Error bounds for computer arithmetics

This research was partially supported by CREST, Japan Science and Technology Agency.

Siegfried M. Rump
Institute for Reliable Computing,
Hamburg University of Technology,
Am Schwarzenberg-Campus 3, 21071 Hamburg, Germany,
and Visiting Professor at Waseda University,
Faculty of Science and Engineering,
3–4–1 Okubo, Shinjuku-ku, Tokyo 169–8555, Japan.
Email: rump@tuhh.de

Abstract—**This note summarizes recent progress in error bounds for compound operations performed in some computer arithmetic. Given a general set of real numbers together with some operations satisfying the first standard model, we identify three types A, B, and C of weak sufficient assumptions implying new results and sharper error estimates. Those include linearized error estimates in the number of operations, faithfully rounded and reproducible results. All types of assumptions are satisfied for an IEEE-754 p-digit base-β floating-point arithmetic.**

I. Introduction

In this note we summarize some recent results on IEEE-754 floating-point and on a more general computer arithmetic. Several findings are presented unifying and generalizing previous ones. At few places short proofs are given rather than lengthy explanations if we think it is the easier way to understand the matter.

An introduction and basic properties of floating-point and of more general computer arithmetics can be found in Higham's ASNA [13], see also [9], and in particular in the excellent books [29] by Muller et al. and [4] by Brent and Zimmermann.

We start with the most popular model of computer arithmetic, namely [14] IEEE-754 p-digit floating-point arithmetic to some base β. For this model we show that traditional estimates using the relative rounding error unit $\mathrm{u} = \frac{1}{2}\beta^{1-p}$ can be replaced by optimal bounds for individual operations. Moreover, it was a surprise that the traditional bound $\gamma_k = k\mathrm{u}/(1-k\mathrm{u})$ for compound operations, for example for summation, dot products and also some matrix factorizations, can be linearized to $k\mathrm{u}$.

Next, we consider an arbitrary set $\mathbb{A}$ of real numbers and a computer arithmetic on $\mathbb{A}$ with the sole requirement to satisfy the first standard model [13], i.e., the relative error of the computed result to the exact result is bounded by some constant. Then we ask what additional assumptions are necessary to prove error estimates such as the linearized bounds mentioned above. This approach is in some way opposite to fixing the set $\mathbb{A}$ to p-digit base-β numbers and precisely defining the operations as in IEEE-754 arithmetic. We identify three mutually different additional assumptions.

For a computer arithmetic and $a, b \in \mathbb{A}$ denote by $a \boxplus b$ the computed result of $a + b$. The first Assumption A is that the absolute error of an addition or subtraction is bounded by the minimum absolute value of the operands, that is

$$\text{Assumption A:} \qquad |a \boxplus b - (a+b)| \leqslant \min(|a|, |b|).$$

That suffices to prove the linearized error bounds mentioned above. Assumption A is satisfied for any IEEE-754 p-digit base-β floating-point arithmetic with some nearest rounding. On the one hand, Assumption A is very weak [the error of $2 \boxplus 3$ to 5 is bounded by 2], on the other hand Assumption A excludes directed rounding [adding an arbitrarily small number to 1 results in one of the neighbors of 1].

In IEEE-754 arithmetic the relative error of an operation $a \boxed{\circ} b$ is not only bounded by $\mathrm{u}|a \circ b|$, but in fact by $\mathrm{u} \cdot \mathrm{ufp}(a \circ b)$, where $\mathrm{ufp}(x)$ denotes the largest power of β less than or equal to $|x|$. That is the kernel of the alternative second Assumption B, namely that errors are bounded relative to specific numbers "near" the actual result rather than to the real result itself. That suffices to prove the linear error estimates. For IEEE-754 p-digit base-β floating-point arithmetic such numbers are powers of β, and that second Assumption B is satisfied for any rounding, including the directed and faithful ones, the latter meaning that there is no other floating-point number between the real result and the computed result.

Another interpretation of the fact that in IEEE-754 errors are bounded by $\mathrm{u} \cdot \mathrm{ufp}(a \circ b)$ is that those numbers are a constant times a power of β. This is the kernel of the third Assumption C, namely that errors grow by $b \cdot \beta^k$ for some constants b, β, k. With Assumption C it follows that errors grow linearly with the height of a summation tree. For IEEE-754 p-digit base-β floating-point arithmetic the third Assumption C is satisfied for any rounding, including directed or faithful ones.

The conclusions by Assumption A are true without restriction on the number of summands. The second and third Assumptions B and C are targeted to prove results for a general rounding, including directed, faithful but also nearest. Such results require a mandatory but weak restriction on the size of the problem. Therefore Assumptions B and C are more general than Assumption A, but imply such a restriction.

The first standard model together with Assumptions A and B leads to the optimal error bound for summation, namely that $k\mathrm{u}$ can be replaced by $k\mathrm{u}/(1+k\mathrm{u})$, see Table I. A mandatory restriction on k of size u^{-1} applies.

Depending on the assumption, the error bounds for summation are as follows. Let a set $\mathbb{A} \subseteq \mathbb{R}$ with a computer arithmetic according to the first standard model with relative rounding error unit u be given. For $p_1, \ldots, p_n \in \mathbb{A}$ denote by $\hat{s}$ the computed sum in any order. Depending on the additional assumption, the error of summation satisfies $|\hat{s} - \sum_{i=1}^{n} p_i| \leqslant \Phi \sum_{i=1}^{n} |p_i|$ with constants Φ according to Table I. The bounds depend on the number of summands n except the third one depending on the height h of the summation tree.

TABLE I: Error bounds for summation.

Assumption	Φ	rounding	condition
A	$(n-1)\mathrm{u}$	nearest	none
B	$(n-1)\mathrm{u}$	any	$n \leqslant 1 + \frac{\beta-1}{2}\mathrm{u}^{-1}$
C	$h\mathrm{u}$	any	$h \leqslant \mathrm{u}^{-1/2} - 1$
A and B	$\frac{(n-1)\mathrm{u}}{1+(n-1)\mathrm{u}}$	nearest	$n \leqslant 1 + \frac{\beta-1}{2}\mathrm{u}^{-1}$

The motivation of the improved error estimates is not only a matter of beauty, but it often suffices to show similar linearized estimates for other types of compound operations such as dot products, blocked summation or sums of products.

Next we reduce the arithmetic to the first standard model, not requiring any of the Assumptions A, B or C. Based on that we introduce a simplified pair arithmetic producing a faithfully rounded result under precisely specified conditions. In the world of IEEE-754 arithmetic this widens the applicability because existing pair arithmetics rely on error-free transformations. Those, however, do not exist in case of directed rounding because the error of an approximate operation needs not to be representable.

Finally we close the circle and return to IEEE-754 binary arithmetic with rounding to nearest. For that we discuss how to obtain efficiently a reproducible result for summation.

II. Optimal error bounds for the two standard models

We start with the most popular model of computer arithmetic, namely IEEE-754 p-digit base-β floating-point arithmetic. For simplicity, we assume the set $\mathbb{F}$ of floating-point numbers to have no restriction on the exponent range:

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

Let $\mathrm{fl}\colon \mathbb{R} \to \mathbb{F}$ denote a round-to-nearest function, that is

$$|t - \mathrm{fl}(t)| = \min_{f \in \mathbb{F}} |t - f|, \qquad t \in \mathbb{R}. \tag{2}$$

To carry out rounding error analysis of algorithms, frequently the first or second standard model of computer arithmetic is used. That means that, according to Table II, the relative error of a floating-point operation $\circ \in \{+, -, \times, /\}$ shall be bounded with respect to the true result for the first, and with respect to the computed result for the second standard model. The relative error of the first and second standard model for rounding $t \in \mathbb{R}$ is [13]

$$E_1(t) = \frac{|t - \mathrm{fl}(t)|}{|t|} \qquad \text{and} \qquad E_2(t) = \frac{|t - \mathrm{fl}(t)|}{|\mathrm{fl}(t)|},$$

respectively, with the convention $0/0 = 0$. It is well known [20] that $E_1(t) \leqslant \mathrm{v} := \frac{\mathrm{u}}{1+\mathrm{u}}$ and $E_2(t) \leqslant \mathrm{u}$ for the relative rounding error unit $\mathrm{u} := \frac{1}{2}\beta^{1-p}$.

TABLE II: First and second standard model for $x, y \in \mathbb{F}$.

standard model	property
I	$\mathrm{fl}(x \circ y) = (x \circ y)(1+\delta), \quad |\delta| \leqslant \mathrm{u}$
II	$\mathrm{fl}(x \circ y) = \frac{x \circ y}{1+\delta}, \quad |\delta| \leqslant \mathrm{u}$

For arithmetic operations $x \boxdot y := \mathrm{fl}(x \circ y)$, however, the upper bound for E_1 or E_2 is not always attained. The maximum value for E_1 is achieved if and only if there exist $x, y \in \mathbb{F}$ with $x \circ y = 1 + \mathrm{u}$, the number half-way between 1 and its successor $1 + 2\mathrm{u}$. The same is true for E_2 if ties are rounded to even.

The worst case bounds for the individual operations were proved by Jeannerod et al. in [16] as given in Table III. The bounds are optimal, possibly under some mild (necessary and sufficient) conditions on β and p outlined in [16]. Specifically, for addition, subtraction, and multiplication the condition for optimality is that β is even, and in the case of multiplication in base 2 it requires that $2^p + 1$ is not a Fermat prime. In most practical situations such conditions are satisfied.

TABLE III: Optimal relative error bounds for various inputs t and $x, y \in \mathbb{F}$ for IEEE-754 p-digit base-β floating-point arithmetic.

t	bound on $E_1(t)$	bound on $E_2(t)$
real number	$\frac{\mathrm{u}}{1+\mathrm{u}}$	u
$x \pm y$	$\frac{\mathrm{u}}{1+\mathrm{u}}$	u
xy	$\frac{\mathrm{u}}{1+\mathrm{u}}$	u
x/y	$\begin{cases} \mathrm{u} - 2\mathrm{u}^2 & \text{if } \beta = 2, \\ \frac{\mathrm{u}}{1+\mathrm{u}} & \text{if } \beta > 2 \end{cases}$	$\begin{cases} \frac{\mathrm{u}-2\mathrm{u}^2}{1+\mathrm{u}-2\mathrm{u}^2} & \text{if } \beta = 2, \\ \mathrm{u} & \text{if } \beta > 2 \end{cases}$
$\sqrt{x}$	$1 - \frac{1}{\sqrt{1+2\mathrm{u}}}$	$\sqrt{1+2\mathrm{u}} - 1$

A rounding function $\mathrm{fl}\colon \mathbb{R} \to \mathbb{F}$ is defined by its "switching points" (called rounding boundary in [4]). In IEEE-754 these are the midpoints, i.e., the arithmetic mean of adjacent floating-point numbers, thus minimizing the maximum relative error $E_1(t)$ of the first standard model.

We may ask for switching points minimizing the maximum relative error for a nearest rounding of the second, or of both standard models. These have been identified in [37] as by Table IV. This is not only true for the grid of p-digit base-β floating-point numbers, but for general sets of real numbers.

TABLE IV: Optimal switching points within adjacent elements of $\mathbb{F}$.

Minimizing $E_1(t)$	Minimizing $E_2(t)$	Minimizing $\max(E_1(t), E_2(t))$
Arithmetic mean	Harmonic mean	Geometric mean

III. Linearly bounded error estimates for compound operations

In this section we still assume an IEEE-754 p-digit base-β floating-point arithmetic with some nearest rounding. Here "some" means that ties may be rounded in any way.

The most simple and important compound operation is the sum of floating-point numbers. Let n numbers $p_1, \ldots, p_n \in \mathbb{F}$ be given, set $\hat{s}_1 := p_1$ and define recursively

$$\hat{s}_i := \hat{s}_{i-1} \boxplus p_i \qquad \text{for} \quad i \in \{2, \ldots, n\}. \tag{3}$$

Since $\hat{s}_{i-1} \boxplus p_i = (\hat{s}_{i-1} + p_i)(1 + \delta_i)$ for some $|\delta_i| \leqslant \mathrm{u}$, a straightforward and standard computation yields [13]

$$|\hat{s}_n - \sum_{i=1}^{n} p_i| \leqslant \left((1+\mathrm{u})^{n-1} - 1\right) \sum_{i=1}^{n} |p_i|. \tag{4}$$

This is the classical Wilkinson-type bound [41]. To cover higher order terms, the unwieldy factor on the right-hand side is often replaced by $\gamma_{n-1} = \frac{(n-1)\mathrm{u}}{1-(n-1)\mathrm{u}}$, provided that $(n-1)\mathrm{u} < 1$ (see [13]).

It was a surprise when it was proved in [34] that, at least for recursive summation (3), the factor γ_{n-1} can be replaced by $(n-1)\mathrm{u}$ without restriction on n:

$$|\ \hat{s}_n - \sum_{i=1}^{n} p_i\ | \leqslant (n-1)\mathrm{u} \sum_{i=1}^{n} |p_i|\ . \tag{5}$$

That theoretical estimate was supplemented in [34] by the computable estimate

$$|\ \hat{s}_n - \sum_{i=0}^{n} p_i\ | \leqslant (n-1)\mathrm{u} \cdot \mathrm{ufp}(\hat{S}_n),$$

where $\hat{S}_n$ is obtained by (3) when replacing p_i by $|p_i|$. The *unit in the first place* $\mathrm{ufp}(x)$ of $x \in \mathbb{R}$ denotes the value of the leading digit in the β-adic representation of x (with $\mathrm{ufp}(0) := 0$). It was introduced in [38] and proved to be useful to transform complicated rounding error analyses into inequalities. At least for $\beta = 2$ it can be computed, without branch, using three floating-point operations and one absolute value, see [34, Algorithm 3.5] and [17].

Open problem 1. Design an algorithm of similar complexity and without branch to compute $\mathrm{ufp}(x)$ for $\beta > 2$.

After that first linear error estimate, the race began. For the remainder of this section, in order to show the historical progress, we still restrict our attention to a computer arithmetic in base $\beta \geqslant 2$ with $p \geqslant 2$ mantissa digits following the IEEE-754 standard, so that $\mathrm{u} = \frac{1}{2}\beta^{1-p}$.

The next target was dot products. Rather than treating sums of products of floating-point numbers, the error of a sum of *real* numbers was estimated. To our knowledge that was the first time to take that more general perspective. More precisely, consider a real vector $x_1, \ldots, x_n \in \mathbb{R}$ and suppose the sum of $\mathrm{fl}(x_i)$ is evaluated in floating-point arithmetic with result $\hat{r}$. Then it was shown by Jeannerod in [15] that, no matter what the order of evaluation of the floating-point sum,

$$|\hat{r} - \sum_{i=1}^{n} x_i| \leqslant n\mathrm{u} \sum_{i=1}^{n} |x_i|. \tag{6}$$

The result is true without any restriction of n. For floating-point vectors $a, b \in \mathbb{F}^n$ it follows as a corollary that the result $\hat{r}$ of the floating-point dot product, no matter what the order of evaluation and barring underflow, satisfies

$$|\hat{r} - a^T b| \leqslant n\mathrm{u}|a^T||b|. \tag{7}$$

As a consequence, the error of the floating-point product of two matrices A, B with inner dimension k is bounded by $k\mathrm{u}|A||B|$. In [15] it was also shown that the bound for summation (5) is true in IEEE-754 p-digit base-β floating-point arithmetic for any order of evaluation; in the next section we will see that (5), (6) and (7) are true for any computer arithmetic satisfying the first error model and $|a \boxplus b - (a+b)| \leqslant \min(|a|, |b|)$. The latter is Assumption A.

Given a vector $p \in \mathbb{F}^n$, let $\hat{r}$ be the value of the Euclidean norm $\|p\|_2$ calculated in floating-point arithmetic in any order of evaluation. The standard error estimate

$$\left|\, \hat{r} - \|p\|_2 \,\right| \leqslant \left((1+\mathrm{u})^{n/2+1} - 1\right) \|p\|_2$$

was, without restriction on n, improved in [16] to

$$\left|\, \hat{r} - \|p\|_2 \,\right| \leqslant (n/2+1)\mathrm{u}\|p\|_2.$$

Next consider the product of floating-point numbers. First, Graillat et al. proved in [12] that for $a \in \mathbb{F}$, $\beta = 2$, the result $\hat{r}$ of the power a^{k+1} computed by successive multiplications and barring over- and underflow satisfies

$$|\hat{r} - a^{k+1}| \leqslant k\mathrm{u}|a^{k+1}| \quad \text{if } k+1 \leqslant \sqrt{2^{1/3} - 1} \cdot \mathrm{u}^{-1/2}. \tag{8}$$

More generally, in [35] the product of real numbers was treated. For $x_0, x_1, \ldots, x_k \in \mathbb{R}$ with ℓ of them in $\mathbb{F}$, denote by $\hat{r}$ the floating-point product of all the $\mathrm{fl}(x_i)$ in any order of evaluation, and set

$$K := 2k + 1 - \ell \quad \text{and} \quad \omega := \begin{cases} 1 & \text{if } \beta \text{ is odd,} \\ 2 & \text{if } \beta \text{ is even.} \end{cases} \tag{9}$$

Then, in the absence of underflow and overflow,

$$|\hat{r} - \prod_{i=0}^{k} x_i| \leqslant K\mathrm{u}\,|\prod_{i=0}^{k} x_i| \quad \text{if } K < \sqrt{\frac{\omega}{\beta}}\ \mathrm{u}^{-1/2}. \tag{10}$$

Note that the index i starts from 0. In particular, if $\beta = 2$ and all the x_i are in $\mathbb{F}$, then $(K, \omega) = (k, 2)$ and (10) becomes

$$|\hat{r} - \prod_{i=0}^{k} x_i| \leqslant k\mathrm{u}\,|\prod_{i=0}^{k} x_i| \quad \text{if } k < \mathrm{u}^{-1/2}. \tag{11}$$

For $\beta = 2$ and $p \geqslant 4$, the constraint in (11) cannot be replaced by $k < 12\mathrm{u}^{-1/2}$.

Open problem 2. Assume IEEE-754 p-digit base-β arithmetic. Let T be a binary tree with $k+1$ leaves, where each inner node represents a division. Associate to each leaf a floating point number, denote by r the value of the root for real division $/$, and by $\hat{r}$ for floating-point division $\boxed{/}$. Is

$$|\hat{r} - r| \leqslant k\mathrm{u}|r| \qquad \text{if } k < \mathrm{u}^{-1/2}$$

true? Is it also true for mixed multiplications and divisions? If yes and assuming the first standard model, what are the necessary assumptions on the computer arithmetic?

As another consequence, the classical factor γ_{2n} for Horner's scheme was improved as well in [35]. Let $x, a_0, a_1, \ldots, a_n \in \mathbb{F}$ be given and let $\hat{r}$ be the approximation to $\sum_{i=0}^{n} a_i x^i$ produced by Horner's scheme. Then, using ω defined in (9) and in the absence of underflow and overflow,

$$\left| \hat{r} - \sum_{i=0}^{n} a_i x^i \right| \leqslant 2n\mathtt{u} \sum_{i=0}^{n} |a_i x^i| \text{ if } n < \frac{1}{2}\left(\sqrt{\frac{\omega}{\beta}}\ \mathtt{u}^{-1/2} - 1\right). \tag{12}$$

Finally, it was shown in [36] that the concept of linearizing bounds by replacing γ_k by $k\mathtt{u}$ is also true for some standard numerical linear algebra algorithms. If for some $A \in \mathbb{F}^{m \times n}$ with $m \geqslant n$ Gaussian elimination runs to completion, then the computed factors $\hat{L}$ and $\hat{U}$ satisfy (comparison and absolute value to be understood entrywise)

$$\hat{L}\hat{U} = A + \Delta A, \qquad |\Delta A| \leqslant n\mathtt{u}|\hat{L}||\hat{U}|. \tag{13a}$$

If for symmetric $A \in \mathbb{F}^{n \times n}$ the Cholesky decomposition runs to completion, then the computed factor $\hat{R}$ satisfies

$$\hat{R}^T\hat{R} = A + \Delta A, \qquad |\Delta A| \leqslant (n+1)\mathtt{u}|\hat{R}^T||\hat{R}|. \tag{13b}$$

If $Tx = b$ is solved by substitution for $b \in \mathbb{F}^n$ and nonsingular triangular $T \in \mathbb{F}^{n \times n}$, then the computed solution $\hat{x}$ satisfies

$$(T + \Delta T)\hat{x} = b, \qquad |\Delta T| \leqslant n\mathtt{u}|T|. \tag{13c}$$

Each of these bounds improves upon the corresponding classical ones, that is,

$$\gamma_n|\hat{L}||\hat{U}|, \quad \gamma_{n+1}|\hat{R}^T||\hat{R}|, \quad \gamma_n|T|.$$

In contrast to the classical ones, the new bounds are valid without restriction on n.

IV. General computer arithmetic

Up to now we *actively* assumed to use an IEEE-754 p-digit base-β floating-point arithmetic. Then, for example, linearized error estimates follow. Next, for a computer arithmetic satisfying the first standard model, we will *passively* identify sufficient additional assumptions to prove certain results, for example linearized error estimates. In particular we move away from a specified grid, working instead on an arbitrary set of real numbers.

Let a set $\mathbb{A}$ of real numbers together with operations $\boxdot : \mathbb{A} \times \mathbb{A} \to \mathbb{A}$ for $\circ \in \{+, -, \times, /\}$ be given satisfying the first standard model [13]

$$\forall x, y \in \mathbb{A}: \quad x \boxdot y = (x \circ y)(1 + \delta), \quad |\delta| \leqslant \mathtt{u} \tag{14}$$

for a constant $\mathtt{u}$. That is our minimum accuracy requirement bounding the relative error of $x \boxdot y \in \mathbb{A}$ with respect to the real result $x \circ y \in \mathbb{R}$.

Most types of computer arithmetic used for numerical computations satisfy the first standard model, chief amongst IEEE-754-type arithmetics. That includes *flush-to-zero* models [no gradual underflow] if the underflow range is excluded.

The first and second standard model are not satisfied for multiplication and division in fixed-point arithmetic (although addition and subtraction is always exact). They are also not satisfied for an arithmetic without guard digit [13, Section 2.4] and [9] which has been used in the early days of computers.[1]

The standard model (14) leaves much freedom for the actual definition of the computer arithmetic, it neither implies $x \boxdot y = x \circ y$ if $x \circ y \in \mathbb{A}$, nor $a \boxdot b = c \boxdot d$ if $a \circ b = c \circ d$.

Moreover, there is quite a gap between the active "best approximation" property (2) and the mere accuracy requirement (14) as by the following example.

Example 1: Consider a 3-digit decimal arithmetic, and $x + y$ for $x = 4.96$ and $y = 5$. Then $x + y = 9.96$ is representable in 3 decimal digits, and $x \boxdot y = x + y = 9.96$ is the best approximation in the sense of (2). However, any choice of

$$x \boxdot y \in \{\, 9.92,\ 9.93,\ 9.94,\ 9.95,\ 9.96,\ 9.97,\ 9.98,\ 9.99,\ 10.0 \,\}$$

satisfies the standard model (14) for $\mathtt{u} = \frac{1}{2}\beta^{1-p} = 0.005$.

Consider the sum of $p_1, \ldots, p_n \in \mathbb{A}$. The first standard model (14) suffices to prove the standard estimate (4), but without additional assumptions, the factor $(1 + \mathtt{u})^{n-1} - 1$ cannot be replaced by $(n-1)\mathtt{u}$.

Example 2: Consider a logarithmic number system $\mathbb{F} := \{0\} \cup \{\pm c^k : k \in \mathbb{Z}\}$ for $1 < c \in \mathbb{R}$ with rounding upwards. Then $\mathtt{u} = \frac{c-1}{c+1}$ and, for sufficiently small $e \in \mathbb{F}$, $(1 \boxplus e) \boxplus e = c^2$ but $c^2 - (1 + 2e) > 2\frac{c-1}{c+1}(1 + 2e) = 2\mathtt{u}(1 + 2e)$. The reason is that an arbitrarily small summand e causes a relative error of almost size $\mathtt{u}$.

A. The standard model together with Assumption A

Surprisingly, for the improved and linearized bound (5) the following Assumption A suffices:

$$\forall a, b \in \mathbb{A}: \qquad |\,(a \boxplus b) - (a + b)\,| \leqslant \min(|a|, |b|). \tag{15}$$

For a computer arithmetic with some nearest rounding (2) that follows by

$$\begin{aligned} |\,(a \boxplus b) - (a + b)\,| &\leqslant \min(|a - (a + b)|, |b - (a + b)|) \\ &= \min(|a|, |b|). \end{aligned} \tag{16}$$

It is not true for a directed rounding, but it is true for Dekker's truncated rounding [5, Definition 3.5], i.e., a faithful rounding such that a nonzero error $(a \boxplus b) - (a + b)$ and the summand of smallest absolute value have opposite signs.

To demonstrate how the first standard model and Assumption A, that is (14) and (15) interplay, we repeat the proof in [34] of the linearized estimate (5). Given $p \in \mathbb{A}^n$ and proceeding by induction we set $s_n := \hat{s}_{n-1} + p_n$, so that the induction hypothesis implies

$$\begin{aligned} \Delta := |\ \hat{s}_n - \textstyle\sum_{i=1}^{n} p_i\ | &= |\hat{s}_n - s_n + \hat{s}_{n-1} - \textstyle\sum_{i=1}^{n-1} p_i| \\ &\leqslant |\hat{s}_n - s_n| + (n-2)\mathtt{u} \textstyle\sum_{i=1}^{n-1} |p_i|\ . \end{aligned} \tag{17}$$

[1] Note that still today almost all cheap decimal pocket calculators without exponent have no guard digit so that completely wrong results may be produced.

We distinguish two cases. First, assume $|p_n| \leqslant \mathrm{u}\sum_{i=1}^{n-1}|p_i|$. Then (15) implies

$$|\hat{s}_n - s_n| = |(\hat{s}_{n-1} \boxplus p_n) - (\hat{s}_{n-1} + p_n)| \leqslant |p_n| \leqslant \mathrm{u}\sum_{i=1}^{n-1}|p_i| \;, \tag{18}$$

and inserting into (17) finishes this part of the proof. Henceforth, assume $\mathrm{u}\sum_{i=1}^{n-1}|p_i| < |p_n|$. Then (14) gives

$$|\hat{s}_n - s_n| \leqslant \mathrm{u}|s_n| = \mathrm{u}\Big|\hat{s}_{n-1} - \sum_{i=1}^{n-1} p_i + \sum_{i=1}^{n} p_i\Big| \;,$$

so that (17) and the induction hypothesis yield

$$\begin{aligned} \Delta &\leqslant \mathrm{u}\big[\ (n-2)\mathrm{u}\textstyle\sum_{i=1}^{n-1}|p_i| + \sum_{i=1}^{n}|p_i|\ \big] \\ &\quad +(n-2)\mathrm{u}\textstyle\sum_{i=1}^{n-1}|p_i| \\ &< \mathrm{u}\big[\ (n-2)|p_n| + |p_n| + \textstyle\sum_{i=1}^{n-1}|p_i|\ \big] \\ &\quad +(n-2)\mathrm{u}\textstyle\sum_{i=1}^{n-1}|p_i| \\ &= (n-1)\mathrm{u}\textstyle\sum_{i=1}^{n}|p_i| \;. \qquad \square \end{aligned}$$

By Table III, u can be replaced by $\frac{\mathrm{u}}{1+\mathrm{u}}$ for an IEEE-754 p-digit base-β floating-point arithmetic.

Assumption A, that is (15) is the key to the proof. In fact, the weaker assumption (20) in the following Theorem 4.1 suffices to prove more: in [22] it is shown that, for any order of evaluation, both the linearized error estimates (5) for the sum and (7) for the dot product remain true for an arithmetic according to the first standard model (14) with the additional assumption (20). The proof is more involved.

Theorem 4.1: [22] Let a binary tree T with root r be given. For a node j of T, denote the set of inner nodes of the subtree with root j by N_j, and the set of its leaves by L_j. To each leaf $i \in L_r$ associate a real number x_i, and let to each inner node $j \in N_r$ a real number ε_j be associated. Define

$$s_j := \begin{cases} x_j & \text{if} \quad j \in L_r \\ (s_{\mathrm{left}(j)} + s_{\mathrm{right}(j)})(1+\varepsilon_j) & \text{if} \quad j \in N_r, \end{cases}$$

where $\mathrm{left}(j)$ and $\mathrm{right}(j)$ denote the left and right child of an inner node j, respectively. Furthermore, define for all inner nodes j

$$\delta_j := s_j - s_{\mathrm{left}(j)} - s_{\mathrm{right}(j)} \tag{19}$$

as well as, with the convention $\frac{0}{0} := 0$,

$$\xi_j := \frac{|\delta_j|}{\sum_{i\in L_j}|s_i| + \sum_{i\in N_j\backslash\{j\}}|\delta_i|}.$$

Suppose

$$|\delta_j| \leqslant \min_{k\in\{\mathrm{left}(j),\mathrm{right}(j)\}} \Big\{|s_k| + \sum_{i\in N_j\backslash N_k}\xi_i \sum_{i\in L_k}|s_i|\Big\} \tag{20}$$

is true for all inner nodes j. Then $\Delta_r := s_r - \sum_{i\in L_r} s_i$ satisfies

$$|\Delta_r| \leqslant \sum_{i\in N_r}|\delta_i| \leqslant \sum_{i\in N_r}\xi_i \sum_{i\in L_r}|s_i| \leqslant \sum_{i\in N_r}|\varepsilon_i| \sum_{i\in L_r}|s_i|. \tag{21}$$

The estimate is sharp in the sense that for arbitrary $\varepsilon_j \in [0,1]$ there exists a tree T such that (20) is satisfied and there are equalities in (21).

Here s_j is the computed value of $s_{\mathrm{left}(j)} + s_{\mathrm{right}(j)}$ with error δ_j for an inner node j. Therefore, the only assumption (20) in Theorem 4.1 is a trivial consequence of Assumption A. For the specific case of p-digit arithmetic to base β it follows $\sum\xi_i \leqslant \sum|\varepsilon_i| \leqslant (n-1)\mathrm{v}$ with $\mathrm{v} := \mathrm{u}/(1+\mathrm{u})$ according to Table III.

The replacement of u by v was used in [16] for a simple proof of the linearized estimate (7) for dot products.

Example 3: For real x_j, denote the floating-point sum of all $\mathrm{fl}(x_j)$ by $\hat{r}$, so that $|\hat{r} - \sum_{j=1}^{n}\mathrm{fl}(x_j)| \leqslant (n-1)\mathrm{v}\sum_{j=1}^{n}|\mathrm{fl}(x_j)|$. Rounding to nearest implies $|\mathrm{fl}(x_j) - x_j| \leqslant \mathrm{v}|x_j|$, so that [16]

$$\begin{aligned} |\hat{r} - \textstyle\sum_{j=1}^{n} x_j| &\leqslant \big|\hat{r} - \textstyle\sum_{j=1}^{n}\mathrm{fl}(x_j)\big| + \textstyle\sum_{j=1}^{n}\big|x_j - \mathrm{fl}(x_j)\big| \\ &\leqslant (\mathrm{v} + (n-1)\mathrm{v}(1+\mathrm{v}))\textstyle\sum_{j=1}^{n}|x_j|, \end{aligned}$$

and $\mathrm{v} + (n-1)\mathrm{v}(1+\mathrm{v}) \leqslant n\mathrm{v}(1+\mathrm{v}) \leqslant n\frac{\mathrm{v}}{1-\mathrm{v}} = n\mathrm{u}$ proves the desired estimate (7).

We will use the same concept later by improving $(n-1)\mathrm{u}$ into the optimal factor $\frac{(n-1)\mathrm{u}}{1+(n-1)\mathrm{u}}$ for summation to show linearized estimates for other compound operations such as blocked summation or sums of products.

B. The standard model together with Assumption B

Next we are interested in linear estimates for other types of rounding, for example directed or some faithful rounding. Following the first standard model (14) one might think just to replace u by $2\mathrm{u}$ to obtain similar results. That is not true because Assumption A (and also the weaker (20)) is not satisfied: adding an arbitrarily small positive $e \in \mathbb{F}$ to 1 in rounding upwards results in the successor of 1 with an error larger than e.

For the specific case of rounding upwards and IEEE-754 binary floating-point arithmetic, the following estimate, similar to (5), with adapted relative rounding error unit was shown for $\beta = 2$ in [32]. Let $x \in \mathbb{F}^n$ be given, and denote by $\hat{r}$ the sum computed in any order of evaluation and with all additions in rounding upwards. Then in [32] it was shown that

$$|\hat{r} - r| \leqslant 2(n-1)\mathrm{u}\sum_{i=1}^{n}|x_i| \quad \text{provided that } 4n\mathrm{u} \leqslant 1. \tag{22}$$

The restriction on n is necessary for any base β: for $x_1 = 1$ and $x_{2\ldots n}$ arbitrarily small positive numbers the sum increases each intermediate result to the next successor in $\mathbb{F}$. Up to $n \leqslant \beta^p - \beta^{p-1}$ summands the error is $2\mathrm{u}$, but passing the intermediate result β increases the error to $\beta\mathrm{u}$, eventually spoiling the estimate (22).

Estimate (22) holds true for rounding upwards. We improve that by showing that under Assumption B, that is (26) for the ε_k defined in (25), a linearized error bound is true for any directed or faithful rounding. In terms of IEEE-754 p-digit base-β floating-point arithmetic this assumption amounts to the fact that bounds on the maximum relative error in the intervals $\pm[\beta^m, \beta^{m+1}]$ are constant and with respect to β^m, namely $\mathrm{u}\beta^m$. The mathematical formulation is Assumption B, that is (26) with respect to (25). It implies the mandatory restriction on n.

Theorem 4.2: [22] Let a binary tree T with n leaves be given. To each leaf associate a real number x_i and to each inner node associate a real number s_k forming vectors $x \in \mathbb{R}^n$ and $s \in \mathbb{R}^{n-1}$. Denote by σ_k the sum of the values associated with the children of an inner node k, and define

$$\delta_k := s_k - \sigma_k \qquad \text{for} \quad 1 \leqslant k \leqslant n-1. \tag{23}$$

Let nonnegative real numbers λ, μ be given such that

$$\lambda \leqslant \sum_{i=1}^{n} |x_i| < \mu. \tag{24}$$

Define for $1 \leqslant k \leqslant n-1$

$$\varepsilon_k := \begin{cases} \frac{|\delta_k|}{\lambda} & \text{if} \quad |\sigma_k| < \mu, \\ \frac{|\delta_k|}{\mu} & \text{otherwise,} \end{cases} \tag{25}$$

with the convention $\frac{0}{0} := 0$. Assume

$$\sum_{i=1}^{n-1} \varepsilon_i \leqslant \frac{\mu - \lambda}{\lambda}. \tag{26}$$

Then $|\sigma_k| < \mu^2/\lambda$ for $1 \leqslant k \leqslant n-1$, and for r denoting the root of T,

$$\left|s_r - \sum_{i=1}^{n} x_i\right| \leqslant \sum_{i=1}^{n-1} |\delta_i| \leqslant \sum_{i=1}^{n-1} \varepsilon_i \sum_{i=1}^{n} |x_i|. \tag{27}$$

The interpretation, in particular the role of λ and μ for Assumption B, becomes clear from the proof of the following corollary.

Corollary 4.3: [22] Let $\mathbb{F}$ be a p-digit floating-point number system in base β, and denote by s the result of a floating-point summation of $x_1, \ldots, x_n \in \mathbb{F}$ using some faithful addition. If $n \leqslant 1 + \frac{\beta-1}{2}\mathrm{u}^{-1}$, then

$$\left|s - \sum_{j=1}^{n} x_j\right| \leqslant 2(n-1)\mathrm{u} \sum_{j=1}^{n} |x_j|.$$

PROOF. Let $m \in \mathbb{Z}$ such that $\lambda := \beta^m \leqslant \sum_{j=1}^{n} |x_j| < \beta^{m+1} =: \mu$. Let σ_k be as in Theorem 4.2, and denote by $\mathrm{ufp}(\sigma_k)$ the largest power of β being less than or equal to $|\sigma_k|$. If $|\sigma_k| < \mu$, then $\mathrm{ufp}(\sigma_k) \leqslant \lambda$ and (25) implies

$$\varepsilon_k = \frac{|\delta_k|}{\lambda} \leqslant \frac{|\delta_k|}{\mathrm{ufp}(\sigma_k)} = \frac{|\mathrm{fl}(\sigma_k) - \sigma_k|}{\mathrm{ufp}(\sigma_k)} \leqslant 2\mathrm{u},$$

and otherwise $\mu \leqslant |\sigma_k| < \mu^2/\lambda = \beta\mu$ shows $\mathrm{ufp}(\sigma_k) = \mu$ and

$$\varepsilon_k = \frac{|\delta_k|}{\mu} = \frac{|\delta_k|}{\mathrm{ufp}(\sigma_k)} = \frac{|\mathrm{fl}(\sigma_k) - \sigma_k|}{\mathrm{ufp}(\sigma_k)} \leqslant 2\mathrm{u}.$$

Thus, all ε_k are bounded by $2\mathrm{u}$. Additionally, the limit on n implies

$$\sum_{j=1}^{n-1} \varepsilon_j \leqslant (n-1)2\mathrm{u} \leqslant \beta - 1 = \frac{\mu - \lambda}{\lambda}, \tag{28}$$

so that the assumption (26) in Theorem 4.2 is satisfied. Thus,

$$\left|s - \sum_{j=1}^{n} x_j\right| \leqslant \sum_{j=1}^{n-1} |\delta_j| \leqslant (n-1) \cdot 2\mathrm{u} \sum_{j=1}^{n} |x_j|. \qquad \square$$

Theorem 4.2 is tailored to the fact that the maximum absolute error of a floating-point operation is uniformly bounded by $2\mathrm{u}/\lambda$ in the interval $[\lambda, \mu]$ with $\lambda = \beta^m$ and $\mu = \beta^{m+1}$, that is Assumption B. As we have seen, some restriction on n is mandatory; here the maximal number of such errors in $[\lambda, \mu]$, see (28), bounds the number of summands n to $1 + \frac{\beta-1}{2}\mathrm{u}^{-1}$.

The application of the theorem to faithful rounding is just an example, it applies to nearest rounding with replacing $2\mathrm{u}$ by u and other roundings as well.

C. The standard model together with Assumption C

So far we saw that given $x_1, \ldots, x_n$ the error of the sum in some computer arithmetic is bounded by $(n-1)\mathrm{u}\sum |x_i|$ no matter what the order of evaluation, and for a nearest rounding without restriction on n. The corresponding traditional constant $(1+\mathrm{u})^{n-1} - 1$ in the estimate is straightforward to prove, whereas for the linearized bounds some effort is necessary.

An interesting generalization concerns error bounds depending on the height of a summation tree. For an addition tree of height h, the traditional Wilkinson-type constant $(1+\mathrm{u})^h - 1$ in the estimate follows straightforwardly. That bound can be linearized to $h\mathrm{u}$ for a standard model together with an Assumption C. For a balanced tree in binary64 that restricts the number of summands to $n \leqslant 2^{94{,}906{,}264}$.

In terms of IEEE-754 p-digit base-β floating-point arithmetic Assumption C amounts to the fact that bounds on the maximum relative error increase from the interval $\pm[\beta^m, \beta^{m+1}]$ to the interval $\pm[\beta^{m+1}, \beta^{m+2}]$ by a constant factor β. That factor β is associated with the arithmetic model. The mathematical formulation of Assumption C are assumptions (29) and (30) in Theorem 4.4. Note that β is some real number, not necessarily related to some grid.

Theorem 4.4: [24] Let an α-ary tree T with root r and height h be given. For an inner node j of T, denote the set of leaves of the corresponding subtree by L_j and the set of all its inner nodes including j by N_j. To each leaf i of T associate a real number x_i. Moreover, let positive real numbers b, ε as well as $\beta \geqslant \alpha$ be given, and let two numbers

$$\delta_j \in \mathbb{R} \qquad \text{and} \qquad b_j \in \{0\} \cup \{\beta^m b \mid m \in \mathbb{Z}\} \tag{29}$$

be assigned to each inner node j of T. Suppose that for each inner node j

$$|\delta_j| \leqslant b_j \leqslant \varepsilon \left(\sum_{i \in L_j} |x_i| + \sum_{i \in N_j \setminus \{j\}} |\delta_i| \right). \tag{30}$$

If h is restricted by

$$h \leqslant 2\sqrt{c_h \varepsilon^{-1}} - 1 \text{ with } c_h := \begin{cases} \beta^{-1} - \beta^{-2} & \text{if} \quad \alpha = \beta \\ 1 - \alpha\beta^{-1} & \text{otherwise,} \end{cases} \tag{31}$$

then

$$\sum_{i\in N_r} |\delta_i| \leqslant h\varepsilon \sum_{i\in L_r} |x_i|. \tag{32}$$

The following corollary formulates the result for IEEE-754 p-digit base-β floating-point arithmetic. From the proof the previous interpretation of Assumption C becomes clear. It is satisfied for any nearest as well as for any directed or faithfully rounded summation.

Corollary 4.5: [24] For an IEEE-754 p-digit base-β floating-point arithmetic, let s be the result of a floating-point summation of $p_1, \ldots, p_n \in \mathbb{F}$ in some nearest addition in any order. If the height h of the corresponding binary summation tree satisfies

$$h \leqslant \begin{cases} \mathrm{u}^{-\frac{1}{2}} - 1 & \text{if } \ \beta = 2 \\ \sqrt{4-8\beta^{-1}}\mathrm{u}^{-\frac{1}{2}} - 1 & \text{otherwise,} \end{cases} \tag{33}$$

then

$$\left| s - \sum_{j=1}^{n} p_j \right| \leqslant h\mathrm{u} \sum_{j=1}^{n} |p_j|. \tag{34}$$

The result remains valid for any faithful addition when replacing the error constant u by $2\mathrm{u}$ in (33) and (34).

PROOF. Let T denote a binary summation tree, where to each inner node j of T the respective intermediate summation result s_j including the perturbations δ_i is associated. Using the notation as in Theorem 4.4 it follows $s_j = \sum_{i\in L_j} x_i + \sum_{i\in N_j} \delta_i$. Furthermore, let $b = \varepsilon = \eta$, where $\eta = \mathrm{u}$ in case of a nearest addition, and $\eta = 2\mathrm{u}$ in case of faithful addition. Define $b_j := \eta \cdot \mathrm{ufp}(s_j - \delta_j)$ for all inner nodes j. This definition of b_j complies with assumption (29), i.e., $b_j \in \{0\} \cup \{\beta^m \eta \mid m \in \mathbb{Z}\}$. Moreover,

$$|\delta_j| \leqslant b_j \leqslant \eta |s_j - \delta_j| \leqslant \eta \left(\sum_{i\in L_j} |x_i| + \sum_{i\in N_j\backslash\{j\}} |\delta_i| \right)$$

validates the assumption (30). Finally, for $\alpha = 2$,

$$h \leqslant \begin{cases} \eta^{-\frac{1}{2}} - 1 = 2\sqrt{(\beta^{-1}-\beta^{-2})\eta^{-1}} - 1 & \text{if } \ \beta = \alpha \\ \sqrt{4-8\beta^{-1}}\,\eta^{-\frac{1}{2}} - 1 & \\ \quad = 2\sqrt{(1-\alpha\beta^{-1})\eta^{-1}} - 1 & \text{otherwise} \end{cases}$$

shows the equivalence of (31) and (33). Thus (34) follows. □

Similar to Example 3, the result extends to dot products. Denote by s the result of a floating-point dot product of $a, b \in \mathbb{F}^n$ in some rounding to nearest. Let the height h of the corresponding binary evaluation tree satisfy (33). Then, barring over- and underflow,

$$\left| s - a^T b \right| \leqslant h\mathrm{u} \sum_{i=1}^{n} |a_i b_i|. \tag{35}$$

For faithful rounding the result is true when replacing the error constant u by $2\mathrm{u}$.

Given some computer arithmetic satisfying Assumption B or C, there is any freedom for the corresponding subset of $\mathbb{R}$ of representable numbers. Unless that set is fairly exotic, we can expect that Assumption C implies Assumption B.

D. The standard model together with Assumptions A and B

Even the linearized bounds presented so far are not optimal for an IEEE-754 p-digit base-β floating-point arithmetic. We guess everybody thinking about a worst case error for recursive summation quickly constructs $x = (1, \mathrm{u}, \ldots, \mathrm{u})$. The result $\hat{r}$ of a nearest addition with rounding ties to even is 1, so that the error is $(n-1)\mathrm{u}$ and satisfies

$$|\hat{r} - \sum_{i=1}^{n} x_i| = \frac{(n-1)\mathrm{u}}{1+(n-1)\mathrm{u}} \sum_{i=1}^{n} |x_i|. \tag{36}$$

By means of explicit examples (cf. [24]) it is easy to see that some restriction on n is mandatory for (36). But although it was common belief that this is the worst case, it could not be proved.

The first result in this direction can be found in [28], where Mascarenhas introduces a new concept of using continuous mathematics to analyze floating-point arithmetic to prove (36) for recursive summation provided that $n \leqslant \frac{1}{20}\mathrm{u}^{-1}$.

However, despite the comparatively small upper bound on n and the restriction to recursive summation, the given proof is rather complicated and longish. In [24, Theorem 5] a more general result was proved using fairly simple arguments. The assumptions are in fact more general than Assumptions A and B together. However, the mathematical statement is technical, so we state only the following corollary (which is [24, Proposition 1]) for IEEE-754 p-digit base-β arithmetic. Note that this arithmetic satisfies Assumptions A and B.

Theorem 4.6: Let a p-digit floating-point arithmetic to base β be given. Let $\hat{r}$ be the result of a floating-point summation of $p_1, \ldots, p_n \in \mathbb{F}$ in some nearest addition in arbitrary order. Then

$$\left| \hat{r} - \sum_{j=1}^{n} p_j \right| \leqslant \frac{(n-1)\mathrm{u}}{1+(n-1)\mathrm{u}} \sum_{j=1}^{n} |p_j| \quad \text{if } \ n \leqslant 1 + \frac{\beta-1}{2}\mathrm{u}^{-1}. \tag{37}$$

As has been mentioned, the result holds true for a more general computer arithmetic as well. In that case the mandatory restriction on n has to be re-computed.

The upper bound on n is almost sharp. For rounding ties away from zero it cannot be replaced by the next larger integer; for rounding ties to even, $p \geqslant 3$ mantissa digits and even β the upper bound cannot be increased by $2 + \frac{\beta}{2}$, see [24].

OPEN PROBLEM 3. Devise a sharp error estimate for dot products in the spirit of (37).

E. Some applications

As exploited in [24], this Theorem 4.4 has a number of consequences. Denote by $s = \mathrm{float}(\textit{expression})$ the result of an expression with each operation replaced by the corresponding floating-point operation in some nearest rounding. The evaluation may be in any order but, if applicable, respecting parentheses. First, consider a sum of products

$$s := \sum_{i=1}^{n} \prod_{j=1}^{m} p_{ij} \qquad \text{for } p_{ij} \in \mathbb{F}. \tag{38}$$

Provided that $(n+m-2)\mathrm{u} < 1$, the standard Wilkinson-type error estimate [41] gives

$$\left|\mathrm{float}\Big(\sum_{i=1}^{n}\prod_{j=1}^{m} p_{ij}\Big) - s\right| \leqslant \gamma_{n+m-2}\sum_{i=1}^{n}\prod_{j=1}^{m}|p_{ij}|.$$

Corollary 4.5 and barring over- and underflow implies the linearized estimate

$$\left|\mathrm{float}\Big(\sum_{i=1}^{n}\prod_{j=1}^{m} p_{ij}\Big) - s\right| \leqslant (n+m-2)\mathrm{u}\sum_{i=1}^{n}\prod_{j=1}^{m}|p_{ij}| \qquad (39)$$

provided that

$$m \leqslant \beta^{-\frac{1}{2}}\mathrm{u}^{-\frac{1}{2}}\ , \quad n \leqslant 1 + \frac{\beta-1}{2}\mathrm{u}^{-1}, \quad \text{and} \quad m \leqslant n.$$

For binary floating-point numbers, assuming $m \leqslant \mathrm{u}^{-\frac{1}{2}}$ suffices for (39) to hold true. The proof is very similar to that in Example 3 for dot products, where the improved error bound $\mathrm{v} = \frac{\mathrm{u}}{1+\mathrm{u}}$ instead of u was sufficient to obtain the error bound $n\mathrm{u}$; now the optimal error bound (37) is used instead of $(n-1)\mathrm{u}$.

Another direct application is a bound on the error of a Vandermonde matrix times a vector. Let $V_{ij} = \alpha_j^i$ for $0 \leqslant i, j \leqslant n$ for given $\alpha_j \in \mathbb{F}$. Then $(Vx)_i = \sum\limits_{j=0}^{n}\alpha_j^i x_j$, so that for a vector $x \in \mathbb{F}^{n+1}$, starting with index 0, we obtain

$$\begin{aligned} |\mathrm{float}(Vx) - Vx| &\leqslant \text{diag } (n\mathrm{u}, n\mathrm{u}+\mathrm{u}, \ldots, 2n\mathrm{u})\,|V|\,|x| \\ &\leqslant 2n\mathrm{u}\,|V|\,|x|. \end{aligned}$$

Another application is an error estimate for blocked summation. Let a vector $p \in \mathbb{F}^{mn}$ be given and consider

$$s := \mathrm{float}\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{m} p_{ij}\right)\right). \qquad (40)$$

Then $|s - \sum_{ij} p_{ij}| \leqslant \gamma_{n+m-2}\sum_{ij}|p_{ij}|$, the standard Wilkinson-type error estimate, improves to

$$\left|s - \sum_{ij} p_{ij}\right| \leqslant (n+m-2)\mathrm{u}\sum_{ij}|p_{ij}| \qquad (41)$$

provided that $\max(m,n) \leqslant 1 + \frac{\beta-1}{2}\mathrm{u}^{-1}$.

V. Faithful results by a simplified pair arithmetic

In the previous sections error bounds for single or compound operations were shown, either for an actively given arithmetic such as IEEE-754, or for a computer arithmetic passively satisfying the first error model (14) with some additional weak assumptions. In this section, accuracy estimates related to the condition number of the problem will be investigated, in particular methods to achieve a faithfully rounded result.

Common methods to improve the accuracy are compensated algorithms. Prominent examples are Kahan's and Shewchuk's summation algorithms [19], [40] for which a small backward error[2] of size $2\mathrm{u}$ follows. The doubly compensated summation by Priest [33] requires ordering of the summands and proves a forward error of size $2\mathrm{u}$.

A notable exception to the many algorithms is Neumaier's summation [30] which he found as a master student in 1974. Obviously without knowing, he uses what we call today "error-free transformations", a term I coined in [31]. For example, consider

```
function [x,y] = FastTwoSum(a,b)
  x = a + b;
  y = a - (x - b);
```

For any two floating-point numbers $a, b \in \mathbb{F}$ with $|a| \geqslant |b|$ it holds $a + b = x + y$ [5] for a nearest rounding and $\beta \leqslant 3$. Similar algorithms for addition without constraint on the ordering of the summands (`TwoSum`) and for products (`TwoProduct`) are known [5], [20], [29]. Note that for IEEE-754 precision-p base-β arithmetic it is necessary [29] that $a + b$ is computed in rounding to nearest, otherwise the error $a \boxplus b - (a+b)$ need not be representable.

Algorithm `TwoSum` implies an error-free vector transformation. Given a vector $p \in \mathbb{F}^n$, the call `q = VecSum(p)` of

```
function p = VecSum(p)
  for i=2:n
    [p(i),p(i-1)] = TwoSum(p(i),p(i-1))
```

produces a vector $q \in \mathbb{F}^n$ with $\sum p_i = \sum q_i$ and $q_n = \mathrm{float}(\sum p_i)$. Summing the vector q in floating-point after a single call of `VecSum` is Algorithm `Sum2` in [31], which is identical to Neumaier's fourth algorithm in [30]. The accuracy of the result depends on the condition number $\boldsymbol{k} = (\sum|p_i|)/|\sum p_i|$. The results in [31] imply that basically for $\boldsymbol{k} \lesssim 1/(2n^2\mathrm{u})$ the result of `Sum2` is faithfully rounded, so that there is no other floating-point number between the true and the computed result. A similar result holds true for the dot product algorithm `Dot2` in [31].

Using similar techniques, a number of algorithms with faithfully rounded result have been developed for several standard problems in numerical analysis. For example, Graillat gave in [10] a compensated scheme for $\prod_{i=1}^n x_i$ with faithfully rounded result provided that

$$n < \frac{\sqrt{1-\mathrm{u}}}{\sqrt{4+2\mathrm{u}} + 2\sqrt{(1-\mathrm{u})\mathrm{u}}}\mathrm{u}^{-1/2}\ ;$$

Boldo and Muñoz showed in [3] a compensated polynomial evaluation to be faithful provided that

$$\boldsymbol{k} := \frac{\sum_{i=0}^{n}|p_i||x^i|}{|\sum_{i=0}^{n} p_i x^i|} < \frac{(1-\mathrm{u})(1-2n\mathrm{u})^2}{4n^2\mathrm{u}(2+\mathrm{u})}\ ;$$

in [31] algorithm `Sum2` is shown to be faithful if

$$\frac{(n-2)(n-1)}{(1-(n-2)\mathrm{u})(1-(n-1)\mathrm{u})} \leqslant \frac{1}{2\boldsymbol{k}\mathrm{u}}.$$

In binary64 the assumptions read $n < 47{,}453{,}132$, and, for $n = 1000$, $\boldsymbol{k} < 1.13 \cdot 10^9$ and $\boldsymbol{k} < 4.52 \cdot 10^9$, respectively. Sometimes restrictions apply, for example the latter result

[2] The computed result is the true result for a small perturbation of the input data [13].

for summation supposes recursive summation in binary, and all results suppose that transformation algorithms such as `TwoSum` are indeed error-free, i.e., produce $x + y = a + b$.

Another approach to compute a faithfully rounded result, also based on error-free transformations, is Bailey's double-double arithmetic [1]. Here numbers are represented as an unevaluated sum of two elements of $\mathbb{F}$.

The double-double arithmetic is analyzed in [18]. For addition, for example, two algorithms are given. The first algorithm [18, Algorithm 5], called "sloppy addition", was already given by Dekker as add2 in [5]. However, the result may have no significance at all.

```
Function $(c, g)$ = SloppyDWPlusDW$(a, e, b, f)$
    $[c, t] = \mathrm{TwoSum}(a, b)$
    $s = \mathrm{fl}(e + f)$
    $g = \mathrm{fl}(t + s)$
    $[c, g] = \mathrm{FastTwoSum}(c, g)$
```

Alternatively, an accurate algorithm AccurateDWPlusDW is analyzed with relative error not larger than $\frac{3\mathrm{u}^2}{1-4\mathrm{u}}$. The double-double arithmetic is based on IEEE-754 binary arithmetic and error-free transformations.

The target for this section is to introduce a new and simplified pair arithmetic with the goal to give conditions for which the final result is faithful. That applies to general arithmetic expressions comprising of $+, -, \times, /, \sqrt{\cdot}$. As we will see this includes all methods mentioned at the beginning of this section. Another target is to require as weak assumptions on the arithmetic as possible, but nevertheless guaranteeing a faithfully rounded result under specified conditions.

Our pair arithmetic is more general than previous approaches in several aspects. First, we require only an arithmetic following the first standard model, neither of the previous Assumptions A, B or C has to be satisfied. Hence a situation as in Example 1 may occur.

Second, to estimate the error of an individual operation only an approximation of the residual is needed, for example of $a \boxplus b - (a + b)$ for addition. Again that approximation is only required to satisfy the first standard model. In particular, "error-free transformations" are no longer needed but replaced by "approximate transformations".

Third, for a pair (c, g), no relation between c and g is required. Fourth, the pair operations are simplified requiring less operations compared to double-double. Fifth, for every pair (c, g) the first part c is equal to the result when computing in the given computer arithmetic.

As a special example, all of the following results are true for an IEEE-754 p-digit base-β arithmetic and any rounding scheme. As has been mentioned, for directed rounding error-free transformations are not possible because the error $a \boxplus b - (a + b)$ need not be representable. Nevertheless, our arithmetical model remains applicable.

Let $\mathbb{A}$ be an arbitrary discrete set of real numbers. For a given positive constant $\mathrm{v} < 1$ define the *working set* of $\mathbb{A}$ by

$$\mathbb{W} := \{r \in \mathbb{R} \mid \exists f \in \mathbb{A} \colon |f - r| \leqslant \mathrm{v}|r|\}. \qquad (42)$$

Consider a real function $\mathrm{g} \colon \mathbb{R}^n \to \mathbb{R}$ and let $x \in \mathbb{A}^n$ be such that $\mathrm{g}(x) \in \mathbb{W}$. The left-hand side of $c \leftarrow \mathrm{g}(x)$ for $c \in \mathbb{A}$ is called an $\mathbb{A}$-arithmetic approximation if

$$c = \mathrm{g}(x)(1 + \varepsilon) \qquad \text{with} \quad |\varepsilon| \leqslant \mathrm{v}, \qquad (43)$$

abbreviated by $c \leftarrow g(x)$. We choose the notation "$\leftarrow$" rather than "$\mathrm{fl}(\cdot)$" to indicate that only the error estimate (43) has to be satisfied, i.e., a relation rather than a function. Our general assumption on the arithmetic in $\mathbb{A}$ is as follows.

Assumption 5.7: For $a, b \in \mathbb{A}$ and $\circ \in \{+, -, \times, /\}$, let $\hat{c} := a \circ b$. If $\hat{c} \in \mathbb{W}$, we assume that $c \leftarrow a \circ b$ can be evaluated and satisfies $|c - \hat{c}| \leqslant \mathrm{v}|\hat{c}|$ according to (43). A similar statement is true for the square root. Moreover, assume that for

$$\begin{array}{lll} t \leftarrow a \circ b - c & \text{if } c \leftarrow a \circ b & \text{for } \circ \in \{+, -, \times\}, \\ t \leftarrow a - bc & \text{if } c \leftarrow a/b, & \\ t \leftarrow a - c^2 & \text{if } c \leftarrow \sqrt{a} & \end{array} \qquad (44)$$

a method to evaluate t is available satisfying the estimate in (43) with appropriate interpretation.

For the special case of IEEE-754 binary64 floating-point arithmetic with rounding to nearest, Assumption 5.7 is satisfied if the real result $\hat{c}$ does not cause over- or underflow by setting $\mathrm{v} := \mathrm{u}/(1 + \mathrm{u})$ for $\mathrm{u} := 2^{-53}$, by replacing $c \leftarrow a \circ b$ by $c = \mathrm{fl}(a \circ b)$ and evaluating the expressions in (44) by appropriate error-free transformations[3], possibly using the fused multiply-add operation FMA.

Next we define our pair arithmetic [23]. An algorithm for subtraction follows directly from addition, and all results hold true *mutatis mutandis*. To ease the exposition we omit subtraction by the technical assumption $\mathbb{A} = -\mathbb{A}$. The comments "`//TwoSum or Add3`", "`FMA or TwoProduct`" etc. in the following algorithms refer to a possible implementation when using IEEE-754 arithmetic; they are not mandatory.

```
Function $(c,g)$ = CPairSum$((a, e),(b, f))$
    $c \leftarrow a + b$
    $t \leftarrow a + b - c$                  // TwoSum or Add3
    $s \leftarrow e + f$
    $g \leftarrow t + s$
```

The first part c of the result of our pair operations is always equal to the one computed in the given arithmetic. That property is spoiled by the final normalization step in SloppyDWPlusDW, also used in [25]. Technically, the double-double SloppyDWPlusDW and our CPairSum are identical up to the final normalization; however, the assumptions of our pair

[3] Error-free transformations require the absence of over- and underflow not only for the results but also for all intermediate values; see Boldo et al. [2].

```
Function (c,g) = CPairProd((a,e),(b,f))
    $c \leftarrow ab$
    $t \leftarrow ab - c$              // FMA or TwoProduct
    $q \leftarrow af$
    $r \leftarrow be$
    $s \leftarrow q + r$
    $g \leftarrow t + s$
```

```
Function (c,g) = CPairDiv((a,e),(b,f))
    $c \leftarrow a/b$
    $t \leftarrow a - bc$                                // FMA
    $p \leftarrow t + e$
    $q \leftarrow cf$
    $r \leftarrow p - q$
    $s \leftarrow b + f$
    $g \leftarrow r/s$
```

arithmetic are far less. The flop counts for the pair addition algorithms are as follows.

CPairSum	8 flops
SloppyDWPlusDW	11 flops
AccurateDWPlusDW	20 flops

Compared to double-double arithmetic, our other pair operations have a smaller flop count as well.

In turn, the results of the double-double arithmetic are usually more accurate than those of our pair arithmetic. However, one target was to derive provable conditions for a faithful result with as weak assumptions on the arithmetic as possible.

Consider an arbitrary arithmetic expression represented by a binary tree. For given input data we henceforth assume that all intermediate operations are well defined with result in the working set $\mathbb{W}$. That is in particular satisfied if, when using IEEE-754 p-digit base-β floating-point arithmetic, no intermediate over- or underflow occur.

To formulate the conditions for a faithfully rounded result, we need to define the condition number of an arithmetic expression. An essential ingredient is the notation of the *No Inaccurate Cancellation* (NIC) principle. Demmel et al. used that in [6] to identify algorithms computing accurate results basically independent of the condition number.[4]

```
Function (c,g) = CPairSqrt((a,e))
    $c \leftarrow \sqrt{a}$
    $t \leftarrow a - c^2$                               // FMA
    $r \leftarrow t + e$
    $s \leftarrow c + c$
    $g \leftarrow r/s$
```

Definition 5.8: Let T be an evaluation tree with input data p (the values at the leaves), and inner nodes consisting of operations from the set $\{+, \times, /, \sqrt{\cdot}\}$. If no sum with at least one addend not being input data is performed on numbers with opposite signs, then (T, p) complies with the *No Inaccurate Cancellation* (NIC) principle.

The rationale is to avoid catastrophic cancellation. If an arithmetic expression does not satisfy the NIC principle, for example $x + y - x$, then for large positive x and small positive y cancellation and a large relative error to the true result y occurs. That cannot happen for an arithmetic expression satisfying the NIC principle, the relative error of every intermediate to the corresponding true result may grow, but very slowly.

Definition 5.9: Consider an evaluation tree T with input data $p \in \mathbb{A}^n$ and inner nodes consisting of operations from the set $\{+, \times, /, \sqrt{\cdot}\}$. Let any pair of input numbers p_i and p_j that is added in T with negative result be replaced by $p'_i := -p_i$ and $p'_j := -p_j$, respectively. Moreover, let all other input numbers p_k be replaced by their absolute value $p'_k := |p_k|$. The so obtained data p' is called *NIC remodeled input data* to (T, p).

The rationale behind this definition is as follows. Let a compound operation be given depending on $x \in \mathbb{R}^n$. Then the error of an approximation is usually estimated relative to the maximal possible value S for all possible sign combinations of the x_i. Examples are $S = \sum |x_i|$ for summation (4) or $S = \sum_{i=0}^{n} |a_i x^i|$ for Horner's scheme (12). In those examples, the ratio between S and the true value for the original x_i is the condition number, and that is the result of the problem with NIC remodeled data[5].

Suppose evaluating an arithmetic expression by our pair arithmetic results in (c, g). In order to obtain a faithfully rounded result, an element of $\mathbb{A}$, we need to add c and g approximately. Here assuming the first standard model is not sufficient as by Example 1; this (and only this) final addition has to be done in some nearest rounding, otherwise the result cannot be guaranteed to be faithful. To be precise, we say $\hat{c} \in \mathbb{A}$ is a nearest $\mathbb{A}$-approximation to $c \in \mathbb{R}$ if

$$\forall a \in \mathbb{A}: \qquad |\hat{c} - c| \leqslant |a - c|. \tag{45}$$

Finally, we need a measurement for the minimum relative distance between two adjacent numbers in $\mathbb{A}$, namely

$$\eta := \inf \left\{ \frac{|s - t|}{|s + t|} : s, t \in \mathbb{A}, s \neq t \right\}, \tag{46}$$

For a p-digit base-β floating-point arithmetic it follows $\eta = \frac{\mathrm{u}}{\beta - \mathrm{u}} > \mathrm{u}/\beta$.

Based on that we can state our result for general arithmetic expressions.

Theorem 5.10: [23] Let an arithmetic expression be given by an evaluation tree T with n leaves, where to each inner node j an operation $\circ_j$ out of $\{+, \times, /, \sqrt{\cdot}\}$ is assigned. Moreover, to

[4] A famous example is to treat Hilbert matrices as Cauchy matrices allowing to faithfully compute the inverse or smallest singular value up to about dimension 10^8 solely in binary64.

[5] That is not always true, for example when divisions occur; however, for our condition to prove that a result is faithful it is sufficient.

every node j, inner node or leaf, let an integer k_j be assigned according to

$$k_j := \begin{cases} 0 & \text{if } j \text{ is leaf} \\ \max\{k_{\text{left}(j)}, k_{\text{right}(j)}\} + 1 & \text{if } \circ_j = + \\ k_{\text{left}(j)} + k_{\text{right}(j)} + 1 & \text{if } \circ_j = \times \\ k_{\text{left}(j)} + k_{\text{right}(j)} + 2 & \text{if } \circ_j = / \\ \lceil \frac{4}{5} k_{\text{child}(j)} + \frac{5}{4} \rceil & \text{if } \circ_j = \sqrt{\cdot}, \end{cases} \tag{47}$$

where left(j), right(j), and child(j) are denoting the left, right, and only child of j, respectively. For given input data $p \in \mathbb{A}^n$, let $(p_i, 0)$ be the pairs at the leaves of T, and denote by (c, g) the result evaluated at root r using our pair arithmetic. Furthermore, let $\hat{c}$ be the true result of the expression for input data p, and let $\hat{C}$ be the true result for the NIC remodeled input data p'.

Suppose that all denominators and all expressions below a square root comply with the NIC principle. Furthermore, suppose that k_j is not larger than $\mathrm{u}^{-\frac{1}{2}}$ for any node j comprising of division or square root. Otherwise k_j is unbounded.

Assume the pair arithmetic produces $(c, g) \in \mathbb{A}^2$ as a final result with final k according to (47), and define

$$\boldsymbol{k} := \frac{\hat{C}}{|\hat{c}|} \qquad \text{with the convention } \quad \frac{0}{0} := 1. \tag{48}$$

Let η be defined as in (46). If k is restricted via

$$k \leqslant \sqrt{\frac{\min\{\eta, \mathrm{u}\}}{\boldsymbol{k}\mathrm{u}^2}} - 2, \tag{49}$$

then a nearest $\mathbb{A}$-approximation of $c + g$ according to (45) is a faithful rounding of $\hat{c}$. For an expression complying with the NIC principle condition (49) reduces to

$$k \leqslant \frac{\sqrt{\min\{\eta, \mathrm{u}\}}}{\mathrm{u}} - 2.$$

That theorem yields conditions on $\boldsymbol{k}$ and the length of input data for all examples mentioned at the beginning of this section, and more.

For simplicity, assume an IEEE-754 p-digit base-β floating-point arithmetic. Then $\eta > \mathrm{u}/\beta$ reduces the condition on k to $k \leqslant \sqrt{\frac{1}{\beta \boldsymbol{k} \mathrm{u}}} - 2$ for general expressions, and the same with $\boldsymbol{k} = 1$ for expressions complying with the NIC principle. The error estimates in Table V follow by calculating the maximum value of k for the input expression.

For all examples mentioned at the beginning of this section, a faithfully rounded result is computed with as many or fewer operations and with weaker condition on n, but for any order of evaluation, and for any base β.

In the third example "binary" refers to binary summation. The sixth example "mixed $\times, /$" means that for n input numbers x_i a tree is evaluated with each node being multiplication or division. The final k satisfies $k \leqslant 2(n-1)$ and the tree complies with the NIC principle, so that $\boldsymbol{k} = 1$ and the bound on n for a faithfully rounded result follows.

Note in particular for polynomial interpolation, the second last problem, all denominators satisfy the NIC principle so that Theorem 5.10 is applicable. With adapted constants that holds also true for the faster approach

TABLE V: Faithfully rounded results by the pair arithmetic.

Problem	$\boldsymbol{k}$	bound on n
$s := \sum_{i=0}^{n} p_i x^i$	$\frac{\sum_{i=0}^{n} \lvert p_i x^i \rvert}{\lvert s \rvert}$	$n \leqslant \frac{1}{2\sqrt{\beta \boldsymbol{k} \mathrm{u}}} - 1$
$s := \sum_{i=1}^{n} x_i$	$\frac{\sum_{i=1}^{n} \lvert x_i \rvert}{\lvert s \rvert}$	$n \leqslant \frac{1}{\sqrt{\beta \boldsymbol{k} \mathrm{u}}} - 1$
$s := \sum_{i=1}^{n} x_i$ binary	$\frac{\sum_{i=1}^{n} \lvert x_i \rvert}{\lvert s \rvert}$	$\lceil \log_2(n) \rceil \leqslant \frac{1}{\sqrt{\beta \boldsymbol{k} \mathrm{u}}} - 2$
$s := \sum_{i=1}^{n} x_i y_i$	$\frac{\sum_{i=1}^{n} \lvert x_i y_i \rvert}{\lvert s \rvert}$	$n \leqslant \frac{1}{\sqrt{\beta \boldsymbol{k} \mathrm{u}}} - 2$
$\prod_{i=1}^{n} x_i$	1	$n \leqslant \frac{1}{\sqrt{\beta \mathrm{u}}} - 1$
mixed $\times, /$	1	$n \leqslant \frac{1}{2\sqrt{\beta \mathrm{u}}}$
$s := \sum_{i=0}^{n} \frac{\prod_{j \neq i} x - x_j}{\prod_{j \neq i} x_i - x_j} y_i$ $=: \sum_{i=0}^{n} \Theta_i(x)\, y_i$	$\sum_{i=0}^{n} \lvert \Theta_i(x) y_i \rvert / \lvert s \rvert$	$n \leqslant \frac{1}{5}(\beta \boldsymbol{k} \mathrm{u})^{-\frac{1}{2}} - \frac{3}{5}$
$s := \lVert x \rVert_2$	1	$n \leqslant \frac{1}{\sqrt{\beta \mathrm{u}}} - 4$

$$R := \prod_{j=0}^{n} (x - x_j), \quad p(x) = \sum_{i=0}^{n} \frac{R}{(x - x_i) \prod_{j \neq i} (x_i - x_j)} \, y_i.$$

For the last example in Table V, the Euclidean norm of a vector, double-double arithmetic adapted to nonnegative summands is used by Graillat et al. in [11] requiring $13n + 1$ operations compared to $10n + 1$ for our pair arithmetic. They prove, however, that for recursive summation in binary arithmetic the result is faithful for considerably larger maximum vector length $n \leqslant \frac{1}{24\mathrm{u} + \mathrm{u}^2} - 3$. That is useful for binary32, lifting the bound in Table V on n from $2,892$ to $699,047$; for binary64 the bound $n \leqslant 67,108,860$ from Table V may be sufficient.

VI. Faithfully rounded and reproducible results

The results in the previous section are proved to be faithfully rounded for not too large condition number. However, the condition number is, in general, not known. Besides, for small condition number likely the nearest approximation is computed, but not always.

Recently, so-called "reproducible" results became popular. In this section we restrict our attention to summation. Then the true sum is a real number, and reproducibility means to produce exactly the same floating-point approximation no matter what the order of evaluation. That implies addition to become associative, a property which is outside the scope of traditional floating-point algorithms. In addition to "always exactly the same result" we add some not explicitly specified accuracy requirement such as backward stability in a certain sense. Such a requirement is often forgotten in the literature.

For the remaining of this note we assume IEEE-754 p-digit binary arithmetic with the nearest rounding tie-to-even. It is not difficult to extend the methods to general base β.

Following we discuss two approaches producing a faithfully rounded and/or reproducible result, independent of the condition number and guaranteeing associativity. First, a limited exponent range allows to compute the exact sum, for example using a long accumulator as popularized by Kulisch [21], or Malcolm's adding by exponents [27] which is based on Wolfe's approach [42] in 1964. In either case the exact value of the sum can be extracted and rounded faithfully and/or reproducibly.

The second approach splits the bits of a given vector into slices which can be regarded as scaled integers. For example, in binary64 corresponding to 53-bit mantissa (including the implicit 1), suppose each slice is $53 - M$ bits wide. Then at least 2^M numbers [scaled integers representable in at most $53 - M$ bits] within a slice can be added without error, see Figure 1.

This idea is due to Zielke and Drygalla [43] who developed it to improve the accuracy of summation and dot products. No analysis is given, and they use a splitting by integer scaling in a way that the exponent range of the input is severely limited.

In [38] the shortcomings are removed, and a complete analysis is given to achieve a faithfully rounded result. Recently, that method has been used for reproducible summation and popularized by Demmel et al. [7], [8].

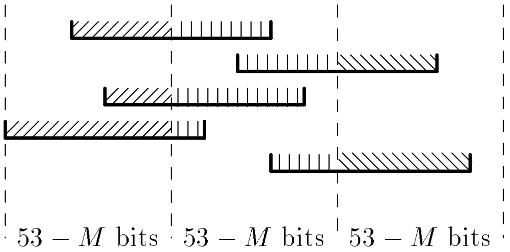

Fig. 1: Splitting bits of a vector by Zielke/Drygalla [43].

The Wolfe/Malcolm methods adjoins each input to a fixed exponent, whereas the slices by Zielke/Drygalla are created depending on the actual input data, a static versus dynamic approach. As a consequence, the first method generates an array of variables over the whole exponent range, independent of the input data, whereas the second method uses just as many slices as necessary to produce a faithfully rounded or reproducible result. In [26] that is used to compute a correctly rounded approximation of a sum with arbitrary precision.

A key point is an efficient way to extract the input into slices [38]. The extraction is relative to some σ so that $p = q + p'$ and the bits of q and p' do not overlap as by `ExtractVector`. A choice for σ is a power of 2 larger than $\sum |p_i|$.

The bit patterns of Algorithm `ExtractVector` is outlined in Figure 2. The main property of the extraction is the error-free transformation $\sum_{i=1}^{n} p_i = \tau + \sum_{i=1}^{n} p'_i$, see [38] for details.

Algorithm 1: Error-free vector transformation extracting high order part.

```
function $[\tau, p'] = \text{ExtractVector}(\sigma, p)$
    $\tau = 0$
    for $i = 1 : n$
        $q_i = \text{fl}(\text{fl}(\sigma + p_i) - \sigma)$
        $p'_i = \text{fl}(p_i - q_i)$
        $\tau = \text{fl}(\tau + q_i)$
    end for
```

Here τ is the error-free sum of the first slice, comprised of the leading bits of the vector entries p_i belonging to this slice [the bold lines in Figure 2]. Note in particular that τ is a sum of scaled integers, thus the computation is associative and error-free.

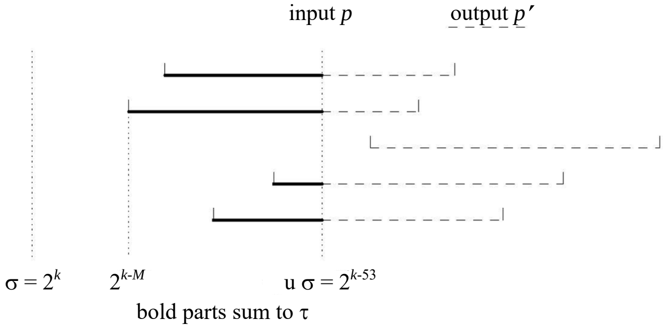

Fig. 2: ExtractVector: error-free transformation $\sum p_i = \tau + \sum p'_i$.

Algorithm 2: Error-free transformation of a vector $p^{(0)}$ of length n.

```
function $[\tau_1, \tau_2, p^{(m)}, \sigma] = \text{Transform}(p^{(0)})$
  $\mu = \max(|p_i^{(0)}|)$
  if $\mu = 0,\ \tau_1 = \tau_2 = p^{(m)} = \sigma = 0,$ return, end if
  $M = \lceil \log_2(n+2) \rceil$
  $\sigma_0 = 2^{M + \lceil \log_2(\mu) \rceil}$
  $t^{(0)} = 0,\ m = 0$
  repeat
      $m = m + 1$
      $[\tau^{(m)}, p^{(m)}] = \text{ExtractVector}(\sigma_{m-1}, p^{(m-1)})$
      $t^{(m)} = \text{fl}(t^{(m-1)} + \tau^{(m)})$
      $\sigma_m = \text{fl}(2^M \mathrm{u} \sigma_{m-1})$
  until  $|t^{(m)}| \geqslant \text{fl}(2^{2M} \mathrm{u} \sigma_{m-1})$  or  $\sigma_{m-1} \leqslant \text{realmin}$
  $\sigma = \sigma_{m-1}$
  $[\tau_1, \tau_2] = \text{FastTwoSum}(t^{(m-1)}, \tau^{(m)})$
```

Next, this process is applied to the vector p iteratively, resulting in Algorithm `Transform`. Here *realmin* denotes the smallest positive normalized floating-point number. In [38] it is shown that this algorithm stops, and for each intermediate

m between 1 and its final value

$$\begin{array}{l} \sum_{i=1}^n p_i^{(0)} = t^{(m-1)} + \tau^{(m)} + \sum_{i=1}^n p_i^{(m)} \quad \text{and} \\ \max|p_i^{(m)}| \leqslant (2^M \mathtt{u})^m \sigma_0 \end{array} \tag{50}$$

is satisfied. Moreover it is shown that, denoting the final $p^{(m)}$ by p', $\text{float}(\tau_1 + (\tau_2 + (\sum_{i=1}^n p_i')))$ is a faithful approximation of the exact sum.

That algorithm offers a convenient way to compute a reproducible result by choosing a fixed maximum number of extractions m and return `res` $= t^{(m-1)} + \tau^{(m)}$, i.e., ignoring the remainder terms $p_i^{(m)}$. By the nature of the algorithm and (50), the quantities $t^{(m-1)}$ and $\tau^{(m)}$ are uniquely determined and, as sums of scaled integers, independent of the order of evaluation in `ExtractVector` and `Transform`.

The quantity σ_0 is of the order $\sum |x_i|$, so that the omitted summands $p_i^{(m)}$ are bounded by about $(n\mathtt{u})^m \sum |x_i|$. Hence for a condition number up to about $\mathtt{u}(n\mathtt{u})^{-m}$ the result `res` is faithful, where for larger condition number the accuracy decreases.

For any value of m the result is reproducible. In binary64 and $n = 1000$, the result is faithful up to condition number $\boldsymbol{k} \leqslant 9 \cdot 10^9$ for $m = 2$, and up to $\boldsymbol{k} \leqslant 8 \cdot 10^{22}$ for $m = 3$.

The equality in (50) implies that for any m the exact sum is available, so a rounded to nearest or K-fold, i.e. an unevaluated sum of K numbers, result can be computed as well [39].

VII. Summary

We first assumed an IEEE-754 p-digit binary floating-point arithmetic, for which the general bound for the first and second standard model is $\mathtt{u}/(1 + \mathtt{u})$ and $\mathtt{u}$, respectively. Optimal bounds were given, in particular improved for division and square root.

Next we considered a general computer arithmetic satisfying the first standard model, where "rounding" mutated to an arbitrary perturbation of the true real result. With the additional Assumption A, that is $|\,(a \boxplus b) - (a + b)\,| \leqslant \min(|a|, |b|)$, standard error estimates for summation, dot products and others of type $\gamma_k = k\mathtt{u}/(1 - k\mathtt{u})$ reduce to $k\mathtt{u}$ for any order of evaluation and without restriction on k. Assumption A is not satisfied for directed or faithful rounding.

For the first standard model together with Assumption B, the same holds true for general perturbations of the true result including directed or faithful rounding with adapted relative rounding error unit $\mathtt{u}$. A mandatory but weak restriction of order $\mathtt{u}^{-1}$ on the number of operations applies.

For the first standard model together with Assumption C instead, the results were extended to replacing γ_h by $h\mathtt{u}$ for h denoting the height of a tree. Again, this is true for any rounding and a mandatory weak restriction on the height.

For a first standard error model together with the additional Assumptions A and B, an optimal error bound $k\mathtt{u}/(1+k\mathtt{u})$ was shown for $k + 1$ summands provided that $k \leqslant \frac{\beta-1}{2}\mathtt{u}^{-1}$. That implies linearized error bounds for other compound operations such as blocked summation or sums of products.

Next we assumed nothing but the first standard model, in particular none of the Assumptions A, B or C. Based on that a pair arithmetic was introduced, simpler and more generally applicable than existing ones. In particular, the often used error-free transformations were weakened by not assuming equality in the transformation. That opens this approach, for example, for IEEE-754 to any base β and any rounding scheme. Sufficient conditions were given that the computed result is a faithfully rounded exact result for arbitrary expressions consisting of $+, -, \times, /, \sqrt{\cdot}$.

Finally, for IEEE-754 binary arithmetic, a summation method was introduced for computing a result that is guaranteed to be faithfully rounded or rounded to nearest independent of the condition number of the sum. The method computes slices of the result and is an efficient way to compute reproducible results.

Acknowledgment

The author is indebted to Claude-Pierre Jeannerod, Marko Lange and Paul Zimmermann for their thorough reading and many valuable comments, including advices on the structure of the paper. Moreover many thanks to Sylvie Boldo, Florian Bünger, Stef Graillat, Christoph Lauter, Jean-Michel Muller and Takeshi Ogita for very fruitful suggestions and remarks.

References

[1] D.H. Bailey. A Fortran-90 based multiprecision system. *ACM Trans. Math. Software*, 21(4):379–387, 1995.

[2] S. Boldo, S. Graillat, and J.-M. Muller. On the robustness of the 2Sum and Fast2Sum algorithms. *ACM Trans. Math. Softw.*, 44(1):1–14, 2017.

[3] S. Boldo and C. Muñoz. Provably faithful evaluation of polynomials. In *Proceedings of the 2006 ACM Symposium on Applied Computing, SAC '06*, 1328–1332, 2006.

[4] R. Brent and P. Zimmermann. *Modern Computer Arithmetic*. Cambridge University Press, New York, NY, USA, 2010.

[5] T.J. Dekker. A floating-point technique for extending the available precision. *Numerische Mathematik*, 18:224–242, 1971.

[6] J. Demmel, I. Dumitriu, O. Holtz, and P. Koev. Accurate and efficient expression evaluation and linear algebra. *Acta Numerica*, 2008:87–145, 2008.

[7] J. Demmel and H.D. Nguyen. Fast reproducible floating-point summation. 163–172. Proc. 21st IEEE Symposium on Computer Arithmetic, Austin, Texas, 2013.

[8] J. Demmel and H.D. Nguyen. Parallel reproducible summation. *IEEE Trans. Comp.*, 64(7):2060–2070, 2015.

[9] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computaing Surveys*, 23(1):5–47, 1991.

[10] S. Graillat. Accurate floating-point product and exponentiation. *IEEE Trans. Comp.*, 58(7):994–1000, 2009.

[11] S. Graillat, C. Lauter, P. Tang, N. Yamanaka, and S. Oishi. Efficient calculations of faithfully rounded l_2-norms of n-vectors. *ACM TOMS*, 41(4):24:1–20, 2015.

[12] S. Graillat, V. Lefèvre, and J.-M. Muller. On the maximum relative error when computing integer powers by iterated multiplications in floating-point arithmetic. *Numerical Algorithms*, 70(3):653–667, 2015.

[13] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM Publications, Philadelphia, 2nd edition, 2002.

[14] IEEE, New York. *ANSI/IEEE 754-2008: IEEE Standard for Floating-Point Arithmetic*, 2008.

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

[16] C.-P. Jeannerod and S.M. Rump. On relative errors of floating-point operations: optimal bounds and applications. *Math. Comp.*, 87:803–819, 2017.

[17] C.-P. Jeannerod, J.-M. Muller, and P. Zimmermann. On various ways to split a floating-point number. In *ARITH 2018 - 25th IEEE Symposium on Computer Arithmetic*, pages 53–60. IEEE, June 2018.

[18] M. Joldes, J.-M. Muller, and V. Popescu. Tight and rigorous error bounds for basic building blocks of double-word arithmetic. *ACM Trans. Math. Softw.*, 44(2):1–27, 2017.

[19] W. M. Kahan. A survey of error analysis. In *Proceedings of the IFIP Congress, Ljubljana*, Information Processing 71, 1214–1239. North–Holland, Amsterdam, 1972.

[20] D.E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*, volume 2. Addison Wesley, Reading, Massachusetts, 3rd edition, 1998.

[21] U. Kulisch and W.L. Miranker. *Computer Arithmetic in Theory and Practice*. Academic Press, New York, 1981.

[22] M. Lange and S.M. Rump. Error estimates for the summation of real numbers with application to floating-point summation. *BIT*, 57:927–941, 2017.

[23] M. Lange and S.M. Rump. Faithfully rounded floating-point operations. *ACM Trans. Math. Softw.*, to appear, 2019.

[24] M. Lange and S.M. Rump. Sharp estimates for perturbation errors in summations. *Math.Comp.*, 88:349–368, 2019.

[25] P. Langlois and N. Louvet. More instruction aevel parallelism explains the actual efficiency of compensated algorithms. Technical report, 2007. `https://hal.archives-ouvertes.fr/hal-00165020`.

[26] V. Lefèvre. Correctly rounded arbitrary-precision floating-point summation. *IEEE Transactions on Computers*, 66(12):14, 2017.

[27] M. Malcolm. On accurate floating-point summation. *Comm. ACM*, 14(11):731–736, 1971.

[28] W.F. Mascarenhas. Floating-point numbers are real numbers. ArXi:1605:09202, 2016.

[29] J.-M. Muller, N. Brunie, F. de Dinechin, C.-P. Jeannerod, M. Joldes, V. Lefèvre, G. Melquiond, N. Revol, and S. Torres. *Handbook of Floating-Point Arithmetic*. Birkhäuser Boston, 2nd edition, 2018.

[30] A. Neumaier. Rundungsfehleranalyse einiger Verfahren zur Summation endlicher Summen. *Zeitschrift für Angew. Math. Mech. (ZAMM)*, 54:39–51, 1974.

[31] T. Ogita, S.M. Rump, and S. Oishi. Accurate sum and dot product. *SIAM Journal on Scientific Computing (SISC)*, 26(6):1955–1988, 2005.

[32] K. Ozaki, T. Ogita, F. Bünger, and S. Oishi. Accelerating interval matrix multiplication by mixed precision arithmetic. *Nonlinear Theory and Its Applications, IEICE*, 6(3):364–376, 2015.

[33] D.M. Priest. *On properties of floating-point arithmetics: numerical stability and the cost of accurate computations*. PhD thesis, Mathematics Department, University of California at Berkeley, CA, 1992. `ftp://ftp.icsi.berkeley.edu/pub/theory/priest-thesis.ps.Z`.

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

[35] S.M. Rump, F. Bünger, and C.-P. Jeannerod. Improved error bounds for floating-point products and Horner's scheme. *BIT*, 56(1):293–307, 2015.

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

[37] S.M. Rump and M. Lange. On the definition of unit roundoff. *BIT*, 56(1):309–317, 2015.

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

[39] S.M. Rump, T. Ogita, and S. Oishi. Accurate floating-point summation part II: sign, K-fold faithful and rounding to nearest. *Siam J. Sci. Comput. (SISC)*, 31(2):1269–1302, 2008.

[40] J.R. Shewchuk. Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete Comput. Geom.*, 18(3):305–363, 1997.

[41] J.H. Wilkinson. Error analysis of floating-point computation. *Numerische Mathematik*, 2:319–340, 1960.

[42] J.M. Wolfe. Reducing truncation errors by programming. *Comm. ACM*, 7(6):355–356, 1964.

[43] G. Zielke and V. Drygalla. Genaue Lösung linearer Gleichungssysteme. *GAMM Mitt. Ges. Angew. Math. Mech.*, 26:7–108, 2003.