

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:

$$x = \left( \frac{M}{2^{p-1}} \right) \cdot 2^e = m_0.m_1m_2 \cdots m_{p-1} \cdot 2^e$$

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
- 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 → use small formats whenever possible.
    - small formats → much larger rounding errors → 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  $1/\sqrt{x}$ ,  $\text{hypot}(x, y) = \sqrt{x^2 + y^2}$are “basic building blocks” of numerical computing : users expect same behavior as for  $+$ ,  $-$ ,  $\times$ ,  $\div$ ,  $\sqrt{\cdot}$ .
- 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

$r \leftarrow \text{RN}(y/x)$

$t \leftarrow \text{RN}(1 + r^2)$

$s \leftarrow \text{RN}(\sqrt{t})$

$\rho_2 = \text{RN}(|x| \cdot s)$

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 ?

- ❶ to check, by following the proof **step by step**, that the claimed property holds;
- ❷ 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... embarrassing) 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  
→ 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).

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

# ulp (“unit in last place”) and “absolute error” due to rounding

## Definition (ulp function)

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

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

### Properties:

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

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

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

therefore,

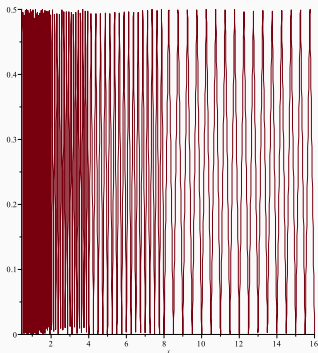
$$|x - \text{RN}(x)| \leq u \cdot |x|, \quad (1)$$

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

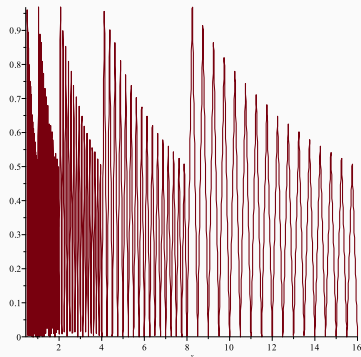
$$\frac{|x - \text{RN}(x)|}{|x|}$$

(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( $a, b$ )

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

## Fast2Sum( $a, b$ )

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

### Barring overflow:

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

Such algorithms: **Error-free transforms.**

# Error-free transforms and double-word arithmetic

## 2Prod( $a, b$ )

$\pi \leftarrow \text{RN}(ab)$

$\rho \leftarrow \text{RN}(ab - \pi)$

**return**  $(\pi, \rho)$

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  $(\pi, \rho)$  returned by Fast2Sum satisfies  $\pi = \text{RN}(ab)$  and  $\rho = (ab) - \pi$ .

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

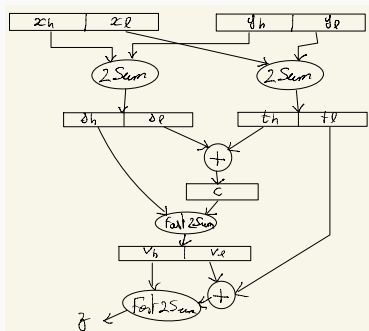
Algorithms for manipulating DW suggested by various authors since 1971.

# DW+DW: “accurate version”

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

## DWPlusDW

- 1:  $(s_h, s_\ell) \leftarrow 2\text{Sum}(x_h, y_h)$
- 2:  $(t_h, t_\ell) \leftarrow 2\text{Sum}(x_\ell, y_\ell)$
- 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**  $(z_h, z_\ell)$





We have (after a rather tedious proof):

### Theorem (Joldeş, Popescu, M., 2017)

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

$$\frac{3u^2}{1-4u} = 3u^2 + 12u^3 + 48u^4 + \dots, \quad (2)$$

That theorem has an interesting history...

**ALGORITHM 6:** *AccurateDWPlusDW*( $x_h, x_l, y_h, y_l$ ). Calculation of  $(x_h, x_l) + (y_h, y_l)$  in binary, precision- $p$ , floating-point arithmetic.

```

1:  $(u_h, u_l) \leftarrow \text{2Sum}(x_h, y_h)$ 
2:  $(t_h, t_l) \leftarrow \text{2Sum}(x_l, y_l)$ 
3:  $c \leftarrow \text{RN}(u_l + t_h)$ 
4:  $(v_h, v_l) \leftarrow \text{Fast2Sum}(u_h, c)$ 
5:  $w \leftarrow \text{RN}(t_l + v_l)$ 
6:  $(z_h, z_l) \leftarrow \text{Fast2Sum}(v_h, w)$ 
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 incorrect, as shown by the following example: If

$$\begin{aligned} x_h &= 9007199254740991, \\ x_l &= -9007199254740991/2^{54}, \\ y_h &= -9007199254740987/2, \text{ and} \\ y_l &= -9007199254740991/2^{56}, \end{aligned} \quad (2)$$

then the relative error of Algorithm 6 is

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

Note that this example is somehow “generic”: In precision- $p$  FP arithmetic, the choice  $x_h = 2^p - 1$ ,  $x_l = -(2^p - 1) \cdot 2^{p-1}$ ,  $y_h = -(2^p - 5)/2$ , and  $y_l = -(2^p - 1) \cdot 2^{p-3}$  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*

$$\frac{3u^2}{1 - 4u} = 3u^2 + 12u^3 + 48u^4 + \dots, \quad (3)$$

which is less than  $3u^2 + 13u^4$  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  $3u^2 + 13u^4$ ) 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  $\text{2Sum}(x_l, y_l)$ , 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  $e_1$  as the error committed at Line 3 of the algorithm:

$$e_1 = c - (x_l + t_h) \quad (4)$$

and  $e_2$  as the error committed at Line 5:

$$e_2 = w - (t_l + v_l). \quad (5)$$

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

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

$$\sigma u \leq x_h \leq 1 - \sigma u. \quad (6)$$

We have  $|x_l| \leq u$  and  $|y_l| \leq \frac{u}{2}u$ , so

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

From Equation (6), we deduce that the floating-point exponent of  $x_h$  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

$$v_h + v_l = x_h + c = x_h + t_h = x + y - t_l.$$

Equations (6) and (7) imply

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

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

$$|e_2| \leq \frac{1}{2}u|p(t_l + v_l)| \leq \frac{1}{2}u|p\left(u^2 + \frac{u}{2}\right)| = \frac{u^2}{2} \quad (8)$$

and

$$|e_1| \leq \frac{1}{2}u|p\left[\frac{1}{2}u|p(x_l + y_l)| + \frac{1}{2}u|p\left((x + y) + \frac{1}{2}u|p(x_l + y_l)\right)|\right]. \quad (9)$$

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

Now,

- If  $|v_h| = \sigma u^2$ , then  $|v_l + t_l| \leq u|v_h| + u^2 = \sigma u^3 + u^2$ , which implies  $|w| = |\text{RN}(t_l + v_l)| \leq \sigma u^2 = |v_h|$ .
- 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  $\sigma u^2$ , which is  $\sigma(1 + 2u)u^2$ . Hence  $|v_h| \geq \sigma u^2/(1 - u)$ , so  $|v_h| \geq u \cdot |v_h| + \sigma u^2 \geq |v_l| + |t_l|$ . So,  $|w| = |\text{RN}(t_l + v_l)| \leq |v_h|$ .

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

$$z_h + z_l = v_h + w = x + y + e_2. \quad (10)$$

Directly using Equation (10) and the bound  $u^2/2$  on  $|e_2|$  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  $\sigma u$ . Hence, since  $|x_l + y_l| \leq (1 + \frac{\sigma}{2})u$ , we have  $|x_l + y_l| \leq \frac{1}{2}(x_h + y_h)$ . Let us now consider the two possible cases:

- If  $-\frac{1}{2}(x_h + y_h) \leq x_l + y_l \leq -\frac{1}{2}(x_h + y_h)$ , which implies  $-\frac{1}{2}x_h \leq t_h \leq -\frac{1}{2}y_h$ , then Sterbenz lemma applies to the floating-point addition of  $x_h$  and  $c = t_h$ . Therefore line 4 of the algorithm results in  $v_h = x_h$  and  $v_l = 0$ . An immediate consequence is  $e_2 = 0$ , so  $z_h + z_l = v_h + w = x + y$ : the computation of  $x + y$  is errorless;

- If  $-\frac{1}{2}(x_b + y_b) < x_r + y_r \leq \frac{1}{2}(x_b + y_b)$ , then  $\frac{1}{2}(x_r + y_r) \leq \frac{1}{2}(x_b + y_b + x_r + y_r) = \frac{1}{2}(x + y)$ , and  $-\frac{1}{2}(x + y) < \frac{1}{2}(x_r + y_r)$ . Hence,  $|x_r + y_r| < |x + y|$ , so  $\text{ulp}(x_r + y_r) \leq \text{ulp}(x + y)$ . Combined with Equation (9), this gives

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

## 2. If $-x_b/2 < y_b \leq x_b$

Notice that we have  $x_b/2 < x_b + y_b \leq 2x_b$ , so  $x_b/2 \leq s_b \leq 2x_b$ . Also notice that we have  $|x_r| \leq u$ .

- If  $\frac{1}{2} < x_b + y_b \leq 2 - 4u$ . Define

$$\sigma = \begin{cases} 1 & \text{if } x_b + y_b \leq 1 - 2u, \\ 2 & \text{if } 1 - 2u < x_b + y_b \leq 2 - 4u. \end{cases}$$

We have

$$\frac{\sigma}{2}(1 - 2u) \leq s_b \leq \sigma(1 - 2u) \quad \text{and} \quad |s_r| \leq \frac{\sigma}{2}u. \quad (11)$$

When  $\sigma = 1$ , we necessarily have  $-x_b/2 < y_b < 0$ , so  $|y_r| \leq u/2$ . And when  $\sigma = 2$ ,  $|y_b| \leq x_b \leq 2 - 2u$  implies  $|y_r| \leq u$ . Hence we always have  $|y_r| \leq \frac{\sigma}{2}u$ . This implies  $|x_r + y_r| \leq (1 + \sigma/2)u$ , therefore

$$|t_b| \leq \left(1 + \frac{\sigma}{2}\right)u \quad \text{and} \quad |t_r| \leq u^2. \quad (12)$$

Now,  $|s_r + t_b| \leq (1 + \sigma)u$ , so

$$|c| \leq (1 + \sigma)u \quad \text{and} \quad |e_c| \leq \sigma u^2. \quad (13)$$

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

$$v_b + v_r = s_b + c.$$

Therefore  $|v_b + v_r| = |s_b + c| \leq \sigma(1 - 2u) + (1 + \sigma)u \leq \sigma$ . This implies

$$|v_b| \leq \sigma \quad \text{and} \quad |v_r| \leq \frac{\sigma}{2}u. \quad (14)$$

Thus  $|t_r + v_r| \leq u^2 + \frac{\sigma}{2}u$ , so

$$|w| \leq \frac{\sigma}{2}u + u^2 \quad \text{and} \quad |e_c| \leq \frac{\sigma}{2}u^2. \quad (15)$$

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

Therefore,

$$z_b + z_r = x + y + \eta,$$

with  $|\eta| = |e_1 + e_2| \leq \frac{3}{2}u^2$ . Since

$$x + y \geq (x_b - u) + (y_b - u/2) > \begin{cases} \frac{1}{2} - \frac{3}{2}u & \text{if } \sigma = 1, \\ 1 - 4u & \text{if } \sigma = 2, \end{cases}$$

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

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

15res:10

M. Joldes et al.

- If  $2 - 4u < x_b + y_b \leq 2x_b$ , then  $2 - 4u \leq s_b \leq \text{RN}(2x_b) = 2x_b \leq 4 - 4u$  and  $|s_r| \leq 2u$ . We have

$$t_b + t_r = x_r + y_r,$$

with  $|x_r + y_r| \leq 2u$ , hence  $|t_b| \leq 2u$ , and  $|t_r| \leq u^2$ . Now,  $|s_r + t_b| \leq 4u$ , so  $|c| \leq 4u$ , and  $|e_1| \leq 2u^2$ . Since  $s_b \geq 2 - 4u$  and  $|c| \leq 4u$ , if  $p \geq 3$ , then Algorithm Fast2Sum introduces no error at line 4 of the algorithm. Therefore,

$$v_b + v_r = s_b + c \leq 4 - 4u + 4u = 4,$$

so  $v_b \leq 4$  and  $|v_r| \leq 2u$ . Thus,  $|t_r + v_r| \leq 2u + u^2$ . Hence, either  $|t_r + v_r| < 2u$  and  $|e_c| \leq \frac{1}{2} \text{ulp}(t_r + v_r) \leq u^2$ , or  $2u \leq t_r + v_r \leq 2u + u^2$ , in which case  $w = \text{RN}(t_r + v_r) = 2u$  and  $|e_c| \leq u^2$ . In all cases,  $|e_c| \leq u^2$ . Also,  $s_b \geq 2 - 4u$  and  $|c| \leq 4u$  imply  $v_b \geq 2 - 8u$ , and  $|t_r + v_r| \leq 2u + u^2$  implies  $|w| \leq 2u$ . Hence if  $p \geq 3$ , then Algorithm Fast2Sum introduces no error at line 6 of the algorithm.

All this gives

$$z_b + z_r = v_b + w = x + y + \eta,$$

with  $|\eta| = |e_1 + e_2| \leq 3u^2$ .

Since  $x + y \geq (x_b - u) + (y_b - u) > 2 - 6u$ , 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.,  $p \geq 6$ ) this is always less than  $3u^2 + 13u^3$ .  $\square$

The bound (3) is probably not optimal. The largest relative error we have obtain through many tests is around  $2.23 \times 2^{-39} = 2.25u^2$ . An example is the input values given in Equation (2), for which, with  $p = 53$  (binary64 arithmetic), we obtain a relative error equal to  $2.24999999999999956 \dots \times 2^{-396}$ .

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

```

Li et al. (2000, 2002) claim that in binary64 arithmetic ( $p = 53$ ) the relative error of Algorithm 6 is upper bounded by  $2 \cdot 2^{-166}$ . This bound is incorrect, in which one of the operations is not performed.

then the relative error of Algorithm 6 is  $2.2499999999999999$  in which one of the operations is not performed. The returned result is  $2\text{Sum}(x_\ell, y_\ell)$ , without loss of generality, we assume  $x_h \neq 0$ .

Note that this example is somehow "generic".

$2^p - 1, x_\ell = -(2^p - 1), 2^{p-1}, y_h = -(2^p - 5)/2, y_\ell = 1$

that is asymptotically equivalent (as  $p$  goes to infinity).

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

**THEOREM 3.1.** If  $p \geq 3$ , then the relative error of Algorithm 6 is

by

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

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  $2\text{Sum}(x_\ell, y_\ell)$ , 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  $e_1$  as the error committed at Line 3 of the algorithm:

$$e_1 = c - (x_\ell + y_\ell) \quad (4)$$

and  $e_2$  as the error committed at Line 5:

$$e_2 = w - (t_\ell + v_\ell). \quad (5)$$

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

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  $-x_h < y_h \leq (1 - \sigma) \cdot (x - 2)$ , so  $0 \leq x_h + y_h \leq 1 + \sigma \cdot (\frac{3}{2} - 1) \leq 1 - \sigma u$ . Also, since  $x_h$  is a multiple of  $2u$  and  $y_h$  is a multiple of  $\sigma u$ ,  $x_h + y_h$  is a multiple of  $\sigma u$ . Since  $x_h$  is nonzero, we finally obtain

$$\sigma u \leq x_h \leq 1 - \sigma u. \quad (6)$$

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

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

From Equation (6), we deduce that the floating-point exponent of  $x_h$  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

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

Equations (6) and (7) imply

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

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

$$|e_1| \leq \frac{1}{2}u\text{ulp}(t_\ell + v_\ell) \leq \frac{1}{2}u\text{ulp}\left(u^2 + \frac{u}{2}\right) = \frac{u^2}{2} \quad (8)$$

and

$$|e_2| \leq \frac{1}{2}u\text{ulp}\left[\frac{1}{2}u\text{ulp}(x_\ell + y_\ell) + \frac{1}{2}u\text{ulp}\left((x + y) + \frac{1}{2}u\text{ulp}(x_\ell + y_\ell)\right)\right]. \quad (9)$$

Lemma 2.1 and  $|v_h| \geq \sigma u$  imply that either  $x_h + t_h = 0$ , or  $|v_h| = |\text{RN}(x_h + c)| = |\text{RN}(x_h + t_h)| \geq \sigma u^2$ . If  $x_h + t_h = 0$ , 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  $|w| = |\text{RN}(t_\ell + v_\ell)| \leq \sigma u^2 = |v_h|$ .
- 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  $\sigma u^2$ , 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,  $|w| = |\text{RN}(t_\ell + v_\ell)| \leq |v_h|$ .

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

$$z_h + z_\ell = v_h + w = x + y + e_2. \quad (10)$$

Directly using Equation (10) and the bound  $u^2/2$  on  $|e_1|$  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  $\sigma u$ . 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  $-\frac{3}{2}x_h \leq t_h \leq -\frac{1}{2}x_h$ , then Sterbenz Lemma applies to the floating-point addition of  $x_h$  and  $c = t_h$ . Therefore line 4 of the algorithm results in  $v_h = x_h$  and  $v_\ell = 0$ . An immediate consequence is  $e_2 = 0$ , 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:  $(v_h, v_\ell) \leftarrow \text{Fast2Sum}(x_h, y_h)$ 
2:  $(u_h, u_\ell) \leftarrow \text{Fast2Sum}(x_\ell, y_\ell)$ 
3:  $c \leftarrow \text{RN}(v_\ell + u_\ell)$ 
4:  $(v_h, v_\ell) \leftarrow \text{Fast2Sum}(v_h, c)$ 
5:  $w \leftarrow \text{RN}(v_\ell + u_\ell)$ 
6:  $(v_h, v_\ell) \leftarrow \text{Fast2Sum}(v_h, w)$ 
7: return  $(x_h, x_\ell)$ 

```

Li et al. (2000, 2002) claim that  $T$  is upper bounded by  $2 \cdot 2^{-16k}$ . The

then the relative error of Algorithm 6 is

$$2.24999999999999956 \dots$$

Note that this example is somehow “generic”: In precision- $p$  FP arithmetic,  $2^p - 1, x_\ell = -(2^p - 1) \cdot 2^{-p-1}, y_h = -(2^p - 5)/2$ , 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*

$$\frac{3u^2}{1 - 4u} = 3u^2 + 12u^3 + 48u^4 + \dots, \quad (3)$$

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  $|x_\ell + y_\ell| = 2 \cdot |v_\ell + u_\ell|$ .

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

and  $v_\ell$  as the error, we can apply to the floating-point

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  $c = \text{RN}(v_h)$  is:

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

$$\sigma u \leq s_h \leq 1 - \sigma u. \quad (6)$$

$|u| \leq u$  and  $|y_\ell| \leq \frac{u}{2}$ , so

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

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

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

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

$$|e_\ell| \leq \frac{1}{2} \text{ulp}(t_\ell + v_\ell) \leq \frac{1}{2} \text{ulp}\left(u^2 + \frac{u}{2}\right) = \frac{u^2}{2} \quad (8)$$

and

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

Lemma 2.1 and  $|s_h| \geq \sigma u$  imply that either  $s_h = 0$ , or  $|v_h| = |\text{RN}(s_h + c)| = |\text{RN}(s_h + t_h)| \geq \sigma u^2$ . 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  $|w| = |\text{RN}(t_\ell + v_\ell)| \leq \sigma u^2 = |v_h|$ .
- 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  $\sigma u^2$ , which is  $\sigma(1 + 2u)u^2$ . Hence  $|v_h| \geq \sigma u^2/(1 - u)$ , so  $|v_h| \geq u \cdot |v_h| + u^2 \geq |v_\ell| + |t_\ell|$ . So,  $|w| = |\text{RN}(t_\ell + v_\ell)| \leq |v_h|$ .

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

$$z_h + z_\ell = v_h + w = x + y + e_2. \quad (10)$$

By using Equation (10) and the bound  $u^2/2$  on  $|e_\ell|$  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,  $s_h + y_h$  is a nonzero multiple of  $\sigma u$ . Hence, since  $|x_\ell + y_\ell| \leq (1 + \frac{\sigma}{2})u$ , we have  $|x_\ell + y_\ell| \leq \frac{3}{2}(s_h + y_h)$ . Let us now consider the two possible cases:

- If  $-\frac{3}{2}(s_h + y_h) \leq x_\ell + y_\ell \leq -\frac{1}{2}(s_h + y_h)$ , 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  $e_2 = 0$ , so  $z_h + z_\ell = v_h + w = x + y$ : the computation of  $x + y$  is errorless;

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

$$|e_1| \leq \frac{1}{2} \text{ulp} \left( \frac{3}{2} \text{ulp}(x + y) \right) \leq 2^{-p} \text{ulp}(x + y) \leq 2 \cdot 2^{-2p} \cdot (x + y).$$

## 2. If $-x_h/2 < y_h \leq x_h$

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

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

We have

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

The bound (3) is probably

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

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

$$|c| \leq (1 + \sigma)u \quad \text{and} \quad |e_1| \leq \sigma u^2. \quad (13)$$

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

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

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

$$|v_h| \leq \sigma \quad \text{and} \quad |v_\ell| \leq \frac{\sigma}{2}u. \quad (14)$$

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

$$|w| \leq \frac{\sigma}{2}u + u^2 \quad \text{and} \quad |e_2| \leq \frac{\sigma}{2}u^2. \quad (15)$$

From Equations (11) and (13), we deduce  $s_h + c \geq \frac{\sigma}{2} - u(2\sigma + 1)$ , so  $|v_h| \geq \frac{\sigma}{2} - u(2\sigma + 1)$ . If  $p \geq 3$ , then  $|v_h| \geq |w|$ , 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  $|\eta| = |e_1 + e_2| \leq \frac{3}{2}\sigma u^2$ . Since

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

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

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

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

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

with  $|x_\ell + y_\ell| \leq 2u$ , hence  $|t_h| \leq 2u$ , and  $|t_\ell| \leq u^2$ . Now,  $|s_\ell + t_h| \leq 4u$ , so  $|c| \leq 4u$ , and  $|e_1| \leq 2u^2$ . Since  $s_h \geq 2 - 4u$  and  $|c| \leq 4u$ , if  $p \geq 3$ , then Algorithm Fast2Sum introduces no error at line 4 of the algorithm. Therefore,

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

so  $v_h \leq 4$  and  $|v_\ell| \leq 2u$ . Thus,  $|t_\ell + v_\ell| \leq 2u + u^2$ . Hence, either  $|t_\ell + v_\ell| < 2u$  and  $|e_2| \leq \frac{1}{2} \text{ulp}(t_\ell + v_\ell) \leq u^2$ , or  $2u \leq t_\ell + v_\ell \leq 2u + u^2$ , in which case  $w = \text{RN}(t_\ell + v_\ell) = 2u$  and  $|e_2| \leq u^2$ . In all cases,  $|e_2| \leq u^2$ . Also,  $s_h \geq 2 - 4u$  and  $|c| \leq 4u$  imply  $v_h \geq 2 - 8u$ , and  $|c| \leq 2u + u^2$  implies  $|w| \leq 2u$ . Hence if  $p \geq 3$ , then Algorithm Fast2Sum introduces no error at line 6 of the algorithm.

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

with  $|\eta| = |e_1 + e_2| \leq 3u^2$ .

Since  $x + y \geq (x_h - u) + (y_h - u) > 2 - 6u$ , 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.,  $p \geq 6$ ) this is always less than  $3u^2 + 13u^3$ . □

The bound (3) is *probably* not optimal. The largest relative error we have obtained through many tests is around  $2.25 \times 2^{-29} = 2.25u^2$ . An example is the input values given in Equation (2), for which, with  $p = 53$  (binary64 arithmetic), we obtain a relative error equal to  $2.24999999999999956 \dots \times 2^{-106}$ .

## DW+DW: “accurate version”

So the theorem gives an error bound

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

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

$$x_h = 1$$

$$x_\ell = u - u^2$$

$$y_h = -\frac{1}{2} + \frac{u}{2}$$

$$y_\ell = -\frac{u^2}{2} + u^3.$$

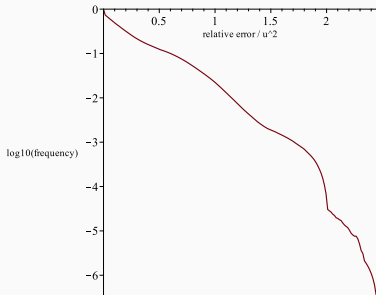
**Exercise:** all these values are FP numbers.

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



## 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.

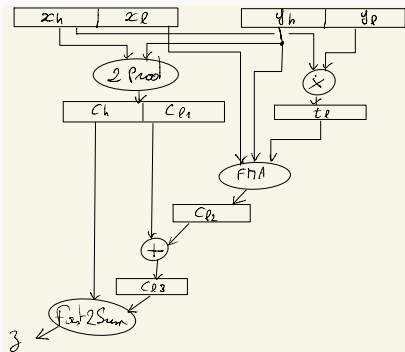


$\log_{10}$  of the frequency of cases for which the relative error of DWPlusDW is  $\geq \lambda u^2$  as a function of  $\lambda$ .

- 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

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



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?)

- after a few nights of bad sleep, turn-around. . . that also improved the bound:  $6u^2 \rightarrow 5u^2$ !
- 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  $\infty$ , even if exact result  $\ll \Omega$ , or
  - **spurious underflow**: very inaccurate result if subnormal intermediate values.

# The naive algorithm

## NaiveHypot

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

- 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  $96 \times 2^{-542}$ , exact result  $97 \times 2^{-542}$ .

$\Rightarrow$  need to **scale** the operands.

# Simple scaling

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

- several versions;
- this one requires availability of an FMA  
(*fused multiply-add*:  $\text{RN}(ab + c)$ );
- 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)

```
1: if  $|x| < |y|$  then
2:   swap( $x, y$ )
3: end if
4:  $r \leftarrow \text{RN}(y/x)$ 
5:  $t \leftarrow \text{RN}(1 + r^2)$ 
6:  $s \leftarrow \text{RN}(\sqrt{t})$ 
7:  $\epsilon \leftarrow \text{RN}(t - s^2)$ 
8:  $c \leftarrow \text{RN}(\epsilon/(2s))$ 
9:  $\nu \leftarrow \text{RN}(|x| \cdot c)$ 
10:  $\rho_3 \leftarrow \text{RN}(|x| \cdot s + \nu)$ 
```

- 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)

```
1: if  $|x| < |y|$  then
2:   swap( $x, y$ )
3: end if
4:  $(s_x^h, s_x^\ell) \leftarrow \text{Fast2Mult}(x, x)$ 
5:  $(s_y^h, s_y^\ell) \leftarrow \text{Fast2Mult}(y, y)$ 
6:  $(\sigma_h, \sigma_\ell) \leftarrow \text{Fast2Sum}(s_x^h, s_y^h)$ 
7:  $s \leftarrow \text{RN}(\sqrt{\sigma_h})$ 
8:  $\delta_s \leftarrow \text{RN}(\sigma_h - s^2)$ 
9:  $\tau_1 \leftarrow \text{RN}(s_x^\ell + s_y^\ell)$ 
10:  $\tau_2 \leftarrow \text{RN}(\delta_s + \sigma_\ell)$ 
11:  $\tau \leftarrow \text{RN}(\tau_1 + \tau_2)$ 
12:  $c \leftarrow \text{RN}(\tau/s)$ 
13:  $\rho_4 \leftarrow \text{RN}(0.5c + s)$ 
```

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

# Kahan's algorithm (1987)

```
1:  $\delta \leftarrow \text{RN}(x - y)$ 
2: if  $\delta > y$  then
3:    $r \leftarrow \text{RN}(x/y)$ 
4:    $t \leftarrow \text{RN}(1 + r^2)$ 
5:    $s \leftarrow \text{RN}(\sqrt{t})$ 
6:    $z \leftarrow \text{RN}(r + s)$ 
7: else
8:    $r_2 \leftarrow \text{RN}(\delta/y)$ 
9:    $tr_2 \leftarrow \text{RN}(2r_2)$ 
10:   $r_3 \leftarrow \text{RN}(tr_2 + r_2^2)$ 
11:   $r_4 \leftarrow \text{RN}(2 + r_3)$ 
12:   $s_2 \leftarrow \text{RN}(\sqrt{r_4})$ 
13:   $d = \text{RN}(R_2 + s_2)$ 
14:   $q = \text{RN}(r_3/d)$ 
15:   $r_5 \leftarrow \text{RN}(P_\ell + q)$ 
16:   $r_6 \leftarrow \text{RN}(r_5 + r_2)$ 
17:   $z \leftarrow \text{RN}(P_h + r_6)$ 
18: end if
19:  $z_2 \leftarrow \text{RN}(y/z)$ 
20:  $\rho_7 \leftarrow \text{RN}(x + z_2)$ 
```

In this presentation, requires an FMA . We assume  $0 \leq y \leq x$ . Precomputed constants  $R_2 = \text{RN}(\sqrt{2})$ ,  $P_h = \text{RN}(1 + \sqrt{2})$ , and  $P_\ell = \text{RN}(1 + \sqrt{2} - P_h)$ . 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

$$x_k = x_i \text{ op } x_j \quad \text{or} \quad x_k = \text{sqrt}(x_i)$$

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

- Computed values:

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

- basic relations:

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

(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**.
- 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;
- use of heuristics;

# Prototype implementation: illustration with the naive alg.

## NaiveHypot

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

```
> 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_sx = 1, _eps_sy = 1 }

> quad:=BoundQuadraticTerm(linpart,sys);

    quad := 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**

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

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

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

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

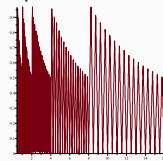
- The analysis consists in iteratively computing relative error bounds  $\epsilon_k^\ell(u)$  and  $\epsilon_k^r(u)$  such that (here, for positive  $x_k$  and  $x_k^*$ )

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



# 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**.
- **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 $\text{RN}(\text{RN}(x \cdot x) - 2)$

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

```
1: if  $|x| < |y|$  then  
2:    $\text{swap}(x, y)$   
3: end if  
4:  $r \leftarrow \text{RN}(y/x)$   
5:  $t \leftarrow \text{RN}(1 + r^2)$   
6:  $s \leftarrow \text{RN}(\sqrt{t})$   
7:  $\epsilon \leftarrow \text{RN}(t - s^2)$   
8:  $c \leftarrow \text{RN}(\epsilon/(2s))$   
9:  $\nu \leftarrow \text{RN}(|x| \cdot c)$   
10:  $\rho_3 \leftarrow \text{RN}(|x| \cdot s + \nu)$ 
```

# Analysis of Beebe's algorithm

Simplification:  $x \geq y > 0$

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

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

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

- define  $\alpha$  by  $y = \alpha x$ , so that  $r = \text{RN}(\alpha)$ ;

- $r = \alpha + u\epsilon_r$ , 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

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

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

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

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

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

with  $|\epsilon_c| \leq 1$ .

# Analysis of Beebe's algorithm

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

- $\nu = xc(1 + u\epsilon_\nu)$  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:

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

## Lemma

*The relative error of the algorithm is*

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

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  $\epsilon_r, \epsilon_t$  and  $\epsilon_\rho$ ,
- $\epsilon_r \leq 1/4$  if  $\alpha \leq 1/2$ ,  $\epsilon_r \leq 1/2$  otherwise,
- $\epsilon_t, \epsilon_\rho \leq 1$

→ max. value 8/5;

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

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

→ it suffices to consider the extremum values of  $\epsilon_\rho$ ,  $\epsilon_t$ ,  $\epsilon_r$ , and  $\epsilon_c$ ;

- 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

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

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;
- **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. . .