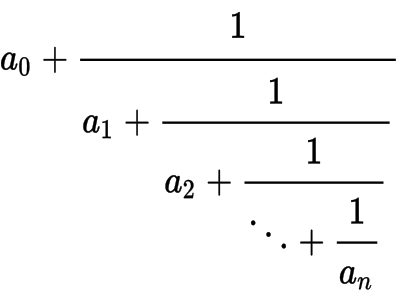
**Continued Fractions**

A continued fraction is a number represented through an iterative series of fractions of the form:

Using this form can show deeper patterns in the structure of numbers, and can be used to form rational approximations of irrational numbers, often with small amounts of computation.

**Alternative Notation**

The more fractions that form a number, the more cumbersome they are to display. This problem leads to another notation for writing continued fractions; a list [a0; a1; a2; … an] with the elements in this particular list corresponding to the picture above.

**Evaluating a continued fraction**

To get the number to its simple form of a single numerator and denominator, evaluate the fraction at the very bottom, and work backwards, using the result of that fraction to evaluate the next one.

**Forming a continued fraction**

Take the number 123/49.   
This can be written as 2.510204… or 2 + 0.510204…  
Taking the reciprocal of 0.510204… we get 1.96, which can be written as 1 + 0.96.  
The reciprocal of 0.96 is 1.041666666 which is 1 + 0.041666666.  
The reciprocal of 0.041666666 is simply 24, which terminates the continued fraction.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Fraction** | **Integer Part** | **Non-Integer Part** | | **Reciprocal of Non-Integer Part** | | **Continued Fraction** |
| 123/49 | 2 | 0.510204 | 25/49 | 1.96 | 49/25 | [2 … ] |
| 49/25 | 1 | 0.96 | 24/25 | 1.041666 | 25/24 | [2; 1 … ] |
| 25/24 | 1 | 0.041666 | 1/24 | 24 | 24 | [2; 1; 1 … ] |
| 24 | 24 | - | - | - | - | [2; 1; 1; 24] |

All rational numbers will eventually terminate through this process and give a continued fraction which can be evaluated. However, irrational numbers will never terminate through this process, and instead give infinite continued fractions. These are incredibly useful for giving approximations to these numbers, with more iterations giving more accuracy. Some notable infinite continued fractions are shown below:

π: [3; 7; 15; 1; 292; 1; 1; 1; 2; 1; 3; 1; 14; 3 … ]

e: [2; 1; 2; 1; 1; 4; 1; 1; 6; 1; 1; 8; 1; 1; 10 … ]

: [1; 2; 2; 2; 2; 2; 2; 2; 2; 2; 2; 2; 2; 2; 2; 2 … ]

φ (Golden Ratio): [1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; … ]

**Rate of Approximations**

Some approximations require more computation that others to converge to the desired result. Having larger values for denominators will cause the fraction to approach more quickly. Famous mathematician Ramanujan worked closely with infinite fractions and found remarkably accurate approximations. Take π4:

[97; 2; 2; 3; 1; 16539 … ]

Notice that the 6th element is abnormally large, which allows a continued   
fraction terminating with 16539 to give a very accurate π4. Taking the 4th   
root of this, a close approximation to π is given:

Now looking at φ we have:

[1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1 … ]

which has the smallest possible values for each denominator, meaning it takes many iterations to get close to the real value. For this reason, the golden ratio is often referred to as the “most irrational” fraction. The table below shows the speed at which pi and phi approach their true numbers, with each bold section representing correct digits.

|  |  |  |
| --- | --- | --- |
| Iterations: | π | Φ |
| 1 | **3**.00000000000000 | **1**.00000000000000 |
| 2 | **3.14**285714285714 | 2.00000000000000 |
| 3 | **3.1415**0943396226 | **1**.50000000000000 |
| 4 | **3.141592**92035398 | **1.6**6666666666666 |
| 5 | **3.141592653**01190 | **1.6**0000000000000 |

**References:**  
<https://plus.maths.org/content/chaos-numberland-secret-life-continued-fractions>  
<https://www.youtube.com/watch?v=CaasbfdJdJg&ab_channel=Mathologer>

**Image Sources:**

http://codegolf.stackexchange.com/questions/93223/simplify-a-continued-fraction

**(\*OCaml code to evaluate continued fraction given a list\*)**

let rec cftodec xs acc = match xs with

| [] -> 0.

| [x] -> x +. acc

| x::xs -> cftodec xs (1. /. (x +. acc));;

let float\_rev xs = List.rev\_map float xs;;

let cf2dec xs = cftodec (float\_rev xs) 0.;;

**(\*OCaml code to create a continued fraction from a decimal (requires previous code. Maximum value of 15 for n due to the accuracy of OCaml float)\*)**

let rec dectocf decimal acc originaldec accuracy =

if((abs\_float ((cf2dec (List.rev acc)) -. (originaldec))) <= accuracy) then List.rev acc

else

let intpart = truncate decimal in

let decpart = decimal -. (float intpart) in

dectocf (1./.decpart) (intpart::acc) originaldec accuracy;;

let rec dec2cf decimal n = dectocf decimal [] decimal (1./.(10. \*\* (float n)));;