Computing with Infinite Data

What infinite data?
How to compute with it?
What for?

Task: Name something infinite!

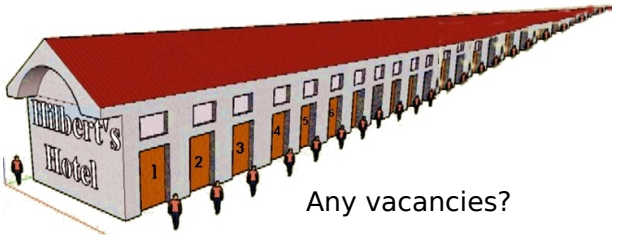
This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 731143.

Aston University
Birmingham

Michal Konečný
m.konecny@aston.ac.uk

Strange but useful

David Hilbert



Any vacancies?

Strange but useful

David Hilbert

Any vacancies?

YES! ... BUT can this be computer-simulated?

Strange but useful

David Hilbert

Any vacancies?

YES! ... BUT can this be computer-simulated?

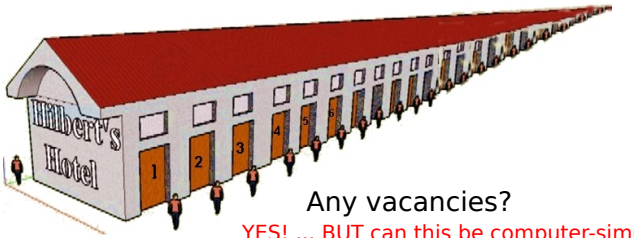
X^2-1 divisible by 11

demo

take 10 sqdiv11 =
[1,10,12,21,23,32,34,43,45,54]

Strange but useful

David Hilbert



Any vacancies?

YES! ... BUT can this be computer-simulated?

X^2-1 divisible by 11

demo

take 10 sqdiv11 =
[1,10,12,21,23,32,34,43,45,54]

shrinking infinite repetition

When does the ball stop bouncing?

Strange but useful

x^2-1 divisible by 11

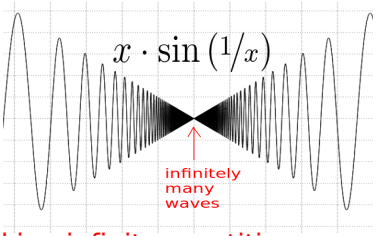
demo

take 10 sqdiv11 =
[1,10,12,21,23,32,34,43,45,54]

David Hilbert

Any vacancies?

YES! ... BUT can this be computer-simulated?



shrinking infinite repetition

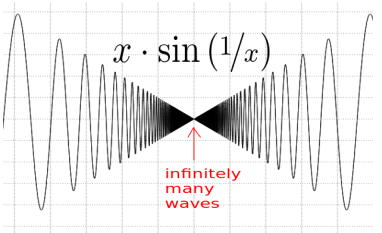
When does the ball stop bouncing?

Strange but useful

David Hilbert

Any vacancies?

YES! ... BUT can this be computer-simulated?



shrinking infinite repetition

When does the ball stop bouncing?

X^2-1 divisible by 11

demo

take 10 sqdiv11 =
[1,10,12,21,23,32,34,43,45,54]

sqrt(s) ~ $\sqrt{s}$

for all s ...

around 4 billion values of s

```
sqrt(s) = sqrtB(10,s)
sqrtB(0, s) = 1
sqrtB(n, s) =
      let x = sqrtB(n-1, s)
      in (x + s/x) / 2
```

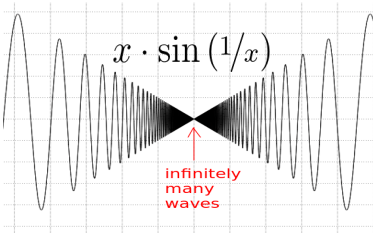
is the program correct?

Strange but useful

David Hilbert

Any vacancies?

YES! ... BUT can this be computer-simulated?



shrinking infinite repetition

When does the ball stop bouncing?

X^2-1 divisible by 11

demo

take 10 sqdiv11 =
[1,10,12,21,23,32,34,43,45,54]

sqrt(s) ~ $\sqrt{s}$

for all s ...

around 4 billion values of s

infinity is easier...

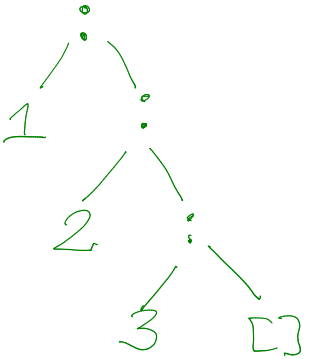
```
sqrt(s) = sqrtB(10,s)
sqrtB(0, s) = 1
sqrtB(n, s) =
      let x = sqrtB(n-1, s)
      in (x + s/x) / 2
```

is the program correct?

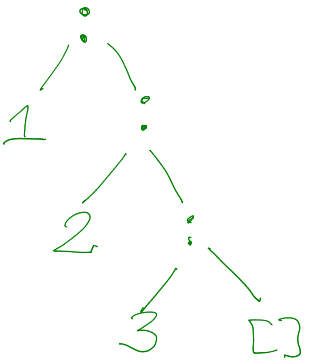
a model with inifinity helps to check this

Inifinite lists

[1,2,3] = 1 : (2 : (3 : []))



Inifinite lists



[1,2,3] = 1 : (2 : (3 : []))

from n = n : (from (n + 1))

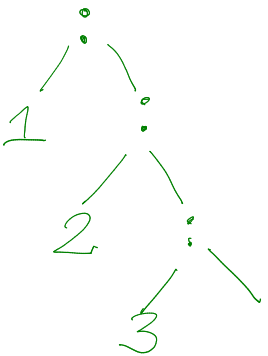
[1..]
  = from 1
  = 1 : (from 2)
  = 1 : (2 : (from 3))
  = 1 : (2 : (3 : (from 4)))
  = ...
  = 1 : (2 : (3 : (4 : ...)))

Inifinite lists

[1,2,3] = 1 : (2 : (3 : []))

from n = n : (from (n + 1))

[1..]
  = from 1
  = 1 : (from 2)
  = 1 : (2 : (from 3))
  = 1 : (2 : (3 : (from 4)))
  = ...
  = 1 : (2 : (3 : (4 : ...)))



take 3 [1..]
  = take 3 (1:(2:(3:(from 4))))
  = 1:(2:(3:[]))
  = [1,2,3]

Very large numbers ≠ ∞

1000! = 1 * 2 * 3 * ... * 999 * 1000 = product [1..1000] =

40238726007709377354370243392300398571937486421071463254379991042993851239862902059204420848696940480047998861019719605863166687299480855890132382966994459099742450408707375991882362772718873251977950595099527612087497546249704360141827809464649629105639388743788648733711918104582578364784997701247663288983595573543251318532395846307555740911426241747434934755342864657661166779739666882029120737914385371958824980812686783837455973174613608537953452422158659320192809087829730843139284440328123155861103697680135730421616874760967587134831202547858932076716913244842623613141250878020800026168315102734182797770478463586817016436502415369139828126481021309276124489635992870511496497541990934222156683257208082133318611681155361583654698404670897560290095053761647584772842188967964624494516076535340819890138544248798495995331910172335555660213945039973628075013783761530712776192684903435262520001588853514733161170210396817592151090778801939317811419454525722386554146106289218796022383897147608850627686296714667469756291123408243920816015378088989396451826324367161676217916890977991190375403127462228998800519544441428201218736174599264295658174662830295557029902432415318161721046583203678690611726015878352075151628422554026517048330422614397428693306169089796848259012545832716822645806652676995865268227280707578139185817888965220816434834482599326604336766017699961283186078838615027946595513115655203609398818061213855860030143569452722420634463179746059468257310379008402443243846565724501440282188525247093519062092902313649327349756551395872055965422874977401141334696271542284586237738753823048386568897646192738381490014076731044664025989949022222176590433990188601856652648506179970235619389701786004081188972991831102117122984590164192106888438712185564612496079872290851929681937238864261483965738229112312502418664935314397013742853192664987533721894069428143411852015801412334482801505139969429015348307764456909907315243327828826986460278986432113908350621709500259738986355427719674282224875758676575234422020757363056949882508796892816275384886339690995982628095612145099487170124451646126037902930912088908694202851064018215439945715680594187274899809425474217358240106367740459574178516082923013535808184009699637252423056085590370062427124341690900415369010593398383577793941097002775347200

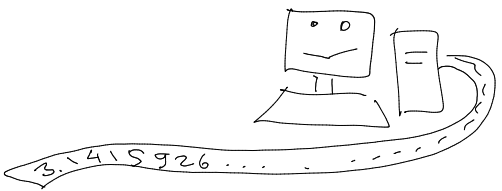
100000! = ...(456574 digits)...

demo

Real numbers are infinite

π = 3.14159265358979323846264338327950288419716939937510582097494459230781
64062862089986280348253421170679821480865132823066470938446095505822317
25359408128481117450284102701938521105559644622948954930381964428810
97566593344612847564823378678316527120190914564856692346034861045432
66482133936072602491412737245870066063155881748815209209628292540917153
64367892590360011330530548820466521384146951941511609433057270365759
59195309218611738193261179310511854807446237996274956735188575272489
12279381830119491298336733624406566430860213949463952247371907021798
...

inifinitely many digits
no "simple" pattern
but a "complex" pattern:
a computer program

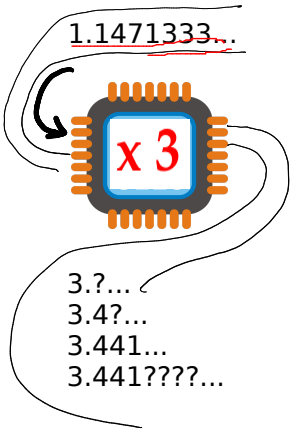



```haskell
{-|
Computes a sequence of approximations of π using binary splitting summation of
Ramanujan's series. See Haible and Papanikolaou 1998.
-}
piRaw :: [Approx]
piRaw = unfoldr f (1, (1, 1, 1, 13591409))
    where as = [13591409,13591409+545140134..]
          bs = ones
          ps = (1:[-(6*n-5)*(2*n-1)*(6*n-1) | n <- [1,2..]])
          qs = (1:[n^3*640320^2*26680 | n <- [1,2..]])
          f (i, (pl, ql, bl, tl)) =
            let i2 = i*2
                (pr, qr, br, tr) = abpq as bs ps qs i i2
                n = 21+47*(i-1)
                x = fromIntegral tl * recipA (setMB n $ fromIntegral (bl*ql))
                x1 = fudge x $ fromDyadicMB n (1:^(-n))
                x2 = boundErrorTermMB $ sqrtA (setMB n 1823176476672000) * recipA x1
            in Just ( x2
                    , (i2, (pl * pr, ql * qr, bl * br, fromIntegral br * qr * tl + fromIntegral bl * pl * tr))
                    )
```

(by Jens Blanck, https://github.com/jensblanck/cdar)

Trippling a real number

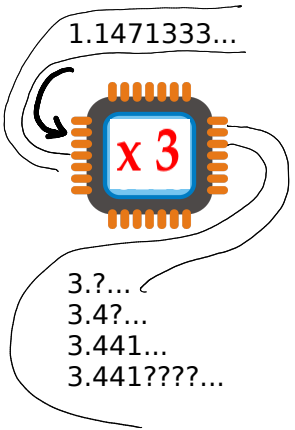
Using streams of digits:

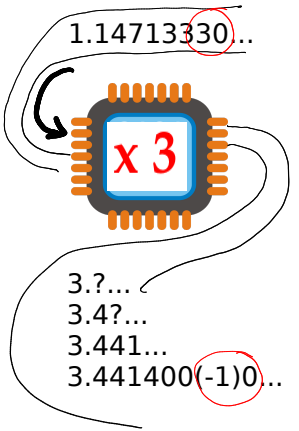


Digits 0..9
Digits 0..1

Trippling a real number

Using streams of digits:



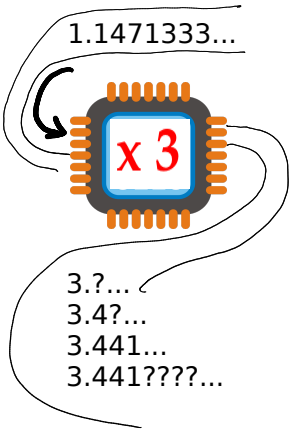


Digits 0..9
Digits 0..1

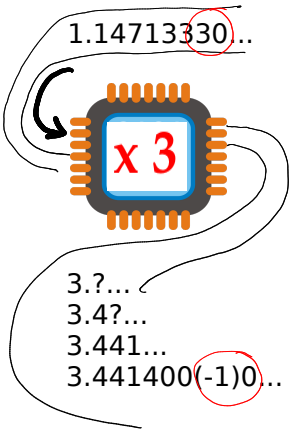
Digits -9..9
Digits -1..1

Trippling a real number

Using streams of digits:

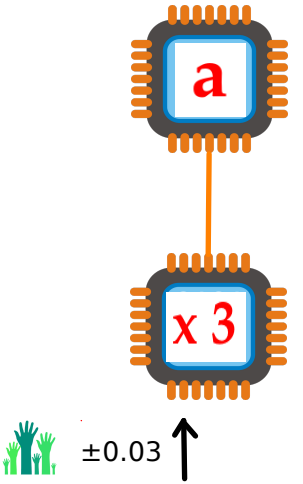


Digits 0..9
Digits 0..1



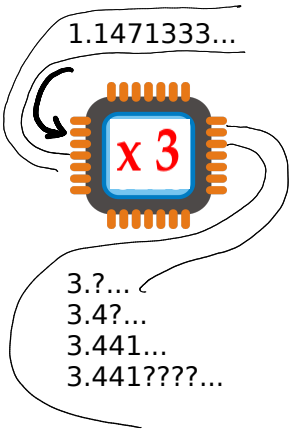
Digits -9..9
Digits -1..1

Using accuracy queries:

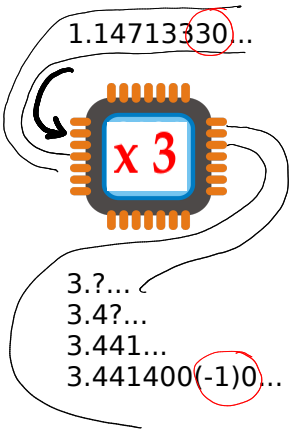


Trippling a real number

Using streams of digits:

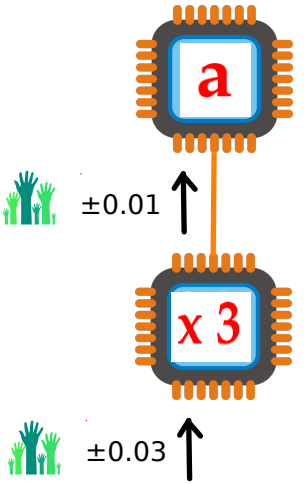


Digits 0..9
Digits 0..1



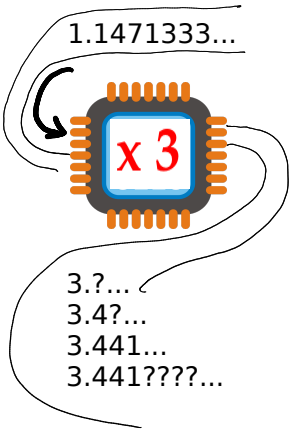
Digits -9..9
Digits -1..1

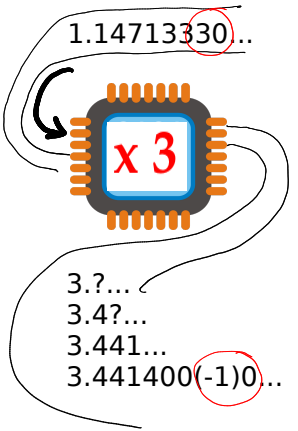
Using accuracy queries:



Trippling a real number

Using streams of digits:

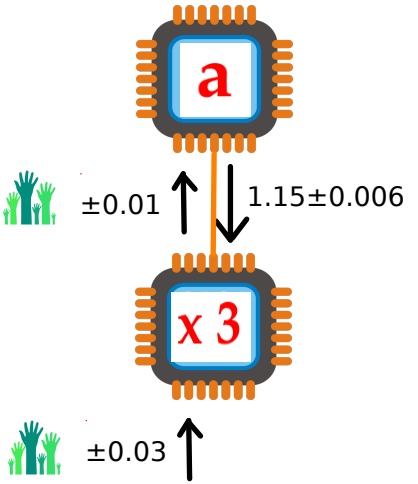




Digits 0..9
Digits 0..1

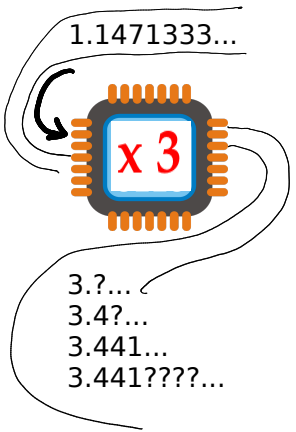
Digits -9..9
Digits -1..1

Using accuracy queries:

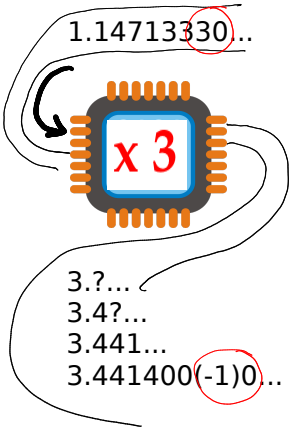


Tripling a real number

Using streams of digits:

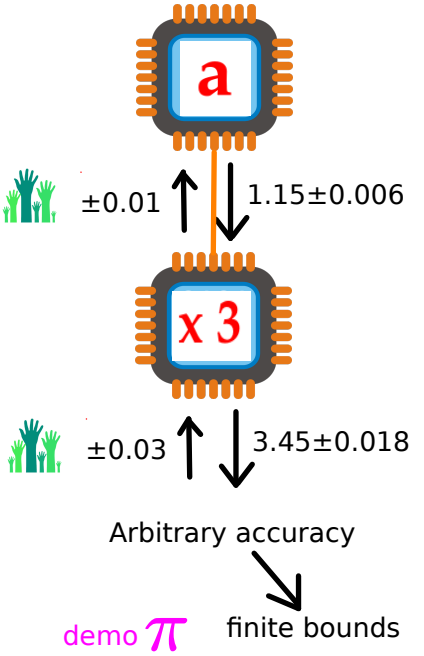


Digits 0..9
Digits 0..1



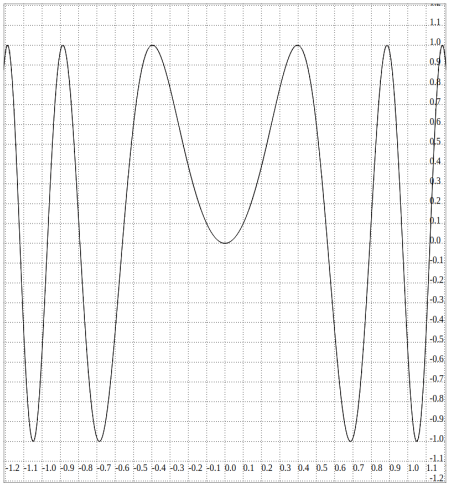
Digits -9..9
Digits -1..1

Using accuracy queries:



Arbitrary accuracy plotting

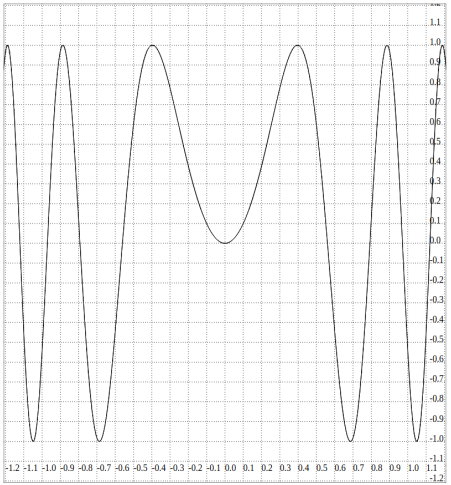
high accuracy



$$\sin(10x^2)$$

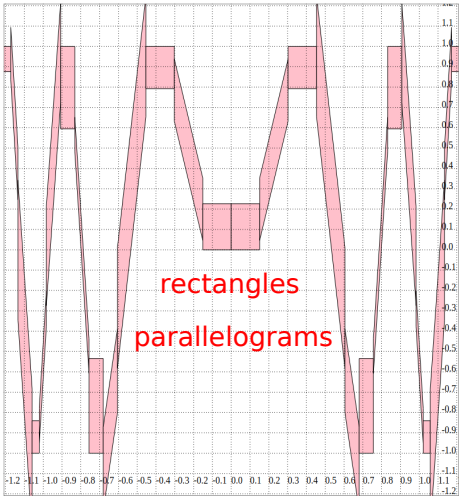
Arbitrary accuracy plotting

high accuracy



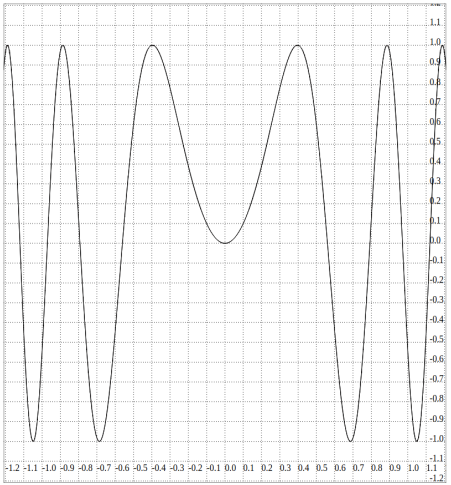
$$\sin(10x^2)$$

low accuracy

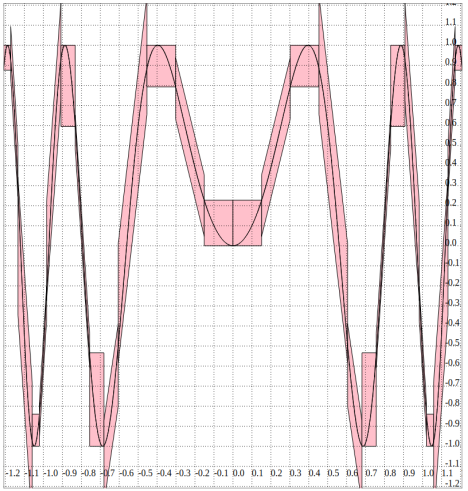


Arbitrary accuracy plotting

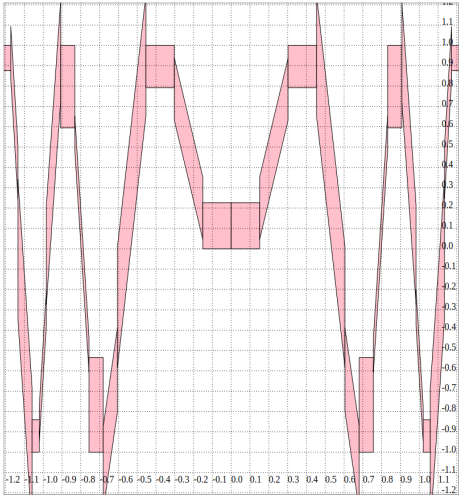
high accuracy



always enclosing exact graph

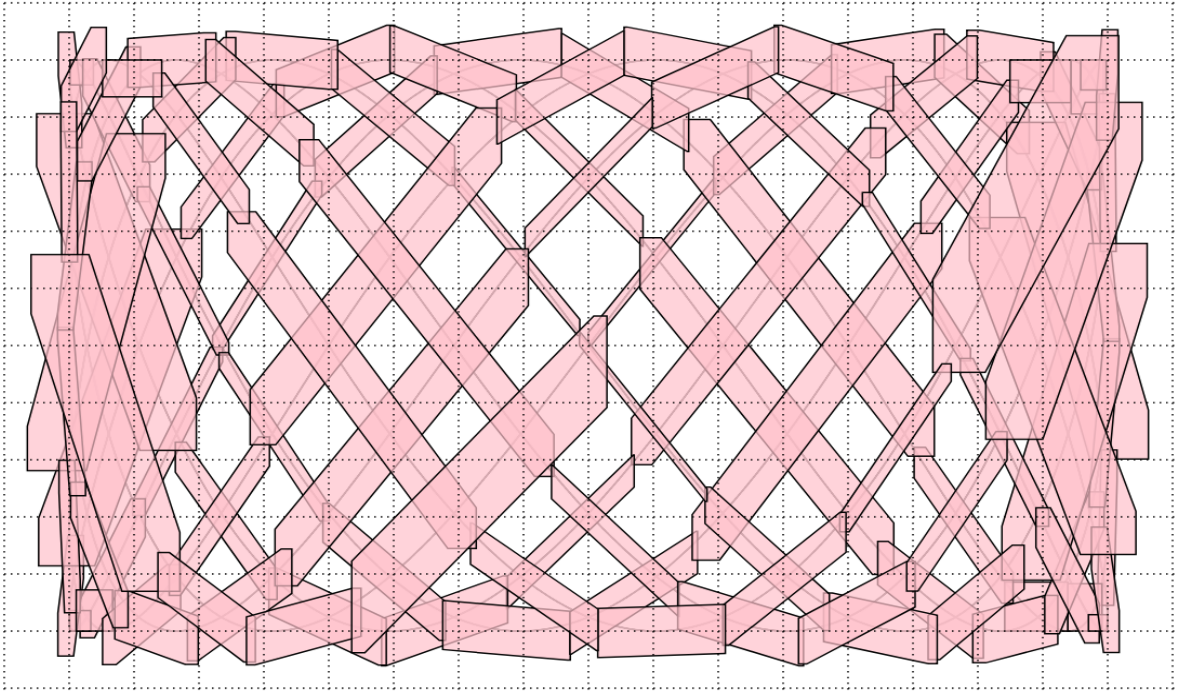


low accuracy

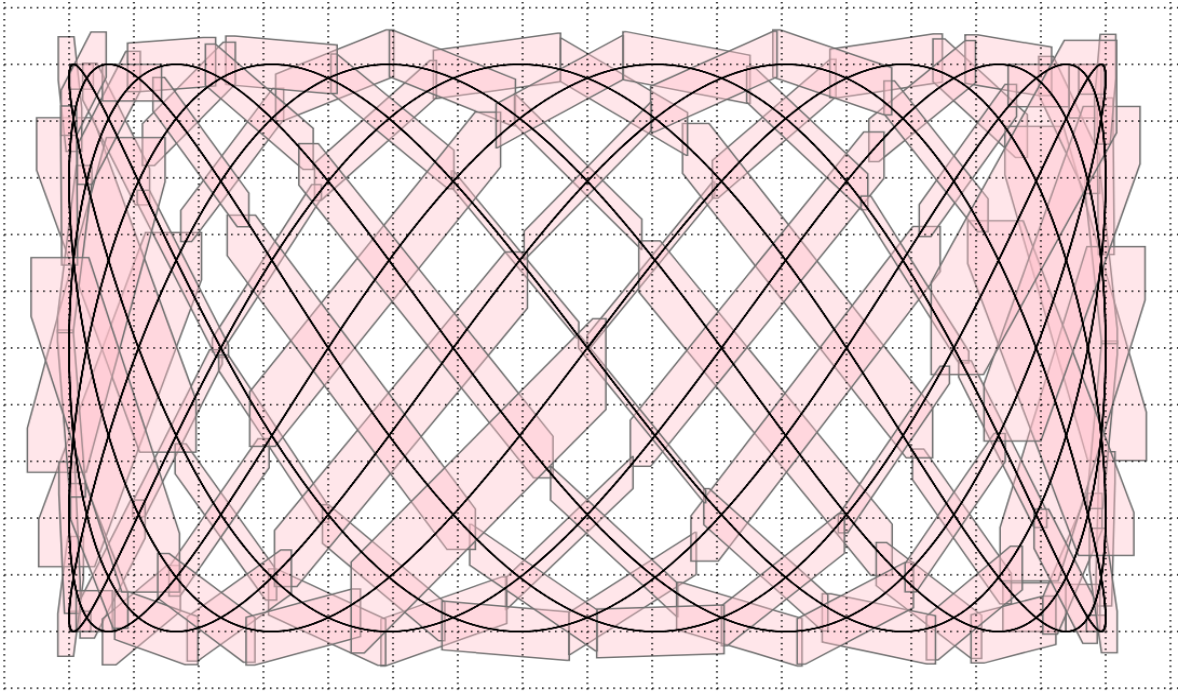


$$\sin(10x^2)$$

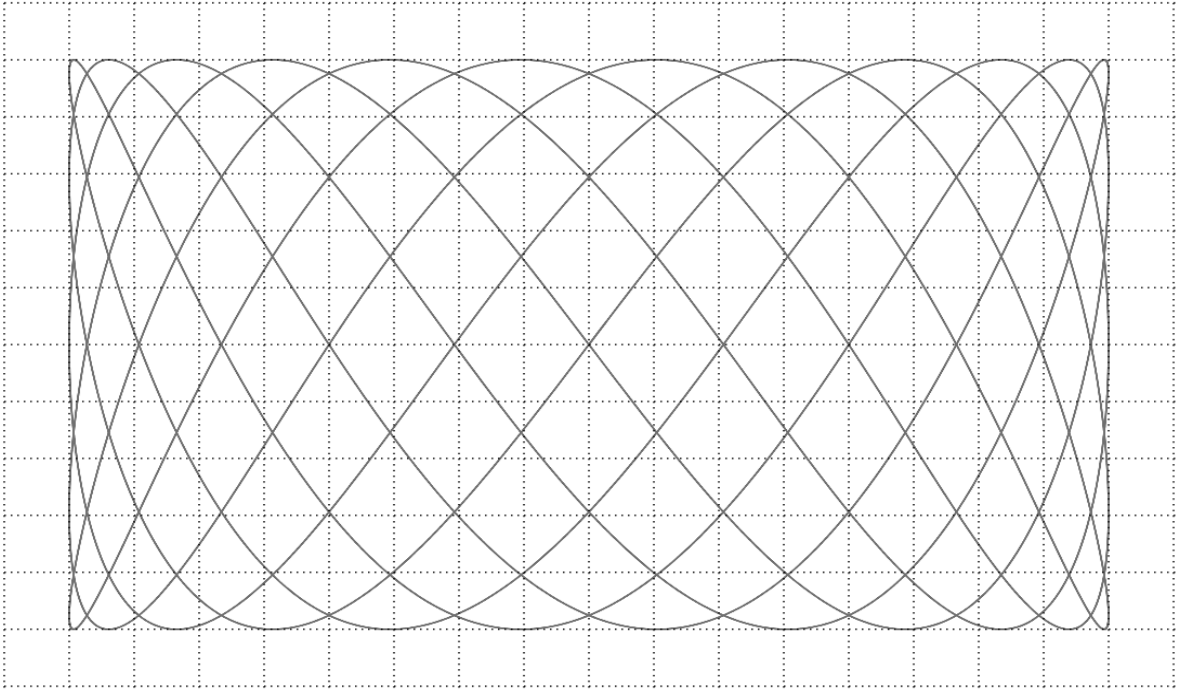
Arbitrary accuracy drawing



Arbitrary accuracy drawing



Arbitrary accuracy drawing



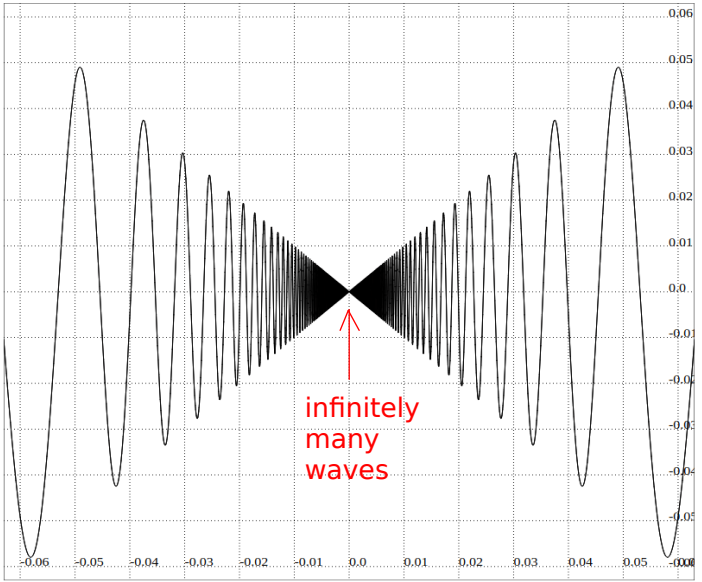
Infinite behaviours

$$x \cdot \sin\left(1/x\right)$$

Function f1(x) = 10
f1(x) accuracy ~ w/ 5 1 <= segments <= 128

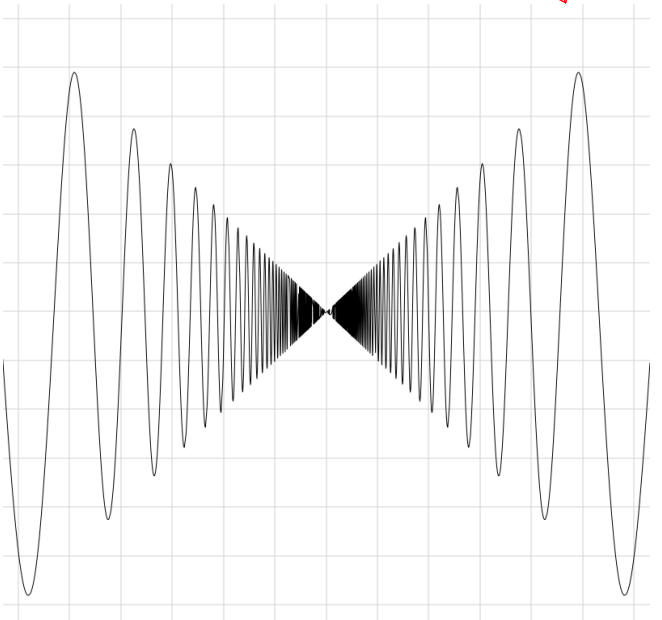
Function f2(x) = x*sin(1/x)
f2(x) accuracy ~ w/ 5000 8 <= segments <= 10000

using exact plotter

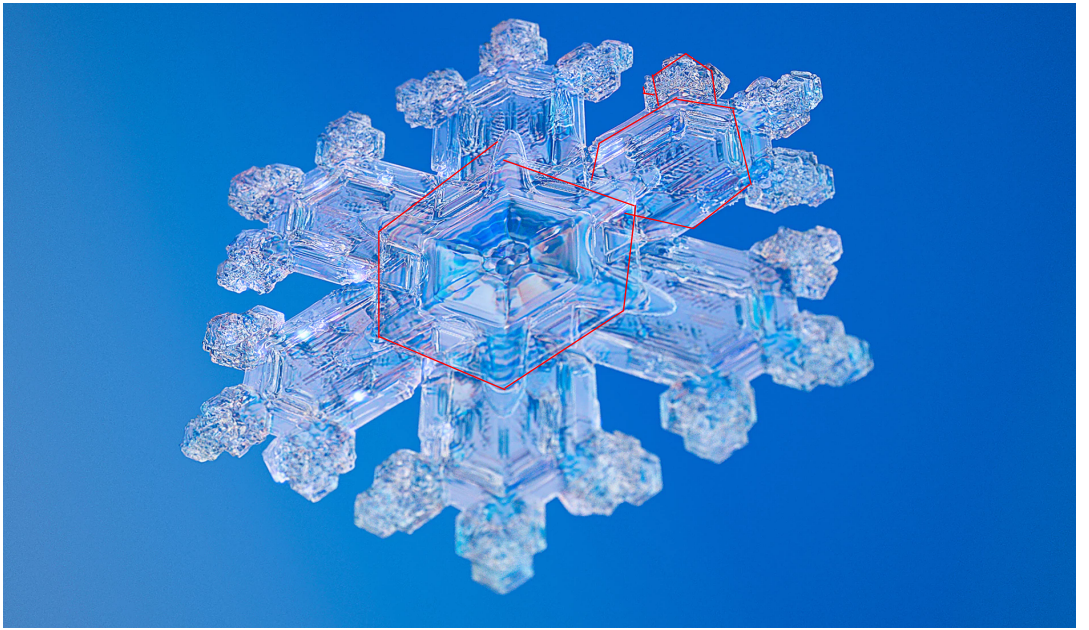


Plot area: -0.0630 <= x <= 0.0630 , -0.0630 <= y <= 0.0630

using conventional plotter
(http://fooplot.com)



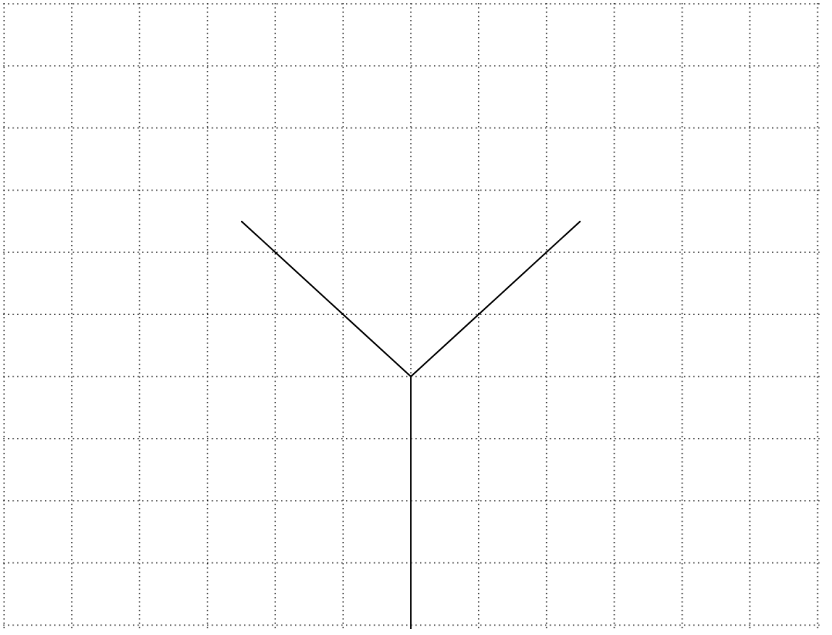
Fractals



Transformations Repeated … infinitely?

(Credit: Ian Cuming / Getty)

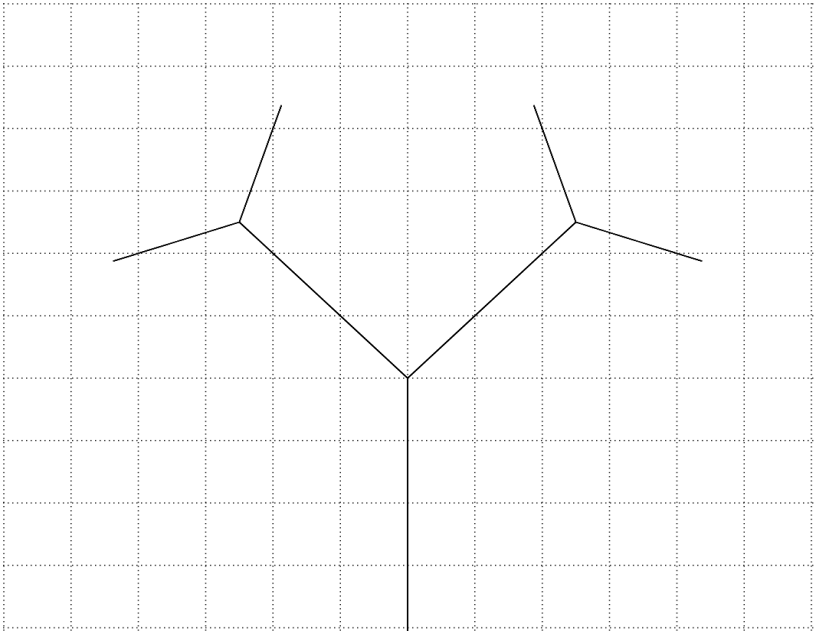
Defining fractals



Transformations

Repeated infinitely

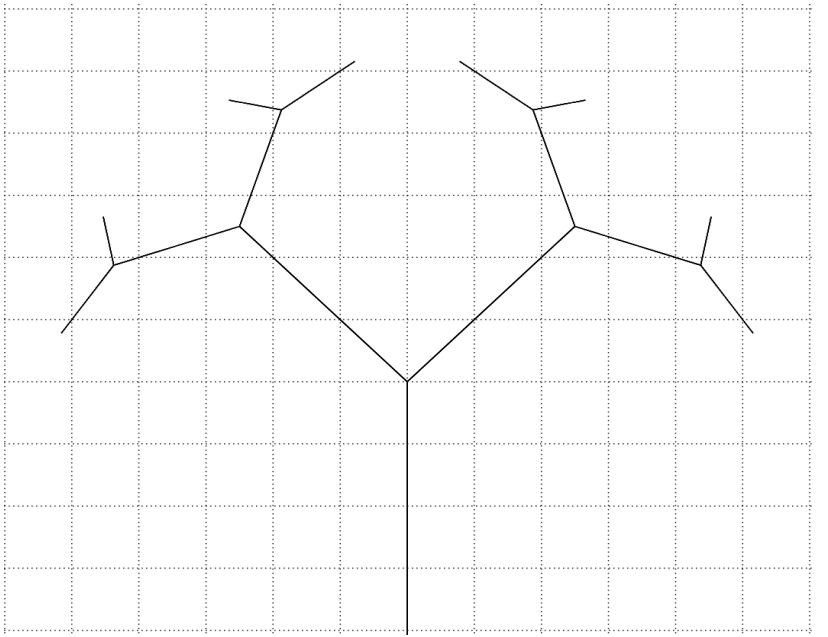
Defining fractals



Transformations

Repeated infinitely

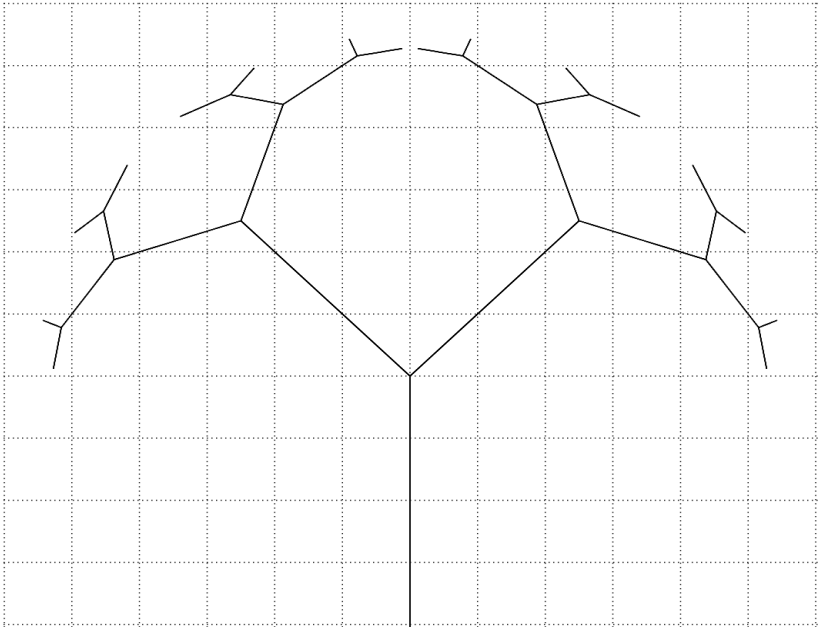
Defining fractals



Transformations

Repeated infinitely

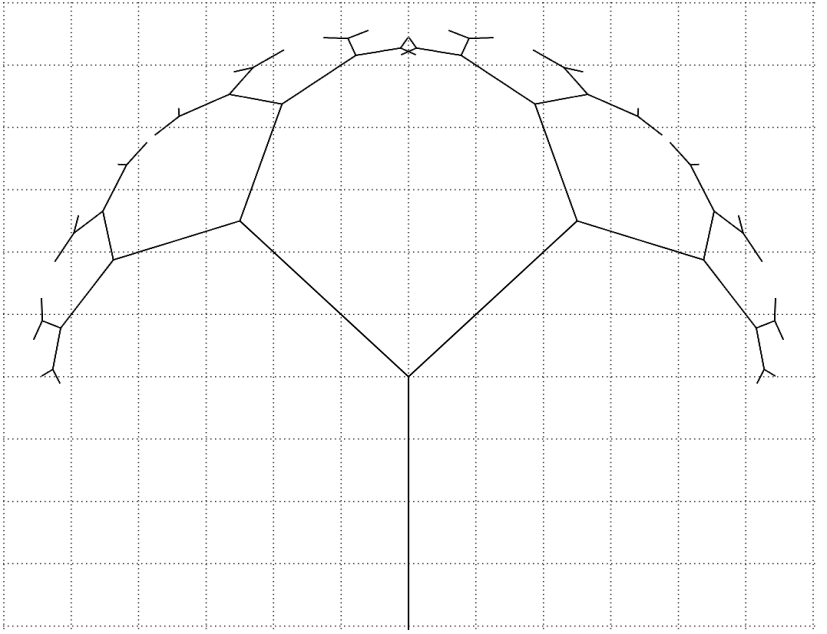
Defining fractals



Transformations

Repeated infinitely

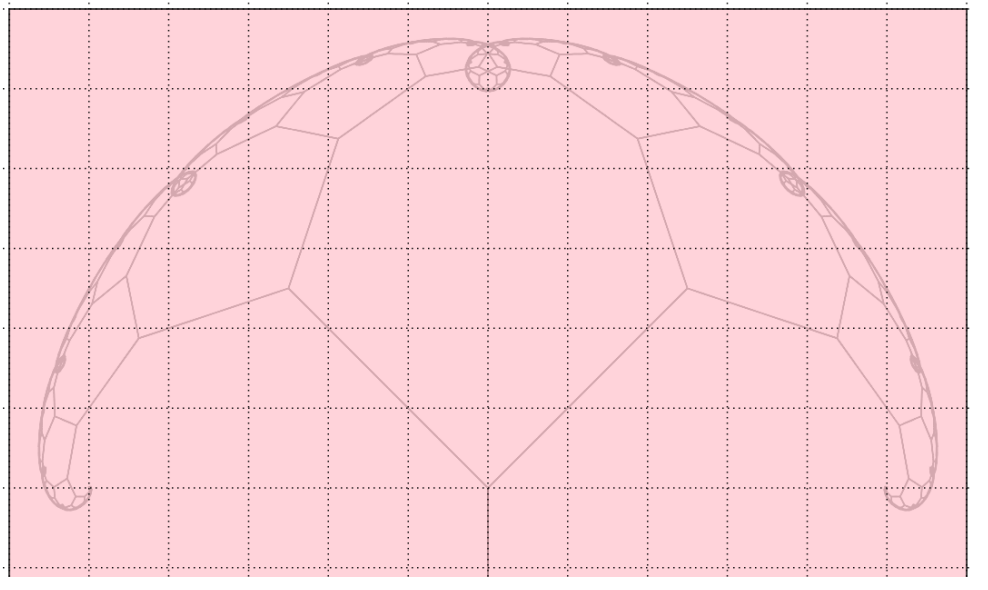
Defining fractals



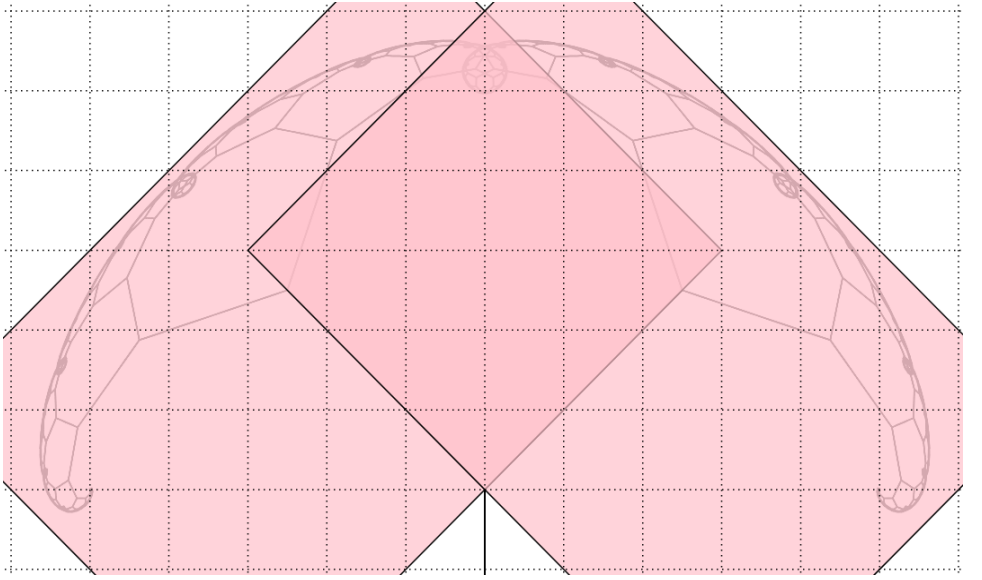
Transformations

Repeated infinitely

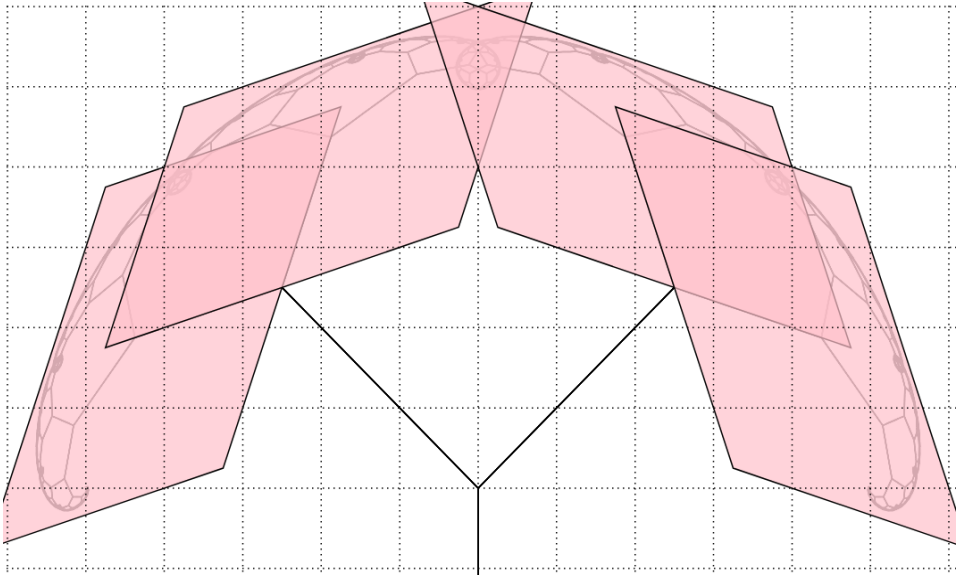
Arbitrary accuracy fractals



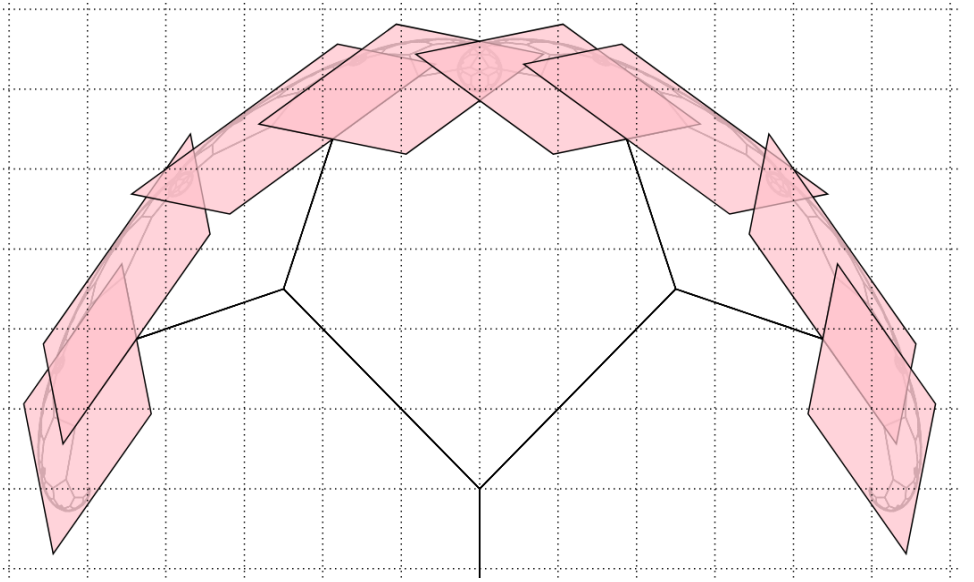
Arbitrary accuracy fractals



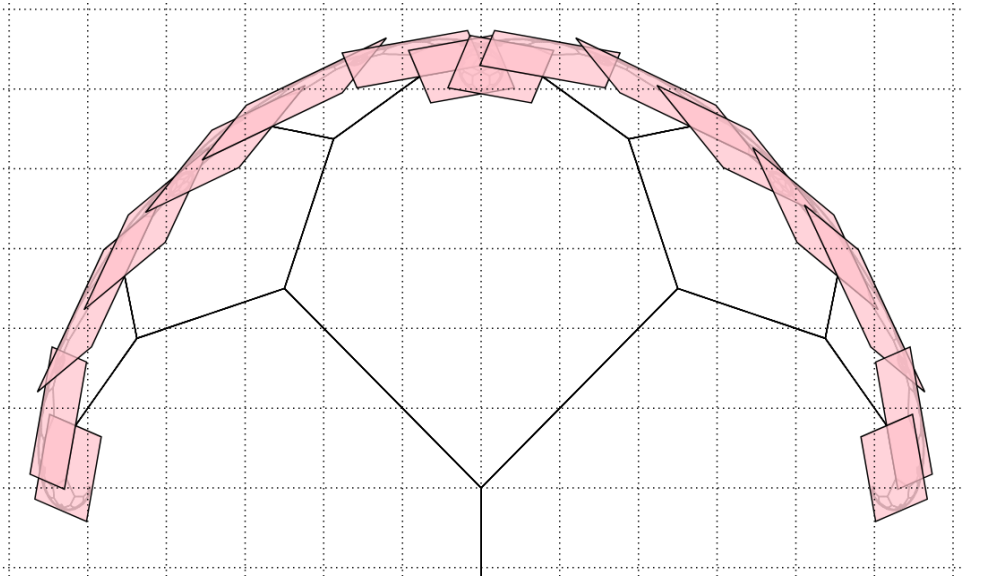
Arbitrary accuracy fractals



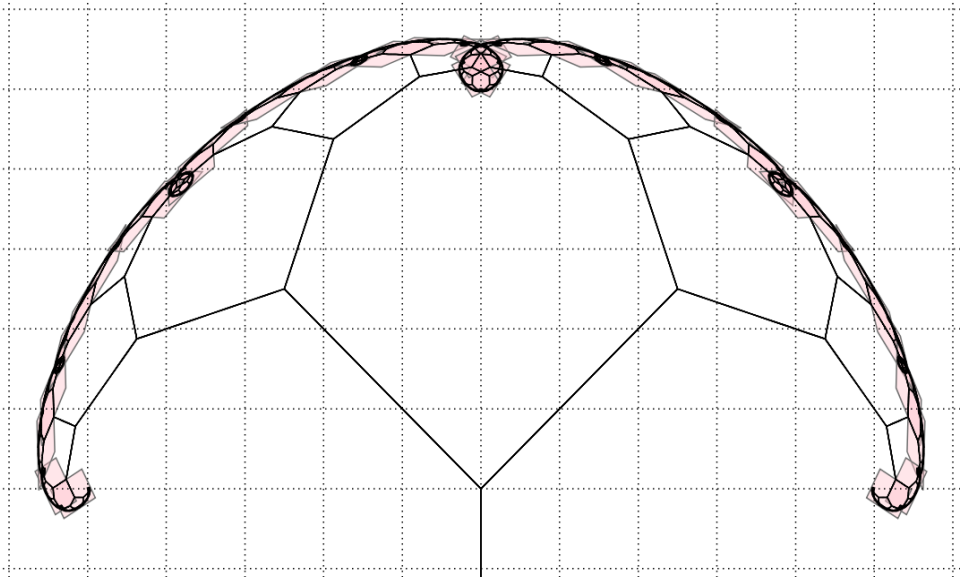
Arbitrary accuracy fractals



Arbitrary accuracy fractals



Arbitrary accuracy fractals

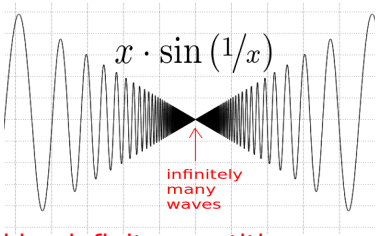


Strange but useful

David Hilbert

Any vacancies?

YES! ... BUT can this be computer-simulated?



shrinking infinite repetition

When does the ball stop bouncing?

demo

X^2-1 divisible by 11

take 10 sqdiv11 =
[1,10,12,21,23,32,34,43,45,54]

sqrt(s) ~ $\sqrt{s}$

for all s ...

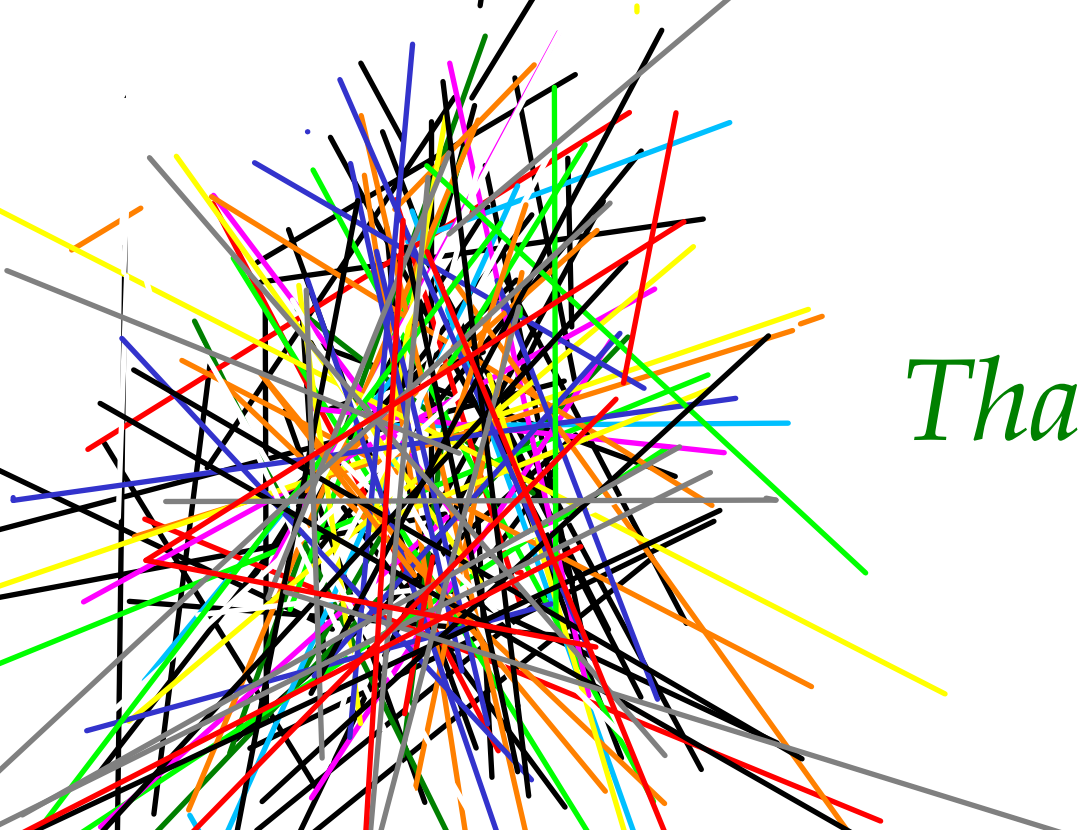
around 4 billion values of s

infinity is easier...

```
sqrt(s) = sqrtB(10,s)
sqrtB(0, s) = 1
sqrtB(n, s) =
      let x = sqrtB(n-1, s)
      in (x + s/x) / 2
```

is the program correct?

a model with inifinity helps to check this



Thank you!