Numerical validation of compensated summation algorithms with stochastic arithmetic

S. Graillat[1]

Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France
CNRS, UMR 7606, LIP6, F-75005, Paris, France

F. Jézéquel[2]

Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France
CNRS, UMR 7606, LIP6, F-75005, Paris, France
Université Panthéon-Assas, 12 place du Panthéon, F-75231 Paris CEDEX 05, France

R. Picot[3]

Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France
CNRS, UMR 7606, LIP6, F-75005, Paris, France

Abstract

Compensated summation algorithms are designed to improve the accuracy of ill-conditioned sums. They are based on algorithms, such as FastTwoSum, which are proved to provide, with rounding to nearest, the sum of two floating-point numbers and the associated rounding error. Discrete stochastic arithmetic enables one to estimate rounding error propagation in numerical codes. It requires a random rounding mode which consists in rounding each computed result toward $-\infty$ or $+\infty$ with the same probability. In this paper we analyse the impact of this random rounding mode on compensated summations based on the FastTwoSum algorithm. We show the accuracy improvement obtained using such compensated summations in numerical simulations controlled with discrete stochastic arithmetic.

Keywords: floating-point arithmetic, rounding errors, discrete stochastic arithmetic, error-free transformations, compensated algorithms, summation algorithms, CADNA

1 Introduction

The power of computational resources continues to increase. Exascale computing (10^{18} operations per second) is planned to be reached within a decade. In floating-point arithmetic, each operation is likely to produce a rounding error. These errors

[1] Email: stef.graillat@lip6.fr
[2] Email: fabienne.jezequel@lip6.fr
[3] Email: romain.picot@lip6.fr

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

can accumulate and at the end of a computation, the computed result can be very far from the exact one. Moreover, the more operations are performed, the more the accumulation of rounding errors is likely to be important.

As a consequence, it is fundamental to have some information on the numerical quality (for example the number of exact significant digits) of the computed result. To answer this question, a numerical library called CADNA [1] has been developed. It implements Discrete Stochastic Arithmetic (DSA) [2] and makes it possible to provide a confident interval of the computed result.

But if the accuracy of the computed result is not sufficient, it is necessary to increase the precision of the computation. A well-known and efficient technique for that is the use of compensated algorithms. These algorithms are based on the fact that it is often possible to compute exactly the rounding errors of some elementary operations like addition and multiplication. We now assume that we work with a floating-point arithmetic adhering to the IEEE754-2008 Standard [3]. In that case, if we use rounding to nearest, then the rounding error of an addition is a floating-point number that can be computed exactly. The algorithms that enable the computation of rounding errors are called *error-free transformations* (EFT). An algorithm that relies on EFT to increase the accuracy is called a *compensated algorithm* (see [4]).

However if we use directed rounding, the error of a floating-point addition is not necessarily a floating-point number. Yet, directed roundings are required in DSA. As a consequence, it is not clear whether we can use stochastic arithmetic to validate some numerical codes that heavily rely on the use of error-free transformations.

In this article, we show that we can use stochastic arithmetic to validate compensated summation. Several compensated algorithms exist for summation. The first one is Kahan's compensated summation [5]. Another one is the doubly compensated summation algorithm by Priest (see [6] or chapter 4 of [7]). We mainly focus here on the compensated algorithm derived by Ogita, Rump and Oishi [8].

In Section 2, we give some definitions and notations used in the sequel. In Section 3, we present the principles of DSA. In Section 4, we analyse the impact of directed roundings on an EFT for floating-point addition, the FastTwoSum algorithm [9]. We show in Section 5 that we can still use stochastic arithmetic with compensated summation. Section 6 confirms the accuracy of the algorithm and shows performances.

2 Definitions and notations

Throughout the paper, we assume to work with a binary floating-point arithmetic adhering to IEEE 754 floating-point standard [3]. We suppose that no overflow occurs. The set of floating-point numbers is denoted by $\mathbb{F}$, the relative rounding error by $\mathbf{u}$. For IEEE 754 double precision, we have $\mathbf{u} = 2^{-53}$ and for single precision $\mathbf{u} = 2^{-24}$.

We denote by $\mathrm{fl}_*(\cdot)$ the result of a floating-point computation, where all operations inside parentheses are done in floating-point working precision with a directed rounding (that is to say toward $-\infty$ or $+\infty$). Floating-point operations in IEEE 754 satisfy [7]

$\exists\, \varepsilon_1 \in \mathbb{R}$, $\varepsilon_2 \in \mathbb{R}$ such that

$$\mathrm{fl}_*(a \circ b) = (a \circ b)(1+\varepsilon_1) = (a \circ b)/(1+\varepsilon_2) \text{ for } \circ = \{+,-\} \text{ and } |\varepsilon_\nu| \leq 2\mathbf{u}.$$

This implies that

$$|a \circ b - \mathrm{fl}_*(a \circ b)| \leq 2\mathbf{u}|a \circ b| \text{ and } |a \circ b - \mathrm{fl}_*(a \circ b)| \leq 2\mathbf{u}|\,\mathrm{fl}_*(a \circ b)| \text{ for } \circ = \{+,-\}.$$

We use standard notation for error estimations. The quantities γ_n are defined as usual [7] by

$$\gamma_n(\mathbf{u}) := \frac{n\mathbf{u}}{1-n\mathbf{u}} \quad \text{for } n \in \mathbb{N},$$

where we implicitly assume that $n\mathbf{u} \leq 1$.

3 Principles of Discrete Stochastic Arithmetic (DSA)

Based on a probabilistic approach, the CESTAC method [10] allows the estimation of rounding error propagation which occurs with floating-point arithmetic. It uses a random rounding mode which consists in rounding each computed result toward $-\infty$ or $+\infty$ with the same probability. The computer's deterministic arithmetic is replaced by a stochastic arithmetic where each arithmetic operation is performed N times before the next one is executed, thereby propagating the rounding error differently each time. Therefore, for each computed result, the CESTAC method furnishes us with N samples $R_1, \ldots, R_N$. The value of the computed result $\overline{R}$ is chosen to be the mean value of $\{R_i\}$ and, if no overflow occurs, the number of exact significant digits in $\overline{R}$ can estimated as

$$C_{\overline{R}} = \log_{10}\left(\frac{\sqrt{N}\,|\overline{R}|}{\sigma\tau_\beta}\right), \text{ where } \overline{\mathrm{R}} = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{R_i} \text{ and } \sigma^2 = \frac{1}{\mathrm{N}-1}\sum_{\mathrm{i}=1}^{\mathrm{N}} \left(\mathrm{R_i} - \overline{\mathrm{R}}\right)^2. \quad (1)$$

τ_β is the value of Student's distribution for $N-1$ degrees of freedom and a probability level $1-\beta$.

The validity of $C_{\overline{R}}$ is compromised if both operands in a multiplication or the divisor in a division are not significant [11]. It is essential that such numbers with no significance are detected and reported. Therefore multiplications and divisions must be dynamically controlled in order to perform a so-called *self-validation* of the method. The need for this control has led to the concept of computational zero [12]. A computed result is a computational zero, denoted by @.0, if $\forall i, R_i = 0$ or $C_{\overline{R}} \leq 0$. This means that a computational zero is either a mathematical zero or a number without any significance, *i.e.* numerical noise.

To establish consistency between the arithmetic operators and the relational operators, discrete stochastic relations [13] are defined as follows. Let $X = \{X_i\}$ and $Y = \{Y_i\}$ be two results computed with the CESTAC method,

(i) $X = Y$ if and only if $\mathrm{X} - \mathrm{Y} = @.0$,

(ii) $X > Y$ if and only if $\overline{\mathrm{X}} > \overline{\mathrm{Y}}$ and $\mathrm{X} - \mathrm{Y} \neq @.0$,

(iii) $X \geq Y$ if and only if $\overline{\mathrm{X}} \geq \overline{\mathrm{Y}}$ or $\mathrm{X} - \mathrm{Y} = @.0$.

Discrete Stochastic Arithmetic (DSA) is the combination of the CESTAC method, the concept of computational zero, and the discrete stochastic relationships [2].

The CADNA [4] software [1] is a library which implements DSA with $N = 3$ and $\beta = 0.05$. In contrast to interval arithmetic, that computes guaranteed results, the CADNA software provides, with the probability 95% the number of exact significant digits of any computed result. CADNA allows to use new numerical types: the stochastic types. In practice, classic floating-point variables are replaced by the corresponding stochastic variables, which are composed of three perturbed floating-point values. When a stochastic variable is printed, only its exact significant digits appear. Because the library contains the definition of all arithmetic operations and order relations for the stochastic types, the use of CADNA in a program requires only a few modifications: essentially changes in the declarations of variables and in input/output statements. During the execution, CADNA can detect numerical instabilities, which are usually due to the presence of numerical noise. When a numerical instability is detected, dedicated CADNA counters are incremented. At the end of the run, the value of these counters together with appropriate warning messages are printed on standard output.

4 FastTwoSum with faithful rounding

If Algorithm 1 [9] is executed using a binary floating-point system adhering to IEEE 754 standard, with subnormal numbers available, and providing correct rounding with rounding to nearest, then it computes two floating-point numbers s and t such that

- $s + t = a + b$ exactly;
- s is the floating-point number that is closest to $a + b$.

Algorithm 1 FastTwoSum

function $[s, t] =$ `FastTwoSum`(a, b)

1: **if** $|b| \geq |a|$ **then**
2: exchange a and b
3: **end if**
4: $s \leftarrow a + b$
5: $z \leftarrow s - a$
6: $t \leftarrow b - z$

The floating-point number t is the error on the floating-point addition of a and b if Algorithm 1 is executed with rounding to nearest. With another rounding mode this error may not be exactly representable ([14] page 125). In [15], a condition on a and b is given for the `FastTwoSum` algorithm to provide the exact error on the floating-point addition of a and b with directed rounding. In this paper, we aim at analysing the impact of the random rounding mode required by DSA on Algorithm 1. Therefore in the rest of this section, any arithmetic operation in Algorithm 1 is rounded using the fl_* function defined in Section 2. The results

[4] URL address: http://www.lip6.fr/cadna

given in this section have been established using Sterbenz's lemma [16] which is recalled below. As a remark, Sterbenz's lemma is valid with directed rounding. In the Propositions presented in Sections 4 and 5, we assume underflow may occur because, in this case, additions or subtrations generate no rounding error if subnormal numbers are available [17].

Lemma 4.1 (Sterbenz) *In a floating-point system with subnormal numbers available, if x and y are finite floating-point numbers such that $y/2 \leq x \leq 2y$, then $x-y$ is exactly representable.*

In [15], it is shown that the floating-point number z in Algorithm 1 is computed exactly with directed rounding. This property is also true with the random rounding mode. This associated proof is detailed below for completeness.

Proposition 4.2 *The floating-point number z provided by Algorithm 1 using directed rounding is computed exactly, i.e. $z = s - a$.*

Proof:

Let us distinguish two cases.

(i) $a, b \geq 0$:

Because $0 \leq b \leq a$,

$$a \leq a + b \leq 2a \tag{2}$$

From the monotonicity of the fl_* function we deduce

$$a \leq \mathrm{fl}_*(a + b) \leq 2a \tag{3}$$

Then

$$a \leq s \leq 2a \tag{4}$$

Therefore, according to Sterbenz's lemma, $z = s - a$.

(ii) $a \geq 0, b \leq 0$:

- if $-b \geq \frac{a}{2}$, then

$$a \geq -b \geq \frac{a}{2} \tag{5}$$

So $a - (-b)$ is exactly representable because of Sterbenz's lemma. Therefore $s = a + b$ which implies $z = s - a$.

- if $-b < \frac{a}{2}$, then

$$0 \geq b > -\frac{a}{2} \tag{6}$$

Hence

$$a \geq a + b > \frac{a}{2} \tag{7}$$

From the monotonicity of the fl_* function we deduce

$$a \geq s \geq \frac{a}{2} \tag{8}$$

Therefore, from Sterbenz's lemma, $z = s - a$.

The two cases $a, b \leq 0$ and $a \leq 0, b \geq 0$, have a similar proof mainly using $-a$ and $-b$. □.

In general the correction t computed by Algorithm 1 using directed rounding is different from the rounding error e on the sum of a and b. We establish below a relation between t and e.

Proposition 4.3 *Let s and t be the floating-point addition of a and b and the correction both computed by Algorithm 1 using directed rounding. Let e be the error on s: $a + b = s + e$. Then*

$$|e - t| \leq 2\mathbf{u}|e|.$$

Proof:

From Proposition 4.2, z is computed exactly. However with directed rounding, t may not be computed exactly. So $\delta \in \mathbb{R}$ exists such that

$$t = b - z + \delta \tag{9}$$

and

$$|\delta| \leq 2\mathbf{u}|b - z|. \tag{10}$$

From Proposition 4.2, we deduce

$$|\delta| \leq 2\mathbf{u}|a + b - s| \tag{11}$$

Let e be the error on the floating-point addition of a and b, then

$$a + b = s + e \tag{12}$$

with

$$|e| \leq 2\mathbf{u}|a + b|. \tag{13}$$

From Equations (11) and (12), we deduce a bound on $|\delta| = |e - t|$:

$$|\delta| \leq 2\mathbf{u}|e| \tag{14}$$

□.

5 Compensated summation with faithful rounding

The classic algorithm for computing summation is the recursive Algorithm 2.

Algorithm 2 Summation of n floating-point numbers $p = \{p_i\}$

function `res` = `Sum`(p)

1: $s_1 \leftarrow p_1$
2: **for** $i = 2$ to n **do**
3: $\quad s_i \leftarrow s_{i-1} + p_i$
4: **end for**
5: `res` $\leftarrow s_n$

If we denote by $s = \sum_{i=1}^{n} p_i$ the exact summation, $S = \sum_{i=1}^{n} |p_i|$ and `res` the computed summation, it is possible to show [7] that $|s - \texttt{res}| \leq \gamma_{n-1}(2\mathbf{u})|S|$ with directed roundings. This accuracy is sometimes not sufficient in practice. Indeed,

when the condition number $|s|/S$ is large (greater than $1/\mathbf{u}$) then the recursive algorithm does not even return one correct digit.

In Figure 1 and Algorithm 3 [8], a compensated scheme to evaluate the sum of floating-point numbers is presented, *i.e.* the error of individual summation is somehow corrected. Indeed, with Algorithm 1 (`FastTwoSum`), one can compute the rounding error. Algorithm 1 can be cascaded and sum up the errors to the ordinary computed summation.

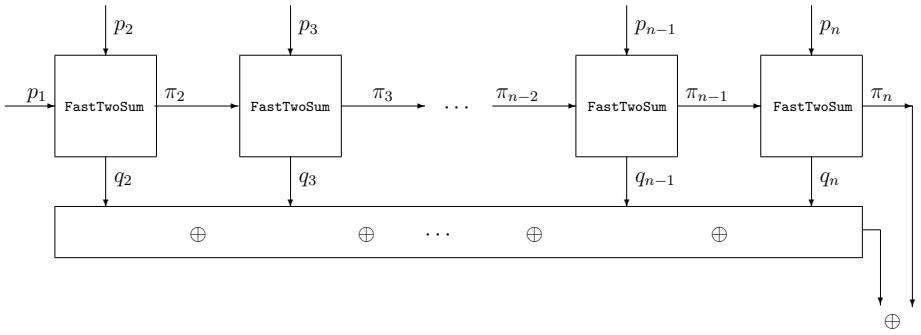

Fig. 1. Compensated summation algorithm

Algorithm 3 Compensated summation of n floating-point numbers $p = \{p_i\}$

function `res` = `FastCompSum`(p)

1: $\pi_1 \leftarrow p_1$
2: $\sigma_1 \leftarrow 0$
3: **for** $i = 2$ to n **do**
4: $\quad [\pi_i, q_i] \leftarrow$ `FastTwoSum`(π_{i-1}, p_i)
5: $\quad \sigma_i \leftarrow \sigma_{i-1} + q_i$
6: **end for**
7: `res` $\leftarrow \pi_n + \sigma_n$

Assuming Algorithm 3 is executed with rounding to nearest, a bound on the accuracy of the result, established in [8], is recalled in Proposition 5.1.

Proposition 5.1 *Let us suppose Algorithm* `FastCompSum` *is applied, with rounding to nearest, to floating-point numbers* $p_i \in \mathbb{F}$, $1 \leq i \leq n$. *Let* $s := \sum p_i$ *and* $S := \sum |p_i|$. *If* $n\mathbf{u} < 1$, *then*

$$|\texttt{res} - s| \leq \mathbf{u}|s| + \gamma_{n-1}^2(\mathbf{u})S \quad \text{with} \quad \gamma_n(\mathbf{u}) = \frac{n\mathbf{u}}{1 - n\mathbf{u}}. \tag{15}$$

We aim at analysing the effects of the random rounding mode on Algorithm 3. In [15], the impact of directed rounding on compensated summation is presented. However the algorithm considered in [15] is slightly different from Algorithm 3. Moreover in [15], the summation is assumed to be performed using one rounding mode, whereas DSA requires frequent changes of the rounding mode. If Algorithm 3 is executed with the random rounding mode, then the EFT are no more exact. However it is shown in Proposition 5.2 that the accuracy obtained with directed rounding is similar to the one given in Proposition 5.1. Because in the proof of

Proposition 5.1, rounding mode changes are allowed, we have an upper bound on the error generated by Algorithm 3 with DSA. As a remark, in this paper, $\mathbf{u}$ has a constant value independent of the roundind mode and previously mentioned in Section 2.

Proposition 5.2 *Let us suppose Algorithm* `FastCompSum` *is applied, with directed rounding, to floating-point numbers* $p_i \in \mathbb{F}$, $1 \leq i \leq n$. *Let* $s := \sum p_i$ *and* $S := \sum |p_i|$. *If* $n\mathbf{u} < \frac{1}{2}$, *then*

$$|\mathtt{res} - s| \leq 2\mathbf{u}|s| + 2(1+2\mathbf{u})\gamma_n^2(2\mathbf{u})S \quad \text{with} \quad \gamma_n(2\mathbf{u}) = \frac{2n\mathbf{u}}{1-2n\mathbf{u}}. \tag{16}$$

Proof:

Let e_i be the error on the floating-point addition of π_{i-1} and p_i $(i = 2, ..., n)$:

$$\pi_i + e_i = \pi_{i-1} + p_i \tag{17}$$

From Proposition 4.3,

$$|e_i - q_i| \leq 2\mathbf{u}|e_i| \tag{18}$$

Because s is the exact addition of the n floating-point numbers p_i and π_n is the associated floating-point addition,

$$s = \sum_{i=1}^{n} p_i = \pi_n + \sum_{i=2}^{n} e_i \tag{19}$$

The error on the floating-point number `res` computed using Algorithm 3 with the random rounding mode is

$$|\mathtt{res} - s| = |\,\mathrm{fl}_*(\pi_n + \sigma_n) - s| \tag{20}$$

Therefore

$$|\mathtt{res} - s| = |(1+\varepsilon)(\pi_n + \sigma_n) - s| \quad \text{with} \quad |\varepsilon| \leq 2\mathbf{u} \tag{21}$$

and

$$|\mathtt{res} - s| = |(1+\varepsilon)(\pi_n + \sigma_n - s) + \varepsilon s| \tag{22}$$

From Equation (19),

$$|\mathtt{res} - s| = |(1+\varepsilon)(\sigma_n - \sum_{i=2}^{n} e_i) + \varepsilon s| \tag{23}$$

Therefore

$$|\mathtt{res} - s| \leq (1+2\mathbf{u})|\sigma_n - \sum_{i=2}^{n} e_i| + 2\mathbf{u}|s| \tag{24}$$

Let us evaluate an upper bound on $|\sigma_n - \sum_{i=2}^{n} e_i|$.

$$|\sigma_n - \sum_{i=2}^{n} e_i| \leq |\sigma_n - \sum_{i=2}^{n} q_i| + |\sum_{i=2}^{n} q_i - \sum_{i=2}^{n} e_i| \tag{25}$$

The error on σ_n is [7]

$$|\sigma_n - \sum_{i=2}^{n} q_i| \leq \gamma_{n-2}(2\mathbf{u}) \sum_{i=2}^{n} |q_i| \tag{26}$$

From Equation (18),

$$|\sum_{i=2}^{n} q_i - \sum_{i=2}^{n} e_i| \leq 2\mathbf{u} \sum_{i=2}^{n} |e_i| \tag{27}$$

Therefore from Equations (26) and (27),

$$|\sigma_n - \sum_{i=2}^{n} e_i| \leq \gamma_{n-2}(2\mathbf{u}) \sum_{i=2}^{n} |q_i| + 2\mathbf{u} \sum_{i=2}^{n} |e_i| \tag{28}$$

Let us first evaluate an upper bound on $\sum_{i=2}^{n} |e_i|$ and then an upper bound on $\sum_{i=2}^{n} |q_i|$. Let us show by induction that

$$\sum_{i=2}^{n} |e_i| \leq \gamma_{n-1}(2\mathbf{u}) \sum_{i=1}^{n} |p_i| \tag{29}$$

From Equation (17), we deduce that if $n = 2$,

$$\pi_2 + e_2 = \pi_1 + p_2 \quad \text{and} \quad \pi_1 = p_1 \tag{30}$$

Therefore

$$|e_2| \leq \gamma_1(2\mathbf{u}) \left(|p_1| + |p_2|\right) \tag{31}$$

Let us assume that Equation (29) is true for n and that an extra floating-point number p_{n+1} is added. Then

$$\pi_{n+1} = \mathrm{fl}_*(\pi_n + p_{n+1}) \tag{32}$$

$$\pi_{n+1} = \mathrm{fl}_* \left(\sum_{i=1}^{n+1} p_i \right) \tag{33}$$

From [7],

$$|\pi_{n+1}| \leq (1 + \gamma_n(2\mathbf{u})) \sum_{i=1}^{n+1} |p_i| \tag{34}$$

Let e_{n+1} be the error on the floating-point addition of π_n and p_{n+1}:

$$\pi_{n+1} + e_{n+1} = \pi_n + p_{n+1} \tag{35}$$

From Equation (34),

$$|e_{n+1}| \leq 2\mathbf{u}|\pi_{n+1}| \leq 2\mathbf{u} \left(1 + \gamma_n(2\mathbf{u})\right) \sum_{i=1}^{n+1} |p_i| \tag{36}$$

Hence, assuming that Equation (29) is true for n,

$$\sum_{i=2}^{n+1} |e_i| \leq (\gamma_{n-1}(2\mathbf{u}) + 2\mathbf{u}(1+\gamma_n(2\mathbf{u}))) \sum_{i=1}^{n+1} |p_i| \tag{37}$$

From Proposition 8.1 in the appendix, we deduce

$$\sum_{i=2}^{n+1} |e_i| \leq \gamma_n(2\mathbf{u}) \sum_{i=1}^{n+1} |p_i| \tag{38}$$

Therefore by induction Equation (29) is true.

Let us evaluate an upper bound on $\sum_{i=2}^{n} |q_i|$:

$$\sum_{i=2}^{n} |q_i| \leq \sum_{i=2}^{n} |e_i| + \sum_{i=2}^{n} |q_i - e_i| \tag{39}$$

From Equations (18) and (29),

$$\sum_{i=2}^{n} |q_i| \leq \gamma_{n-1}(2\mathbf{u}) \sum_{i=1}^{n} |p_i| + 2\mathbf{u} \sum_{i=2}^{n} |e_i| \tag{40}$$

From Equation (29),

$$\sum_{i=2}^{n} |q_i| \leq \gamma_{n-1}(2\mathbf{u}) \sum_{i=1}^{n} |p_i| + 2\mathbf{u}\gamma_{n-1}(2\mathbf{u}) \sum_{i=1}^{n} |p_i| \tag{41}$$

Therefore

$$\sum_{i=2}^{n} |q_i| \leq (\gamma_{n-1}(2\mathbf{u}) + 2\mathbf{u}\gamma_{n-1}(2\mathbf{u})) \sum_{i=1}^{n} |p_i| \tag{42}$$

From Proposition 8.2 in the appendix, we deduce

$$\sum_{i=2}^{n} |q_i| \leq \gamma_n(2\mathbf{u}) \sum_{i=1}^{n} |p_i| \tag{43}$$

From Equations (28), (29) and (43), we deduce

$$|\sigma_n - \sum_{i=2}^{n} e_i| \leq \gamma_{n-2}(2\mathbf{u})\gamma_n(2\mathbf{u}) \sum_{i=1}^{n} |p_i| + 2\mathbf{u}\gamma_{n-1} \sum_{i=1}^{n} |p_i| \tag{44}$$

Therefore

$$|\sigma_n - \sum_{i=2}^{n} e_i| \leq (\gamma_{n-2}(2\mathbf{u})\gamma_n(2\mathbf{u}) + 2\mathbf{u}\gamma_{n-1}(2\mathbf{u})) \sum_{i=1}^{n} |p_i| \tag{45}$$

From Proposition 8.3 in the appendix, we deduce

$$|\sigma_n - \sum_{i=2}^{n} e_i| \leq 2\gamma_n^2(2\mathbf{u}) \sum_{i=1}^{n} |p_i| \tag{46}$$

Therefore, from Equations (24) and (46),

$$|\texttt{res} - s| \leq 2\mathbf{u}|s| + 2(1+2\mathbf{u})\gamma_n^2(2\mathbf{u}) \sum_{i=1}^{n} |p_i| \tag{47}$$

□.

6 Numerical results

In the numerical experiment presented here, the sum of 200 randomly generated floating-point numbers is computed in double precision with the CADNA library using the `Sum` and the `FastCompSum` algorithms. In Figure 2, one can observe the number of exact significant digits of the results estimated by CADNA from Equation (1). Using the `Sum` algorithm, if the condition number increases, the number of exact significant digits of the result decreases and the result has no more correct digit for condition numbers greater than 10^{15}. Using the `FastCompSum` algorithm, as long as the condition number is less than 10^{15}, the compensated summation algorithm produces results with the maximal accuracy (15 exact significant digits in double precision). For condition numbers greater than 10^{15}, the accuracy decreases and there is no more correct digit for condition numbers greater than 10^{30}. The results provided by CADNA are consistent with well known properties of compensated summation algorithms [7]: with the current precision, the `FastCompSum` algorithm computes results that could have been obtained with twice the working precision.

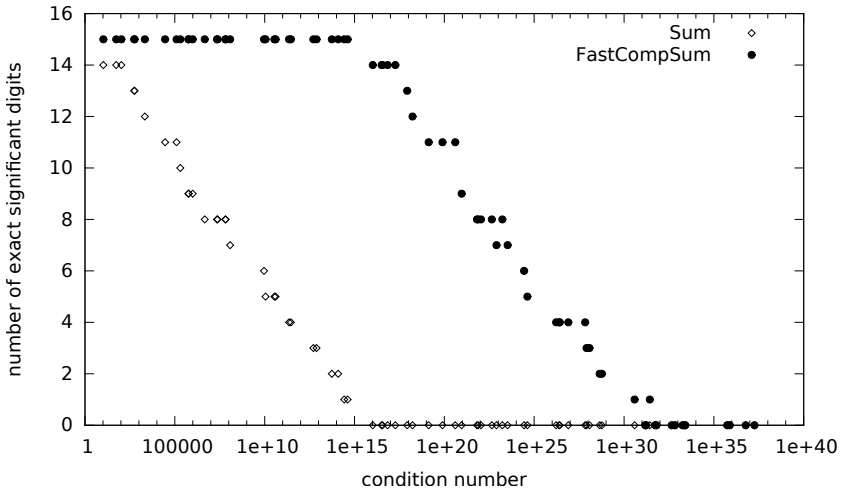

Fig. 2. Accuracy estimated by CADNA using the `Sum` and the `FastCompSum` algorithms with 200 randomly generated floating-point numbers

The number of numerical instabilities detected during the execution depends on the condition number. These numerical instabilities are of various types:

- using the `Sum` algorithm,
  - cancellation (subtraction of two very close values which generates a sudden loss of accuracy)

- using the `FastCompSum` algorithm,
  - cancellation
  - unstable branching (due to a non significant difference between the operands in a relational test)
  - non significant argument in the absolute value function.

Because no multiplication and no division is performed, no instability related to the self-validation of DSA can be detected.

Table 1 presents the execution times for the sum of 100 000 floating-point numbers computed in double precision. Execution times have been measured with and without CADNA on an Intel Core 2 quad Q9550 CPU at 2.83 GHz using g++ version 4.8.3. The code has been run using CADNA with two kinds of instability detection:

- the detection of all kinds of instabilities;
- no detection of instabilities. With this mode, the execution time can be considered the minimum that can be obtained whatever instability detection chosen. This mode is usually not recommended because it does not enable the self-validation of DSA. However, as previously mentioned, using summation algorithms, no instability can invalidate the estimation of accuracy.

From Table 1 the cost of the `FastCompSum` algorithm over the classic summation is about 6 without CADNA and varies from 4 to 9 with CADNA, depending on the level of instability detection. The cost of CADNA in terms of execution time varies from 10 to 15 if no instability detection is activated. This overhead is higher if any instability is detected because of the heavy cost of the cancellation detection.

Sum			
execution	instability detection	execution time (s)	ratio
IEEE	-	3.25E-04	1
CADNA	all instabilities	1.40E-02	43.2
	no instability	3.40E-03	10.5
FastCompSum			
execution	instability detection	execution time (s)	ratio
IEEE	-	2.00E-03	1
CADNA	all instabilities	6.11E-02	30.6
	no instability	2.98E-02	14.9

Table 1
Execution times with and without CADNA for the sum of 100 000 floating-point numbers

7 Conclusion and perspectives

In this article, we have shown that we can validate compensated summation based on the `FastTwoSum` algorithm with discrete stochastic arithmetic even if EFT are not valid as we use directed rounding modes. In a future article, we will generalize this analysis to other EFT like `TwoSum` and `TwoProduct`. Then we will see if we can still use discrete stochastic arithmetic for validating compensated algorithms for dot product and polynomial evaluation (compensated Horner scheme).

Acknowledgement

The authors wish to thank EDF (Electricité De France) for its financial support.

References

[1] F. Jézéquel and J.-M. Chesneaux. CADNA: a library for estimating round-off error propagation. *Computer Physics Communications*, 178(12):933–955, 2008.

[2] J. Vignes. Discrete Stochastic Arithmetic for validating results of numerical software. *Numerical Algorithms*, 37(1–4):377–390, December 2004.

[3] IEEE Computer Society. *IEEE Standard for Floating-Point Arithmetic*. IEEE Standard 754-2008, August 2008.

[4] J.-M. Chesneaux, S. Graillat, and F. Jézéquel. *Encyclopedia of Computer Science and Engineering*, volume 4, chapter Rounding Errors, pages 2480–2494. Wiley, 2009.

[5] W. Kahan. Further remarks on reducing truncation errors. *Comm. ACM*, 8:40, 1965.

[6] D. M. Priest. *On Properties of Floating Point Arithmetics: Numerical Stability and the Cost of Accurate Computations*. PhD thesis, Mathematics Department, University of California, Berkeley, CA, USA, November 1992. ftp://ftp.icsi.berkeley.edu/pub/theory/priest-thesis.ps.Z.

[7] N.J. Higham. *Accuracy and stability of numerical algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2002.

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

[9] T.J. Dekker. A floating-point technique for extending the available precision. *Numerische Mathematik*, 18(3):224–242, 1971.

[10] J. Vignes. A stochastic arithmetic for reliable scientific computation. *Mathematics and Computers in Simulation*, 35:233–261, 1993.

[11] J.-M. Chesneaux. *L'arithmétique stochastique et le logiciel CADNA*. Habilitation à diriger des recherches, Université Pierre et Marie Curie, Paris, France, November 1995.

[12] J. Vignes. Zéro mathématique et zéro informatique. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 303:997–1000, 1986. also: La Vie des Sciences, 4 (1) 1-13, 1987.

[13] J.-M. Chesneaux and J. Vignes. Les fondements de l'arithmétique stochastique. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 315:1435–1440, 1992.

[14] J.-M. Muller, N. Brisebarre, F. de Dinechin, C.-P. Jeannerod, V. Lefèvre, G. Melquiond, N. Revol, D. Stehlé, and S. Torres. *Handbook of Floating-Point Arithmetic*. Birkhäuser, Boston, 2010.

[15] J. Demmel and H. D. Nguyen. Fast reproducible floating-point summation. In *21st IEEE Symposium on Computer Arithmetic, Austin, TX, USA, April 7-10*, pages 163–172, 2013.

[16] P.H. Sterbenz. *Floating-point computation*. Prentice-Hall series in automatic computation. Prentice-Hall, 1973.

[17] J.R. Hauser. Handling floating-point exceptions in numeric programs. *ACM Trans. Program. Lang. Syst.*, 18(2):139–174, 1996.

8 Appendix

The same notations as in Section 2 are used. We consider a precision-p binary floating-point system. Let $\mathbf{u} = 2^{-p}$ and $\gamma_n(2\mathbf{u}) = \frac{2n\mathbf{u}}{1-2n\mathbf{u}}$. Let us assume that $n\mathbf{u} < \frac{1}{2}$.

Proposition 8.1

$$\gamma_{n-1}(2\mathbf{u}) + 2\mathbf{u}(1 + \gamma_n(2\mathbf{u})) \leq \gamma_n(2\mathbf{u}) \tag{48}$$

Proof:

$$\gamma_{n-1}(2\mathbf{u}) \leq \frac{2(n-1)\mathbf{u}}{1-2n\mathbf{u}} \tag{49}$$

and

$$2\mathbf{u}(1 + \gamma_n(2\mathbf{u})) = \frac{2\mathbf{u}}{1-2n\mathbf{u}} \tag{50}$$

Therefore from Equations (49) and (50), we deduce

$$\gamma_{n-1}(2\mathbf{u}) + 2\mathbf{u}(1 + \gamma_n(2\mathbf{u})) \leq \frac{2n\mathbf{u}}{1-2n\mathbf{u}} \tag{51}$$

□.

Proposition 8.2

$$\gamma_n(2\mathbf{u}) + 2\mathbf{u}\gamma_n(2\mathbf{u}) \leq \gamma_{n+1}(2\mathbf{u}) \tag{52}$$

Proof:

Because $n\mathbf{u} < \frac{1}{2}$,

$$\gamma_n(2\mathbf{u}) < \frac{1}{1-2n\mathbf{u}} \tag{53}$$

Therefore

$$\gamma_n(2\mathbf{u}) + 2\mathbf{u}\gamma_n(2\mathbf{u}) < \gamma_n(2\mathbf{u}) + \frac{2\mathbf{u}}{1-2n\mathbf{u}} \tag{54}$$

and

$$\gamma_n(2\mathbf{u}) + 2\mathbf{u}\gamma_n(2\mathbf{u}) < \frac{2(n+1)\mathbf{u}}{1-2n\mathbf{u}} \tag{55}$$

Therefore we can deduce Equation (52). □.

Proposition 8.3

$$\gamma_{n-2}(2\mathbf{u})\gamma_n(2\mathbf{u}) + 2\mathbf{u}\gamma_{n-1}(2\mathbf{u}) \leq 2\gamma_n^2(2\mathbf{u}) \tag{56}$$

Proof:

$$\gamma_n(2\mathbf{u}) - 2\mathbf{u} = \frac{2n\mathbf{u} - 2\mathbf{u} + 4n\mathbf{u}^2}{1-2n\mathbf{u}} \tag{57}$$

Because $1 - 2n\mathbf{u} > 0$ and $2n\mathbf{u} + 4n\mathbf{u}^2 > 2\mathbf{u}$, $\gamma_n(2\mathbf{u}) - 2\mathbf{u} > 0$.

Therefore

$$2\mathbf{u} < \gamma_n(2\mathbf{u}) \tag{58}$$

Furthermore, because $\gamma_{n-1}(2\mathbf{u}) \leq \gamma_n(2\mathbf{u})$, we can deduce Equation (56). □.