Appendix 2

Continued Fraction Arithmetic

by

Bill Gosper

Abstract: Contrary to everybody, this self contained paper will show that continued fractions are not only perfectly amenable to arithmetic, they are amenable to perfect arithmetic.

Output

Suppose we want the continued fraction for an exact rational number, say 2.54, the number of centimeters per inch. We can execute Euclid's algorithm neatly as follows:

```
254             Initially, 2.54 = 254/100
100     2       Short divide 100 into 254, getting 2, remainder 54
 54     1       54 into 100 goes once, remainder 46
 46     1       46 into 54
  8     5       8 into 46
  6     1
  2     3
  0             We incidentally have found that gcd(254,100) = 2.
```

This says that the continued fraction of 2.54 is 2 1 1 5 1 3, or

$$2.54 = 2 + 1/(1 + 1/(1 + 1/(5 + 1/(1 + 1/3))))$$

$$= 2 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{5 + \cfrac{1}{1 + \cfrac{1}{3}}}}}$$

Similarly, if you want 100/2.54, the number of inches per meter, you will find

39 2 1 2 2 1 4

which is much nicer than

39.(370078740157480314960629921259842519685039)

where the part in parentheses repeats forever. (Incidentally, this repeating decimal is easily computed since the remainder of 2 after the quotient digits 3937 ensures that, starting with 7874..., the answer will be just twice the original one, ignoring the decimal point. Thus you just double the quotient, starting from the left (using one digit lookahead for carries), placing the answer digits on the right, so as to make the number chase its tail. This trick is even easier in expansions of ratios where some remainder is exactly 1/nth of an earlier one, for small n. You forget about the divisor and simply start shortdividing by n at the quotient digit corresponding to the earlier remainder. If this seems confusing, forget it--it has nothing to do with continued fractions.)

Now suppose we want the continued fraction of

$$\frac{70\,t + 29}{12\,t + 5}\,,$$

knowing only that t is positive. We can only give a partial answer-- if t happened to be irrational, for instance, the true answer would go on forever. We do know, however, that as t varies between oo and 0,

$$\frac{70}{12} < \frac{70\,t + 29}{12\,t + 5} < \frac{29}{5}$$

so the first term, at least, is 5. Following the same procedure for Euclid's algorithm:

70 t + 29		
12 t + 5	5	(70 t + 29 - 5 (12 t + 5) = 10 t + 4)
10 t + 4	1	(I.e. between 12/10 and 5/4)
2 t + 1	4	(It could only be 5 if t were truly oo)
2 t		

All we can say about our next quotient is that it lies between 1 and oo. But we have managed to reexpress

$$\frac{70\,t + 29}{12\,t + 5} \quad \text{as} \quad 5 + \cfrac{1}{1 + \cfrac{1}{4 + \cfrac{1}{\cfrac{2\,t + 1}{2\,t}}}}$$

and thereby determine the first three terms of the continued fraction.

If we knew that t > 1/2, we could get another term:

$$\frac{2\,t + 1}{2\,t} = 1 + \frac{1}{2\,t}\,.$$

Input

Now, the opposite problem: suppose you are receiving the terms of a continued fraction 5 1 4 1 ... which may go on forever, or possibly end with a symbolic expression. We wish to reconstruct the value as the terms come in, rather than awaiting an end which may never come.

Let x symbolize the value of the "rest" of the continued fraction, so that before we learn its first term, x stands for the whole thing. When we learn that the first term is 5,

$$x \text{ becomes } 5 + \frac{1}{x} \quad \text{or} \quad \frac{5\,x + 1}{x}\,.$$

When the next term turns out to be 1,

$$x \text{ becomes } 1 + \frac{1}{x} \quad \text{and} \quad \frac{5x+1}{x} \text{ becomes } \frac{6x+5}{x+1} .$$

Then

$$x \text{ becomes } 4 + \frac{1}{x} \quad \text{and} \quad \frac{6x+5}{x+1} \text{ becomes } \frac{29x+6}{5x+1} .$$

Then

$$x \text{ becomes } 1 + \frac{1}{x} \quad \text{and} \quad \frac{29x+6}{5x+1} \text{ becomes } \frac{35x+29}{6x+5} .$$

Finally, when x becomes 2 t, we have reconstructed the original expression.

In general, when

$$x \text{ becomes } a + \frac{1}{x} \text{, then } \frac{qx+r}{sx+t} \text{ becomes } \frac{(aq+r)x+q}{(as+t)x+s} .$$

Although this looks messy, it can be handled almost as neatly as Euclid’s algorithm:

From RIGHT TO LEFT across the page, we write the incoming terms as we learn them:

```
                . . .   1   4   1   5
```

Under the first (rightmost) term, we place the left edge of the array

$$\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \quad \text{symbolizing} \quad \frac{1x+0}{0x+1} \quad \text{i.e. the identity function:}$$

```
                . . .   1   4   1   5

                                    1   0

                                    0   1
```

Then, again from RIGHT TO LEFT, we extend the lower two rows with the simple recurrence: multiply each element by the term in the top row above it, then add the element to the right and write the sum on the left:

```
                . . .   1   4   1   5

                   35  29   6   5   1   0

                    6   5   1   1   0   1
```

That is, 29 = 4 * 6 + 5, and in the bottom row, 6 = 1 * 5 + 1. Letting the last term be 2t,

```
                  2t   1   4   1   5

       70t + 29   35  29   6   5   1   0

       12t + 5     6   5   1   1   0   1
```

The great thing about this process is that you can take other functions by initializing the rightmost matrix to other than

```
        1   0
```

```
        0   1 .
```

Furthermore, it is possible to intersperse steps of the Euclid process between input steps, thereby computing the continued fraction of the value of the function while the value of the argument is still being learned.

Throughput

Suppose, for instance, that we want to compute the continued fraction

```
             2
for     -----------
        3 - sqrt(2)
```

knowing that the continued fraction for sqrt(2) is 1 2 2 2 2 2 ... .
We set up

```
        . . .   2   2   2   2   2   2   1

                                        0   2

                                       -1   3
```

symbolizing

```
        0 sqrt(2) + 2
        ------------- .
        - sqrt(2) + 3
```

Filling in two elements of each row:

```
        . . .   2   2   2   2   2   2   1

                                4   2   0   2

                                3   2  -1   3
```

```
        4   2          4 x + 2
But            means   -------  and since x, the rest of the continued
        3   2          3 x + 2
```

fraction, is between 0 and oo, we know that the answer is between 4/3 and 2/2, so we can perform a Euclid step with the quotient 1 as the first answer term:

```
        . . .   2   2   2   2   2   2   1

                                4   2   0   2

                                3   2  -1   3   1

                                1   0
```

(As before, we list the output terms down the right side.)
Now we must proceed to the left again (unless we are willing to admit that we know x > 2):

```
    . . .   2   2   2   2   2   2   1

                                4   2   0   2

                            8   3   2  -1   3   1

                            2   1   0
```

Any number between 8/2 and 3/1 has 3 as its integer part, so 3 is our second term.

```
        . . .   2   2   2   2   2   2   1

                                4   2   0   2

                            8   3   2  -1   3   1

                            2   1   0           3

                            2   0
```

Continuing:

```
        . . .   2   2   2   2   2   2   1

                                4   2   0   2

                            8   3   2  -1   3   1

                        5   2   1   0           3

                   10   4   2   0               1

                5   2   1   0                   4

           10   4   2   0                       1

            2   1   0                           4
```

But we have been in a loop since the second occurrence of the pattern

```
        2   1

        2   0
```

so we have found that the continued fraction for

```
             2
        -----------  is  1 3 1 4 1 4 1 4 . . .  .
        3 - sqrt(2)
```

A more interesting case: suppose we want the continued fraction for

```
                1     e - 1
           tanh -  =  -----
                2     e + 1
```

and we know that e = 2.71828... = 2 1 2 1 1 4 1 1 6 1 1 8 1 ...,
which we can abbreviate 2 (1 2k+2 1).

```
        . . .   1   1   4   1   1   2   1   2

                                        1   1  -1

                                    4   3   1   1   0

                       12   7   5   2   1   1       2

           20  11   9   2   1   1   0   1           6

            2   1   1   0   1                      10

            0   1
```

Our suspicions aroused by the arithmetic progression developing in
the answer, and especially by the third occurrence of the pattern

```
        2   1
```

```
        0   1 ,
```

we introduce the symbolic term 2k+6

```
  . . .  1     1   2k+6   1   1   4   1   1   2   1   2
                                                  1   1  -1
                                              4   3   1   1   0
                                 12   7   5   2   1   1       2
                     20  11   9   2   1   1   0   1           6
8k+28  4k+15  4k+13   2   1   1   0   1                      10
  2      1      1     0   1                                4k+14
  0      1
```

The reoccurrence, independent of k, of the pattern

```
        2   1                e - 1
            proves that      -----  =  0 2 6 10 (4k+14)  =  0 (4k+2) .
        0   1                e + 1
```

In fact we have proved a more general result. We can replace 2k by f(k), an arbitrary positive integer-valued function, to get the theorem:

```
        if  x = (f(k) 1 1)
```

then

```
        2 x + 1
        -------  =  2 x + 1  =  (2f(k)+2) .
        0 x + 1
```

A handy check on the arithmetic is provided by the fact that the determinant of each of the 2 by 2 matrices is simply negated upon each output and input. Thus, for example, the magnitude of these determinants in the preceding example was always 2:

```
        (8k+12)*1 - (4k+7)*2  =  -2

           1 * 1  - (-1) * 1  =  2
```

Another source of redundancy is the possibility of postponing the output (Euclid) steps for a while, then performing them in a long burst, arriving at the same point in the array via a different route. The disadvantage of this scheme is that larger intermediate numbers are generated.

Functions of the form

```
        p x + q
        ------- ,
        r x + s
```

which we have been abbreviating with the matrix

```
        p   q

        r   s
```

are known as homographic functions. If f(x) and g(x) are two such functions, the matrix for f(g(x)) is simply the product of the matrices for f and g. This can be verified directly by substitution. This means that we can regard a continued fraction

$$x = a\ b\ c\ \ldots = a + \cfrac{1}{b + \cfrac{1}{c + \cfrac{1}{\ldots}}}$$

as a composition of homographic functions:

$$\begin{pmatrix} a & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} b & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} c & 1 \\ 1 & 0 \end{pmatrix} \ \ldots$$

and a homographic function of such an x is merely

$$\begin{pmatrix} p & q \\ r & s \end{pmatrix} \begin{pmatrix} a & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} b & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} c & 1 \\ 1 & 0 \end{pmatrix} \ \ldots$$

Carrying out these multiplications from left to right will produce the same sequence of matrices as successively inputing the terms a, b, c, ... in our array process. To output a term using matrices, multiply on the left by the inverse of the matrix for inputing that term. Thus, our theorem that the function 2 x + 1 will remain unchanged by inputing the sequence f 1 1 and then outputing the term 2f+2 can be written as the matrix identity

$$\begin{pmatrix} 0 & 1 \\ 1 & -2f-2 \end{pmatrix} \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} f & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix} .$$

Here is why we bother with these clumsy matrices: we know that

$$\frac{e + 1}{e - 1} = 2\ 6\ 10\ 14\ \ldots$$

since deleting (or adding) an initial 0 term reciprocates the value of a continued fraction. We wish to use this result to get the continued fraction for 4/e. The problem is: what is the initial matrix? Answer:

$$\begin{pmatrix} 0 & 4 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \begin{matrix} 2 & -2 \\ 1/2 & 1/2 \end{matrix} == \begin{matrix} 4 & -4 \\ 1 & 1 \end{matrix}$$

The left hand matrix says 4/x, the next one says

$$\frac{x + 1}{x - 1} .$$

Inverting it says solve for x (in our case e). (If function composition comes from matrix multiplication, then function inversion must come from matrix inversion.) Multiplying by the first matrix applies the "4 divided by" function. Multiplying all of the elements by 2 is just equivalent to multiplying the numerator and denominator of a fraction.

The following computation of 4/e is much simpler if we squeeze out additional terms from patterns like

8 0

3 1

by using the fact that x > 1, instead of the weaker condition x > 0,
so that we have

```
        8   8 x + 0   8+0                8   8 x + 0   0
        - > ------- > ---  instead of  - > ------- > - .
        3   3 x + 1   3+1                3   3 x + 1   1
```

```
  . . .   8k+26   8k+22  18  14  10   6   2

                                      4   4  -4
                                                             (Determinant = +or- 8)
                                 19   3   1   1   1

                             91   9   1   3       2

                             11   1   1           8
                             -------
                         43 | 3   1 |             3
                            |       |
                         26 | 2  -2 |             1
                             -------
                         17   1                   1

                          9   1                   1

                   144    8   0                   1

                    19    1   1                   7

                    11    1                       1

                     8    0                       1
                   --------
        24k+67    | 3    1 |                      2   (Pattern recurs,
                  |        |                           prompts input of
        16k+42    | 2   -2 |    period begins     1    symbolic terms.)
                   --------
         8k+25      1                             1

         8k+17      1                             1

64k+208     8       0                            k+2

 8k+27      1       1                             7

 8k+19      1                                     1

   8        0                                    k+2
 ------------
| 3        1 |  Pattern recurs, period ends       2
|            |
| 2       -2 |
 ------------
```

This gives us

```
4
-  =  1  2  8  3  1  1  1  1  7  1  1  2 (1 1 1 k+2 7 1 k+2 2)
e

   =  1  2  8  3 (1  1  1 k+1 7  1 k+1 2) .
```

The reason for introducing the input term 8k+22 instead of 4k+22 is
that the matrix recurred only every other input term, thus apparently
regarding the input sequence to be (8k+2, 8k+6) instead of simply
(4k+2). This is evidently related to the fact that this process
is characterized by a determinant of +or- 8. A fun conjecture to
test would be the following generalization of Hurwitz's theorem: The

homographic matrix is periodic iff the input sequence is periodic modulo the determinant.

Inverting Homographic Functions

A very useful trick to add to your high school algebra repertoire:

```
             a x + b           -d y + b
        y  =  -------  ->  x  =  --------  .
             c x + d            c y - a
```

That is, to invert a homographic function, just interchange and negate the diagonal elements of its matrix. This is a shortcut equivalent to inverting the matrix, then multiplying all four elements by minus the determinant. Of course if the determinant, ad - bc, is 0, then we can't solve for x because y is a constant independent of x.

Addition, Multiplication, etc. of Two Continued Fractions

> There is, however, another and yet more significant practical demand that the apparatus of continued fractions does not satisfy at all. Knowing the representations of several numbers, we would like to be able, with relative ease, to find the representations of the simpler functions of these numbers (especially their sum and product).
>
> ...
>
> On the other hand, for continued fractions there are no practically applicable rules for arithmetical operations; even the problem of finding the continued fraction for a sum from the continued fractions representing the addends is exceedingly complicated, and unworkable in computational practice.
>
> --A. YA. KHinchin, 1935

Until now, we have only taken functions of single continued fractions, but to be really useful, our algorithm must extend to arithmetic combinations of two continued fractions, x and y. This we do neatly by expanding to three dimensions. We start with a 2 by 2 by 2 matrix of eight integers. The position of each integer in the matrix is determined by whether or not it is a coefficient of x, whether or not it is a coefficient of y, and whether or not it is in the numerator. (The coefficients of xy are simultaneously coefficients of x and of y.) If we continue the convention of writing the terms of x from right to left across the top of the page, and write the terms of y down the right, where we used to put the outputs, then we can put the 2 by 2 matrix corresponding to the numerator expression where we used to put the initial homographic function matrix. That is, if

```
        x  =  x1  x2  x3 ...
        y  =  y1  y2  y3 ...
```

then we represent

```
        axy + bx + cy + d
```

by

```
        . . .  x3  x2  x1

                       b   d

                       a   c   y1

                               y2
```

```
                                .
                                .
                                .
```

Floating below the surface of the page, directly beneath the bdac matrix, we can imagine the denominator matrix

```
                      f   h

                      e   g
```

representing

```
        exy + fx + gy + h .
```

Thus we can compute any expression of the form

```
        axy + bx + cy + d
        -----------------
        exy + fx + gy + h
```

by starting with the three dimensional matrix

```
        b   d
         f   h
        a   c
         e   g .
```

Let us call such expressions bihomographic.

The algorithms for continued fraction addition, subtraction, multiplication, and division are all identical but for the initialization of the matrix!

```
                  1   0                   1   0
        x + y  =   0   1    x - y  =   0   1
                  0   1                   0  -1
                   0   0                   0   0

                  0   0                   1   0
        x * y  =   0   1    x / y  =   0   0
                  1   0                   0   0
                   0   0                   0   1
```

We shall work through a slightly fancier example function, for no extra cost. Using

```
        y  =  sqrt(6)  =  2 2 4 2 4 2 4 . . .

                            2
                           e  + 1
        x  =  coth 1   =  ------  =  1 3 5 7 9 . . .
                            2
                           e  - 1
```

we will compute

```
        2xy + x           -2        sqrt(6)
        -------  =  (1 + e  ) (1 + -------)
         xy + y                       12
```

which dictates the initial setup

```
   . . . 5   3   1   <- x
                          y
                 1   0    |
                  0   0

                 2   0    2
                  1   1
                          2
```

```
                          4
```

Now as x and y independently vary from 0 to oo,

```
        axy + bx + cy + d
        -----------------
        exy + fx + gy + h
```

varies between limits among the ratios a/e, b/f, c/g, and d/h, provided that the denominator doesn't change sign. For the matrix in question, two of these denominators are zero, and they must be shifted out of the picture by inputing a term of y.

```
   . . . 5   3   1

                 1   0
                  0   0

                 2   0    2
                  1   1

                 5   0    2
                  2   2

                          4
```

```
Here the 2 by 2 matrix  2   0  was multiplied by 2 (the first term of y)
                         1   1

and added to  1   0  to get  5   0  .  Now we observe that the lefthand
               0   0          2   2
```

pair of ratios, 2/1 and 5/2, have the same integer parts, whereas the bottom pair, 5/2 and 0/2, do not. Since our goal is to get them to be equal so that we can perform a Euclid output step, we proceed to the left with an x input. Inputing from y instead would not get rid of the 5/2 and 0/2 for another step.

```
   . . . 5   3   1

                 1   0
                  0   0

             2   2   0    2
              2   1   1

             5   5   0    2
              4   2   2

                          4
```

Unfortunately, this input of 1 still does not provide enough information to determine the output (smaller terms are less informative than larger ones). Nevertheless, the two new ratios, 2/2 and 5/4 have equal integer parts, so we continue leftward.

```
   . . . 5   3   1

                 1   0
                  0   0

         8   2   2   0    2
          7   2   1   1

        20   5   5   0    2
         14   4   2   2

                          4
```

At last we have determined that the first answer term is 1, and we subtract 1 times the denominator matrix $\begin{matrix} 7 & 2 \\ 14 & 4 \end{matrix}$ from the numerator matrix $\begin{matrix} 8 & 2 \\ 20 & 5 \end{matrix}$ to get the new denominator matrix $\begin{matrix} 1 & 0 \\ 6 & 1 \end{matrix}$.

```
   . . . 5    3    1
                                        outputs
                   1    0
                    0    0                 1

         8    2    2    0     2
          7    2    1    1
           1    0

        20    5    5    0     2
         14    4    2    2
           6    1

                              4
```

Again a 0 denominator thwarts immediate hope of another output, but it is in the corner where either an x or a y input will get rid of it. In fact, since the integer parts of the other three ratios are all different, we will need at least two more input terms to get rid of them. We can further deduce that we need at least one y input, otherwise the y-independent ratios will remain between 7 and oo, while the other pair will stay in the disjoint interval between 14/6 and 4. So let’s sample y first.

```
   . . . 5    3    1
                                        outputs
                   1    0
                    0    0                 1

         8    2    2    0     2
          7    2    1    1
           1    0

        20    5    5    0     2
         14    4    2    2
           6    1

         35   10              4
          13    2
```

Now 14/6 and 35/13 have equal integer parts, so we move x-ward.

```
   . . . 5    3    1
                                        outputs
                   1    0
                    0    0                 1

         8    2    2    0     2
          7    2    1    1
           1    0

        20    5    5    0     2
    74   14    4    2    2
     31    6    1

   185   35   10              4
     67   13    2
```

which nets us an output term of 2.

```
   . . . 5    3    1
                                        outputs
                   1    0
                    0    0                 1
                                            2
         8    2    2    0     2
          7    2    1    1
           1    0

        20    5    5    0     2
    74   14    4    2    2
     31    6    1
      12    2

   185   35   10              4
     67   13    2
      51    9
```

```
Now we must input the 4 from the y sequence, whereupon we will get an
output of 1, leaving us with the matrix  51   9
                                          16   4

                                        216  38
                                          83  20 .
```

On the next page is a perspective view of the entire process up until now. The extremely elongated numbers are the inputs, and the three outputs 1 2 1 are in the upper right. This picture was produced with Bruce Baumgart's Geometric Editor (Stanford AI Memo 232).

Another x and y go in and a 2, a 1, and a 1 bubble out:

```
. . .  7   5    3    1
                                          outputs
                     1    0
                      0    0                 1
                                              2
           8    2    2    0     2              1
            7    2    1    1                    2
             1    0                              1
                                                  1
          20    5    5    0     2                  .
      74   14    4    2    2                        .
       31    6    1                                  .
        12    2

     185   35   10              4
       67   13    2
 366    51    9
  116    16    4

      299   58                  2
1550   216   38
  601    83   20
   348    50                    .
    253    33
      95    17                  .
3466   483                      .
 1318   182
   830   119
    488    63
     342    56
```

Unfortunately, except for a few degenerate cases, each matrix variable will tend to grow so as to contain about a quarter as many digits as have been input. However, there is no need for the most difficult multiprecision routines--namely multiply and divide--since multiplies will nearly always involve small terms, and divides are merely integer estimates of ratios. The rare case of a large term can be handled by breaking it up, e.g. 78629 = 78000 0 629 . (See the Zero and Negative Terms section.) Also on the brighter side, the rate of information output will keep up with the inputs except for a slight lag of a term or two due to the crude bounds (0 to oo) used for the unread segments.

But Why All This Trouble?

Why use algorithms that are at least twice as hard as the usual algorithms on numbers in positional notation (e.g. decimal or binary)? One answer is that many numbers, such as pi and e, can be represented exactly, using little programs (coroutines) to provide indefinitely many continued fraction terms on demand.

But the algorithms for sum, product, etc. of two such numbers have this same property, for they produce their output as they read their input. Thus we can cascade several of these routines so as to evaluate arithmetic expressions in such a way that output stream begins almost immediately, and yet can continue almost indefinitely. The user is freed from having to decide in advance just how much precision is necessary and yet not wasteful. Later we will extend this claim to cover series terms and approximating iterations.

When you analyze why people actually bother with numerical arithmetic instead of leaving things symbolic, you find that the main reason is for deciding inequalities. Imagine how much computational effort has been wasted in generating the bits to the right of the decisive ones in inequalities. A fixed word length is almost always too long, yet sometimes too short, whereas term-at-a-time arithmetic can shut off as soon as the "comparands" differ.

Another great waste of effort is the generation of information which is destroyed by truncation and rounding, or discarded in the form of remainders from division. By contrast, information is completely conserved during continued fraction processes, making the arithmetic reversible. In fact, continued fraction arithmetic is information-driven: processing is always directed to the subexpressions upon which the final answer depends most. No input is needlessly read, and no output is needlessly delayed. As a result, quantities which are multipled by small coefficients or 0 will be evaluated only a little, or not at all.

The original arithmetic expression, whose value we seek to print out, is expressed as a composition of homographic and bihomographic functions. (At the bottom level are the constants, which act like functions of no arguments.) These functions are the subexpressions over which the computational effort is distributed. When a function is asked for a term, it performs the algorithm we have described in the preceding pages, asking in turn for zero or more terms from its subfunctions until its pending output term is insensitive to them. If multiple processors are available, a fork can be executed whenever a function finds itself dependent on more than one subfunction.

(Problem: how do you write an arbitrary arithmetic expression as a minimal number of homographic and bihomographic functions?)

Contrary to convention, processing begins at the output routine. The output routine asks the top level function for a datum (term or digit) and the top level function inturn asks for data from the input to which it is most sensitive. Eventually, a bottom level number routine will be reached, whereupon information begins to percolate upward.

But most computations involve imprecise quantites, so why bother with errorless arithmetic? Because the built-in error analysis is guaranteed to stop output before erroneous terms, simultaneously indicating the quantity which limited the significance. The trick is to implement imprecise quantities as bottom level functions of no arguments analogous to those for pi and e, but instead of containing and endless description, these programs emit a loud croak when asked for one more term than they have.

A drawback of this scheme is that continued fraction terms are inadequately small units of information, so that imprecise quantities will usually have a fraction of a term left over, that is, a term whose exact value is uncertain, but bounded more narrowly than between 1 and oo. Furthermore, most of the subexpression modules will also contain partial terms when a computation stalls. The best solution to this problem is to use continued logarithms (later section) instead of continued fractions, but we have a tolerable solution which uses the reversiblity of continued fraction computations. The idea is for each imprecise quantity to describe its upper bound, then take back a term or so and proceed to describe its lower bound. For example the gas constant

```
        PV
        --  =  R  =  8.31432 +or- .00034  Joules/deg/mole
        nT
```

could be converted to two continued fractions--one for the lower limit of 8.31398, and one for the upper limit of 8.31466, but we can effectively get both by running Euclid's algorithm on the lower limit while keeping track of the error interval:

```
        8.31398 + .00068
        1.00000      0     8
         .31398 + .00068   3
         .05806 - .00184   5
         .02368 + .00988   2
         .01070 - .02160   2
         .00328 + .05308
```

In the fifth row a negative error has greater magnitude than the corresponding remainder, indicating that we would be outputing different terms by then if we were doing the upper limit instead. But we can switch over to doing the upper limit simply by adding the last two errors to their corresponding remainders and then continuing:

```
         .01070 - .02160 = -.01090
         .00328 + .05308 =  .05636   0
                           -.01090  -6
                           -.00904
```

Note the 0 term, charateristic of retraction.

This tells us that the true value is between

```
        8 3 5 2 3

and     8 3 5 2 2 0 -6  =  8 3 5 2 -4  =  8 3 5 1 1 3

    =   8 3 5 2 3 0 -7
```

This means that we can describe our imprecise number as

```
        8 3 5 2 3 oo 0 -oo -7 oo
```

where oo means a very large term or, equivalently, a termination signal. The desired effect, anyway, is to squeeze out the partial terms from the immediate superior function by making it think it has gotten a lot of information. Probably the best thing for f(x,y) to do when it receives its first oo from an imprecise x is to set aside copies of its coefficients of x, freeze y input (in case y might deliver a confusing oo), read the rest of x (to the last oo), then replace all of the noncoefficients of x with the copies of the corresponding coefficients of x that had been set aside. Unfortunately, when a function depends on two imprecise arguments, it must go through extra pain to select the extremes from the four values it achieves as each argument saturates at each limit.

Square Roots of Continued Fractions

To find y = sqrt(x), rewrite the equation as

```
        y  =  x/y .
```

Our plan is to extract a term at a time from the continued fraction process for x/y subject to the condition that the output terms of the process must equal the input terms of y.

We will be concerned with matrices whose last element is minus their first. This property is preserved if we always input what we output:

```
                  x

          ax+b     a   b

          cx-a     c  -a    x

              2
      b+2ax-cx   a-cx
```

Another important property of these matrices: if

```
                a y + b
        f(y)  =  -------
                c y - a
```

and we wish to find the "fixed point" of f, i.e. solve the equation

```
        y  =  f(y) ,
```

then the simple iteration

```
        y  <-  average (y, f(y))
```

will be equivalent to the rapidly convergent Newton's iteration for the equivalent equation

```
           2
        c y  - 2 a y - b  =  0 .
```

These particular homographic functions are the self-inverse ones, that is, f(f(y)) = y for all y.

For a warmup exercise, we will assume x to be merely a rational, 17/10, instead of a continued fraction. We set up the matrix for

```
                17
         f(y) = ----
               10 y
```

namely

```
                                        0   17

                                       10    0 .
```

Since the output must equal the input, the next term of y must always be the integer part of the fixed point of the (homographic) function defined by the matrix. To find this we can run a miniature successive approximation for each term. For example, the arbitrary guess that y = 3 gives f(y) = 17/30 , whose integer part is 0. The average of y and f(y), whose integer part is 1, will be much closer, being equivalent to a step of Newton's method. Now f(1) = 17/10, and since the actual value always lies between y and f(y), 1 must be the integer part of sqrt(17/10) and hence the first input and output term. Outputing and inputing a 1:

```
                                        1

                                        0   17

                                   10  10    0    1
```

```
                                   7  -10   17
```

The next term will be the integer part of the solution of

$$y = \frac{10\,y + 10}{7\,y - 10} .$$

We could start by guessing y = 2. Note that since we desire positive terms, we must choose x > 10/7 to avoid the negative root. [f(2)] = 4, so we try the average, 3. [f(3)] = 3, so we output and input 3:

```
                                   3    1

                                        0   17

                                  10   10    0    1

                             11    7  -10   17    3

                              7  -11   40
```

Here we find f(3) = 4, which is no problem, since the actual fixed point is in between and thus must start with 3.

```
                              3    3    1

                                        0   17

                                  10   10    0    1

                             11    7  -10   17    3

                        10    7  -11   40         3

                        10  -10   40
```

Then [f(2)] = 2,

```
                         2    3    3    1

                                        0   17

                                  10   10    0    1

                             11    7  -10   17    3

                        10    7  -11   40         3

                   10   10  -10   40              2

                    7  -10   27
```

But we had this same matrix after the first term, so

$$\mathrm{sqrt}(17/10) = 1\ (3\ 3\ 2) .$$

(Actually, in this special case where the radicand is rational, it is possible to eliminate the guesswork from each iteration by observing that the determinant is preserved. In general, when

$$y = \frac{a\,y + b}{c\,y - a}$$

we have the determinant

$$D = -a^2 - b\,c$$

and

$$y = \frac{a + \sqrt{-D}}{c}$$

so [y] is merely [(a+d)/c], where d = [sqrt(-D)], which we need only compute once at the beginning. The algorithm is then a close equivalent to the one in Knuth, exercise 4.5.3.12. Unfortunately, when the radicand is a continued fraction to begin with, there is no such convenient invariant, so we will need a small iteration for each term.)

The Real Thing

Actually, we can solve any quadratic equation by rewriting it as the fixed point of a self-inverse homographic function:

$$p x^2 + q x + r = 0 \quad \rightarrow \quad x = \frac{- q x - 2 r}{2 p x + q} .$$

So instead of simply taking the square root of a continued fraction, it will be more illustrative to solve a quadratic equation, one of whose coefficients is a continued fraction. We will compute coth 1/2 from

$$\coth 1 = 1\ 3\ 5\ \ldots = (2k+1)$$

using

$$\frac{(\coth t)^2 + 1}{2 \coth t} = \coth 2t$$

with t = 1/2.

The equation we want is

$$y = \frac{x y - 1}{y - x}$$

where x = coth 1 and y = coth 1/2 , giving us the initial setup

```
   . . .  7    5    3    1

                         0   -1
                         -1    0

                         1    0
                          0    1
```

Corresponding to x = oo and x = 0 are the left and righthand 2 by 2 matrices, which, as functions of y, also have the useful property of being self-inverse. This means that we can use the Newton averaging trick to quickly find the integer part of the fixed point of the left matrix, and if it satisfies the righthand one, it is the term to output in the answer and input as y. If the two matrices have different fixed points, more x input is needed. This sounds harder than it is.

Initially, the lefthand (x = oo) equation says

$$y = - y .$$

Guessing y = 69 will give us a value of -69, which, averaged with 69 gives our second approximation, 0, which is exactly right, since the equation happened to be degenerately linear. The righthand equation is

$$y = - 1/y ,$$

which is decidedly not solved by y = 0, so, hardly to our surprise, we must read in a term or more of x before we can begin to output some algebraic function of it.

```
   . . .  7    5    3    1

                   -1    0   -1
                    -1   -1    0

                    1    1    0
                     1    0    1
```

The new lefthand function is truly pathological, being identically 1 except at 1, where it is 0/0. Assuming that we make our algorithm perform more input upon division by 0, two more inputs will occur.

```
   . . .  7    5    3    1

        -16   -3   -1    0   -1
         -21   -4   -1   -1    0

         21    4    1    1    0
          16    3    1    0    1
```

Finally, both of the last two matrices have fixed points solidly between 2 and 3, so we output a 2 in the z direction and input a 2 in the y direction.

```
   . . .  7    5    3    1

        -16   -3   -1    0   -1
         -21   -4   -1   -1    0
           26    5

         21    4    1    1    0     2
          16    3    1    0    1
          -11   -2

          11    2
            4    1
```

Now the lefthand matrix has fixed point between 6 and 7, while 6 plugged into the righthand one gives 15/4. More input.

```
   . . .  7    5    3    1

        -16   -3   -1    0   -1
         -21   -4   -1   -1    0
           26    5

         21    4    1    1    0     2
    115   16    3    1    0    1
     -79  -11   -2

     79   11    2
      29    4    1
```

Trying our 6 in the new lefthand matrix, we win.

```
   . . .  7    5    3    1

        -16   -3   -1    0   -1
         -21   -4   -1   -1    0
           26    5

         21    4    1    1    0     2
    115   16    3    1    0    1
     -79  -11   -2
      589   82

     79   11    2                   6
```

```
      29    4    1
      -95  -13

      95   13
       19    4
```

Now the lefthand matrix says 10, but not the right. Inputing 9 from x confirms the 10 for y.

```
      . . .   9   7    5    3    1

                -16   -3   -1    0   -1
                 -21   -4   -1   -1    0
                   26    5

                 21    4    1    1    0     2
            115   16    3    1    0    1
             -79  -11   -2
              589   82

             79   11    2                  6
        265   29    4    1
        -868  -95  -13
         8945  979

        868   95   13                     10
         175   19    4
         -882  -95

         882   95
          125   29
```

It is not obvious how to show the the answer will indeed be
2 6 10 14 ... = (4k+2).

For a while, this computation will be typical in that the output rate will about match the input rate, while the matrix integers slowly grow, but in this peculiar case, the output terms outgrow the input terms, so that input must occur somewhat oftener to make up the information rate.

Come to think of it, the schoolboy algorithm for square root is also digit-at-a-time, but requires two inputs for each output to avoid souring future outputs.

Square Roots etc. Using Feedback

Suppose that continued fraction arithmetic is being used in a successive approximations process, and suppose further that this process is converging at better than one term per iteration. (Newton's method, for example, delivers exponentially more terms each iteration.) This means that the next approximation will contain at least one more correct term than the current one, independent of the erroneous terms which follow. But a continued fraction process will not request data of which it is independent, and thus it will have already computed the new, correct term by the time it reads the corresponding incorrect term. But then there is no need at all to read the incorrect term, since the correct one is already available. So once a process starts to converge, it can gobble its own output in a feedback loop. (This idea is due to Rich Schroeppel.) There is a minor catch in all of this: in order to be able to output early, the module which computes the approximating expression must "realize" that all instances of the input approximation must vary from 0 to oo in unison. Thus all instances of the approximation variable must be grouped into a single expression which may be more complicated than the ones for simple arithmetic. Generalization of the algorithm to higher dimensions, in order to accomodate additional variables or higher powers, is straightforward but tedious. Someday, I would like to spend some time contemplating the consequences of more complicated feedback arrangements.

Worked Example

To compute x = sqrt(y), Newton's method says

$$x \leftarrow \frac{x^2 + y}{2x}$$

where x is the approximating variable. Unfortunately, if both x and y are continued fractions, the general form of the expression required will be

$$\frac{ax^2y + bxy + cy + dx^2 + ex + f}{gx^2y + hxy + iy + jx^2 + kx + l}$$

which involves twelve integer variables and four dimensions. The feedback technique is more easily demonstrated if y is simply an integer, like 6 for instance.

Then we can use the mechanism for simple arithmetic, starting with the matrix

```
                        0    6
                         1    0

                        1    0
                         0    1
```

i.e. $\frac{x\,y + 6}{x + y}$, which, when y = x, is Newton's method for

x = sqrt(6). In the denominator, the choice of x + y instead of 2x + 0y, for instance, will provide convenient symmetry which will be preserved by the fact that both inputs (and the output) will always be equal.

The four ratios amount to two 0s and two oos, indicating that we will have to warm up the process before it produces terms automatically. To get a term, we must estimate the integer part of the answer, which we do simply by successive substitution using integer arithmetic. Starting with a guess of 3, for instance:

$$\frac{3*3 + 6}{3 + 3} = 2+ , \quad \frac{2*2 + 6}{2 + 2} = 2+$$

so 2 is the first term, which we output and input for both x and y:

```
                . . .   2

                        0    6
                    2    1    0
                     2   -2    6

                        1    0      2
                    1    0    1
                     0    1   -2
                                    .
                    4    1          .
                     2    0         .
```

(We could have done this in any of the six possible orders.) Again the ratios disagree, so we must take another guess and resubstitute it until it stabilizes. Among the four ratios, 0/1 is the limit when both inputs approach 0 (unrealisitic, they are greater than 1), the two 1/0s are the limits when one input approaches 0 while the other approaches oo (absurd, they are equal), and the 4/2 is the limit when they both approach oo. Since the curve is asymptotically flat, this last, lower left ratio, when finite, is the best first guess:

```
           2
        4*2  + (1+1)*2 + 0
        ------------------  =  2+
           2
        2*2  + (0+0)*2 + 1
```

So 2 it is. Again, outputing, inputing, and inputing:

```
           . . .    2   2

                        0    6
                    2    1    0
                     2   -2    6

                        1    0       2
                    1    0    1
                1    0    1   -2
                 0    1   -2

                    4    1           2
                4    2    0
                 1    0    1

                9    4
                 2    1
```

This time, 9/2 suggests 4, which is confirmed by 178/40 = 4+ , so

```
        . . .   4   2   2

                        0    6
                    2    1    0
                     2   -2    6

                        1    0       2
                    1    0    1
                1    0    1   -2
                 0    1   -2

                    4    1           2
                4    2    0
            4    1    0    1
             2    0    2

                9    4               4
            9    2    1
             4    1    0

           40    9
            18    4
```

Now we are cooking, since the three ratios, 40/18, 9/4, and 2/1, all say that the next term is 2, and since everything is positive, the true value must be between the greatest and least ratio. Pumping through this 2, 2 4 2 2

```
                        0    6
                    2    1    0
                     2   -2    6

                        1    0       2
                    1    0    1
                1    0    1   -2
                 0    1   -2

                    4    1           2
                4    2    0
            4    1    0    1
             2    0    2
```

```
              9    4               4
          9    2    1
      9    4    1    0
       2    1    0

         40    9                   2
     40   18    4
       9    4    1
                                   .
     89   40                       .
      20    9                      .
```

We are now to the point of producing outputs twice as fast as they are needed for input, so our matrix is getting overfed. Let's drain it out without inputing to see what's left.

```
      40   18
        9    4
         4    2
          1    0

      89   40
       20    9
         9    4
          2    1
```

outputing a 4 and a 2. But we had this matrix

```
      4    2
       1    0

      9    4
       2    1
```

before, right after processing the second term. Since the matrix alone determines its future inputs, a repetition immediately implies a loop, proving

```
      sqrt(6)  =  2 2 (4 2 4 2)  =  2 (2 4) .
```

Non-regular Continued Fractions

From the (non-regular) continued fraction for arctan 1,

```
     4              1
     --  =  1 + ---------
     pi                 4
               3 + ---------
                            9
                   5 + ---------
                                16
                       7 + ---------
                                     .
                                       .
                                         .
```

we can compute the regular continued fraction for pi:

```
 . . .         100    81    64    49    36    25    16    9    4    1   (1)
         21     19    17    15    13    11     9     7    5    3    1

                                                         12    4    0    4

                                                     24    4    1    1    0    3

                                                      4    0    1
                                                     -------
                                          555   51    6    1

                             16416 1044    79    7    1    0                7

                              1008   72     2    2
```

```
                               ----------
98692840 3891940 169621 8261  456   29

 6169520  243320  10598  518   28    2
 ---------------
 4934642  194597

  308476   12166                                                    15

  307502   12107                                                     1

     974      59
```

The differences between this algorithm and the one for regular continued fractions (all 1s in the numerators):

1. The list of numerators is written down just above the denominator terms.
2. Each element is computed from the two to its right by multiplying the nearer one by the denominator term above it, in the next to top line--then multiplying the further (rightmost) element by the numerator term above it (in the top line)--then finally adding the two products. (When the numerators are all 1, this is the same as the regular algorithm.) Thus, in the pi conversion, 555 = 9*51 + 16*6.
3. The determinant is not preserved, and it is possible for a 2 by 2 pattern to have a gcd of all four elements greater than 1. This gcd will always divide the last numerator used. In the pi conversion, this gcd exceeded 1 three times, successively reaching 4, 36, and 20. In an effort to keep the elements small, these gcds were divided out each time. The reducible matrices were underlined and the reduced ones were then copied directly beneath.
4. You soon need scratch paper or a calculator.

The output steps are the same as for Euclid's algorithm.

The regular continued fraction for pi is particularly hard to get out of any process, due to the difficulty in deciding whether the third term is going to be 15 or 16. (The value of pi with its first two terms gone is 15.997... .) These problems are due to the particularly large term which will follow the 1--we can already tell it will be around 300 from looking at the last matrix. This is also what makes

```
                                  355
        3 7 15 1  =  3 7 16  =  ---  =  3.1415929...
                                  113
```

such a good approximation to pi.

A partial remedy to this "constipation" problem is simply to guess what a pending output term will be, relying on the process to correct itself later. The corrections, if necessary, will take the form of negative terms and possibly 0. These can be "cleaned up" by running the regular Euclidean process starting with the identity function. In the case of the pi conversion, the pattern

```
        8261  456

         518   28
```

tells us that the next term is between 15.9 and 16.3 so we could (incorrectly) guess that the next term was 16:

```
    . . .         100    81    64    49    36    25    16    9    4    1   (1)
             21    19    17    15    13    11     9     7    5    3    1

                                                               12    4    0    4

                                                           24    4    1    1    0    3
```

```
                                                    4   0   1
                                                   -------
                                        555   51    6   1

                            16416 1044   79    7    1   0               7

                             1008   72    2    2
                             ----------
                        8261  456   29

 6169520  243320  10598  518   28    2                                16

   -1948   -1180     53  -27    8
  --------------
  308476   12166

    -974     -59
```

Although this gets us an earlier output, the next three input terms still fail to get us that big term near 300--not until the ingestion of the pair

```
                196
           29
```

do we get our desired -294. After that, five more inputs yield the three small outputs 3, -4, 5. (Small terms contain less information and therefore come out quicker.) This data is based on the assumption of an output whenever the nearest integers to both the upper and lower limits are equal.

Zero and Negative Terms

Converting the preceding result to all-positive form, we use the identity function:

```
    . . .   5     -4    3  -294  16   7   3

                                 22   3   1   0

                            113   7   1   0   1   3

              -14093  -4703  16   1   0           7

                -881   -294   1   0              15

                -878   -293                       1

                  -3     -1                     292

       33   7     -2     -1                       1

       19   4     -1      0                       1

       14   3                                     1

        5   1                                     2

        4   1                                     1

        1   0
```

which is correct as far as it goes. The careful reader may wonder at the seemingly premature input of the term 5 to the matrix

```
        7   -2     7 y - 2
                = -------
        4   -1     4 y - 1
```

which seems to say "between 7/4 and 2", thus foreordaining an output of 1. Beware denominator elements of opposite sign! y between 0 and oo actually says "OUTSIDE 7/4 and 2", with a singularity at y = 1/4. y must be outside 0 and 1/3 to assure an output of 1 (as was the case).

This raises the question of just what are reasonable assumptions about the range of y when we are dealing with an admittedly messed up continued fraction. The answer is that there are none, at least without some very special preprocessing of the input sequence.

The problem is mainly with 0. The sequence ... a 0 b ... is equivalent to the single term ... a+b ..., since if p and q were any adjacent elements of an input process,

$$\begin{array}{ccccc} \ldots & a+b & \ldots & & \\ (a+b)p+q & p & q & & \end{array} \quad == \quad \begin{array}{cccccc} \ldots & b & 0 & a & \ldots & \\ (a+b)p+q & p & ap+q & p & q & \end{array}$$

i.e. the last two adjacent elements are in the same state either way.

This seemingly innocent fact explains why the addition of an initial 0 term is equivalent to the deletion (when possible) of one:

$$0\ 0\ x\ \ldots\ =\ 0+x\ \ldots\ =\ x\ \ldots$$

It also partly explains the funny preambles on certain "linear Hurwitz" numbers:

$$e\ =\ 2\ (1\ \ 2k+2\ \ 1)\ =\ 1\ 0\ 1\ (1\ \ 2k+2\ \ 1)\ =\ (1\ \ 2k\ \ 1)$$

$$\frac{4}{e}\ =\ 1\ 2\ 8\ 3\ (1\ 1\ 1\ \ k+1\ \ 7\ 1\ \ k+1\ \ 2)$$

$$=\ 1\ 2\ 1\ 0\ 7\ 1\ 0\ 2\ (1\ 1\ 1\ \ k+1\ \ 7\ 1\ \ k+1\ \ 2)$$

$$=\ 1\ 2\ (1\ k\ 7\ 1\ k\ 2\ 1\ 1)\ .$$

$$\mathrm{sqrt}\left(\frac{17}{10}\right)\ =\ 1\ (3\ 3\ 2)\ =\ (1\ 3\ 3\ 1\ 0)$$

(Hurwitz numbers are those which can be written in this parenthesis notation using polynomials in k. Hurwitz's theorem states that this property is preserved by homographic functions with rational coefficients. Square roots of rationals are all of the form

$$a\ (b\ c\ \ldots\ c\ b\ 2a)\ =\ (a\ b\ c\ \ldots\ c\ b\ a\ 0)\ .$$

More on this later.)

The mischief comes with sequences like

$$\ldots\ 1\ 2\ 3\ 4\ 5\ 0\ -5\ -4\ -3\ -2\ -1\ \ldots\ =\ \ldots\ 0\ \ldots$$

wherein it seems to be saying something and then takes it all back. This allows a peculiar but complete reversibility of continued fraction computations--by merely inputing or outputing 0 and then several negated terms in reverse order, the computation can back up to any previous state, but unless the maximum length of these "retraction palindromes" can be bounded in advance, there is no reliable way to collapse them out with a sequential process. Even further confusion can be introduced with several applications of the rule:

$$-a\ -b\ -c\ -d\ \ldots\ =\ -a-1\ \ 1\ \ b-1\ \ c\ \ d\ \ldots$$

In practical situations, however, you really can avoid these problems, and the only other nonsense sequence to watch out for is

$$\ldots\ -1\ \ 1\ -1\ \ 1\ -1\ \ 1\ \ \ldots$$

But these can be detected when they begin, whereupon you can shut of output until they stop. You can also simply discard three pairs at a time, since the only effect is to negate the whole state matrix:

```
        . . .  1   -1   1  -1   1   -1  . . .

           -p  -q  q-p  -p   q  q-p   p  q   .
```

Increasing the Convergence Rate of Continued Fractions

Reexpressing Series as Continued Fractions

R notation

Conversion to Decimal

```
              . . .    1  15   7   3

                          22   3   1   0

                           7   1   0   1   3
                          ******
                     150  10   0

                     106   7   1           1
                     *******
                     440  30

                     106   7               4
                     *******
                180  160  20

                113  106   7               1
                ********
                670  540

                113  106
```

That is, just follow the recipe for the homographic function of one argument, except on output, you multiply by 10 after the subtraction, instead of reciprocating. On paper, this necessitates recopying the denominators, which resembles the outputing of 0. Thus, a decimal fraction can be thought of as a continued fraction with two terms for every digit. The denominators are the decimal digits alternated with 0s, while the numerators are alternately 1 and 10. On output, the gcd of the matrix can be multiplied by a divisor of 10. This can be detected simply by maintaining modulo 10 versions of the two denominator coefficients. In the special case that the input continued fraction is regular (as above), only a finite number of such reductions can occur, corresponding to the total number of factors of 2 and 5 that the initial denominator coefficients shared in common. Thus, although there is little reduction to be gained in the regular case, there is also little effort-- you need only count the 2s and 5s in the gcd of the initial denominator terms, and cancel out at most one of each with each output multiplier of 10.

A curiosity worth noting is that when this decimal (or especially octal) conversion is performed on the nonregular fraction for arctan 1 (as in the section on nonregular fractions), the number of reductions by 2 depends drastically upon the original denominator coefficients. If they are 0 and 1, for instance, there will be four times as many cancellable powers of 2 than if they are 1 and 0. Thus, by this method, 1/pi is significantly easier to calculate than pi. This fact may be connected with the observation that infinite series for 1/pi seem to be simpler and more rapidly convergent than series for pi.

Conversion From Decimal

is immediate, since, for instance

```
                                    1
     pi  =  3.141592...  =  3 + ---------
                                       10
                                0 + ---------
                                            1
                                    1 + ---------
                                               10
                                        0 + ---------
                                                    1
                                            4 + ---------
                   1                                   10
         = 3 + ----------                       0 + ---------
                      1000                                  1
               0 + ----------                       1 + ---------
                            1                                  10
                  141 + ----------                      0 + ---------
                                1000                                1
                        592 + ---------                     5 + ---------
                                                                       10
                                    . . .                       0 + ---------
                                                                            1
                                                                    9 + --------
-

                                                                           . . .
```

and we already know how to deal with non-regular continued fractions.

Conflicting Notations

oo
matrices
left to right.

Approximations

Comparison rule: If we call the integer part of a continued fraction the 0th term, then we can say that the a (regular) continued fraction is an increasing function of its even numbered terms, and a decreasing function of its odd numbered terms. Thus, to compare two continued fractions, compare the terms at the first position where they differ, then reverse the decision if this position is odd numbered. If one continued fraction terminates before differeing from the other, regard the shorter one to be suffixed by an infinite term.

The comparison rule can also follow from the rule for subtracting two continued fractions with zero or more initial terms in common. If

```
        w  =  a b c ... p x
and
        y  =  a b c ... p z ,
```

where x and z are the tails of the two fractions, then

```
              Nx + n             Nz + n           N   n
        w  =  ------  and  y  =  ------ ,  where          is the
              Dx + d             Dz + d           D   d
```

matrix resulting from the input of a b c ... p to the identity matrix

```
                1   0
```

0 1 .

Then

$$w - y = \frac{u\,(x - z)}{(Dx + d)\,(Dz + d)}$$

where u is the determinant Nd - nD = 1 or -1 respective of whether there was an even or odd number of inputs to th matrix. Note that this expression is independent of N and n, so we need only compute the bottom row of the matrix. But the bottom row is what you would get by dropping the original input term, the computing the top row. We thus save another step. (If two continued fractions start with the same term, it is clear that their difference is independent of the value of that term.)

Simplest intervening rational: In a recent popularity poll, parents were asked what they thought of the idea of teaching schoolchildren continued fractions instead of long division. Sixty nine percent of those responding though it was a communist plot. What is the smallest number of people who could have been polled?

Presumably, by 69% the pollsters didn't mean exactly 69 of every 100 but rather some fraction p/q which is at least .685 but less than .695. Our problem is to find in the half-open interval [.685, .695) the rational with smallest q. (A half-open interval contains its left endpoint but not its right one.)

If p/q and r/s are distinct nonnegative rationals in lowest terms, we will say that p/q is simpler than r/s if p<=r and q<=s. It may be that of p/q and r/s, neither is simpler than the other. In this case, however, there is always some rational numerically between them which is simpler than either. (E.g., 11/8 is between and simpler than 11/10 and 13/8. 11/8 is the minimum of the numerators over the minimum of the denominators.) It follows that there is a simplest rational in every finite interval, since there can only be a finite number, pq, of rationals simpler than any rational p/q. If our intervals can include oo, we shall treat it as if it were 1/0, thus defining oo to be simpler than any rational besides 0 (i.e., 0/1). The motivation for this is related to oo having the empty continued fraction. Now we have defined the simplest rational in any interval with nonnegative endpoints which does not include both 0 and oo. We leave it to the philosophers to determine which of these two numbers is simplest.

The pollster problem now becomes: what is the denominator of the simplest rational in [.685, .695) ? This is easy to solve if we first determine that the continued fractions of the two endpoints are

    .685 = 0 1 2 3+

and

    .695 = 0 1 2 5+

through the first term where they differ. By 3+ we mean some number greater than 3 but less than 4, and similarly for 5+. From this comparison rule, all of the numbers in the interval have continued fractions = 0 1 2 x, where x is in the interval (3+, 5+], what ever those two numbers happen to be. The simplest number in this interval is 4. The simplest rational in [.685, .695) is therefore the number whose continued fraction is

    0 1 2 4 = 9/13 .(692307)

so as few as 13 people may have been polled, provided that they all responded. This rationale is amplified on the page after next.

(I can't resist pointing out that dividing 13 into 9 is a great example of the tail chasing trick mentioned on page 1: after producing the digits 6 and 9, the remainder is 3, which is 1/3 of the initial "remainder" (namely, the dividend) 9. This means that we can compute the rest of the quotient digits 230769... simply by dividing 3 into 692307... .)

Of course, since we really only wanted the denominator 13, we could have skipped the computation of the numerator 9:

```
            4   2   1   0
       13    3   1   1   0   1 ,
```

(relying on the input process to produce lowest terms.) But then if we wanted to check the answer, we would have to multiply 13 by .69 and see if the result was very near an integer.

The computation of two nearby continued fractions can be streamlined considerably, if we do not wish to go much beyond where they disagree. Begin an ordinary Euclid output process on one number (for variety, we choose the larger), while keeping track of the separation interval, as we did with the gas constant. For .695/1 and .685/1,

```
        .695  -.010
       1.000   .000   0
        .695  -.010   1
        .305  +.010   2
        .085  -.030   3
        .050  +.100
```

stopping when the magnitude of the interval width exceeds the last remainder. At this stage, the last term would have been different, had we used the other endpoint. But we can easily switch over to computing the other endpoint by adding in the interval widths on whichever two consecutive lines we wish:

```
        .695  -.010
       1.000   .000   0
        .695  -.010   1
        .305  +.010   2   .315
        .085  -.030   3   .055  5
        .050  +.100       .040
```

Since both contnued fractions were infinished when we stopped them, we have shown, with very little manipulative effort, that

```
        .695  =  0 1 2 3+
and
        .685  =  0 1 2 5+
```

as required.

Fact (theorem): If A and B are positive rationals, with A simpler than B, then C + 1/A is simpler than C + 1/B, where C is a nonnegative integer.

But C + 1/A and C + 1/B are what you get by prefixing the term C to the continued fractions for A and B. This means that in determining which of two (terminating) continued fractions represents the simpler rational, we can ignore any initial sequence of terms that they have in common. The continued fraction of the simplest rational included in an interval consists of this common initial term sequence, to which is appended the continued fraction of the simplest rational in the interval determined by erasing the common initial sequence from the original endpoints.

We characterize an interval as a pair of endpoints in [0, oo]. Associated with each endpoint is a flag saying whether or not the endpoint is included in the interval. When we modify the continued fraction of an endpoint to produce a new endpoint, we will be careful not to modify this flag.

Recipe #1 for the simplest included rational: Write as continued fractions the two endpoints of the interval in question, stopping at least one term beyond the one where they first disagree, except that if one of them terminates before disagreeing, suffix to it a term of oo. Set aside whatever initial sequence of terms they have in common. These terms will be the initial ones of the simplest included rational, as well as all of the other numbers in the interval. Now we need only find the simplest rational in an interval whose endpoints have continued fractions which start with different terms. (If we have set aside all of the terms of one endpoint, we are left with an explicit oo.) The only way this new interval can fail to contain oo or at least one integer is if the upper endpoint is itself an exact integer, but is excluded by virtue of being the tail of an endpoint which was excluded in the original problem. It must also be the case that the lower endpoint has an initial term 1 less than the upper one. The easiest thing to do in this case is to rewrite the single term which is the upper endpoint as the previous integer followed by a 1, e.g., 3 1 instead of 4. Then we again have two continued fractions which start with the same term, and can proceed with the recipe.

Recipe #2 for simplest included rational: If the interval contains oo, return the answer oo. IF the interval contains any integers, return the smallest one. Otherwise let g be the greatest integer in the smaller endpoint. Return g + the reciprocal of the simplest rational in the inttefval determined by reciprocating the fraction parts of the original endpoints.

The reason Recipe #2 sounds easier is that it avoids the question of how to do the arithmetic. When it comes down to doing the work, Recipe #1 will save you plenty.

Here is an example which illustrates the tricky points. A sportscaster remarks that Joe diMolisho batted over .312 last year. Unfortunately, the sportscaster is notoriously pro diMolisho, so you can bet theat if Joe batted as much as .3125, his friend in the booth would have said "Joe batted .313 last year". At least we know Joe saw a fair amount of action, by determining the simplest rational in the open interval (.312, .3125) (both endpoints excluded):

```
         .3120  +.0005
        1.000    .0000  0
         .3120  +.0005  3
         .0640  -.0015  4
         .0560  +.0065  1   .0625
         .0080  -.0080  7   .0000  oo
         .0000  +.0625
```

thus establishing that

```
        .312   =  0 3 4 1 7
and
        .3125  =  0 3 4 1 oo  =  0 3 5 .
```

Notice that our streamlined algorithm conveniently produced the continued fraction for .3125 in just the (nonstandard) form we needed for the recipe. We have only to find the simplest rational in (7, oo), which is 8 because the oo is not in the interval. So, dMolisho's simplest possible performance ratio is

```
        0 3 4 1 8  = 44/141  =  .312051...
```

indicating at least 141 at bats.

Rounding rule: When you discard the tail of a continued fraction, you effectively subtract from the last term retained the reciprocal of the quantity represented by the tail. This reciprocal is greater than 1/2 iff the first term of the tail is 1, indicating that the last retained term should be incremeted just when the first discarded term is 1. Another way to look at it is that a final 1 can always be combined into the preceding term, so why truncate just before a 1 when truncating just after it will give a more accurate estimate with the same number of terms?

Best truncations: Whether or not you round, a truncated continued fraction has the property of being the closest rational approximation to the untruncated number, subject to having such a small numerator. (No simpler rational comes as close.) The only other rationals with this property can all be formed by reducing the last term of the truncated fraction by up to 50%. For example, after 355/113, what is the next better rational approximation to pi?

```
           pi   = 3 7 15 1 292 1 1 1 2 1 . . .
```

and

355/113 = 3 7 15 1

so 103993/33102 = 3 7 15 1 292 is better than 355/113 (and by the rounding rule, 104348/33215 = 3 7 15 1 293 is better still), but are there any approximations of intermediate accuracy and simplicity? Indeed, reducing the terminal 292 to any integer greater than 292/2 = 146 will produce intermediate approximations, while the approximation 51808/16491 = 3 7 15 1 145 is actually worse than 355/113. To test the borderline case of 52141/16604 = 3 7 15 1 146, we perform the following simple but mysterious ritual: write pi as

3 7 15 1 146 0 146 1 1 2 1 . . . .

Then, delete the first term and fold the left-hand portion over the 0:

```
146 1 15 7
146 1  1 1 2 1 . . . .
```

Because the upper continued fraction numerically exceeds the lower one (by the comparison rule), 52163/16604 = 3 7 15 1 146 is the next better approximation to pi after 355/113 (!) The improvement, however, is microscopic: less than 2 parts in a billion.

Mathematical explanation: suppose we wish to truncate

$$z = a_0 \; a_1 \; \ldots \; a_i \; a_{i+1} \; a_{i+2} \; \ldots$$

by discarding the terms beginning with term i+2. How far can we reduce term i+1 before it would be better to simply discard it too? Define

$$\frac{N}{D} = a_0 \; a_1 \; \ldots \; a_i \qquad\qquad x = a_{i+1} \; a_{i+2} \; \ldots$$

$$\frac{n}{d} = a_0 \; a_1 \; \ldots \; a_{i-1}$$

$$Nd - Dn = u \quad (u = \pm 1)$$

Then

$$z = \frac{Nx + n}{Dx + d}$$

$$\begin{pmatrix} N & n \\ D & d \end{pmatrix} = \begin{pmatrix} a_0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_1 & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} a_i & 1 \\ 1 & 0 \end{pmatrix}$$

Transposing both sides,

$$\begin{pmatrix} N & D \\ n & d \end{pmatrix} = \begin{pmatrix} a_i & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} a_1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_0 & 1 \\ 1 & 0 \end{pmatrix}$$

implying

$$\frac{D}{d} = a_i \; \ldots \; a_2 \; a_1$$

The error introduced by replacing the tail x by the single term p is

$$\frac{Nx + n}{Dx + d} - \frac{Np + n}{Dp + d} = \frac{u\,(x - p)}{(Dx + d)(Dp + d)}$$

while the error introduced by simply discarding the tail is

N   Nx + n   u

```
   -  -  ------  =  ----------
   D     Dx + d     D (Dx + d)
```

The crossover point is when these two errors are equal, i.e., when

$$p + \frac{d}{D} = x - p$$

or

$$p\ a_i\ \ldots\ a_0\ a_1 = (a_{i+1} - p)\ a_{i+2}\ \ldots$$

so if we think of this truncation as chopping off the continued fraction "in the middle of a term", we have the following peculiar yet simple rule: to test whether our chop has produced the best rational approximation possible with such a small numerator, reverse the sequence of terms that we kept, and discard what was originally the zeroth term. Compare this, as a continued fraction, with the part we chopped off (x-p). If the part we chopped off is greater than or equal to the reversed part, we would have done better to chop off the whole term.

Continued Logarithms

There is a mutation of continued fractions, which I call continued logarithms, which have several advantages over regular continued fractions, especially for computational hardware. As with ordinary continued fractions, each number and subexpression will be a microprocess which describes itself a little at a time, but instead of continued fraction terms, our description language will have only two words, "0" and "1". A 1 means "I was at least 2, so I halved myself". A 0 means "I was between 1 and 2, so I subtracted 1 and reciprocated myself (so now I am > 1)". For example, we compute the continued logarithm of 100/2.54 :

```
11111   100/2.54  ->  50/2.54  ->  25/2.54  ->  25/5.08  ->  25/10.16  ->  25/20.32

0       25/20.32 - 1  =  4.68/20.32

11      20.32/4.68  ->  10.16/4.68  ->  5.08/4.68

0       5.08/4.68 - 1  =  .40/4.68

111     4.68/.40  ->  2.34/.40  ->  1.17/.40  ->  1.17/.80

0       1.17/.80 - 1  =  .37/.80

1       .80/.37  ->  .40/.37

0       .40/.37 - 1  =  .03/.37

111     .37/.03  ->  .37/.06  ->  .37/.12  ->  .37/.24

0       .37/.24 - 1  =  .13/.24

0       .24/.13 - 1  =  .11/.13

0       .13/.11 - 1  =  .02/.11

11      .11/.02  ->  .11/.04  ->  .11/.08

0       .11/.08 - 1  =  .03/.08

1       .08/.03  ->  .04/.03

0       .04/.03 - 1  =  .01/.03
```

```
1       .03/.01  ->  .03/.02

0       .03/.02 - 1 =  .01/.02

1       .02/.01  ->  .01/.01

0       .01/.01 - 1  =  0

        oo
```

so 100/2.54 = 11111011011101011100011010101010 . Alternatively, we could write it as the sequence of lengths of bursts of 1s: 5,2,3,1,3,0,0,2,1,1,1. In the latter representation, each term is the integer part of the log base 2 of the number remaining to be described. As with regular continued fractions, oo is the empty sequence, rationals are the finite sequences, and many (but not all!) quadratic surds have periodic sequences. Unlike regular continued fractions, integers can have long sequences, and Hurwitz numbers seem patternless. The direct expression of a continued logarithm as a nonregular continued fraction:

```
                     5
100         5       2
----   =   2   + ---------
2.54                      2
                  2      2
                 2  + ---------
                               3
                       3      2
                      2  + ---------

                                .
                                 .
                                  .

                                        1
                               1       2
                              2  + ---------
                                        1
                                       2   .
```

That is, each denominator and succeeding numerator must be equal powers of 2.

Why Use Continued Logarithms?

The primary advantage is the conveniently small information parcel. The restriction to integers of regular continued fractions makes them unsuitable for very large and very small numbers. The continued fraction for Avogadro’s number, for example, cannot even be determined to one term, since its integer part contains 23 digits, only 6 of which are known. Also, mechanisms for handling such gigantic terms would have to be built, only to lie dormant throughout most computations, since huge terms are very rare except at the beginning of huge numbers. By contrast, the continued logarithm of Avogadro’s number begins with its binary order of magnitude, and only then begins the description equivalent to the leading digits--a sort of recursive version of scientific notation.

Another problem related to huge terms could be called the .999... problem. Suppose you were using continued fraction arithmetic to multiply sqrt(2) = 1 2 2 2 ... by sqrt(2), but without any assurance that the two numbers are, in fact, identical. This means that at any time one of the input terms might turn out to be something other than 2. Depending upon whether this occurs on an even or odd term, the numerical value of the product will be 2.0000+ or 1.9999+, or, as continued fractions

```
        2 <gigantic term> ...
```

or   1 1 <gigantic term> ...

But until that deviation occurs (maybe never), the first term of the continued fraction is in doubt. A partial solution to this problem is to forcibly egest a 2 when it becomes clear that this module is unduly stuck. If we are wrong, the gigantic term will simply come out negative. What we would like to say now is "regard my next term as infinite until further notice", hoping that we are indeed through. But this is not enough for the functions which depend on our output, to which they will eventually become extremely sensitive. They will need to know "just how close to infinity are you?", but we, faced with an ever growing number with oscillating sign, have no way to answer. We do not even know when we should input more 2s (at least we hope they are 2s). The information-drivenness has broken down.

With continued logarithms, there is no problem at all, if we regard a 1 to mean "my MAGNITUDE was at least 2, so I halved myself". Our function will simply produce 1s as long as the input patterns hold, and a constant stream of 1s is at least as informative to a superior function as any other string. Simply stated, continued logarithms allow us to say that the first digit of infinity is 1. A slight embarrassment could occur if it turned out that one of the inputs was really < sqrt(2), since we have not included in our language a mechanism for negative numbers (in this case it would serve as an expletive). We will see to this later.

The Simple Details

Suppose you are given the integers a, b, c, and d, and wish to compute the homographic function

$$y = \frac{a x + b}{c x + d}$$

with continued logarithms. For each x input of 1, y's value must be preserved by doubling a and c, or if possible, halving b and d. This is because x has halved itself. When x announces that it has reduced itself by 1, add a to b and c to d. When x announces it has reciprocated itself, interchange a with b and c with d. Equally easy is output of y. The knowledge that x is > 1 gives us

$$y \text{ between } \frac{a + b}{c + d} \text{ and } \frac{a}{c},$$

provided c+d and d have the same sign. If both of these quantities are >= 2, y can emit a 1 and halve itself by doubling c and d, or if possible, halving a and b. If the two ratios lie between 1 and 2, y decrements itself by subtracting c from a and d form b, then reciprocates itself by interchanging a with c and b with d and finally announces all this by emitting a 0.

Although these operations are not as nice on paper, they are beautifully suited to binary machines, requiring only shift, add, subtract, exchange, and compare-for-greater.

To illustrate the power and simplicity of this mechanism, we will compute the continued log of sqrt(6) from first principles, by solving

$$y = \frac{6}{y} .$$

Setting up the matrix for 6/y,

```
             0   6

             1   0
```

we test whether y is greater or less than 2 by plugging in y = 2 to get 3. The fact that the value went up instead of down says that y > 2, since by Newton’s method, the average is closer than either. So we outpulse and receive a 1, meaning to halve a and b, then double a and c.

```
             0   3

             2   0    1
```

Now plugging in y = 2 gives 3/4, so y < 2 and we oupulse and receive a 0, which is just like outputing and inputing a term of 1 with regular continued fractions. (In fact the golden ratio, whose continued fraction is all 1s, has continued log all 0s.)

```
             0   3

         2   2   0    10

         1  -2   3
```

Here the assumption y >= 2 fulfills itself, while 1 <= y < 2 will drive

$$\frac{2y + 2}{y - 2}$$

negative. So again we emit and receive a 1 by halving a and b, then doubling a and c.

```
             0   3

         2   1   0    10

         2  -2   3    1
```

Here again y > 2, requiring another 1, but since b is odd, we must double c and d, then also double a and c.

```
             0   3

         4   1   0    10

         8  -4   3    11
```

Here y < 2, ending another 1 burst.

```
             0   3

         4   1   0    10

     4   8  -4   3    110

     1  -4   5
```

Here y must be > 2 (in fact > 4) to avoid chasing the negative root, and this time we can process the 1 by halving a and b, then halving b and d.

```
             0   3

         4   1   0    10

     2   2  -4   3    110
```

1 -2 5 1

But the matrix was in this state right after emitting the first 0, so

sqrt(6) = 10(1101) .

In computational practice, we will need two other words besides 1 and 0 in the language. For negative numbers we could have either "-" for "I negated myself" or "+" for "I incremented myself". For fractions initially < 1, "*" would mean "I doubled myself", the opposite of "1". Also, for finite inputs, we formally require an end-of-string mark, which is logically oo, since the empty continued logarithm, like the empty continued fraction, represents +or- oo. Continued logs can also represent oo as an endless burst of 1s, if it results from nonterminating inputs.

The Continued Logarithm of pi etc.

Architecture

If it is possible to make very long parallel adders, it should be possible to make a high-precision, ultrahigh speed arithmetic processor based on continued logarithms. It would be an extremely parallel device consisting entirely of registers and having no static memory. Such an architecture is feasible because, within a given bihomographic octet of registers, each bit must connect to only five others. Here is why.

Schematically, we can think of our 2 by 2 by 2 matrix as a cube with a register at each vertex. Into this cube flow the two bit streams describing the operands x and y, and out of it flows the bit stream of the answer, z. No matter which of the three possible transactions we perform, the additions, subtractions, comparisons, and exchanges take place between registers joined by the edges of this cube. In fact, we could imagine that within each edge was the control logic for the transaction determined by that edge's direction. Thus each register bit need only talk to its three counterparts in the x, y, and z directions, plus its left and right neighbors (for shifting and carrying).

Instead of wasting time testing which of the three possible transactions is most relevant, we will synchronously and cyclicly input x, input y, and output z on each major cycle. This will simplify the hardware at the cost of diluting the information density of the output stream by a small percentage, due to the occasional retraction of a premature output. Sadly, this will also cripple our automatic error analysis, but such is the price of speed. We could gain even more speed by making output decisons before the carries have settled, since this should introduce only slight further dilution.

After our octet has received about 2n inputs and produced about n outputs, each of our registers will contain about n/4 significant bits. Carry times will grow as log n, so our quotient or product or whatever will have taken time proportional to n log n, like the FFT algorithms, but with a far smaller constant of proportionality.

The four main advantages of this scheme over the FFT schemes are:

1) Simplicity--it is hardly more than a "smart memory", with a minimal form of processor distributed throughout.

2) Speed--we beat the traditional cost factor of dozens or even hundreds for multiprecision, with output bits typically separated by only slightly more than an integer add time. With all of the internal

thrashing, it may waste energy, but not time.

3) Consequently, the crossover point is toward much smaller numbers, thus encompassing more applications.

4) Additional parallelism--we can interconnect networks of these octets which evaluate whole arithmetic expressions in parallel. In fancy models, we could imagine "octet pools" containing allocatable segments of register octets, to be dynamically distributed as register contents grew and shrank. Fancy or not, it should be possible to sustain ultraprecise computations to megabit accuracy, at megabit/second rates, using something not much more complicated than a large semiconductor memory. More conventional processors can not come anywhere near sustaining such an ouput rate since most of their bits are lying dormant in storage for relatively long periods. Even Illiac IV using FFT multiplication can only provide pi in megabit quantities at about 5 kilobits/sec, and only then with prodigious programming effort.

The key idea is that with every bit of storage there be the associated logic to operate on that bit. The continued logarithm formulation restricts the number of data paths to a conceivably practical level.