Exploiting Structure
in Floating-Point Arithmetic

Claude-Pierre Jeannerod[(✉)]

Inria, Laboratoire LIP (U. Lyon, CNRS, ENSL, Inria, UCBL),
ENS de Lyon, 46 allée d'Italie, 69364 Lyon Cedex 07, France
`claude-pierre.jeannerod@inria.fr`

Abstract. The analysis of algorithms in IEEE floating-point arithmetic is most often carried out via repeated applications of the so-called standard model, which bounds the relative error of each basic operation by a common epsilon depending only on the format. While this approach has been eminently useful for establishing many accuracy and stability results, it fails to capture most of the low-level features that make floating-point arithmetic so highly *structured*. In this paper, we survey some of those properties and how to exploit them in rounding error analysis. In particular, we review some recent improvements of several classical, Wilkinson-style error bounds from linear algebra and complex arithmetic that all rely on such structure properties.

Keywords: Floating-point arithmetic · IEEE standard 754-2008 · Rounding error analysis · High relative accuracy

1   Introduction

When analyzing a priori the behaviour of a numerical algorithm in IEEE floating-point arithmetic, one most often relies exclusively on the so-called *standard model*: for base β, precision p, and rounding to nearest, this model says that the result $\hat{r}$ of each basic operation op $\in \{+, -, \times, /\}$ on two floating-point numbers x and y satisfies

$$\hat{r} = (x \operatorname{op} y)(1 + \delta), \qquad |\delta| \leqslant u \tag{1}$$

with $u = \frac{1}{2}\beta^{1-p}$ the *unit roundoff*. (Similar relations are also assumed for the square root and the fused multiply-add (FMA) operations.)

This model has been used long before the appearance of the first version of IEEE standard 754 [17,18], and the fact that it gives backward error results is already emphasized by Wilkinson [43]: considering for example floating-point addition, it is easily deduced from (1) that $\hat{r}$ is the exact sum of the slightly perturbed data $x(1 + \delta)$ and $y(1 + \delta)$, and, applying this repeatedly, that the computed approximation to the sum of n floating-point numbers x_i has the form $\sum_{i=1}^{n} \widetilde{x}_i$ with $|\widetilde{x}_i - x_i|/|x_i| \leqslant (1 + u)^{n-1} - 1 = (n-1)u + O(u^2)$ for all i.

Backward error analysis based on the standard model (1) has developed far beyond this basic example and turned out to be eminently useful for establishing many accuracy and stability results, as Higham's treatise [14] shows.

Although the standard model holds for IEEE 754 arithmetic as long as underflow and overflow do not occur, it fails, however, to capture most of the low-level features that make this arithmetic so highly *structured*. For example, by ensuring a relative error less than one, (1) implies that $\hat{r}$ has the same sign as the exact value $x \operatorname{op} y$, but it does not say that δ should be zero when $x \operatorname{op} y$ is a floating-point number.

Such low-level features are direct consequences of the two main ingredients of IEEE standard arithmetic. The first ingredient is the set $\mathbb{F}$ of floating-point numbers, which (up to ignoring underflow and overflow) can be viewed as

$$\mathbb{F} = \{0\} \cup \big\{ M\beta^e \,:\, M, e \in \mathbb{Z}, \ \beta^{p-1} \leqslant |M| < \beta^p \big\}. \tag{2}$$

The second ingredient is a rounding function $\mathrm{RN} : \mathbb{R} \to \mathbb{F}$, which maps any real number to a nearest element in $\mathbb{F}$:

$$|\mathrm{RN}(t) - t| = \min_{f \in \mathbb{F}} |f - t| \qquad \text{for all } t \in \mathbb{R}, \tag{3}$$

with ties broken according to a given rule (say, round *to nearest even*). This rounding function is then used by IEEE standard arithmetic to operate on floating-point data as follows: in the absence of underflow and overflow, $x \operatorname{op} y$ must be computed as

$$\hat{r} = \mathrm{RN}(x \operatorname{op} y).$$

This way of combining the structured data in (2) and the minimization property (3) implies that $\hat{r}$ enjoys many more mathematical properties than just (1).

The goal of this paper is to show the benefits of exploiting such lower level features in the context of rounding error analysis. We begin by recalling some of these features in Sect. 2. Although the list given there is by no means exhaustive (cf. Rump, Ogita, and Oishi [37, Sect. 2]), it should already give a good idea of what can be deduced from (2) and (3). We then review some recent improvements of several classical, Wilkinson-style error bounds from linear algebra and complex arithmetic that all rely on such structure properties. Specifically, we will see in Sect. 3 that various general algorithms (for summation, inner products, matrix factorization, polynomial evaluation, ...) now have a priori error bounds which are both simpler and sharper than the classical ones. In Sect. 4 we will focus on more specific algorithms for core computations like 2×2 determinants or complex products, and show that in such cases exploiting the low-level features of IEEE standard arithmetic leads to proofs of high relative accuracy and tight error bounds.

Throughout this paper we assume for simplicity that β is even, that RN rounds to nearest even, and that underflow and overflow do not occur. (For summation, however, the results presented here still hold in the presence of underflow, since then floating-point addition is known to be exact; see Hauser [13].)

For more on floating-point arithmetic, we refer to the complementary texts by Brent and Zimmermann [3, Sect. 3], Corless and Fillion [6, Appendix A], Demmel [9, Sect. 1.5], Goldberg [10], Golub and Van Loan [11, Sect. 2.7], Higham [14, Sect. 2], [15], Knuth [27, Sect. 4.2], Monniaux [29], Muller et al. [31], Overton [33], Priest [34], Trefethen [41], and Trefethen and Bau [42, Sect. 13].

2   Low-Level Properties

Structure of the Floating-Point Number Set. By construction, the set $\mathbb{F}$ contains zero, has the symmetry property $\mathbb{F} = -\mathbb{F}$, and is invariant under *scaling* (that is, multiplication by an integer power of the base): $x\beta^k \in \mathbb{F}$ for all $x \in \mathbb{F}$ and $k \in \mathbb{Z}$. More precisely, every element of $\mathbb{F}$ is a multiple (by some $\pm\beta^k$) of an element of the subset $\mathbb{F} \cap [1, \beta)$. The elements of this subset have the form $1 + j\beta^{1-p}$, where j is an integer such that $0 \leqslant j < (\beta - 1)\beta^{p-1}$ and, since $u = \frac{1}{2}\beta^{1-p}$, this can be expressed concisely as follows:

$$\mathbb{F} \cap [1, \beta) = \{1, 1 + 2u, 1 + 4u, 1 + 6u, \ldots\}.$$

The numbers lying exactly halfway between two consecutive elements of $\mathbb{F}$, such as for example $1 + u$ and $1 + 3u$, are called *midpoints* for $\mathbb{F}$.

Some First Consequences of Rounding to Nearest. Since by definition $|\mathrm{RN}(t) - t| \leqslant |f - t|$ for all f in $\mathbb{F}$, choosing $t = x + \epsilon$ with $x \in \mathbb{F}$ and $\epsilon \in \mathbb{R}$ gives $|\mathrm{RN}(x + \epsilon) - (x + \epsilon)| \leqslant |\epsilon|$. With $\epsilon = 0$ we recover the obvious property that rounding a floating-point number leaves it unchanged:

$$x \in \mathbb{F} \quad \Rightarrow \quad \mathrm{RN}(x) = x. \tag{4}$$

Setting $\epsilon = y$ with y in $\mathbb{F}$, we deduce further that for floating-point addition the error bound implied by the standard model (1) can be refined slightly:

$$x, y \in \mathbb{F} \quad \Rightarrow \quad |\mathrm{RN}(x + y) - (x + y)| \leqslant \min\{u|x + y|, |x|, |y|\}. \tag{5}$$

(Similarly, a sharper bound can be deduced for the FMA operation by taking $\epsilon = yz$.) We will see in Sect. 3 how to exploit such a refinement in the context of floating-point summation.

Besides (4), other basic features include the following ones:

$$t \in \mathbb{R} \quad \Rightarrow \quad |\mathrm{RN}(t)| = \mathrm{RN}(|t|), \tag{6}$$

$$t \in \mathbb{R}, \quad k \in \mathbb{Z} \quad \Rightarrow \quad \mathrm{RN}(t\beta^k) = \mathrm{RN}(t)\beta^k, \tag{7}$$

$$t, t' \in \mathbb{R}, \quad t \leqslant t' \quad \Rightarrow \quad \mathrm{RN}(t) \leqslant \mathrm{RN}(t'). \tag{8}$$

Combining (4) with the monotonicity property (8), we see for example that if $x \in \mathbb{F}$ satisfies $x \leqslant t$ for some real t, then $x \leqslant \mathrm{RN}(t)$.

As another example, we note that (4), (7), and (8) already suffice to prove that the classical approximation to the mean of two floating-point numbers behaves as expected in base 2 (but not in base 10): using (7) and then (4) gives $\hat{r} := \mathrm{RN}(\mathrm{RN}(x+y)/2) = \mathrm{RN}((x+y)/2)$; then, using $f := \min\{x, y\} \leqslant (x+y)/2 \leqslant \max\{x, y\} =: g$ together with (8), we deduce that $\mathrm{RN}(f) \leqslant \hat{r} \leqslant \mathrm{RN}(g)$ and, applying (4) again, we conclude that $f \leqslant \hat{r} \leqslant g$.

The Functions ufp and ulp. A very convenient tool to go beyond the standard model is provided by the notion of *unit in the first place* (ufp), defined in [37] as

$$\mathrm{ufp}(t) = \begin{cases} 0 & \text{if } t = 0, \\ \beta^{\lfloor \log_\beta |t| \rfloor} & \text{if } t \in \mathbb{R}\backslash\{0\}. \end{cases}$$

Its relationship with the classical notion of *unit in the last place* (ulp) is via the equality $\mathrm{ulp}(t) = 2u\,\mathrm{ufp}(t)$, and its definition implies immediately that

$$t \in \mathbb{R}\backslash\{0\} \quad \Rightarrow \quad \mathrm{ufp}(t) \leqslant |t| < \beta\mathrm{ufp}(t). \tag{9}$$

From (4), (6), (8), it then follows that

$$t \in \mathbb{R} \quad \Rightarrow \quad \mathrm{ufp}(t) \leqslant |\mathrm{RN}(t)| \leqslant \beta\mathrm{ufp}(t).$$

Thus, $\mathrm{RN}(t)$ belongs to a range for which the distance between two consecutive floating-point numbers is exactly $2u\,\mathrm{ufp}(t)$, and being nearest to t implies

$$|\mathrm{RN}(t) - t| \leqslant u\,\mathrm{ufp}(t).$$

In terms of ulp's, this is just the usual half-an-ulp absolute error bound (attained at every midpoint) and, dividing further by $|t| > 0$, we arrive at

$$t \in \mathbb{R}\backslash\{0\} \quad \Rightarrow \quad \frac{|\mathrm{RN}(t) - t|}{|t|} \leqslant u\frac{\mathrm{ufp}(t)}{|t|}. \tag{10}$$

This inequality is interesting for at least three reasons. First, recalling (9), it allows us to recover the uniform bound u claimed by the standard model (1). Second, it shows that the relative error can be bounded by about u/β instead of u when $|t|$ approaches its upper bound $\beta\mathrm{ufp}(t)$; this is related to a phenomenon called *wobbling precision* [14, p. 39] and indicates that when deriving sharp error bounds the most difficult cases are likely to occur when $|t|$ lies in the leftmost part of its range $[\mathrm{ufp}(t), \beta\mathrm{ufp}(t))$. Third, it makes it easy to check that the bound u is in fact never attained, as noted in [14, p. 38], since either $|t| = \mathrm{ufp}(t) \in \mathbb{F}$ or $\mathrm{ufp}(t)/|t| < 1$. Indeed, the following slightly stronger statement holds:

$$t \in \mathbb{R}\backslash\{0\} \quad \Rightarrow \quad \frac{|\mathrm{RN}(t) - t|}{|t|} \leqslant \frac{u}{1+u}. \tag{11}$$

If $|t| \geqslant (1+u)\mathrm{ufp}(t)$, the above inequality follows directly from the one in (10). Else, rounding to nearest implies that $|\mathrm{RN}(t)| = \mathrm{ufp}(t) \leqslant |t| < (1+u)\mathrm{ufp}(t)$ and, recalling that t has the same sign as its rounded value, we conclude that

$$\frac{|\mathrm{RN}(t) - t|}{|t|} = 1 - \frac{\mathrm{ufp}(t)}{|t|} < 1 - \frac{1}{1+u} = \frac{u}{1+u}.$$

The bound in (11) is given by Knuth in [27, p. 232] and, in the special case where $t = x + y$ or $t = xy$ with $x, y \in \mathbb{F}$, it was already noted by Dekker [8] (in base 2) and then by Holm [16] (in any base). Furthermore, it turns out to be attained if and only if t is the midpoint $\pm(1 + u)\mathrm{ufp}(t)$; see [25]. This best possible bound refines the standard model (1) only slightly, but we shall see in the rest of this paper that it can be worth exploiting in various situations.

Exact Floating-Point Subtraction and EFTs. We now briefly review what can be obtained *exactly* using floating-point and rounding to nearest. A first classical result is Sterbenz' theorem [40, p. 138], which ensures that floating-point subtraction is exact when the two operands are close enough to each other:

$$x, y \in \mathbb{F}, \quad y/2 \leqslant x \leqslant 2y \qquad \Rightarrow \qquad x - y \in \mathbb{F}.$$

Another exactness property is that the absolute error due to floating-point addition or multiplication is itself a floating-point number:

$$x, y \in \mathbb{F}, \quad \mathrm{op} \in \{+, \times\} \qquad \Rightarrow \qquad x \mathop{\mathrm{op}} y - \mathrm{RN}(x \mathop{\mathrm{op}} y) \in \mathbb{F}.$$

Furthermore, various floating-point algorithms are available for computing simultaneously the rounded value $\hat{r} = \mathrm{RN}(x \mathop{\mathrm{op}} y)$ and the exact value of the associated rounding error $e = x \mathop{\mathrm{op}} y - \hat{r}$. For addition, these are the Fast2Sum algorithm of Kahan [26] and Dekker [8], and the more general 2Sum algorithm of Knuth [27] and Møller [28]. For multiplication, it suffices to use the FMA operation as follows:

$$\hat{r} \leftarrow \mathrm{RN}(xy), \qquad e \leftarrow \mathrm{RN}(xy - \hat{r}). \tag{12}$$

(If no FMA is available, the pair $(\hat{r}, e)$ can be obtained using 7 multiplications and 10 additions, as shown by Dekker in [8].) These algorithms define in each case a so-called *error-free transformation* (EFT) [32], which maps $(x, y) \in \mathbb{F}^2$ to $(\hat{r}, e) \in \mathbb{F}^2$ such that $x \mathop{\mathrm{op}} y = \hat{r} + e$. In Sect. 4 we will see in particular how to exploit the transformation given by (12), possibly in combination with Sterbenz's theorem. For more examples of EFT-based, provably accurate algorithms—especially in the context of summation and elementary function evaluation—we refer to [35] and [31] and the references therein.

3   Revisiting Some Classical Wilkinson-Style Error Bounds

3.1   Summation

Given $x_1, \ldots, x_n \in \mathbb{F}$, we consider first the evaluation of the sum $\sum_{i=1}^{n} x_i$ by means of $n - 1$ floating-point additions, in any order. Following Wilkinson [43], we may apply the standard model (1) repeatedly in order to obtain the backward error result shown in Sect. 1, from which a forward error bound for the computed value $\hat{r}$ then follows directly:

$$\left| \hat{r} - \sum_{i=1}^{n} x_i \right| \leqslant \alpha \sum_{i=1}^{n} |x_i|, \qquad \alpha = (1 + u)^{n-1} - 1. \tag{13}$$

Such a bound is easy to derive, valid for any order, and a priori essentially best possible since there exist special values of the x_i for which the ratio error/(error bound) tends to 1 as $u \to 0$. The expression giving α, however, is somehow unwieldy and it is now common practice to have it replaced by the concise yet

rigorous upper bound γ_{n-1}, using Higham's γ_k notation "$\gamma_k = ku/(1 - ku)$ if $ku < 1$" [14, p. 63]. Both bounds have the form $(n-1)u + O(u^2)$ and the second one further assumes implicitly that the dimension n satisfies $(n-1)u < 1$.

Recently, it was shown by Rump [36] that for recursive summation one can in fact always replace α in (13) by the simpler and sharper expression

$$\alpha = (n-1)u.$$

In other words, the terms of order $O(u^2)$ can be removed, and this without any restriction on n. The proof given in [36, p. 206] aims to bound the forward error $|\hat{r} - \sum_{i=1}^{n} x_i|$ directly, focusing on the last addition and proceeding by induction on n; in particular, one key ingredient is the refined model (5) of floating-point addition, which is used here to handle the case $|x_n| \leqslant u \sum_{i=1}^{n-1} |x_i|$. As noted in [24, Sect. 3], this proof technique is in fact not restricted to recursive summation, so the constant $(n-1)u$ eventually holds for any summation order.

3.2   Other Examples of $O(u^2)$-Free Error Bounds

Similar improvements have been obtained for the error bounds of several other computational problems, which we summarize in Table 1. The algorithms for which these new bounds hold are the classical ones (described for example in [14]) and the role played by α depends on the problem as follows: for dot products, α should be such that $|\hat{r} - x^T y| \leqslant \alpha |x|^T |y|$ with $x, y \in \mathbb{F}^n$ and $\hat{r}$ denoting the computed value; for matrix multiplication, $|\hat{C} - AB| \leqslant \alpha |A||B|$ with $A \in \mathbb{F}^{*\times n}$ and $B \in \mathbb{F}^{n\times *}$; for Euclidean norms (in dimension n), powers, and products, $|\hat{r} - r| \leqslant \alpha |r|$; for triangular system solving and LU and Cholesky matrix factorizations, we consider the usual backward error bounds $|\Delta T| \leqslant \alpha |T|$ for $(T + \Delta T)\hat{x} = b$, $|\Delta A| \leqslant \alpha |\hat{L}||\hat{U}|$ for $\hat{L}\hat{U} = A + \Delta A$, and $|\Delta A| \leqslant \alpha |\hat{R}^T||\hat{R}|$ for $\hat{R}^T \hat{R} = A + \Delta A$. (Here the matrices T, $\hat{U}$, $\hat{R}$ have dimensions $n \times n$, and $\hat{L}$ has dimensions $m \times n$ with $m \geqslant n$.) Finally, for the evaluation of $a(x) = \sum_{i=0}^{n} a_i x^i$ with Horner's rule, α is such that $|\hat{r} - a(x)| \leqslant \alpha \sum_{i=0}^{n} |a_i x^i|$.

The new values of α shown in Table 1 are free of any $O(u^2)$ term and thus *simpler and sharper* than the classical ones. In the last three cases, the price to be paid for those refined constants is some mild restriction on n; we refer to [38] for a precise condition and an example showing that it is indeed necessary.

4   Provably Accurate Numerical Kernels

4.1   Computation of $ab + cd$

As a first example of such kernels, let us consider the evaluation of $ab + cd$ for $a, b, c, d \in \mathbb{F}$. This operation occurs frequently in practice and is especially useful for complex arithmetic, discriminants, and robust orientation predicates. Since it is not part of the set of core IEEE 754-2008 functions for which correct rounding is required or recommended (and despite the existence of hardware designs as the one by Brunie [4, Sect. 3.3.2]), this operation will in general be implemented

Table 1. Some classical Wilkinson-style constants made simpler and sharper. Unless otherwise stated these results hold for any ordering, and $(\star)$ means "if $n \lesssim u^{-1/2}$".

Problem	Classical α	New α		Reference(s)
summation	$(n-1)u + O(u^2)$	$(n-1)u$		[24, 36]
dot prod., mat. mul.	$nu + O(u^2)$	nu		[24]
Euclidean norm	$(\frac{n}{2}+1)u + O(u^2)$	$(\frac{n}{2}+1)u$		[25]
$Tx = b, \quad A = LU$	$nu + O(u^2)$	nu		[39]
$A = R^T R$	$(n+1)u + O(u^2)$	$(n+1)u$		[39]
x^n (recursive, $\beta = 2$)	$(n-1)u + O(u^2)$	$(n-1)u$	$(\star)$	[12]
product $x_1 x_2 \cdots x_n$	$(n-1)u + O(u^2)$	$(n-1)u$	$(\star)$	[38]
poly. eval. (Horner)	$2nu + O(u^2)$	$2nu$	$(\star)$	[38]

in software using basic floating-point arithmetic. When doing so, however, some care is needed and a classical scheme like $\mathrm{RN}(\mathrm{RN}(ab) + \mathrm{RN}(cd))$ or, if an FMA is available, $\mathrm{RN}(ab + \mathrm{RN}(cd))$ can produce a highly inaccurate answer.

To avoid this, the following sequence of four operations was suggested by Kahan (see [14, p. 60]):

$$\hat{w} := \mathrm{RN}(cd); \quad \hat{f} := \mathrm{RN}(ab + \hat{w}); \quad e := \mathrm{RN}(cd - \hat{w}); \quad \hat{r} := \mathrm{RN}(\hat{f} + e).$$

Here the FMA operation is used to produce $\hat{f}$ and also to implement an EFT for the product cd, as in (12), thus giving $e = cd - \hat{w}$ exactly. By applying to $\hat{w}$, $\hat{f}$, and $\hat{r}$ the refined standard model given by (11) it is then easy to prove that

$$\frac{|\hat{r} - r|}{|r|} \leqslant 2u(1 + \psi), \qquad r = ab + cd, \qquad \psi = \frac{u|cd|}{2|r|}. \tag{14}$$

This kind of analysis (already done by Higham in the 1996 edition of [14]) shows that Kahan's algorithm computes $ab + cd$ with high relative accuracy as long as $\psi \not\gg 1$. The latter condition, however, does not always hold, as there exist inputs for which ψ is of the order of u^{-1} and the relative error bound $2u(1 + \psi)$ is larger than 1.

This classical analysis was refined in [21], where we show that Kahan's algorithm above is in fact *always* highly accurate: first, a careful analysis of the absolute errors $\epsilon_1 = \hat{f} - (ab + \hat{w})$ and $\epsilon_2 = \hat{r} - (\hat{f} + e)$ using the ufp (or ulp) function gives $|\epsilon_1|, |\epsilon_2| \leqslant \beta u\, \mathrm{ufp}(r)$, so that $|\hat{r} - r| = |\epsilon_1 + \epsilon_2| \leqslant 2\beta u|r|$; then, by studying ϵ_1 and ϵ_2 simultaneously via a case analysis comparing $|\epsilon_2|$ to $u\, \mathrm{ufp}(r)$, we deduce that the constant $2\beta u$ can be replaced by $2u$ (that is, the term ψ can in fact be removed from the bound in (14)); third, we show that this bound is asymptotically optimal (as $u \to 0$) by defining

$$a = b = \beta^{p-1} + 1, \qquad c = \beta^{p-1} + \tfrac{\beta}{2}\beta^{p-2}, \qquad d = 2\beta^{p-1} + \tfrac{\beta}{2}\beta^{p-2},$$

and checking (by hand or, since recently, using a dedicated Maple library [22]) that the error committed for such inputs has the form $2u - 4u^2 + O(u^3)$.

A similar scheme was proposed by Cornea, Harrison, and Tang [7, p. 273], which ensures further that the value returned for $ab+cd$ is the same as for $cd+ab$. (Such a feature may be desirable when, say, implementing complex arithmetic.) We refer to [19, 30] for sharp error analyzes combining ufp-based arguments, the refined bound $u/(1+u)$, and Sterbenz' theorem.

4.2 Complex Multiplication

Another important numerical kernel is the evaluation of the real and imaginary parts $R = ac - bd$ and $I = ad + bc$ of the complex product $z = (a + ib)(c + id)$. Consider first the conventional way, which produces $\hat{R} = \mathrm{RN}(\mathrm{RN}(ac) - \mathrm{RN}(bd))$ and $\hat{I} = \mathrm{RN}(\mathrm{RN}(ad) + \mathrm{RN}(bc))$. Although $\hat{R}$ or $\hat{I}$ can be completely inaccurate, it is known that high relative accuracy holds in the *normwise* sense: Brent, Percival, and Zimmermann [2] showed that $\hat{z} = \hat{R} + i\hat{I}$ satisfies

$$\frac{|\hat{z} - z|}{|z|} \leqslant \sqrt{5}u$$

and that this bound is asymptotically optimal (at least in base 2); in particular, the constant $\sqrt{5} = 2.23\ldots$ improves upon classical and earlier ones like $\sqrt{8} = 2.82\ldots$ by Wilkinson [44, p. 447] and $1 + \sqrt{2} = 2.41\ldots$ by Champagne [5].

Assume now that an FMA is available. In this case, $\hat{R}$ can be obtained as $\mathrm{RN}(ac - \mathrm{RN}(bd))$ or $\mathrm{RN}(\mathrm{RN}(ac) - bd)$, and similarly for $\hat{I}$, so that z can be evaluated using four different schemes. We showed in [20] that for each of these schemes the bound $\sqrt{5}u$ mentioned above can be reduced further to $2u$ and that this new bound is asymptotically optimal. We also proved that this normwise bound $2u$ remains sharp even if both $\hat{R}$ and $\hat{I}$ are computed with high relative accuracy as in Sect. 4.1.

The bound $\sqrt{5}u$ was obtained in [2] via a careful ulp-based case analysis. For the bound $2u$ we have proceeded similarly in [20, Sect. 3] but, as we observe in [25], in this case a much shorter proof follows from using just the refined standard model given by (11).

A direct application of these error bounds is to complex division: as noted by Baudin in [1], if αu bounds the normwise relative error of multiplication, then the bound $(\alpha+3)u+O(u^2)$ holds for division—assuming the classical formula $x/y = (x\overline{y})/(y\overline{y})$—and thus we can take $\alpha + 3 = 5$ or $5.23\ldots$ depending on whether the FMA operation is available or not. However, despite this and some recent progress made in the case of complex inversion [23], the best possible constants for complex division (with or without an FMA) remain to be determined.

References

1. Baudin, M.: Error bounds of complex arithmetic, June 2011. http://forge.scilab. org/upload/compdiv/files/complexerrorbounds_v0.2.pdf
2. Brent, R.P., Percival, C., Zimmermann, P.: Error bounds on complex floating-point multiplication. Math. Comput. **76**, 1469–1481 (2007)
3. Brent, R.P., Zimmerman, P.: Modern Computer Arithmetic. Cambridge University Press, Cambridge (2010)
4. Brunie, N.: Contributions to Computer Arithmetic and Applications to Embedded Systems. Ph.D. thesis, École Normale Supérieure de Lyon, Lyon, France, May 2014. https://tel.archives-ouvertes.fr/tel-01078204
5. Champagne, W.P.: On finding roots of polynomials by hook or by crook. Master's thesis, University of Texas, Austin, Texas (1964)
6. Corless, R.M., Fillion, N.: A Graduate Introduction to Numerical Methods, From the Viewpoint of Backward Error Analysis. Springer, New York (2013)
7. Cornea, M., Harrison, J., Tang, P.T.P.: Scientific Computing on Itanium$^{\circledR}$-based Systems. Intel Press, Hillsboro (2002)
8. Dekker, T.J.: A floating-point technique for extending the available precision. Numer. Math. **18**, 224–242 (1971)
9. Demmel, J.W.: Applied Numerical Linear Algebra. SIAM, Philadelphia (1997)
10. Goldberg, D.: What every computer scientist should know about floating-point arithmetic. ACM Comput. Surv. **23**(1), 5–48 (1991)
11. Golub, G.H., Van Loan, C.F.: Matrix Computations, 4th edn. Johns Hopkins University Press, Baltimore (2013)
12. Graillat, S., Lefèvre, V., Muller, J.M.: On the maximum relative error when computing integer powers by iterated multiplications in floating-point arithmetic. Numer. Algorithms **70**, 653–667 (2015). http://link.springer.com/article/10.1007/ s11075-015-9967-8
13. Hauser, J.R.: Handling floating-point exceptions in numeric programs. ACM Trans. Program. Lang. Syst. **18**(2), 139–174 (1996)
14. Higham, N.J.: Accuracy and Stability of Numerical Algorithms, 2nd edn. SIAM, Philadelphia (2002)
15. Higham, N.J.: Floating-point arithmetic. In: Higham, N.J., Dennis, M.R., Glendinning, P., Martin, P.A., Santosa, F., Tanner, J. (eds.) The Princeton Companion to Applied Mathematics, pp. 96–97. Princeton University Press, Princeton (2015)
16. Holm, J.E.: Floating-Point Arithmetic and Program Correctness Proofs. Ph.D. thesis, Cornell University, Ithaca, NY, USA, August 1980
17. IEEE Computer Society: IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754–1985. IEEE Computer Society, New York (1985)
18. IEEE Computer Society: IEEE Standard for Floating-Point Arithmetic, IEEE Standard 754–2008. IEEE Computer Society, New York (2008)
19. Jeannerod, C.P.: A radix-independent error analysis of the Cornea-Harrison-Tang method, to appear in ACM Trans. Math. Softw. https://hal.inria.fr/hal-01050021
20. Jeannerod, C.P., Kornerup, P., Louvet, N., Muller, J.M.: Error bounds on complex floating-point multiplication with an FMA, to appear in Math. Comput. https:// hal.inria.fr/hal-00867040v4
21. Jeannerod, C.P., Louvet, N., Muller, J.M.: Further analysis of Kahan's algorithm for the accurate computation of 2×2 determinants. Math. Comput. **82**(284), 2245–2264 (2013)

22. Jeannerod, C.P., Louvet, N., Muller, J.M., Plet, A.: A library for symbolic floating-point arithmetic (2015). https://hal.inria.fr/hal-01232159

23. Jeannerod, C.-P., Louvet, N., Muller, J.-M., Plet, A.: Sharp error bounds for complex floating-point inversion. Numer. Algorithms 1–26 (2016). https://hal-ens-lyon.archives-ouvertes.fr/ensl-01195625

24. Jeannerod, C.P., Rump, S.M.: Improved error bounds for inner products in floating-point arithmetic. SIAM J. Matrix Anal. Appl. **34**(2), 338–344 (2013)

25. Jeannerod, C.P., Rump, S.M.: On relative errors of floating-point operations: optimal bounds and applications (2014). https://hal.inria.fr/hal-00934443

26. Kahan, W.: Further remarks on reducing truncation errors. Commun. ACM **8**(1), 40 (1965)

27. Knuth, D.E.: The Art of Computer Programming. Seminumerical Algorithms, vol. 2, 3rd edn. Addison-Wesley, Reading (1998)

28. Møller, O.: Quasi double-precision in floating point addition. BIT **5**, 37–50 (1965)

29. Monniaux, D.: The pitfalls of verifying floating-point computations. ACM Trans. Program. Lang. Syst. **30**(3), 12:1–12:41 (2008)

30. Muller, J.M.: On the error of computing $ab+cd$ using Cornea, Harrison and Tang's method. ACM Trans. Math. Softw. **41**(2), 7:1–7:8 (2015)

31. Muller, J.M., Brisebarre, N., de Dinechin, F., Jeannerod, C.P., Lefèvre, V., Melquiond, G., Revol, N., Stehlé, D., Torres, S.: Handbook of Floating-Point Arithmetic. Birkhäuser, Boston (2010)

32. Ogita, T., Rump, S.M., Oishi, S.: Accurate sum and dot product. SIAM J. Sci. Comput. **26**(6), 1955–1988 (2005)

33. Overton, M.L.: Numerical Computing with IEEE Floating Point Arithmetic: Including One Theorem, One Rule of Thumb, and One Hundred and One Exercises. Society for Industrial and Applied Mathematics, Philadelphia (2001)

34. Priest, D.M.: On Properties of Floating Point Arithmetics: Numerical Stability and the Cost of Accurate Computations. Ph.D. thesis, Mathematics Department, University of California, Berkeley, CA, USA, November 1992

35. Rump, S.M.: Ultimately fast accurate summation. SIAM J. Sci. Comput. **31**(5), 3466–3502 (2009)

36. Rump, S.M.: Error estimation of floating-point summation and dot product. BIT **52**(1), 201–220 (2012)

37. Rump, S.M., Ogita, T., Oishi, S.: Accurate floating-point summation part I: faithful rounding. SIAM J. Sci. Comput. **31**(1), 189–224 (2008)

38. Rump, S.M., Bünger, F., Jeannerod, C.P.: Improved error bounds for floating-point products and Horner's scheme. BIT (2015). http://link.springer.com/article/10.1007/s10543-015-0555-z

39. Rump, S.M., Jeannerod, C.P.: Improved backward error bounds for LU and Cholesky factorizations. SIAM J. Matrix Anal. Appl. **35**(2), 684–698 (2014)

40. Sterbenz, P.H.: Floating-Point Computation. Prentice-Hall, Englewood Cliffs (1974)

41. Trefethen, L.N.: Computing numerically with functions instead of numbers. Math. Comput. Sci. **1**(1), 9–19 (2007)

42. Trefethen, L.N., Bau III, D.: Numerical Linear Algebra. SIAM, Philadelphia (1997)

43. Wilkinson, J.H.: Error analysis of floating-point computation. Numer. Math. **2**, 319–340 (1960)

44. Wilkinson, J.H.: The Algebraic Eigenvalue Problem. Oxford University Press, Oxford (1965)

Mathematical Aspects of Computer and Information
Sciences
6th International Conference, MACIS 2015, Berlin,
Germany, November 11-13, 2015, Revised Selected
Papers
Kotsireas, I.S.; Rump, S.M.; Yap, C.K. (Eds.)