v1: 10 April 2025

Research Article

Arithmetic on Continued Fractions

Peer-approved: 10 April 2025

Qeios, Vol. 7 (2025)
ISSN: 2632-3834

Michael J. Collins[1]

1. Daniel H. Wagner Associates (United States), Exton, United States

Gosper developed algorithms for adding, subtracting, multiplying, or dividing two CFs, and for solving quadratics with CF coefficients, getting a CF as the result. Straightforward implementation of these algorithms can lead to infinite loops; here we present modified versions of those algorithms which avoid all difficulties with infinite loops. We have implemented these algorithms in Haskell.

Corresponding author: Michael J. Collins, mjcollins10@gmail.com

1. Introduction

A continued fraction is a (possibly infinite) expression of the form

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cdots}}}$$

where the terms a_i are integers, with a_i positive when $i > 0$. It is written more compactly as $[a_0, a_1, \cdots]$. The properties of continued fractions are very well-known[1], so here we only remind the reader of some notation we will use:

- A quadratic irrational has a periodic CF expansion. We denote the periodic part by putting it in parentheses, i.e $[1, (2, 3)] = [1, 2, 3, 2, 3, 2, 3, \cdots]$.
- A rational number has a finite CF expansion $[a_0, a_1, \cdots a_k]$. It will be convenient to consider a finite CF as ending with an infinite term $a_{k+1} = \infty$, treating $\frac{1}{\infty}$ as equal to zero.

Gosper (in an appendix to the famous HAKMEM report[1]) developed algorithms for adding, subtracting, multiplying, or dividing two CFs, and for solving quadratics with CF coefficients, getting a CF as the result; the point of course is that we can do this entirely within the CF representation, making no use of floating-point arithmetic. Here we present modified versions of those algorithms which avoid all difficulties with infinite loops; any combination of arithmetic computations can be carried out to any required degree of accuracy in a finite number of steps. We have implemented these algorithms in Haskell[2].

2. Arithmetic on One CF

Before we describe how to add or multiply two CFs, we consider the simpler problem of operations combining a single CF with a rational number p/q. A few examples reveal there is no evident general pattern for transforming the CF terms of irrational x into the terms of px/q or $x + p/q$, even in the apparently simplest cases:

$$\begin{aligned}
\sqrt{7} &= [2, (1, 1, 1, 4)] \\
\sqrt{7}/2 &= [1, (3, 10, 3, 2)] \\
\sqrt{11} &= [3, (3, 6)] \\
\sqrt{11}/2 &= [1, (1, 1, 1, 12, 1, 1, 1, 2)] \\
\pi &= [3, 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, \cdots] \\
\pi + 1/2 &= [3, 1, 1, 1, 3, 1, 3, 4, 73, 6, 3, 3, 2, 1, 3 \cdots]
\end{aligned}$$

As a motivating example for the general algorithm, we compute

$$[y_0, y_1, y_2 \cdots] = \pi/2 = 1.5707963267948966 \cdots .$$

It will be convenient to have notation for the "tail" of a continued fraction, so let

$$r_i = [\pi_{i+1}, \pi_{i+2}, \cdots] .$$

 doi.org/10.32388/DD1PU3.2

Note that all $r_i \geq 1$, and $r_i = \pi_{i+1} + 1/r_{i+1}$. The fact that $\pi_0 = \lfloor \pi \rfloor = 3$ is enough to determine $y_0 = \lfloor \pi/2 \rfloor = 1$; more precisely

$$y = \frac{\pi}{2} = \frac{3 + 1/r_0}{2} = \frac{3r_0 + 1}{2r_0} = 1 + \frac{r_0 + 1}{2r_0} = 1 + \frac{1}{[y_1, y_2, \cdots]} \ .$$

Now we have to get y_1 from

$$= \frac{2r_0}{r_0 + 1} \ .$$

The mere fact that $r_0 > 1$ is enough to tell us that $\lfloor \frac{2r_0}{r_0+1} \rfloor = 1$, so we obtain $y_1 = 1$ and continue

$$= 1 + \frac{r_0 - 1}{r_0 + 1} = y_1 + \frac{1}{[y_2, y_3 \cdots]}$$

$$= \frac{r_0 + 1}{r_0 - 1} \ .$$

Now the floor of $\frac{r_0+1}{r_0-1}$ ranges from one to infinity as r_0 ranges from one to infinity, so we need to make use of $\pi_1 = 7$; we substitute $r_0 = 7 + 1/r_1$ to get

$$= \frac{8 + 1/r_1}{6 + 1/r_1} = \frac{8r_1 + 1}{6r_1 + 1} = 1 + \frac{2r_1}{6r_1 + 1} = 1 + \frac{1}{[y_3, \cdots]} \ .$$

Thus $y_2 = 1$. We also get $y_3 = 3$ from

$$= \frac{6r_1 + 1}{2r_1} = 3 + \frac{1}{2r_1} = 3 + \frac{1}{[y_4, \cdots]} \ .$$

Going further will require substituting $r_1 = 15 + 1/r_2$ into $2r_1 = [y_4, \cdots]$. It is now clear that we will generate terms of y by repeatedly determining the integer part (i.e. floor) of expressions of the form

$$\frac{px + q}{rx + s} \tag{1}$$

where $x > 1$ and the continued fraction expansion of x is known. Functions of this form are called *homographic*. We will identify a homographic function with the matrix $\left(\begin{smallmatrix} p & q \\ r & s \end{smallmatrix}\right)$.

To formalize our observations, we first define the *range* of the function given by (1) as the set of possible floors, i.e.

$$\mathcal{R}\left(\begin{smallmatrix} p & q \\ r & s \end{smallmatrix}\right) = \left\{ \left\lfloor \frac{px + q}{rx + s} \right\rfloor \mid 1 \leq x < \infty \right\}$$

We will be able to produce the next term of output when this set is a singleton. If $-s/r \leq 1$ then the denominator cannot be zero, and the upper and lower limits are the min and max of $\{p/r, (p+q)/(r+s)\}$: if p/r is the max, and is also an integer, then the integer part cannot be larger than $p/r - 1$, since we are approaching p/r from below. If $-s/r > 1$ the range will be infinite, so there is no need to compute it explicitly; we need to transform the expression by incorporating more information about x.

We define two transformations on homographics. When the range of the homographic expression for $[y_j, y_{j+1}, \cdots]$ is not a singleton set, we do not yet have enough information to determine y_j, and must *ingest* the next term of x. If this term is k we make the substitution $x \leftarrow k + 1/x$, leading to the definition

$$\text{ingest}(k, \left(\begin{smallmatrix} p & q \\ r & s \end{smallmatrix}\right)) = \begin{pmatrix} q + kp & p \\ s + kr & r \end{pmatrix}$$

When the range of the homographic expression for $[y_j, y_{j+1}, \cdots]$ consists of a single integer k, we can *produce* the next output term $y_j = k$; the rest of the output, $[y_{j+1}, y_{j+2}, \cdots]$, is the continued fraction expansion of

$$\frac{1}{\frac{px+q}{rx+s} - k} = \frac{rx + s}{(p - kr)x + (q - ks)}$$

so we define

$$\text{produce}(k, \left(\begin{smallmatrix} p & q \\ r & s \end{smallmatrix}\right)) = \begin{pmatrix} r & s \\ p - kr & q - ks \end{pmatrix} \ .$$

Note that if we started with rational x, we will eventually reach $x_i = \infty$; ingesting infinity returns the limit $\left(\begin{smallmatrix} 0 & p \\ 0 & r \end{smallmatrix}\right)$. This is the constant function at the rational number p/q, so there will never be any need to ingest the (nonexistent) subsequent terms of x. If we proceed as in the examples above, we will compute the CF representation of p/q as the last part of the finite expansion of y, ending with a production that subtracts the final term y_k from the expression $\frac{y_k}{1}$; this leads to $M = \left(\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right)$, which we may read as infinity.

So now we have an algorithm for producing the CF expansion $[y_0, y_1, \cdots]$ of $\frac{px+q}{rx+s}$, assuming the expansion of x is known. To compute x/k we would start with $M = \left(\begin{smallmatrix} 1 & 0 \\ 0 & k \end{smallmatrix}\right)$; to get kx we would start with $M = \left(\begin{smallmatrix} k & 0 \\ 0 & 1 \end{smallmatrix}\right)$; while $x + \frac{j}{k}$ would be $M = \left(\begin{smallmatrix} k & j \\ 0 & k \end{smallmatrix}\right)$.

doi.org/10.32388/DD1PU3.2

$i \leftarrow 0$ {index of the next term of x we will read}
$j \leftarrow 0$ {index of the next term of y we will generate}
$M \leftarrow \left(\begin{smallmatrix} p & q \\ r & s \end{smallmatrix}\right)$ {the initial matrix, i.e. the expression whose CF we will compute}
$M \leftarrow \text{ingest}(x_0, M)$
$i \leftarrow 1$
while $M \neq \infty$ **do**
  while $\mathcal{R}(M)$ is not a singleton **do**
    $M \leftarrow \text{ingest}(x_i, M)$
    $i \leftarrow i + 1$
  end while
  $y_j \leftarrow \mathcal{R}(M)$
  $M \leftarrow \text{produce}(y_j, M)$
  $j \leftarrow j + 1$
end while

Figure 1. Algorithm for arithmetic on a CF and a rational number

2.1. Properties of the Algorithm

The outer loop "while $M \neq \infty$" is an infinite loop if $\mathbf{x} = [\mathbf{x}_0, \mathbf{x}_1, \cdots]$ is irrational (here we use boldface $\mathbf{x}$ for the real-valued input, as distinct from the symbolic variable x in the homographic expression). This can be implemented quite directly in a language like Haskell[3] with lazy evaluation; such languages support conceptually infinite data structures, which are a perfect fit for working with continued fractions. Under lazy evaluation, defining a variable y to be the result of an algorithm just associates y with the finite recursive expression that defines the algorithm; the number of y terms actually generated will be only what is needed by subsequent computations.

The inner loop "while $\mathcal{R}(M)$ is not a singleton" *is* guaranteed to terminate; we will never be in an infinite loop waiting for the next term. We have already observed that this loop terminates if $\mathbf{x}$ is rational, so assume it is irrational. As we iterate through this loop, we are repeatedly rewriting the initial $M = \left(\begin{smallmatrix} p & q \\ r & s \end{smallmatrix}\right)$ with $[\mathbf{x}_0, \cdots \mathbf{x}_i, x]$ in place of x. As i goes to infinity, the upper and lower bounds on $[\mathbf{x}_0, \cdots \mathbf{x}_i, x]$ approach $\mathbf{x}$, so the upper and lower bounds of the matrix approach $\frac{p\mathbf{x}+q}{r\mathbf{x}+s}$. This limit is irrational, so eventually the bounding interval contains no integers; at this point $\mathcal{R}(M)$ will be a singleton (namely the common floor of all numbers in the interval), and we will produce a term.

2.2. Visualizing the Algorithm

Since $\text{ingest}(k, \left(\begin{smallmatrix} p \\ r \end{smallmatrix}\right))$ returns a matrix whose right column is $\left(\begin{smallmatrix} p \\ r \end{smallmatrix}\right)$, and $\text{produce}(k, \left(\begin{smallmatrix} p & q \\ r & s \end{smallmatrix}\right))$ returns a matrix whose top row is $(r\ s)$, we can visualize the progress of the algorithm as moving a 2×2 window through a two-dimensional grid of integers. We start with M in the upper right corner; *ingest* moves left and *produce* moves down. We put the x_i on the top row, at the positions where we ingest, and y_j in the right column, at the positions where we produce. Here is such a representation of our earlier example of $\pi/2$, taken further to use $\pi = [3, 7, 15, 1 \cdots]$ to obtain $\pi/2 = [1, 1, 1, 3, 31, \cdots]$.

		1	15	7	3		
				3	1	0	
				2	0	2	1
			8	1	1		1
			6	1	−1		1
32	30	2	0				3
1	1	0	1				31
1	−1						

3. Arithmetic on Two Continued Fractions

We now turn to adding or multiplying two CFs. Conceptually, this is hardly any different from what we have already done; but it will raise some implementation issues that require care. As a motivating example, let

$$z = \pi + \sqrt{2} = 4.555806215962888\cdots$$

with $[y_0, y_1, \cdots] = \sqrt{2} = [1, (2)]$. We can write this sum as

$$3 + \frac{1}{[\pi_1, \pi_2 \cdots]} + 1 + \frac{1}{[y_1, \cdots]} = \frac{4[\pi_1, \cdots][y_1, \cdots] + [\pi_1, \cdots] + [y_1 \cdots]}{[\pi_1, \cdots][y_1 \cdots]}.$$

Substituting $[\pi_1, \cdots] = 7 + 1/[\pi_2, \cdots]$ and $[y_1, \cdots] = 2 + 1/[y_2, \cdots]$ leads (after a lot of high-school algebra) to

$$z = \frac{65[\pi_2, \cdots][y_2, \cdots] + 29[\pi_2, \cdots] + 9[y_2, \cdots] + 4}{14[\pi_2, \cdots][y_2, \cdots] + 7[\pi_2, \cdots] + 2[y_2, \cdots] + 1}$$

The integer parts of $\frac{65}{14}$, $\frac{29}{7}$, $\frac{9}{2}$ and $\frac{4}{1}$ are all 4; therefore $\lfloor z \rfloor = 4$. The next term will be the floor of

$$\frac{1}{z - 4} = \frac{14[\pi_2, \cdots][y_2, \cdots] + 7[\pi_2, \cdots] + 2[y_2, \cdots] + 1}{9[\pi_2, \cdots][y_2, \cdots] + [\pi_2, \cdots] + [y_2, \cdots]}$$

into which we can substitute $[\pi_2, \cdots] = 15 + \frac{1}{[\pi_3, \cdots]}, [y_2, \cdots] = 2 + \frac{1}{[y_3, \cdots]}$ and so on. Similarly, the product $\pi\sqrt{2} = 4.442882938158366\cdots$ is

$$(3+\frac{1}{[\pi_1,\cdots]})(1+\frac{1}{[y_1\cdots]})$$
$$=\frac{3[\pi_1,\cdots][y_1\cdots]+3[\pi_1,\cdots]+[y_1\cdots]+1}{[\pi_1,\cdots][y_1\cdots]}.$$

into which we can make the same substitutions.

In general, computing terms of a sum or product of CFs will require finding the floors of two-variable expressions of the form

$$\frac{axy+bx+cy+d}{exy+fx+gy+h} \tag{2}$$

where x and y independently vary from 1 to ∞. Such an expression is called *bihomographic*, and will be represented by the matrix $\left(\begin{smallmatrix}a&b&c&d\\e&f&g&h\end{smallmatrix}\right)$. To determine bounds on the floor of (2), it is convenient to make the substitutions $\hat{x}=x-1, \hat{y}=y-1$, and consider

$$\frac{a\hat{x}\hat{y}+(a+b)\hat{x}+(a+c)\hat{y}+(a+b+c+d)}{e\hat{x}\hat{y}+(e+f)\hat{x}+(e+g)\hat{y}+(e+f+g+h)}$$

as $\hat{x},\hat{y}$ range independently from *zero* to infinity. If the denominator cannot be zero, the floor is always between the minimum and maximum of

$$\left\{\frac{a}{e},\frac{a+b}{e+f},\frac{a+c}{e+g},\frac{a+b+c+d}{e+f+g+h}\right\}$$

where we may ignore fractions with numerator and denominator both zero.

When the floor (i.e. the next term of output) is known, we produce output $z_j=k$ and the next bihomographic expression is

$$\left(\frac{axy+bx+cy+d}{exy+fx+gy+h}-k\right)^{-1}$$
$$=\frac{exy+fx+gy+h}{(a-ke)xy+(b-kf)x+(c-kg)y+(d-kh)}$$

so we define the corresponding operation on matrices

$$\text{produce}(k,\left(\begin{smallmatrix}a&b&c&d\\e&f&g&h\end{smallmatrix}\right))=\left(\begin{smallmatrix}e&f&g&h\\(a-ke)&(b-kf)&(c-kg)&(d-kh)\end{smallmatrix}\right)$$

If the floor is not determined (in particular when the denominator might be zero), we must narrow the range by ingesting the next term of either x or y. If we use $x=[s,x_{k+1},x_{k+2}\cdots]$, we make the substitution $x\leftarrow s+1/x$ to get

$$\frac{(sa+c)xy+(sb+d)x+ay+b}{(ae+g)xy+(af+h)x+ey+f}$$

so we define

$$\text{ingest\_x}(s,\left(\begin{smallmatrix}a&b&c&d\\e&f&g&h\end{smallmatrix}\right))=\left(\begin{smallmatrix}(sa+c)&(sb+d)&a&b\\(ae+g)&(af+h)&e&f\end{smallmatrix}\right)$$

with the analogous

$$\text{ingest\_y}(s,\left(\begin{smallmatrix}a&b&c&d\\e&f&g&h\end{smallmatrix}\right))=\left(\begin{smallmatrix}(sa+b)&a&(sc+d)&c\\(se+f)&e&(sg+h)&g\end{smallmatrix}\right)$$

for the result of substituting $y\leftarrow s+1/y$. If one or both inputs are rational, we will eventually ingest infinity, which we define by taking the limit:

$$\text{ingest\_x}(\infty,\left(\begin{smallmatrix}a&b&c&d\\e&f&g&h\end{smallmatrix}\right))=\left(\begin{smallmatrix}0&0&a&b\\0&0&e&f\end{smallmatrix}\right)$$

$$\text{ingest\_y}(\infty,\left(\begin{smallmatrix}a&b&c&d\\e&f&g&h\end{smallmatrix}\right))=\left(\begin{smallmatrix}0&a&0&c\\0&e&0&g\end{smallmatrix}\right)$$

$i\leftarrow 0$ {index of the next term of x and y we will read}
$j\leftarrow 0$ {index of the next term of z we will generate}
$M\leftarrow\left(\begin{smallmatrix}a&b&c&d\\e&f&g&h\end{smallmatrix}\right)$ {the initial matrix, i.e. the expression whose CF we will compute}
$M\leftarrow\text{ingest\_y}(y_0,\text{ingest\_x}(x_0,M))$
$i\leftarrow 1$
while $M\neq\infty$ **do**
  while $\mathcal{R}(M)$ is not a singleton **do**
    $M\leftarrow\text{ingest\_y}(y_i,\text{ingest\_x}(x_i,M))$
    $i\leftarrow i+1$
  end while
  $z_j\leftarrow\mathcal{R}(M)$
  $M\leftarrow\text{produce}(z_j,M)$
  $j\leftarrow j+1$
end while

Figure 2. Preliminary algorithm for arithmetic on two CFs

We now have a preliminary algorithm for arithmetic; this will in fact require substantial modification, but we include it as a summary of the key ideas. It is not much different from the single-CF case. Computations of $x+y, x-y, xy,$ and x/y begin with matrices $\left(\begin{smallmatrix}0&1&1&0\\0&0&0&1\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}0&1&-1&0\\0&0&0&1\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}1&0&0&0\\0&0&0&1\end{smallmatrix}\right)$ and $\left(\begin{smallmatrix}0&1&0&0\\0&0&1&0\end{smallmatrix}\right)$.

Note that it is not at all necessary to ingest terms of both inputs in lockstep as we have here; Gosper suggests heuristics for accelerating the algorithm by choosing the input term most likely to get us to the next production step more quickly. This appears to be a worthwhile subject for further work.

It is interesting that the algorithms for all arithmetic operations are identical, except for the initial M. The reason, one might say, is that division is the hardest arithmetic operation, and the definition of continued fractions has division built-in to everything.

3.1. Failure to Converge

Our preliminary algorithm demonstrates the basic idea of CF arithmetic, but is insufficient because it can fail to converge; we might endlessly ingest terms of both x and y without ever obtaining a bihomographic

doi.org/10.32388/DD1PU3.2

expression whose floor is known. The simplest case is multiplying $\sqrt{2} = [1, (2)]$ by itself. The first iteration, ingesting 1 from x and from y, gives

$$z = \sqrt{2} * \sqrt{2} = [z_0, z_1, \cdots] = \frac{xy + x + y + 1}{xy} \ .$$

Repeatedly ingesting 2 from both x and y yields

$$[z_0, z_1, \cdots] = \frac{9xy + 3x + 3y + 1}{4xy + 2x + 2y + 1} \tag{3}$$

$$[z_0, z_1, \cdots] = \frac{49xy + 21x + 21y + 9}{25xy + 20x + 10y + 4} \tag{4}$$

$$[z_0, z_1, \cdots] = \frac{289xy + 119x + 119y + 49}{144xy + 60x + 60y + 25} \tag{5}$$

and so on. The matrix (3) takes values ranging from $\frac{16}{9}$ to $\frac{9}{4}$ (i.e. 1.777 to 2.25) as x and y range from one to infinity; similarly matrix (4) is between $\frac{49}{25}$ and $\frac{100}{49}$ (i.e. 1.96 to 2.0408). And the last ranges from $\frac{576}{289}$ to $\frac{289}{144}$, which is about 1.993 to 2.00694.

The source of the problem is now clear: we should have $z = [z_0, z_1] = [2, \infty]$, but no finite number of terms can ever tell us that $z_0 = 2$. It is always possible that one input is in fact less than $\sqrt{2}$, so it is always possible that $z < 2$ and $z_0 = 1$.

Our approach to the problem (see[2] for an entirely different approach) is to separate the internal representation of a CF as a data structure from the mathematical notion of a sequence of integer terms. The internal representation will contain explicit terms (i.e. $z_k = n$) whenever such can be determined, but might also contain arbitrary bounds on the tail $[z_k, z_{k+1} \cdots]$, such as the sequence of bounds we derived for $\sqrt{2} * \sqrt{2}$. This is a direct generalization, because the explicit term $z_k = n$ is equivalent to the bound

$$n \le [z_k, z_{k+1} \cdots] < n + 1 \ .$$

Therefore, when a CF is constructed explicitly from a known sequence of integer terms $[a_0, a_1, \cdots]$, it will be represented internally as the sequence of half-open intervals

$$[[a_0, a_0 + 1), [a_1, a_1 + 1), \cdots] \ .$$

At the opposite extreme, the internal representation for $\sqrt{2}^2$ will consist entirely of ever-tighter bounds on the first term, i.e. the sequence of intervals

$$[[\frac{16}{9}, \frac{9}{4}), [\frac{49}{25}, \frac{100}{49}), [\frac{576}{289}, \frac{289}{144}), \cdots] \ .$$

The general idea (elaborated more fully in section 3.3) is that we will generate output on *every* iteration of the arithmetic algorithm; when we cannot produce the next term of z, we output upper and lower bounds on the current homographic matrix, leaving the matrix unchanged. Reading such output, there can be no ambiguity about the term to which a bound applies; the first bound applies to z_0, and as soon as we encounter an explicit term $[a, a + 1)$ we know that $z_k = n$ and the next bound will apply to z_{k+1}. This implies that any finite prefix of the output sequence will determine a finite prefix $[z_0 \cdots z_{k-1}]$ of the actual CF, followed by a bound $[\ell_k, u_k)$ on the tail $[z_k, z_{k+1} \cdots]$. The exact value of the CF is between $[z_0 \cdots z_{k-1}, \ell_k]$ and $[z_0 \cdots z_{k-1}, u_k]$; here we extend the usual CF notation by allowing a rational number, instead of an integer, as the last term.

Note that storage of a long sequence of intervals is not an issue for implementation; successive bounds on the same term are always tighter, so there is no need to retain earlier bounds when a new one is obtained.

3.2. Extracting Terms

We must note that our representation of CF arithmetic makes it impossible to simply ask a question like "what are the first five terms of $z = xy$?" An explicit list of terms can only be relative to a required degree of accuracy ε. Our arithmetic algorithms consume and return theoretically infinite sequences of intervals; extracting a finite sequence of terms is a separate "post-processing" step, carried out only when a numerical approximation is required. To obtain such an approximation, we find a prefix of the output interval sequence which is long enough to make the difference between $[z_0 \cdots z_{k-1}, \ell_k]$ and $[z_0 \cdots z_{k-1}, u_k]$ less than ε; then our approximate CF is $[z_0 \cdots z_k]$, where z_k is some integer in the interval $[\ell_k, u_k)$. We might as well always take $z_k = \lfloor u_k \rfloor$, which is the only integer in the interval if the interval is small.

In the case of $\sqrt{2}^2$, any accuracy threshold will yield the same result, $z = [2]$.

3.3. An Algorithm for CF Arithmetic

We now present the full details of the algorithm suggested in the previous section. We augment the matrix M with a pair of intervals $\{I_x, I_y\} = \{[x_\ell, x_u), [y_\ell, y_u)\}$, the current bounds on the remaining tails of the inputs x, y. These intervals start at $(-\infty, \infty)$, since the first term could be any integer. When we read an interval of the form $[a, a + 1)$ from x or from y, we modify M as before with $M \leftarrow \text{ingest\_x}(a, M)$ or $M \leftarrow \text{ingest\_y}(a, M)$,

doi.org/10.32388/DD1PU3.2

and we change that variable's bound to $[1, \infty)$. When we read an *ambiguous* interval $[\ell, u)$ (i.e. a bound that does not determine the next term of the input), we replace the bound I_x or I_y with $[\ell, u)$, leaving M unchanged.

For output, we first redefine $\mathcal{R}(M)$ to take the bounds into account, i.e.

$$\mathcal{R}\left(\begin{pmatrix} a & b & c & d \\ e & f & g & h \end{pmatrix}, I_x, I_y\right) = \left\{ \left\lfloor \frac{axy + bx + cy + d}{exy + fx + gy + h} \right\rfloor \mid x \in I_x, y \in I_y \right\}$$

When this is a singleton set $\{a\}$, we output $[a, a+1)$ and modify M as before with $M \leftarrow \text{produce}(a, M)$; the intervals I_x and I_y do not change. Otherwise we leave M unchanged and output the interval from min to max of $M(x, y)$ subject to $x \in I_x, y \in I_y$: denote this interval $\rho\left(\begin{pmatrix} a & b & c & d \\ e & f & g & h \end{pmatrix}, I_x, I_y\right)$. These bounds are easily computed, see section (3.4). To retain unlimited accuracy, the bounds must be computed with exact rational arithmetic.

```
{input x, y: CFs represented as lists of intervals}
{input M: bihomographic matrix}
{output z: CF expansion of M(x,y), represented as a list of intervals}
i ← 0 {index of the next element of x and y we will read}
j ← 0 {index of the next element of z we will generate}
I_x, I_y ← (−∞,∞),(−∞,∞) {current bounds on remaining parts of x, y}
while M ≠ ∞ do
  {read input}
  if x_i = [a, a+1) (where a is an integer) then
    M ← ingest_x(a, M)
    I_x ← [1, ∞)
  else
    I_x ← x_i
  end if
  if y_i = [a, a+1) (where a is an integer) then
    M ← ingest_y(a, M)
    I_y ← [1, ∞)
  else
    I_y ← y_i
  end if
  i ← i + 1
  {produce output}
  if R(M, I_x, I_y) is a singleton set {a} then
    M ← produce(a, M)
    z_j ← [a, a+1)
  else
    z_j ← ρ(M, I_x, I_y)
  end if
  j ← j + 1
end while
```

Figure 3. Algorithm for arithmetic on two CFs (without optimizations)

3.4. Computing the range of M

To compute $\rho\left(\begin{pmatrix} a & b & c & d \\ e & f & g & h \end{pmatrix}, [x_\ell, x_u), [y_\ell, y_u)\right)$:

- Let $\delta_x, \delta_y = x_u - x_\ell, y_u - y_\ell$
- Substitute $x \leftarrow x_\ell + \frac{\delta}{x'+1}$ and $y \leftarrow y_\ell + \frac{\delta}{y'+1}$. Now x goes from x_ℓ to x_u as x' goes from 0 to infinity, similarly for y, y'
- Let the coefficients of the resulting matrix be $\begin{pmatrix} a' & b' & c' & d' \\ e' & f' & g' & h' \end{pmatrix}$

The range of M subject to the bounds will be the min and max of

$$\left\{ \frac{a'}{e'}, \frac{b'}{f'}, \frac{c'}{g'}, \frac{d'}{h'} \right\}$$

assuming no sign change in the denominator as x' goes from 0 to infinity; fractions with numerator and denominator both 0 can be ignored.

3.5. Possible Optimizations of the CF Arithmetic Algorithm

3.5.1. Winnowing Interval Sequences

When we have to ingest the next interval from x or y, and we see an ambiguous interval at the head of the list, it is not necessary to ingest that interval; we may skip it and go on to the next element of the list, which will give a tighter bound on the next term. Any such heuristic must guarantee that, given an infinite list of ambiguous input intervals, we eventually ingest something. Similarly, the algorithm does not need to output every ambiguous interval of z; but we must guarantee that we eventually output something, even if we are generating an infinite list of ambiguous intervals. The most obvious idea is to read and produce ambiguous intervals only if the gap is smaller than some threshold.

3.5.2. Taming Large Coefficients

If a high degree of accuracy is required, the coefficients of M may grow very large; performing rational arithmetic on arbitrarily large integers will incur a performance penalty. We might be able to take advantage of the fact that we do not require the tightest possible bounds on the next term; the only requirement for correctness is that the bounds converge to the correct value. We might divide every coefficient of M by some large integer k, rounding each result up or down in whichever direction slightly widens the range of M. We might do the same with intervals whose numerators and denominators are very large. To guarantee convergence, we must never make an adjustment which makes a new bound weaker than the current bound.

Any such decision to "simplify" M must consider the gap between the upper and lower bounds of M; if the

doi.org/10.32388/DD1PU3.2

gap is large relative to the reciprocals of the coefficients, we might be carrying too many digits. However, with lazy evaluation, all computations are driven by how much accuracy we need; if the gap is small, yet we are still continuing the computation, it means that we need even tighter bounds. In such cases large coefficients might be unavoidable.

4. Solving Quadratic Equations

4.1. Square root of a rational number

The arithmetic algorithm can be extended to solve quadratic equations with CF coefficients. We begin with the special case of computing the square root of a rational number α. The key idea is that $\sqrt{\alpha}$ is the *fixed point* of a homographic function, i.e. the solution to

$$y = \frac{\alpha}{y} \ .$$

We thus need to find a continued fraction y which produces y given the initial matrix $\left(\begin{smallmatrix} 0 & p \\ q & 0 \end{smallmatrix}\right)$, where $\alpha = \frac{p}{q}$.

As a concrete example, we find $\sqrt{11}$. The first term will be the floor of the fixed point of $M = \left(\begin{smallmatrix} 0 & 11 \\ 1 & 0 \end{smallmatrix}\right)$. It is easy to observe in this case that $M(3) > 3$ and $M(4) < 4$, so the first term is $y_0 = 3$; but we must find a better algorithm than linear search through $M(1), M(2), M(3) \cdots$. We can make use of the fact that any homographic function of the form

$$M = \begin{pmatrix} p & q \\ r & -p \end{pmatrix}$$

is *self-inverse*: i.e. $M(M(y)) = y$. This immediately implies that, given any $\hat{y}$, the fixed point is between $\hat{y}$ and $M(\hat{y})$. So we can use binary search to find the desired term. Search can begin with the smallest positive integer greater than the root of the denominator, $\frac{p}{r}$. In fact, the self-inverse property implies the existence of an integer $\hat{y}$ such that $\hat{y} = \lfloor M(\hat{y}) \rfloor$ or $\hat{y} = 1 + \lfloor M(\hat{y}) \rfloor$; in either case $\lfloor M(\hat{y}) \rfloor$ is the next term of y. The self-inverse property is preserved by ingesting and then producing the same term:

$$\text{produce}\left(\hat{y}, \text{ingest}\left(\hat{y}, \left(\begin{smallmatrix} p & q \\ r & -p \end{smallmatrix}\right)\right)\right) = \left(\begin{smallmatrix} \hat{y}r - p & r \\ q + 2\hat{y}p - \hat{y}^2 r & p - \hat{y}r \end{smallmatrix}\right) \ .$$

Thus each subsequent iteration can be handled the same way. After $y_0 = 3$ we obtain

$$M = \begin{pmatrix} 3 & 1 \\ 2 & -3 \end{pmatrix}$$

which yields $y_1 = 3$ since now $M(3) = \frac{10}{3}$. Ingesting and producing 3 leads to

$$M = \begin{pmatrix} 3 & 2 \\ 1 & -3 \end{pmatrix}$$

implying $y_2 = 6$ because $M(6) = \frac{20}{3}$. The next iteration puts us in a loop: ingesting and producing 6 takes us back to

$$\begin{pmatrix} 3 & 1 \\ 2 & -3 \end{pmatrix}$$

proving that $\sqrt{11} = [3, (3, 6)]$.

4.2. Square root of an explicit CF

Now we consider how to extract the square root of a real number x given as a continued fraction. We will start with the simpler case in which x expressed as an explicit list of integer terms, before generalizing to an algorithm that operates on our internal representation of a CF as a list of intervals.

The square root of x is the fixed point $y = \frac{x}{y}$. The matrix $\left(\begin{smallmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{smallmatrix}\right)$ is bihomographic, but we can also think of it as a homographic matrix $\left(\begin{smallmatrix} 0 & x \\ 1 & 0 \end{smallmatrix}\right)$ with constant x and variable y. More generally let $M_{\mathbf{x}}$ denote M with x fixed at $\mathbf{x}$, i.e. if $M = \left(\begin{smallmatrix} a & b & c & d \\ e & f & g & h \end{smallmatrix}\right)$ then

$$M_{\mathbf{x}} = \begin{pmatrix} (a\mathbf{x} + c) & (b\mathbf{x} + d) \\ (e\mathbf{x} + g) & (f\mathbf{x} + h) \end{pmatrix} \ .$$

As before, we use boldface $\mathbf{x}$ to distinguish a specific input from the symbolic variable. M_∞ is the limit $\left(\begin{smallmatrix} a & b \\ e & f \end{smallmatrix}\right)$.

Starting from $\left(\begin{smallmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{smallmatrix}\right)$, we must ingest terms of $\mathbf{x}$ until we reach a stage at which the floor of the fixed point does not depend on x; in other words, until the floor of the fixed point is the same when $x = 1$ and $x = \infty$. This floor will be the next term of y. To find the next term, we first use binary search to get $\hat{y}$, the floor of the fixed point of of M_1; then we check whether $\hat{y}$ is also the floor of the fixed point of M_∞. If so, we ingest $\hat{y}$ and produce $\hat{y}$; if not, we ingest another term of $\mathbf{x}$ and continue.

This works because the matrices M_x will always have the self-inverse property. Such is clearly the case with the initial matrix, and a similar computation to what we did for rational $\mathbf{x}$ proves the property is preserved when we ingest and produce a term.

doi.org/10.32388/DD1PU3.2

As an example, we compute the fourth root of 2, i.e. the square root of $\mathbf{x} = [1, (2)]$. Making the substitution $x \leftarrow 1 + \frac{1}{x}$ gives

$$M = \frac{x+1}{xy} \ .$$

The fact that $\lfloor \mathbf{x} \rfloor = 1$ should be enough to imply that $\lfloor y \rfloor = 1$, and indeed this is the case:

$$\begin{array}{cc} M_1 = \frac{2}{y} & M_\infty = \frac{1}{y} \\ M_1(1) = 2 & M_\infty(1) = 1 \end{array}$$

Ingesting and producing $y_0 = 1$ leads to $M = \frac{yx+x}{y-x}$; then ingesting 2 from $\mathbf{x}$ gives

$$M = \frac{2xy + 2x + y + 1}{xy - 2x - 1} \ .$$

We do not yet have enough information to determine the next output term:

$$\begin{array}{cc} M_1 = \frac{3y+3}{y-3} & M_\infty = \frac{2y+2}{y-2} \\ M_1(6) = 7 & M_\infty(5) = 4 \end{array}$$

Ingesting another 2 from $\mathbf{x}$ will suffice:

$$M = \frac{5xy + 5x + 2y + 2}{2xy - 5x + y - 2}$$

and

$$\begin{array}{cc} M_1 = \frac{7y+7}{3y-7} & M_\infty = \frac{5y+5}{2y-5} \\ M_1(5) = \frac{21}{4} & M_\infty(5) = 5 \end{array}$$

so $y_1 = 5$. After ingesting/producing 5 and then reading another 2 from $\mathbf{x}$ we reach

$$M = \frac{13xy + 5x + 5y + 2}{7xy - 13x + 5y - 5}$$

which tells us that $y_2 = 3$: $M_1(3) = \frac{61}{18}$ and $M_\infty(4) = \frac{19}{5}$. The first few terms of $\sqrt[4]{2}$ are $1.18921\cdots = [1, 5, 3, 1, 1, 40, 5, \cdots]$.

```
{input x: CF represented as a list of intervals}
{output y: CF expansion of √x, represented as a list of intervals}
i ← 0 {index of the next element of x we will read}
j ← 0 {index of the next element of y we will generate}
(x_ℓ, x_h) ← (1, ∞) {current bounds on remaining part of x}
M = ( 0 1 0 0 ) {solve y = x/y}
    ( 0 0 1 0 )
while M ≠ ∞ do
  while floor of fixed point of M_{x_h} does not equal floor of fixed point of M_{x_ℓ} do
    {read input}
    if x_i = [a, a + 1) (where a is an integer) then
      M ← ingest_x(a, M)
      (x_ℓ, x_h) ← [1, ∞)
    else
      (x_ℓ, x_h) ← x_i
    end if
    i ← i + 1
  end while
  while floor of fixed point of M_{x_h} equals floor of fixed point of M_{x_ℓ} do
    {produce output}
    ŷ ← floor of fixed point of M_{x_ℓ}
    if ŷ < 1 then
      y_j ← ∞
      exit{see section (4.5)}
    end if
    y_j ← [ŷ, ŷ + 1)
    j ← j + 1
    M ← produce(ŷ, ingest_y(ŷ, M))
  end while
end while
```

Figure 4. Algorithm for square root of a CF

4.3. Square root of a general CF

The algorithm is not fundamentally different for x represented as a list of intervals. When we read an ambiguous interval $[x_\ell, x_h)$, we do not change M; instead we check whether the floors of the fixed points of M_{x_ℓ} and M_{x_h} are the same. If so, this common value is the next term of y.

To simplify the algorithm we have assumed that all terms of x, including the first term, are greater than or equal to 1. The case of $x_0 = 0$ is easily handled: if $x = [0, x_1, x_2, \cdots]$ then $\sqrt{x} = [0, z_0, z_1, \cdots]$ where $z = \sqrt{[x_1, x_2, \cdots]} = \sqrt{1/x}$.

4.4. Algorithm Termination

It is possible for the square-root algorithm to stall, reading infinitely many terms from $\mathbf{x}$ without ever determining the next term of y. Suppose that the true fixed point of $M_\mathbf{x}$ is an integer k; as we ingest more terms from $\mathbf{x}$, the fixed points of M_{x_ℓ} and M_{x_h} will both approach k, but the floor of one fixed point will be k and the floor of the other will be $k - 1$. We cannot guarantee that a finite number of terms of $\mathbf{x}$ will ever determine $y_i = k$. This would happen, for instance, if we tried to compute the square root of 4, if 4 were given not as $x = [4, \infty]$, but instead as an infinite sequence of ambiguous terms approaching 4.

The solution is the same as in the case of arithmetic; we can output ambiguous terms of y. Let $\phi(M)$ denote the fixed point of M; we would like to output

$y_i = (\phi(M_{x_\ell}), \phi(M_{x_h}))$. This does not quite work, since the bounds can be irrational, so we must instead define rational bounds ${\phi_\ell}'(M_{x_\ell}) \le \phi(M_{x_\ell}), {\phi_h}'(M_{x_h}) \ge \phi(M_{x_h})$; these bounds must be tight enough to guarantee convergence. We can find approximate rational bounds with binary search.

Note that stalling cannot happen if $\mathbf{x}$ is irrational, since the fixed point will never be integral; $\phi(M_{x_\ell})$ and $\phi(M_{x_h})$ will approach a non-integral limit, and after a finite number of steps, the floors of the fixed points will be the same.

4.5. Further Implementation Details

We must take some care in considering the fixed point of M_∞. In our computation of $\sqrt[4]{2}$, we obtained $M = \frac{yx+x}{y-x}$ after ingesting $x_0 = 1$ and then ingesting/producing $y_0 = 1$. At this point, rather than immediately ingesting $x_1 = 2$, we might see if we can get another term of y from M. We have

$$M_1 = \frac{y+1}{y-1} \quad M_\infty = \frac{y+1}{-1} \ .$$

The fixed point of M_1 is between 2 and 3, but the fixed point of M_∞ is negative. What is going on? M_∞ is the limit as $x \to \infty$ for any fixed y, but in this case the fixed point of M_x approaches infinity as $x \to \infty$, leading to the invalid result. At this point we may validly generate $[\alpha, \infty)$ as a bound on y_1, where α is any rational lower bound on the fixed point of M_1; for instance $\alpha = 2$ would be correct.

A similar issue arises in the case where y is in fact a rational number, i.e. a finite continued fraction. Suppose we apply our algorithm to get the square root of 9, i.e. $x = [9, \infty]$. After making the initial substitution $x \leftarrow 9 + \frac{1}{x}$, then ingesting/producing $y_0 = 3$, we will be seeking the fixed point

$$y = \frac{3xy + x}{-3x + y} \ . \tag{6}$$

Note that (6) is simply a rearrangement of $3 + \frac{1}{y} = \sqrt{9 + \frac{1}{x}}$, i.e. of

$$3 + \frac{1}{y} = \frac{9 + \frac{1}{x}}{3 + \frac{1}{y}} \ .$$

Written this way it is obvious that y approaches infinity as x approaches infinity. After ingesting $x = \infty$, the variable x disappears and we will be trying to solve

$$y = \frac{3y + 1}{-3} \ .$$

Now M_1 and M_∞ must be the same; with x gone, both matrices are just $\left(\begin{smallmatrix} 3 & 1 \\ 0 & -3 \end{smallmatrix}\right)$. The fixed point is $y = -1/6$, again meaning that in fact the fixed point goes to infinity as x does; so we terminate the computation with $y_1 = \infty$.

4.6. Quadratic equations

A quadratic equation can be written as the fixed point of a self-inverse homographic function: $py^2 + qy + r = 0$ is equivalent to

$$y = \frac{-qy - 2r}{2py + q} \ . \tag{7}$$

If q is given as a continued fraction, we can solve (7) by starting with $M = \left(\begin{smallmatrix} -x & -2r \\ 2p & x \end{smallmatrix}\right)$ and finding the fixed point, ingesting terms of $q = \mathbf{x}$. The cases of p or r as continued fractions are similar. Computationally, this should be faster than direct application of the quadratic formula, which would require multiple arithmetic operations as well as a square root.

5. Arithmetic with Arbitrary Intervals

In the algorithm for CF arithmetic, all intervals are either ambiguous (i.e. there is an integer in the interior of the interval), or are of the form $[a, a+1)$ for integer a. But this potentially throws away information. Whenever we are able to produce an explicit output term $z_j \leftarrow [a, a+1)$, we in fact know that the tail lies in the possibly smaller interval $\rho(M, I_x, I_y)$, and can set $z_j \leftarrow \rho(M, I_x, I_y)$.

If we generate such intervals as output then we must be able to read them as input. If $x_i = (a + \varepsilon_\ell, a + \varepsilon_u)$, we can ingest $x = a$ into M, and set $I_x \leftarrow (\varepsilon_u^{-1}, \varepsilon_\ell^{-1})$ instead of $I_x \leftarrow [1, \infty)$. This will yield a narrower range for z_j.

Of course there is no guarantee that the more precise representation would give any computational advantage in practice, since more computation is required to manipulate the tighter bounds.

Footnotes

[1] https://en.wikipedia.org/wiki/Simple_continued_fraction

[2] https://github.com/mjcollins10/ContinuedFractions

[3] haskell.org

References

1. ^ *M. Beeler, R. W. Gosper, R. Schroeppel. (1972). HAKMEM. MIT; MIT 1972. Report No.: AIM-293. Available from: https://dspace.mit.edu/handle/1721.1/6086*
2. ^ *David Lester. (2001). Effective continued fractions. In: Proceedings of 15th IEEE symposium on computer arithmetic (ARITH-15 2001). pp. 163–170.*

Declarations

Funding: No specific funding was received for this work.
Potential competing interests: No potential competing interests to declare.