Applied Mathematics and Computation 329 (2018) 339–363

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Numerical validation of compensated algorithms with stochastic arithmetic

Stef Graillat[a,*], Fabienne Jézéquel[a,b], Romain Picot[a,c]

[a] Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, Paris F-75005, France
[b] Université Panthéon-Assas, 12 place du Panthéon, Paris, F-75005, France
[c] EDF R&D, 7 boulevard Gaspard Monge, Palaiseau, F-91120, France

ARTICLE INFO

Keywords:
CADNA
Compensated algorithms
Discrete stochastic arithmetic
Error-free transformations
Floating-point arithmetic
Numerical validation
Rounding errors

ABSTRACT

Compensated algorithms consist in computing the rounding errors of individual operations and then adding them later on to the computed result. This makes it possible to increase the accuracy of the computed result efficiently. Computing the rounding error of an individual operation is possible through the use of a so-called *error-free transformation*. In this article, we show that it is possible to validate the result of compensated algorithms using stochastic arithmetic. We study compensated algorithms for summation, dot product and polynomial evaluation. We prove that the use of the random rounding mode inherent to stochastic arithmetic does not change much the accuracy of compensated methods. This is due to the fact that error-free transformations are no more exact but still sufficiently accurate to improve the numerical quality of results.

1. Introduction

Computing power rapidly increases and Exascale computing (10^{18} floating-point operations per second) should be reached in a few years. Such a computing power also means a large number of rounding errors. Indeed, nearly all floating-point operations imply a small rounding which can accumulate along the computation and finally an incorrect result may be produced. As a consequence, it is fundamental to be able to give some information about the numerical quality of the computed results. By numerical quality, we mean here the number of significant digits of the computed result that are not affected by rounding errors.

A well-known solution to assert the numerical quality is to use the numerical library called CADNA[1] that implements Discrete Stochastic Arithmetic (DSA) and makes it possible to provide a confidence interval of the computed result [1–3]. DSA requires several executions of the user program with a random rounding mode that consists in rounding any result to plus or minus infinity with the same probability.

If the accuracy of the computed result is not sufficient, it is necessary to increase the precision of the computation. For simple enough calculations, a possible technique is the use of *compensated algorithms* (see [4]). These algorithms are based on error-free transformations (EFTs) that make it possible to compute the rounding errors of some elementary operations like addition and multiplication exactly. We now assume a floating-point arithmetic adhering to the IEEE 754-2008

* Corresponding author.
E-mail addresses: stef.graillat@lip6.fr (S. Graillat), fabienne.jezequel@lip6.fr (F. Jézéquel), romain.picot@lip6.fr (R. Picot).
[1] URL address: http://cadna.lip6.fr

https://doi.org/10.1016/j.amc.2018.02.004
0096-3003/

standard [5]. In that case, when using rounding to nearest, the rounding error of an addition is a floating-point number that can be computed exactly via an EFT. But EFTs are no longer valid when used with directed rounding (rounding to plus or minus infinity). Indeed, if we use directed rounding, the error of a floating-point addition is not necessarily a floating-point number. However, directed rounding is required in DSA. As a consequence, it is not clear whether we can use DSA to validate some numerical codes that heavily rely on the use of error-free transformations.

In this article, we show that compensated algorithms enable one to increase the results accuracy even with directed rounding. Whatever the faithful rounding mode chosen, compensated summation, dot product, and Horner scheme algorithms provide a result almost as accurate as if it was computed with twice the working precision. Concerning compensated summation, although part of this work has been done in [6], for completeness some previously obtained results are recalled in Section 3. We also show in this article that compensated algorithms *as in K*-fold precision for summation and dot product [7] provide satisfactory results even with directed rounding: the accuracy obtained is almost as if the working precision was multiplied by *K*. Therefore compensated algorithms can improve accuracy even if they are executed with DSA. Furthermore DSA enables one to estimate the numerical quality of results of compensated algorithms. This satisfactory behavior of compensated algorithms with DSA is confirmed by the numerical experiments described in this article.

This outline of this article is as follows. In Section 2 we give some definitions and notations used in the sequel. In the next sections, we show the impact of a directed rounding mode on compensated algorithms. Sections 3–7 are successively devoted to the error analysis with directed rounding of compensated summation, compensated dot product, compensated Horner scheme, summation *as in K*-fold precision, and dot product *as in K*-fold precision. Finally, numerical experiments carried out using the CADNA library are presented in Section 8.

2. Definitions and notations

In this paper, we assume to work with a binary floating-point arithmetic adhering to IEEE 754–2008 floating-point standard [5] and we suppose that no overflow occurs. The error bounds for the compensated summation that are presented in Sections 3 and 6 remain valid in the presence of underflow. For the other compensated algorithms considered in this article (dot product and Horner scheme) we assume that no underflow occurs so as to present simpler error bounds.

The set of floating-point numbers is denoted by $\mathbb{F}$, the bound on relative error for round to nearest by $\mathbf{u}$. With the IEEE 754 *binary64* format (double precision), we have $\mathbf{u} = 2^{-53}$ and with the *binary32* format (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 [8]

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

$$\mathrm{fl}_*(a \circ b) = (a \circ b)(1+\varepsilon_1) = (a \circ b)/(1+\varepsilon_2) \text{ for } \circ = \{+,-\} \text{ and } |\varepsilon_v| \leq 2\mathbf{u}. \tag{2.1}$$

As a consequence,

$$|a \circ b - \mathrm{fl}_*(a \circ b)| \leq 2\mathbf{u}|a \circ b| \text{ and } |a \circ b - \mathrm{fl}_*(a \circ b)| \leq 2\mathbf{u}|\mathrm{fl}_*(a \circ b)| \text{ for } \circ = \{+,-\}. \tag{2.2}$$

We use standard notations for error estimations. The quantities γ_n are defined as usual [8] by

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

where it is implicitly assumed that $n\mathbf{u} < 1$.

To keep track of the $(1+\varepsilon)$ factors in our error analysis, we use the relative error counters introduced by Stewart [9]. For a positive integer n, $\langle n \rangle$ denotes the following product

$$\langle n \rangle(\mathbf{u}) = \prod_{i=1}^{n} (1+\varepsilon_i)^{\rho_i} \quad \text{with} \quad \rho_i = \pm 1 \quad \text{and} \quad |\varepsilon_i| \leq \mathbf{u} \quad (i = 1, \ldots, n).$$

The relative error counters satisfy $\langle j \rangle(\mathbf{u})\langle k \rangle(\mathbf{u}) = \langle j \rangle(\mathbf{u})/\langle k \rangle(\mathbf{u}) = \langle j+k \rangle(\mathbf{u})$. When $\langle n \rangle$ denotes any error counter, then there exists a quantity θ_n such that

$$\langle n \rangle(\mathbf{u}) = 1 + \theta_n(\mathbf{u}) \quad \text{and} \quad |\theta_n(\mathbf{u})| \leq \gamma_n(\mathbf{u}).$$

Remark 1. We give the following relations on γ_n, that will be frequently used in the sequel of the paper. For any positive integer n,

$$n\mathbf{u} \leq \gamma_n(\mathbf{u}), \quad \gamma_n(\mathbf{u}) \leq \gamma_{n+1}(\mathbf{u}), \quad (1+\mathbf{u})\gamma_n(\mathbf{u}) \leq \gamma_{n+1}(\mathbf{u}), \quad 2n\mathbf{u}(1+\gamma_{2n-2}(\mathbf{u})) \leq \gamma_{2n}(\mathbf{u}).$$

Remark 2. Recently, it has been shown that classic Wilkinson-type error bounds for summation, dot product and polynomial evaluation [10–12] can be slightly improved by replacing the factor $\gamma_n(\mathbf{u})$ by $n\mathbf{u}$ with no condition on n (for summation and dot product). In the sequel, it is likely that all the error bounds could also be improved by replacing all the $\gamma_n(\mathbf{u})$ by $n\mathbf{u}$. However, it is clear that $\gamma_n(\mathbf{u})$ is very close to $n\mathbf{u}$. Moreover, the proofs for improving the bounds would be more complicated and tricky, and would not be useful for the paper. We just aim at showing that the relative accuracy is in $\mathcal{O}(\mathbf{u})$ for classic algorithms and in $\mathcal{O}(\mathbf{u}^2)$ for compensated algorithms with directed roundings.