Numerische Mathematik (2020) 145:383–403
https://doi.org/10.1007/s00211-020-01114-2

Numerische
Mathematik

A note on Dekker's FastTwoSum algorithm

Marko Lange[2] · Shin'ichi Oishi[1]

Received: 13 December 2017 / Revised: 16 March 2020 / Published online: 24 April 2020

Abstract
More than 45 years ago, Dekker proved that it is possible to evaluate the exact error of a floating-point sum with only two additional floating-point operations, provided certain conditions are met. Today the respective algorithm for transforming a sum into its floating-point approximation and the corresponding error is widely referred to as FastTwoSum. Besides some assumptions on the floating-point system itself—all of which are satisfied by any binary IEEE 754 standard conform arithmetic, the main practical limitation of FastTwoSum is that the summands have to be ordered according to their exponents. In most preceding applications of FastTwoSum, however, a more stringent condition is used, namely that the summands have to be sorted according to their absolute value. In remembrance of Dekker's work, this note reminds the original assumptions for an error-free transformation via FastTwoSum. Moreover, we generalize the conditions for arbitrary bases and discuss a possible modification of the FastTwoSum algorithm to extend its applicability even further. Subsequently, a range of programs exploiting the wider applicability is presented. This comprises the OnlineExactSum algorithm by Zhu and Hayes, an error-free transformation from a product of three floating-point numbers to a sum of the same number of addends, and an algorithm for accurate summation proposed by Demmel and Hida.

Mathematics Subject Classification 65G99 · 65Y99

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

✉ Marko Lange
m.lange@tuhh.de

Shin'ichi Oishi
oishi@waseda.jp

[1] Faculty of Science and Engineering, Waseda University, 3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, Japan

[2] Institute for Reliable Computing, Hamburg University of Technology, Am Schwarzenberg-Campus 3, 21073 Hamburg, Germany

Contents

1 Introduction and notation

A floating-point number system with base β, mantissa length p, and exponent range $[\mathrm{e}_{\min}, \mathrm{e}_{\max}]$ may be defined via

$$\mathbb{F} := \{m \cdot \beta^e : m, e \in \mathbb{Z}, -\beta^p < m < \beta^p, \mathrm{e}_{\min} \le e \le \mathrm{e}_{\max}\}. \tag{1}$$

Let $\mathbb{F}$ be accompanied by a set of floating-point operations $\{\oplus, \ominus, \odot, \ldots\}$ that approximate their real equivalents $\{+, -, \cdot, \ldots\}$ in accordance to some mapping $\mathrm{fl}\colon \mathbb{R} \to \mathbb{F}$. More specifically, $x \circledcirc y = \mathrm{fl}(x \circ y)$ for all $x, y \in \mathbb{F}$, where $\circ$ can be any supported operation between two numbers. If the mapping $\mathrm{fl}(\cdot)$ is conform with a rounding from the IEEE 754 floating-point standard, we obtain a model for an arithmetic that is in line with the same standard.

If not stated otherwise, we henceforth assume the operations on $\mathbb{F}$ to be evaluated in *rounding to nearest*, i.e., the mapping $\mathrm{fl}\colon \mathbb{R} \to \mathbb{F}$ satisfies

$$\forall r \in \mathbb{R}, f \in \mathbb{F}\colon \quad |\mathrm{fl}(r) - r| \le |f - r|.$$

For instance, $\oplus\colon \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ is called *nearest-addition* if it approximates the addition over real numbers by a nearest number within $\mathbb{F}$, that is,

$$\forall x, y \in \mathbb{F}\colon \quad |(x \oplus y) - (x + y)| = \min\{|f - (x + y)|\colon f \in \mathbb{F}\}.$$

Another frequently considered assumption on floating-point approximations is *faithful rounding*. We call an operation faithfully rounded if there lies no other floating-point number between the rounded and the real result.

The FastTwoSum procedure first appeared in 1965 as a part of Kahan's compensated summation algorithm [6]. Kahan introduced his algorithm as a simpler alternative to Wolfe's summation method, which is based on cascaded accumulators [19]. However, Kahan neither provided an error estimate for his algorithm nor gave the conditions for an error-free transformation.

```
Function $(s, e) \leftarrow \mathrm{FastTwoSum}(x, y)$
$s \leftarrow x \oplus y$
$t \leftarrow s \ominus x$
$e \leftarrow y \ominus t$
```

The introduction of the FastTwoSum algorithm as a technique for extending the available floating-point precision as well as the proof of its error-free transformation property in 1971 is due to Dekker [2]. The modern term FastTwoSum was likely coined by Shewchuk [16]. Occasionally, this algorithm is also referred to as Quick-Two-Sum.

Let $\mathrm{e}(f) = e_f$ denote the exponent of $f \in \mathbb{F}$ according to a representation $f = m_f \cdot \beta^{e_f}$ that complies with definition (1). Dekker proved the following relation between the input and the return values of FastTwoSum.

Theorem 1 *Let $x, y \in \mathbb{F}$ with the base of $\mathbb{F}$ being restricted to $\beta \in \{2, 3\}$. If $\oplus$ realizes a nearest-addition, $\ominus$ realizes some faithful-subtraction, and*

$$\mathrm{e}(x) \geq \mathrm{e}(y), \tag{2}$$

then

$$s + e = x + y \quad \textit{with} \quad s = x \oplus y. \tag{3}$$

Furthermore, in [2] properly truncated addition[1] was considered, for which Dekker proved that (3) holds true without the restriction on β. Nevertheless, due to the absence of fast hardware implementations of properly truncated rounding, this result is of rather theoretical interest and typically disregarded.

Another rarely considered property of Dekker's theorem emerges from his definition of an exponent $\mathrm{e}(x)$ to a floating-point number $x \in \mathbb{F}$. Since the original inequality $\mathrm{e}(x) \geq \mathrm{e}(y)$ is difficult to check, usually only the more stringent condition $|x| \geq |y|$ is regarded. In the following section we will take a closer look at the generality of the former inequality and generalize Dekker's result for arbitrary bases β. Subsequently, a range of applications will be presented for which the transformation by FastTwoSum is error-free without the usually considered inequality $|x| \geq |y|$ being met.

2 Multiple representations

From a mathematical perspective – and maybe also reasoned in the wide-spread usage of the IEEE 754 standard –, we typically connect the exponent of a floating-point number with the exponent to a normalized representation of the same. This may be a major reason for leaving a wider applicability of the FastTwoSum algorithm unrecognized.

Definition (1) allows multiple representations for many of its elements. Exemplary, let $\beta = 2$, $p = 4$ and consider the number $x = 3$. Supposing a sufficiently wide range of feasible exponents, there are three presentations

$$x = 12 \cdot \beta^{-2} = 6 \cdot \beta^{-1} = 3 \cdot \beta^{0}$$

that comply with (1). Hence, $\mathrm{e}(x)$ can be any integer from the set $\{-2, -1, 0\}$. In [2], Dekker simply assumes that there are feasible representations of $x, y \in \mathbb{F}$ for which $\mathrm{e}(x) \geq \mathrm{e}(y)$ is satisfied.

[1] An addition is properly truncated if the rounding is faithful and the approximation error, if existent, has the same sign as the summand with smaller absolute value.

Using the notation of the *unit in the last place* (ULP), it is possible to give an equivalent, more explicit condition than the one due to Dekker. The ULP is defined for real numbers $r \in (-\beta^{\mathrm{e}_{\max}+p},\ \beta^{\mathrm{e}_{\max}+p})$ as

$$\operatorname{ulp}(r) := \min\{g - f\colon\ f, g \in \mathbb{F} \cup \{\beta^{\mathrm{e}_{\max}+p}\},\ f \leq |r| < g\}. \tag{4}$$

Hence, the unit in the last place of a nonnegative number $x \in \mathbb{F}$ is the step-length to its successor.

Lemma 1 *Inequality (2) is satisfied for some representation of $x, y \in \mathbb{F}$ complying with (1) if, and only if,*

$$\exists k \in \mathbb{Z}\colon\ x = k \operatorname{ulp}(y). \tag{5}$$

Proof Let $e_x^{\max}$ denote the maximal exponent over all feasible representations of x, and let $e_y^{\min}$ denote the minimal exponent of y, accordingly. Then an equivalent condition to (2) is $e_x^{\max} \geq e_y^{\min}$. In the trivial case $x = 0$, it is $e_x^{\max} = \mathrm{e}_{\max}$ and the equivalence with (5) is evident. We henceforth assume $x \neq 0$. By (1) and (4), we have $\beta^{e_y^{\min}} = \operatorname{ulp}(y)$. Another consequence of (1) is that $\beta^{e_x^{\max}}$ denotes the maximal power of β that divides x. Since $x\beta^{-e_x^{\max}} \in \mathbb{Z}$ but $x\beta^{-e_x^{\max}-1} \notin \mathbb{Z}$,

$$\frac{x}{\operatorname{ulp}(y)} = x\beta^{-e_x^{\max}}\beta^{(e_x^{\max}-e_y^{\min})}$$

lies in $\mathbb{Z}$ if, and only if, $e_x^{\max} \geq e_y^{\min}$. □

It is noteworthy that, in [15], the sufficiency of condition (5) was already proved for $\beta = 2$ and rounding to nearest in every operation. However, neither was [15, Lemma 3] linked to Dekker's original condition nor has the result been exploited for any of the applications given in Section 4.

3 Generalization for arbitrary bases

In the previous section it was recalled that Dekker's result is applicable to more general constellations than $x, y \in \mathbb{F}$ with $|x| \geq |y|$. Before discussing applications where weaker presupposition are beneficial, here we discuss a possible generalization of Dekker's result for arbitrary bases β.

A typical example to pinpoint the necessity of the restriction $\beta \in \{2, 3\}$ in Theorem 1 is

$$x = 99,\ y = 98 \quad \text{with} \quad \beta = 10,\ p = 2, \tag{6}$$

for which FastTwoSum returns $s = 200$ and $e \in \{-12, -2\}$. The ambiguity of e is due to the faithful evaluation of $t \leftarrow s \ominus x$. In either case the identity $s + e = x + y$ is not satisfied. Apparently, in the context of floating-point systems with larger bases, presupposition (2) is not sufficient to ensure (3).

In the following we give an alternative result that covers Dekker's Theorem as a special case.

Theorem 2 *Consider the* FastTwoSum *algorithm for given input* $x, y \in \mathbb{F}$. *Let* $\oplus$ *and* $\ominus$ *realize a nearest-addition and some faithful-subtraction, respectively. If there is a representation of* x *such that*

$$|y| \le \left\lceil \beta^p - \frac{\beta}{2} \right\rceil \beta^{\mathrm{e}(x)}, \tag{7}$$

then the computed $s, e \in \mathbb{F}$ *satisfy (3).*

Proof As a consequence of (7), clearly (2) is satisfiable. Moreover, by definition (1) the difference of two floating-point numbers a, b is a multiple of $\beta^{\min\{\mathrm{e}(a),\mathrm{e}(b)\}}$ such that, in the absence of overflow,

$$|a - b| \le \beta^p \beta^{\min\{\mathrm{e}(a),\mathrm{e}(b)\}} \quad \Longrightarrow \quad a \ominus b = a - b. \tag{8}$$

A similar statement applies to the addition of two floating-point numbers.

We use inequality (2) and implication (8) to prove (3), first verifying the equality $t = s - x$ and then $e = y - t$. The proof of the former is by distinction into three cases.

Case 1 Assume that $\mathrm{e}(x) = \mathrm{e}(y)$. Definition (1) and condition (7) imply

$$|s - (x + y)| \le \frac{\mathrm{ulp}(x + y)}{2} \le \frac{\mathrm{ulp}(\beta^p \beta^{\mathrm{e}(x)} + \lceil \beta^p - \frac{\beta}{2} \rceil \beta^{\mathrm{e}(x)})}{2} = \frac{\beta}{2} \beta^{\mathrm{e}(x)}.$$

Since x, y, and s are necessarily multiples of $\beta^{\mathrm{e}(x)} = \beta^{\mathrm{e}(y)}$, we have

$$|s - x| \le |s - (x + y)| + |y| \le \left\lfloor \frac{\beta}{2} \right\rfloor \beta^{\mathrm{e}(x)} + \left\lceil \beta^p - \frac{\beta}{2} \right\rceil \beta^{\mathrm{e}(x)} = \beta^p \beta^{\min\{\mathrm{e}(x),\mathrm{e}(s)\}}$$

for some representation of s. Then implication (8) yields $t = s - x$.

Case 2 Suppose that $|x + y| \le \beta^p \beta^{\mathrm{e}(y)}$ is satisfied for all feasible representations of y. By (8) we have $s = x + y$ and thereby $t = s - x = y \in \mathbb{F}$.

Case 3 Complimentary to the previous cases, taking (2) into account, assume that $\mathrm{e}(x) > \mathrm{e}(y)$ and $|x + y| > \beta^p \beta^{\mathrm{e}(y)}$ are satisfiable. The latter implies $|s| \ge \beta^p \beta^{\mathrm{e}(y)}$ such that $\min\{\mathrm{e}(s), \mathrm{e}(x)\} > \mathrm{e}(y)$. Moreover, nearest-addition in line 1 of FastTwoSum and $x \in \mathbb{F}$ imply

$$|s - (x + y)| = \min\{|f - (x + y)| : \ f \in \mathbb{F}\} \le |x - (x + y)| = |y| < \beta^p \beta^{\mathrm{e}(y)}, \tag{9}$$

so that, using $2 \le \beta$,

$$|s - x| \le |s - (x + y)| + |y| \le \beta^p \beta^{\mathrm{e}(y)} + |y| < \beta \beta^p \beta^{\mathrm{e}(y)} \le \beta^p \beta^{\min\{\mathrm{e}(s),\mathrm{e}(x)\}}.$$

Yet again (8) yields $t = s - x$.

It remains to prove the equality $e = y - t$. Using the satisfiability of

$$\mathrm{e}(t) \ge \min\{\mathrm{e}(s), \mathrm{e}(x)\} \ge \min\{\mathrm{e}(y), \mathrm{e}(x)\} = \mathrm{e}(y)$$

together with $t = s - x$ and (9), we show that

$$|y - t| = |s - (x + y)| \leq \beta^p \beta^{\mathrm{e}(y)} = \beta^p \beta^{\min\{\mathrm{e}(y),\mathrm{e}(t)\}}$$

and validate $e = y - t$. □

For $\beta \in \{2, 3\}$ there is no number in $\mathbb{F}$ larger than $\lceil \beta^p - \frac{\beta}{2} \rceil \beta^{\mathrm{e}(x)}$ but also smaller than $\beta^p \beta^{\mathrm{e}(x)}$. It is thus straightforward to show that condition (7) and Dekker's original condition $\mathrm{e}(x) \geq \mathrm{e}(y)$ are equivalent for base $\beta \in \{2, 3\}$.

In the proof of Theorem 2, we are not so much concerned with the sum $x \oplus y$ being evaluated in rounding to nearest. The only property of nearest-addition that we make use of is given in (9). In particular, for any mapping $\oplus \colon \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ satisfying

$$|x \oplus y - (x + y)| \leq |y|, \tag{10}$$

the inequality in (7) can be adapted so that (3) remains valid.

Unfortunately, (10) is not necessarily satisfied for a faithfully rounded summation. If y is smaller than half the distance between x and its nearest floating-point neighbor but $x \oplus y \neq x$, then (10) does not hold true and typically $x \oplus y - (x + y) \notin \mathbb{F}$. On the other hand, as long as the exponents of x and y are not too far apart, we can prove that also the rounding error of any faithfully rounded sum $x \oplus y$ necessarily lies in $\mathbb{F}$. To be precise, one can show the following:

Remark 1 If condition (7) in Theorem 2 is replaced with

$$|y| \leq (\beta^p - \beta + 1)\beta^{\mathrm{e}(x)} \quad \text{and} \quad |x| \leq (\beta^p - 1)\beta^p \beta^{\mathrm{e}(y)}, \tag{11}$$

then (3) is true also for faithfully rounded addition in line 1 of FastTwoSum.

Proof We use a similar argument as for Theorem 2. The proof has to be modified in two places:

In Case 1, faithful rounding only implies $|s - (x + y)| < \mathrm{ulp}(x + y) \leq \beta\beta^{\mathrm{e}(x)}$ without the factor $\frac{1}{2}$. Nevertheless, by $y, x, s \in \beta^{\mathrm{e}(x)}\mathbb{Z}$ and the tighter bound on $|y|$ given in (11), we still have

$$|s - x| \leq |s - (x + y)| + |y| \leq (\beta - 1)\beta^{\mathrm{e}(x)} + (\beta^p - \beta + 1)\beta^{\mathrm{e}(x)} = \beta^p \beta^{\min\{\mathrm{e}(x),\mathrm{e}(s)\}}.$$

Also the argument in (9) is no more applicable for faithful rounding. Nevertheless, $|s - (x + y)| \leq \beta^p \beta^{\mathrm{e}(y)}$ still holds valid and can be shown as follows. The right inequality in (11) is equivalent to $\mathrm{ulp}(x) \leq \beta^p \beta^{\mathrm{e}(y)}$ by which

$$|s - (x + y)| \leq \mathrm{ulp}(x) \quad \Longrightarrow \quad |s - (x + y)| \leq \beta^p \beta^{\mathrm{e}(y)}.$$

On the other hand, if $|s - (x + y)| > \mathrm{ulp}(x)$, then $\mathrm{ulp}(x + y) > \mathrm{ulp}(x)$ and therefore $|y| = |x + y| - |x| \geq \beta^{-1} \mathrm{ulp}(x + y)$. Hence

$$|s - (x + y)| \leq \mathrm{ulp}(x + y) \leq \beta^p \,\mathrm{ulp}(\beta^{-1} \mathrm{ulp}(x + y)) \leq \beta^p \,\mathrm{ulp}(y) \leq \beta^p \beta^{\mathrm{e}(y)}$$

proves that the outer inequality in (9) remains valid. □

It is noteworthy that for $\beta = 2$ the right inequality in (11) is again equivalent to (2). Since most modern computers implement IEEE 754 binary floating-point formats, generalizations of certain FastTwoSum applications on theses platforms are straightforward.

Theorem 2 and Remark 1 pinpoint the actual conditions under which the transformation due to FastTwoSum is error-free. Though these conditions do not directly restrict the base β, the limitation illustrated in (6) remains.

To overcome this issue, we need to modify the original algorithm. Define the constant $c_\beta := \lceil \beta^p - \frac{\beta-2}{2} \rceil \beta^{-p}$ and assume $c_\beta \in \mathbb{F}$. The condition $c_\beta \in \mathbb{F}$ necessarily holds valid if the normal range of $\mathbb{F}$ encompasses $(\beta^{-1}, 1]$, which is a reasonable assumption. We extend the code of FastTwoSum as follows.

```
Function $(s, e) \leftarrow c_\beta$-FastTwoSum$(x, y)$
$\check{y} \leftarrow c_\beta \odot y$
$s \leftarrow x \oplus \tilde{y}$
$t \leftarrow s \ominus x$
$e \leftarrow y \ominus t$
```

In return for an additional operation and loosing the beneficial property $s = x \oplus y$, c_β-FastTwoSum allows an error-free transformation of any pair $x, y \in \mathbb{F}$ satisfying $\mathrm{e}(x) \geq \mathrm{e}(y)$, independent of the choice of β.

Theorem 3 *Consider the* c_β-FastTwoSum algorithm for given input $x, y \in \mathbb{F}$ with $c_\beta := \lceil \beta^p - \frac{\beta-2}{2} \rceil \beta^{-p} \in \mathbb{F}$. Let $\odot$, $\oplus$, and $\ominus$ realize a nearest-multiplication, a nearest-addition, and some faithful-subtraction, respectively. If x is a multiple of $\mathrm{ulp}(y)$, i.e., condition (2) is satisfiable, then $s, e \in \mathbb{F}$ satisfy

$$s + e = x + y \quad \text{with} \quad |e| \leq \frac{1}{2}\,\mathrm{ulp}(x + \tilde{y}) + \left\lfloor \frac{\beta - 2}{2} \right\rfloor \mathrm{ufp}(y). \tag{12}$$

Proof To avoid a separate argument for the underflow case, we exploit the notation of the *unit in the first place* (ufp) introduced in [14]:

$$\mathrm{ufp}(a) := \begin{cases} 0 & \text{if } a = 0, \\ \beta^{\lfloor \log_\beta(|a|) \rfloor} & \text{otherwise.} \end{cases}$$

Certain equalities, such as $\mathrm{ufp}(a) = \beta^{p-1}\,\mathrm{ulp}(a)$, are only valid for numbers in the normalized range of $\mathbb{F}$. In the following argument, we only use relations that are also satisfied in the underflow case, for instance, $\mathrm{ufp}(a) \leq \beta^{p-1}\,\mathrm{ulp}(a)$.

By definition of c_β and $|y| \leq (\beta^p - 1)\beta^{\mathrm{e}(y)}$, we have

$$c_\beta \cdot |y| \leq \left\lceil \beta^p - \frac{\beta - 2}{2} \right\rceil \beta^{-p} \cdot (\beta^p - 1)\beta^{\mathrm{e}(y)}$$

$$= \left(\left\lceil \beta^p - \frac{\beta}{2} \right\rceil + \left\lfloor \frac{\beta}{2} \right\rfloor \beta^{-p} - \beta^{-p} \right) \beta^{\mathrm{e}(y)}$$
$$\leq \left(\left\lceil \beta^p - \frac{\beta}{2} \right\rceil + \frac{1}{2} - \beta^{-p} \right) \beta^{\mathrm{e}(y)}.$$

Since $\lceil \beta^p - \frac{\beta}{2} \rceil \beta^{\mathrm{e}(y)}$ is the unique nearest floating-point number to this upper bound and $\mathrm{e}(x) \geq \mathrm{e}(y)$, we have

$$|\tilde{y}| = |\mathrm{fl}(c_\beta \cdot y)| \leq \left\lceil \beta^p - \frac{\beta}{2} \right\rceil \beta^{\mathrm{e}(x)}.$$

Hence x, $\tilde{y}$ are in accordance with (7) and one can exploit the first part of the proof of Theorem 2 to show that $t = s - x$.

To prove the equality $e = y - t$, it is necessary to modify the respective argument. If $|y| \neq \mathrm{ufp}(y)$, then $|y| \geq (1 + \beta^{1-p}) \, \mathrm{ufp}(y)$ and

$$c_\beta \cdot |y| \geq \left(\beta^p - \frac{\beta}{2} \right) \beta^{-p} \cdot (1 + \beta^{1-p}) \, \mathrm{ufp}(y) = \left(1 + \frac{1 - \beta^{1-p}}{2\beta^{p-1}} \right) \mathrm{ufp}(y) \geq \mathrm{ufp}(y),$$

so that $|y| < \beta \, \mathrm{ufp}(y) = \beta \, \mathrm{ufp}(c_\beta \cdot y) \leq \beta^p \, \mathrm{ulp}(\tilde{y})$. On the other hand, for $|y| = \mathrm{ufp}(y)$, we have $\mathrm{ufp}(y) \leq \beta \, \mathrm{ufp}(c_\beta \cdot y)$ and once again $|y| \leq \beta^p \, \mathrm{ulp}(\tilde{y})$. Moreover, by $t = s - x$ and a similar argument as in (9), we derive

$$|y - t| = |y - \tilde{y} - s + x + \tilde{y}| \leq |y - \tilde{y}| + |s - (x + \tilde{y})| \leq |y - \tilde{y}| + |\tilde{y}| = |y|.$$

Together with

$$\mathrm{ulp}(\tilde{y}) \leq \beta^{\min\{\mathrm{e}(\tilde{y}), \mathrm{e}(y), \mathrm{e}(x)\}} \leq \beta^{\min\{\mathrm{e}(y), \mathrm{e}(y - ((x \oplus \tilde{y}) - x))\}} = \beta^{\min\{\mathrm{e}(y), \mathrm{e}(t)\}}$$

for some representation of t, this yields

$$|y - t| \leq |y| \leq \beta^p \, \mathrm{ulp}(\tilde{y}) \leq \beta^p \beta^{\min\{\mathrm{e}(t), \mathrm{e}(y)\}}.$$

By implication (8), we then prove $e = y - t = y - (s - x)$.

Finally, $|y| - \lfloor \frac{\beta - 2}{2} \rfloor \mathrm{ulp}(y) \in \mathbb{F}$, $|y| \in \mathbb{F}$, and

$$|y| - \left\lfloor \frac{\beta - 2}{2} \right\rfloor \mathrm{ulp}(y) = |y| - (1 - c_\beta) \beta^p \, \mathrm{ulp}(y) < |y| - (1 - c_\beta)|y| = |c_\beta y| \leq |y|$$

imply $|\tilde{y} - y| \leq \lfloor \frac{\beta - 2}{2} \rfloor \mathrm{ulp}(y)$, such that

$$|e| = |s - (x + y)| \leq |s - (x + \tilde{y})| + |\tilde{y} - y| \leq \frac{1}{2} \mathrm{ulp}(x + \tilde{y}) + \left\lfloor \frac{\beta - 2}{2} \right\rfloor \mathrm{ulp}(y),$$

which completes the argument. □

For $\beta \in \{2, 3\}$ the scaling factor is $c_\beta = 1$ and c_β-FastTwoSum works just like the original implementation. Our code demonstrates a possible generalization of Dekker's FastTwoSum algorithm.

The above approach can be generalized further for faithful-addition as in Remark 1. For this purpose, we just need to redefine the scaling factor $c_\beta := 1 - \beta^{1-p} + \beta^{-p}$, assume $p \geq 2$, and adapt the error estimate together with the respective argument. The statement remains true if the product $c_\beta \odot y$ is rounded faithfully. We leave the analysis to the well-disposed reader.

If embedded or low-level programming is used, alternative approaches to compute a suitable $\tilde{y}$ are available. One possibility is to round y towards zero into a floating-point format with same base and exponent range as $\mathbb{F}$ but a by 1 reduced mantissa length $p-1$. Such a rounding is easily implemented by resetting the last mantissa bit in a normalized representation of the respective number. Also $\tilde{y} \leftarrow \text{sign}(y) \cdot \min\{|y|, (\beta^p - \beta)\beta^{\mathrm{e}(y)}\}$ is an option that can be realized efficiently by applying suitable integer operations solely to the mantissa bits of y. For the sake of clarity and transparency, here we refrain from going any further into detail.

4 Applications

The examples in the following subsections serve to illustrate a wider applicability of the FastTwoSum algorithm. We follow the same notation as above. In accordance with (1), p shall denote the mantissa length, β denotes the base, and $\mathrm{e}_{\min}, \mathrm{e}_{\max}$ define the feasible range of exponents. Unless otherwise specified, we assume that the arithmetic operations are evaluated in rounding to nearest. Moreover, we generally assume the absence of overflow. All other assumptions, including possible restriction on the base β as well as exceptions for underflow, are explicitly mentioned for each case individually if present.

4.1 Error-free transformation - single exponent summation

As an immediate application of Dekker's original theorem, let us consider the recursive summation of floating-point numbers with the same ULP. Since all intermediate sums are multiples of this ULP, these numbers may be added accurately using FastTwoSum. The respective error term can be summed up without introducing any errors by applying plain floating-point addition; at least until the error grows above β^p times the respective ULP.

To prove that the transformation due to Algorithm 1 is error-free for a limited number of summands, we first show the following two auxiliary results.

Lemma 2 *For given numbers $s, x \in \mathbb{F}$, choose $e, l_s, l_x, u_s, u_x \in \mathbb{Z}$ such that*

$$l_s \beta^e \leq s \leq u_s \beta^e \quad \textit{and} \quad l_x \beta^e \leq x \leq u_x \beta^e.$$

Algorithm 1: Summation into two addends

```
Data: $x_1, x_2, \dots, x_n \in \mathbb{F}$, where $n \geq 2$ and $\forall i, j: \frac{x_i}{\mathrm{ulp}(x_j)} \in \mathbb{Z}$
Result: $s, e \in \mathbb{F}$
$(s, e) \leftarrow \mathrm{FastTwoSum}(x_1, x_2)$                // initial summation
for $i \leftarrow 3$ to $n$ do
    $(s, t) \leftarrow \mathrm{FastTwoSum}(s, x_i)$
    $e \leftarrow e \oplus t$
end
```

If $s \oplus x$ is rounded faithfully, then

$$|l_s + l_x| \leq \beta^p \implies (l_s + l_x)\beta^e \leq s \oplus x, \tag{13a}$$

$$|u_s + u_x| \leq \beta^p \implies s \oplus x \leq (u_s + u_x)\beta^e. \tag{13b}$$

Proof The left-hand side of (13a) implies $|l_s + l_x|\beta^e \leq \beta^{p+e}$, such that $(l_s + l_x)\beta^e$ either lies in the underflow range of $\mathbb{F}$ or is itself a floating-point number.[2] In the former case, the result is evident due to error-free summation in the underflow range. On the other hand, for $(l_s + l_x)\beta^e \in \mathbb{F}$, faithfully rounded evaluation and $(l_s + l_x)\beta^e \leq s + x$ imply $(l_s + l_x)\beta^e \leq s \oplus x$. The implication (13b) can be shown by a similar argument. □

Lemma 3 *For given $x_0, \dots, x_n \in \mathbb{F}$, let $e, k, l, u \in \mathbb{Z}$ satisfy*

$$l\beta^e \leq x_0 \leq u\beta^e \quad \textit{and} \quad \forall i \in \{1, \dots, n\}\colon\ |x_i| \leq k\beta^e.$$

If s_n denotes the result of $\sum_{i=0}^{n} x_i$ evaluated faithfully and in any order, then

$$\max\{l - k,\ nk,\ nk - l\} \leq \beta^p \implies (l - nk)\beta^e \leq s_n, \tag{14a}$$

$$\max\{-u - k,\ nk,\ u + nk\} \leq \beta^p \implies s_n \leq (u + nk)\beta^e. \tag{14b}$$

Lemma 3 can be proved by a simple induction argument using Lemma 2. We exploit this result to show the desired behavior of Algorithm 1.

Corollary 1 *Let given $x_1, x_2, \dots, x_n \in \mathbb{F}$ with $\beta \in \{2, 3\}$ satisfy $\frac{x_i}{\mathrm{ulp}(x_j)} \in \mathbb{Z}$ for all index pairs $1 \leq i, j \leq n$. If*

$$n \leq \frac{\beta^q}{\beta + 1} + 2\beta^{p-q} \quad \textit{with} \quad q := \left\lfloor \frac{p + 2 + \log_\beta 2}{2} \right\rfloor, \tag{15}$$

then Algorithm 1 transforms $\sum_{i=1}^{n} x_i$ error-free into $s + e$.

[2] The overflow case is disregarded by our general assumptions.

Remark 2 The statement in Corollary 1 remains true for faithful-addition if $\beta = 2$ and the restriction on n in (15) is replaced by

$$n \leq \frac{\beta^{q_f}}{\beta+1} + \beta^{p-q_f} \quad \text{with} \quad q_f := \left\lfloor \frac{p+2}{2} \right\rfloor.$$

Remark 3 Moreover, the transformation by Algorithm 1 remains error-free without the restriction on β but with rounding to nearest if we assume

$$n \leq \frac{\beta^q}{\beta+1} + 2\beta^{p-q} - \frac{6}{\beta}\left\lfloor \frac{\beta-2}{2} \right\rfloor$$

and replace the FastTwoSum calls with their c_β-FastTwoSum equivalents.

Proof The initial assumption on the addends x_i imply a similar property for the intermediate values s_j of s, i.e.,

$$\forall j,k\colon\ \frac{x_j}{\mathrm{ulp}(x_k)} \in \mathbb{Z} \implies \frac{\sum_{i=1}^{j} x_i}{\mathrm{ulp}(x_k)} \in \mathbb{Z} \implies \frac{s_j}{\mathrm{ulp}(x_k)} \in \mathbb{Z}.$$

Thus, the requirements for an error-free transformation are met for each call of FastTwoSum. It remains to show that the summation of the error terms does not involve further rounding errors.

In each call of FastTwoSum the variable s is updated simply by adding the respective summand x_i. Hence, $s_i := s_{i-1} \oplus x_i$ for $i = 2, \ldots, n$. Let k be the index of the summand with maximum absolute value, such that

$$\forall i\colon\ |x_i| \leq |x_k| < \beta^p \,\mathrm{ulp}(x_k).$$

Since this upper bound is a power of β, we can apply Lemma 3 to show that

$$|s_{i-1} + x_i| < i \cdot \beta^p \,\mathrm{ulp}(x_k)$$

and thereby

$$|s_i - (s_{i-1} + x_i)| \leq \frac{1}{2}\beta^{\lceil \log_\beta(i) \rceil} \,\mathrm{ulp}(x_k) \tag{16}$$

for $i = 2, 3, \ldots, n$.

By definition of q, we have $2q \geq p + 1 + \lfloor \log_\beta 2 \rfloor$, such that

$$n \leq \frac{\beta^q}{\beta+1} + 2\beta^{p-q} < \beta^{q-1} + 2\beta^{q-1-\lfloor \log_\beta 2 \rfloor} \leq \beta^q.$$

Let $r := \lfloor \log_\beta n \rfloor$. Then $n < \beta^q \implies r < q$ and

$$\sum_{i=2}^{n} |s_i - s_{i-1} - x_i| = \sum_{i=\beta^r+1}^{n} |s_i - s_{i-1} - x_i| + \sum_{i=\beta^{r-1}+1}^{\beta^r} |s_i - s_{i-1} - x_i| + \ldots$$

$$
\begin{aligned}
&\dots + \sum_{i=\beta+1}^{\beta^2} |s_i - s_{i-1} - x_i| + \sum_{i=2}^{\beta}|s_i - s_{i-1} - x_i| \\
&\leq (n-\beta^r)\frac{1}{2}\beta^{r+1}\,\mathrm{ulp}(x_k) + (\beta^r - \beta^{r-1})\frac{1}{2}\beta^r\,\mathrm{ulp}(x_k) + \dots \\
&\quad \dots + (\beta^2-\beta)\frac{1}{2}\beta^2\,\mathrm{ulp}(x_k) + (\beta-1)\frac{1}{2}\beta^1\,\mathrm{ulp}(x_k) \\
&= \frac{1}{2}\left((n-\beta^r)\beta^{r+1} + (\beta-1)\sum_{i=1}^{r}\beta^{2i-1}\right)\mathrm{ulp}(x_k) \\
&\leq \frac{1}{2}\left(\max\{n-\beta^{q-1},0\}\cdot\beta^q + (\beta-1)\sum_{i=1}^{q-1}\beta^{2i-1}\right)\mathrm{ulp}(x_k).
\end{aligned}
$$

Moreover, $2\beta^{p-q} \geq 2\beta^{q-2-\lfloor \log_\beta 2\rfloor} \geq \beta^{q-2} > \frac{\beta^{q-1}}{\beta+1}$ and the bound on n imply

$$
\max\{n-\beta^{q-1},0\} \leq \max\left\{\frac{\beta^q}{\beta+1} + 2\beta^{p-q} - \beta^{q-1}, 0\right\} = 2\beta^{p-q} - \frac{\beta^{q-1}}{\beta+1}.
$$

Together with

$$
(\beta-1)\sum_{i=1}^{q-1}\beta^{2i-1} \leq (\beta-1)\beta^{2q-3}\sum_{i=0}^{\infty}\beta^{-2i} = \frac{(\beta-1)\beta^{2q-3}}{1-\beta^{-2}} = \frac{\beta^{2q-1}}{\beta+1},
$$

this yields

$$
\sum_{i=2}^{n}|s_i - s_{i-1} - x_i| \leq \frac{1}{2}\left(\left(2\beta^{p-q} - \frac{\beta^{q-1}}{\beta+1}\right)\beta^q + \frac{\beta^{2q-1}}{\beta+1}\right)\mathrm{ulp}(x_k) = \beta^p\,\mathrm{ulp}(x_k).
$$

Since each error term is as well a multiple of $\mathrm{ulp}(x_k)$, we have $e + t \in \mathbb{F}$ in every iteration of the `for`-loop; the summation is error-free.

The argument for Remark 2 is very similar. However, for faithful-addition $|a+b| \leq \beta^t$ only implies $|a \oplus b - (a+b)| < \beta^{t-p}$ so that we loose the factor $\frac{1}{2}$ in (16). To prove that the right-hand side of

$$
\sum_{i=2}^{n}|s_i - s_{i-1} - x_i| \leq \left(\max\{n-\beta^{q_f-1},0\}\cdot\beta^{q_f} + (\beta-1)\sum_{i=1}^{q_f-1}\beta^{2i-1}\right)\mathrm{ulp}(x_k)
$$

is less than or equal to $\beta^p\,\mathrm{ulp}(x_k)$, we distinguish the cases $n < \beta^{q_f-1}$ and $\beta^{q_f-1} \leq n \leq \frac{\beta^{q_f}}{\beta+1} + \beta^{p-q_f}$. Both cases can be shown by similar arguments as above.

For the proof of Remark 3, we follow again a similar approach. Due to the slightly worse estimate for $|e|$ in (12), we have to update the inequality for the overall sum of errors as follows:

$$\sum_{i=2}^{n} |s_i - s_{i-1} - x_i| \leq \frac{1}{2} \left(\max\{n - \beta^{q-1}, 0\} \cdot \beta^q + (\beta - 1) \sum_{i=1}^{q-1} \beta^{2i-1} \right) \operatorname{ulp}(x_k)$$
$$+ n \cdot \left\lfloor \frac{\beta - 2}{2} \right\rfloor \operatorname{ulp}(x_k).$$

The cases $n < \beta^{q-1}$ and $\beta^{q-1} \leq n \leq \frac{\beta^q}{\beta+1} + 2\beta^{p-q} - \frac{6}{\beta} \left\lfloor \frac{\beta-2}{2} \right\rfloor$ may then be treated individually using the inequalities from above. □

A good example for the benefit of Corollary 1 is the OnlineExactSum algorithm introduced in [20]. The core element of this algorithm is the addition of all summands into the respective accumulators. This is done according to the exponent of the most significant digit of each summand. To every possible exponent position there is an accumulator pair assigned; one floating-point number for the approximate sum and another for the corresponding error. The authors, Zhu and Hayes, advised to use Dekker's procedure together with the error sum after an `if`-statement for a branch depending on the comparison of the exponents of the intermediate sum and the current summand. Corollary 1 demonstrates that the branching is not necessary and that their bound on the number of summands until possible loss of digits can be improved.[3]

Moreover, Remarks 2 and 3 show that the algorithm also works for faithfully rounded operations and that it can be easily modified for general bases β, requiring only one more operation at each step instead of the three additional operations that would be introduced if we replaced FastTwoSum with TwoSum [8, Theorem B, 4.2.2].

4.2 Error-free transformation - ThreeProduct

Many adaptive and accurate algorithms for problems involving products of three numbers use error-free transformations to transform these terms into unevaluated sums of four floating-point numbers. Exemplary, we want to mention the adaptive algorithms for the 3D orientation problem given in [3,4,12,16]. Here we designate the algorithm that realizes this transformation as FourSumThreeProduct.

```
Function $(s_1, s_2', s_3', s_4') \leftarrow$ FourSumThreeProduct($x_1, x_2, x_3$)
$(t_h, t_l) \leftarrow$ TwoProduct($x_2, x_3$)
$(s_1, s_2') \leftarrow$ TwoProduct($x_1, t_h$)
$(s_3', s_4') \leftarrow$ TwoProduct($x_1, t_l$)
```

The subroutine TwoProduct is a well-known algorithm [2,11] for the transformation of a product of two floating-point numbers into an unevaluated sum of two floating-point numbers. If neither under- nor overflow occurs, this transformation is error-free. To be more specific, we have

[3] For the IEEE 754 `binary64` format, our bound on n is more than twice as high as the bound $\beta^{\lfloor p/2 \rfloor}$ given in [20].

$$t_h + t_l = x_2 \cdot x_3, \quad t_h = x_2 \odot x_3, \quad |t_l| \leq \frac{1}{2}\operatorname{ulp}(x_2 \cdot x_3) \tag{17}$$

in line 1 of FourSumThreeProduct. From (17) and the respective conditions for line 2 and 3, the equality $s_1 + \sum_{i=2}^{4} s_i' = \prod_{i=1}^{3} x_i$ is evident.

Nevertheless, the aforementioned application in mind, this transformation is improvable. We will show that the error of the sum $\mathrm{fl}(s_2' + s_3')$ can be added to s_4' without introducing another rounding error. It is therefore possible to replace s_2', s_3', s_4' with only two addends. In particular, we prove that—although $|s_2'| \geq |s_3'|$ and even $\operatorname{ulp}(s_2') \geq \operatorname{ulp}(s_3')$ do not generally hold true—condition (7) is always satisfiable. Thus, it is possible to use FastTwoSum without any restriction on the base β.

Function $(s_1, s_2, s_3) \leftarrow \mathrm{ThreeProduct}(x_1, x_2, x_3)$

```
$(s_1, s_2', s_3', s_4') \leftarrow \mathrm{FourSumThreeProduct}(x_1, x_2, x_3)$
$(s_2, s_3'') \leftarrow \mathrm{FastTwoSum}(s_2', s_3')$
$s_3 \leftarrow s_3'' \oplus s_4'$
```

Lemma 4 *Consider the procedure* ThreeProduct *and assume the absence of underflow errors within the* FourSumThreeProduct *call. Then*

$$\sum_{i=1}^{3} s_i = \prod_{i=1}^{3} x_i. \tag{18}$$

Proof In the absence of underflow errors, the FourSumThreeProduct transformation is free of errors. We will prove (18) by validating $s_2 + s_3 = \sum_{i=2}^{4} s_i'$. The following argument applies independently of a scaling by a power of β, provided overflow and underflow do not occur. In this respect and by triviality of the case $x_1x_2x_3 = 0$, we henceforth assume without loss of generality $\operatorname{ulp}(x_i) = 1$ for $i = 1, 2, 3$, such that

$$x_1, x_2, x_3 \in \mathbb{Z} \qquad \text{and} \qquad \max\{|x_1|, |x_2|, |x_3|\} \leq \beta^p - 1.$$

Let t_h and t_l be the output of TwoProduct in line 1 of FourSumThreeProduct and denote by $\mathrm{fl}_\triangle(\cdot)$ a rounding to $+\infty$, that is,

$$\forall r \in (\beta^p - 1)\beta^{\mathrm{e_{max}}}[-1, 1]: \quad \mathrm{fl}_\triangle(r) := \min\{f \in \mathbb{F} : r \leq f\}.$$

For the absolute value of t_h, we have

$$|t_h| \leq \mathrm{fl}_\triangle(|x_2x_3|) \leq \mathrm{fl}_\triangle((\beta^p - 1)^2) = \mathrm{fl}_\triangle(\beta^{2p} - 2\beta^p + 1) = \beta^{2p} - \beta^p \in \mathbb{F}.$$

Together with (17), we further derive (in order)

$$|t_l| \leq \frac{1}{2}\operatorname{ulp}(x_2x_3) \leq \frac{1}{2}\operatorname{ulp}((\beta^p - 1)(\beta^p - 1)) \leq \frac{1}{2}\beta^p,$$

$$\begin{aligned}
|s_2'| &\le \frac{1}{2}\,\mathrm{ulp}(x_1 t_h) \le \frac{1}{2}\,\mathrm{ulp}((\beta^p - 1)(\beta^{2p} - \beta^p)) \le \frac{1}{2}\beta^{2p},\\
|s_3'| &\le \mathrm{fl}_\triangle(|x_1 t_l|) \le \mathrm{fl}_\triangle((\beta^p - 1)|t_l|) \le \beta^p |t_l| \le \frac{1}{2}\beta^{2p},\\
|s_4'| &\le \frac{1}{2}\,\mathrm{ulp}(x_1 t_l) \le \frac{1}{2}\,\mathrm{ulp}((\beta^p - 1)\frac{1}{2}\beta^p) \le \frac{1}{2}\beta^p.
\end{aligned}$$

The error term s_2' of the product $x_1 t_h$ is necessarily a multiple of $\mathrm{ulp}(x_1)\,\mathrm{ulp}(t_h)$. Hence, there is a representation of s_2' satisfying $\beta^{\mathrm{e}(s_2')} \ge \mathrm{ulp}(t_h)$ by which

$$|s_3'| \le \beta^p |t_l| \le \frac{1}{2}\beta^p\,\mathrm{ulp}(t_h) \le \frac{1}{2}\beta^p \beta^{\mathrm{e}(s_2')} \le \left\lceil \beta^p - \frac{\beta}{2} \right\rceil \beta^{\mathrm{e}(s_2')}.$$

Condition (7) is satisfied and thereby

$$s_2 + s_3'' = s_2' + s_3'.$$

By $|s_2' + s_3'| \le |s_2'| + |s_3'| \le \beta^{2p}$, we have $|s_3''| = |s_2 - (s_2' + s_3')| \le \frac{1}{2}\beta^p$ and

$$|s_3'' + s_4'| \le |s_3''| + |s_4'| \le \frac{1}{2}\beta^p + \frac{1}{2}\beta^p \le \beta^p,$$

so that $s_3'' + s_4' \in \mathbb{Z}$ lies in $\mathbb{F}$ and $s_3 = s_3'' + s_4'$ is evaluated without error. □

4.3 Accurate summation of preordered addends

As a final example for the applicability of FastTwoSum, we consider a summation approach due to Demmel and Hida. In [3], the authors were concerned with recursive summation of floating-point numbers that are sorted according to their ULP in non-ascending order. For the summation via an extended floating-point register with k bits additional precision and assuming that the number of addends is bounded by $1 + \left\lfloor \frac{2^k}{1-2^{-p}} \right\rfloor$, Demmel and Hida proved a small relative error ($\approx$ 1.5 ulp) of the computed result. However, the availability of extended precision formats depends on the CPU architecture as well as the programming language. If such a format is not available, high-precision numbers need to be emulated. In this context, we consider the DoubleDouble type implemented in the QD library [5].

The algorithm for adding a floating-point `double` number to a DoubleDouble number requires 10 operations. This is less than the 14 additions used in the respective implementation in the DoubleDouble library [1] but still improvable for our purpose. For the summation within the loop of Algorithm 2, we simply took the code from the QD library and replaced the TwoSum call with its FastTwoSum equivalent. This is possible due to the ordering of the addends. Though Algorithm 2 requires only 7 operations per addition into the double-word accumulator (s_h, s_l), this pair of p-bit floating-point numbers behaves almost the same as an actual $2p$-bit floating-point number.

Algorithm 2: Preordered summation using a double-word number

```
Data: $x \in \mathbb{F}^n$ where $2 \le n \le \beta^p + 1$
Result: $s_h \in \mathbb{F}$ satisfying $|s_h - \sum_i x_i| \lessapprox 1.5\,\mathrm{ulp}(s_h)$
1 sort the $x_i$ so that $\mathrm{e}(x_1) \ge \mathrm{e}(x_2) \ge \ldots \ge \mathrm{e}(x_n)$
2 $(s_h, s_l) \leftarrow \mathrm{FastTwoSum}(x_1, x_2)$                    // initial summation
3 for $i \leftarrow 3$ to $n$ do
4     $(t_h, v_l) \leftarrow \mathrm{FastTwoSum}(s_h, x_i)$
5     $t_l \leftarrow s_l \oplus v_l$
6     $(s_h, s_l) \leftarrow \mathrm{FastTwoSum}(t_h, t_l)$
7 end
```

Corollary 2 *Let $s_h, s_l, x_i \in \mathbb{F}$ with $\beta \in \{2, 3\}$ satisfy*

$$\frac{s_h}{\mathrm{ulp}(x_i)} \in \mathbb{Z}, \quad \frac{s_l}{\mathrm{ulp}(x_i)} \in \mathbb{Z}, \quad \textit{and} \quad s_h = s_h \oplus s_l. \tag{19}$$

Assume a mantissa length $p \ge 2$ and let $\mathbb{F}_{2p}$ denote a floating-point system with the same base and exponent range as $\mathbb{F}$ but twice the mantissa length. If $t_h, t_l, v_l \in \mathbb{F}$ are evaluated as in the lines 4 and 5 of Algorithm 2, then

$$|t_h + t_l - (s_h + s_l + x_i)| \le \min_{f \in \mathbb{F}_{2p}} |f - (s_h + s_l + x_i)|. \tag{20}$$

Moreover, the pair (t_h, t_l) meets the conditions for Theorem 2.

Proof For the trivial case $s_h = 0$ the result is evident. By the symmetry of $\mathbb{F}$, we henceforth assume without loss of generality that s_h is positive. Due to (19) and our general assumptions, s_h and x_i satisfy the conditions in Theorem 1. Thus,

$$t_h + v_l = s_h + x_i \iff t_l - (s_l + v_l) = t_h + t_l - (s_h + s_l + x_i)$$

enables us to replace the right-hand side of (20) with $|t_l - (s_l + v_l)|$. Since additions in the underflow range are evaluated without rounding errors, the following estimates remain valid for addends and intermediate results in the underflow range. Nevertheless, for reasons of clarity, we henceforth assume that all numbers lie in the normalized range. The proof of (20) is by distinction into the following four cases, for which we define $s_{\max} := \max\{s_h + s_l, |s_h + x_i|\}$.

Case 1 Suppose $|x_i| \ge \beta^{-1}\,\mathrm{ulp}(s_{\max})$. Using (19), we derive

$$\frac{s_l + v_l}{\mathrm{ulp}(x_i)} \in \mathbb{Z} \implies \frac{s_l + v_l}{\beta^{-p}\,\mathrm{ulp}(s_{\max})} \in \mathbb{Z}.$$

Together with

$$|s_l + v_l| \le |s_l| + |v_l| \le \frac{1}{2}\,\mathrm{ulp}(s_h + s_l) + \frac{1}{2}\,\mathrm{ulp}(s_h + x_i) \le \mathrm{ulp}(s_{\max}), \tag{21}$$

this implies that $s_l + v_l$ is representable by p mantissa digits and therefore evaluated without rounding error.

Case 2 Assume $\mathrm{ulp}(s_{\max}) \le \mathrm{ulp}(s_h + s_l + x_i)$. Then (21) implies

$$|s_l + v_l| \le \mathrm{ulp}(s_{\max}) \le \mathrm{ulp}(s_h + s_l + x_i).$$

If these inequalities are actually equalities, the computation is error-free. On the other hand, if the outer inequality is strict, $p \ge 2$ and $|s_l + v_l| < \mathrm{ulp}(s_h + s_l + x_i) = \mathrm{ulp}(t_h + s_l + v_l)$ imply $\mathrm{ulp}(s_l + v_l) \le \mathrm{ulp}(t_h)$. Hence, t_h has no significant digits whose exponents are smaller than the exponent of the digit at rounding position. The number represented by $t_h + t_l$ results from a nearest rounding of the base β representation of $t_h + v_l + s_l$ at the position with value $\mathrm{ulp}(s_l + v_l)$. Together with $\mathrm{ulp}(s_l + v_l) \le \beta^{-p}\,\mathrm{ulp}(s_h + s_l + x_i)$, this yields (20).

Case 3 Assume

$$|s_h + x_i| \ge \beta^{p-1}\,\mathrm{ulp}(s_{\max}) > s_h + s_l + x_i \quad \text{and} \quad |x_i| < \beta^{-1}\,\mathrm{ulp}(s_{\max}).$$

Then $s_l < 0$ and $s_h = \mathrm{fl}(s_h + s_l) \le \beta^{p-1}\,\mathrm{ulp}(s_{\max})$. Since the difference between $\beta^{p-1}\,\mathrm{ulp}(s_{\max})$ and its neighbored floating-point numbers is strictly greater than x_i, the only feasible choice for s_h is $s_h = \beta^{p-1}\,\mathrm{ulp}(s_{\max})$. This also implies $v_l = x_i \ge 0$, $-\frac{1}{2\beta}\,\mathrm{ulp}(s_{\max}) \le s_l$, and $s_l + x_i < 0$, by which

$$|s_l + v_l| \le \frac{1}{2\beta}\,\mathrm{ulp}(s_{\max}) = \frac{1}{2}\,\mathrm{ulp}(s_h + s_l + x_i).$$

The remainder follows by a similar argument as in *Case* 2.

Case 4 Suppose that none of the previous cases apply, so that

$$s_h + s_l \ge \beta^{p-1}\,\mathrm{ulp}(s_{\max}) > s_h + s_l + x_i \quad \text{and} \quad |x_i| < \beta^{-1}\,\mathrm{ulp}(s_{\max}).$$

Similarly as above, we follow that $s_h = \beta^{p-1}\,\mathrm{ulp}(s_{\max})$ is the only feasible choice for s_h. By $t_h = \mathrm{fl}(s_h + x_i) \ge s_h - \beta^{-1}\,\mathrm{ulp}(s_h) \in \mathbb{F}$, we have

$$s_l + v_l = s_h + s_l + x_i - t_h < s_h - t_h \le \beta^{-1}\,\mathrm{ulp}(s_h).$$

Together with $s_l \ge 0$ and $|v_l| \le \frac{1}{2}\,\mathrm{ulp}(s_h + x_i) = \frac{1}{2\beta}\,\mathrm{ulp}(s_{\max})$, this gives

$$|s_l + v_l| \le \beta^{-1}\,\mathrm{ulp}(s_{\max}) = \mathrm{ulp}(s_h + s_l + x_i).$$

Using once again the argument from *Case* 2, we prove (20).

For the proof of the second statement of Corollary 2, we distinguish two cases. First, assume that $x_i > -\frac{1}{2}s_h$ and therefore $\frac{1}{\beta}s_h \le \frac{1}{2}s_h \le t_h$. Then $|s_l| \le \frac{1}{2}\,\mathrm{ulp}(s_h) \le \frac{\beta}{2}\,\mathrm{ulp}(t_h)$, $|v_l| \le \frac{1}{2}\,\mathrm{ulp}(t_h)$, and $p \ge 2$ yields

$$|s_l + v_l| \le \frac{\beta}{2}\,\mathrm{ulp}(t_h) + \frac{1}{2}\,\mathrm{ulp}(t_h) \le \frac{\beta+1}{2}\beta^{\mathrm{e}(t_h)} \implies |t_l| \le \left\lceil \beta^p - \frac{\beta}{2} \right\rceil \beta^{\mathrm{e}(t_h)}.$$

On the contrary, suppose $x_i \leq -\frac{1}{2}s_h$. Then $|s_h + x_i| \leq |x_i|$ and $\frac{s_h+x_i}{\operatorname{ulp}(x_i)} \in \mathbb{Z}$ imply $t_h = s_h + x_i \in \mathbb{F}$. We then use $v_l = s_h + x_i - t_h = 0$ to validate

$$|t_l| = |s_l| \leq \frac{1}{2}\operatorname{ulp}(s_h) \leq \frac{\beta}{2}\operatorname{ulp}(x_i) \leq \frac{\beta}{2}\beta^{\min\{\mathrm{e}(s_h),\mathrm{e}(x_i)\}} \leq \left\lceil \beta^p - \frac{\beta}{2} \right\rceil \beta^{\mathrm{e}(t_h)}$$

and complete the proof. □

If the considered arithmetic obeys an unambiguous tie-breaking rule, this result can be proved also for $p = 1$.[4] Nevertheless, due to the absence of practical relevance, we skip the argument for this case.

To treat floating-point systems with bases other than 2 or 3, we may replace the FastTwoSum calls in line 2 and 4 of Algorithm 2 with their TwoSum equivalents or safe two operations per iteration by using c_β-FastTwoSum instead. The latter modification requires $p \geq 3$ and may cause a loss of accuracy by two mantissa digits. Moreover, although Remark 1 is not applicable here, a generalization to faithful-summation is possible if we use any of the means of computing a suitable $\tilde{y}$ described at the end of Section 3.

For the sake of clarity, we refrain from discussing either of the above mentioned modifications and leave it to the well-disposed reader. Instead, we conclude this note by deducing an error estimate for the output of Algorithm 2 similar to the one in [3]. In exchange for a tighter estimate, unlike [3, Theorem 1], the following result only regards the range from 2 to $\beta^p + 1$ for the number of addends. On the other hand, due to the restriction to our specific problem and the use of techniques from optimization, our proof is much more compact than the argument by Demmel and Hida.

Theorem 4 *For given $x \in \mathbb{F}^n$, let $s_h, s_l \in \mathbb{F}$ be evaluated according to Algorithm 2. If $\beta \in \{2, 3\}$, $p \geq 2$, and $2 \leq n \leq \beta^p + 1$, then*

$$\left| s_h + s_l - \sum_{i=1}^{n} x_i \right| \leq (n-2)\frac{\beta^{1-2p}}{2}|s_h + s_l| \leq \frac{\beta^{1-p}}{2}|s_h + s_l|. \tag{22}$$

Proof Let t_{i+2} denote the computed approximation represented by the unevaluated sum $t_h + t_l$ in the i-th step of the `for`-loop of Algorithm 2. Moreover, let k denote the index where the accumulation is erroneous the first time, i.e., $t_k \neq t_{k-1} + x_k = \sum_{i=1}^{k} x_i$. By design the initial transformation is always error-free and therefore $k \geq 3$.

With regard to $I := \{k, k+1, \dots, n\}$, define $u_s := \max\{\operatorname{ulp}(t_i) \colon i \in I\}$ as well as the index sets

$$I_1 := \{i \in I \colon \operatorname{ulp}(t_i) = u_s\}, \quad I_2 := \{i \in I \colon \operatorname{ulp}(t_i) < u_s,\ t_i \neq t_{i-1} + x_i\}.$$

In the context of estimate (20), the first erroneous accumulation satisfies

$$0 < |t_k - (t_{k-1} + x_i)| \leq \frac{1}{2}\beta^{-p}\operatorname{ulp}(t_{k-1} + x_i) \leq \frac{1}{2}\beta^{-p}\operatorname{ulp}(t_k).$$

[4] For this purpose one can use Sterbenz Lemma [18] and discuss the few possible cases occurring for $p = 1$ and $\beta = 2$.

Thus, there is no power of β larger than $\beta^{-p-1}\,\mathrm{ulp}(t_k)$ that divides both x_k and t_{k-1}. By the ordering in line 1 of Algorithm 2, the same is true for all subsequent addends, so that

$$\forall i \in I\colon\ \mathrm{ulp}(x_i) \le \beta^{-p-1}\,\mathrm{ulp}(t_k) \le \beta^{-p-1}u_s \implies |x_i| < \beta^{-1}u_s.$$

In a similar way, using the definition of I_2, we derive the stricter bound

$$\forall i \in I_2\colon\ \mathrm{ulp}(x_i) \le \beta^{-p-1}\,\mathrm{ulp}(t_i) \le \beta^{-p-2}u_s \implies |x_i| < \beta^{-2}u_s.$$

Denote by n_1 and n_2 the cardinalities of the sets I_1 and I_2, respectively. Note that $|I\setminus(I_1\cup I_2)| \le n-2-n_1-n_2$ and $\forall i \in I_1\colon |t_i| \ge \beta^{p-1}u_s$. Without loss of generality, we further assume that I_1 is not empty. This is possible because $I_1 = \emptyset$ implies $I = \emptyset$ and therefore the absence of approximation errors. By a similar argument as for Lemmas 2 and 3, we derive

$$\begin{aligned} |t_n| &\ge \min_{i\in I_1}|t_i| - |I \setminus (I_1 \cup I_2)| \max_{i\in I}|x_i| - |I_2|\beta^{-2}u_s \\ &\ge \beta^{p-1}u_s - (n-2-n_1-n_2)\beta^{-1}u_s - n_2\beta^{-2}u_s > 0. \end{aligned}$$

On the other hand, Corollary 2 implies the following individual error bounds:

$$|t_i - (t_{i-1}+x_i)| \le \begin{cases} \frac{1}{2}\beta^{-p}u_s & \text{if } i \in I_1, \\ \frac{1}{2}\beta^{-p-1}u_s & \text{if } i \in I_2, \\ 0 & \text{otherwise.} \end{cases}$$

By combining these inequalities, we derive the estimate

$$\begin{aligned} \left|t_n - \sum_{i=1}^{n} x_i\right| &\le \sum_{i\in I_1}|t_i - (t_{i-1}+x_i)| + \sum_{i\in I_2}|t_i - (t_{i-1}+x_i)| \\ &\le n_1\frac{1}{2}\beta^{-p}u_s + n_2\frac{1}{2}\beta^{-p-1}u_s \\ &\le \frac{1}{2}\frac{n_1\beta^{-p}u_s + n_2\beta^{-p-1}u_s}{\beta^{p-1}u_s - (n-2-n_1-n_2)\beta^{-1}u_s - n_2\beta^{-2}u_s}|t_n| \\ &= \frac{1}{2}\frac{\beta^{1-p}n_1 + \beta^{-p}n_2}{\beta^p - n + 2 + n_1 + (1-\beta^{-1})n_2}|t_n|. \end{aligned}$$

Then

$$\frac{\left|t_n - \sum_{i=1}^{n} x_i\right|}{|t_n|} \le \sup_{\substack{n_1,n_2\in\mathbb{R}_+ \\ n_1+n_2\le n-2}} \frac{1}{2}\frac{\beta^{1-p}n_1 + \beta^{-p}n_2}{\beta^p - n + 2 + n_1 + (1-\beta^{-1})n_2}, \tag{23}$$

where the right-hand side of (23) defines a linear-fractional programming problem. As is well-known [17], such programs are pseudoconvex, therefore every local optimum is a stationary point [10]. Thus, we could proceed by discussing the Karush–Kuhn–Tucker conditions [7,9].

Here we give a simplified argument for the optimal point of this problem. Let $f(n_1, n_2)$ denote the objective function in (23). For any feasible choice of $n_2^* \in [0, n-3]$, nonnegativity of n_1 and the bound $n \leq \beta^p + 1$ imply

$$\frac{\partial f(n_1, n_2^*)}{\partial n_1} = \frac{\beta^{1-p}}{2} \frac{\beta^p - n + 2 + (1 - 2\beta^{-1})n_2^*}{(\beta^p - n + 2 + n_1 + (1 - \beta^{-1})n_2^*)^2} > 0.$$

Evidently, $f(n_1, n_2^*)$ is maximized for the largest feasible n_1, such that the optimal point (n_1^*, n_2^*) satisfies $n_1^* + n_2^* = n - 2$. Moreover, the objective function $f(n_1, n - 2 - n_1)$ is again strictly monotonically increasing for all $n_1 \geq 0$. The point that maximizes the right-hand side of (23) is $(n_1^*, n_2^*) := (n-2, 0)$ and the bound proposed in Theorem 4 follows immediately. □

5 Conclusion

In most previous works that involve the use of Dekker's FastTwoSum algorithm, not only is the floating-point system restricted to bases $\beta \in \{2, 3\}$ but also it is assumed that the summands x, y satisfy $|x| \geq |y|$. In this note, we reminded that Dekker's original result is more general than this. The three examples in the previous section and further examples in the literature, including the accurate summation algorithms introduced in [13,15], demonstrate a wider applicability of FastTwoSum.

Theorem 2 generalizes Dekkers's condition for floating-point systems with larger bases β. Here our result is used to show that the transformation by ThreeProduct is error-free independent of the choice of β. It also can be used to prove similar generalizations for the algorithms in [13,15].

Moreover, we introduced a modified version of FastTwoSum that requires four instead of three operations but enables us to apply the FastTwoSum approach in cases where the conditions of Theorem 2 are not met. In the first and the third application discussed above, this is a better alternative than using the TwoSum function whose implementation require six basic operations.

We also brought up the applicability of FastTwoSum in the presence of faithful rounding. This can be useful if we work on a platform that, for the sake of performance or for other reasons, does not support rounding to nearest. The consideration of faithful rounding is also necessary if one has no control about possible changes of rounding modes caused by other routines.

Acknowledgements The authors wish to thank the two anonymous referees for their fruitful critics and helpful comments. Without their feedback, we may not have come up with a generalization of Dekker's original result.

References

1. Briggs, K.: The `doubledouble` library. https://boutell.com/fracster-src/doubledouble/doubledouble.html (1998)

2. Dekker, T.J.: A floating-point technique for extending the available precision. Numer. Math. **18**(3), 224 (1971). https://doi.org/10.1007/BF01397083
3. Demmel, J., Hida, Y.: Fast and accurate floating point summation with application to computational geometry. Numer. Algorithms **37**(1–4), 101–112 (2004). https://doi.org/10.1023/b:numa.0000049458.99541.38
4. Graillat, S., Louvet, N.: Applications of fast and accurate summation in computational geometry. Technical report, Laboratoire LP2A, University of Perpignan, Perpignan, France (2006)
5. Hida, Y., Li, X.S., Bailey, D.H.: C++/fortran-90 double-double and quad-double package, version 2.3.17. http://crd-legacy.lbl.gov/~dhbailey/mpdist/ (2012)
6. Kahan, W.: Further remarks on reducing truncation errors. Commun. ACM **8**(1), 40 (1965). https://doi.org/10.1145/363707.363723
7. Karush, W.: Minima of functions of several variables with inequalities as side conditions. In: Traces and Emergence of Nonlinear Programming, pp. 217–245. Springer, Berlin (2013). https://doi.org/10.1007/978-3-0348-0439-4_10
8. Knuth, D.E.: The Art of Computer Programming. Pearson Education (US) (1997)
9. Kuhn, H.W., Tucker, A.W.: Nonlinear programming. In: Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, pp. 481–492. University of California Press, Berkeley and Los Angeles (1951)
10. Mangasarian, O.L.: Pseudo-convex functions. SIAM Ser. A. Control **3**(2), 281–290 (1965). https://doi.org/10.1137/0303020
11. Ogita, T., Rump, S.M., Oishi, S.: Accurate sum and dot product. SIAM J. Sci. Comput. **26**(6), 1955–1988 (2005). https://doi.org/10.1137/030601818
12. Ozaki, K., Ogita, T., Rump, S.M., Oishi, S.: Fast and robust algorithm for geometric predicates using floating-point arithmetic. Trans. JSIAM **4**(16), 553–562 (2006). [in Japanese]
13. Rump, S.M.: Ultimately fast accurate summation. SIAM J. Sci. Comput. **31**(5), 3466–3502 (2009). https://doi.org/10.1137/080738490
14. Rump, S.M., Ogita, T., Oishi, S.: Accurate floating-point summation. Part I: faithful rounding. SIAM J. Sci. Comput. **31**(1), 189–224 (2008). https://doi.org/10.1137/050645671
15. Rump, S.M., Ogita, T., Oishi, S.: Fast high precision summation. NOLTA **1**(1), 2–24 (2010). https://doi.org/10.1587/nolta.1.2
16. Shewchuk, J.R.: Adaptive precision floating-point arithmetic and fast robust geometric predicates. Discrete Comput. Geom. **18**(3), 305–363 (1997). https://doi.org/10.1007/pl00009321
17. Stancu-Minasian, I.M.: Fractional Programming. Springer, Berlin (1997). https://doi.org/10.1007/978-94-009-0035-6
18. Sterbenz, P.H.: Floating-Point Computation. Prentice-Hall, Engelwood Cliffs (1974)
19. Wolfe, J.M.: Reducing truncation errors by programming. Commun. ACM **7**(6), 355–356 (1964). https://doi.org/10.1145/512274.512287
20. Zhu, Y.K., Hayes, W.B.: Algorithm 908. ACM Trans. Math. Softw. **37**(3), 1–13 (2010). https://doi.org/10.1145/1824801.1824815