5-12-1998

Exact Real Computational Arithmetic via Infinite Composition of Mobiüs Transformations as presented by Dr. Peter John Potts, et al.

David Michael Robertson

Exact Real Computational Arithmetic via Infinite Composition of Mobiüs Transformations as presented by Dr. Peter John Potts, et al.

David Robertson

In completion of requirements for College Honors

Louisiana State University and A&M College
College of Arts & Sciences

May 12, 1998

ABSTRACT

It is necessarily difficult to precisely represent and manipulate extended and infinite real numbers within the finite bounds of digital computing. Previous algorithms have proven either inaccurate or inefficient, consuming large amounts of memory and/or relying on slow, cumbersome calculations. Dr. Peter John Potts has recently expanded the work of Vuillemin, Nielsen, and Kornerup to produce efficient algorithms for exact real computational arithmetic based on infinite compositions of linear fractional transformations. This paper addresses the major points of his recent research, including algorithms for the rational and transcendental functions. In addition, original source code was composed for simple implementation of these algorithms and comparison to preexisting algorithms of approximation (e.g. on-board transcendental functions, Newton's method, etc.).

1 INTRODUCTION

From basic number theory, recall that the set of **integers** ($\mathbb{Z}$) consists of all of the whole numbers from negative infinity to infinity including zero:

$\mathbb{Z} \subseteq \{-\infty \ldots \text{-2, -1,0,1,2} \ldots \infty\}$

The **rational numbers** ($\mathbb{Q}$) consists of all numbers that can be expressed in the following fashion

$\frac{p}{q}$ where $p \in \mathbb{Z}$, $q \in \mathbb{Z}$

The integers are a subset of the rational numbers since every $n \in \mathbb{Z}$ can be represented as $n/1$. The **irrational numbers** ($\mathbb{I}$) are non-repeating, non-terminating decimals, which cannot be represented as rational fractions. They include numbers like $\sqrt{2}$ and π. Together the irrational and rational numbers compose the entire set of **real numbers**.

In digital computing, real numbers are typically stored as a finite string of digits. Clearly, this method only accurately represents rational numbers of length less than or equal to the prescribed string length. When dealing with irrational numbers or rational number of greater length, round-off error quickly becomes a problem in any type of iterated or repeated calculation.

Alternatively, **interval notation** can be used to approximately define any real number x by the endpoints of an interval guaranteed to contain it. The smaller the distance between these endpoints, the more precisely the number is defined. However, if the endpoints are represented as rationals of a fixed degree of accuracy, calculations quickly become meaningless. For example, let us restrict our discussion to the integers and define 3 by the interval (2, 4). Adding the number to itself, we arrive at the representation (4, 8) to define the integer 6, which could alternatively represent the integers 5 or 7. This inaccuracy grows with every subsequent calculation, until the representation essentially contains no data. This phenomenon extends naturally to intervals denoted by rational endpoints. It is clear that interval notation is useless when the endpoints are fixed.

The logical alternative is to employ computational algorithms that allow infinite strings themselves to be manipulated. This field of study is known as **exact real computational arithmetic**. Previous algorithms include the use of infinite sequences of linear maps and continued fraction expansions. However, in several recent articles Dr. Peter John Potts of London's Imperial College has offered a third method involving infinite composition of Mobiüs transformations, which can be viewed as a generalization of the two previous methods. This method and the Potts' algorithms are the focus of this paper.

2 CONTINUED FRACTIONS

Continued fractions offer an alternative method of representing real numbers. A typical continued fraction is of the form

$$a_0 + \cfrac{b_0}{a_1 + \cfrac{b_1}{a_2 + \cfrac{b_2}{\cdots}}}$$

with the usual assumption that all of the coefficients are positive integers. We call a continued fraction a simple coninued fraction if for all $n \in \mathbb{Z}$, $b_n = 1$. For convinience we shall denote the continued fraction above by

$$[a_0, b_0; a_1, b_1; a_2, b_2 \ldots]$$

Finite continued fractions represent rational numbers, since any terminating continued fraction can be manipulated algebraically to yield a simple fraction of the form $\frac{a_\Omega}{b_\Omega}$ $a_\Omega, b_\Omega \in N$. However, note that a_Ω and b_Ω can become quite large for rational numbers of extended length. It has been shown that infinite real numbers can be represented by non-terminating continued fractions. Further, the continued fraction representation of both rational and irrational number is unique in standard decimal representation.

Continued fractions can also be used to represent functions. The Taylor Series expansion of a function f(x) can often be used to derive a number of continued fractions with the general form

$$f(x) = \alpha_0(x) + \cfrac{\beta_0(x)}{\alpha_1(x) + \cfrac{\beta_1(x)}{\alpha_2(x) + \cfrac{\beta_2(x)}{\cdots}}}$$

For continued fraction representation to offer a viable solution to the problem of exact real arithmetic, it must posses two qualities. First, it must be possible to derive efficient algorithms for storing continued fractions. Second, we must be able to manipulate these algorithms quickly and efficiently.

3 Linear Fractional Transformations

In the case of finite continued fractions, each "level" can be represented as a rational operation performed on the value that the function tree below it returns. This process begins with the last node, or terminal function, which simply returns a rational value, and then continues "up" the continued fraction to the "root" function, which returns the final value of the entire fraction. It is very practical then, to view the continued fraction as a nested series of these rational functions, which are called linear fractional transformations (LFTs). We introduce two types of LFTs: Mobiüs transformations, and tensors.

The predecessor of the linear fractional transformation is the simple fraction. For the purposes of this discussion, simple fractions are those which contain single-term, integer values in the numerator and denominator. As we proceed, it will become desirable to represent LFTs simply by the value of their integer coefficients. Therefore, we represent the simple fraction $\frac{a}{b}$, $a, b \in \mathbb{Z}$ with the vector $\begin{pmatrix} a \\ b \end{pmatrix}$ and define

$$\Phi\begin{pmatrix} a \\ b \end{pmatrix} = \frac{a}{b}.$$

We define $\mathbb{V}$ as the set of vectors with integer coefficients, and let V denote any general element of $\mathbb{V}$ ($V \in \mathbb{V}$). We say that two numbers x and y have the same sign if $x \times y \geq 0$. Therefore, we define $\mathbb{V}^+$ as the subset of all $V \in \mathbb{V}$ with same sign coefficients. The vector product X is defined by

$$\begin{pmatrix} a \\ b \end{pmatrix} X \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ bd \end{pmatrix}$$

Mobiüs transformations are linear fractional transformations that manipulate one real variable x as shown, $x \mapsto \frac{ax+c}{bx+d}$ $a,b,c,d \in \mathbb{Z}$, and which can be represented in 2 X 2 matrix form as $M = \begin{pmatrix} a & c \\ b & d \end{pmatrix} \in \mathbb{M}$

by defining

$$\Psi\begin{pmatrix} a & c \\ b & d \end{pmatrix}(x) = \frac{ax+c}{bx+d}$$

where $\mathbb{M}$ is the set of 2X2 matrices, $\mathbb{M}^+$ is the subset of $\mathbb{M}$ with same-sign coefficients, and $M \in \mathbb{M}$. The function $\Psi(\mathbb{M})$ can stretch, shrink or shift any single variable $x \in \mathbb{R}$. The symbol • is used to denote the dot product between matrices so that

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix} \bullet \begin{pmatrix} e & g \\ f & h \end{pmatrix} = \begin{pmatrix} ae+cf & ag+ch \\ be+df & bg+dh \end{pmatrix}$$

Composition of matrices then corresponds to matrix multiplication. It is also clear that

$$\Phi(M)(\Psi(V)) = \Phi(M \bullet V)^1$$

We define the inverse of a matrix by

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix}^{-1} = \begin{pmatrix} d & -c \\ -b & a \end{pmatrix}$$

We define a third operation, the **2-dimensional Mobiüs transformation**, which acts on the two real numbers, represented below as the ordered pair (x,y)

---

[1] For proof of axioms presented in this section see Apendix A

$$(x,y) \mapsto \frac{axy+cx+ey+g}{bxy+dx+fy+h} \qquad a,b,c,d,e,f,g,h \in \mathbb{Z}$$

represented by the **tensor**

$$\mathrm{T} = \begin{pmatrix} a & e \\ & c & g \\ b & f \\ & d & h \end{pmatrix} \in \mathbb{T} \quad \text{or more compactly} \quad \mathrm{T} = \begin{pmatrix} a & c & e & g \\ b & d & f & h \end{pmatrix} \in \mathbb{T}$$

by defining

$$\mathrm{Y}\begin{pmatrix} a & c & e & g \\ b & d & f & h \end{pmatrix}(x,y) = \frac{axy+cx+ey+g}{bxy+dx+fy+h}$$

where $\mathbb{T}$ is the set of 2 X 2 X 2 or 2 X 4 tensors with integer coefficients, $\mathbb{T}^+$ is the subset of $\mathbb{T}$ with same-sign coefficients, and $\mathrm{T} \in \mathbb{T}$. It is clear that through application of tensors to x, y$\in$R the rational operations of addition, subtraction, mutiplication and division are possible. For instance, the tensor $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ represents the multiplication operation.

We can now further define $\mathbb{L} = \mathbb{V}\mathbb{U}\mathbb{M}\mathbb{U}\mathbb{T}$, and $\mathrm{L} \in \mathbb{L}$ any linear fractional transformation, and $\mathbb{L}^+ = \mathbb{V}^+\mathbb{U}\mathbb{M}^+\mathbb{U}\mathbb{T}^+$, and $\mathrm{L}^+ \in \mathbb{L}^+$. For any two matrices $\mathrm{R} = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$ and $\mathrm{S} = \begin{pmatrix} e & g \\ f & h \end{pmatrix}$, let (R S) denote the tensor $\begin{pmatrix} a & & e \\ & c & & g \\ b & & f \\ & d & & h \end{pmatrix}$ and $\begin{pmatrix} R \\ & S \end{pmatrix}$ denote $\begin{pmatrix} a & & c \\ & e & & g \\ b & & d \\ & f & & h \end{pmatrix}$.

We then define the **transpose** of a tensor by

$$(\mathrm{R\ S})^{\mathsf{T}} = \begin{pmatrix} R \\ & S \end{pmatrix}$$

Let us define the **left product** $\bullet_1$ of a tensor with $\mathrm{L} \in \mathbb{V}\mathbb{U}\mathbb{M}$ by

$$\begin{pmatrix} R \\ & S \end{pmatrix} \bullet \mathrm{L} = \begin{pmatrix} R \bullet L \\ & S \bullet L \end{pmatrix}$$

and the **right product** $\bullet_2$ with $L \in \mathbb{V}\mathbb{U}\mathbb{M}$ by

$$(R\ S) \bullet L = (R \bullet L\ \ S \bullet L)$$

It is not possible to apply either of the tensor product operations to $L \in \mathbb{T}$ without further defining a set of four-dimensional tensors. The following noteworthy axioms are proven in Appendix A:

$$Y(T)(M(x), y) = Y\,(T \bullet_1 M)(x,y)$$

$$Y(T)(x, M(y)) = Y\,(T \bullet_2 M)(x,y)$$

$$Y(T)(V,y) = \Psi(T \bullet_1 V)(y)$$

$$Y(T)(x, V) = \Psi(T \bullet_2 V)(x)$$

and

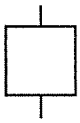
$$M(Y(T)(x,y)) = Y(M \bullet T)(x,y)$$

where $M \bullet (R\ S) = (M \bullet R\ \ M \bullet S)$

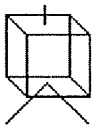
4 Expression Trees

The expression tree representation offers a more suitable means for visualizing linear fractional transformations, although they can still be thought of as branched continued fractions. Vectors are represented by the symbol

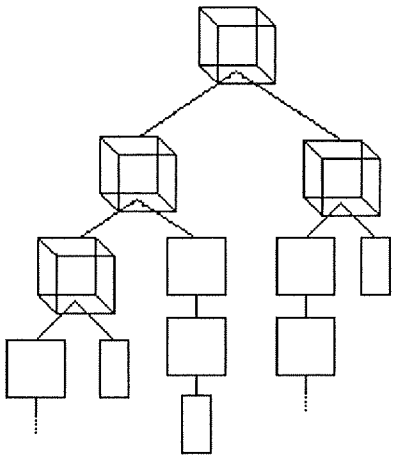
as they take no argument but return a rational value. Matrices are represented as



since they take one argument and return one value. Finally, tensors are represented by



as they take two arguments and return one real value. A typical expression tree is shown below.



In order to derive an expression tree of nested linear fractional transformations from a continued fraction as described above we use the simple matrix identities

$$\begin{pmatrix} a & c\mu \\ b & 0 \end{pmatrix}\begin{pmatrix} d & f \\ e & 0 \end{pmatrix} = \begin{pmatrix} a & c \\ b & 0 \end{pmatrix}\begin{pmatrix} d & f \\ e\mu & 0 \end{pmatrix}$$

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix}\begin{pmatrix} e & g \\ f & h \end{pmatrix} = \begin{pmatrix} c & a \\ d & b \end{pmatrix}\begin{pmatrix} f & h \\ e & g \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\prod_{n=0}^{\infty}\begin{pmatrix} a_n & c_n \\ b_n & d_n \end{pmatrix} = \prod_{n=0}^{\infty}\begin{pmatrix} dn & bn \\ cn & an \end{pmatrix}$$

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a\lambda & b\lambda \\ c\lambda & d\lambda \end{pmatrix} \quad for \lambda \neq 0$$

to find

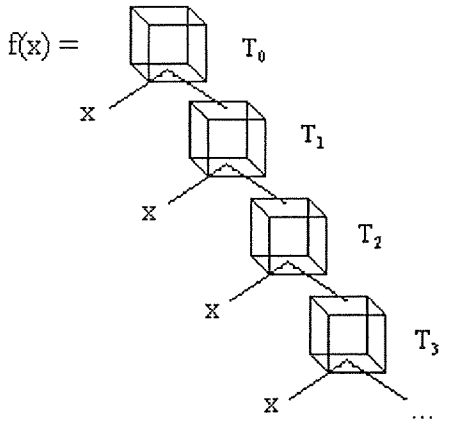
$$\mathrm{T_n} = \begin{pmatrix} a_n & c_n & e_n & g_n \\ b_n & d_n & f_n & h_n \end{pmatrix} \in \mathbb{T}$$

such that

$$\prod_{n=0}^{\infty}\begin{pmatrix} \alpha n(x) & \beta n(x) \\ 1 & 0 \end{pmatrix} \equiv \prod_{n=0}^{\infty}\begin{pmatrix} a_n x + e_n & c_n x + g_n \\ b_n x + f_n & d_n x + h_n \end{pmatrix}$$

The nested series of derived tensors can then be represented as an expression tree.

To evaluate such an expression tree, the downward progression of the tree is "cut" at a desirable point, appropriate values are plugged in for x and y, and the value of the expression tree is then calculated "upward" until the root is reached.



This method of evaluating the expression trees follows an algorithm called strategy by Dr. Potts and which he has shown to be the most efficient means of evaluating these trees. If this is for a single y value, we have accomplished very little, for there is no way of determining the accuracy of the calculation performed (i.e. to what decimal place the calculation is correct. Therefore, we reintroduce interval notation. In the first calculation of the value of the expression tree, at the point of the cut, the tensor is calculated over the entire positive real line $[0, \infty]$, by using the ordered pair (x,0) to find one endpoint, and (x, ∞) to find the other. By evaluating the tensors over the entire real line, we start with the assumption that the tensor contains no information, and then refine our interval through each successive level of the expression tree.

There are two problems with this algorithm. The most immediate is the means of choosing an appropriate cutting point. If we have derived an algorithm for the arcsine function, and our calculations require 25 decimal places of accuracy, at which level of the expression tree should we begin calculations? The second problem is perhaps less glaring, but is at least of equal significance. As mentioned above, denominators and numerators quickly become very large in the simple fraction representation of decimal continued fractions. The size of these integers first, may slow the flow of information rendering this method completely impractical, or second expand beyond the maximum word size permitted by the computer, reintroducing gross miscalculations.

5 INCREMENTAL DIGIT REPRESENTATION

To tackle both of these problems, it is necessary to cultivate a strict understanding of the mappings that allow us to represent the abstract concept of number as strings. The majority of calculations we perform on a daily basis are made possible through the decimal representation of real numbers. This mapping of the reals to strings is known as an incremental digit representation, and can be expressed rigorously. The tuple $(B,\Delta,\psi,\Omega,\phi)$ is called an **unsigned incremental digit representation** if

the **base interval** B is a member of $\mathbb{IR}^{\infty}$,

the **digits set** Δ is a countable set of symbols,

the **digit map** ψ is a function $\Delta \to B \to B$,

the **terminator set** Ω is a countable set of symbols and

the **terminator map** ϕ is a function $\Omega \to B$.

The defining tuple for decimal representation on the interval [0,1] is

$$([0,1],\ \{0,\ldots,9\},\ d \mapsto x \mapsto \tfrac{d+x}{10},\ \{1,\ldots,9\},\ \tau \mapsto \tfrac{\tau}{10}).$$

The decimal $x = 0.2876189$ is represented as a nested series of the function ψ :

$$\psi(2)(\psi(8)(\psi(7)(\psi(6)(\psi(1)(\psi(8)(\Omega(9))))))).$$

To compute the value of this expression, we begin by applying Ω to the last digit of x that lies in the terminator set $\{1,\ldots,9\}$, in this case 9.

$$\Omega(9) = 0.9$$

This value then becomes our d for the next iteration, in which we begin applying the digit map ψ.

$$\psi(8)(0.9) = \frac{8+0.9}{10} = 0.89$$

$$\psi(1)(0.89) = \frac{1+0.89}{10} = 0.189$$

Continuing until we reach the outer "layer" of this nested set, we arrive at the anticipated value 0.2876189. This concept can be expanded to the entire real line by defining the signed incremental digit representation. The tuple $(B,\Sigma,\varphi,\Delta,\psi,\Omega,\phi)$ is called a **signed incremental digit representation** if

$(B,\Delta,\psi,\Omega,\phi)$ is an unsigned incremental digit representation,

the **sign set** Σ is a countable set of symbols and

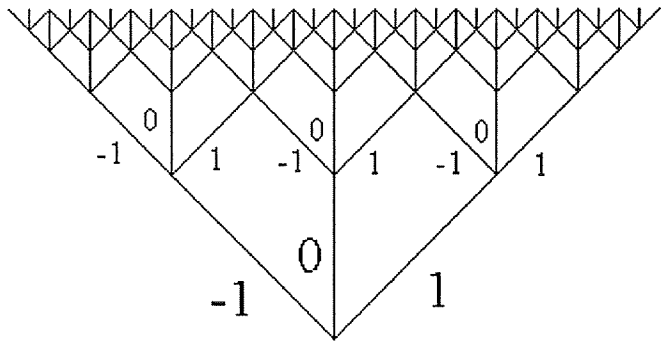
the **sign map** φ is a function $\Sigma \to B \to I\mathbb{R}^{\infty}$.

Full decimal representation on the real line can be represented by the tuple

$$([0,1], Z, \sigma \mapsto x \mapsto \sigma + x, \{0,\ldots,9\}, d \mapsto x \mapsto \tfrac{d+x}{10}, \{1,\ldots,9\}, \tau \mapsto \tfrac{\tau}{10}).$$

The sign map Σ simply takes the result of the iterated digit map ψ and adds an element of the integers to the left of the decimal place, producing any value on the real line.

Another powerful signed incremental digit representation is the redundant binary system, represented by the diagram below.



The redundant binary system defines real number by repeated division of an interval by two. Movement towards the lower (left) bound of the interval is denoted by –1, movement "up" the division tree is denoted by 0, and movement towards the upper (right) bound is denoted by 1. This system is called redundant because the same real value can be represented by different series of strings. For instance, on the interval [0,1], the following strings are equivalent

$$-1\ 0\ 1 = 0\ -1\ -1\ = \frac{5}{16}\ (\textit{decimal}).$$

The signed incremental digit representation for the redundant binary system on the real line is expressed by the tuple

$$([-1,1], Z, \sigma \mapsto x \mapsto \sigma + x, \{-1, 0, 1\}, d \mapsto x \mapsto \tfrac{d+x}{2}, \{-1,0,1\}, \tau \mapsto \tfrac{\tau}{2}).$$

The redundant representation system is valuable in comparing values stored in interval notation. Ideally, the point represented by any interval should lie near the interval's center. However, as calculations are made, it is possible that the value we want moves

towards one endpoint. This makes comparisons inaccurate. Suppose we used the interval [1,2] to represent the number 1, giving an accuracy of +-0.5. However, comparison with the number 0.9999... would always yield a negative result since it is not in the interval. In redundant binary representation, since there is more than one way to represent any number it is possible to choose an interval appropriate for comparison. To illustrate we present the following algorithm to convert real numbers into redundant binary notation, using "rif" to denote a redundant if statement

$$\text{f(x)} = \quad \text{rif x} < \left[-\frac{1}{2},0\right] \text{then } -1 : \text{f(2x+1) else}$$

$$\text{rif x} < \left[0,\frac{1}{2}\right] \text{then } 0 : \text{f(2x) else } 1\text{: f(2x-1)}$$

For example

$$\left[\left[f\left(\frac{8}{25}\right)\right]\right] = \left\{0 : \left[\left[f\left(\frac{16}{25}\right)\right]\right], 1 : \left[\left[f\left(-\frac{9}{25}\right)\right]\right]\right\}$$

$$= \left\{0 : 1 : \left[\left[f\left(\frac{7}{25}\right)\right]\right], 1 : -1 : \left[\left[f\left(\frac{7}{25}\right)\right]\right], 1 : 0 : \left[\left[f\left(-\frac{18}{25}\right)\right]\right]\right\}$$

6 EXACT FLOATING-POINT REPRESENTATION

As mentioned earlier, when using the decimal system to represent the rational numbers generated by continued fractions, these numbers tend to become large very quickly. Redundant binary representation does not suffer the same fate. It can be shown that each "movement" along one branch of the redundant binary tree can be represented by one of the following digit matrices

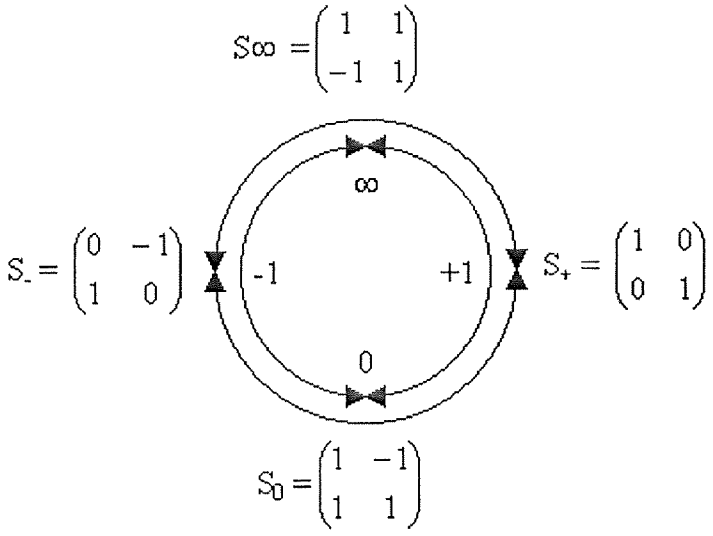
$$D_{-1} = \begin{pmatrix} 1 & 0 \\ 1 & 2 \end{pmatrix} \qquad D_0 = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix} \qquad D_1 = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}$$

Thus the continued refinement of accuracy as one moves up the redundant binary tree can be expressed as the product of the series of corresponding digit matrices

$$D_{d1}D_{d2}\ldots D_{dn} = \mathcal{D}_c^n = \begin{pmatrix} 2^n + c + 1 & 2^n + c - 1 \\ 2^n - c - 1 & 2^n - c + 1 \end{pmatrix} \text{where } c = \sum_{i=1}^{n} d_i 2^{n-i}$$

It is therefore possible to store a sequence of n digit matrices in n+1 bits of memory. It is important to note that the original sequence of digit matrices usually cannot be recovered, with the exception of the case $n = 1$.

We now define **unsigned exact floating point representation** as the unsigned incremental digit representation denoted by the tuple $([0,\infty], D_{\{-1,0,1\}}, \Psi, \mathbb{V}^+, \Phi)$. This can be expanded to a signed incremental digit representation by the inclusion of a sign term. To do this we represent sign with by composition with one of the four sign matrices below.



The use of four sign matrices rather than two, as in decimal representation, is necessary to take full advantage of the redundant binary representation scheme we have chosen, by allowing each digit set to potentially represent four unique values. The **signed exact floating point representation** is defined by $([0,\infty], S_{\{+,\infty,-,0\}}, \Psi, D_{\{-1,0,1\}}, \Psi, \mathbb{V}^+, \Phi)$. Thus, using the characteristics of redundant binary representation, exact floating point representation makes is possible to store and manipulate large rational numbers with only a fraction of the space required to accomplish the same in decimal representation.

7 Conversion of Expression Trees to Exact Real Representation

It has been shown that infinite expression trees can exactly represent real numbers, and that exact floating point representation offers tools to store and manipulate these

numbers with relative efficiency. It is now necessary to define algorithms that allow for the conversion of expression trees into exact floating-point representation. These take the form of emission and absorption rules. To describe tha amount of information contained in a matrix, We define the function

$$\text{info}\begin{pmatrix} a & c \\ b & d \end{pmatrix} = \begin{cases} \left[\frac{a}{b}, \frac{c}{d}\right] \text{ if } \det\begin{pmatrix} a & c \\ b & d \end{pmatrix} < 0 \\ \left[\frac{c}{d}, \frac{a}{b}\right] \text{ if } \det\begin{pmatrix} a & c \\ b & d \end{pmatrix} > 0 \end{cases}$$

In the neglected case $\det\begin{pmatrix} a & c \\ b & d \end{pmatrix} = 0$, it can be shown that $\exists p,q,r,s \in \mathbb{Z}$ such that $\begin{pmatrix} a & c \\ b & d \end{pmatrix} = \begin{pmatrix} rp & sp \\ rq & sq \end{pmatrix}$ in which case

$$\text{info}\begin{pmatrix} a & c \\ b & d \end{pmatrix} = \mathbb{R}^{\infty} \quad \text{if } rs < 0$$

$$\text{info}\begin{pmatrix} a & c \\ b & d \end{pmatrix} = \left\{\frac{p}{q}\right\} \quad \text{if } rs > 0$$

We now define two methods for converting expression trees into exact floating-point representation. **Emission** is to extract information directly from the root node. **Absorption** is to assimilate into the root node from further down the expression tree. Emission is the quicker and easier of the two, and is always more desirable if possible.

A matrix E can be emitted from the root L node if

$$\text{info (E)} \supseteq \text{info(L)} \Leftrightarrow E^{-1} \bullet L \in \mathbb{L}.$$

For a signed expression tree only a sign matrix can be emitted, resulting in an unsigned expression tree

$$E \to S_{\sigma}\left\{S_{\sigma}^{-1}[E]\right\}$$

provided $S_{\sigma}^{-1}[E] \in \mathbb{E}^{+}$. For an unsigned expression tree only digit matrices may be emitted, resulting in another unsigned expression tree:

$$E \to D_d\{D_d^{-1}[E]\}$$

provided Dd-1[E] $\in \mathbb{E}^+$.

Any continued fraction representation of a function can be converted in eaxect real representation by

(i) Converting the continued fraction into an expression tree

(ii) Finding a sequence of matrices $M_n \in \mathbb{M}$ for n ≥ 0 and a matrix N such that

$$M_{n-1}^{-1} \bullet T_n \bullet_1 N \bullet_2 M_n \in \mathbb{T}^+$$

for all n ≥ 1. In which case

$$f(N(y)) = (T_0 \bullet_1 N \bullet_2 M_0)\{y,E_1(y)\} \text{ where}$$

$$E_n(y) = (M_{n-1}^{-1} \bullet T_n \bullet_1 N \bullet_2 M_n)\{y,E_{n+1}(y)\}$$

for all $y \in [0,\infty]$.

It has been shown that the above techniques can be used to construct algorithms for the basic transcendental functions: square root, exponent, natural logarithm, tangent, and inverse tangent. From these the remaining real functions can be derived. However, these algorithms and their derivation are beyond the scope of this discussion[2].

8 INFORMATION FLOW ANALYSIS

One of the main advantages of the exact floating-point representation is that it gives a natural unit of information. The metric $d_J : [0,\infty]X[0,\infty] \to [0,2]$ defined by

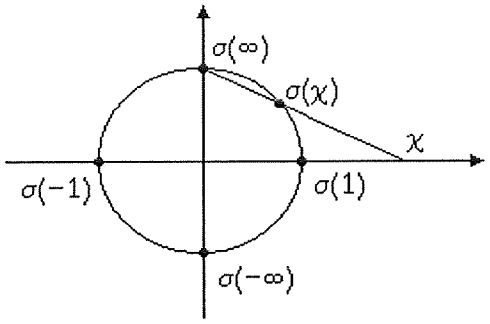
$$d_J(x,y) = |S_0(x) - S_0(y)|$$

is topologically equivalent to the chordal metric

$$d_C(x,y) = |\sigma(x) - \sigma(y)| = \frac{2|x-y|}{\sqrt{x^2+1}\sqrt{y^2+1}}$$

which defines the distance between two points on the compactifiaction of the real line shown below

[2] Tensor representation of the transcendental functions are listed in Appendix B



defined by the equation

$$\sigma(x) = \frac{2x + (x^2 - 1)i}{x^2 + 1}.$$

From this relation, it is possible to define functions that allow us to determine the level of precision contained in each level of an algorithm. Thus, we can efficiently determine exactly how many iterations of any algorithm are necessary to obtain the desired level of precision. Again, these algorithms are beyond the scope of this paper.

9 Conclusion

We started this discussion by demonstrating some of the shortcomings of straight digital and fixed incremental digital representation in the exact representation and manipulation of real numbers. We discussed the value of continued fractions in representing real numbers and their functions. We then introduced algorithms for converting continued fractions into expression trees, for easy computational analysis.

In order to combat extremely large integer size, we introduced redundant binary representation, via a discussion of incremental digit representation. We introduced exact real incremental digit representation as a signed redundant binary representation on the entire real line. We then outlined methods for converting signed expression trees into exact floating point representation. We introduced the concept of non-fixed interval notation. Finally, we discussed the derivation of the transcendental functions themselves and the properties of exact floating-point representation that make efficient computation possible. It has been shown that Peter John Potts and his colleagues have developed

efficient algorithms for exact online real analysis. However, the algorithms outline herein are more time consuming than standard methods of analysis, and are therefore useful only when extreme precision is required. It has been predicted, though, that hardware assisted software and advances in parallel processing will improve the efficiency of these methods, perhaps one day making them suitable for everyday use.

REFERENCES

Crown, G.D., Fenrick, H. M., Valenza, R. J.. *Abstract Algebra*. Marcel Dekker, Inc, New York, 1986.

Potts, P.J., Edalat, A. "Efficient on-line computation of real functions using exact floating point," October 1997, Imperial College, available at http://theory.doc.ic.ac.uk/~pjp.

Potts, P.J., Edalat, A. "Computable real arithmetic using linear fractional transformations", June 1996. Early draft PhD thesis, Imperial College, available from http://theory.doc.ic.ac.uk/~pjp.

Potts, P.J., Edalat, A. "Exact Real Arithmetic based on Linear Fractional Transformations", December 1996. Imperial College, available at http://theory.doc.ic.ac.uk/~pjp.

Moore, R.E. *Interval Analysis*. Prentice Hall, Englewood Cliffs, 1966.

Khinchin, A.Ya.. *Continued Fractions*, 3rd edition. University of Chicago Press, Chicago Illinois, 1964.

APPENDIX A : COMPOSITION OF LINEAR FRACTIONAL TRANSFORMATIONS

Selected proofs of axioms from the main text are offered here. Remaining axioms can be either directly inferred from those below.

Let $M = \begin{pmatrix} k & m \\ l & n \end{pmatrix}$, $V = \begin{pmatrix} r \\ s \end{pmatrix}$, and $T = \begin{pmatrix} a & c & e & g \\ b & d & f & h \end{pmatrix}$. For the purposes of tensor multiplication we further define $R = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$, $S = \begin{pmatrix} e & g \\ f & h \end{pmatrix}$, and $T = (R\ S)$.

Axiom: $\Phi(\mathrm{M})(\Psi(\mathrm{V})) = \Phi(\mathrm{M}\bullet\mathrm{V})$

Proof: $\Phi(M)(\Psi(V)) = \dfrac{l\frac{r}{s}+m}{k\frac{r}{s}+n} = \dfrac{lr+ms}{kr+ns} = \Phi\begin{pmatrix} lr & ms \\ kr & ns \end{pmatrix} = \Phi(M \bullet V)$

Axiom: $\mathrm{Y(T)\ (V},y) = \Psi(\mathrm{T} \bullet_1 \mathrm{V})(y)$

Proof:

$$\mathrm{Y(T)\ (V},y) = \frac{a\frac{r}{s}y + cy + e\frac{r}{s} + g}{b\frac{r}{s}y + dy + f\frac{r}{s} + h} = \frac{ary + csy + er + gs}{bry + dsy + fr + hs} = \frac{(ar+cs)y + (er+gs)}{(br+ds)y + (fr+hs)}$$

$$= \Psi\begin{pmatrix} ar+cs & er+gs \\ br+ds & fr+hs \end{pmatrix}(y) = \Psi\begin{pmatrix} ar+cs & \\ & er+gs \\ br+ds & \\ & fr+hs \end{pmatrix}(y) = \Psi\begin{pmatrix} R \bullet V & \\ & S \bullet V \end{pmatrix}(y)$$

$$= \Psi(T \bullet_1 V)(y)$$

Axiom: $\mathrm{M(Y(T)}(x,y)) = \mathrm{Y(M} \bullet \mathrm{T})(x,y)$

Proof: $\mathrm{M(Y(T)}(x,y)) = M\left(\dfrac{axy+cx+ey+g}{bxy+dx+fy+h}\right) = \dfrac{k\left(\frac{axy+cx+ey+g}{bxy+dx+fy+h}\right)+m}{l\left(\frac{axy+cx+ey+g}{bxy+dx+fy+h}\right)+n}$

$$= \frac{k(axy+cx+ey+g)+m(bxy+dx+fy+h)}{l(axy+cx+ey+g)+n(bxy+dx+fy+h)} = \frac{(ak+bm)xy+(ck+dm)x+(ek+fm)y+(g+hm)}{(al+bn)xy+(cl+dn)x+(el+fn)y+(gl+hn)}$$

$$= \mathrm{Y}\begin{pmatrix} ka+mb & kc+md & ke+mf & gk+mn \\ (la+nb & lc+nd & le+nf & \lg+hn \end{pmatrix}(x,y) = \mathrm{Y}\begin{pmatrix} k & m \\ l & n \end{pmatrix} \bullet \begin{pmatrix} a & c & e & g \\ b & d & f & h \end{pmatrix}$$

$= \mathrm{Y(M} \bullet \mathrm{T})(x,y)$

APPENDIX B : REAL FUNCTIONS

Basic Functions

$$\log(x) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 2n+1 \end{pmatrix} \bullet_2 \begin{pmatrix} 0 & n+1 \\ n+1 & 2 \end{pmatrix}$$

$$\exp(x) = \begin{cases} \begin{pmatrix} 1 & 1 & 2 & 0 \\ -1 & 1 & 2 & 0 \end{pmatrix} \text{if } n = 0 \\ \begin{pmatrix} 0 & 1 & 4n+2 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \text{if } n \geq 1 \end{cases} \quad \text{for } x \in [-1,1]$$

if x is outside this range, apply $\exp(x) - \exp(x/2))_2$ until it is.

$$\tan(x) = \begin{cases} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 2n+1 \end{pmatrix} \textit{if } n \textit{ even} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -2n-1 \end{pmatrix} \textit{if } n \textit{ odd} \end{cases}$$

$$\arctan(x) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ (n+1) & 0 & 0 & 2n+1 \end{pmatrix}$$

Composite Functions

$$x^y = \exp(y \log(x))$$

$$\sin(x) = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \left[\tan\left(\frac{x}{2}\right), \tan\left(\frac{x}{2}\right) \right]$$

$$\cos(x) = \begin{pmatrix} -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \left[\tan\left(\frac{x}{2}\right), \tan\left(\frac{x}{2}\right) \right]$$

$$\arcsin(x) = \arctan \sqrt{\begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix} [x, x]}$$

$$\arccos(x) = \arctan \sqrt{\begin{pmatrix} -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} [x, x]}$$

$$\sinh(x) = \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 \end{pmatrix} [\exp(x), \exp(x)]$$

$$\cosh(x) = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} [\exp(x), \exp(x)]$$

$$\tanh(x) = \begin{pmatrix} 1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 1 \end{pmatrix} [\exp(x), \exp(x)]$$

$$\text{arcsinh}(x) = \log\left(x + \sqrt{x^2 + 1}\right.$$

$$\text{arccosh}(x) = \log\left(x + \sqrt{x^2 - 1}\right.$$

$$\text{arctanh}(x) = \frac{1}{2}\log(S_{\infty}[x])$$