Exact real computer arithmetic with continued fractions

J. Vuillemin

Rapports de Recherche

N° 760

EXACT REAL COMPUTER ARITHMETIC WITH CONTINUED FRACTIONS

Jean VUILLEMIN

NOVEMBRE 1987

Exact Real Computer Arithmetic with Continued Fractions

Jean Vuillemin
INRIA [*]

August 27, 1987

[*]Institut National de Recherche en Informatique et Automatique, 78150, Rocquencourt, France. This paper was written while the author was a Visiting Scientist, at Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto CA 94306, U.S.A

Arithmétique Réelle Exacte par les Fractions Continues

Jean Vuillemin
INRIA *

*Institut National de Recherche en Informatique et Automatique, 78150, Rocquencourt, France.

Abstract

We discuss a representation of the *computable real numbers* by continued fractions. This deals with the subtle points of undecidable and integer division, as well as representing the infinite $\infty = 1/0$ and undefined $\perp = 0/0$ numbers. Two general algorithms for performing arithmetic operations are introduced. The *algebraic algorithm*, which computes sums and products of continued fractions as a special case, basically operates in a positional manner, producing one term of output for each term of input. The *transcendental algorithm* uses a general formula of Gauss to compute the continued fractions of exponentials, logarithms, trigonometric functions, as well as a wide class of special functions. This work has been implemented in Le_Lisp [1]and the performance of these algorithms appears to be quite good; however, no competing system has been available for comparison.

Résumé

Nous étudions une représentation des réels calculables par fractions continues, qui prend en compte l'infini $\infty = 1/0$ et l'indéfini $\perp = 0/0$. Deux algorithmes généraux sont introduits pour effectuer les opérations. L'*Algorithme Algébrique*, qui se particularise au calcul de sommes et produits de fractions continues, opère de façon positionnelle, en produisant un terme du résultat par terme des paramètres. L'*Algorithme Transcendant*, fondé sur la fraction continue de Gauss, permet le calcul d'une vaste classe de fonctions spéciales (exp, log,...). Ces algorithmes ont été implantés en LeLisp, avec des performances très satisfaisantes.

God gave us the integers.
All the rest is man's work.

Leopold Kronecker

[1]Le_Lisp is a registered trade-mark by INRIA.

1 Introduction

In ancient Greece, the Pythagorean school sought to explain the *world* in terms of finite integers and their ratios. The discovery of the irrational nature of $\sqrt{2}$, worth sacrificing dozens of oxen, and kept secret for a century, dealt a serious blow to this *finitist* program. Uneasy with the many paradoxes associated with arguments involving infinity, the most famous being attributed to Zeno of Eleus, Greek mathematicians pragmatically replaced numbers by geometry at the heart of their preoccupations.

We wait until the XIX-th century, to find mathematically rigorous constructions of the real numbers. Any such construction starts from the choice of a *representation* for numbers: Cauchy sequence, Dedekind cut, interval, decimal, continued fraction, ... It then defines the arithmetic operations $+, -, \times, /$ in terms of the chosen representation, and must prove that the resulting structure is indeed a real field. Since all these constructions lead to isomorphic algebraic structures, we can then *abstract* from the chosen underlying representation, and start building Analysis, as we know it, without ever having to worry about Cauchy sequences or Dedekind cuts.

Soon after the discovery of computable functions by Turing [36] and Church [41], the notion of *computable* real numbers, also called *recursive*, *representable*, and *constructive* has received a great deal of mathematical attention [54],[66],[85]. Despite this, very little effort has been made to provide practical implementations of exact real computer arithmetic. Notable exceptions are [80] and [86], both of which are based on radix representations.

The present work originated with our own effort to implement exact reals in order to supplement arbitrary precision rational arithmetic in Le_Lisp [82]. Our starting point was the unpublished work of Gosper, on rational continued fraction arithmetic, an account of which is given in [81]. This programming effort has had its rewards: our first version, including transcendental functions, had only five pages of (Lisp) code. In a second version, the amount of code had doubled, for a tenfold gain in speed: if we restrict ourselves to rational operations, this implementation is only two to three times slower than a numerator/denominator representation of the rational numbers. Providing a complete justification of this code has brought out enough problems for us to seem worthwhile reporting in details on the construction. This paper is organized as follows:

- Section 2 presents a construction of the computable reals, based on intervals. We examine why "classical" comparison and integer division are not computable. As a consequence, neither normal radix nor continued fraction representations of numbers are directly suitable for the computable reals; we thus have to use alternative, computable, *redundant* representations. While it can be mostly ignored in classical mathematics, the issue of specific number representation is central to computer science: indeed, comparison and integer division are not effectively definable, independently of a number representation. We also argue that it is necessary, in such an implementation, to explicitly deal with *infinite* ($\infty = 1/0$) and *undefined* ($\perp = 0/0$) numbers. We conclude by showing that the computable reals, with ∞ and $\perp$, form a projective field.

$$\begin{array}{rcl}
\perp & = & [0\;0\;0\;0\;,0] \\
\infty & = & [1\;0\;1\;0\;,1\;0] = [2\;0\;2\;0\;,2\;0] = [3\;0\;9\;0\;,3^{n+3}\;0] \\
\frac{1+\sqrt{5}}{2} & = & [1\;1\;1\;1\;,1] = [2\;\;-3\;3\;\;-3\;3\;,-3\;3] \\
\sqrt{2} & = & [1\;2\;2\;2\;,2] \\
e & = & [2\;1\;2\;1\;1\;4\;1\;1\;6\;1\;,1\;2n{+}8\;1] = [3\;\;-4\;,2\;4n{+}5\;\;-2\;\;-4n{-}7] \\
\tan 1 & = & [2\;\;-2\;\;-4\;7\;,(-)^{n+1}(2n{+}9)] \\
 & & \\
\frac{\pi}{4} & = & \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 3 & 4 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 5 & 9 \\ 1 & 0 \end{bmatrix} , \begin{bmatrix} 2n+7 & (n+4)^2 \\ 1 & 0 \end{bmatrix}
\end{array}$$

Figure 1: Some Remarkable Continued Fractions.

- Section 3 presents a representation of the computable reals, including ∞ and $\perp$, by continued fractions. Such fractions are finite sequences of finite integers, except for the last term which is a *continuation.* A continuation is an algorithm for computing the next term of the continued fraction expansion, finitely represented by a *program* in some computer language. Evaluation of this continuation is only performed when a comparison involving that number ultimately requires to compute the corresponding term in the continued fraction. This provides an *automatic control* on the precision required for each represented number.
- Section 4 introduces the techniques of positional continued fraction arithmetic: in a suitably redundant representation system, our *algebraic algorithm* works in a positional manner, by successively reading the next input term, before producing each output term. It thus operates in much the same way as ordinary serial addition on radix b numbers, with obviously quite different carry rules. Special cases of the algorithm are shown to perform the operations $+, -, \times, /$ on continued fractions.
- Section 5 deals with non rational computations, through the *transcendental algorithm*, based on the use of Gauss continued fraction expansion of hypergeometric functions. As special cases, we obtain the exponential, logarithm and trigonometric functions, as well as many other interesting special functions.
- In each section, theorems are stated without proofs; these can be found in Section 7, following some concluding remarks in Section 6.

2 Computable Real Numbers

To serve as a reference, we first build the reals as limits of rational intervals, assuming that we have an exact, arbitrary precision, arithmetic on Q at our disposal.

2.1 Numbers as sequences of intervals

Definition 1 *A real number $r \in \mathbf{R}$ is represented by a sequence of intervals*

$$r(0) = [i(0), s(0)], \ldots, r(n) = [i(n), s(n)], \ldots$$

such that:

(i) Each interval $r(n)$ contains the following $r(n+1)$:

$$r(0) \supseteq r(1) \supseteq \cdots \supseteq r(n) \supseteq r(n+1) \supseteq \cdots.$$

(ii) The distance Δ between the end-points of the successive intervals vanishes,

$$\lim_{n \to \infty} \Delta(s(n), i(n)) = 0,$$

thus r is the unique number common to all intervals:

$$r = r(\infty) = \bigcap_{n \geq 0} r(n).$$

(iii) The end-points $i(n), s(n) \in \mathbf{Q}$ of each interval $r(n)$ are rational numbers, and the mapping $r : \mathbf{N} \mapsto [\mathbf{Q}, \mathbf{Q}]$ from natural numbers $\mathbf{N}$ to intervals $[\mathbf{Q}, \mathbf{Q}]$ is a computable function.

To be specific, let us consider two examples of computable real numbers:

- The golden ratio $\phi = (1 + \sqrt{5})/2$ can be represented by the function

$$\phi = \lambda n.[\frac{F_{2n+2}}{F_{2n+1}}, \frac{F_{2n+1}}{F_{2n}}],$$

where $F_0 = 0, F_1 = 1, \ldots, F_{n+2} = F_{n+1} + F_n, \ldots$ are the Fibonacci numbers.

- Fermat conjectured that equation $x^n + y^n = z^n$ has no integer solution for $n > 2$. An elementary argument shows that counter-examples to this conjecture are necessarily of the form $1 < x < y < z, 2 < n < z$. For any fixed integer z, we can thus effectively compute the number $\mathcal{F}_z$ of solutions to Fermat's equation, and define the number

$$\mathcal{F}_{ermat} = \lambda z.[\sum_{i \leq z} \frac{\mathcal{F}_i}{i!}, \sum_{i \leq z} \frac{\mathcal{F}_i}{i!} + 1/z].$$

Number $\mathcal{F}_{ermat}$ is a perfectly legitimate member of the computable real numbers, and it is easy to show that $0 \leq \mathcal{F}_{ermat} < 2$. Of course, $\mathcal{F}_{ermat} = 0$ if and only if Fermat was right in his conjecture!

2.2 Numbers as programs

Without condition (iii), the numbers defined above would be isomorphic to those of Cauchy, Dedekind and Cantor, the *classical* reals. Condition (iii) demands that the interval sequence defining a number be given in an *effective* manner.

To do so, we introduce a *programming language*, say $\mathcal{L}$. We need that $\mathcal{L}$ be *universal*, so we can program in $\mathcal{L}$ a Turing Machine [36] simulator and, equivalently [41], a λ-calculus evaluator. The reader should feel free to identify $\mathcal{L}$ with the programming language she or he is most familiar with: FORTRAN, PASCAL, C, Lisp, ADA, ...

Language $\mathcal{L}$ has a *syntax*, defining valid $\mathcal{L}$-expressions as sequences of letters in some finite alphabet. It also has a *semantics*, defining an *evaluation* mechanism, mapping $\mathcal{L}$-expressions into $\mathcal{L}$-expressions.

A real number r is represented by an $\mathcal{L}$-expression, which defines the algorithm for computing $r(n) = [i(n), s(n)]$ from input $n \in \mathbf{N}$.

Theorem 1 (Cantor) *Let $\mathcal{R}$ denote the subset of $\mathcal{L}$-expressions which represent numbers, that is the computable reals expressed as programs in $\mathcal{L}$.*

(i) The set $\mathcal{R}$ is denumerable.

(ii) No algorithm can effectively enumerate all the elements of $\mathcal{R}$.

As equivalent to (ii), we see that no algorithm can decide, in finite time, if an arbitrary $\mathcal{L}$-expression e denotes a number, i.e. $e \in \mathcal{R}$. Therefore, since we cannot expect the computer to do it for us, we must provide a correctness proof with every real number definition, i.e. program. In order to relieve the $\mathcal{L}$ programmer from that burden, language designers should endow $\mathcal{L}$ with an abstract data-type $\mathcal{A}$ containing number 1, and closed under the operators $+, -, \times, /$, as well as roots, exponentials, logarithms, ...:

$$\begin{array}{rcl} 1 & \in & \mathcal{A} \\ \mathcal{A}+\mathcal{A} & \subset & \mathcal{A} \\ -\mathcal{A} & \subset & \mathcal{A} \\ \mathcal{A}\times\mathcal{A} & \subset & \mathcal{A} \\ 1/\mathcal{A}_{\neq 0} & \subset & \mathcal{A} \\ \sqrt{\mathcal{A}}_{\geq 0} & \subset & \mathcal{A} \\ e^{\mathcal{A}} & \subset & \mathcal{A} \\ \log \mathcal{A}_{>0} & \subset & \mathcal{A} \end{array}$$

All numbers in $\mathcal{A}$ can be used freely, without correctness proof; the proofs must have been provided, once and for all by the implementors of the language. By Theorem 1, no matter how many operators we put into $\mathcal{A}$, it remains a strict subset of $\mathcal{R}$; this leaves for the $\mathcal{L}$ programmer, an occasional need to "hand code" certain special numbers, such as, for example, our $\mathcal{F}_{ermat}$ example.

2.3 What the best computers cannot do

Describing one possible implementation of data-type $\mathcal{A}$ is the object of this work. Before doing so, we need to identify some operators, mainly comparison and euclidean division, which are taken for granted in traditional exact integer and rational arithmetic[2], yet *cannot* be part of $\mathcal{A}$, for the following reason:

Theorem 2 (Turing,Rice) *Let $r \in \mathbf{R}$ represent an arbitrary computable real number. No algorithm can compute, in finite time:*

1. *if number r is null, i.e. $r = 0$;*
2. *if $r > r'$, for* any *fixed $r' \in \mathbf{R}$;*
3. *if number r is rational, i.e. $r \in \mathbf{Q}$;*
4. *the first digit of the decimal expansion of r;*
5. *the first term $\lfloor r \rfloor \in \mathbf{Z}$ of the regular continued fraction expansion of r.*
6. *the value $f(r) \in \mathbf{R}$ of any function f which is* not continuous *at r.*

As a consequence, no algorithm can decide equality $r = r'$ of arbitrary computable reals in finite time. Indeed, if $r \neq r'$, the approximate intervals $r(n)$ and $r'(n)$ will be disjoint, for n large enough. We could at this point adopt Eudoxus (c. 375 B.C.) definition of equality between real numbers: $r = r'$ means that $r < q \iff r' < q$, for all rationals q. This, however, does not provide us with an effective way of finding out when $r = r'$, in a finite time. The alternative is to resort to some type of symbolic manipulation system, by incorporating in the evaluation mechanism of language $\mathcal{L}$ useful identities, such as, for example:

$$\log e^r \vdash r.$$

Expression $r \vdash r'$ denotes that r has been reduced to r' by an effective succession of equality preserving symbolic transformations, hence that $r = r'$ has been effectively established.

Symbolic manipulation systems, such as Macsyma or Reduce, have shown the power of these techniques, which often exceed human capabilities. A system for computing effective equalities $r \vdash r'$, and an implementation of the computable reals, effectively computing un-equalities $r \neq r'$, are *dual* weapons in the arsenal of the modern mathematician. While the latter should always succeed in establishing $r \neq r'$, the former, by Theorem 2, can only do a partial job and there are (infinitely many) numbers $r = r'$, such as $r \not\vdash r'$. Each number, say 2, is represented by infinitely many expressions in $\mathcal{R}$, partitioned into equivalence classes by relation $\vdash$. Assuming that $\mathcal{L}$ possesses an exact integer arithmetic, exactly one such class $\{r \in \mathcal{R} : r \vdash 2\}$ contains integer 2 and its equivalent representations, recognized as such. The other representations of number two are all obtained as infinite limits, say 2+. We call them *Zeno's integers*: although the value of each such numbers is an integer, e.g. 2=2+, they are not recognized as such by the system, i.e. $2 \not\vdash 2+$.

[2]Not so in floating point arithmetic, where the status of comparison is rather murky, and integer division not usually available.

2.4 Approximate Comparison

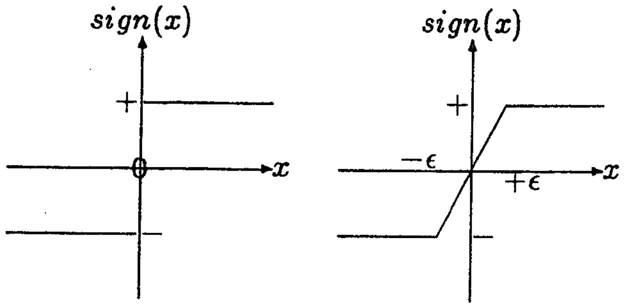

Figure 2: Discontinuous non computable exact *sign*, and its continuous computable ϵ-approximation.

The *sign* of number r is a subset of $\{+, 0, -\}$, defined as:

$$\begin{aligned} sign(r) \equiv \{+\} &\iff r > 0; \\ sign(r) \equiv \{0\} &\iff r = 0; \\ sign(r) \equiv \{-\} &\iff r < 0. \end{aligned}$$

Using *sign*, we define the comparison operator ? between numbers r and r' as

$$r?r' \equiv sign(r - r');$$

this, in turn, yields the six related operators:

$$\begin{array}{llllll} r < r' \iff & \{-\} \equiv r?r'; & r = r' \iff & \{0\} \equiv r?r'; & r > r' \iff & \{+\} \equiv r?r'; \\ r \geq r' \iff & - \notin r?r'; & r \neq r' \iff & 0 \notin r?r'; & r \leq r' \iff & + \notin r?r'. \end{array}$$

Unfortunately, by Theorem 2, none of these operators is computable! Suppose for example that we want to find the sign of a very small number r. After computing successive approximations to r for a while, we may find ourselves in the situation where r has not been found to be $\neq 0$, yet it has not been established that $r \vdash 0$ either. To effectively "time out" this potentially infinite search for the *sign* of r, we choose a small positive rational $0 < \epsilon < 1$, and post the decree:

$$sign(r) \equiv \{-, 0, +\} \quad if \quad |r| < \epsilon.$$

We thus give up when the "normal" *sign* has not been determined, and r has been found to be smaller than ϵ. In some cases, we can refine the *sign* information to $\{-, 0\}$, or $\{0, +\}$, the latter being the case for our $\mathcal{F}_{ermat}$ example. For $r = \infty$, to be introduced latter, we have $sign(r) \equiv \{-, +\}$ as soon as $|1/r| < \epsilon$.

Remark 1 *The sign operator computes a subset of $\{-, 0, +\}$ instead of a single element. For some small numbers r, the value of $sign(r)$ depends upon the representation of r: two different representations r_0 and r_1 of the same number r may be assigned two different signs, e.g. $sgn(r_0) \equiv \{0, +\}$ and $sgn(r_1) \equiv \{+\}$; if $\sigma(r)$ is the* true *mathematical sign of r, we have $\sigma(r) \in sign(r_n)$, for any representation r_n of r.*

2.5 Approximate Integer Division

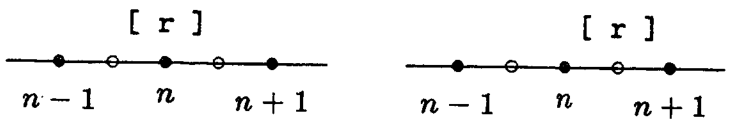

Figure 3: Intervals $[r]$ such that $[r] \div 1 = n > 0$.

Let us review three classical manners to define an integer $z \in \mathbf{Z}$ "close" to a given number $r \in \mathbf{R}$:

$$\begin{array}{rlll} \textbf{floor:} & z = \lfloor r \rfloor & \text{is such that} & 0 \leq r - z < 1; \\ \textbf{round:} & z = \lfloor r \rceil & \text{is such that} & -\frac{1}{2} < r - z \leq \frac{1}{2}; \\ \textbf{ceiling:} & z = \lceil r \rceil & \text{is such that} & -1 < r - z \leq 0. \end{array}$$

By Theorem 2, floor and ceiling are not computable when r is a Zeno integer; round is not computable when $r + 1/2$ is a Zeno integer. Contrast this observation with:

Algorithm 1 (Euclidean part of a finite number) *Let $r \in \mathbf{R}$ be a* finite *computable real. The Euclidean part $z = r \div 1$ of r is an integer $z \in \mathbf{Z}$, computed in finite time, such that $-1 < r - z < 1$.*

Implementation: Compute an approximant $r(n) = [i(n), s(n)]$ with $|s(n) - i(n)| < 1/2$, and choose:

$$\begin{array}{rcll} r \div 1 & = & \lfloor i(n) \rceil & \text{when} \quad 0 \leq i(n); \\ r \div 1 & = & \lfloor s(n) \rceil & \text{when} \quad i(n) < 0. \end{array}$$

This computation can be performed with exact rational arithmetic, since $i(n), s(n) \in \mathbf{Q}$. From this choice of $z = r \div 1$, we can derive:

$$\begin{array}{lll} z > 0 & \text{implies} & r - z \in (-\frac{1}{2}, 1); \\ z = 0 & \text{implies} & r - z \in (-1, 1); \\ z < 0 & \text{implies} & r - z \in (-1, \frac{1}{2}). \end{array}$$

■

Remark 2 *The value of $r \div 1$, which is either $\lfloor r \rfloor$ or $\lceil r \rceil$, depends upon the* representation *of r, as in Remark 1.*

2.6 Computing with infinity

$+$	0	1	∞	$\perp$
0	0	1	∞	$\perp$
1	1	2	∞	$\perp$
∞	∞	∞	$\perp$	$\perp$
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

$\times$	0	1	∞	$\perp$
0	0	0	$\perp$	$\perp$
1	0	1	∞	$\perp$
∞	$\perp$	∞	∞	$\perp$
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

$-$	0	1	∞	$\perp$
0	0	-1	∞	$\perp$
1	1	0	∞	$\perp$
∞	∞	∞	$\perp$	$\perp$
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

$/$	0	1	∞	$\perp$
0	$\perp$	0	0	$\perp$
1	∞	1	0	$\perp$
∞	∞	∞	$\perp$	$\perp$
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

Figure 4: Operation tables for ∞ and $\perp$.

Since, by Theorem 2, we cannot always test whether or not $r \neq 0$, we cannot avoid dividing by 0 in the computation of $1/r$! We thus have to incorporate an infinite number

$$\infty = \frac{1}{0}$$

in our collection of computable reals. For the same reason, we cannot avoid having to deal with an undefined number

$$\perp = \frac{0}{0} = 0 \times \infty = 0^0 = \infty - \infty.$$

Undefined $\perp$ should be regarded as an expression which says: "I am a number"; in view of Theorem1(ii), this is indeed a useful information; apart from that, $\perp$ carries no information whatever regarding its value, which we identify with the interval of all *defined* numbers, including ∞:

$$\perp = (-\infty, +\infty].$$

The result of *any* operation involving $\perp$ is undefined, and the operation tables for $\perp$ and ∞ are given in Figure 4. To distinguish $\perp$ and ∞ from other numbers, we say that $r \in \mathbf{R}$ is *defined* when $r \neq \perp$, and that r is *finite* when $r \neq \perp$ and $r \neq \infty$.

2.7 Stereographic representation of R

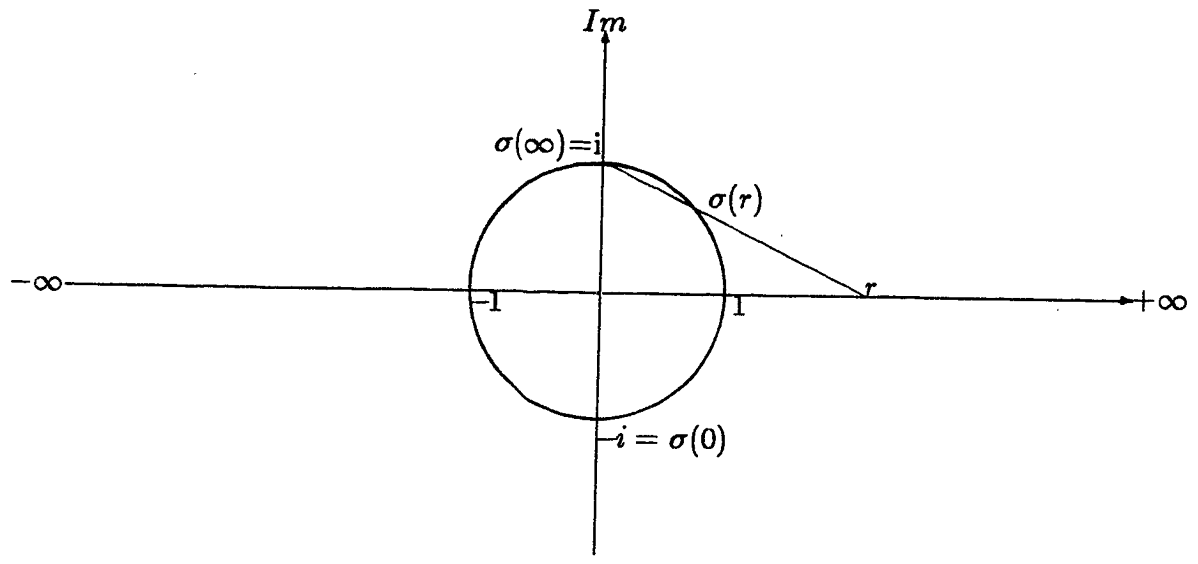

Figure 5: Stereographic representation of **R**.

In order to treat ∞ as any other number, it is convenient to use the stereographic representation of the real line onto the unit complex circle centered at the origin, as shown in Figure 5. The image

$$\sigma(r) = \frac{ir+1}{r+i} = \frac{2r}{r^2+1} + i\frac{r^2-1}{r^2+1}$$

of point $r \in (-\infty, +\infty]$ is found by intersecting the unit complex circle with the line joining r to the pure imaginary point i. Conversely, any point $\sigma = e^{i\theta}$ of the circle is the stereographic image of $r = \frac{\cos\theta}{1-\sin\theta}$. The length Δ of the chord joining $\sigma(x)$ to $\sigma(y)$ is given by:

$$\Delta(x,y) = \frac{2|x-y|}{\sqrt{(1+x^2)(1+y^2)}}. \tag{1}$$

In particular:

$$\Delta(0,r) = \frac{2|r|}{\sqrt{1+r^2}}, \Delta(r,\infty) = \frac{2}{\sqrt{1+r^2}}, \Delta(0,1) = \Delta(1,\infty) = \sqrt{2}.$$

From its geometric definition, it is clear that $0 \leq \Delta(x,y) \leq 2$, and (1) defines a *finite* distance between arbitrary numbers x and y:

$$\begin{aligned} \Delta(x,y) &= \Delta(x,z); \\ \forall z : \Delta(x,y) &\leq \Delta(x,z) + \Delta(z,y). \end{aligned}$$

We shall use Δ to discuss convergence at infinity as well as at any other finite point.

2.8 Interval arithmetic

In the stereographic correspondence, interval $[r, r']$ is mapped onto the arc $\widehat{\theta,\theta'}$ of the circle joining $\sigma(r) = e^{i\theta}$ to $\sigma(r') = e^{i\theta'}$ counterclockwise. Conversely, any such arc $\widehat{\theta,\theta'}$ is the image of an interval $[r,r']$. The novelty here lies in *open* intervals, such as $(2,-2)$, which represents the set $\{r \in \mathbf{R} : r > 2 \text{ or } r < -2\}$ of points whose absolute value is greater than two; it is the complement of interval $[-2,2] = \{r : -2 \leq r \leq 2\}$.

Definition 2 *Intervals may be classified as follows:*

1. *A finite interval* $[i,s]$, *with* $i < s$, *represents* $\{r \in \mathbf{R} : i \leq r \leq s\}$. *We use parentheses to exclude the corresponding bound, as in* $(i,s] = \{r \in \mathbf{R} : i < r \leq s\}$; *similarly for* $[i,s)$ *and* (i,s). *A finite interval* $[i,s]$ *is* positive *if* $i > 0$, negative *if* $s < 0$; *it* contains zero *otherwise.*
2. *An infinite interval* $[i,s]$, *with* $i > s$, *represents the complement* $\{r \in \mathbf{R} : r \notin (s,i)\}$ *of the finite interval* (s,i); *similarly for* $(i,s], [i,s)$ *and* (i,s).
3. *An interval of length zero, is either* $[i,i] = i$, *or* $(i,i) = \{r \in \mathbf{R} : r \neq i\}$; *it can also be the empty interval* $\emptyset$, *denoting a non-number, and its complement* $\perp$, *the set of all defined numbers. Interval notation being ambiguous in this case, we adopt the (arbitrary) convention that* $[i,i) = \emptyset$ *and* $(i,i] = \perp$; *indeed, we use* $(0+,0-] = (-\infty,+\infty] = \perp$ *to remove any ambiguity.*

Arithmetic operations on intervals are defined as follows:

- **Addition:** The sum of two intervals $[i,s]$ and $[i',s']$ is given by:

$$[i,s] + [i',s'] = [i+i', s+s'],$$

whenever $i \leq s$, $i' \leq s'$, or $i \leq s, i' > s', i+i' > s+s'$, or $i > s, i' \leq s', i+i' > s+s'$. In all the other cases,

$$[i,s] + [i',s'] = \perp .$$

- **Substraction:** The opposite of interval $[i,s]$ is given by:

$$-[i,s] = [-s,-i].$$

- **Multiplication:** The product of intervals $[i,s]$ and $[i',s']$ is given by:

$$[i,s] \times [i',s'] = [min(ii', is', si', ss'), max(ii', is', si', ss')],$$

if $\infty \notin [i,s]$ and $\infty \notin [i',s']$. If $0 \notin [i,s], 0 \notin [i',s']$ and $\infty \in [i,s]$ or $\infty \in [i',s']$, then:

$$[i,s] \times [i',s'] = [min(|ii'|, |is'|, |si'|, |ss'|), -min(|ii'|, |is'|, |si'|, |ss'|)].$$

In all the other cases,

$$[i,s] \times [i',s'] = \perp .$$

- **Division:** The inverse of interval $[i,s]$ is given by:

$$1/[i,s] \;=\; [1/s, 1/i].$$

By Definition 1, a real number is the limit of an effective sequence of enclosing intervals with vanishing length. Set $\mathbf{R}$ is thus made of finite points $[r,r]$, together with infinity $[\infty,\infty]$ and undefined $\bot = (-\infty,+\infty]$ (see Definition 2, case 3). With the operations above, $\mathbf{R}$ is a field in the usual algebraic sense, to within a finite number of exceptions to the field properties, having to do with ∞ and $\bot$:

Theorem 3 *For arbitrary numbers $x, y, z \in \mathbf{R}$:*

$$\begin{array}{rcll}
x+y &=& y+x; & & (+)\\
x+(y+z) &=& (x+y)+z; & & (++)\\
(-x)+x &=& 0 & \textit{for}\text{ finite } x; & (-)\\
x\times y &=& y\times x; & & (\times)\\
x\times(y\times z) &=& (x\times y)\times z; & & (\times\times)\\
x\times(1/x) &=& 1 & \textit{for } x\neq 0 \text{ finite;} & (/)\\
x\times(y+z) &=& (x\times y)+(x\times z) & \textit{for } x \text{ not infinite.} & (\times+)
\end{array}$$

3 Representations of R by continued fractions

A direct computer implementation of $\mathbf{R}$, based on the interval construction of the preceeding section, has two drawbacks:

(i) Choosing to represent each number r by a unique function $\mathbf{N} \mapsto [\mathbf{Q},\mathbf{Q}]$ blends well with a (higher order) functional programming style. It has the unfortunate property of being memoryless: if a previous computation has required to compute a specific number, say π, to within a great precision 10^{-100}, this has no beneficial effect on the speed of subsequent high precision computations involving the same number π.

(ii) Direct implementation of interval arithmetic with rational end-points, as described in Section 2, is not efficient for lack of control over the size of the integers representing numerators and denominators of the end-points; these can grow very quickly, making such a system unusable.

In this section, we first discuss possible solutions to problems (i) and (ii). We review some classical notations and results regarding continued fractions, and use them to choose a computable representation for infinite continued fractions.

3.1 Finite representations of infinite sequences of numbers

An elegant way to solve problem (i) is to represent a number as a *stream*

$$\langle z_0 z_1 \cdots z_{n-1} r(n)\rangle. \tag{2}$$

The n first terms of such a stream are integers, and the last one $r(n)$ is a *continuation*, also commonly called *closure* in the Lisp community. This continuation is a function, finitely represented in language $\mathcal{L}$, which is capable of evaluating the next integer term z_n, as well as the next continuation $r(n+1)$.

In the decimal system, the value (interpretation) $\mathcal{V}$ of such a stream (2) is given by:

$$\mathcal{V}\langle z_0 z_1 \cdots z_{n-1} r(n)\rangle = \sum_{0 \le i < n} z_i 10^{-i} + \frac{\mathcal{V}\langle r(n)\rangle}{10^n}.$$

With continued fractions, we have:

$$\mathcal{V}[z_0 z_1 \cdots z_{n-1}] r(n) = z_0 + \cfrac{1}{z_1 + \cfrac{1}{\ddots + \cfrac{1}{z_{n-1} + \cfrac{1}{\mathcal{V}[r_n]}}}}.$$

We must however be careful in our formulation, since Theorem 2 shows that neither decimal[3], nor continued fraction, representations are computable! A solution, advocated in this context by Wiedmer [80], is to resort to *redundant* representations, such as those used for a long time in finite hardware arithmetic. By doing so, we lose *uniqueness* of representation, and gain the ability to compute longer and longer representations, which *monotonically* increases our approximation to the result. Indeed, in any such representation system, it is sufficient to control the variation interval of the continuation

$$\mathcal{V}\langle r(n)\rangle \in [i, s],$$

with enough precision to ensure the interval containment

$$\mathcal{V}\langle z_0 z_1 \cdots z_{n-1}[i,s]\rangle \supset \mathcal{V}\langle z_0 z_1 \cdots z_n[i,s]\rangle \tag{3}$$

of Definition 1(i). In the decimal system, we may for example set

$$z_0 = r \div 1, r(1) = r - z_0;$$

and compute the digits z_{n+1} after the decimal point, by:

$$z_{n+1} = 10r(n+1) \div 1, r(n+2) = r(n+1) - z_{n+1}.$$

The digits of this balanced decimal system, used by [86], are such that $-10 < z_{n+1} < 10$, since we control the variation of the continuation by $r(n+1) \in (-10, 10)$. Containment of the successive intervals (3) follows from:

$$z_n + \frac{1}{10} r(n) \in [-9, 9] + \frac{1}{10}(10, -10) \subseteq (-10, 10).$$

[3]The decimal representation of r is not computable, when $10^k r$ is a Zeno integer, for some $k \in \mathbf{Z}$, the problem arising from the identity $1 = 0.999\cdots$.

What ultimately forces to increase the precision, hence the length of the stream representing a given number, is the operation ? of comparison, or rather, as noted earlier, its computable analog, the ϵ-comparison. In the redundant decimal representation above, comparison $x?y$ can be performed digit-wise, starting from the most significant $t = 0$:

1. if $t > -\log_{10}\epsilon$, numbers x and y are called ϵ-equals, and the computation terminates;
2. on equal digits, $x_t = y_t$, the comparison proceeds with the next digit pair $t + 1$;
3. if digits x_t and y_t differ by more than one unit, the process stops with $x?y = x_t?y_t$;
4. if digits x_t and y_t differ by one unit, we decide, based upon the values of the next digits x_{t+1}, y_{t+1}, which of the previous two cases is applicable.

If either of the streams representing x and y has less than the number of digits required to perform the comparison, the corresponding continuation procedure is automatically called and the computed result memorized in a longer stream, extending the previous one. Later in this section, we present a redundant continued fraction representation having the same basic property. We prove [75], in a rather general setting, that this type of *call by need*, also known as *lazy* evaluation, is *optimal*, with respect to the total number of procedure calls.

The decimal representation (or any radix 2^k generalization) described above does not however provide a satisfactory solution to problem (ii). Indeed, drawing conclusions from [86], Boehm [87] abandons stream representation, in order to be able to "round-off" intervals to the nearest b-adic integer; he argues that the advantages of all integer arithmetic overcome the handicap of not being able to memorize earlier computations.

With continued fractions, we can combine the best of both worlds: it has been known for a long time that approximants associated to the continued fraction of number r have the *smallest* possible denominator, among all rational approximations closer to r. In this sense, classical continued fractions provide the best possible solution to problem (ii). While we have to use redundant (non-classical) continued fractions in order to also solve problem (i), we shall see that this does not change the representation size of the rational approximants involved.

3.2 Classical N and Z continued fractions

Let $r = r_0$ be a number; compute:

$$r_0 = z_0 + \frac{1}{r_1},\ r_1 = z_1 + \frac{1}{r_2},\ \ldots,\ r_t = z_t + \frac{1}{r_{t+1}},$$

where, at each step t, integer $z_t \in \mathbf{Z}$ is "close" to r_t. Different choices of z_t lead to different types of continued fractions:

$$\text{N-fractions:} \quad z_t = \lfloor r_t \rfloor; \tag{4}$$

$$\text{Z-fractions:} \quad z_t = \lfloor r_t \rceil. \tag{5}$$

By eliminating $r_1, \ldots, r_t$, we write, using standard matrix notation:

$$\begin{bmatrix} r_0 \\ 1 \end{bmatrix} = \begin{bmatrix} z_0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} z_1 & 1 \\ 1 & 0 \end{bmatrix} \cdots \begin{bmatrix} z_t & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_{t+1} \\ 1 \end{bmatrix}.$$

For the sake of notational simplicity, we use, in place of the above:

$$r_0 = [z_0\, z_1\, \cdots\, z_t] r_{t+1}.$$

The name, *continued fraction*, comes from the identity:

$$r_0 = z_0 + \cfrac{1}{z_1 + \cfrac{1}{\ddots + \cfrac{1}{z_t + \cfrac{1}{r_{t+1}}}}}.$$

Consider for example the continued fraction expansions of $\phi, \sqrt{2}, \sqrt{3}, e$ and π:

- N-fraction:
  - $\phi = \frac{1+\sqrt{5}}{2} = [1\,1\,1\,1 \cdots] = [\,,1]$.
  - $\sqrt{2} = [1\,2\,2\,2 \cdots] = [1\,,2]$.
  - $\sqrt{3} = [1\,1\,2\,1\,2\,1\,2 \cdots] = [1\,,1\,2]$.
  - $e = [2\,1\,2\,1\,1\,4\,1\,1\,6\,1 \cdots] = [2\,,1\,2n+2\,1]$.
  - $\pi = [3\,7\,15\,1\,292\,1\,1\,1\,2\,1 \cdots]$.
- Z-fraction:
  - $\phi = \frac{1+\sqrt{5}}{2} = [2\ -3\,3\ -3\,3 \cdots] = [2\,,-3\,3]$.
  - $\sqrt{2} = [1\,,2]$.
  - $\sqrt{3} = [2\ -4\,4\,,-4\,4] = [2\,,-4\,4]$.
  - $e = [3\ -4\,2\,5\ -2\ -7\,2\,9\ -2 \cdots] = [3\ -4\,,2\,4n+5\ -2\ -4n-7]$.
  - $\pi = [3\,7\,16\ -294\,3\ -3 \cdots]$.

Notation 1 (Use of comma to mark periodic terms) *We use here the comma "," to mark the beginning of a periodic sequence of numbers, e.g.:*

$$[1\,,1\,2] = [1\,1\,2\,,1\,2] = [1\,1\,,2\,1] = [1\,,1\,2\,1\,2].$$

The same convention applies for writing the decimal representation of rational numbers, as in $\frac{1}{3} = 0.3{,}3 = 0.33{,}3$ *or* $\frac{1}{7} = 0.1{,}428571$.

When the symbolic variable n appears in the expression to the right of the comma, that expression should be repeated for each successive integer values $n = 0, n = 1, \ldots,$ as in:

$$e = [,1\,2n\,1] = [1\,0\,1\,,1\,2n+2\,1] = [,1\,4n\,1\,1\,4n+2\,1].$$

Similarly for $\frac{\pi^2}{6} = [1\,,0\,\frac{1}{(n+1^2)}]$.

Theorem 4 (Lagrange) *Let $r_0 = [z_0\ z_1\ \cdots\ z_{n+1}\ \cdots]$ be the* **N** *or* **Z** *continued fraction expansion of some finite number $r = r_0$.*

(i) The continued fraction is finite *$r_0 = [z_0\ z_1\ \cdots\ z_{n+1}]\infty$ if and only if number $r \in \mathbf{Q}$ is rational. It has exactly one term $r_0 = [z_0]\infty$ if and only if number $r \in \mathbf{Z}$ is integer.*

(ii) The continued fraction is eventually periodic *$r_0 = [z_0\ z_1\ \cdots\ z_{n-1}\ ,z_n \cdots z_{n+p}]$ if and only if number $r \in \sqrt{\mathbf{Q}_{>0}}$ is the irrational square root of a positive rational.*

(iii) The continued fraction approximants $q_n = [z_0\ z_1\ \cdots\ z_n]\infty$ converge to r with speed: $|r - q_n| < \phi^{-n}$, where $\phi = [,1] = 1.618\cdots$ is the golden ratio.

The following result shows how the choice of the integer part of r_{n+1} affects the terms z_{n+1} of the continued fraction:

Theorem 5 (Hurwitz) *Let $r_0 = [z_0\ z_1\ \cdots\ z_{n+1}\ \cdots]$ be the continued fraction expansion of some finite number $r = r_0$.*

(i) **N***-fractions are characterized as follows:*

– *All terms $z_1, \ldots, z_{n+1}, \ldots$, past the first one z_0, are positive integers:*

$$n \geq 0 \quad \textit{implies} \quad z_{n+1} \geq 1.$$

– *If the fraction is finite, its last term is greater than 1:*

$$n \geq 0, r_{n+2} = \infty \quad \textit{implies} \quad z_{n+1} \geq 2.$$

(ii) **Z***-fractions are characterized as follows:*

– *All terms $z_1, \ldots, z_{n+1}, \ldots$, past the first one z_0, are integers in $[2, -2]$:*

$$n \geq 0 \quad \textit{implies} \quad |z_{n+1}| \geq 2. \tag{6}$$

Furthermore, when $|z_{n+1}| = 2$, the subsequent term z_{n+2} has the same sign as z_{n+1}:

$$\begin{array}{rl} n \geq 0, z_{n+1} = 2 & \textit{implies} \quad z_{n+2} \geq 2, \\ n \geq 0, z_{n+1} = -2 & \textit{implies} \quad z_{n+2} \leq -2. \end{array} \tag{7}$$

– *If the fraction is finite, its last term is different from -2:*

$$n \geq 0, r_{n+2} = \infty \quad \textit{implies} \quad z_{n+1} \neq -2. \tag{8}$$

From Theorem 5(ii), we see that the $n+1$-rst continuation $r(n+1) = [z_{n+1} \cdots]$ of the **Z** fraction $r = [z_0 z_1 \cdots]$ belongs to the interval:

$$[z_{n+1} \cdots] \in (|z_{n+1}| - \frac{1}{2}, \frac{1}{2} - |z_{n+1}|) \cap [2, -2).$$

3.3 Redundant Euclidean continued fractions

We now attempt to approximate the **Z**-fraction of number $r = r_0$, replacing the choice $z_t = \lfloor r_t \rceil$ by its computable analog $z_t = r_t \div 1$. This leads to a well defined notion of continued fraction, having properties analogous to the N and **Z** fractions for *irrational* numbers. Indeed, the computation defined in Algorithm 1 always terminates for *finite* r_t. However, it does not terminate if $r = \perp$ is undefined or $r = \infty$ is infinite; it also fails if $r_t = \infty$, i.e. when r is a (Zeno) rational. To properly handle all these cases, we have to change our computation of the Euclidean part:

Algorithm 2 (Euclidean part of an arbitrary computable number) *Let $r \in \mathbf{R}$ be an* arbitrary *computable real. The euclidean part $z = r \div 1$ of r is an integer $z \in \mathbf{Z}$, computed in finite time, such that $\Delta(r, z) < 1$.*

Implementation: Compute an approximant $r(n) = [i(n), s(n)]$ whose end-points are close enough, namely $\Delta(s(n), i(n)) < 1$. Let $z_i = \lfloor i(n) \rceil$ and $z_s = \lfloor s(n) \rceil$ denote the integers nearest to the end-points. Choose:

$$\begin{array}{rclll} r \div 1 & = & 0 & \text{when} & 0 \in [i(n), s(n)]; \\ r \div 1 & = & \lfloor i(n) \rceil & \text{when} & 0 \notin [i(n), s(n)], \text{ and } |z_i| \le |z_s|; \\ r \div 1 & = & \lfloor s(n) \rceil & \text{when} & 0 \notin [i(n), s(n)], \text{ and } |z_i| > |z_s|. \end{array} \tag{9}$$

This computation can be performed with exact rational arithmetic, since $i(n), s(n) \in \mathbf{Q}$. ■

We see that any *infinite* **Z** or **N** fraction is also an **E**-fraction. The **E**-fractions are characterized by:

Theorem 6 *Let $r_0 = [z_0\ z_1\ \cdots\ z_{n+1}\ \cdots]$ be some* **E** *continued fraction expansion of number $r = r_0$; let $r_n = [z_n\ z_{n+1}\ \cdots]$ represent the n-th continuation of r.*

(i) *All terms are identical to zero if and only if $r = \perp$ is undefined. Assuming from now on that $r \neq \perp$ is* defined, *we can effectively eliminate all pairs of consecutive 0 at the begining of r, so that one of the first two terms z_0, z_1 is non zero.*

(ii) *The n-th continuation r_n belongs to the interval: $\mathcal{V}(r_n) = (|z_n| - \frac{1}{2}, \frac{1}{2} - |z_n|)$, for $z_n \neq 0|$, and $\mathcal{V}(r_n) = (-2, 2)$ for $z_n = 0$. The rational approximants $[z_0\ z_1\ \cdots\ z_{n-1}]\mathcal{V}(r_n)$ converge to r:*

$$\mathcal{V}(r_0) \supset \cdots \supset [z_0\ z_1\ \cdots\ z_{n-1}]\mathcal{V}(r_n) \supset \cdots \supset r = [z_0\ z_1\ \cdots].$$

(iii) *Adjacent terms z_n and z_{n+1} are never simultaneously null: $|z_{n+1}| + |z_{n+2}| \ge 2$. If successive odd terms are null, $z_{n+2t+1} = 0$ for $0 \le t \le k$, the partial sums of such terms have increasing absolute values:*

$$|\sum_{i \le 2t+1} z_{i+n}| \;>\; |\sum_{i \le 2t-1} z_{i+n}|.$$

A number r is rational if and only if it terminates as:

$$r_k = \infty \;=\; [z_k\ 0\ z_{k+2}\ 0\ \cdots\ z_{k+2+2n}\ 0\ \cdots].$$

(iv) The terms z_n can be characterized as follows:

$$\begin{aligned} z_{n+1} = 0 &\quad \textit{implies} \quad z_n z_{n+2} \neq 0; \\ z_n z_{n+1} < 0 &\quad \textit{implies} \quad |z_{n+1}| \geq 2; \\ z_n z_{n+1} < 0, |z_{n+1}| = 2 &\quad \textit{implies} \quad z_{n+1} z_{n+2} > 0. \end{aligned}$$

To summarize, one can derive the following information from *the first two terms* of any **E** continued fractions $r = [z_0 z_1 \cdots]$:

1. If $z_0 = z_1 = 0$, number $r \in (-\infty, +\infty]$ is *potentially undefined.*
2. If $z_0 \neq 0, z_1 = 0$, number $r \in (|z_0| - \frac{1}{2}, \frac{1}{2} - |z_0|)$ is *potentially infinite.*
3. If $z_0 = 0, z_1 \neq 0$, number $r \in (-\frac{1}{2}, \frac{1}{2})$ is *potentially zero.*
4. If $z_0 > 0, z_1 \neq 0$, number $r \in (z_0 - \frac{1}{2}, z_0 + 1)$ is *positive.*
5. If $z_0 < 0, z_1 \neq 0$, number $r \in (z_0 - 1, z_0 + \frac{1}{2})$ is *negative.*

3.4 Normalization of Euclidean Continued Fractions

The following, easily verified identities, are useful in understanding the relations between the **N**,**Z** and **E** fractions of a number r:

$$\begin{array}{lcll} [z \;\; 0 \;\; z']r &=& [z+z']r & (\nu_0) \\ [z \;\; 1 \;\; z']r &=& [z+1 \;\; -z'-1]-r & (\nu_1) \\ [z \;\; -1 \;\; z']r &=& [z-1 \;\; -z'+1]-r & (\nu_{-1}) \\ [z \;\; 2 \;\; z']r &=& [z+1 \;\; -2 \;\; z'+1]r & (\nu_2) \\ [z \;\; -2 \;\; z']r &=& [z-1 \;\; 2 \;\; z'-1]r & (\nu_{-2}) \end{array} \qquad (10)$$

By repeatedly applying (ν_1) and (ν_{-1}), in order to eliminate all terms $|z| < 2$, we transform a **N** fraction into a **Z** fraction; for example:.

$$[1\;1\;1\;1\;1]r \Rightarrow [2\;\; -2\;\; -1\;\; -1\;\; -1]-r \Rightarrow [2\;\; -3\;2\;1]r \Rightarrow [2\;\; -3\;3]-r.$$

Applying the same rules backwards, we can eliminate all negative terms and transform a **Z** fraction into a **N** fraction:

$$[3\;\; -4\;2\;5\;\; -2]r \Rightarrow [2\;1\;3\;\; -2\;\; -5\;2]-r \Rightarrow [2\;1\;2,1\;1\;5\;\; -2]r \Rightarrow [2\;1\;2,1\;1\;4\;1]-r.$$

We now show that a similar process (almost) converts **E** into **Z** fractions:

Algorithm 3 (Normalization of E-fractions) *Let $r = [z_0 z_1 \cdots z_{n-1}]r(n)$ be a partial E-fraction, of length $n > 3$. We construct, in linear $O(n)$ time, an equivalent fraction*

$$\mathcal{N}(z_0, z_1, z_2, r_3, n-3) = [z'_0 z'_1 \cdots z'_{k-1}]\langle z'_k z'_{k+1} z'_{k+2}\rangle \pm r(n),$$

whose terms $[z'_0 z'_1 \cdots z'_{k-1}]$ form a $\mathbb{Z}$-fraction, except for the last three $\langle z'_k z'_{k+1} z'_{k+2}\rangle$. The computation starts at $t = 0$, and $\mathcal{N}(z_t, z_{t+1}, z_{t+2}, r_{t+3}, n)$ is processed as follows:

1. If $n \leq 0$, the normalization terminates:

$$\mathcal{N}(z_t, z_{t+1}, z_{t+2}, r_{t+3}, n) \Rightarrow \langle z_t z_{t+1} z_{t+2} \rangle r(t+3).$$

2. If $z_{t+1} = 0$, then:

$$\mathcal{N}(z_t, 0, z_{t+2}, r_{t+3}, n) \Rightarrow \mathcal{N}(z_t + z_{t+2}, z_{t+3}, z_{t+4}, r_{t+5}, n-2).$$

3. If $|z_{t+1}| = 1$, then:

$$\begin{array}{rcl} \mathcal{N}(z_t, 1, z_{t+2}, r_{t+3}, n) & \Rightarrow & \mathcal{N}(z_t+1, -z_{t+2}-1, -z_{t+3}, -r_{t+4}, n-1); \\ \mathcal{N}(z_t, -1, z_{t+2}, r_{t+3}, n) & \Rightarrow & \mathcal{N}(z_t-1, -z_{t+2}+1, -z_{t+3}, -r_{t+4}, n-1). \end{array}$$

4. If $|z_{t+1}| = 2$ and $z_{t+2} z_{t+1} < 0$, then:

$$\begin{array}{rcl} \mathcal{N}(z_t, 2, z_{t+2}, r_{t+3}, n) & \Rightarrow & [z_t+1]\mathcal{N}(-2, z_{t+2}+1, z_{t+3}, r_{t+4}, n-1); \\ \mathcal{N}(z_t, -2, z_{t+2}, r_{t+3}, n) & \Rightarrow & [z_t-1]\mathcal{N}(2, z_{t+2}-1, z_{t+3}, r_{t+4}, n-1). \end{array}$$

5. In all remaining cases, $|z_{t+1}| > 2$ or $|z_{t+1}| = 2, z_{t+2} z_{t+1} > 0$:

$$\mathcal{N}(z_t, z_{t+1}, z_{t+2}, r_{t+3}, n) \Rightarrow [z_t]\mathcal{N}(z_{t+1}, z_{t+2}, z_{t+3}, r_{t+4}, n-1).$$

While any finite continued fraction may be normalized by repeatedly applying the rules (10), it is not true that this can be achieved by proceeding from left to right, with bounded memory and linear time, as in Algorithm 3. Indeed,

$$[z_0 \cdots z_k \, 0 \, -z_k \cdots -z_0] = [0]$$

is not an **E**-fraction. Although they can be normalized by Algorithm 3, the following are not **E**-fractions either:

$$\begin{array}{rcccl} [,1 \; -1] & = & [,0\,0] & = & \perp; \\ [,2 \; -2] & = & [1 \; ,2\,0] & = & [1]\infty. \end{array}$$

Normalization of **E** fractions can be performed "off-line", prior to a comparison; it can also be applied "on-line", by systematically simplifying at generation time.

Notation 2 *We call $\mathbf{Z}_\infty$-fraction the result of the normalization of an **E** fraction.*

We write a $\mathbf{Z}_\infty$ fraction in the form $[z_0 z_1 \cdots z_{n-1}]\langle z_n z_{n+1} z_{n+2}\rangle r(n+3)$, where $[z_0 z_1 \cdots z_{n-1}]$ is a regular **Z**-fraction, satisfying (6) and (7). The only missing rule is (8), since we cannot always recognize if $r(n) = \infty$. We can thus produce arbitrarily long segments of the **Z** fraction of r, provided that all continuations remain finite, so the following holds:

- The $\mathbf{Z}_\infty$ fraction of undefined is $\perp = \langle 0\;0\;0 \rangle \perp(t)$, with $\perp(t)$ producing zero terms.
- The $\mathbf{Z}_\infty$ fractions of infinity are $\infty = \langle z_t \; 0 \; z_{t+2}\rangle \infty(t)$, where $|z_t|$, $|z_t + z_{t+2}|$ and $|\infty(t)|$ become arbitrarily large with time t.

- Any irrational number has a *unique* infinite $\mathbf{Z}_\infty$ fraction which is identical to its $\mathbf{Z}$ fraction.
- Let $r = [z_0 \cdots z_{n-1} z_n]\infty$ be the $\mathbf{Z}$ fraction of some rational number $r \in \mathbf{Q}$.
  - If $z_n \neq 2$, all $\mathbf{Z}_\infty$ fractions for r having $n+1$ terms are of the form, $r = [z_0 \cdots z_{n-1} z_n]\langle z_{n+1}\ 0\ z_{n+3}\rangle r(n+3)$, and thus only differ in their terminal representation of ∞.
  - If $z_n = 2$, the $\mathbf{Z}_\infty$ fractions for r having $n+1$ terms are of one of the two forms:

$$\begin{array}{llll} r = [z_0 \cdots & z_{n-1} & 2]\langle z_{n+1}\ 0\ z_{n+3}\rangle r(n+3), \\ r = [z_0 \cdots & z_{n-1}{+}1 & -2]\langle z_{n+1}{+}1\ 0\ z_{n+3}\rangle r(n+3). \end{array}$$

3.5 Comparison between $\mathbf{Z}_\infty$ continued fractions

We now describe how to compare numbers given by their respective $\mathbf{Z}_\infty$-fractions. The procedure is reminiscent of decimal comparison:

Algorithm 4 (Comparison between $\mathbf{Z}_\infty$-fractions) *Let*

$$r(0) = [z_0 \cdots z_{n-1}]r(n) \text{ and } r'(0) = [z'_0 \cdots z'_{n'-1}]r'(n')$$

be two finite numbers, to be compared from their respective $\mathbf{Z}_\infty$-fractions . The comparison starts with $t = 0$, and step t goes as follows:

1. *If $z_t \neq z'_t$, we terminate with $r?r' = z_t?z'_t$.*
2. *If $t > n_\epsilon = -\log_\psi \epsilon$, we call r and r' ϵ-equals, and terminate with $r?r' = \{+, 0, -\}$.*
3. *If both $r(t+1)$ and $r'(t+1)$ are potentially infinite, and $|r(t+1)|, |r'(t+1)| > \frac{1}{\epsilon}$, we terminate as in the previous case: $r?r' = \{+, 0, -\}$.*
4. *If $r(t+1) = [z_{t+1}]r(t+2)$ is finite, $r'(t+1)$ potentially infinite, we terminate with $r?r' = z_{t+1}?0$; symmetrically, if $r'(t+1) = [z'_{t+1}]r'(t+2)$ is finite and $r(t+1)$ potentially infinite, we terminate with $r?r' = 0?z'_{t+1}$.*
5. *If $z_t = z'_t$ and both $r(t+1) = [z_{t+1}]r(t+2)$ and $r'(t+1) = [z'_{t+1}]r'(t+2)$ are finite, we repeat the procedure $r'(t+1)?r(t+1)$ for $t+1$,* after exchanging *the operands r and r'.*

3.6 Representing Numbers as Infinite Products of Homographies

Our stream representation (2) of E-fractions is not well suited to current computer architectures, and storage management techniques. Each term z_n, which is typically small ($z_n = 2, 3$ in about 60% of the cases, see [81]), has to be represented by a computer word, say 32 bits, a link to the next term, again 32 bits, and a *tag*, indicating the type (small or big integer, regular term or continuation). We end up with a large storage to information content ratio. In our first implementation, it was 64 bits per term; if we estimate each term at 1. 5 bits of information, this ratio is about 42. To improve upon this ratio, we code $n+1$ consecutive terms as a 2×2 matrix:

$$h = [z_0 \cdots z_n] = \begin{bmatrix} z_0 & 1 \\ 1 & 0 \end{bmatrix} \cdots \begin{bmatrix} z_n & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} N_n & N_{n-1} \\ D_n & D_{n-1} \end{bmatrix}. \tag{11}$$

The value of n is chosen so that each of the four integer coefficient is an integer which can be represented with 32 bits[4]. We now need 4×32 plus a 32 bits link to represent each homography h. The $n+1$ terms thus encoded amount to 32 bits of the represented number, and the storage to information ratio is now down to 5. We are thus representing a real number by an infinite product of homographies of the form (11), each having determinant ± 1, implemented by a stream:

$$r = \langle h_0 \cdots h_n \rangle r(n).$$

Besides optimizing storage utilization, another advantage of this representation is that the normalization rules (10) are automatically performed when we multiply together the terms of the fractions by (11). For example, the Algebraic Algorithm, to be introduced in the next Section, produces the output:

$$p = [2\ 1\ \ 2\ 0\ \ -1\ \ -6\ 12\ 0\ \ -10\ \ -5]r = \begin{bmatrix} 218 & -49 \\ 89 & -20 \end{bmatrix} r = \frac{218r - 49}{89r - 20}.$$

Although it is not an **E** fraction, p can be normalized by Algorithm 3, to produce:

$$p' = [2\ 2\ 4\ 2\ 4]{-r} = \begin{bmatrix} 218 & 49 \\ 89 & 20 \end{bmatrix} {-r} = \frac{-218r + 49}{-89r + 20}.$$

We see that p and p' are equivalent homographies, to within a change in the sign of the argument.

4 Rational Arithmetic with Continued Fractions

We now describe how to perform arithmetic operations with continued fractions. After discussing the easy cases of $-r$ and $1/r$, we introduce the general *algebraic algorithm* for computing $f(r)$, function f being the quotient of two degree k polynomials in r. We describe an efficient variant of the algebraic algorithm, for which the continued fraction arithmetic is performed in a *strictly positional* manner, producing one term of output for each term of input. Special attention is given to the case of homographies, functions of degree one. We then generalize the algorithm to functions of two variables, which, for degree 1, particularizes to the computation of sums and products of continued fractions.

[4]The rare case of huge terms $|z_n| > 2^{32}$ is handled in a special way.

4.1 Inverse and Opposite of a Continued Fraction

As a direct consequence of (10), we see that, with continued fractions, computing the inverse $1/r$ of number $r = [z_0 z_1 \cdots]$, is performed, in *constant* time, by:

$$\begin{array}{lcll} \frac{1}{r} & = & [0\, z_0 z_1 \cdots] & \text{if } z_0 \neq 0; \\ \frac{1}{r} & = & [z_1 \cdots] & \text{if } z_0 = 0. \end{array}$$

The opposite $-r$ of number $r = [z_0 z_1 z_2 \cdots]$ is given by:

$$-[z_0\ z_1\ z_2 \cdots] = [-z_0\ -z_1\ -z_2 \cdots].$$

We perform such a computation in a *lazy* way, by introducing the continuation:

$$[-] = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = [1\ -1\,1] = [-1\,1\ -1].$$

One step of the evaluation of continuation $[-]$ can be represented by:

$$[-z_0 \cdots -z_{t-1}][-][z_t\, z_{t+1} \cdots] \Rightarrow [-z_0 \cdots -z_{t-1}\ -z_t][-][z_{t+1} \cdots].$$

The strictly positional simplicity of this algorithm is appealing, and we shall strive to use the same paradigm in computing more complex operations.

4.2 Algebraic Algorithm

We now show how to compute the value $f(r)$, where $f \in \mathcal{Q}$ is a rational function, presented as the quotient of two polynomials N and D, with integer coefficients, and degree k:

$$f(r) = \frac{N(r)}{D(r)} = \frac{n_k r^k + n_{k-1} r^{k-1} + \cdots + n_0}{d_k r^k + d_{k-1} r^{k-1} + \cdots + d_0}.$$

Let $\mathcal{Q}$ denote the class of such functions, and note that $f \in \mathcal{Q}$ and $q \in \mathbb{Q}$ implies $f(q) \in \mathbb{Q}$.

Definition 3 • *We say that f is monotone* increasing, *noted $f \nearrow$, in interval I, if, for all $[i,s] \subseteq I$, we have $f[i,s] = [f(i), f(s)]$.*

- *We say that f is monotone* decreasing, *noted $f \searrow$, in interval I, if, for all $[i,s] \subseteq I$, we have $f[i,s] = [f(s), f(i)]$.*

Note that, if f is $\nearrow$ (resp. $\searrow$), then $\lambda x. f(z + \frac{1}{x})$ and $\lambda x. \frac{1}{z + f(x)}$ are $\searrow$ (resp. $\nearrow$).

Algorithm 5 (Algebraic Algorithm) *Let $f \in \mathcal{Q}$ be monotone in $(|x_0| - \frac{1}{2}, \frac{1}{2} - |x_0|)$. Given the $\mathbf{E}$ fraction $x = [x_0 x_1 \cdots]$ of a defined number $x \neq \perp$, we compute that of $y = f(x) = f_0(x) = [y_0 y_1 \cdots]$. At time $t = n + m$, n terms of the output y have been produced, by looking at m terms of the input x, and we are currently computing $y(n) = f_t(x(m))$, written as:*

$$[y_0 \cdots y_{n-1}]\ \boxed{\ f_t\ }\ [x_m x_{m+1} \cdots].$$

We compute the rational end-points[5] $i = f_{n,m}(|x_m| - \frac{1}{2})$ *and* $s = f_{n,m}(\frac{1}{2} - |x_m|)$, *and consider two cases, depending on the distance* $\Delta(i,s)$ *between these two points:*

1. *If* $\Delta(i,s) < 1$, *we produce one term of output* $y_n = [i,s] \div 1$ *by (9):*

$$[y_0 \cdots y_{n-1} y_n] \boxed{f_{t+1}} [x_m x_{m+1} \cdots],$$

and the next function is:

$$f_{t+1} = \lambda x. \frac{1}{f_t(x) - y_n}.$$

2. *If* $\Delta(i,s) \geq 1$, *we consume one term of input:*

$$[y_0 \cdots y_{n-1}] \boxed{f_{t+1}} [x_{m+1} \cdots];$$

the next function is:

$$f_{t+1} = \lambda x. f_t(x_m + \frac{1}{x}).$$

Implementation: In order to avoid computing $\Delta(i,s) = \frac{2|i-s|}{\sqrt{1+i^2}\sqrt{1+s^2}}$, we set $z_i = \lfloor i \rceil, z_s = \lfloor s \rceil$ and $z =$ if $|z_s| < |z_i|$ then z_s else z_i. We can then replace condition $\Delta(i,s) < 1$ by an equivalent, Pythagorean, all integer case analysis based on z_i, z_s and z. Indeed, one can simply use: $|z| > n_{min}$ or $|z_i - z_s| < 2$, with $n_{min} = 3$. Replacing $n_{min} = 3$ by a larger integer threshold $n_{min} > 3$ has the same effect as imposing a more stringent pre-condition $\Delta(i,s) < \Delta(n_{min}, -n_{min}) < 1$ on output, in case 1 of the Algorithm. The gross effect is to bring the computed fraction "closer" to its $\mathbf{Z}_\infty$ form, by forcing the absolute value of terms preceeding a 0 to be at least n_{min}. ∎

At this point, it helps to run the algorithm through some examples.

Example: Computation of the **E** fraction for

$$\sin 2 = \frac{2t}{1+t^2}, \text{ where } t = \tan 1 = [0\ 1\ \ -3\ 5\ \ -7\ \cdots].$$

Function $\lambda t. \frac{2t}{1+t^2}$ is monotone in $(0,1)$, so we absorb two terms before starting the algo-

[5]If x is a $\mathbf{Z}_\infty$ fraction, simply use $i = f(2)$ and $s = f(-2)$.

	e	3	−4	2	5	−2	−7	⋯
$10e$	0	10	30					
25	1	0	1	−4				
0		10	5	−10				
2			1	−4				
1			3	−2	−1			
0			−2	−2	−6			
4				−2	−1	−7		
2				6	−2	−4	6	
3					3	1	1	−6
0						−7	3	−28
5							1	−6
−2							−2	2
⋮								

Figure 6: A practical paper and pencil layout of the computation of $10e$.

rithm. We then compute:

$$\begin{array}{lll}
\sin 2 = \begin{bmatrix} 2 & 2 & 0 \\ 2 & 2 & 1 \end{bmatrix} [-3\;5\;-7\;\cdots] & & [\frac{12}{13}, \frac{4}{5}] \div 1 = 1 \\
\Rightarrow [1] \begin{bmatrix} 2 & 2 & 1 \\ 0 & 0 & -1 \end{bmatrix} [-3\;5\;-7\;\cdots] & & [-13, -5] \div 1 = -5 \\
\Rightarrow [1\;-5] \begin{bmatrix} 0 & 0 & -1 \\ 2 & 2 & -4 \end{bmatrix} [-3\;5\;-7\;\cdots] & & \Delta(\frac{-1}{8}, \infty) > 1 \\
\Rightarrow [1\;-5] \begin{bmatrix} -1 & 0 & 0 \\ 8 & -10 & 2 \end{bmatrix} [5\;-7\;\cdots] & & [\frac{-2}{7}, \frac{-2}{27}] \div 1 = 0 \\
\Rightarrow [1\;-5\;0] \begin{bmatrix} 8 & -10 & 2 \\ -1 & 0 & 0 \end{bmatrix} [5\;-7\;\cdots] & & [\frac{-7}{2}, \frac{-27}{2}] \div 1 = -4 \\
\Rightarrow [1\;-5\;0\;-4] \begin{bmatrix} -1 & 0 & 0 \\ 4 & -10 & 2 \end{bmatrix} [5\;-7\;\cdots] & & \Delta(\frac{-2}{19}, 2) > 1 \\
\Rightarrow [1\;-5\;0\;-4] \begin{bmatrix} -25 & -10 & -1 \\ 52 & 30 & 4 \end{bmatrix} [-7\;\cdots] & & [\frac{-81}{152}, \frac{-121}{272}] \div 1 = 0 \\
\Rightarrow [1\;-5\;0\;-4\;0] \begin{bmatrix} 52 & 30 & 4 \\ -25 & -10 & -1 \end{bmatrix} [-7\;\cdots] & & [\frac{-152}{81}, \frac{-272}{121}] \div 1 = -2 \\
\Rightarrow [1\;-5\;0\;-4\;0\;-2] \begin{bmatrix} -25 & -10 & -1 \\ 2 & 10 & 2 \end{bmatrix} [-7\;\cdots] & & \Delta(\frac{121}{30}, \frac{-81}{10}) > 1 \\
\Rightarrow [1\;-5\;0\;-4\;0\;-2] \begin{bmatrix} -1156 & 340 & -25 \\ 30 & -18 & 2 \end{bmatrix} [\cdots] & & [\frac{-3969}{86}, \frac{-5329}{158}] \div 1 = -34 \\
\Rightarrow [1\;-5\;0\;-4\;0\;-2\;-34] \begin{bmatrix} 30 & -18 & 2 \\ -136 & -272 & 43 \end{bmatrix} [\cdots] & &
\end{array}$$

After normalization, we find that:

$$\sin 2 = [1 \ -11 \ \cdots].$$

Example: Let us compute the **E** fraction of $10e$ from that of e:

$$\begin{array}{rcll}
10e & = & \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix} [3 \ -4\,2\,5 \ -2 \ -7 \ \cdots] & \\
 & \Rightarrow & \begin{bmatrix} 30 & 10 \\ 1 & 0 \end{bmatrix} [-4\,2\,5 \ -2 \ -7 \ \cdots] & [\frac{70}{2}, \frac{50}{2}] \div 1 = 25 \\
 & \Rightarrow & [25] \begin{bmatrix} 1 & 0 \\ 5 & 10 \end{bmatrix} [-4\,2\,5 \ -2 \ -7 \ \cdots] & \Delta(\frac{2}{20}, \infty) > 1 \\
 & \Rightarrow & [25] \begin{bmatrix} -4 & 1 \\ -10 & 5 \end{bmatrix} [2\,5 \ -2 \ -7 \ \cdots] & [\frac{7}{15}, \frac{9}{25}] \div 1 = 0 \\
 & \Rightarrow & [25\ 0] \begin{bmatrix} -10 & 5 \\ -4 & 1 \end{bmatrix} [2\,5 \ -2 \ -7 \ \cdots] & [\frac{15}{7}, \frac{25}{9}] \div 1 = 2 \\
 & \Rightarrow & [25\ 0\ 2] \begin{bmatrix} -4 & 1 \\ -2 & 3 \end{bmatrix} [2\,5 \ -2 \ -7 \ \cdots] & [7, \frac{9}{7}] \div 1 = 1 \\
 & \Rightarrow & \cdots \text{ cf. Figure 6 } \cdots & \\
 & \Rightarrow & [25\,0\,2\,1\,0\,4\,2\,3\,0\,5\ -2] \begin{bmatrix} -6 & 1 \\ 2 & -2 \end{bmatrix} [\cdots] &
\end{array}$$

If we normalize at this point, we get $10e = [27\ 5\ 2\ \cdots]$.

4.3 Positional Algebraic Algorithm

In the Algebraic Algorithm above, consuming one term of x_t of input involves computing

$$f_{t+1} \ = \ \lambda x. f_t(x_t + \frac{1}{x}), \tag{12}$$

while producing one term y_t of output requires to compute:

$$f_{t+1} \ = \ \lambda x. \frac{1}{f_t(x) - y_t}. \tag{13}$$

In addition to evaluating (12) or (13) at each step $t \longrightarrow t+1$, the Algebraic Algorithm computes $\lfloor f_t(z) \rceil$ and $\lfloor f_t(-z) \rceil$, with $z = |x_t| - \frac{1}{2}$, to determine if it should apply (12) or (13) , and choose y_t in the case (13) .

An empirical observation of the Algebraic Algorithm shows that input steps (12) statistically alternate with output steps (13). We also observe that the output term y_t is "close" to $f_t(x_t)$. This suggests the following *positional* variant of the Algebraic Algorithm:

Algorithm 6 (Positional Algebraic Algorithm) *Let $f \in \mathcal{Q}$ be monotone in the interval $(|x_0| - \frac{1}{2}, \frac{1}{2} - |x_0|)$. Given some continued fraction $x = [x_0 x_1 \cdots]$ of number x, we compute*

$$y = f(x) = [y_0 \cdots y_{t-1}] \boxed{\ f_t\ } [x_t x_{t+1} \cdots],$$

for $t = 0, 1, \ldots$ *as follows:*

$$\begin{aligned} f_0 &= f; \\ y_t &= \lfloor f_t(x_t) \rceil; \\ f_{t+1} &= \lambda x . \frac{1}{f_t(x_t + \frac{1}{x}) - y_t}. \end{aligned}$$

Implementation: We perform the computation in three steps:

1. Compute $f'_t = \lambda x . f_t(x_t + \frac{1}{x})$, as in step (12).
2. Evaluate $y_t = \lfloor f'_t(\infty) \rceil$.
3. Compute $f_{t+1} = \lambda x . \frac{1}{f'_t(x) - y_t}$, as in step (13).

■

In the Positional Algebraic Algorithm, we have reduced the computation time of each term in the output continued fraction, at the price of crippling our automatic error analysis. Indeed, we can no longer guarantee that the output fraction can be normalized "on-line", as in Algorithm 3. We can combine the speed advantage of Algorithm 6, with the precision of Algorithm 5, by running the first one for k steps, before computing one step of the latter one; we are thus trading granularity in our automatic error analysis, for speed in the computation.

4.4 Homographic Algorithm

	e	3	-4	2	5	-2	-7	$\cdots$
$10e$	0	10	30					
30	1	0	1	-4				
0		10	0	10	20			
-3			1	-4	-7	-39		
5				-2	-1	-7	13	
2					-2	-4	6	-46
8						1	1	-6
$\vdots$							-2	2

Figure 7: The positional computation of $10e$.

We consider in some detail the computation of a homographic transform $y = h(x) = \frac{Nx+n}{Dx+d}$ with integer coefficient, special case of the Algebraic Algorithm with degree 1. This algorithm allows one to compute sums and products between a rational number $q = \frac{n}{d}$, given by numerator and denominator, and a number $x = [x_0 x_1 \cdots]$ given by its continued fraction:

$$\frac{n}{d} + x \Rightarrow \begin{bmatrix} d & n \\ 0 & d \end{bmatrix} [x_0 x_1 \cdots];$$
$$\frac{n}{d} \times x \Rightarrow \begin{bmatrix} n & 0 \\ 0 & d \end{bmatrix} [x_0 x_1 \cdots].$$

Algorithm 7 (Homographic Algorithm) *Computes the terms $[y_0 y_1 \cdots]$ of the homographic transform*

$$y = h(x) = \frac{Nx+n}{Dx+d} = h_0(x),$$

from the terms $[x_0 x_1 \cdots]$ of x, in successive steps $t = 0, 1, \ldots$:

$$[y_0 \cdots y_{t-1}][h_t][x_t x_{t+1} \cdots] \Rightarrow [y_0 \cdots y_{t-1} y_t][h_{t+1}][x_{t+1} \cdots]$$

where y_t and h_{t+1} are computed by:

$$\begin{aligned} h'_t &= \lambda x. h_t(x_t + \frac{1}{x}); \\ y_t &= \lfloor h'_t(\infty) \rceil; \\ h_{t+1} &= \lambda x. \frac{1}{h'_t(x) - y_t}. \end{aligned}$$

Implementation: Using matrix notation, the three steps of the Positional Homographic Algorithm are:

1. Two additions and multiplications to consume one term:
$$[h'_t] = \begin{bmatrix} N'_t & n'_t \\ D'_t & d'_t \end{bmatrix} = \begin{bmatrix} N_t & n_t \\ D_t & d_t \end{bmatrix} \begin{bmatrix} x_t & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} N_t x_t + n_t & N_t \\ D_t x_t + d_t & D_t \end{bmatrix}.$$

2. One division to produce the output term:
$$y_t = \lfloor N'_t / D'_t \rceil.$$

3. Two substractions and multiplications to update h:
$$[h_{t+1}] = \begin{bmatrix} N_{t+1} & n_{t+1} \\ D_{t+1} & d_{t+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -y_t \end{bmatrix} \begin{bmatrix} N'_t & n'_t \\ D'_t & d'_t \end{bmatrix} = \begin{bmatrix} D'_t & d'_t \\ N'_t - D'_t y_t & n'_t - d'_t y_t \end{bmatrix}.$$

■

A computation of $10e$ by this Algorithm can be found in Figure 7, and it should be compared with that of Figure 6. Both results, namely $[30\ 0\ -3\ 5\ 2\ \cdots]$ and $[25\ 0\ 2\ 1\ 0\ 4\ 2\ \cdots]$, are equivalent, after normalization, to $[27\ 5\ 2\ \cdots]$. Figure 6 however does not reflect the true cost of the general Algebraic Algorithm, since it omits the computation of $[h(2), h(-2)]$, carried out at each step, in order to decide if the next operation will absorb or emit a term. All accounted for, the Positional Homographic Algorithm requires 63 arithmetic operations to perform the computation of Figure 7, and the general Algebraic Algorithm 238 for Figure 6.

4.5 Quadratic Algorithm

$$
\begin{aligned}
&\sqrt{3}+\sqrt{3} = [2\ ,-4\ 4] + [2\ ,-4\ 4] \\
&\Rightarrow [4] \left\| \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{matrix} \right\| [-4\ -4\ 4\ 4\ -4\ -4\ 4\ 4\ -4\ -4\ \cdots] \\
&\Rightarrow [4\ -2] \left\| \begin{matrix} -8 & 1 & 1 & 0 \\ 0 & -2 & -2 & 1 \end{matrix} \right\| [-4\ 4\ 4\ -4\ -4\ 4\ 4\ -4\ -4\ \cdots] \\
&\Rightarrow [4\ -2\ 15] \left\| \begin{matrix} -2 & -7 & 0 & -2 \\ -1 & 109 & -8 & 31 \end{matrix} \right\| [4\ 4\ -4\ -4\ 4\ 4\ -4\ -4\ \cdots] \\
&\Rightarrow [4\ -2\ 15\ 0] \left\| \begin{matrix} 105 & -1 & -1 & -8 \\ -15 & -2 & -2 & 0 \end{matrix} \right\| [4\ -4\ -4\ 4\ 4\ -4\ -4\ \cdots] \\
&\Rightarrow [4\ -2\ 15\ 0\ -7] \left\| \begin{matrix} 58 & 8 & -15 & -2 \\ -15 & 52 & 02 & -15 \end{matrix} \right\| [-4\ -4\ 4\ 4\ -4\ -4\ \cdots] \\
&\Rightarrow [4\ -2\ 15\ 0\ -7\ -2] \left\| \begin{matrix} 112 & -15 & -15 & 0 \\ 0 & 28 & 28 & -15 \end{matrix} \right\| [-4\ 4\ 4\ -4\ -4\ \cdots] \\
&\Rightarrow [4\ -2\ 15\ 0\ -7\ -2\ 15] \left\| \begin{matrix} 28 & 97 & 0 & 28 \\ 13 & -1523 & 112 & -435 \end{matrix} \right\| [4\ 4\ -4\ -4\ \cdots] \\
&\Rightarrow [4\ -2\ 15\ 0\ -7\ -2\ 15\ 0] \left\| \begin{matrix} -1471 & 13 & 13 & 112 \\ 209 & 28 & 28 & 0 \end{matrix} \right\| [4\ -4\ -4\ \cdots] \\
&\Rightarrow [4\ -2\ 15\ 0\ -7\ -2\ 15\ 0\ -7] \left\| \begin{matrix} -808 & -112 & 209 & 28 \\ 241 & -724 & -8 & 209 \end{matrix} \right\| [-4\ -4\ \cdots] \\
&\Rightarrow [4\ -2\ 15\ 0\ -7\ -2\ 15\ 0\ -7\ -2] \left\| \begin{matrix} -1688 & 241 & 241 & -8 \\ -256 & -326 & -326 & 193 \end{matrix} \right\| [-4\ \cdots]
\end{aligned}
$$

Figure 8: Computation of the sum of two continued fractions.

The Algebraic Algorithm can be generalized to the computation of functions in many variables, expressed as quotients of two integer coefficient polynomials in these variables. Rather than describe the general case, we content ourselves with the special case of two variables and degree one, hence
computing functions hh of the form:

$$y = hh(x,x') = \frac{(Nx+N')x' + (nx+n')}{(Dx+D')x' + (dx+d')}.$$

If we fix one argument, say $x' = r$, such a function $hh = \lambda x.hh(x,r)$ is a homographic transform with respect to its other argument.

Algorithm 8 (Quadratic Algorithm) *Let us compute the continued fraction* $[y_0y_1\ldots]$ *for*

$$y = hh(x,x') = hh_0(x,x') = \frac{(Nx+N')x' + (nx+n')}{(Dx+D')x' + (dx+d')},$$

in terms of the continued fractions $x = [x_0 x_1 \ldots]$ *and* $x' = [x'_0 x'_1 \ldots]$, *by successive steps:*

$$[y_0 \cdots y_{2t-1}] \boxed{hh_{2t}} \begin{matrix}[x_t x_{t+1} \cdots] \\ [x'_t x'_{t+1} \cdots]\end{matrix} \Rightarrow [y_0 \cdots y_{2t-1} y_{2t} y_{2t+1}] \boxed{hh_{2t+2}} \begin{matrix}[x_{t+1} \cdots] \\ [x'_{t+1} \cdots]\end{matrix}$$

Here, y_{2t}, y_{2t+1} *and* hh_{2t+2} *are computed by:*

$$\begin{aligned} y_{2t} &= \lfloor hh_{2t}(x_t, \infty) \rceil; \\ hh_{2t+1}(x, x') &= \lambda x, x'. \frac{1}{hh_{2t}(x_t + \frac{1}{x}, x') - y_{2t}}; \\ y_{2t+1} &= \lfloor hh_{2t+1}(\infty, x'_t) \rceil; \\ hh_{2t+2}(x, x') &= \lambda x, x'. \frac{1}{hh_{2t+1}(x, x'_t + \frac{1}{x'}) - y_{2t+1}}. \end{aligned}$$

Implementation: To obtain a uniform treatment of both inputs $x = [x_0 x_1 \cdots]$ and $x' = [x'_0 x'_1 \ldots]$, we simply *merge* the two continued fraction into $X = [x_0 x'_0 x_1 x'_1 \cdots]$. We use matrix notation to represent functions hh_t

$$\|hh_t\| = \left\| \begin{matrix} N_t & N'_t & n_t & n'_t \\ D_t & D'_t & d_t & d'_t \end{matrix} \right\|.$$

The positional transition $t \to t+1$:

$$\|hh_t\| \, [x_t] \Rightarrow [y_t] \, \|hh_{t+1}\|,$$

is computed by the formula:

$$\|hh_{t+1}\| = \left\| \begin{matrix} D_t x_t + D'_t & D_t & d_t x_t + d'_t & d_t \\ N_t x_t + N'_t - y_t(D_t x_t + D'_t) & N_t - y_t D_t & n_t x_t + n'_t - y_t(d_t x_t + d'_t) & n_t - y_t d_t \end{matrix} \right\|$$

where term y_t is obtained by division:

$$y_t = \lfloor \frac{N_t x_t + N'_t}{D_t x_t + D'_t} \rceil.$$

Altogether, this computation requires 1 division, 10 multiplications, and 10 additions or substractions. ∎

As a special case, the Quadratic Algorithm allows us to compute sums and products of continued fractions:

$$\begin{aligned} [x_0 x_1 x_2 \cdots] + [x'_0 x'_1 x'_2 \cdots] &= [x_0 + x'_0] \left\| \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{matrix} \right\| [x_1 x'_1 x_2 x'_2 \cdots]; \\ [x_0 x_1 x_2 \cdots] \times [x'_0 x'_1 x'_2 \cdots] &= [x_0 \times x'_0] \left\| \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & x'_0 & x_0 & 1 \end{matrix} \right\| [x_1 x'_1 x_2 x'_2 \cdots]. \end{aligned}$$

The computation of the sum $\sqrt{3} + \sqrt{3} = [3 \; ,2 \; 6]$ is shown in Figure 8. The computed result $[4 \; -2 \; 15 \; 0 \; -7 \; -2 \; 15 \; 0 \; -7 \; -2 \; \cdots]$ is typical of the manner in which the Positional Algebraic Algorithm retracts a premature value; after normalization, the final sum becomes $[3 \; 2 \; 6 \; 2 \; 6 \; 2 \; \cdots]$. Similarly, the product $\sqrt{2} \times \sqrt{3} = \sqrt{6} = [2 \; ,2 \; 4]$ is computed in Figure 9; the result $[2 \; 1 \; 2 \; 0 \; -1 \; -6 \; 12 \; 0 \; -10 \; \cdots]$ normalizes to $[2 \; 2 \; 4 \; 2 \; \cdots]$.

$$\begin{aligned}
&\sqrt{2} \times \sqrt{3} = [1\ ,2] \times [2\ ,-4\ 4] \\
&\Rightarrow [2] \left\| \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 1 \end{array} \right\| [2\ -4\ 2\ 4\ 2\ -4\ 2\ 4\ 2\ -4\ 2\ 4\ \cdots] \\
&\Rightarrow [2\ 1] \left\| \begin{array}{cccc} 2 & 0 & 1 & 0 \\ 2 & 3 & 0 & 1 \end{array} \right\| [-4\ 2\ 4\ 2\ -4\ 2\ 4\ 2\ -4\ 2\ 4\ \cdots] \\
&\Rightarrow [2\ 1\ 2] \left\| \begin{array}{cccc} -3 & -5 & 0 & 1 \\ 1 & 11 & 2 & -2 \end{array} \right\| [2\ 4\ 2\ -4\ 2\ 4\ 2\ -4\ 2\ 4\ \cdots] \\
&\Rightarrow [2\ 1\ 2\ 0] \left\| \begin{array}{cccc} 13 & 2 & 1 & 2 \\ -11 & 1 & -3 & 0 \end{array} \right\| [4\ 2\ -4\ 2\ 4\ 2\ -4\ 2\ 4\ \cdots] \\
&\Rightarrow [2\ 1\ 2\ 0\ -1] \left\| \begin{array}{cccc} -43 & -12 & -11 & -3 \\ 11 & -6 & 2 & -2 \end{array} \right\| [2\ -4\ 2\ 4\ 2\ -4\ 2\ 4\ \cdots] \\
&\Rightarrow [2\ 1\ 2\ 0\ -1\ -6] \left\| \begin{array}{cccc} 16 & 2 & 11 & 2 \\ -2 & -13 & 23 & 1 \end{array} \right\| [-4\ 2\ 4\ 2\ -4\ 2\ 4\ \cdots] \\
&\Rightarrow [2\ 1\ 2\ 0\ -1\ -6\ 12] \left\| \begin{array}{cccc} -5 & -91 & -2 & 23 \\ -2 & -1050 & 40 & -265 \end{array} \right\| [2\ 4\ 2\ -4\ 2\ 4\ \cdots] \\
&\Rightarrow [2\ 1\ 2\ 0\ -1\ -6\ 12\ 0] \left\| \begin{array}{cccc} 1046 & -185 & -2 & 40 \\ -101 & 19 & -5 & -2 \end{array} \right\| [4\ 2\ -4\ 2\ 4\ \cdots] \\
&\Rightarrow [2\ 1\ 2\ 0\ -1\ -6\ 12\ 0\ -10] \left\| \begin{array}{cccc} -385 & -22 & -101 & -5 \\ 149 & -188 & 36 & -48 \end{array} \right\| [2\ -4\ 2\ 4\ \cdots]
\end{aligned}$$

Figure 9: Computation of the product of two continued fractions.

4.6 Square Roots of Rational Numbers

We now apply the Positional Homographic Algorithm to computing square roots of rational numbers, as fixed points of homographies with integer coefficients. By Theorem 4, we know that the continued fraction representation of such numbers is *periodic.*

Algorithm 9 (Square Root of a Rational.) *To compute the positive square root* $\sqrt{\frac{n}{d}}$ *of some finite rational* $\frac{n}{d} \geq 1$*:*

1. *Find the integer square root* $\sigma = \lfloor\sqrt{nd}\rfloor$*, and let:*

$$z_0 = \lfloor\frac{\sigma}{d}\rfloor, \delta = \sigma - z_0 d.$$

   Set $\sqrt{\frac{n}{d}} = [z_0],[h_1],,$ *with the initial value:*

$$h_1 = \begin{bmatrix} 2dz_0 & d \\ n - dz_0^2 & 0 \end{bmatrix}.$$

2. *For* $t = 1, 2, \ldots,$ *compute:*

$$h_t = \begin{bmatrix} N_t & n_t \\ D_t & d_t \end{bmatrix};$$

$$\begin{aligned} z_t &= \lfloor \frac{N_t+\delta}{D_t} \rfloor; \\ h_{t+1} &= \lambda x . \frac{1}{h_t(z_t+\frac{1}{x})-z_t} = \begin{bmatrix} 0 & 1 \\ 1 & -z_t \end{bmatrix} [h_t] \begin{bmatrix} z_t & 1 \\ 1 & 0 \end{bmatrix}. \end{aligned}$$

3. Stop when either:

- $z_t = \infty$, *in which case* $\sqrt{\frac{n}{d}} = [z_0 \cdots z_{t-1}]\infty$ *is a rational number.*
- $h_{t+1} = h_1$, *in which case* $\sqrt{\frac{n}{d}} = [z_0 \ , z_1 \cdots z_{t-1}]$ *is periodic.*

After the initial step $t = 0$, Algorithm 9 above behaves almost exactly like the Positional Homographic Algorithm, in which the last term produced is the next one to be consumed. They only differ in the computation of $z_t = \lfloor \frac{N_t+\delta}{D_t} \rceil$ rather than $z'_t = \lfloor \frac{N_t}{D_t} \rceil = \lfloor h_t(\infty) \rceil$, and the cases in which $z_t \neq z'_t$ are quite rare, since $|\delta| < d/2$; an example is found in term -21, Figure 10. In our implementation, periodic continued fractions are represented by graph

	$\sqrt{6} =$	[2	,2	4]	
$\sqrt{6}$	6	2			
2	2	1	4		
2	2	0	2	4	
4		1	0	1	4
			2	0	2

	$\sqrt{\frac{3}{2}} =$	[1	,4	2]	
$\sqrt{\frac{3}{2}}$	3	2			
1	2	2	4		
4	1	0	1	4	
2		2	0	2	4
			1	0	1

	$\sqrt{\frac{37}{3}} =$	[4	,−2	−21	2	7]	
$\sqrt{\frac{37}{3}}$	37	12					
4	12	3	24				
−2	−11	0	−11	22			
−21		3	2	−1	23		
2			31	1	10	21	
7				−3	3	3	24
					−11	0	−11

Figure 10: Three Square Root computations by Algorithm 9.

structures with a loop, so they contain no explicit continuation. The representation of our two favorite Pythagorean numbers is drawn in Figure 11.

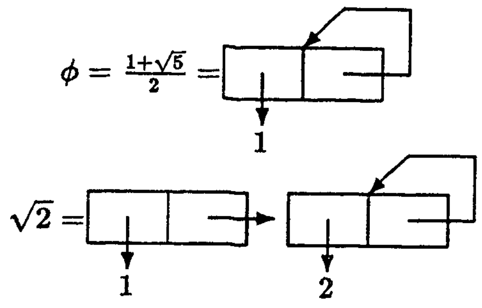

Figure 11: Internal representation of the golden ratio and the diagonal of the unit square.

5 Transcendental Continued Fraction Arithmetic

Using the Homographic Algorithm, we can transform a continued fraction

$$[q_0 q_1 \cdots]$$

whose terms $q_n \in \mathbb{Q}$ are *rational* numbers, into a fraction with all-integer terms. In order to be able to normalize such a fraction by Algorithm 3, it is sufficient that, for n large enough, terms q_n have absolute value greater than two: $|q_n| > 2$.

Similarly, the Quadratic Algorithm gives a way to compute the $\mathbf{Z}_\infty$ fraction of a number given by a continued fraction

$$[r_0 r_1 \cdots]$$

whose terms r_n are finite real numbers, provided their absolute value is eventually greater than two: $|r_n| > 2$. As a consequence, we can compute a transcendental function $f(r)$ from a continued fraction expansion $f(r) = [t(0)t(1)\cdots]$ in which all the terms $t(n)$ are rational functions of the argument r. Many such formulae can be derived from Gauss's continued fraction (see [48]). We simply review some special case of practical interest.

5.1 Exponentials and Logarithms

The remarkable formula of Legendre

$$e^r = [,1 \ \frac{2n+1}{r} -1 \ 1] = [1 \ ,\frac{4n+1}{r} \ 2 \ -\frac{4n+3}{r} \ -2] \tag{14}$$

allows us to compute the continued fractions for e^r, from that of r. Note that (14) converges since all the terms are greater than 2 for $n \geq \frac{|r|}{2}$. We compute logarithms in the same way with the help of:

$$\log(1+r) = [0 \ ,\frac{2n+1}{r} \ \frac{2}{n+1}]. \tag{15}$$

Fraction (15) converges since the $2n+1$-th continuation has absolute value greater than 2 as soon as $n > |r|$.

By combining (14) and (15), we compute arbitrary powers and roots of numbers. One can also use the direct formula:

$$(1+r)^k = \begin{bmatrix} 1 & kr \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 2n+1 & (n+1)(n+1+k)r \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2n+3 & (n+1)(n+1-k)r \\ 1 & 0 \end{bmatrix}. \quad (16)$$

5.2 Trigonometric Functions

From (14), we derive

$$\frac{e^r + e^{-r}}{e^r - e^{-r}} = [, \frac{2n+1}{r}],$$

a formula of Lambert, from which we get:

$$\tan(r) = [0 , \frac{4n+1}{r} \ -\frac{4n+3}{r}]. \quad (17)$$

From (17) and the usual

$$\begin{aligned} t &= \tan\frac{r}{2}; \\ \sin(r) &= \frac{2t}{1+t^2}; \\ \cos(r) &= \frac{1-t^2}{1+t^2}; \end{aligned}$$

we compute sin(r) and cos(r), using the Algebraic Algorithm. Finally, the inverse trigonometric functions can be computed from:

$$\arctan r = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 2n+1 & (n+1)^2 \\ r & 0 \end{bmatrix}. \quad (18)$$

As a special case, we have the following expression of *pi*:

$$\pi = \begin{bmatrix} 0 & 4 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 3 & 4 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 5 & 9 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 2n+7 & (n+4)^2 \\ 1 & 0 \end{bmatrix} \quad (19)$$

6 Conclusion

- This work, as well as the one reported in [86], shows that it is feasible to incorporate a usable exact real arithmetic package in any of today's programming language. Worthwhile applications are expected to be drawn from physics, numerical analysis, and mathematics. In each of these areas, the current interest in *chaotic, fractal, catastrophic* behaviours provides ample justification for an *exact* computational tool, providing an automatic, arbitrary precision, error analysis.

- In view of its capital importance, the choice of number representation deserves a better treatment than our discussion in Section 3.1. Representing numbers by infinite products of integer coefficients homographies, as in Section 3.6, is a general technique. There are many ways to define a *normal form* for such infinite products:

  1. Continued fractions lead to
  $$r = \begin{bmatrix} z_0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} z_1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} z_2 & 1 \\ 1 & 0 \end{bmatrix} \cdots,$$
  the integer values z_{n+1} being characterized by Theorem 5.
  2. Radix β representation is expressed by
  $$r = \begin{bmatrix} 1 & z_0 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} 1 & z_1 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} 1 & z_2 \\ 0 & \beta \end{bmatrix} \cdots,$$
  where $0 \leq z_{n+1} < \beta$ for $\mathbf{N}$ radix, and $-\frac{\beta}{2} < z_{n+1} \leq \frac{\beta}{2}$ for $\mathbf{Z}$ radix.
  3. In the following, lesser known system, we use
  $$r = \begin{bmatrix} 1 & 1 \\ 0 & z_0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & z_1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & z_2 \end{bmatrix} \cdots,$$
  with $1 \leq |z_0| \leq \cdots \leq |z_n| \leq |z_{n+1}| \leq \cdots \leq \infty$. Successive continuations $r(n)$ are related by
  $$r(n) = \frac{1}{z_n}(1 + r(n+1)) \text{ with } z_n = \lfloor \frac{1}{r(n)} \rceil.$$

  It is difficult to make a rational choice between such promising candidates. We miss a general result relating the size (total number of bits) of each such representation, to the amount of information obtained about the represented number.

- In the continued fraction representation, what is the *computational complexity* of the Algebraic Algorithm? D. Greene and L. Monier have shown[6] that, in the homographic case of degree one and one variable, the coefficients of f_t in Algorithm 5, Case 2 are all bounded by the determinant of f_0, independently of $t > 0$. This directly implies that the Homographic Algorithm runs in *average linear time*. Analysing the behaviour of the Quadratic Algorithm is an *open question*.

- As suggested by F. Laborde, it appears worthwhile to generalize our construction of the real numbers $\mathbf{R}$, to the algebraically closed field (see [54]) of computable complex numbers $\mathbf{C}$.

Acknowledgments: My debt to W. Gosper, who first showed how to add and multiply continued fractions, is clear. I express my gratitude to:

[6]Private communication.

- my father, for his help in reading the work of Hurwitz. Chenetier
- J.M. Hullot, M. Mendés France, L. Monier, J. Morgenstern and G. Viennot for stimulating discussions.
- J. Chailloux, F. Dupond, M. Devin and B. Serpette for their cooperation in bringing up the generic arithmetic in Le_Lisp [82].

7 Proofs of the theorems

7.1 Proof of Theorem 1

Statement of Theorem 1 *Let $\mathcal{R}$ denote the subset of $\mathcal{L}$-expressions which represent numbers, that is the computable reals expressed as programs in $\mathcal{L}$.*

(i) The set $\mathcal{R}$ is denumerable.

(ii) No algorithm can decide, in finite time, if an arbitrary $\mathcal{L}$-expression e denotes a number, i.e. $e \in \mathcal{R}$.

Proof: Since there are denumerably many $\mathcal{L}$-expressions, set $\mathcal{R} \subset \mathcal{L}$ is also denumerable. Cantor's famous diagonal argument shows that the classical reals *are not* denumerable; it follows that "most" reals are not computable, although no one will ever be able to show me a single specific non-computable number!

Cantor's argument, applied to the computable instead of the classical reals, goes as follows:

1. Suppose that someone claims to have written a program deciding whether or not $e \in \mathcal{R}$, for arbitrary $e \in \mathcal{L}$. This is equivalent to having a program $E \in \mathcal{L}$ which recursively enumerates all expressions in $\mathcal{R} = \{E(0), E(1), \ldots, E(n), \ldots\}$.

2. Define the number $P = \sum_{n \geq 0} d_n/10^n$, whose digits d_n, for each $n \geq 0$, are computed as follows:

   - Compute a rational approximation q_n to $E(n)$ such that
   $$|E(n) - q_n| < 1/10^{n+1}.$$
   - Compute the n-th decimal digit d'_n of q_n, after the fractional point; let $d_n = d'_n - 2$ if $d'_n > 3$, $d_n = d'_n + 2$ otherwise.

3. - The definition above being clearly constructive, we can write a program in $\mathcal{L}$ which compute number P, hence $P \in \mathcal{R}$.
   - For any $k \geq 0$, number P is different from $E(k)$, since
   $$|10^k(P - E(k))| \geq 2 - \frac{1}{10}(9 + 9/10 + 9/100 + \cdots) = 1.$$

   As we assumed that E enumerates all numbers in $\mathcal{R}$, it follows that P /in$\mathcal{R}$, a contradiction. ∎

7.2 Proof of Theorem 2

Statement of Theorem 2 *Let $r \in \mathbf{R}$ represent an arbitrary computable real number. No algorithm can compute, in finite time:*

1. if number r is null, i.e. $r = 0$;

2. if $r > r'$, for any *fixed $r' \in \mathbf{R}$;*

3. if number r is rational, i.e. $r \in \mathbf{Q}$;

4. the first digit of the decimal expansion of r;

5. the first term $\lfloor r \rfloor \in \mathbf{Z}$ of the regular continued fraction expansion of r.

6. the value $f(r) \in \mathbf{R}$ of any function f which is not continuous *at r.*

Proof: Rice's Theorem [53] shows that no function f from $\mathbf{R}$ to a discrete set $\{T, F\}$ is computable, unless it is *constant.* The six statements of Theorem 2 follow directly, from appropriate choices of f:

1. $f(r) = T \iff r = 0$;
2. $f(r) = T \iff r > r'$;
3. $f(r) = T \iff r \in \mathbf{Q}$;
4. $f(r) = T \iff -1 < r < 1$ is equivalent to having 0 for first digit;
5. $f(r) = T \iff 0 < r < 1$ is equivalent to having 0 for first term;
6. $f(r) = T \iff \lim_{\epsilon \to 0} f(r - \epsilon^2) = \lim_{\epsilon \to 0} f(r + \epsilon^2)$.

A more direct reduction of number comparison to the halting problem, shown undecidable by Turing [36], goes as follows: to any type of machine, attach a *Zeno counter*, containing a number ς, whose initial value is $\varsigma_0 = 1$; whenever the machine changes state, number ς is updated as $\varsigma_{t+1} = 1 + \frac{1}{2}\varsigma_t$, so that, after t computational steps, we have

$$\varsigma_t = 1 + \frac{1}{2} + \ldots + \frac{1}{2^t}.$$

By this process, we associate a computable number ς to each computation C, with the property that C terminates iff $\varsigma < 2$, and C is infinite iff $\varsigma = 2$. ∎

7.3 Proof of Theorem 3

Statement of Theorem 3 *For arbitrary $x, y, z \in \mathbf{R}$:*

$$\begin{array}{rcllr} x+y &=& y+x; & & (+)\\ x+(y+z) &=& (x+y)+z; & & (++)\\ (-x)+x &=& 0 & \textit{for } \text{finite } x; & (-)\\ x\times y &=& y\times x; & & (\times)\\ x\times(y\times z) &=& (x\times y)\times z; & & (\times\times)\\ x\times(1/x) &=& 1 & \textit{for } x\neq 0 \text{ finite;} & (/)\\ x\times(y+z) &=& (x\times y)+(x\times z) & \textit{for } x \text{ not infinite.} & (\times+) \end{array}$$

Proof: Represent rational numbers $\mathbf{Q}$ by vectors with integer coefficients, and define:

$$\perp = \begin{bmatrix} 0 \\ 0 \end{bmatrix}; \qquad \infty = \begin{bmatrix} 1 \\ 0 \end{bmatrix};$$

$$\begin{bmatrix} n \\ d \end{bmatrix} \sim \begin{bmatrix} n' \\ d' \end{bmatrix} \iff nd' = dn';$$

$$\begin{bmatrix} n \\ d \end{bmatrix} + \begin{bmatrix} n' \\ d' \end{bmatrix} = \begin{bmatrix} nd'+dn' \\ dd' \end{bmatrix}; \qquad \begin{bmatrix} n \\ d \end{bmatrix} \times \begin{bmatrix} n' \\ d' \end{bmatrix} = \begin{bmatrix} nn' \\ dd' \end{bmatrix};$$

$$-\begin{bmatrix} n \\ d \end{bmatrix} = \begin{bmatrix} -n \\ d \end{bmatrix}; \qquad 1/\begin{bmatrix} n \\ d \end{bmatrix} = \begin{bmatrix} d \\ n \end{bmatrix}.$$

- Relation $\sim$ is symmetric, reflexive, and it identifies every vector $x \sim \perp$ with the undefined vector. Transitivity $x \sim y \,\&\, y \sim z \Rightarrow x \sim z$ holds provided that $y \neq \perp$, so $\sim$ induces an equivalence relation amongst the non undefined vectors, including ∞. Set $\mathbf{Q}$ is thus made of $\perp$, plus the equivalence classes of non undefined vectors.
- We routinely check that, for $x, y, z \in \mathbf{Q}$, the seven projective field laws hold for the arithmetic operations defined above.
- The statement for $x, y, z \in \mathbf{R}$ follows by continuity. By Definition 1, let:

$$x = \bigcap_{n\geq 0}[x_i(n), x_s(n)], y = \bigcap_{n\geq 0}[y_i(n), y_s(n)], z = \bigcap_{n\geq 0}[z_i(n), z_s(n)].$$

Using now the definition of interval addition, we see that $y + x \in [x_i(n), x_s(n)] + [y_i(n), y_s(n)]$ for all $n \geq 0$. It follows that $x + y = y + x$, hence $(+)$. The proofs of the other six relations are similar.

■

7.4 Proof of Theorem 4

Statement of Theorem 4 *Let $r_0 = [z_0\ z_1\ \cdots\ z_{n+1}\ \cdots]$ be the* N *or* Z *continued fraction expansion of some finite number $r = r_0$.*

(i) *The continued fraction is* finite $r_0 = [z_0\ z_1\ \cdots\ z_{n+1}]\infty$ *if and only if number $r \in \mathbf{Q}$ is rational. It has exactly one term $r_0 = [z_0]\infty$ if and only if number $r \in \mathbf{Z}$ is integer.*

(ii) *The continued fraction is eventually* periodic $r_0 = [z_0\ z_1\ \cdots\ z_{n-1}\ ,z_n \cdots z_{n+p}]$ *if and only if number $r \in \sqrt{\mathbf{Q}_{>0}}$ is the irrational square root of a positive rational.*

(iii) *The continued fraction approximants $q_n = [z_0\ z_1\ \cdots\ z_n]\infty$ converge to r with speed: $|r - q_n| < \phi^{-n}$, where $\phi = [,1] = 1.618\cdots$ is the golden ratio.*

Proof: This is a classical result: see [88] or [81]. ∎

7.5 Proof of Theorem 5

Statement of Theorem 5 *Let $r_0 = [z_0\ z_1\ \cdots\ z_{n+1}\ \cdots]$ be the continued fraction expansion of some finite number $r = r_0$.*

(i) N*-fractions are characterized as follows:*

– *All terms $z_1, \ldots, z_{n+1}, \ldots$, past the first one z_0, are positive integers:*

$$n \geq 0 \quad \textit{implies} \quad z_{n+1} \geq 1.$$

– *If the fraction is finite, its last term is greater than 1:*

$$n \geq 0, r_{n+2} = \infty \quad \textit{implies} \quad z_{n+1} \geq 2.$$

(ii) Z*-fractions are characterized as follows:*

– *All terms $z_1, \ldots, z_{n+1}, \ldots$, past the first one z_0, are integers in $[2, -2]$:*

$$n \geq 0 \quad \textit{implies} \quad |z_{n+1}| \geq 2.$$

Furthermore, when $|z_{n+1}| = 2$, the subsequent term z_{n+2} has the same sign as z_{n+1}:

$$n \geq 0, z_{n+1} = 2 \quad \textit{implies} \quad z_{n+2} \geq 2,$$
$$n \geq 0, z_{n+1} = -2 \quad \textit{implies} \quad z_{n+2} \leq -2.$$

– *If the fraction is finite, its last term is different from -2:*

$$n \geq 0, r_{n+2} = \infty \quad \textit{implies} \quad z_{n+1} \neq -2.$$

Proof: Another classical result: see [89]. ∎

7.6 Proof of Theorem 6

Statement of Theorem 6 *Let $r_0 = [z_0\ z_1\ \cdots\ z_{n+1}\ \cdots]$ be some* **E** *continued fraction expansion of number $r = r_0$; let $r_n = [z_n\ z_{n+1}\ \cdots]$ represent the n-th continuation of r.*

(i) All terms are identical to zero if and only if $r = \perp$ is undefined. Assuming from now on that $r \neq \perp$ is defined, *we can effectively eliminate all pairs of consecutive 0 at the begining of r, so that one of the first two terms z_0, z_1 is non zero.*

(ii) The n-th continuation r_n belongs to the interval $\mathcal{V}(r_n) = (|z_n| - \frac{1}{2}, \frac{1}{2} - |z_n|)$, for $z_n \neq 0$, and $\mathcal{V}(r_n) = (-2, 2)$ for $z_n = 0$. The rational approximants $[z_0\ z_1\ \cdots\ z_{n-1}]\mathcal{V}(r_n)$ converge to r:

$$\mathcal{V}(r_0) \supset \cdots \supset [z_0\ z_1\ \cdots\ z_{n-1}]\mathcal{V}(r_n) \supset \cdots \supset r = [z_0\ z_1\ \cdots].$$

(iii) Adjacent terms z_n and z_{n+1} are never simultaneously null: $|z_{n+1}| + |z_{n+2}| \geq 3$. If successive odd terms are null, $z_{n+2t+1} = 0$ for $0 \leq t \leq k$, the partial sums of such terms have increasing absolute values:

$$|\sum_{i \leq 2t+1} z_{i+n}| > |\sum_{i \leq 2t-1} z_{i+n}|.$$

A number r is rational if and only if it terminates as:

$$r_k = \infty = [z_k\ 0\ z_{k+2}\ 0\ \cdots\ z_{k+2+2n}\ 0\ \cdots].$$

(iv) The terms z_n can be characterized as follows:

$$\begin{aligned} z_{n+1} = 0 &\quad \textit{implies} \quad z_n z_{n+2} \neq 0; \\ |z_n| = 1 &\quad \textit{implies} \quad z_n z_{n+1} > 0; \\ z_n z_{n+1} < 0 &\quad \textit{implies} \quad |z_{n+1}| \geq 2; \\ z_n z_{n+1} < 0, |z_{n+1}| = 2 &\quad \textit{implies} \quad z_{n+1} z_{n+2} > 0. \end{aligned}$$

Proof: Let $\rho_n = [i_n, s_n]$ be the rational approximant to r_n used by Algorithm 2 for computing $z_n = \rho_n \div 1$. By (9) and $\Delta(i_n, s_n) < 1$, we have:

$$\begin{array}{rll} 0 < i_n < s_n < \infty & \text{implies} & r_n \in (z_n - \frac{1}{2}, \infty); \\ \infty < i_n < s_n < 0 & \text{implies} & r_n \in (\infty, z_n + \frac{1}{2}); \\ i_n < 0 < s_n & \text{implies} & r_n \in (-1, 1); \\ i_n > 0 > s_n & \text{implies} & r_n \in (|z_n| - \frac{1}{2}, \frac{1}{2} - |z_n|). \end{array} \tag{20}$$

(i) Let us compute the **E** fraction of some fixed interval ρ. We see, by (9), that:

1. $\rho \div 1 = 0$ if and only if $\rho \cap (-1/2, 1/2) \neq \emptyset$.
2. $\rho \div 1 = 0$ and $\frac{1}{\rho} \div 1 = 0$ if and only if $\rho \supset [\frac{1}{2}, 2]$, or $\rho \supset [-2, -\frac{1}{2}]$.

By Definition 1, the only real number, all of whose approximants contain either of the above intervals, is undefined $\perp$.

(ii) - If $|z_n| \geq 2$, then $r_n \in (|z_n| - \frac{1}{2}, \frac{1}{2} - |z_n|)$ follows from (20).

- If $z_n = 1$, then $i_n \leq 3/2$ and, since $\Delta(i_n, s_n) < 1$, we get $r_n \in (\frac{1}{2}, \infty)$, a stronger condition than $r_n \in (\frac{1}{2}, -\frac{1}{2})$. Similarly, if $z_n = -1$, then $r_n \in (\infty, -\frac{1}{2}) \subset (\frac{1}{2}, -\frac{1}{2})$.
- Finally, if $z_n = 0$, then $\Delta(\frac{1}{2}, r_n) < 1$ implies $|r_n| < 2$.

(iii) By (ii), $z_n = 0$ and $\rho_n \neq \perp$ implies $\rho_n \subset (2, -2)$, hence $\rho_{n+1} = \frac{1}{\rho_n} \subset (-\frac{1}{2}, \frac{1}{2})$. By (i), it follows that $z_{n+1} = \rho_{n+1} \div 1 \neq 0$.

We now show that $z_{n+1} = 0$ implies that $|z_n + z_{n+2}| > |z_n|$. This certainly holds if $z_n \times z_{n+2} > 0$, so we can assume, w.l.o.g., that $z_n > 0 > z_{n+2}$. Since $z_n = \rho_n \div 1$, it follows that $z_n = \lfloor i_n \rceil$, and $s_n < -z_n + 1/2$. As $z_{n+2} \in \rho_n - z_n$, we obtain $z_{n+2} < -2z_n + 1$, hence $z_n < |z_n + z_{n+2}|$.

(iv) - If $z_{n+1} = 0$, then, by (ii), $z_n \neq 0$ and $z_{n+2} \neq 0$.

- If $|z_n| = 1$, then, by (ii), $r_n \in (\frac{1}{2}, \infty)$ or $r_n \in (\infty, -\frac{1}{2})$, so $r_{n+1} = 1/(r_n - z_n)$ has the same sign as r_n.
- By (20) and $r_{n+1} = 1/(r_n - z_n)$, we observe that $r_{n+1} \in (0, -2)$ or $r_{n+1} \in (2, 0)$, whenever $z_n \neq 0$. It follows that $z_n \times z_{n+1} < 0$ implies $|r_{n+1}| > 2$.

■

8 Bibliography

[88] A. Hurwitz, Über die Entwicklung komplexer Grössen in Kettenbrüche. Acta Mathematica, Bd. 11, S. 187-200, 1888.

[89] A. Hurwitz, Über eine besondere Art der Kettenbruch-Entwicklung reeller Grössen. Acta Mathematica, Bd. 12, S. 367-405, 1889.

[96] A. Hurwitz, Über die Kettenbrüche, deren Teilnenner arithmetische Reihen bilden. Vierteljahrsschrift der Naturforchenden Gesellschaft in Zürich, Jahrg. XLI, Jubelband II, S. 34-64, 1896.

[36] A. M. Turing, On computable numbers, with an application to the Entscheidungsproblem. Proc. London Math. Soc., ser. 2, 42, 3, 4; pp. 230-265, 1936.

[41] A. Church, The Calculi of Lambda-Conversion. Princeton University Press, Princeton, N. J., 1941.

[48] H. S. Wall, Analytic Theory of Continued Fractions. Chelsea Publishing Company, Bronx, N. Y., 1948.

[53] H. G. Rice, Classes of recursively enumerable sets of positive integers and their decision problems. Trans. Amer. Math. Soc., 74, 2, pp. 358-366, 1953.

3.6 Representing Numbers as Infinite Products of Homographies 20

4 Rational Arithmetic with Continued Fractions 21

4.1 Inverse and Opposite of a Continued Fraction 22

4.2 Algebraic Algorithm 22

4.3 Positional Algebraic Algorithm 25

4.4 Homographic Algorithm 26

4.5 Quadratic Algorithm 28

4.6 Square Roots of Rational Numbers 30

5 Transcendental Continued Fraction Arithmetic 32

5.1 Exponentials and Logarithms 32

5.2 Trigonometric Functions 33

6 Conclusion 33

7 Proofs of the theorems 35

7.1 Proof of Theorem 1 35

7.2 Proof of Theorem 2 36

7.3 Proof of Theorem 3 37

7.4 Proof of Theorem 4 38

7.5 Proof of Theorem 5 38

7.6 Proof of Theorem 6 39

8 Bibliography 40

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique

[54] H. G. Rice, Recursive real numbers. Proc. Amer. Math. Soc., 5, 5, pp. 784-791, 1954.

[66] V.A. Ouspenski, Leçons sur les fonctions calculables. Hermann, 1966.

[75] J. Vuillemin Syntaxe, Sémantique et Axiomatique d'un langage de programmation simple. Birkhauser-Verlag, Bâle et Stuttgart, ISR 12, 115 pages, 1975.

[80] E. Wiedmer, Computing with infinite objects. Theoretical Computer Science, 10, pp. 133-155, 1980.

[81] D. E. Knuth, The Art of Computer Programming, vol. 2, Seminumerical Algorithms. Addison Wesley, 1981.

[82] J. Chailloux & al. Le_Lisp, version 15. Manuel de référence. INRIA, Rocquencourt, 1982.

[85] E. Bishop, D. Bridges, Constructive Analysis. Springer Verlag, 1985.

[86] H. J. Boehm, R. Cartwright, M. J. O'Donnel, M. Riggle, Exact real arithmetic: a case study in higher order programming. Proc. ACM conf. on Lisp and Functional Programming, pp. 162-173, 1986.

[87] H. J. Boehm, Constructive Real Interpretation of Numerical Programs. Proc. ACM conf. on Interpretors, pp. 214-221, 1987.

Contents

1 Introduction **2**

2 Computable Real Numbers **3**

2.1 Numbers as sequences of intervals 4
2.2 Numbers as programs 5
2.3 What the best computers cannot do 6
2.4 Approximate Comparison 7
2.5 Approximate Integer Division 8
2.6 Computing with infinity 9
2.7 Stereographic representation of $\mathbf{R}$ 9
2.8 Interval arithmetic 10

3 Representations of R by continued fractions **12**

3.1 Finite representations of infinite sequences of numbers 12
3.2 Classical $\mathbf{N}$ and $\mathbf{Z}$ continued fractions 14
3.3 Redundant Euclidean continued fractions 17
3.4 Normalization of Euclidean Continued Fractions 18
3.5 Comparison between $\mathbf{Z}_\infty$ continued fractions 20