A Library for Symbolic Floating-Point Arithmetic

Claude-Pierre Jeannerod, Nicolas Louvet, Jean-Michel Muller, Antoine Plet

A Library for Symbolic Floating-Point Arithmetic

Claude-Pierre Jeannerod (Inria),
Nicolas Louvet (UCBL),
Jean-Michel Muller (CNRS),
Antoine Plet (ENS de Lyon).

(firstname.lastname@ens-lyon.fr)

LIP laboratory (CNRS, ENS de Lyon, Inria, UCB Lyon 1),
Université de Lyon, France.

August 3, 2016

Abstract

To analyze a priori the accuracy of an algorithm in floating-point arithmetic, one usually derives a uniform error bound on the output, valid for most inputs and parametrized by the precision p. To show further that this bound is sharp, a common way is to build an input example for which the error committed by the algorithm comes close to that bound, or even attains it. Such inputs may be given as floating-point numbers in one of the IEEE standard formats (say, for $p = 53$) or, more generally, as expressions parametrized by p, that can be viewed as *symbolic floating-point* numbers. With such inputs, a sharpness result can thus be established for virtually all reasonable formats instead of just one of them. This, however, requires the ability to run the algorithm on those inputs and, in particular, to compute the correctly-rounded sum, product, or ratio of two symbolic floating-point numbers. The goal of this paper is to show how these basic arithmetic operations can be performed automatically. We introduce a way to model symbolic floating-point data, and present algorithms for round-to-nearest addition, multiplication, fused multiply-add, and division. An implementation as a Maple library is also described, and experiments using examples from the literature are provided to illustrate its interest in practice.

1 Introduction

When designing floating-point algorithms that are building blocks for numerical computing (for example, arithmetic operations on complex numbers, or the evaluation of small-degree polynomials), it is important to provide rigorous and sharp error bounds, that can be safely used for analyzing the behavior of larger algorithms based on them.

To establish the sharpness of an error bound for a floating-point algorithm, one way is to provide input examples for which the error generated is very close to, or reaches the bound. A possible approach to find such examples is to run the algorithm on a set of random inputs, or to perform exhaustive tests in small precisions. However, since the number of possible inputs is exponential in the precision p, one can hardly expect to find examples in practical precisions ($p = 24$ or 53 for the commonly used binary32 and binary64 formats [7]) with such approaches. Hence, it is common to build examples parametrized by the precision p, and to show that the error committed by the algorithm either attains the error bound, or is equivalent to the error bound as $p \to \infty$; see for example [3, 8, 9].

Let us cite an example taken from [8]. In that paper, the authors analyze the algorithm below, due to Kahan (Algorithm 1), for evaluating $x = ad - bc$, where a, b, c, d are floating-point numbers, and assuming that a fused multiply-add instruction is available. Here and hereafter, RN denotes the function that rounds a real to the floating-point number nearest to it and, in case of tie, whose last significand digit is even.

Assuming base β and precision p, [8] show the following result: if no underflow or overflow occurs, then $|\widehat{x} - x| \leqslant 2u|x|$ with $u = \frac{1}{2}\beta^{1-p}$ the unit roundoff; moreover, for any fixed, even value of β

Algorithm 1 (Evaluation of $x = ad - bc$)

algorithm Kahan(a, b, c, d)
$\widehat{w} \leftarrow \mathrm{RN}(bc)$;
$e \leftarrow \mathrm{RN}(\widehat{w} - bc)$;
$\widehat{f} \leftarrow \mathrm{RN}(ad - \widehat{w})$;
$\widehat{x} \leftarrow \mathrm{RN}(\widehat{f} + e)$;

this relative error bound is asymptotically optimal as $p \to \infty$. In particular, to prove asymptotic optimality the authors introduce the following floating-point inputs, parametrized by β and p:

$$\begin{aligned} a = b &= \beta^{p-1} + 1, \\ c &= \beta^{p-1} + \tfrac{\beta}{2}\beta^{p-2}, \\ d &= 2\beta^{p-1} + \tfrac{\beta}{2}\beta^{p-2}; \end{aligned} \tag{1}$$

then, they show that for such inputs the relative error on the final result has the form

$$\frac{2u}{1+2u},$$

which is equivalent to $2u$ as $u \to 0$ (or, β being fixed, as $p \to \infty$). Note that in this example, the inputs a, b, c, d are expressed as functions of the base β and the precision p.

Examples as in (1) can be built by first computing the error generated by the algorithm with small values of the precision p, then guessing some "generic" inputs parametrized by β and p, and finally carrying paper-and-pencil calculations, that are in general tedious and error prone, to check if those guessed inputs effectively correspond to cases we are looking for. Our goal in this paper is to show how to automate this last step in order to save time, to avoid possible errors, and to be able to try many more candidates.

We refer to functions representing floating-point numbers parametrized by the precision p (with $\beta \geqslant 2$ fixed and even) as *symbolic floating-point numbers*. In this paper, we propose to manipulate symbolic floating-point numbers, in a computer algebra system such as Maple, very much as real numbers or polynomials can be manipulated within such a system.

Floating-point arithmetic has already been formalized in various interactive theorem provers: see for example [6, 4], and more recently the description of the Flocq library in [2], that was designed for the Coq proof assistant. Such a formalization can be used to generate formal proofs of properties of floating-point algorithms and programs. Also, multiple-precision floating-point computations can be performed using Flocq for verification purposes (see [10] for details), but in that case the precision for each operation is fixed: a particular format with a fixed precision $p \in \mathbb{N}$ is defined for each intermediate operation. In contrast, when computing with symbolic floating-point numbers as we are going to define them in this paper, the precision p is a linear function of an integer variable k, which allows us to deal with "generic" examples as in (1).

Let us briefly recall how conventional floating-point arithmetic can be formalized. The set $\mathbb{F}_p$ of floating-points numbers (with an unbounded exponent range), in base β and precision p, is a subset of $\mathbb{Q}$ containing all the numbers of the form $M \cdot \beta^e$, with $|M| \in \mathbb{Z}$ less than β^p, and $e \in \mathbb{Z}$. The result of an elementary operation ($+$, $-$, $\times$ or $/$) between two elements of $\mathbb{F}_p$ is a rational number in $\mathbb{Q}$, and the computed result is the exact one rounded to an element in $\mathbb{F}_p$ according to a specified rounding function.

In a similar way, we first define in this paper the set $\mathbb{L}$ of affine functions that will be used to represent both the precision and the exponent of symbolic floating-point numbers. We also define the set $\mathbb{SQ}$ that will be used to represent the exact results of arithmetic operations before rounding. Then, we introduce the set $\mathbb{SZ}$ of symbolic integers, and we show how elements in $\mathbb{SQ}$ can be rounded to elements in $\mathbb{SZ}$. Using the notion of symbolic integer, we finally define the set $\mathbb{SF}_p$ of symbolic floating-point numbers in base β and precision $p \in \mathbb{L}$. The following table summarizes the matching between numerical data and their symbolic counterparts:

Numerical data	Symbolic data
$\mathbb{Z}$	$\mathbb{L}$
$\mathbb{Q}$	$\mathbb{SQ}$
$\mathbb{Z}$	$\mathbb{SZ}$
$\mathbb{F}_p$	$\mathbb{SF}_p$

The exact results of elementary operations on elements of $\mathbb{SF}_p$ can always be represented in $\mathbb{SQ}$, and then rounded to an element of $\mathbb{SF}_p$. We will focus in this paper on rounding to the nearest symbolic floating-point numbers, but the directed rounding modes are provided as well in the library.

Outline. In this paper, we first define in Section 2 the set $\mathbb{SQ}$ as the fraction field of sums of exponentials functions of a symbolic integer variable k, that will serve as the input domain of the rounding function, and we give properties that will be used for handling the elements of $\mathbb{SQ}$. In Section 3, we define the set $\mathbb{SZ}$ of symbolic integers as the subset of elements in $\mathbb{SQ}$ that evaluates to integers if k is large enough. Considering the precision p as a linear function of k, we introduce in Section 4 the set $\mathbb{SF}_p$ of symbolic floating-point numbers in precision p. We then define a round-to-nearest partial function RN_p from $\mathbb{SQ}$ to $\mathbb{SF}_p$, we give an algorithm to compute it in practice, and we show how it matches the roundTiesToEven rounding-direction attribute of the IEEE 754 standard for floating-point arithmetic [7]. An implementation as a Maple library is described in Section 5, and experiments using examples from the literature are provided to illustrate its usefulness. We conclude in Section 6 with some practical time measurements.

Notation. Here and hereafter, k is an integer variable and β is a fixed integer corresponding to the floating-point base. We assume that β is even and at least two, and in practice we shall take $\beta = 2$ or $\beta = 10$ as prescribed by the IEEE 754-2008 standard [7].

We write $\mathbb{Z}$ and $\mathbb{N}$ to denote the integers and the nonnegative integers, respectively. To indicate that a property holds asymptotically, we will often write "$P(k)$ as $k \to \infty$" as a shortcut for the fact that "there exists $k_0 \in \mathbb{N}$ such that $P(k)$ holds for all $k \geqslant k_0$."

Let $\mathbb{L}$ denote the set of linear functions in k with integer coefficients:

$$\mathbb{L} = \{ak + b \,:\, a, b \in \mathbb{Z}\};$$

let also

$$\mathbb{L}_{\geqslant \tau} = \{\ell \in \mathbb{L} \,:\, \ell(k) \geqslant \tau \text{ as } k \to \infty\}, \qquad \tau \in \mathbb{R}.$$

Given a function $g : \mathcal{X} \to \mathbb{R}$ such that $\min_{x \in \mathcal{X}} g(x)$ is well defined, we denote by $\arg\min_{x \in \mathcal{X}} g(x)$ the set

$$\big\{x^* \in \mathcal{X} \;:\; g(x^*) = \min_{x \in \mathcal{X}} g(x)\big\}.$$

2 Fractions of sums of exponentials with base β

Before introducing symbolic integers and floating-point numbers, we have to formalize a more general set $\mathbb{SQ}$ which will be the domain of the exact calculations. In this section, we first define the set $\mathbb{E}$ of sums of exponentials in base β, with integer coefficients and exponents linear in k. We then define $\mathbb{SQ}$ as the set of fractions of elements of $\mathbb{E}$ and derive several useful properties of that set.

2.1 Sums of exponentials

Let $\mathbb{E}$ be the set of sums of exponentials

$$f(k) = \sum_{i=1}^{n} c_i \beta^{e_i(k)}, \qquad k \in \mathbb{N}, \tag{2}$$

where $n \in \mathbb{N}$ and where, for $n \geqslant 1$, the c_i and e_i satisfy

$$|c_i| \in \{1, 2, \ldots, \beta - 1\} \quad \text{and} \quad e_i \in \mathbb{L}, \tag{3}$$

$$e_1(k) > e_2(k) > \cdots > e_n(k) \quad \text{as } k \to \infty; \tag{4}$$

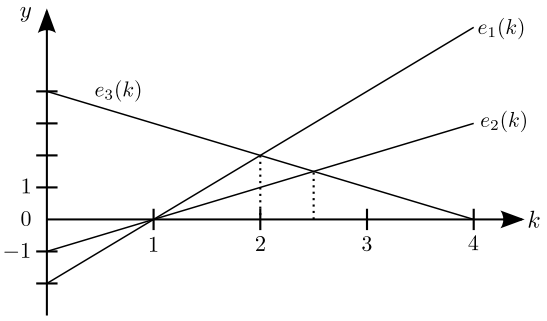

Figure 1: The linear functions $e_1(k) = 2k - 2$, $e_2(k) = k - 1$, and $e_3(k) = -k + 4$ such that $e_1(k) > e_2(k) > e_3(k)$ for all integer $k \geqslant 3$.

for $n = 0$, we take $f(k) = 0$.

If we define the ring $R = \mathbb{Z}[\beta^{-1}] = \{n/\beta^e : n \in \mathbb{Z}, e \in \mathbb{N}\}$ of β-adic fractions, which generalizes dyadic fractions to values of β other than 2, then the elements of $\mathbb{E}$ are functions from $\mathbb{N}$ to R, and can be seen as special cases of so-called *exponential polynomials*, with base β and exponents linear in k; see for example [5, p. 7].

The asymptotic ordering of these linear exponents is illustrated by Figure 1 for $n = 3$. Writing

$$e_i(k) = a_i k + b_i, \qquad i = 1, \ldots, n, \tag{5}$$

we see in particular that the ordering of the e_i in (4) implies

$$a_1 \geqslant a_2 \geqslant \cdots \geqslant a_n. \tag{6}$$

For later use, we give the following property that provides an asymptotic equivalent of any nonzero element in $\mathbb{E}$ as $k \to \infty$.

Property 2.1. *Let $f \in \mathbb{E}$ as in (2) and (5) with $n \geqslant 1$, and let $c \in \mathbb{Q}$ such that*

$$c = \sum_{i\,:\,a_i = a_1} c_i \beta^{b_i - b_1}. \tag{7}$$

Then

$$f(k) \sim c\,\beta^{e_1(k)} \qquad \text{as } k \to \infty.$$

Proof. We use (6) to rewrite $f(k)$ as

$$f(k) = \sum_{i\,:\,a_i = a_1} c_i \beta^{e_i(k)} + \sum_{i\,:\,a_i < a_1} c_i \beta^{e_i(k)} = \left(c + \sum_{i\,:\,a_i < a_1} c_i \beta^{e_i(k) - e_1(k)} \right) \beta^{e_1(k)},$$

with c as in (7). Note that the c_i are independent of k, and that for $a_i < a_1$, $e_i(k) - e_1(k)$ tends to $-\infty$ as $k \to \infty$. Hence the ratio $f(k)/\beta^{e_1(k)}$ tends to c as $k \to \infty$, which establishes the result. □

Representations for $\mathbb{E}$ There are at least two ways of representing the functions in $\mathbb{E}$. A first possible representation is directly suggested by (2) and (5), and consists of the finite sum

$$\sum_{i=1}^{n} c_i \beta^{a_i k + b_i}. \tag{8}$$

Because of the redundancy in the digit set used for the c_i, two or more different sequences of terms can represent the same function. For example, since $\beta^k - \beta^{k-1} = (\beta - 1)\beta^{k-1}$, one can also represent $f(k) = 2^{k-1}$ by $2^k - 2^{k-1}$. This is of course similar to classical redundant, signed-digit number systems *à la* Avizienis [1].

An alternative representation consists of using the change of variable $\beta^k = X$, and identifying $f \in \mathbb{E}$ with the Laurent polynomial

$$\widetilde{f}(X) = \sum_{i=1}^{n} c_i \beta^{b_i} X^{a_i}. \tag{9}$$

More precisely, since $a_i, b_i, c_i \in \mathbb{Z}$, we have $\widetilde{f} \in R[X, X^{-1}]$. By expanding the positive elements of R in base β, it is easy to check that the map $\varphi : f \mapsto \widetilde{f}$ is bijective.

Both representations will be useful in this paper, and we shall use one or the other depending on what is most convenient for the operation at hand.

Example 2.2. *We introduce two examples in base* 2 *that will be used repeatedly:*

$$\xi_1 = 2^{2k} - 5 \cdot 2^{k-1} = 2^{2k} - 2^{k+1} - 2^{k-1} \quad \text{and} \quad \xi_2 = -2^k + 5 \cdot 2^{-1} - 3 \cdot 2^{-k} \tag{10}$$

are sums of exponentials that can be represented by the polynomials

$$\widetilde{\xi_1}(X) = X^2 - \frac{5}{2}X \quad \text{and} \quad \widetilde{\xi_2}(X) = -X + \frac{5}{2} - \frac{3}{X}$$

using the change of variable $2^k = X$*. One should notice that the polynomial representation gathers the terms in the sum of exponentials that contain the same slope in their exponents.*

Addition and multiplication in $\mathbb{E}$ Since the map φ is bijective and since $R[X, X^{-1}]$ is a commutative ring with 1 (for the usual + and $\times$ on polynomials), the set $\mathbb{E}$ inherits the same algebraic structure, with addition and multiplication defined as

$$f_1 \operatorname{op} f_2 = \varphi^{-1}\big(\varphi(f_1) \operatorname{op} \varphi(f_2)\big)$$

for $f_1, f_2 \in \mathbb{SQ}$ and $\operatorname{op} = +, \times$.

In practice, this implies that the system $\mathbb{E}$ is closed under addition and multiplication, and that operating on two functions $f_1, f_2 \in \mathbb{E}$ given in the representation (8) can be done simply by converting them into their polynomial counterparts (9), operating on the latter using standard polynomial arithmetic, and converting the result back from (9) to (8).

2.2 The fraction field $\mathbb{SQ}$

Since $\mathbb{E}$ is an integral ring, we define $\mathbb{SQ}$ as the field of fractions of $\mathbb{E}$. In particular, $\mathbb{E} \subset \mathbb{SQ}$ but $2^k/3$ and $(2^k+1)/(1-2^{1-k})$ are elements of $\mathbb{SQ}$ that do not belong to $\mathbb{E}$. One should notice that the denominator of an element of $\mathbb{SQ}$ may evaluate to zero for finitely many values of k. However, any $f \in \mathbb{SQ}$ is a well defined function from $\mathbb{N}_{\geqslant k_0}$ to $\mathbb{Q}$ for some $k_0 \in \mathbb{N}$, depending on f. For instance, for $(2^k+1)/(1-2^{1-k})$ in the previous example, the domain of definition is $\mathbb{N}_{>1}$.

Moreover, since φ is a ring isomorphism from $\mathbb{E}$ to $R[X, X^{-1}]$, it can be extended to a field isomorphism from $\mathbb{SQ}$ to $\operatorname{Frac}(R[X, X^{-1}]) = R(X) = \mathbb{Z}(X)$, so that elements in $\mathbb{SQ}$ can be represented by rational functions with integer coefficients.

Example 2.3. *The fractions*

$$\xi_3 = \frac{2 + 22 \cdot 2^{-k}}{3} \quad \text{and} \quad \xi_4 = \frac{-2^{3k} - 5 \cdot 2^{2k-1}}{2^{6k} + 2^{5k+1}} \tag{11}$$

belong to $\mathbb{SQ}$ *as ratios of elements of* $\mathbb{E}$*, and can be represented respectively by*

$$\widetilde{\xi_3}(X) = \frac{2X + 22}{3X} \quad \text{and} \quad \widetilde{\xi_4}(X) = \frac{-2X - 5}{2X^4 + 4X^3}.$$

As ξ_1 *and* ξ_2*, these two examples are used as running examples in the following.*

2.3 Sign, absolute value, and exponent of elements of $\mathbb{SQ}$

Besides $+$, $-$, $\times$ and $\div$, it is useful to define several low-level functions on elements of $\mathbb{SQ}$. Especially, the rational numbers have a *sign*, and when they are nonzero, they also have a base-β *exponent* (the base-β exponent of $x \neq 0$ is the largest integer e such that $\beta^e \leqslant |x|$). These low-level functions are essential when defining and manipulating conventional floating-point numbers, we will also need them to deal with symbolic floating-point numbers.

Given a nonzero $f = g/h \in \mathbb{SQ}$, with $g, h \in \mathbb{E}$, let $g(k) \sim c_g \beta^{e_{g,1}(k)}$ and $h(k) \sim c_h \beta^{e_{h,1}(k)}$ be the asymptotic equivalents of g and h obtained according to Property 2.1. We have

$$f(k) = \frac{g(k)}{h(k)} \sim \frac{c_g}{c_h} \beta^{e_{g,1}(k) - e_{h,1}(k)} \quad \text{as } k \to \infty, \tag{12}$$

which shows that any nonzero function $f \in \mathbb{SQ}$ is eventually either positive or negative, that is, there exists $k_0 \in \mathbb{N}$ such that either $f(k) > 0$ for all $k \geqslant k_0$, or $f(k) < 0$ for all $k \geqslant k_0$. Hence there is a unique $s \in \{-1, 1\}$ such that $s = \operatorname{sign}(f(k))$ as $k \to \infty$. This allows us to define the function $\operatorname{sign} : \mathbb{SQ} \to \{-1, 0, 1\}$ as follows:

$$\operatorname{sign}(f) = \begin{cases} \operatorname{sign}\left(\frac{c_g}{c_h}\right) & \text{if } f = \frac{g}{h} \in \mathbb{SQ} - \{0\}, \\ 0 & \text{if } f = 0. \end{cases}$$

Consequently, we can define further the comparison operations $=, \neq, \leqslant, <, \geqslant, >$ for two elements of $\mathbb{SQ}$ by computing the sign of their difference. Also, recalling that $\mathbb{SQ}$ is a commutative field with 1, we get an absolute value by multiplying any element of $\mathbb{SQ}$ by its sign:

$$|f| = \operatorname{sign}(f) f.$$

Using the comparisons and absolute value in $\mathbb{SQ}$, we have the following property.

Property 2.4. *Given $f = g/h \in \mathbb{SQ} - \{0\}$, there exists a unique $e \in \mathbb{L}$ satisfying*

$$\beta^e \leqslant |f| < \beta^{e+1}.$$

Moreover, given $c_g, c_h \in \mathbb{Q}$ and $e_{g,1}, e_{h,1} \in \mathbb{L}$ as in (12), let e_c be the unique integer such that $\beta^{e_c} \leqslant |c_g/c_h| < \beta^{e_c+1}$. Then we have

$$e = \begin{cases} e_{g,1} - e_{h,1} + e_c - 1 & \text{if } |f| < \beta^{e_{g,1} - e_{h,1} + e_c}, \\ e_{g,1} - e_{h,1} + e_c & \text{otherwise.} \end{cases} \tag{13}$$

Proof. Property 2.1 and (12) imply that $|f| = |c_g/c_h| \beta^{e_{g,1} - e_{h,1}} (1 + \epsilon)$ with $\epsilon(k) \to 0$ as $k \to \infty$. Hence

$$|f| = \beta^{e_{g,1} - e_{h,1}} \left(\beta^{e_c - 1} + \mu\right) = \beta^{e_{g,1} - e_{h,1}} \left(\beta^{e_c + 1} + \nu\right),$$

with $\mu = |c_g/c_h|(1 + \epsilon) - \beta^{e_c - 1}$ and $\nu = |c_g/c_h|(1 + \epsilon) - \beta^{e_c + 1}$. By definition of e_c, since $\beta > 1$, we have $\beta^{e_c - 1} < |c_g/c_h| < \beta^{e_c + 1}$, hence $\nu(k) < 0 < \mu(k)$ as $k \to \infty$. Then, we have

$$\beta^{e_{g,1} - e_{h,1} + e_c - 1} < |f| < \beta^{e_{g,1} - e_{h,1} + e_c + 1},$$

and the existence of e as defined above follows from comparing $\beta^{e_{g,1} - e_{h,1} + e_c}$ to $|f|$.

Since the elements of $\mathbb{L}$ are integer-valued, if $e' \in \mathbb{L}$ satisfies $\beta^{e'} \leqslant |f| < \beta^{e'+1}$, then $e'(k) = e(k)$ as $k \to \infty$, so that $e' = e$, which shows the uniqueness of e. □

Property 2.4 allows us to take the exponent of any element of $\mathbb{SQ}$ by defining the following function $\mathbb{SQ} \to \mathbb{L}$:

$$\operatorname{exponent}(f) = \begin{cases} e \text{ as in (13)} & \text{if } f \in \mathbb{SQ} - \{0\}, \\ 0 & \text{if } f = 0. \end{cases}$$

Example 2.5. *We consider again ξ_1 and ξ_4 given by* (10) *and* (11) *respectively. Since $\xi_1(k) > 0$ for $k \geqslant 2$, we have sign$(\xi_1) = 1$, and since $2^{2k-1} \leqslant \xi_1(k) < 2^{2k}$ for $k \geqslant 3$, we have* exponent$(\xi_1) = 2k - 1$. *On the other hand, since $\xi_4(k) \sim -2^{-3k}$ as $k \to \infty$, we have sign$(\xi_4) = -1$ and the exponent of ξ_4 is either $-3k$ or $-3k - 1$. Moreover, checking that $2^{-3k} \leqslant |\xi_4(k)|$ for $k \geqslant 1$, we deduce that* exponent$(\xi_4) = -3k$.

3 Symbolic integers

In this section, we first define the notion of symbolic integers, that will be used in the next section for defining symbolic floating-point numbers. We then show how symbolic integers can be used to approximate other elements of $\mathbb{SQ}$, and next we focus on a particular approximation, namely "rounding-to-nearest".

3.1 Definition and first properties

The set of symbolic integers is defined as the set of the elements in $\mathbb{SQ}$ that evaluate to integers as k tends to infinity.

Definition 3.1. *$\mathbb{SZ}$ is the subset of functions $f \in \mathbb{SQ}$ for which $f(k) \in \mathbb{Z}$ as $k \to \infty$.*

Since $\mathbb{Z}$ is a commutative ring with 1, $\mathbb{SZ}$ inherits the same structure of commutative ring with 1.

Example 3.2. *ξ_1 in (10) is clearly an element of $\mathbb{SZ}$ because each of its terms evaluates to an integer for $k \geqslant 1$. On the contrary, ξ_2 does not belong to $\mathbb{SZ}$ because for k large enough (namely, $k \geqslant 3$), we have $-2^k + 2 < \xi_2(k) < -2^k + 3$, so that it cannot evaluate to an integer. Let us introduce another example:*

$$\xi_3' = \frac{2^k + 11}{3}. \tag{14}$$

For any $\ell \in \mathbb{N}$, $\xi_3'(2\ell+1) = \frac{2^{2\ell+1}+11}{3} \notin \mathbb{Z}$, as $2^{2\ell+1} + 11 \equiv 1 \pmod 3$. Hence $\xi_3' \notin \mathbb{SZ}$. However, for any $\ell \in \mathbb{N}$, we have $\xi_3'(2\ell) \in \mathbb{Z}$, that is, $\xi_3'(2k) \in \mathbb{SZ}$.

Definition 3.3. *We say that $f \in \mathbb{SZ}$ is even if $f(k)$ is even as $k \to \infty$, and odd if $f(k)$ is odd as $k \to \infty$.*

The following theorem will help us proving that any symbolic integer is either even or odd; see Corollary 3.7 below. This theorem states that the rational function representing $f \in \mathbb{SZ}$ through the change of variable $X = \beta^k$ is a polynomial with rational coefficients. The converse is in general not true, since for example $1/3 \notin \mathbb{SZ}$.

Theorem 3.4. *If $f \in \mathbb{SZ}$, then $f(k) = \widetilde{f}(\beta^k)$ with $\widetilde{f} \in \mathbb{Q}[X]$.*

Proof. Let $\widetilde{f} = P_1/P_2 \in \mathbb{Z}(X)$ be such that $f(k) = \widetilde{f}(\beta^k)$. The Euclidean division of P_1 by P_2 in $\mathbb{Q}[X]$ leads to $Q_1, R_1 \in \mathbb{Q}[X]$ such that $P_1(X) = Q_1(X) \cdot P_2(X) + R_1(X)$, with $\deg(R_1) < \deg(P_2)$. If $\lambda \in \mathbb{N}_{>0}$ is a multiple of all denominators in this equality, then $\lambda P_1(X) = Q(X) \cdot P_2(X) + R(X)$, with $Q(X) = \lambda Q_1(X) \in \mathbb{Z}[X]$, $R(X) = \lambda R_1(X) \in \mathbb{Z}[X]$ and we still have $\deg(R) < \deg(P_2)$. Moreover, for k large enough, $\lambda P_1(\beta^k)/P_2(\beta^k) = Q(\beta^k) + R(\beta^k)/P_2(\beta^k) \in \mathbb{Z}$. We deduce that $R(\beta^k)/P_2(\beta^k)$ is an integer that goes to 0 as $k \to \infty$. Hence $R = 0$, so that $P_1/P_2 = Q/\lambda \in \mathbb{Q}[X]$. □

Theorem 3.4 directly implies the following corollary about the asymptotic behavior of symbolic integers: bounded symbolic integers are classical integers.

Corollary 3.5. *Given $f \in \mathbb{SZ}$, either $|f(k)| \to +\infty$ as $k \to +\infty$, or $f \in \mathbb{Z}$.*

Example 3.6. *Since neither $\widetilde{\xi_3}(X)$ nor $\widetilde{\xi_4}(X)$ belongs to $\mathbb{Q}[X]$, we deduce from Theorem 3.4 that ξ_3 and ξ_4 are not symbolic integers. Note that $\xi_3(k) \to 2/3 \notin \mathbb{Z}$ as $k \to \infty$, hence Corollary 3.5 can also be used to conclude in this case.*

Moreover, it allows us to prove that the notion of parity is well defined.

Corollary 3.7. *Any $f \in \mathbb{SZ}$ is either odd or even.*

Proof. According to Theorem 3.4, $f(k)$ can be written as $f(k) = \sigma(k)/v$, with $\sigma(k) = \sum_{i=0}^{n} u c_i \beta^{ik+d}$, $n \in \mathbb{N}$, $u, v, d, c_i \in \mathbb{Z}$ $(i = 0, \ldots, n)$, and $\gcd(\beta, v) = 1$. Then, for k large enough, we have $\sigma(k) \in \mathbb{Z}$, which implies that $u c_0 \beta^d$ belongs to $\mathbb{Z}$, and since β is even and $\gcd(\beta, v) = 1$, it can be checked that the following assertions are equivalent: $u c_0 \beta^d$ is even; σ is even; $f(k)$ is even. □

3.2 Symbolic integral approximations

In this subsection, we introduce the notion of symbolic integral approximation that will be used to define rounding from $\mathbb{SQ}$ to $\mathbb{SZ}$.

Definition 3.8. *Given $f \in \mathbb{SQ}$, $g \in \mathbb{SZ}$ is a symbolic integral approximation of f if $f(k)-g(k) \in \mathcal{O}(1)$ as $k \to \infty$.*

Before illustrating this definition, we give two simple properties that follow directly from Definition 3.8: the first one shows that symbolic integral approximations remain valid when reducing the domain of their input variable, and the second one summarizes their additive properties.

Property 3.9. *Let $f, g \in \mathbb{SQ}$ such that g is an integral approximation of f. Then, for all $\omega \in \mathbb{N}^*$ and $\varphi \in \mathbb{Z}$, $g(\omega k + \varphi)$ is an integral approximation of $f(\omega k + \varphi)$.*

Property 3.10. *If g is an integral approximation of $f \in \mathbb{SQ}$, then the set of integral approximations of f is exactly $g + \mathbb{Z}$. Moreover, if g_1, g_2 are integral approximations of $f_1, f_2 \in \mathbb{SQ}$, respectively, then $g_1 + g_2$ is an integral approximation of $f_1 + f_2$.*

Example 3.11. *It is straightforward to deduce a symbolic integral approximation of ξ_2 in (10) : it suffices to truncate the sum before the constant term to get -2^k. This strategy works fine for elements of $\mathbb{E}$; however, some elements of $\mathbb{SQ}$ have no such approximation.*

For example, let us prove by contradiction that ξ_3' given by (14) cannot have any symbolic integral approximation. If $g \in \mathbb{SZ}$ is an approximation of ξ_3', then we deduce from Property 3.9 that $g(2k)$ is an approximation of $\xi_3'(2k)$. Moreover, since $\xi_3'(2k) \in \mathbb{SZ}$, it is its own approximation and we can use Property 3.10 to get $g(2k) = \xi_3'(2k) + n$, for $n \in \mathbb{Z}$. Therefore, since two rational functions that are equal at infinitely many points are equal everywhere, we have $g = \xi_3' + n$, which implies that $\xi_3' \in \mathbb{SZ}$, whereas we have seen in Example 3.2 that $\xi_3' \notin \mathbb{SZ}$: this proves that ξ_3' has no integral approximation. Note however that $\xi_3'(2k)$ admits an integral approximation: this result is generalized in Theorem 3.12 to any symbolic number.

We detail a third example, in which we compute a symbolic integral approximation for a fraction in $\mathbb{SQ}$. Let us consider

$$\xi_4' = \frac{-2^{3k+1} - 5 \cdot 2^{2k}}{2^{k+2} + 8}, \quad \text{and its counterpart} \quad \widetilde{\xi_4'}(X) = \frac{-2X^3 - 5X^2}{4X + 8}. \tag{15}$$

We first compute in $\mathbb{Q}[X]$ the Euclidean division of the numerator of $\widetilde{\xi_4'}(X)$ by its denominator, and we get

$$\widetilde{\xi_4'}(X) = -\frac{X^2}{2} - \frac{X}{4} + \frac{1}{2} + \frac{R(X)}{4X + 8}.$$

where $R(X)/(4X+8) \to 0$ as $X \to \infty$. Then, if we remove this last term $R(X)/(4X+8)$, we have an expression that belongs to $\mathbb{E}$ and proceed as we did for ξ_2. We define

$$\widetilde{g_4'}(X) = -\frac{X^2}{2} - \frac{X}{4}, \quad \text{and its counterpart} \quad g_4' = -2^{2k-1} - 2^{k-2}.$$

Since $g_4' \in \mathbb{SZ}$ and $g_4' - \xi_4' = 2^k/(2^{k+1}+4) < 1/2$, we known that g_4' is a symbolic integral approximation of ξ_4'.

The following theorem states that, given a symbolic number, we can always divide its input domain into finitely many subdomains so that there exists an integral approximation on each of these subdomains.

Theorem 3.12. *For every $f \in \mathbb{SQ}$, there exists $\omega \in \mathbb{N}_{>0}$ such that for all $\varphi \in \mathbb{Z}$, $f(\omega k + \varphi)$ admits an integral approximation $h(\omega k + \varphi)$.*

In the proof of Theorem 3.12, we first approximate f using a sum of functions of the form $k \mapsto c\beta^{ak}$, with $c \in \mathbb{Q}$ and $a \in \mathbb{N}$. Lemma 3.13 below shows that integral approximations of such functions can always be computed on a subdomain of their inputs. Then, an integral approximation of f can be deduced using Property 3.10.

Lemma 3.13. *Let $g(k) = c\beta^{ak} \in \mathbb{SQ}$ with $c \in \mathbb{Q}$ and $a \in \mathbb{N}$. There exists $\omega \in \mathbb{N}_{>0}$ such that for all $\varphi \in \mathbb{Z}$, $g(\omega k + \varphi)$ admits an integral approximation $h(\omega k + \varphi)$.*

Proof. If $a = 0$, then $g(k) = c$ so that $\omega = 1$ and $h = 0$ give a solution for any value of φ. Let us now assume that $a > 0$. We start by rewriting c as $u\beta^d/v$ with $u, v, d \in \mathbb{Z}$, and $\gcd(\beta, v) = 1$, so that $g(k) = u\beta^{ak+d}/v$. Since β and v are co-prime, there exists

$$\alpha = \min\{j \in \mathbb{N}_{>0} \,:\, \beta^j \equiv 1 \pmod{v}\},$$

and we define

$$\omega = \frac{\operatorname{lcm}(\alpha, a)}{a} \in \mathbb{N}_{>0}.$$

Let us pick out any $\varphi \in \mathbb{Z}$. We define $r \in \{0, 1, \cdots, v-1\}$ such that $r \equiv u\beta^{a\varphi+d} \pmod{v}$, and $h(k) = (u\beta^{ak+d} - r)/v \in \mathbb{SQ}$. We now prove that $h(\omega k+\varphi)$ is an integral approximation of $g(\omega k+\varphi)$. We have $g(k) - h(k) = r/v \in \mathcal{O}(1)$ as $k \to \infty$. Therefore, we just have to prove that $h(\omega k + \varphi) \in \mathbb{SZ}$. We define $\tau = u\beta^{a\omega k+a\varphi+d} - r$, so that $h(\omega k + \varphi) = \tau/v$. Since $a\omega k + a\varphi + d$ is a nonnegative integer for k large enough, we have $\tau \in \mathbb{SZ}$. Moreover, by definition of ω, we have $a\omega k + a\varphi + d \equiv a\varphi + d \pmod{\alpha}$, so that $u\beta^{a\omega k+a\varphi+d} \equiv r \pmod{v}$. Hence $\tau \equiv 0 \pmod{v}$ which implies $h(\omega k + \varphi) \in \mathbb{SZ}$, so that $h(\omega k + \varphi)$ is an integral approximation of $g(\omega k + \varphi)$. □

Proof of Theorem 3.12. We detail the computation of a suitable ω before building an integral approximation of $f(\omega k + \varphi)$.

Let $\widetilde{f} \in \mathbb{Z}(X)$ be such that $f(k) = \widetilde{f}(\beta^k)$, and let $\widetilde{g}(k) = \sum_{i=0}^{n} c_i X^i \in \mathbb{Q}[X]$ be the quotient in the Euclidean division of the numerator of $\widetilde{f}(X)$ by its denominator. We have $\widetilde{f}(X) - \widetilde{g}(X) \in \mathcal{O}(1)$ as $X \to \infty$. Let us also define $g(k) = \widetilde{g}(\beta^k) \in \mathbb{SQ}$. Since $\widetilde{f}(X)-\widetilde{g}(X) \in \mathcal{O}(1)$, any integral approximation of g is also an integral approximation of f, and we can now focus on finding an integral approximation of g. We define $g_i(k) = c_i\beta^{ik}$, so that $g(k) = \sum_{i=0}^{n} g_i(k)$, and we apply Lemma 3.13 to each of the g_i's to get a corresponding ω_i. We then define $\omega = \operatorname{lcm}(\omega_0, \omega_1, \omega_2, \cdots, \omega_n)$.

Now, given $\varphi \in \mathbb{Z}$, let us find an integral approximation of $g(\omega k+\varphi) = \sum_{i=0}^{n} g_i(\omega k+\varphi)$. According to Property 3.10, it is enough to sum the integral approximations of each $g_i(\omega k+\varphi)$. Therefore, we only consider a given $i \in \{0, 1, 2, \cdots, n\}$ and build an integral approximation of $g_i(\omega k + \varphi)$. By definition of ω_i, we know that there exists h_i such that $h_i(\omega_i k + \varphi)$ is an integral approximation of $g_i(\omega_i k + \varphi)$. Since w_i divides ω, we deduce from Property 3.9 that $h_i(\omega k + \varphi)$ is an integral approximation of $g_i(\omega k + \varphi)$. Hence, defining $h = \sum_{i=0}^{n} h_i$, $h(\omega k + \varphi)$ is an integral approximation of $f(\omega k + \varphi)$. □

3.3 Rounding to symbolic integers

The following theorem will allow us to define a rounding to the nearest symbolic integer whenever it exists, and gives a necessary and sufficient condition for its existence, related to the notion of symbolic approximation.

Theorem 3.14. *Given $f \in \mathbb{SQ}$, f admits an integral approximation if and only if $d = \min\{|g - f| \,:\, g \in \mathbb{SZ}\}$ is defined, in which case $d \leqslant 1/2$.*

Proof. If $h \in \mathbb{SZ}$ is an integral approximation of $f \in \mathbb{SQ}$, then there exists $M \in \mathbb{N}_{>0}$ such that $|h-f| \leqslant M$. Since $\{|h+n-f| \,:\, n \in \mathbb{Z}, |n| \leqslant 2M\}$ is a nonempty and finite set, it admits a minimum at $n_0 \in \mathbb{Z}$. Let us prove that for all $g \in \mathbb{SZ}$, $|h + n_0 - f| \leqslant |g - f|$, which shows that $h + n_0$ realizes $\min\{|g - f| \,:\, g \in \mathbb{SZ}\}$.

Given $g \in \mathbb{SZ}$, if g is not an integral approximation of f, then by Definition 3.8 we have $|g - f| > |h + n_0 - f|$. On the contrary, if g is an integral approximation of f, Property 3.10 implies $g = h + n_1$ with $n_1 \in \mathbb{Z}$. Either $|n_1| \leqslant 2M$, in which case by definition of n_0 we have $|g - f| \geqslant |h + n_0 - f|$, or $|n_1| > 2M$. In the latter case, the triangular inequality gives $|g - f| \geqslant |n_1| - |h - f|$, and since $|n_1| > 2M \geqslant 2|h - f|$, this implies $|g - f| > |h - f|$. Since $|0| \leqslant 2M$, the definition of n_0 gives $|g-f| > |h+n_0-f|$. In any case we have $|g-f| \geqslant |h+n_0-f|$, and $\min\{|g-f| \,:\, g \in \mathbb{SZ}\}$ is reached at $h + n_0$.

Conversely, if $\min\{|g-f| \,:\, g \in \mathbb{SZ}\}$ is well defined, then there exists $h \in \mathbb{SZ}$ such that $|h-f| = d$. Since $h-1$, $h+1 \in \mathbb{SZ}$, it implies that $|h-f| \leqslant |h+1-f|$ and $|h-f| \leqslant |h-1-f|$, that is $f \leqslant h+1/2$ and $f \geqslant h - 1/2$, respectively. Therefore, we know that $d \leqslant 1/2$ which implies that h is an integral approximation of f. □

Definition 3.15. *Given a tie-breaking function s, we define a rounding-to-the-nearest-integer partial function as follows: if $\sigma = \arg\min\{|g - f| : g \in \mathbb{SZ}\}$, then*

$$\lfloor f \rceil = \begin{cases} \text{undefined} & \text{if } \sigma = \emptyset, \\ g & \text{if } \sigma \text{ is a singleton } \{g\}, \\ s(\sigma) \in \sigma & \text{if } \sigma \text{ is a pair.} \end{cases}$$

According to Theorem 3.14, as soon as $f \in \mathbb{SQ}$ admits an integral approximation, the nearest symbolic integer, is defined. In this definition, s is a selection function over the pairs of consecutive integers. Indeed, if $\sigma = \{g_1, g_2\}$, then Theorem 3.14 implies that $g_2 = g_1 + 1$, up to a renaming. Moreover, we assume in the following that this tie-breaking function is odd, that is

$$s(\{-g_1, -g_2\}) = -s(\{g_1, g_2\}). \tag{16}$$

In particular, "ties-to-even", that breaks ties to the even integer is a valid tie-breaking function according to Corollary 3.7, as well as "ties-to-away", that selects the integer with the largest absolute value. For example, $2^k + 1/2$ rounds to 2^k when breaking ties to the even integer, and to $2^k + 1$ if ties are rounded away from zero.

We now give Algorithm 2 below that computes $\omega \in \mathbb{N}_{>0}$ and $r \in \mathbb{SQ}$, with $r(\omega k) = \lfloor f(\omega k) \rceil$, following the proof of Theorem 3.12. This algorithm is described using the matching $f(k) = \widetilde{f}(\beta^k)$, and a function *round* that rounds a rational to a closest integer.

Algorithm 2 (Rounding to the nearest integer)

algorithm $\lfloor f \rceil$
  $\widetilde{g}(X) \leftarrow$ Euclidean quotient of the numerator of $\widetilde{f}(X)$ by its denominator
  // $\widetilde{g}(X) = \sum_{i=0}^{n} c_i X^i$ and $\widetilde{g}(X) - \widetilde{f}(X) \in \mathcal{O}(1)$
  for each monomial $c_i X^i$, **compute** ω_i and $\widetilde{h}_i$ following Lemma 3.13, with $\varphi = 0$
  // $\widetilde{h}_i(\beta^{\omega_i k})$ is an integral approximation of $c_i \beta^{\omega_i k}$
  $\omega \leftarrow \mathrm{lcm}(\omega_0, \omega_1, \omega_2, \cdots, \omega_n)$
  $\widetilde{h} \leftarrow \sum_{i=0}^{n} \widetilde{h}_i$
  // from Theorem 3.12, $\widetilde{h}(\beta^{\omega k})$ is an integral approximation of $\widetilde{f}(\beta^{\omega k})$
  $c \leftarrow \lim_{x \to \infty}(\widetilde{f}(x) - \widetilde{h}(x))$
  // since $\widetilde{h}(\beta^{\omega k})$ is an integral approximation of $\widetilde{f}(\beta^{\omega k})$, $c \in \mathbb{Q}$
  $\widetilde{t} \leftarrow \widetilde{h} + \mathrm{round}(c)$
  $\delta \leftarrow \widetilde{f} - \widetilde{t}$
  if $|\delta| < 1/2$ **then** $\widetilde{r} \leftarrow \widetilde{t}$
  elif $|\delta| = 1/2$ **then** $\widetilde{r} \leftarrow s(\{\widetilde{t}, \widetilde{t} + \mathrm{sign}(\delta)\})$
  elif $|\delta| > 1/2$ **then** $\widetilde{r} \leftarrow \widetilde{t} + \mathrm{sign}(\delta)$

After computing ω and $\widetilde{h}$, since $\widetilde{f}(X) - \widetilde{h}(X) \in \mathbb{Z}(X)$, $c = \lim_{x \to \infty}(\widetilde{f}(x) - \widetilde{h}(x))$ is either a finite number in $\mathbb{Q}$, or $\pm\infty$. Moreover, following the proof of Theorem 3.12, $\widetilde{h}(\beta^{\omega k})$ is an integral approximation of $\widetilde{f}(\beta^{\omega k})$, which implies that $c \in \mathbb{Q}$. Since $\widetilde{t} = \widetilde{h} + \mathrm{round}(c)$ and $\delta = \widetilde{f} - \widetilde{t}$, we then have $\lim_{x \to \infty} \delta(x) = \ell \in [-1/2, 1/2]$. In particular, $\delta(\beta^{\omega k}) \to \ell$ as $k \to \infty$. This does not imply $\delta(\beta^{\omega k}) \in [-1/2, 1/2]$, as can be seen in the following example: $1/2 + 5\beta^{-2k} \to 1/2$, as $k \to \infty$, but $1/2 + 5\beta^{-2k} > 1/2$. However, for any $\epsilon > 0$, we have $|\delta(\beta^{\omega k})| < 1/2 + \epsilon$, and if $\epsilon = 1/2$, then $|\delta(\beta^{\omega k})| < 1$. Hence, a last correction step may be required to ensure that $\widetilde{r} = \lfloor f(\omega k) \rceil$. This correction is performed in the last two lines of the algorithm by adding ± 1 to $\widetilde{t}$ according to the sign of δ, or by using the tie-breaking function.

When f admits an integral approximation, the floor and the ceiling of f can be defined in a similar way as its rounding to the nearest symbolic integer: $\lfloor f \rfloor = \max\{g : g \in \mathbb{SZ}, g \leqslant f\}$, and $\lceil f \rceil = \min\{g : g \in \mathbb{SZ}, g \geqslant f\}$. It is possible to provide algorithms for these two directed rounding functions, by adapting Algorithm 2.

Example 3.16. *We first consider ξ_2 defined by* (10)*. We saw in Example 3.11 that -2^k is a symbolic integral approximation of ξ_2, which is a good starting point to find a nearest symbolic integer. We*

then compute $\lim_{k\to\infty} \xi_2(k) - (-2^k) = 5/2$. *It implies that* ξ_2 *is surrounded by the two consecutive symbolic integers* $-2^k + 2$ *and* $-2^k + 3$, *which correspond to the two integers nearest to* $5/2$ *and are therefore the two candidates. We can already deduce that*

$$\lfloor \xi_2 \rfloor = -2^k + 2 \quad \textit{and} \quad \lceil \xi_2 \rceil = -2^k + 3.$$

Moreover, comparing $|\xi_2 - (-2^k + 2)|$ *to* $1/2$ *allows us to conclude that* $\lfloor \xi_2 \rceil = -2^k + 2$.

Let us consider a second example given by ξ'_4 *as in* (15). *Example 3.11 gives* g'_4 *as a symbolic integral approximation of* ξ'_4. *In this case,* $\xi'_4(k) - g'_4(k) \to 1/2$ *as* $k \to \infty$, *with* $\xi'_4 - g'_4 < 1/2$, *so that*

$$\lfloor \xi'_4 \rfloor = g'_4, \quad \lceil \xi'_4 \rceil = g'_4 + 1 \quad \textit{and} \quad \lfloor \xi'_4 \rceil = g'_2.$$

4 Symbolic floating-point arithmetic

In this section, using the definition of symbolic integers, we first define symbolic floating-point numbers as particular elements of $\mathbb{SQ}$. We then define a round-to-nearest partial function from $\mathbb{SQ}$ to $\mathbb{SF}_p$, and next we show the relationship between this partial function and the classical round-to-nearest function from $\mathbb{Q}$ to $\mathbb{F}$.

4.1 Symbolic floating-point numbers

In conventional base-β floating-point arithmetic (and assuming an unlimited exponent range), a nonzero precision-p floating-point number is a number that can be written $M\beta^e$, where M is an integer in the range $[\beta^{p-1}, \beta^p)$ and e is an integer. The formalism introduced earlier allows us to define a set of symbolic floating-point numbers, with M a symbolic integer in $\mathbb{SZ}$, and e a linear exponent in $\mathbb{L}$. More precisely,

Definition 4.1. *Given a function* $p \in \mathbb{L}_{\geqslant 2}$, *let us define*

$$\mathbb{SF}_p = \{0\} \cup \{M\beta^e \,:\, M \in \mathbb{SZ},\, e \in \mathbb{L},\, \beta^{p-1} \leqslant |M| < \beta^p\}.$$

We have $\mathbb{SF}_p \subset \mathbb{SQ}$, and every function $f \in \mathbb{SF}_p$ satisfies the following: there exists $k_0 \in \mathbb{N}$ such that for all $k \geqslant k_0$, its value $f(k)$ is a floating-point number in base β and precision $p(k) \in \mathbb{N}_{\geqslant 2}$.

The set $\mathbb{SF}_p$ can thus be considered as a set of "symbolic floating-point numbers" whose precision p is parametrized by the variable k. In particular, $0 \in \mathbb{SF}_p$ and $\mathbb{SF}_p = -\mathbb{SF}_p$. Moreover, if $f \in \mathbb{SF}_p - \{0\}$ and $e_f = \text{exponent}(f)$, then f can be written as $f = F \cdot \beta^{e_f - p + 1}$, with $M \in \mathbb{SZ}$ and $\beta^{p-1} \leqslant |F| < \beta^p$: we call $|F|$ the significand of f.

The ulp function (unit in the last place function) locates the pth significant digit in the unsigned and possibly infinite representation of a nonzero symbolic number.

Definition 4.2. *Given* $p \in \mathbb{L}_{\geqslant 2}$ *and* $f \in \mathbb{SQ} - \{0\}$, $\text{ulp}_p(f) = \beta^{e_f - p + 1}$.

With this definition we can write $f \in \mathbb{SF}_p - \{0\}$ as

$$f = F \cdot \text{ulp}_p(f), \tag{17}$$

with $F \in \mathbb{SZ}$ and $\beta^{p-1} \leqslant |F| < \beta^p$. Hence, $f \in \mathbb{SQ} - \{0\}$ is a symbolic floating-point number in precision p if and only if $f/\text{ulp}_p(f) \in \mathbb{SZ}$, as the definition of ulp_p already implies that $\beta^{p-1} \leqslant |f/\text{ulp}_p(f)| < \beta^p$.

Example 4.3. *It can be checked that* ξ_1 *and* ξ_2 *given by* (10) *are symbolic floating-point numbers in precision* $p = 2k$. *In the case of* ξ_2, *we have* $\text{ulp}_p(\xi_2) = 2^{-k}$ *and* $\xi_2/2^{-k} \in \mathbb{SZ}$. *We have* $\xi_2 = M \cdot 2^e$, *with* $M = \xi_2/2^{-k}$ *and* $e = -k$, *hence* $\xi_2 \in \mathbb{SF}_p$.

On the other hand, according to (17), ξ_3 *as in* (11) *is not a symbolic floating-point number in precision* $p = k$, *as we proved in Example 3.2 that* $\xi_3/\text{ulp}_p(\xi_3) = \xi'_3 \notin \mathbb{SZ}$. *In fact, whatever the precision is,* ξ_3 *cannot be a symbolic floating-point number. Let us assume, by contradiction, that* $\xi'_3 \in \mathbb{SF}_p$ *for* $p \in \mathbb{L}_{\geqslant 2}$. *Then* $\xi_3/\text{ulp}_p(\xi_3) = (2^p + 11 \cdot 2^{p-k})/3 \in \mathbb{SZ}$, *which means that there exists* $k_0 \in \mathbb{N}_{>0}$ *such that for* $k \geqslant k_0$, $(2^{p(k)} + 11 \cdot 2^{p(k)-k})/3 \in \mathbb{Z}$. *Hence,* $2^{p(k)} + 11 \cdot 2^{p(k)-k} \in \mathbb{Z}$ *and*

$2^{p(k)} + 11 \cdot 2^{p(k)-k} \equiv 0 \pmod 3$. *But* $2^{p(k)} + 11 \cdot 2^{p(k)-k} \equiv (-1)^{p(k)}(1 + (-1)^{k+1}) \pmod 3$, *so that* k *must even, which is not always the case.*

However, if we assume that k *is even, there is no more contradiction and* ξ_3 *is a symbolic floating-point number as soon as* $p(k) \geqslant k$ *(which is equivalent to* $2^p + 11 \cdot 2^{p-k} \in \mathbb{SZ}$*). Formally, it means that* $\xi_3(2k) \in \mathbb{SF}_p$ *if and only if* $p(k) \geqslant 2k$.

4.2 Rounding to symbolic floating-point numbers

We now define a round-to-nearest partial function from $\mathbb{SQ}$ to $\mathbb{SF}_p$. For some $f \in \mathbb{SQ}$, there is no element of $\mathbb{SF}_p$ that is closer to f than any other, in which case $\mathrm{RN}_p(f)$ remains undefined. Moreover, when $\sigma' = \arg\min\{|h - f| \,:\, h \in \mathbb{SF}_p\}$ is not empty, it contains at most two elements.

A tie-breaking function s'_p is needed when σ' is a pair $\{h_1, h_2\}$ of consecutive symbolic floating-point numbers : given any pair of consecutive elements in $\mathbb{SF}_p$, s'_p selects one element in the pair. We assume in the following that s'_p satisfies, for $e \in \mathbb{L}$,

$$s'_p(\{\beta^e \cdot h_1, \beta^e \cdot h_2\}) = \beta^e \cdot s'_p(\{h_1, h_2\}) \quad \text{and} \quad s'_p(\{-h_1, -h_2\}) = -s'_p(\{h_1, h_2\}). \tag{18}$$

Moreover, noticing that the pairs of consecutive integers in $[\beta^{p-1},\, \beta^p]$ match the pairs of consecutive floating-point numbers in precision p in the same interval, we make the additional assumption that, for any precision $p \in \mathbb{L}_{\geqslant 2}$ and any $h \in \mathbb{SZ} \cap [\beta^{p-1},\, \beta^p)$,

$$s'_p(\{h,\, h+1\}) = s(\{h,\, h+1\}), \tag{19}$$

where s is a tie-breaking function for rounding to the nearest symbolic integer, as defined in Subsection 3.3. For instance, "ties-to-even", that selects the symbolic floating-point number whose significand is even, and "ties-away", that selects the one with the largest absolute value, satisfy the hypothesis above with their integer counterparts.

Definition 4.4. *Given* $p \in \mathbb{L}_{\geqslant 2}$*, and a tie-breaking function* s'_p*, the round-to-nearest function* RN_p *is defined as a partial function from* $\mathbb{SQ}$ *to* $\mathbb{SF}_p$ *as follows:*

$$\mathrm{RN}_p(f) = \begin{cases} \textit{undefined} & \textit{if } \sigma' = \emptyset \\ h & \textit{if } \sigma' = \{h\} \\ s'_p(\sigma') \in \sigma' & \textit{if } \sigma' \textit{ is a pair.} \end{cases}$$

Note that RN_p is nondecreasing: given $f_1, f_2 \in \mathbb{SQ}$ such that $\mathrm{RN}_p(f_1)$ and $\mathrm{RN}_p(f_2)$ are defined,

$$f_1 \leqslant f_2 \quad \Rightarrow \quad \mathrm{RN}_p(f_1) \leqslant \mathrm{RN}_p(f_2).$$

Moreover, the regularity assumptions (18) about the tie-breaking function s'_p transfer to RN_p: given $f \in \mathbb{SQ}$, we have

$$\mathrm{RN}_p(\beta^e \cdot f) = \beta^e \cdot \mathrm{RN}_p(f) \quad \text{and} \quad \mathrm{RN}_p(-f) = -\mathrm{RN}_p(f).$$

The following theorem provides a practical way of evaluating this RN_p function, which is related to the rounding-to-integer function previously defined.

Theorem 4.5. *Let* $p \in \mathbb{L}_{\geqslant 2}$*, and let* s *and* s'_p *be two tie-breaking functions for symbolic integers and floating-point numbers respectively, satisfying* (19)*. For any* $f \in \mathbb{SQ} - \{0\}$*, we have*

$$\mathrm{RN}_p(f) = \mathrm{ulp}_p(f) \cdot \left\lfloor \frac{f}{\mathrm{ulp}_p(f)} \right\rceil.$$

Proof. We first prove that, defining

$$\sigma = \arg\min\{|g - f/\mathrm{ulp}_p(f)| \,:\, g \in \mathbb{SZ}\} \quad \text{and} \quad \sigma' = \arg\min\{|h - f| \,:\, h \in \mathbb{SF}_p\},$$

we have

$$\sigma' = \mathrm{ulp}_p(f) \cdot \sigma. \tag{20}$$

Since changing f to $-f$ modifies both σ and σ' to $-\sigma$ and $-\sigma'$, respectively, we can assume without loss of generality that $f > 0$. We note $e_f = \mathrm{exponent}(f)$ so that $\beta^{e_f} \leqslant f < \beta^{e_f+1}$ and $\mathrm{ulp}_p(f) = \beta^{e_f-p+1}$.

If $h^* \in \sigma'$, since β^{e_f} and β^{e_f+1} are both symbolic floating-point numbers in precision p, and are surrounding f, it can be deduced that h^* satisfy $\beta^{e_f} \leqslant h^* \leqslant \beta^{e_f+1}$. We define $g^* = h^*/\mathrm{ulp}_p(f)$: it can be checked that $g^* \in \mathbb{SF}_p$, and we will show that it belongs to σ. Let g be any element in $\mathbb{SZ}$. If $g < \beta^{p-1}$ or $\beta^p < g$, since both β^{p-1} and β^p are symbolic integers, then g cannot minimize $|g - f/\mathrm{ulp}_p(f)|$. Hence we assume $\beta^{p-1} \leqslant g \leqslant \beta^p$, which implies $\beta^{e_f} \leqslant g \cdot \mathrm{ulp}_p(f) \leqslant \beta^{e_f+1}$ and $g \cdot \mathrm{ulp}_p(f) \in \mathbb{SF}_p$. Hence, by definition of h^*, we have $|h^* - f| \leqslant |g \cdot \mathrm{ulp}_p(f) - f|$ so that $|g^* - f/\mathrm{ulp}_p(f)| \leqslant |g - f/\mathrm{ulp}_p(f)|$.

Conversely, if $g^* \in \sigma$, then we define $h^* = g^* \cdot \mathrm{ulp}_p(f)$, and we will check that $h^* \in \sigma'$. Since β^{p-1} and β^p are both symbolic integers, and are surrounding $f/\mathrm{ulp}_p(f)$, we deduce that $\beta^{p-1} \leqslant g^* \leqslant \beta^p$. Hence, $h^* \in \mathbb{SF}_p$. Moreover, since $\beta^{e_f} \leqslant f < \beta^{e_f+1}$, any $h \in \mathbb{SF}_p$ minimizing $|h - f|$ must satisfy $\beta^{e_f} \leqslant h \leqslant \beta^{e_f+1}$, in which case $h/\mathrm{ulp}_p(f) \in \mathbb{SZ}$ and $\beta^{p-1} \leqslant h/\mathrm{ulp}_p(f) \leqslant \beta^p$. Then, by definition of g^*, we have $|g^* - f/\mathrm{ulp}_p(f)| \leqslant |h/\mathrm{ulp}_p(f) - f/\mathrm{ulp}_p(f)|$, so that $|h^* - f| \leqslant |h - f|$, which concludes the proof of (20).

If σ'_p is empty or is a singleton, then the result follows directly from (20). If σ'_p is a pair, then we can use (20), (18) and (19) successively to get the result. □

Example 4.6. *In base 2 and precision $p = 2k$, we consider*

$$\xi_4 = \frac{-2^{3k} - 5 \cdot 2^{2k-1}}{2^{6k} + 2^{5k+1}}.$$

Then, we have $\mathrm{ulp}_p(\xi_4) = 2^{-5k+1}$ and $\xi_4/\mathrm{ulp}_p(\xi_4) = \xi'_4$, with ξ'_4 as in (15). Applying Theorem 4.5, and the result in Example 3.16, we deduce that

$$\mathrm{RN}_p(\xi_4) = -2^{-3k} - 2^{-4k-1}.$$

4.3 Relationship with the classical floating-point arithmetic

In this section, we prove that the symbolic rounding of an element f in $\mathbb{SQ}$ evaluates, for k large enough, to the classical rounding of $f(k)$ to an element in $\mathbb{F}_{p(k)}$.

For this purpose, we assume that we are given a family of tie-breaking functions $(s''_p)_{p \in \mathbb{N}_{\geqslant 2}}$ that choose one element among pairs of consecutive floating-point numbers in $\mathbb{F}_{p(k)}$. Moreover, if $\{h_1, h_2\}$ is a pair of consecutive symbolic floating-point numbers in $\mathbb{SF}_p$, then it can be checked that $\{h_1(k), h_2(k)\}$ is a pair of consecutive floating-point numbers in $\mathbb{F}_{p(k)}$, for k large enough. Therefore, we can make the following assumption: for all precisions $p \in \mathbb{L}_{\geqslant 2}$, and any pair of consecutive elements $\{h_1, h_2\} \subset \mathbb{SF}_p$,

$$s'_p(\{h_1, h_2\}) = s''_{p(k)}(\{h_1(k), h_2(k)\}) \quad \text{as } k \to \infty. \tag{21}$$

This assumption basically means that ties are broken using the same strategy when rounding to $\mathbb{SF}_p$ and rounding to $\mathbb{F}_{p(k)}$. Hence, we will use the same notation s'_p for all tie-breaking functions in this subsection. Also, we will denote by $\mathrm{RN}_p(k)$ the classical function that rounds any real to $\mathbb{F}_{p(k)}$ using the tie-breaking function $s'_{p(k)}$.

Theorem 4.7. *Under the assumption (21), given $p \in \mathbb{L}_{\geqslant 2}$ and $f, \widehat{f} \in \mathbb{SQ}$ such that $\widehat{f} = \mathrm{RN}_p(f)$, we have*

$$\widehat{f}(k) = \mathrm{RN}_{p(k)}(f(k)) \quad \text{as } k \to \infty.$$

Proof. If $f = 0$, then $\widehat{f} = 0$ and the result is clear. We consider $f \in \mathbb{SQ} - \{0\}$ and define $e_f = \mathrm{exponent}(f)$. For k large enough, the definitions of the symbolic exponent and ulp functions match the numerical ones, that is, $e_f(k)$ is the numerical exponent of $f(k)$, and $(\mathrm{ulp}_p(f))(k) = \mathrm{ulp}_{p(k)}(f(k))$. Since $\pm\beta^{e_f}$ and $\pm\beta^{e_f+1}$ are symbolic floating-point numbers in precision p, by definition of $\widehat{f}$ we have $\beta^{e_f} \leqslant |\widehat{f}| \leqslant \beta^{e_f+1}$. Moreover, we deduce from Theorem 4.5 and Theorem 3.14 that

$$|\widehat{f} - f| \leqslant \frac{1}{2}\mathrm{ulp}_p(f). \tag{22}$$

If the inequality in (22) is strict, then, for k large enough, we have $\beta^{e_f(k)} \leqslant |\widehat{f}(k)| \leqslant \beta^{e_f(k)+1}$ and $|\widehat{f}(k) - f(k)| < \frac{1}{2}\mathrm{ulp}_{p(k)}(f(k))$, and the result follows.

In the case of equality in (22), there are two nearest symbolic floating-point numbers to f, namely $\widehat{f}$ and $h_2 = \widehat{f} + \mathrm{sign}(f - \widehat{f}) \cdot \mathrm{ulp}_p(f)$. Hence, $\widehat{f}$ is resulting from the tie-breaking function s'_p, that is $\widehat{f} = s'_p(\{\widehat{f}, h_2\})$. Therefore, $\mathrm{RN}_{p(k)}(f(k)) = s'_{p(k)}(\{\widehat{f}(k), h_2(k)\})$, and the result then follows from (21). □

Let us briefly check that (21) is satisfied for "ties-to-even" and "ties-to-away" tie-breaking functions. Let h_1 and h_2 be two consecutive elements in $\mathbb{SF}_p$, and let us assume that h_1 is selected by "ties-to-even". If $h_1 = H_1 \cdot \mathrm{ulp}_p(h_1)$ and $h_2 = H_2 \cdot \mathrm{ulp}_p(h_2)$, then we know that $|H_1|$ is even and $|H_2|$ is odd. Moreover, for k large enough, the significand of $h_1(k)$ is $|H_1(k)|$, which is even, and the significand of h_2 is $|H_2(k)|$, which is odd. Hence, hypothesis (21) is satisfied. For "ties-to-away" hypothesis (21) is also satisfied as a direct consequence of definition of the comparisons in Section 2.

5 Implementation and experiments

In Sections 2 and 4, we defined the arithmetic over the symbolic data in $\mathbb{SQ}$ and $\mathbb{SF}_p$ as $k \to \infty$. In practice, we want to compute a lower bound for k from which this asymptotic behavior is reached. Here, we explain how we implemented this aspect. We then describe the features of our library, and give three examples to illustrate its use.

5.1 Practical handling of the asymptotic behavior

To compute a lower bound from which the asymptotic behavior is reached, every element of $\mathbb{SQ}$ is given with an additional $k_0 \in \mathbb{N}$ that ensures that the denominator does not vanish. The heuristic to compute such a k_0 is to find the minimal value of k realizing (4) for the denominator, that is,

$$e_1(k) > e_2(k) > \cdots > e_n(k) \quad \text{for all } k \geqslant k_0.$$

Moreover, for any operation on elements of $\mathbb{SQ}$, given with their own k_0, the result is also given with a new k_0 realizing the asymptotic behavior of the function.

We also chose in our implementation to force the new value of k_0, coming with the result, to be greater than or equal to any k_0 given as input. For example, if f and k_0 are given by (2) with $n \geqslant 1$, the exponent function computes a pair $(e, k'_0) \in \mathbb{L} \times \mathbb{N}$ such that $k'_0 \geqslant k_0$ and $\beta^{e(k)} \leqslant |f(k)| < \beta^{e(k)+1}$ for all $k \geqslant k'_0$. Hence, we ensure a nondecreasing behavior for the k_0's, and the last k_0 obtained after a sequence of operations guarantees that the asymptotic behavior is reached for every operation in this sequence.

All the updates of k_0 come from the algorithm comparing two exponents in $\mathbb{L}$. It is then the building block for the computation of a valid k_0, together with the nondecreasing behavior of k_0 we enforce. When comparing two exponents $a_1 k + b_1$ and $a_2 k + b_2$ in $\mathbb{L}$, either $a_1 = a_2$ and the ordering is valid for every $k \in \mathbb{N}$, or $a_1 \neq a_2$ and the comparison is valid as soon as $k \geqslant k_0$, with $k_0 = \lfloor (b_2 - b_1)/(a_1 - a_2) \rfloor + 1$.

5.2 Description of the library

Our Maple library defines a procedure that, given an even fixed base β and a symbolic variable k, builds a Maple package for a symbolic floating-point arithmetic in base β. This package defines the class of symbolic numbers $\mathbb{SQ}$, overloading the basic operations (`+`, `-`, `*`, `/` and `^`). It also defines a set of functions over $\mathbb{SQ}$ objects for a symbolic integer arithmetic and a symbolic floating-point one. Elements of $\mathbb{SZ}$ and $\mathbb{SF}_p$ are just objects of the same class $\mathbb{SQ}$.

More precisely, an object f of the class $\mathbb{SQ}$ contains 3 attributes:

- *fun*: representing f as a rational function over a local variable according to the change of variable $X = \beta^k$;
- k_0: meaning that the calculations that led to f are correct for $k \geqslant k_0$;
- ω: meaning that the calculations that led to f are correct for k multiple of ω.

This class implements the following methods:

- a constructor `SQ`, that takes as parameters a *value* $\in \mathbb{SQ}$, parametrized by k. It computes an initial k_0 that ensures that the denominator does not vanish, and sets $\omega = 1$;
- the overloaded exact operations `+`, `-`, `*`, `/` and `^` on $\mathbb{SQ}$;
- the accessors `get_fun`, that returns fun as a procedure, `get_k0` and `get_omega`;
- `update_k0` that replaces the previous k_0 of $f \in \mathbb{SQ}$ by the maximum between the previous and the new k_0, and returns f.

The package defines 3 elements of $\mathbb{SQ}$ (`zero`, `oneHalf` and `one`), and the following functions over $\mathbb{SQ}$:

- `sign`: returns $(s, k_0) \in \{-1, 0, 1\} \times \mathbb{N}$ such that s is the sign of $f(k)$ for $k \geqslant k_0$;
- `cmp`(f_1, f_2) and `abs`(f): return respectively `sign`$(f_1 - f_2)|$ and $|f|$;
- `exponent`(f): returns $(e, k_0) \in \mathbb{L} \times \mathbb{N}$ such that $\beta^{e(k)} \leqslant |f|(k) < \beta^{e(k)+1}$ for $k \geqslant k_0$;
- `tiesToEven` and `tiesAway`: two tie-breaking rules for the rounding to the nearest integer, whose parameters are two consecutive symbolic integers; `tiesToEven` selects the even symbolic integer and `tiesAway` the largest one comparing the absolute values;
- `floor`, `ceil`, and `round`: rounding functions from $\mathbb{SQ}$ to $\mathbb{SZ}$, overloading the classical ones; the tie-breaking rule for `round` can be given as a second parameter and is set by default to `tiesToEven`;
- `ulp`(f, p): returns $g \in \mathbb{SQ}$ such that $g(k) = \mathrm{ulp}_{p(k)}(f(k))$ for $k \geqslant$ `get_k0`(g);
- `rd`, `ru`, `rn`: rounding functions from $\mathbb{SQ}$ to $\mathbb{SF}_p$, where $p \in \mathbb{L}$ is the second parameter; they rely on the rounding-to-integer functions so that the tie-breaking rule can be chosen similarly to `round`, using an optional third parameter;
- `expr_ur`(f, p, u), where u is a symbolic variable: expresses f as a rational function with respect to the unit roundoff $\beta^{1-p}/2 = u$. If a and b are such that $p = ak + b$, since $X = \beta^k$, the function performs the change of variable $X = (\beta^{1-b}/u)^{1/a}$ in the internal representation of f as a rational function.

5.3 Examples

Below, we give three examples that illustrate the use of our Maple library. The first example is devoted to the computation of a two-by-two determinant with Kahan's algorithm (see Algorithm 1); we run this algorithm on a set of inputs parametrized by the precision p in (1) for which the relative error is equivalent to $2u$ as $u \to 0$ (or $p \to \infty$ while β is fixed). It provides a certificate of asymptotic optimality for the relative error bound $2u$.

The second example concerns the algorithm CompDivS given in [9] to compute a complex floating-point quotient, that is an approximation $\widehat{x} + i\,\widehat{y}$ to $(a + ib)/(c + id)$ where all variables are taking floating-point values. The authors of [9] are interested in the componentwise relative error. Example 8 in that paper provides inputs parametrized by the precision for which the computation of the real part leads to a relative error equivalent to the a priori bound of $5u$. In this example, we use the division operation and an even precision.

In the last example of this subsection (taken from [3]), we briefly illustrate the use of `get_omega()` to figure out if an additional condition on the divisibility of p is needed, when such a condition is not known in advance.

Before any computation, the package has to be built and loaded, for a fixed base. One possible way to do so in base 10 is the following command lines:

```
> read("libsfp0.6.mpl"):
> beta := 10:
> SF10 := SF(beta, k):
> with(SF10):
```

After reading the files and setting the base to 10, the third line creates a package for a symbolic floating-point arithmetic in base 10, with the parameter k. This package is then loaded for an easy use.

5.3.1 Kahan's algorithm

The algorithm is implemented as a Maple procedure as follows, using the exact operations in $\mathbb{SQ}$, and the RN function from our Maple library:

```
> Kahan := proc(a, b, c, d, p)
>   wh := rn(bc, p);
>   e  := rn(wh - bc, p);
>   fh := rn(ad - wh, p);
>   rn(fh + e, p)
> end proc:
```

Then we set the input variables as Maple expressions parametrized by k as in (1), and we compute an expression for the exact result `x`:

```
> p := k:
> a := SQ(beta^(p-1) + 1):
> b := a:
> c := SQ(2*beta^(p-1) + (beta/2)*beta^(p-2)):
> d := SQ(beta^(p-1) + (beta/2)*beta^(p-2)):
> x := ad - bc;
  x := 10^(2k-2) + 10^(k-1)
```

Next, we apply the symbolic version of Kahan's algorithm to get the approximate result `xh`, in precision $p = k$:

```
> xh := Kahan(a, b, c, d, p);
    xh := 10^(2k-2)
> get_omega(xh);
    1
> get_k0(xh);
    3
```

The paper and pencil computation of `xh` is rather tedious and occupies half a page in [8]. Since `get_omega(xh)` returns 1 and `get_k0(xh)` returns 3, we know that the computed result is valid for any $k \geqslant 3$, that is for $p \geqslant 3$, and using a numerical evaluation, it can be checked that for $p = 2$ the algorithm returns 120 while $\mathtt{xh}(2) = 100$.

We now compute the error and express it with respect to $u = 1/2 \cdot 10^{1-p}$ using the representation as a rational function over $X = 10^k = 5/u$.

```
> err := abs((x - xh)/x);
    err := 2/(4^k + 2)
> simplify(get_fun(err)(5/u));
    2*u/(1+2*u)
```

These computations double-check the certificate of optimality for the relative error bound of Kahan's algorithm for $\beta = 10$. The last instruction can be replaced by `simplify(expr_ur(err, p, u))`.

5.3.2 Complex floating-point division

This example deals with floating-point division, and illustrates how to handle the case of an even precision, in base 2. The pen-and-paper computation of the result is more involved than in the previous case, especially for rounding the quotient, while it is of course much more efficient with our library. We first build the appropriate package:

```
> read("libsfp0.6.mpl"):
> beta := 2:
> SF2 := SF(beta, k):
> with(SF2):
```

Since p is even, we set $p = 2k$:

```
> p := 2*k:
> a := SQ(beta^p-5*beta^(p/2-1)):
> b := SQ(-beta^(p/2) + 5/beta - 3*beta^(-p/2)):
> c := SQ(beta^p - beta):
> d := SQ(beta^(3*p/2)+beta^p):
```

As previously mentioned, a, b, c, d are built into the data structure. We compute the exact real part of the result `r` and the approximate one `rh` using the algorithm provided in [9]:

```
> r := (a*c+b*d)/(c^2+d^2):
> Dh := rn(c^2 + rn(d^2, p), p):
> Gh := Kahan(a, -b, d, c, p):
> rh := rn(Gh/Dh, p);
    rh := -8^(-k)-(1/2)*16^(-k)
> get_omega(rh);
    1
> get_k0(rh);
    4
```

Since `get_omega(xh)` and `get_k0(xh)` return respectively 1 and 3, we know that the computed result is valid for any $k \geqslant 4$. In fact, it can be checked by numerical evaluation for $k = 3$ that the algorithm and the expression `rh` produce the same value. Hence, `rh` is valid for all $k \geqslant 3$.

Next, we compute the relative error, express it with respect to the unit roundoff $u = 2^{-2k}$ and compute an equivalent as $u \to 0$.

```
> err := (rh - r)/r:
> series(expr_ur(err, p, u), u=0, 3) assuming u > 0:
```

This gives $\texttt{err} = 5u - \frac{23}{2}u^{3/2} + \mathcal{O}(u^2)$ when $u \to 0$, which confirms the result given in [9].

5.3.3 Complex floating-point multiplication

Finally, we illustrate the use of `get_omega()` when an additional condition on the divisibility of k is needed. The example we give is taken from Corollary 4 in [3]: the authors provide a set of parametrized inputs to prove that, in base 2 and assuming that the precision is even, the relative error bound $\sqrt{5}$u for the complex floating-point multiplication is asymptotically optimal. Let us consider only one of the inputs proposed, namely $f = 2/3 \cdot (1 + 11 \cdot 2^{-p})$, and see if it possible to use this input for any precision. For this purpose, we ignore the assumption on the parity of p, and we round f using the `rn` function from the library:

```
> beta := 2: p := k:
> f := SQ(2/3*(1+11*beta^(-p))):
> fh := rn(f, p);
    fh := 2/3+(22/3)*2^(-k)
> get_k0(fh);
    6
> get_omega(fh);
    2
```

Note that `get_omega(fh)` returns 2, which is the attribute ω of the computed result `fh`. From the integer returned by `get_k0(fh)`, we know that '$k \geqslant 6$ even' is a sufficient condition to ensure that the computed rounding is correct. Under this condition, since $\texttt{rn}(f, p) = f$, we known that f is a symbolic floating-point number in precision $p = k$.

If we want to determine the rounding of f when p is odd, it is of course possible to redo the computation in precision $p = 2k + 1$:

```
> beta := 2: p := 2*k+1:
> f := SQ(2/3*(1+11*beta^(-p))):
> fh := rn(f, p);
    fh := 2/3+(23/6)*4^(-k)
> get_k0(fh);
    3
> get_omega(fh);
    1
```

As indicated by `get_k0(fh)` and `get_omega(fh)`, the computed rounding is then valid for any $k \geqslant 3$.

Table 1: Timings for checking examples with our library.

Example (base 2)	#rn	Timing
• From [8]:		
Example 3.7 (p even)	4	17 ms
Example 3.7 (p odd)	4	16 ms
Example 4.4	4	12 ms
Example 4.6	4	13 ms
Example 6.2	4	15 ms
Example 6.3 (p even)	4	15 ms
Example 6.3 (p odd)	4	15 ms
Example 6.4 (p even)	4	16 ms
Example 6.6 (1st part)	4	12 ms
Example 6.6 (2d part)	4	12 ms
Example 6.7 (1st part)	4	13 ms
Example 6.7 (2d part)	4	13 ms
• From [11]:		
Example	6	21 ms
• From [3]:		
Example (p even)	6	18 ms
Example (p odd)	6	19 ms

Example (base 2)	#rn	Timing
• From [9]:		
Example 1	3	10 ms
Example 2	3	12 ms
Example 3 (p even)	3	10 ms
Example 4	2	9 ms
Example 5 (p even)	6	21 ms
Example 5 (p odd)	6	20 ms
Example 6 (p even)	2	10 ms
Example 6 (p odd)	2	10 ms
Example 7 (p even)	2	11 ms
Example 8 (p even)	12	43 ms

Example (base 10)	#rn	Timing
• From [8]:		
Example 3.7 (p even)	4	16 ms
Example 3.7 (p odd)	4	16 ms
Example 4.4	4	13 ms
Example 4.6	4	12 ms

6 Conclusion

We finally report some practical computing times to show that our library is not only of theoretical interest, but that it can be used to efficiently check examples from the literature, taken from [3, 8, 9, 11].

The individual measured computing times for each example (average on 100 runs) are reported in Table 1: the measurements were performed with the command `time[real]()` under Maple 18, on a laptop equipped with an Intel Core i5 (4310U, 2 GHz) running Linux 3.16. All the examples listed here use rounding to the nearest only, with tiesToEven: we report in the column "#rn" the number of calls to the library function `rn()` required by the examples, as a rough indication of their "size".

Although the code is not designed with high performance as the primary target, each example is checked within a few tens of milliseconds, even Example 8 from [9] involving a symbolic division and 12 calls to the rounding function. Overall, the library checks 29 examples (25 in base 2, and 4 in base 10) similar to the ones given in Subsection 5.3, in less than 0.5 second of CPU time on a recent laptop, which we consider as sufficiently fast for our purposes.

These experiments confirm that it is possible to efficiently check examples from the literature involving symbolic floating-point numbers in a computer algebra system. We also hope that this work will make easier the analysis of small yet important building blocks of numerical computing.

References

[1] Algirdas Avizienis. Signed-digit number representations for fast parallel arithmetic. *IRE Transactions on Electronic Computers*, 10:389–400, 1961.

[2] Sylvie Boldo and Guillaume Melquiond. Flocq: A unified library for proving floating-point algorithms in coq. In *20th IEEE Symposium on Computer Arithmetic (ARITH), Tübingen, Germany*, pages 243–252, Los Alamitos, CA, USA, 2011. IEEE Computer Society Press.

[3] Richard Brent, Colin Percival, and Paul Zimmermann. Error bounds on complex floating-point multiplication. *Mathematics of Computation*, 76:1469–1481, 2007.

[4] Marc Daumas, Laurence Rideau, and Laurent Théry. A generic library for floating-point numbers and its application to exact computing. In *14th International Conference on Theorem Proving in Higher Order Logics (TPHOLs'01), Edinburgh, Scotland, UK*, pages 169–184, Berlin Heidelberg, 2001. Springer-Verlag.

[5] Graham Everest, Alf van der Poorten, Igor Shparlinski, and Thomas Ward. *Recurrence Sequences*, volume 104 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, USA, 2003.

[6] John Harrison. A machine-checked theory of floating point arithmetic. In *12th International Conference on Theorem Proving in Higher Order Logics (TPHOLs'99), Nice, France*, pages 113–130, Berlin Heidelberg, 1999. Springer-Verlag.

[7] IEEE Computer Society. *IEEE Standard for Floating-Point Arithmetic*. IEEE Computer Society, New York, August 2008. available at `http://ieeexplore.ieee.org/servlet/opac?punumber=4610933`.

[8] Claude-Pierre Jeannerod, Nicolas Louvet, and Jean-Michel Muller. Further analysis of Kahan's algorithm for the accurate computation of 2×2 determinants. *Mathematics of Computation*, 82:2245–2264, 2013.

[9] Claude-Pierre Jeannerod, Nicolas Louvet, and Jean-Michel Muller. On the componentwise accuracy of complex floating-point division with an FMA. In *21st IEEE Symposium on Computer Arithmetic (ARITH), Austin, TX, USA*, pages 83–90, Los Alamitos, CA, USA, 2013. IEEE Computer Society Press.

[10] Guillaume Melquiond. Floating-point arithmetic in the Coq system. *Information and Computation*, 216:14–23, 2012.

[11] Jean-Michel Muller. On the error of computing $ab+cd$ using Cornea, Harrison and Tang's method. *ACM Transactions on Mathematical Software*, 41(2):7:1–7:8, 2015.