\ell)$
6: $(z_h, z_\ell) \leftarrow \text{Fast2Sum}(v_h, w)$
7: **return** (z_h, z_ℓ)



DW+DW: “accurate version”

We have (after a rather tedious proof):

Theorem (Joldeș, Popescu, M., 2017)

If $p \geq 3$, the relative error of Algorithm DWPlusDW is bounded by

$$\frac{3u^2}{1-4u} = 3u^2 + 12u^3 + 48u^4 + \cdots, \tag{2}$$

That theorem has an interesting history…

ALGORITHM 6: – AccurateDWPlusDW$(x_h, x_\ell, y_h, y_\ell)$. Calculation of $(x_h, x_\ell) + (y_h, y_\ell)$ in binary, precision-p, floating-point arithmetic.

```
1: (s_h, s_ℓ) ← 2Sum(x_h, y_h)
2: (t_h, t_ℓ) ← 2Sum(x_ℓ, y_ℓ)
3: c ← RN(s_ℓ + t_h)
4: (v_h, v_ℓ) ← Fast2Sum(s_h, c)
5: w ← RN(t_ℓ + v_ℓ)
6: (z_h, z_ℓ) ← Fast2Sum(v_h, w)
7: return (z_h, z_ℓ)
```

Li et al. (2000, 2002) claim that in binary64 arithmetic ($p = 53$) the relative error of Algorithm 6 is upper bounded by $2 \cdot 2^{-106}$. This bound is incorrect, as shown by the following example: If

$$\begin{aligned} x_h &= 9007199254740991, \\ x_\ell &= -9007199254740991/2^{54}, \\ y_h &= -9007199254740987/2, \text{ and} \\ y_\ell &= -9007199254740991/2^{56}, \end{aligned} \tag{2}$$

then the relative error of Algorithm 6 is

$$2.24999999999999956\cdots \times 2^{-106}.$$

Note that this example is somehow "generic": In precision-p FP arithmetic, the choice $x_h = 2^p - 1$, $x_\ell = -(2^p - 1) \cdot 2^{-p-1}$, $y_h = -(2^p - 5)/2$, and $y_\ell = -(2^p - 1) \cdot 2^{-p-3}$ leads to a relative error that is asymptotically equivalent (as p goes to infinity) to $2.25u^2$.

Now let us try to find a relative error bound. We are going to show the following result.

Theorem 3.1. *If $p \geq 3$, then the relative error of Algorithm 6 (AccurateDWPlusDW) is bounded by*

$$\frac{3u^2}{1 - 4u} = 3u^2 + 12u^3 + 48u^4 + \cdots, \tag{3}$$

which is less than $3u^2 + 13u^3$ as soon as $p \geq 6$.

Note that the conditions on p ($p \geq 3$ for the bound (3) to hold, $p \geq 6$ for the simplified bound $3u^2 + 13u^3$) are satisfied in all practical cases.

Proof. First, we exclude the straightforward case in which one of the operands is zero. We can also quickly proceed with the case $x_h + y_h = 0$: The returned result is $2\mathrm{Sum}(x_\ell, y_\ell)$, which is equal to $x + y$, that is, the computation is errorless. Now, without loss of generality, we assume $1 \leq x_h < 2$, $x \geq |y|$ (which implies $x_h \geq |y_h|$), and $x_h + y_h$ nonzero. Notice that $1 \leq x_h < 2$ implies $1 \leq x_h \leq 2 - 2u$, since x_h is a FP number.

Define ϵ_1 as the error committed at Line 3 of the algorithm:

$$\epsilon_1 = c - (s_\ell + t_h) \tag{4}$$

and ϵ_2 as the error committed at Line 5:

$$\epsilon_2 = w - (t_\ell + v_\ell). \tag{5}$$

1. If $-x_h < y_h \leq -x_h/2$. Sterbenz Lemma, applied to the first line of the algorithm, implies $s_h = x_h + y_h$, $s_\ell = 0$, and $c = \mathrm{RN}(t_h) = t_h$.

Define

$$\sigma = \begin{cases} 2 \text{ if } y_h \leq -1, \\ 1 \text{ if } -1 < y_h \leq -x_h/2. \end{cases}$$

We have $-x_h < y_h \leq (1 - \sigma) + \frac{x_h}{2}(\sigma - 2)$, so $0 \leq x_h + y_h \leq 1 + \sigma \cdot (\frac{x_h}{2} - 1) \leq 1 - \sigma u$. Also, since x_h is a multiple of $2u$ and y_h is a multiple of σu, $s_h = x_h + y_h$ is a multiple of σu. Since s_h is nonzero, we finally obtain

$$\sigma u \leq s_h \leq 1 - \sigma u. \tag{6}$$

We have $|x_\ell| \leq u$ and $|y_\ell| \leq \frac{\sigma}{2} u$, so

$$|t_h| \leq \left(1 + \frac{\sigma}{2}\right) u \quad \text{and} \quad |t_\ell| \leq u^2. \tag{7}$$

From Equation (6), we deduce that the floating-point exponent of s_h is at least $-p + \sigma - 1$. From Equation (7), the floating-point exponent of $c = t_h$ is at most $-p + \sigma - 1$. Therefore, the Fast2Sum algorithm introduces no error at line 4 of the algorithm, which implies

$$v_h + v_\ell = s_h + c = s_h + t_h = x + y - t_\ell.$$

Equations (6) and (7) imply

$$|s_h + t_h| \leq 1 + \left(1 - \frac{\sigma}{2}\right) u \leq 1 + \frac{u}{2},$$

so $|v_h| \leq 1$ and $|v_\ell| \leq \frac{u}{2}$. From the bounds on $|t_\ell|$ and $|v_\ell|$, we obtain:

$$|\epsilon_2| \leq \frac{1}{2}\mathrm{ulp}(t_\ell + v_\ell) \leq \frac{1}{2}\mathrm{ulp}\left(u^2 + \frac{u}{2}\right) = \frac{u^2}{2} \tag{8}$$

and

$$|\epsilon_2| \leq \frac{1}{2}\mathrm{ulp}\left[\frac{1}{2}\mathrm{ulp}(x_\ell + y_\ell) + \frac{1}{2}\mathrm{ulp}\left((x + y) + \frac{1}{2}\mathrm{ulp}(x_\ell + y_\ell)\right)\right]. \tag{9}$$

Lemma 2.1 and $|s_h| \geq \sigma u$ imply that either $s_h + t_h = 0$, or $|v_h| = |\mathrm{RN}(s_h + c)| = |\mathrm{RN}(s_h + t_h)| \geq \sigma u^2$. If $s_h + t_h = 0$, then $v_h = v_\ell = 0$ and the sequel of the proof is straightforward. Therefore, in the following, we assume $|v_h| \geq \sigma u^2$.

Now,

- If $|v_h| = \sigma u^2$, then $|v_\ell + t_\ell| \leq u|v_h| + u^2 = \sigma u^3 + u^2$, which implies $|w| = |\mathrm{RN}(t_\ell + v_\ell)| \leq \sigma u^2 = |v_h|$;
- If $|v_h| > \sigma u^2$, then, since v_h is a FP number, $|v_h|$ is larger than or equal to the FP number immediately above σu^2, which is $\sigma(1 + 2u)u^2$. Hence $|v_h| \geq \sigma u^2/(1 - u)$, so $|v_h| \geq u \cdot |v_h| + \sigma u^2 \geq |v_\ell| + |t_\ell|$. So, $|w| = |\mathrm{RN}(t_\ell + v_\ell)| \leq |v_h|$.

Therefore, in all cases, Fast2Sum introduces no error at line 6 of the algorithm, and we have

$$z_h + z_\ell = v_h + w = x + y + \epsilon_2. \tag{10}$$

Directly using Equation (10) and the bound $u^2/2$ on $|\epsilon_2|$ to get a relative error bound would result in a large bound, because $x + y$ may be small. However, when $x + y$ is very small, some simplification occurs thanks to Sterbenz Lemma. First, $x_h + y_h$ is a nonzero multiple of σu. Hence, since $|x_\ell + y_\ell| \leq (1 + \frac{\sigma}{2})u$, we have $|x_\ell + y_\ell| \leq \frac{3}{2}(x_h + y_h)$. Let us now consider the two possible cases:

- If $-\frac{3}{2}(x_h + y_h) \leq x_\ell + y_\ell \leq -\frac{1}{2}(x_h + y_h)$, which implies $-\frac{3}{2}s_h \leq t_h \leq -\frac{1}{2}s_h$, then Sterbenz lemma applies to the floating-point addition of s_h and $c = t_h$. Therefore line 4 of the algorithm results in $v_h = s_h$ and $v_\ell = 0$. An immediate consequence is $\epsilon_2 = 0$, so $z_h + z_\ell = v_h + w = x + y$: the computation of $x + y$ is errorless;

- If $-\frac{1}{2}(x_h + y_h) < x_\ell + y_\ell \le \frac{3}{2}(x_h + y_h)$, then $\frac{5}{2}(x_\ell + y_\ell) \le \frac{3}{2}(x_h + y_h + x_\ell + y_\ell) = \frac{3}{2}(x+y)$, and $-\frac{1}{2}(x+y) < \frac{1}{2}(x_\ell + y_\ell)$. Hence, $|x_\ell + y_\ell| < |x+y|$, so $\mathrm{ulp}(x_\ell + y_\ell) \le \mathrm{ulp}(x+y)$. Combined with Equation (9), this gives

$$|\epsilon_2| \le \frac{1}{2}\mathrm{ulp}\left(\frac{3}{2}\mathrm{ulp}(x+y)\right) \le 2^{-p}\mathrm{ulp}(x+y) \le 2 \cdot 2^{-2p} \cdot (x+y).$$

2. If $-x_h/2 < y_h \le x_h$

Notice that we have $x_h/2 < x_h + y_h \le 2x_h$, so $x_h/2 \le s_h \le 2x_h$. Also notice that we have $|x_\ell| \le u$.

- If $\frac{1}{2} < x_h + y_h \le 2 - 4u$. Define

$$\sigma = \begin{cases} 1 \text{ if } x_h + y_h \le 1 - 2u, \\ 2 \text{ if } 1 - 2u < x_h + y_h \le 2 - 4u. \end{cases}$$

We have

$$\frac{\sigma}{2}(1-2u) \le s_h \le \sigma(1-2u) \quad \text{and} \quad |s_\ell| \le \frac{\sigma}{2}u. \tag{11}$$

When $\sigma = 1$, we necessarily have $-x_h/2 < y_h < 0$, so $|y_\ell| \le u/2$. And when $\sigma = 2$, $|y_h| \le x_h \le 2 - 2u$ implies $|y_\ell| \le u$. Hence we always have $|y_\ell| \le \frac{\sigma}{2}u$. This implies $|x_\ell + y_\ell| \le (1 + \sigma/2)u$, therefore

$$|t_h| \le \left(1 + \frac{\sigma}{2}\right)u \quad \text{and} \quad |t_\ell| \le u^2. \tag{12}$$

Now, $|s_\ell + t_h| \le (1+\sigma)u$, so

$$|c| \le (1+\sigma)u \quad \text{and} \quad |\epsilon_1| \le \sigma u^2. \tag{13}$$

Since $s_h \ge 1/2$ and $|c| \le 3u$, if $p \ge 3$, then Algorithm Fast2Sum introduces no error at line 4 of the algorithm, that is,

$$v_h + v_\ell = s_h + c.$$

Therefore $|v_h + v_\ell| = |s_h + c| \le \sigma(1-2u) + (1+\sigma)u \le \sigma$. This implies

$$|v_h| \le \sigma \quad \text{and} \quad |v_\ell| \le \frac{\sigma}{2}u. \tag{14}$$

Thus $|t_\ell + v_\ell| \le u^2 + \frac{\sigma}{2}u$, so

$$|w| \le \frac{\sigma}{2}u + u^2 \quad \text{and} \quad |\epsilon_2| \le \frac{\sigma}{2}u^2. \tag{15}$$

From Equations (11) and (13), we deduce $s_h + c \ge \frac{\sigma}{2} - u(2\sigma + 1)$, so $|v_h| \ge \frac{\sigma}{2} - u(2\sigma+1)$. If $p \ge 3$, then $|v_h| \ge |w|$, so Algorithm Fast2Sum introduces no error at line 6 of the algorithm, that is, $z_h + z_\ell = v_h + w$.

Therefore,

$$z_h + z_\ell = x + y + \eta,$$

with $|\eta| = |\epsilon_1 + \epsilon_2| \le \frac{3\sigma}{2}u^2$. Since

$$x + y \ge (x_h - u) + (y_h - u/2) > \begin{cases} \frac{1}{2} - \frac{3}{2}u & \text{if} \quad \sigma = 1, \\ 1 - 4u & \text{if} \quad \sigma = 2, \end{cases}$$

the relative error $|\eta|/(x+y)$ is upper bounded by

$$\frac{3u^2}{1-4u}.$$

- If $2 - 4u < x_h + y_h \le 2x_h$, then $2 - 4u \le s_h \le \mathrm{RN}(2x_h) = 2x_h \le 4 - 4u$ and $|s_\ell| \le 2u$. We have

$$t_h + t_\ell = x_\ell + y_\ell,$$

with $|x_\ell + y_\ell| \le 2u$, hence $|t_h| \le 2u$, and $|t_\ell| \le u^2$. Now, $|s_\ell + t_h| \le 4u$, so $|c| \le 4u$, and $|\epsilon_1| \le 2u^2$. Since $s_h \ge 2 - 4u$ and $|c| \le 4u$, if $p \ge 3$, then Algorithm Fast2Sum introduces no error at line 4 of the algorithm. Therefore,

$$v_h + v_\ell = s_h + c \le 4 - 4u + 4u = 4,$$

so $v_h \le 4$ and $|v_\ell| \le 2u$. Thus, $|t_\ell + v_\ell| \le 2u + u^2$. Hence, either $|t_\ell + v_\ell| < 2u$ and $|\epsilon_2| \le \frac{1}{2}\mathrm{ulp}(t_\ell + v_\ell) \le u^2$, or $2u \le t_\ell + v_\ell \le 2u + u^2$, in which case $w = \mathrm{RN}(t_\ell + v_\ell) = 2u$ and $|\epsilon_2| \le u^2$. In all cases, $|\epsilon_2| \le u^2$. Also, $s_h \ge 2 - 4u$ and $|c| \le 4u$ imply $v_h \ge 2 - 8u$, and $|t_\ell + v_\ell| \le 2u + u^2$ implies $|w| \le 2u$. Hence if $p \ge 3$, then Algorithm Fast2Sum introduces no error at line 6 of the algorithm.

All this gives

$$z_h + z_\ell = v_h + w = x + y + \eta,$$

with $|\eta| = |\epsilon_1 + \epsilon_2| \le 3u^2$.

Since $x + y \ge (x_h - u) + (y_h - u) > 2 - 6u$, the relative error $|\eta|/(x+y)$ is upper bounded by

$$\frac{3u^2}{2-6u},$$

The largest bound obtained in the various cases we have analyzed is

$$\frac{3u^2}{1-4u}.$$

Elementary calculus shows that for $u \in [0, 1/64]$ (i.e., $p \ge 6$) this is always less than $3u^2 + 13u^3$. □

The bound (3) is probably not optimal. The largest relative error we have obtain through many tests is around $2.25 \times 2^{-2p} = 2.25u^2$. An example is the input values given in Equation (2), for which, with $p = 53$ (binary64 arithmetic), we obtain a relative error equal to $2.24999999999999956 \cdots \times 2^{-106}$.

ALGORITHM 6: – **AccurateDWPlusDW**$(x_h, x_\ell, y_h, y_\ell)$. Calculation of $(x_h, x_\ell) + (y_h, y_\ell)$ in binary, precision-p, floating-point arithmetic.

1: $(s_h, s_\ell) \leftarrow \text{2Sum}(x_h, y_h)$
2: $(t_h, t_\ell) \leftarrow \text{2Sum}(x_\ell, y_\ell)$
3: $c \leftarrow \text{RN}(s_\ell + t_h)$
4: $(v_h, v_\ell) \leftarrow \text{Fast2Sum}(s_h, c)$
5: $w \leftarrow \text{RN}(t_\ell + v_\ell)$
6: $(z_h, z_\ell) \leftarrow \text{Fast2Sum}(v_h, w)$
7: **return** (z_h, z_ℓ)

Li et al. (2000, 2002) claim that in binary64 arithmetic ($p = 53$) the relative error of Algorithm 6 is upper bounded by $2 \cdot 2^{-106}$. This bound is inco

$x_h = 900719$
$x_\ell = -9007$
$y_h = -9007$
$y_\ell = -9007$

then the relative error of Algorithm 6 is

2.249999999

Note that this example is somehow "gene
$2^p - 1, x_\ell = -(2^p - 1) \cdot 2^{-p-1}, y_h = -(2^p - 5)/2,$
that is asymptotically equivalent (as p goes to infini
Now let us try to find a relative error bound. We are g

in which one of the opera
e returned result is 2Sum(x_ℓ, y_ℓ,
Now, without loss of generality, we
nonzero. Notice that $1 \le x_h <$

Theorem 3.1. *If $p \ge 3$, then the relative error of Algorithm 6 by*

$$\frac{3u^2}{1-4u} = 3u^2 + 12u^3 + 48u^4 + \cdots,$$

which is less than $3u^2 + 13u^3$ as soon as $p \ge 6$.

Note that the conditions on p ($p \ge 3$ for the bound (3) to hold, $p \ge 6$ for the simplified bound $3u^2 + 13u^3$) are satisfied in all practical cases.

Proof. First, we exclude the straightforward case in which one of the operands is zero. We can also quickly proceed with the case $x_h + y_h = 0$: The returned result is $\text{2Sum}(x_\ell, y_\ell)$, which is equal to $x + y$, that is, the computation is errorless. Now, without loss of generality, we assume $1 \le x_h < 2$, $x \ge |y|$ (which implies $x_h \ge |y_h|$), and $x_h + y_h$ nonzero. Notice that $1 \le x_h < 2$ implies $1 \le x_h \le 2 - 2u$, since x_h is a FP number.

Define ϵ_1 as the error committed at Line 3 of the algorithm:

$$\epsilon_1 = c - (s_\ell + t_h) \tag{4}$$

and ϵ_2 as the error committed at Line 5:

$$\epsilon_2 = w - (t_\ell + v_\ell). \tag{5}$$

1. If $-x_h < y_h \le -x_h/2$. Sterbenz Lemma, applied to the first line of the algorithm, implies $s_h = x_h + y_h$, $s_\ell = 0$, and $c = \text{RN}(t_h) = t_h$.

Define

$$\sigma = \begin{cases} 2 \text{ if } y_h \le -1, \\ 1 \text{ if } -1 < y_h \le -x_h/2. \end{cases}$$

We have $-x_h < y_h \le (1-\sigma) + \frac{x_h}{2}(\sigma - 2)$, so $0 \le x_h + y_h \le 1 + \sigma \cdot (\frac{x_h}{2} - 1) \le 1 - \sigma u$. Also, since x_h is a multiple of $2u$ and y_h is a multiple of σu, $s_h = x_h + y_h$ is a multiple of σu. Since s_h is nonzero, we finally obtain

$$\sigma u \le s_h \le 1 - \sigma u. \tag{6}$$

We have $|x_\ell| \le u$ and $|y_\ell| \le \frac{\sigma}{2} u$, so

$$|t_h| \le \left(1 + \frac{\sigma}{2}\right) u \quad \text{and} \quad |t_\ell| \le u^2. \tag{7}$$

From Equation (6), we deduce that the floating-point exponent of s_h is at least $-p + \sigma - 1$. From Equation (7), the floating-point exponent of $c = t_h$ is at most $-p + \sigma - 1$. Therefore, the Fast2Sum algorithm introduces no error at line 4 of the algorithm, which implies

$$v_h + v_\ell = s_h + c = s_h + t_h = x + y - t_\ell.$$

Equations (6) and (7) imply

$$|s_h + t_h| \le 1 + \left(1 - \frac{\sigma}{2}\right) u \le 1 + \frac{u}{2},$$

so $|v_h| \le 1$ and $|v_\ell| \le \frac{u}{2}$. From the bounds on $|t_\ell|$ and $|v_\ell|$, we obtain:

$$|\epsilon_2| \le \frac{1}{2}\text{ulp}(t_\ell + v_\ell) \le \frac{1}{2}\text{ulp}\left(u^2 + \frac{u}{2}\right) = \frac{u^2}{2} \tag{8}$$

and

$$|\epsilon_2| \le \frac{1}{2}\text{ulp}\left[\frac{1}{2}\text{ulp}(x_\ell + y_\ell) + \frac{1}{2}\text{ulp}\left((x+y) + \frac{1}{2}\text{ulp}(x_\ell + y_\ell)\right)\right]. \tag{9}$$

Lemma 2.1 and $|s_h| \ge \sigma u$ imply that either $s_h + t_h = 0$, or $|v_h| = |\text{RN}(s_h + c)| = |\text{RN}(s_h + t_h)| \ge \sigma u^2$. If $s_h + t_h = 0$, then $v_h = v_\ell = 0$ and the sequel of the proof is straightforward. Therefore, in the following, we assume $|v_h| \ge \sigma u^2$.

Now,

- If $|v_h| = \sigma u^2$, then $|v_\ell + t_\ell| \le u|v_h| + u^2 = \sigma u^3 + u^2$, which implies $|w| = |\text{RN}(t_\ell + v_\ell)| \le \sigma u^2 = |v_h|$;
- If $|v_h| > \sigma u^2$, then, since v_h is a FP number, $|v_h|$ is larger than or equal to the FP number immediately above σu^2, which is $\sigma(1 + 2u)u^2$. Hence $|v_h| \ge \sigma u^2/(1-u)$, so $|v_h| \ge u \cdot |v_h| + \sigma u^2 \ge |v_\ell| + |t_\ell|$. So, $|w| = |\text{RN}(t_\ell + v_\ell)| \le |v_h|$.

Therefore, in all cases, Fast2Sum introduces no error at line 6 of the algorithm, and we have

$$z_h + z_\ell = v_h + w = x + y + \epsilon_2. \tag{10}$$

Directly using Equation (10) and the bound $u^2/2$ on $|\epsilon_2|$ to get a relative error bound would result in a large bound, because $x + y$ may be small. However, when $x + y$ is very small, some simplification occurs thanks to Sterbenz Lemma. First, $x_h + y_h$ is a nonzero multiple of σu. Hence, since $|x_\ell + y_\ell| \le (1 + \frac{\sigma}{2})u$, we have $|x_\ell + y_\ell| \le \frac{3}{2}(x_h + y_h)$. Let us now consider the two possible cases:

- If $-\frac{3}{2}(x_h + y_h) \le x_\ell + y_\ell \le -\frac{1}{2}(x_h + y_h)$, which implies $-\frac{3}{2}s_h \le t_h \le -\frac{1}{2}s_h$, then Sterbenz lemma applies to the floating-point addition of s_h and $c = t_h$. Therefore line 4 of the algorithm results in $v_h = s_h$ and $v_\ell = 0$. An immediate consequence is $\epsilon_2 = 0$, so $z_h + z_\ell = v_h + w = x + y$: the computation of $x + y$ is errorless;

ALGORITHM 6: – **AccurateDWPlusDW**$(x_h, x_\ell, y_h, y_\ell)$. Calculation of $(x_h, x_\ell) + (y_h, y_\ell)$ in binary, precision-p, floating-point arithmetic.

1: $(s_h, s_\ell) \leftarrow \mathrm{2Sum}(x_h, y_h)$
2: $(t_h, t_\ell) \leftarrow \mathrm{2Sum}(x_\ell, y_\ell)$
3: $c \leftarrow \mathrm{RN}(s_\ell + t_h)$
4: $(v_h, v_\ell) \leftarrow \mathrm{Fast2Sum}(s_h, c)$
5: $w \leftarrow \mathrm{RN}(t_\ell + v_\ell)$
6: $(z_h, z_\ell) \leftarrow \mathrm{Fast2Sum}(v_h, w)$
7: **return** (z_h, z_ℓ)

...ther $s_h + t_h = 0$, or $|v_h| = |\mathrm{RN}(s_h + c)| = |\mathrm{RN}$

d the sequel of the proof is straightforward. Th

Li et al. (2000, 2002) claim that ... is upper bounded by $2 \cdot 2^{-106}$. Th...

y_h
$y_\ell = -$

then the relative error of Algorithm 6 is

$$2.24999999999999956 \cdots .$$

Note that this example is somehow "generic": In precision-p FP arith... $2^p - 1$, $x_\ell = -(2^p - 1) \cdot 2^{-p-1}$, $y_h = -(2^p - 5)/2$, and $y_\ell = -(2^p - 1) \cdot 2^{-p-3}$ leads to a relative error that is asymptotically equivalent (as p goes to infinity) to $2.25u^2$.

Now let us try to find a relative error bound. We are going to show the following result.

Theorem 3.1. *If $p \geq 3$, then the relative error of Algorithm 6 (AccurateDWPlusDW) is bounded by*

$$\frac{3u^2}{1-4u} = 3u^2 + 12u^3 + 48u^4 + \cdots, \tag{3}$$

which is less than $3u^2 + 13u^3$ as soon as $p \geq 6$.

Note that the conditions on p ($p \geq 3$ for the bound (3) to hold, $p \geq 6$ for the simplified bound $3u^2 + 13u^3$) are satisfied in all practical cases.

Proof. First ... we have $|x_\ell + y_\ell| \le$ ...

- If $-\frac{3}{2}(x_h + y_h) \leq x_\ell + y_\ell \leq -\frac{1}{2}(x_h + y_h)$

... applies to the floating-p...

and ϵ_2 as the err...

$$\epsilon_2 = w - (t_\ell + v_\ell). \tag{5}$$

1. If $-x_h < y_h \leq -x_h/2$. Sterbenz Lemma, applied to the first line of the algorithm, implies $s_h = x_h + y_h$, $s_\ell = 0$, and $c = \mathrm{RN}(t_h) = t_h$.

Define

$$\sigma = \begin{cases} 2 \text{ if } y_h \leq -1, \\ 1 \text{ if } -1 < y_h \leq -x_h/2. \end{cases}$$

We have $-x_h < y_h \leq (1-\sigma) + \frac{x_h}{2}(\sigma - 2)$, so $0 \leq x_h + y_h \leq 1 + \sigma \cdot (\frac{x_h}{2} - 1) \leq 1 - \sigma u$. Also, since x_h is a multiple of $2u$ and y_h is a multiple of σu, $s_h = x_h + y_h$ is a multiple of σu. Since s_h is nonzero, we finally obtain

$$\sigma u \leq s_h \leq 1 - \sigma u. \tag{6}$$

... have $|x_\ell| \leq u$ and $|y_\ell| \leq \frac{\sigma}{2}u$, so

$$\ldots \leq \left(1 + \frac{\sigma}{2}\right)u \quad \text{and} \quad |t_\ell| \leq u^2. \tag{7}$$

...g-point exponent of s_h is at least $-p + \sigma - 1$. From ... is at most $-p + \sigma - 1$. Therefore, the Fast2Sum ...hm, which implies

$$\ldots = x + y - t_\ell.$$

$$\ldots \left(1 - \frac{\sigma}{2}\right)u \leq 1 + \frac{u}{2},$$

...bounds on $|t_\ell|$ and $|v_\ell|$, we obtain:

$$|\epsilon_2| \leq \frac{1}{2}\mathrm{ulp}(t_\ell + v_\ell) \leq \frac{1}{2}\mathrm{ulp}\left(u^2 + \frac{u}{2}\right) = \frac{u^2}{2} \tag{8}$$

and

$$|\epsilon_2| \leq \frac{1}{2}\mathrm{ulp}\left[\frac{1}{2}\mathrm{ulp}(x_\ell + y_\ell) + \frac{1}{2}\mathrm{ulp}\left((x+y) + \frac{1}{2}\mathrm{ulp}(x_\ell + y_\ell)\right)\right]. \tag{9}$$

Lemma 2.1 and $|s_h| \geq \sigma u$ imply that either $s_h + t_h = 0$, or $|v_h| = |\mathrm{RN}(s_h + c)| = |\mathrm{RN}(s_h + t_h)| \geq \sigma u^2$. If $s_h + t_h = 0$, then $v_h = v_\ell = 0$ and the sequel of the proof is straightforward. Therefore, in the following, we assume $|v_h| \geq \sigma u^2$.

Now,

- If $|v_h| = \sigma u^2$, then $|v_\ell + t_\ell| \leq u|v_h| + u^2 = \sigma u^3 + u^2$, which implies $|w| = |\mathrm{RN}(t_\ell + v_\ell)| \leq \sigma u^2 = |v_h|$;
- If $|v_h| > \sigma u^2$, then, since v_h is a FP number, $|v_h|$ is larger than or equal to the FP number immediately above σu^2, which is $\sigma(1 + 2u)u^2$. Hence $|v_h| \geq \sigma u^2/(1-u)$, so $|v_h| \geq u \cdot |v_h| + \ldots^2 \geq |v_\ell| + |t_\ell|$. So, $|w| = |\mathrm{RN}(t_\ell + v_\ell)| \leq |v_h|$.

...ll cases, Fast2Sum introduces no error at line 6 of the algorithm, and we have

$$z_h + z_\ell = v_h + w = x + y + \epsilon_2. \tag{10}$$

...ly using Equation (10) and the bound $u^2/2$ on $|\epsilon_2|$ to get a relative error bound would result in a large bound, because $x + y$ may be small. However, when $x + y$ is very small, some simplification occurs thanks to Sterbenz Lemma. First, $x_h + y_h$ is a nonzero multiple of σu. Hence, since $|x_\ell + y_\ell| \leq (1 + \frac{\sigma}{2})u$, we have $|x_\ell + y_\ell| \leq \frac{3}{2}(x_h + y_h)$. Let us now consider the two possible cases:

- If $-\frac{3}{2}(x_h + y_h) \leq x_\ell + y_\ell \leq -\frac{1}{2}(x_h + y_h)$, which implies $-\frac{3}{2}s_h \leq t_h \leq -\frac{1}{2}s_h$, then Sterbenz lemma applies to the floating-point addition of s_h and $c = t_h$. Therefore line 4 of the algorithm results in $v_h = s_h$ and $v_\ell = 0$. An immediate consequence is $\epsilon_2 = 0$, so $z_h + z_\ell = v_h + w = x + y$: the computation of $x + y$ is errorless;

- If $-\frac{1}{2}(x_h + y_h) < x_\ell + y_\ell \le \frac{3}{2}(x_h + y_h)$, then $\frac{5}{2}(x_\ell + y_\ell) \le \frac{3}{2}(x_h + y_h + x_\ell + y_\ell) = \frac{3}{2}(x + y)$, and $-\frac{1}{2}(x+y) < \frac{1}{2}(x_\ell + y_\ell)$. Hence, $|x_\ell + y_\ell| < |x + y|$, so $\text{ulp}(x_\ell + y_\ell) \le \text{ulp}(x + y)$. Combined with Equation (9), this gives

$$|\epsilon_2| \le \frac{1}{2}\text{ulp}\left(\frac{3}{2}\text{ulp}(x+y)\right) \le 2^{-p}\text{ulp}(x+y) \le 2 \cdot 2^{-2p} \cdot (x+y).$$

2. If $-x_h/2 < y_h \le x_h$

Notice that we have $x_h/2 < x_h + y_h \le 2x_h$, so $x_h/2 \le s_h \le 2x_h$. Also notice that we have $|x_\ell| \le u$.

- If $\frac{1}{2} < x_h + y_h \le 2 - 4u$. Define

Elementary calculus shows that fo

We have

The bound (3) is probabl

When $\sigma = 1$, we h… $x_h \le 2 - 2u$ implies $|y_\ell$… $(1 + \sigma/2)u$, therefore

$$|t_h| \le \left(1 + \frac{\sigma}{2}\right)u \ldots$$

Now, $|s_\ell + t_h| \le (1 + \sigma)u$, so

$$|c| \le (1+\sigma)u \quad \text{and} \quad |\epsilon_1| \le \sigma u^2. \tag{13}$$

Since $s_h \ge 1/2$ and $|c| \le 3u$, if $p \ge 3$, then Algorithm Fast2Sum introduces no error at line 4 of the algorithm, that is,

$$v_h + v_\ell = s_h + c.$$

Therefore $|v_h + v_\ell| = |s_h + c| \le \sigma(1 - 2u) + (1 + \sigma)u \le \sigma$. This implies

$$|v_h| \le \sigma \quad \text{and} \quad |v_\ell| \le \frac{\sigma}{2}u. \tag{14}$$

Thus $|t_\ell + v_\ell| \le u^2 + \frac{\sigma}{2}u$, so

$$|w| \le \frac{\sigma}{2}u + u^2 \quad \text{and} \quad |\epsilon_2| \le \frac{\sigma}{2}u^2. \tag{15}$$

From Equations (11) and (13), we deduce $s_h + c \ge \frac{\sigma}{2} - u(2\sigma + 1)$, so $|v_h| \ge \frac{\sigma}{2} - u(2\sigma+1)$. If $p \ge 3$, then $|v_h| \ge |w|$, so Algorithm Fast2Sum introduces no error at line 6 of the algorithm, that is, $z_h + z_\ell = v_h + w$.

Therefore,

$$z_h + z_\ell = x + y + \eta,$$

with $|\eta| = |\epsilon_1 + \epsilon_2| \le \frac{3\sigma}{2}u^2$. Since

$$x + y \ge (x_h - u) + (y_h - u/2) > \begin{cases} \frac{1}{2} - \frac{3}{2}u & \text{if} \quad \sigma = 1, \\ 1 - 4u & \text{if} \quad \sigma = 2, \end{cases}$$

the relative error $|\eta|/(x+y)$ is upper bounded by

$$\frac{3u^2}{1-4u}.$$

- If $2 - 4u < x_h + y_h \le 2x_h$, then $2 - 4u \le s_h \le \text{RN}(2x_h) = 2x_h \le 4 - 4u$ and $|s_\ell| \le 2u$. We have

$$t_h + t_\ell = x_\ell + y_\ell,$$

with $|x_\ell + y_\ell| \le 2u$, hence $|t_h| \le 2u$, and $|t_\ell| \le u^2$. Now, $|s_\ell + t_h| \le 4u$, so $|c| \le 4u$, and $|\epsilon_1| \le 2u^2$. Since $s_h \ge 2 - 4u$ and $|c| \le 4u$, if $p \ge 3$, then Algorithm Fast2Sum introduces no error at line 4 of the algorithm. Therefore,

$$v_h + v_\ell = s_h + c \le 4 - 4u + 4u = 4,$$

so $v_h \le 4$ and $|v_\ell| \le 2u$. Thus, $|t_\ell + v_\ell| \le 2u + u^2$. Hence, either $|t_\ell + v_\ell| < 2u$ and $|\epsilon_2| \le \frac{1}{2}\text{ulp}(t_\ell + v_\ell) \le u^2$, or $2u \le t_\ell + v_\ell \le 2u + u^2$, in which case $w = \text{RN}(t_\ell + v_\ell) = 2u$ and … $\le u^2$. In all cases, $|\epsilon_2| \le u^2$. Also, $s_h \ge 2 - 4u$ and $|c| \le 4u$ imply $v_h \ge 2 - 8u$, and …$| \le 2u + u^2$ implies $|w| \le 2u$. Hence if $p \ge 3$, then Algorithm Fast2Sum introduces … at line 6 of the algorithm.

…is gives

$$z_h + z_\ell = v_h + w = x + y + \eta,$$

…ith $|\eta| = |\epsilon_1 + \epsilon_2| \le 3u^2$.

Since $x + y \ge (x_h - u) + (y_h - u) > 2 - 6u$, the relative error $|\eta|/(x+y)$ is upper bounded by

$$\frac{3u^2}{2 - 6u},$$

The largest bound obtained in the various cases we have analyzed is

$$\frac{3u^2}{1-4u}.$$

Elementary calculus shows that for $u \in [0, 1/64]$ (i.e., $p \ge 6$) this is always less than $3u^2 + 13u^3$. □

The bound (3) is probably not optimal. The largest relative error we have obtain through many tests is around $2.25 \times 2^{-2p} = 2.25u^2$. An example is the input values given in Equation (2), for which, with $p = 53$ (binary64 arithmetic), we obtain a relative error equal to $2.24999999999999956 \cdots \times 2^{-106}$.

DW+DW: "accurate version"

So the theorem gives an error bound

$$\frac{3u^2}{1-4u} \simeq 3u^2 \ldots$$

As said before, that theorem has an interesting history:

- the authors of the first paper where a bound was given (in 2000) claimed (without published proof) that the relative error was always $\leq 2u^2$ (in binary64 arithmetic);
- when trying (without success) to prove their bound, we found an example with error $\approx 2.25u^2$;
- we finally proved the theorem, and Laurence Rideau started to write a formal proof in Coq;
- of course, that led to finding a (minor) flaw in our proof...

(I hate Coq people)

DW+DW: "accurate version"

- fortunately the flaw was quickly corrected (before final publication of the paper...Phew)!
- still, the gap between worst case found ($\approx 2.25u^2$) and the bound ($\approx 3u^2$) was frustrating, so I spent months trying to improve the theorem...
- and of course this could not be done: it was the worst case that needed spending time!
- we finally found that with

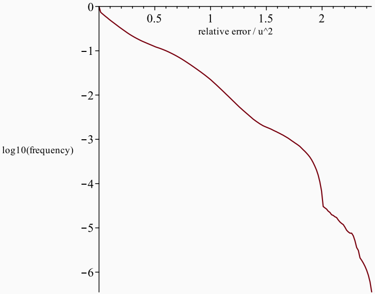
$$\begin{aligned} x_h &= 1 \\ x_\ell &= u - u^2 \\ y_h &= -\frac{1}{2} + \frac{u}{2} \\ y_\ell &= -\frac{u^2}{2} + u^3. \end{aligned}$$

Exercise: all these values are FP numbers.

error $\frac{3u^2-2u^3}{1+3u-3u^2+2u^3}$ is attained. With $p = 53$ (binary64 arithmetic), gives error $2.99999999999999877875\cdots \times u^2$.

DW+DW: “accurate version”

- We suspect the initial authors hinted their claimed bound just by performing zillions of random tests
- in this domain, the worst cases are extremely unlikely: you must build them. Almost impossible to find them by chance.



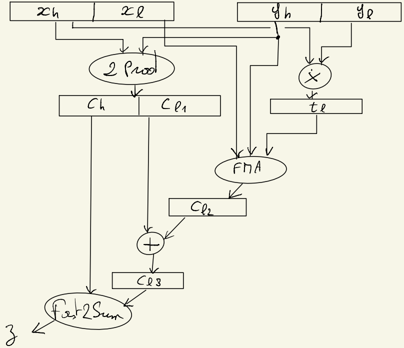
$\log_{10}$ of the frequency of cases for which the relative error of DWPlusDW is $\geq \lambda u^2$ as a function of λ.

DW × DW

- Product $z = (z_h, z_\ell)$ of two DW numbers $x = (x_h, x_\ell)$ and $y = (y_h, y_\ell)$;
- several algorithms $\rightarrow$ tradeoff speed/accuracy. We just give one of them.

DWTimesDW

$(c_h, c_{\ell 1}) \leftarrow \text{2Prod}(x_h, y_h)$
$t_\ell \leftarrow \text{RN}\,(x_h \cdot y_\ell)$
$c_{\ell 2} \leftarrow \text{RN}\,(t_\ell + x_\ell y_h)$
$c_{\ell 3} \leftarrow \text{RN}\,(c_{\ell 1} + c_{\ell 2})$
$(z_h, z_\ell) \leftarrow \text{Fast2Sum}(c_h, c_{\ell 3})$
return (z_h, z_ℓ)



DW $\times$ DW

We have

Theorem (M. and Rideau, 2022)

If $p \geq 5$*, the relative error of Algorithm DWTimesDW is less than or equal to*

$$\frac{5u^2}{(1+u)^2} < 5u^2.$$

and that theorem too has an interesting (hmmm...a bit more annoying?) history!

- in 2017, I participated to the proof of an initial relative error bound $6u^2$;
- again, Laurence tried translating the proof in Coq...and it turned out the proof was based on a wrong lemma (and this was *after* publication).

(what did I say about Coq people?)

DW $\times$ DW

- after a few nights of bad sleep, turn-around. . . that also improved the bound: $6u^2 \to 5u^2$!
- no proof of asymptotic optimality, but in binary64 arithmetic, we have examples with error $> 4.98u^2$;
- *real consolation or lame excuse?* Maybe without the flaw, we would never have found the better bound.

Halfway conclusion

Full set of validated DW algorithms for the arithmetic operations and the square root (M. and Rideau, 2022; Lefèvre, Louvet, Picot, M. and Rideau, 2023).

That class of algorithms really needs formal proof:

- Proofs have too many subcases to be certain you have not forgotten one;
- they are boring: almost nobody reads them.

Alternate–or complementary–solution? try to automatically compute bounds:

- short-term goal: limit human intervention (and therefore, human error);
- long-term goal: bounds correct by construction.

An example: hypotenuse function $\sqrt{x^2 + y^2}$

- function `hypot` listed in Section 9 of the IEEE-754 Std for FP arithmetic and Section 7.12.7.3 of the C17 Std. The C Std even says

  The `hypot` functions compute the square root of the sum of the squares of x and y, without undue overflow or underflow. A range error may occur.

- naive algorithm: reasonably accurate (rel. err. $< 2u$), but risks of
  - spurious overflow: we obtain ∞, even if exact result $\ll \Omega$, or
  - spurious underflow: very inaccurate result if subnormal intermediate values.

The naive algorithm

NaiveHypot

1: $s_x \leftarrow \mathrm{RN}\left(x^2\right)$
2: $s_y \leftarrow \mathrm{RN}\left(y^2\right)$
3: $\sigma \leftarrow \mathrm{RN}\left(s_x + s_y\right)$
4: $\rho_1 = \mathrm{RN}\left(\sqrt{\sigma}\right)$

- classical relative error bound $2u + \mathcal{O}(u^2)$;
- refinement: $2u$ (Jeannerod & Rump);
- asymptotically optimal (Jeannerod, M., Plet).

Examples in binary64/double precision arithmetic ($p = 53$):

- if $x = 2^{600}$ and $y = 0$, returned result $+\infty$, exact result 2^{600};
- if $x = 65 \times 2^{-542}$ and $y = 72 \times 2^{-542}$, returned result 96×2^{-542}, exact result 97×2^{-542}.

$\Rightarrow$ need to scale the operands.

Simple scaling

1: **if** $|x| < |y|$ **then**
2: swap (x, y)
3: **end if**
4: $r \leftarrow \mathrm{RN}(y/x)$
5: $t \leftarrow \mathrm{RN}(1 + r^2)$
6: $s \leftarrow \mathrm{RN}(\sqrt{t})$
7: $\rho_2 = \mathrm{RN}(|x| \cdot s)$

- several versions;
- this one requires availability of an FMA (*fused multiply-add:* $\mathrm{RN}(ab + c)$);
- relative error bounded by $\frac{5}{2}u + \frac{3}{8}u^2$;
- asymptotically optimal.

$\Rightarrow$ avoiding spurious overflow has a significant cost in terms of accuracy.

Improvements?

Simple scaling with compensation (Nelson Beebe, 2017)

if $|x| < |y|$ **then**
 $\text{swap}(x, y)$
end if
$r \leftarrow \text{RN}(y/x)$
$t \leftarrow \text{RN}(1 + r^2)$
$s \leftarrow \text{RN}(\sqrt{t})$
$\epsilon \leftarrow \text{RN}(t - s^2)$
$c \leftarrow \text{RN}(\epsilon/(2s))$
$\nu \leftarrow \text{RN}(|x| \cdot c)$
$\rho_3 \leftarrow \text{RN}(|x| \cdot s + \nu)$

- this version: requires an FMA;
- one Newton-Raphson iteration;
- relative error bound $\frac{8}{5}u + \frac{7}{5}u^2$ (Salvy & M., 2023);
- sharp: known case with error $1.5999739u$ in binary64 FP arithmetic.

Borges' "fused" algorithm (2020)

if $|x| < |y|$ **then**
    swap(x, y)
end if
$(s_x^h, s_x^\ell) \leftarrow \mathrm{Fast2Mult}(x, x)$
$(s_y^h, s_y^\ell) \leftarrow \mathrm{Fast2Mult}(y, y)$
$(\sigma_h, \sigma_\ell) \leftarrow \mathrm{Fast2Sum}(s_x^h, s_y^h)$
$s \leftarrow \mathrm{RN}\left(\sqrt{\sigma_h}\right)$
$\delta_s \leftarrow \mathrm{RN}\left(\sigma_h - s^2\right)$
$\tau_1 \leftarrow \mathrm{RN}\left(s_x^\ell + s_y^\ell\right)$
$\tau_2 \leftarrow \mathrm{RN}\left(\delta_s + \sigma_\ell\right)$
$\tau \leftarrow \mathrm{RN}\left(\tau_1 + \tau_2\right)$
$c \leftarrow \mathrm{RN}\left(\tau / s\right)$
$\rho_4 \leftarrow \mathrm{RN}\left(0.5c + s\right)$

Requires an FMA. DW and NR. Relative error bound $u + 14u^2$ (Salvy & M. 2023). Asymptotically optimal.

Kahan's algorithm (1987)

$\delta \leftarrow \mathrm{RN}\,(x - y)$
if $\delta > y$ **then**
    $r \leftarrow \mathrm{RN}\,(x/y)$
    $t \leftarrow \mathrm{RN}\,(1 + r^2)$
    $s \leftarrow \mathrm{RN}\,(\sqrt{t})$
    $z \leftarrow \mathrm{RN}\,(r + s)$
else
    $r_2 \leftarrow \mathrm{RN}\,(\delta/y)$
    $tr_2 \leftarrow \mathrm{RN}\,(2r_2)$
    $r_3 \leftarrow \mathrm{RN}\,(tr2 + r_2^2)$
    $r_4 \leftarrow \mathrm{RN}\,(2 + r_3)$
    $s_2 \leftarrow \mathrm{RN}\,(\sqrt{r_4})$
    $d = \mathrm{RN}\,(R_2 + s_2)$
    $q = \mathrm{RN}\,(r_3/d)$
    $r_5 \leftarrow \mathrm{RN}\,(P_\ell + q)$
    $r_6 \leftarrow \mathrm{RN}\,(r_5 + r_2)$
    $z \leftarrow \mathrm{RN}\,(P_h + r_6)$
end if
$z_2 \leftarrow \mathrm{RN}\,(y/z)$
$\rho_7 \leftarrow \mathrm{RN}\,(x + z_2)$

In this presentation, requires an FMA . We assume $0 \leq y \leq x$. Precomputed constants $R_2 = \mathrm{RN}\,(\sqrt{2})$, $P_h = \mathrm{RN}\,(1 + \sqrt{2})$, and $P_\ell = \mathrm{RN}\,(1 + \sqrt{2} - P_h)$. Not-fully-trusted paper and pencil proof of a bound $1.5765u + \mathcal{O}(u^2)$, known cases with error $1.4977u$ in binary32 arithmetic.

The various bounds obtained

Algorithm	reference	error bound	condition	status
Naive	folklore	$2u - \frac{8}{5}(9 - 4\sqrt{6})u^2$	$p \geq 2$	asympt. optimal
Simple scaling	folklore	$\frac{5}{2}u + \frac{3}{8}u^2$	$p \geq 2$	asympt. optimal
Scaling w. compensation	N. Beebe (2017)	$\frac{8}{5}u + \frac{7}{5}u^2$	$p \geq 4$	sharp
Borges "fused"	C. Borges (2020)	$u + 14u^2$	$p \geq 5$	asympt. optimal
Kahan	W. Kahan (1987)	$1.5765u + \mathcal{O}(u^2)$?	$p \geq 9$	a bit loose

Goal: tight and certain relative error bounds

- Programs that at step k have an instruction of the form

  `x_k = x_i op x_j`   or   `x_k = sqrt(x_i)`

  where op is +, -, * or /, and `x_i` and `x_j` are either precomputed values or input values $(i, j < k)$;

- Computed values:

$$x_k = \mathsf{RN}\,(x_i \;\; \mathtt{op} \;\; x_j) \quad \text{or} \quad x_k = \mathsf{RN}\,(\sqrt{x_i});$$

- basic relations:

$$\begin{aligned} x_k &= x_i \;\; \mathtt{op} \;\; x_j \pm \tfrac{1}{2}\,\mathsf{ulp}\,(x_i \;\; \mathtt{op} \;\; x_j), \\ x_k &= (x_i \;\; \mathtt{op} \;\; x_j)(1+\epsilon), \quad \text{with } |\epsilon| \le \tfrac{u}{1+u} < u. \end{aligned} \tag{3}$$

  (or the same with $\sqrt{x_i}$)

Optimisation problem: find the maximum and the minimum of the quantity $\rho/\sqrt{x^2+y^2} - 1$ in the region defined by the equalities and inequalities obtained from analyzing the program (e.g., (3)) $\rightarrow$ Algebraic bound.

Goal: tight and certain relative error bounds

- algorithmically the polynomial optimization problem is well-understood (Nie & Ranestad 2009, Bank, Giusti, Heintz, Safey El Din 2014);
- however, it is very expensive.

$\rightarrow$ the natural turn around is to compute approximations of the algebraic bound, or to restrict ourselves to order-1 analyses in u;

- here: testing the limits of what can be computed exactly from the bounds of the individual operations;
- The "general" methods do not exploit the sparsity and the structure of our systems;

$\rightarrow$ use of heuristics;

Prototype implementation: illustration with the naive alg.

NaiveHypot

1: $s_x \leftarrow \mathrm{RN}\left(x^2\right)$

2: $s_y \leftarrow \mathrm{RN}\left(y^2\right)$

3: $\sigma \leftarrow \mathrm{RN}\left(s_x + s_y\right)$

4: $\rho_1 = \mathrm{RN}\left(\sqrt{\sigma}\right)$

```
> with(BoundRoundingError); # loads the package
> Algo1:=[Input(x=0..2^16,y=0..2^16,_u=0..1/4),
> s[x]=RN(x^2),s[y]=RN(y^2),sigma=RN(s[x]+s[y]),
         rho=RN(sqrt(sigma))]:
> sys:=AnalyzeAlgo(Algo1):
> linpart:=BoundLinearTerm(sys);
```

$$linpart := 2\_u, \{\_eps_{\rho} = 1, \_eps_{\sigma} = 1, \_eps_{s_x} = 1, \_eps_{s_y} = 1\}$$

```
> quad:=BoundQuadraticTerm(linpart,sys);
```

$$quad := \mathrm{RootOf}(5\_Z^2 - 144\_Z - 192, -1.276734354), \{\_u = 1/4\}$$

```
> allvalues(quad[1]);
```

$$\frac{72}{5} - \frac{32\sqrt{6}}{5}$$

Goal: tight and certain relative error bounds

- Reminder: computed values

$$x_k = \mathsf{RN}\left(x_i \ \mathsf{op}\ x_j\right) \quad \text{or} \quad x_k = \mathsf{RN}\left(\sqrt{x_i}\right);$$

- we compare the computed values x_k with the exact values:

$$x_k^* = x_i^* \ \mathrm{op}\ x_j^* \quad \text{or} \quad x_k^* = \sqrt{x_i^*};$$

(initial values: $x_i = x_i^*$ for $i \leq 0$).

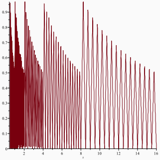
- The analysis consists in iteratively computing relative error bounds $\epsilon_k^\ell(u)$ and $\epsilon_k^r(u)$ such that (here, for positive x_k and x_k^*)

$$x_k^* \left(1 - \epsilon_k^\ell(u)\right) \leq x_k \leq x_k^* \left(1 + \epsilon_k^r(u)\right); \tag{4}$$

Goal: tight and certain relative error bounds

- with care, iteratively computing bounds of the form (4), using at each step the "basic relations" (3) is not so difficult;
- ending up with a tight bound is difficult. Two reasons:
  - requires existence of input values for which the individual rounding errors attain their maximum (with the right sign) at each operation.

    $\rightarrow$ Not always possible: Correlations. $3 \cdot (x \cdot y)$, one cannot have both $(x \cdot y)$ and $3 \cdot (x \cdot y)$ very slightly above a power of 2;



(and, indeed, $3 \cdot (x \cdot y)$ more accurate than $(3 \cdot x) \cdot y$)

  - the "basic relations" (3) are not the last word: there are some additional properties specific to FP arithmetic, and some "bit coincidences".

Examples of additional properties specific to FP arithmetic

Lemma (Sterbenz)

If a and b are floating-point numbers satisfying $a/2 \leq b \leq 2a$ then $b - a$ is a floating-point number, which implies $RN(b - a) = b - a$.

(more generally, some operations are exact: any multiple of 2^k of abs. val. $\leq 2^{k+p}$ is a FPN)

Lemma (Jeannerod-Rump)

When $p \geq 2$, the relative error of a square root is bounded by

$$1 - \frac{1}{\sqrt{1+2u}}; \tag{5}$$

the relative error of a division in binary FP arithmetic is bounded by

$$u - 2u^2. \tag{6}$$

“Bit coincidences”: computation of $x^2 - 2$ as $\mathrm{RN}\,(\,\mathrm{RN}\,(x \cdot x) - 2)$

p	max. relative error		
11	$2048u$	$= 1$	all information lost
12	$670u$	$= 0.16$	not so bad
13	$7001u$	$= 0.85$	
14	$8005u$	$= 0.49$	
15	$11366u$	$= 0.35$	
16	$65536u$	$= 1$	all information lost

Depends on how close $\sqrt{2}$ is to a FP number. In a way, 12-bit arithmetic more accurate than 16-bit arithmetic.

Analysis of Beebe's algorithm

if $|x| < |y|$ **then**
  $\text{swap}(x, y)$
end if
$r \leftarrow \text{RN}\left(y/x\right)$
$t \leftarrow \text{RN}\left(1 + r^2\right)$
$s \leftarrow \text{RN}\left(\sqrt{t}\right)$
$\epsilon \leftarrow \text{RN}\left(t - s^2\right)$
$c \leftarrow \text{RN}\left(\epsilon/(2s)\right)$
$\nu \leftarrow \text{RN}\left(|x| \cdot c\right)$
$\rho_3 \leftarrow \text{RN}\left(|x| \cdot s + \nu\right)$

Analysis of Beebe's algorithm

Simplification: $x \geq y > 0$

$r \leftarrow \mathrm{RN}\left(y/x\right)$
$t \leftarrow \mathrm{RN}\left(1 + r^2\right)$
$s \leftarrow \mathrm{RN}\left(\sqrt{t}\right)$
$\epsilon \leftarrow \mathrm{RN}\left(t - s^2\right)$
$c \leftarrow \mathrm{RN}\left(\epsilon/(2s)\right)$
$\nu \leftarrow \mathrm{RN}\left(x \cdot c\right)$
$\rho_3 \leftarrow \mathrm{RN}\left(x \cdot s + \nu\right)$

Main idea: Newton-Raphson iteration

$$\frac{\epsilon}{2s} + s = \frac{t - s^2}{2s} + s = \sqrt{t} + \frac{(s - \sqrt{t})^2}{2s},$$

so that

$$\left(\frac{\epsilon}{2s} + s\right) - \sqrt{t} = \frac{(s - \sqrt{t})^2}{2s}.$$

Analysis of Beebe's algorithm

```
r ← RN(y/x)
t ← RN(1 + r^2)
s ← RN(√t)
ε ← RN(t − s^2)
c ← RN(ε/(2s))
ν ← RN(x · c)
ρ3 ← RN(x · s + ν)
```

- define α by $y = \alpha x$, so that $r = \mathrm{RN}(\alpha)$;
- $r = \alpha + u\epsilon_r$, with

$$|\epsilon_r| \leq \begin{cases} \frac{1}{4}, & \text{if } \alpha \leq 1/2, \\ \frac{1}{2}, & \text{if } \alpha > 1/2. \end{cases}$$

- $t = 1 + r^2 + u\epsilon_t$, with $|\epsilon_t| \leq 1$ (comes from $1 + r^2 \leq 2$);
- $s = \sqrt{t} + u\epsilon_s$, with $|\epsilon_s| \leq 1$ (comes from $t < 2$);
- $\epsilon = t - s^2$ (comes from Sterbenz Lemma).

Analysis of Beebe's algorithm

$$
\begin{aligned}
r &\leftarrow \mathrm{RN}\,(y/x)\\
t &\leftarrow \mathrm{RN}\,(1+r^2)\\
s &\leftarrow \mathrm{RN}\,(\sqrt{t})\\
\epsilon &\leftarrow \mathrm{RN}\,(t-s^2)\\
c &\leftarrow \mathrm{RN}\,(\epsilon/(2s))\\
\nu &\leftarrow \mathrm{RN}\,(x\cdot c)\\
\rho_3 &\leftarrow \mathrm{RN}\,(x\cdot s+\nu)
\end{aligned}
$$

$$
\begin{aligned}
\left|\frac{\epsilon}{2s}\right| &= \left|\frac{t-s^2}{2s}\right| \\
&= \left|\frac{(s-u\epsilon_s)^2-s^2}{2s}\right| \\
&= \left|-u\epsilon_s+\frac{u^2\epsilon_s^2}{2s}\right| \le u+\frac{u^2}{2}.
\end{aligned}
\tag{7}
$$

- If $|\epsilon/(2s)| \le u$ then the error committed by rounding $\frac{\epsilon}{2s}$ to nearest is $\le u^2/2$;
- If $|\epsilon/(2s)| > u$, then since the FPN above u is $u+2u^2$, (7) implies $\mathrm{RN}\,(\epsilon/(2s)) = \pm u$ $\Rightarrow$ again the rounding error is $\le u^2/2$.

Hence in all cases, $|c| \le u$ and

$$c = \frac{\epsilon}{2s} + \epsilon_c\frac{u^2}{2},$$

with $|\epsilon_c| \le 1$.

Analysis of Beebe's algorithm

1: $r \leftarrow \mathrm{RN}\,(y/x)$
2: $t \leftarrow \mathrm{RN}\,(1 + r^2)$
3: $s \leftarrow \mathrm{RN}\,(\sqrt{t})$
4: $\epsilon \leftarrow \mathrm{RN}\,(t - s^2)$
5: $c \leftarrow \mathrm{RN}\,(\epsilon/(2s))$
6: $\nu \leftarrow \mathrm{RN}\,(x \cdot c)$
7: $\rho_3 \leftarrow \mathrm{RN}\,(x \cdot s + \nu)$

- $\nu = xc(1 + u\epsilon_\nu)$ with $|\epsilon_\nu| \leq 1/(1+u)$;
- $\rho = (\nu + xs)(1 + u\epsilon_\rho)$ with $|\epsilon_\rho| \leq 1/(1+u)$;

Analysis of Beebe's algorithm

Putting all this together:

$$\begin{aligned}
\rho &= (\nu + xs)(1 + u\epsilon_\rho), \\
&= x\left((-u\epsilon_s + \tfrac{u^2}{2}(\epsilon_c + \epsilon_s^2/s))(1 + u\epsilon_\nu) + \sqrt{t} + u\epsilon_s\right)(1 + u\epsilon_\rho), \\
&= x\left(\sqrt{t} + \tfrac{u^2}{2}((\epsilon_c + \epsilon_s^2/s)(1 + u\epsilon_\nu) - 2\epsilon_s\epsilon_\nu)\right)(1 + u\epsilon_\rho) \\
&= x\sqrt{1 + r^2}\sqrt{1 + \tfrac{u\epsilon_t}{1+r^2}}\left(1 + \tfrac{u^2}{2\sqrt{t}}((\epsilon_c + \epsilon_s^2/s)(1 + u\epsilon_\nu) - 2\epsilon_s\epsilon_\nu)\right)(1 + u\epsilon_\rho),
\end{aligned}$$

Lemma

The relative error of the algorithm is

$$\begin{aligned}
R &= \sqrt{1 + \tfrac{r^2-\alpha^2}{1+\alpha^2}}\sqrt{1 + \tfrac{u\epsilon_t}{1+r^2}} \\
&\qquad \times \left(1 + \tfrac{u^2}{2\sqrt{t}}((\epsilon_c + \epsilon_s^2/s)(1 + u\epsilon_\nu) - 2\epsilon_s\epsilon_\nu)\right)(1 + u\epsilon_\rho) - 1, \\
&= \tfrac{r^2-\alpha^2+u\epsilon_t}{2(1+\alpha^2)} + u\epsilon_\rho + O(u^2), \quad u \to 0.
\end{aligned}$$

Moreover, $|\epsilon_s|, |\epsilon_t|, |\epsilon_c|$ *are bounded by 1 and* $|\epsilon_\nu|$ *and* $|\epsilon_\rho|$ *by* $1/(1+u)$.

Now, the painful work

- linear term

$$\left(\frac{2\alpha\epsilon_r + \epsilon_t}{2(1+\alpha^2)} + \epsilon_\rho\right) \cdot u$$

  - increasing function of ϵ_r, ϵ_t and ϵ_ρ,
  - $\epsilon_r \leq 1/4$ if $\alpha \leq 1/2$, $\epsilon_r \leq 1/2$ otherwise,
  - $\epsilon_t, \epsilon_\rho \leq 1$

  $\rightarrow$ max. value 8/5;

- show that for $u \in [0, 1/2]$,

$$\frac{\partial R}{\partial \epsilon_\rho} \geq 0, \quad \frac{\partial R}{\partial \epsilon_t} \geq 0, \quad \frac{\partial R}{\partial \epsilon_r} \geq 0, \quad \frac{\partial R}{\partial \epsilon_c} \geq 0.$$

$\rightarrow$ it suffices to consider the extremum values of ϵ_ρ, ϵ_t, ϵ_r, and ϵ_c;

- process the cases $\alpha < 1/2$ and $1/2 \leq \alpha \leq 1$ separately;
- in each case, lower and upper bound on R...

Analysis of Beebe's algorithm

Theorem

Assuming $u \leq 1/16$ (i.e., $p \geq 4$), the relative error of Beebe's algorithm is bounded by

$$
\begin{aligned}
\chi_4(u) &= (1+2u)\sqrt{\frac{1+u/5}{1+u}} - 1 + u^2 \frac{(1+2u)^2}{(1+u)^2}\left(\frac{\sqrt{5}}{5} + \frac{1}{\frac{5\sqrt{(1+u)\left(1+\frac{u}{5}\right)}}{2} - u} + \frac{2\sqrt{5}}{5(1+2u)}\right), \\
&= \frac{8}{5}u + \left(\frac{3\sqrt{5}}{5} - \frac{2}{25}\right)u^2 + \left(\frac{116}{125} + \frac{14\sqrt{5}}{25}\right)u^3 + \mathrm{O}\left(u^4\right) \\
&\simeq 1.6u + 1.26u^2 + O(u^3) \\
&\leq \frac{8}{5}u + \frac{14}{10}u^2.
\end{aligned}
$$

How do we publish a proof? Have a Maple worksheet publicly available and just get a rough sketch (similar to these slides) in a paper?

And the other algorithms?

- Borges' algorithm: really painful... but we managed to obtain the result;
- Kahan's algorithm:

We may ultimately succeed (already a dirty proof of a bound $1.5765u + \mathcal{O}(u^2)$)
It seems we are approaching a limit...

...and again, as for DW arithmetic, if we fully "expand" the proofs they are terrible (probably unpublishable).

But, really, what were we trying to do?

- obtain the best "algebraic bound": the best one could deduce from the individual bounds on the rounding errors of the operations and a few properties such as Sterbenz Lemma;
- but when the algorithms become complex, does that bound remain tight?
    - we have seen: correlations;
    - even without correlations: tightness requires that for each operation the maximum error is almost reached, with the right signs;
    - in general: probability of this decreases exponentially with number of operations;

$\rightarrow$ Rule of thumb: when the number of operations is no longer small in front of p, little hope of having a worst-case error close to the algebraic bound.

Conclusion

- formal proof and computer algebra:
    - add confidence to the computed bounds;
    - allow us to get to grips with (slightly) bigger algorithms;
    - make it possible to explore many variants of an algorithm (just "replay" the calculation with small modifications);
- long-term goal: use both techniques together (have the computer algebra tool generate a certificate);
- seems we are approaching the limit (in terms of algorithm size) of what can be done "exactly";
- consolation: for larger algorithms, little hope of having a worst-case error close to the algebraic bound;
- what is a *publishable proof*? A human-readable rough sketch along with a Coq file and/or a Maple (or whatever tool) worksheet? What we currently do is just a stylistic exercise...