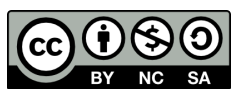
CDM

Iteration

Klaus Sutner

Carnegie Mellon University

Spring 2023



1 **Iteration, Trajectories and Orbits**

2 **The Mandelbrot Set**

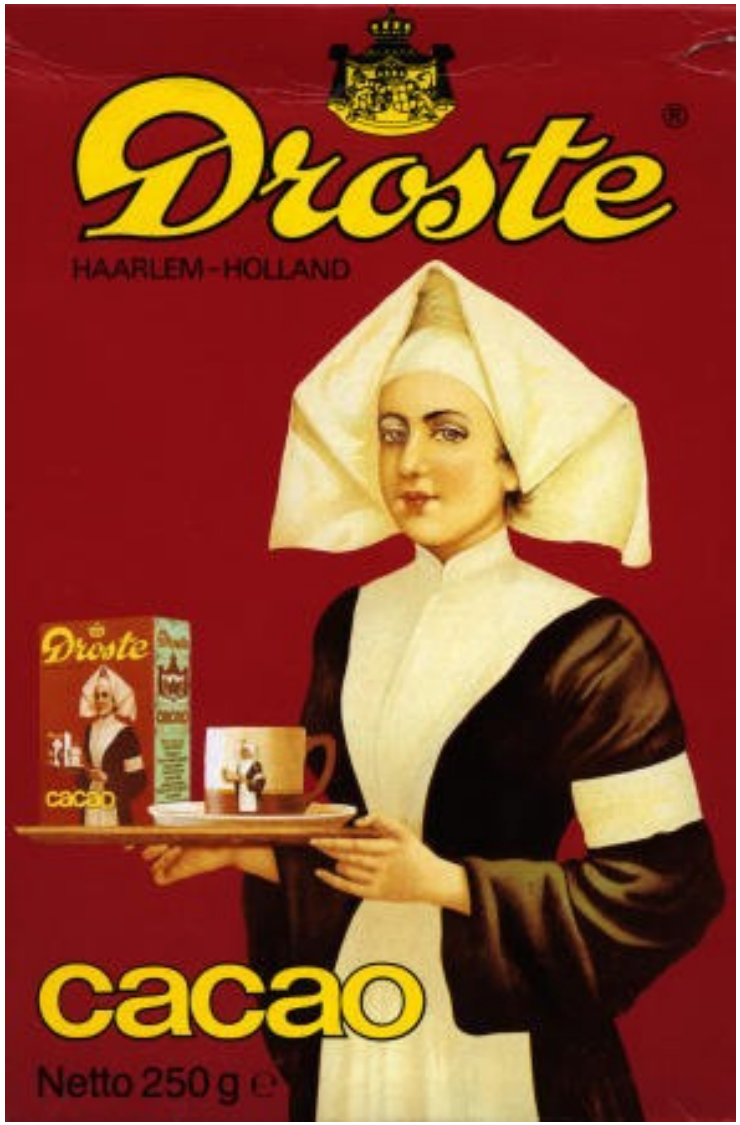
3 **Calculus and Fixed Points**

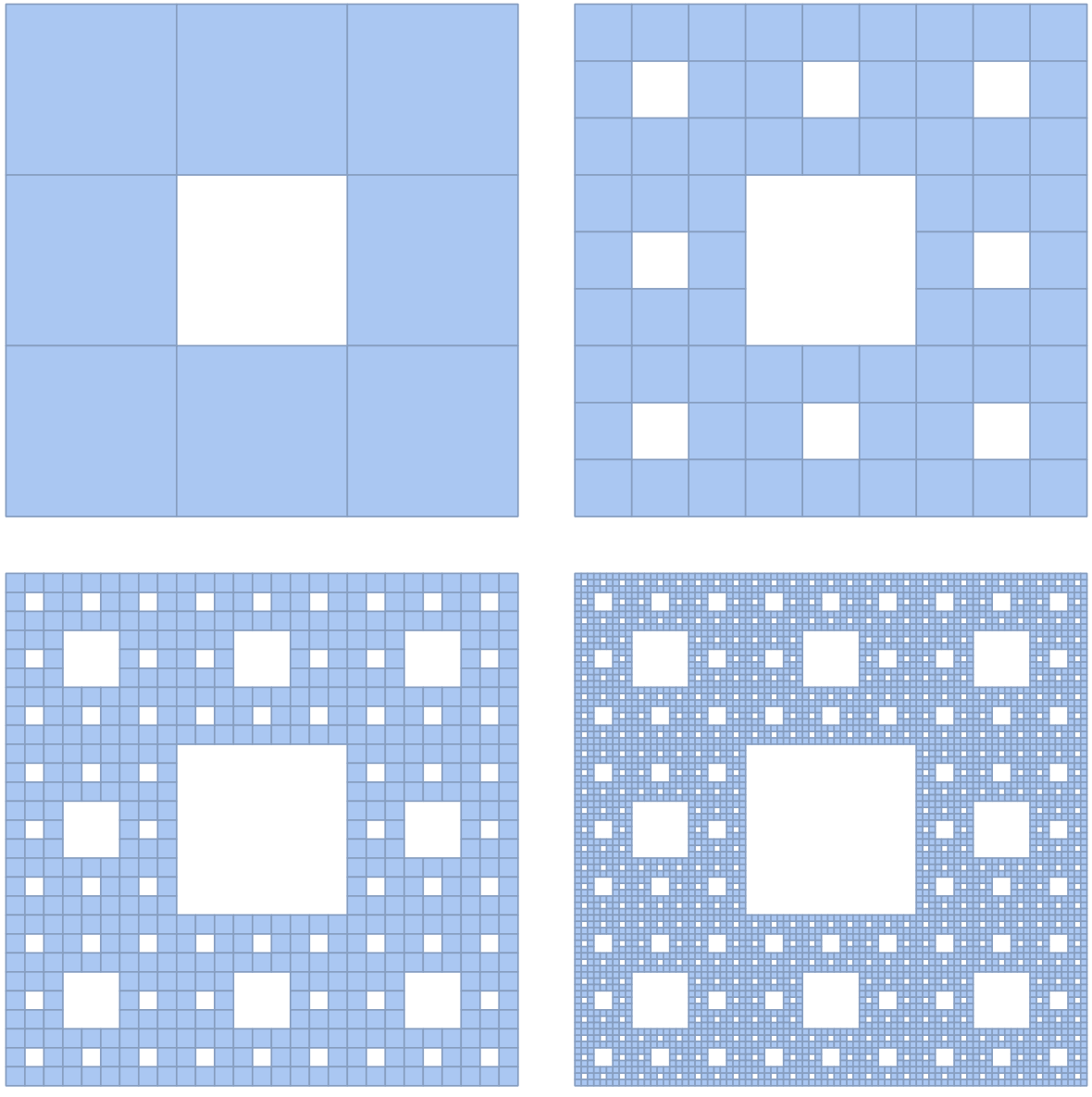
There are several general ideas that are useful to organize computation, perhaps the two most important ones being

- Recursion (self-similarity)
- Iteration (repetition)

Recursion is quite popular and directly supported in many programming languages.

Iteration usually requires some amount of extra work (and, to really make sense, support for functions as first class citizens).


Droste
HAARLEM - HOLLAND
Droste
cacao
cacao
Netto 250 g



Definition

Let $f : A \to A$ be an endofunction. The kth power of f (or kth iterate of f) is defined by induction as follows:

$$f^0 = I_A$$
$$f^k = f \circ f^{k-1}$$

Here I_A denotes the identity function on A and $f \circ g$ denotes composition of functions.

Informally, this just means: compose function f $(k-1)$-times with itself.

$$f^k = \underbrace{f \circ f \circ f \circ \ldots \circ f}_{k \text{ terms}}$$

Without any further knowledge about f there is not much one can say about the iterates f^k. But the following always holds.

Lemma (Laws of Iteration)

- $f^n \circ f = f^{n+1}$
- $f^n \circ f^m = f^{n+m}$
- $(f^n)^m = f^{n \cdot m}$

Exercise

Prove these laws by induction using associativity of composition.

Prof. Dr. Alois Wurzelbrunft* stares at these equations and immediately recognizes a deep analogy to exponentiation.

He also remembers that there is a method for fast exponentiation based on squaring:

$$a^{2e} = (a^e)^2$$
$$a^{2e+1} = (a^e)^2 \cdot a$$

which allows us to compute a^e in $O(\log e)$ multiplications.

Wurzelbrunft's Conclusion:
There is an analogous "fast iteration" method.

*A famous if fictitious professor in the Bavarian hinterland.

> A mathematician is a person who can find analogies between theorems; a better mathematician is one who can see analogies between proofs and the best mathematician can notice analogies between theories. One can imagine that the ultimate mathematician is one who can see analogies between analogies.
>
> S. Banach

So is Wurzelbrunft brilliant?

Suppose we want to compute f^{1000}. The obvious way requires 999 compositions of f with itself.

By copying the standard divide-and-conquer approach for fast exponentiation we could try

$$f^{2n} = (f^n)^2$$
$$f^{2n+1} = f \circ (f^n)^2$$

This seems to suggest that we can compute $f^n(x)$ in $O(\log n)$ applications of the basic function f.

After all, it's just like exponentiation, right?

There is an interesting idea here: we would like to take a plain computation

$$C = C_0, C_1, C_2, \ldots, C_{42}, \ldots, C_n$$

and somehow translate it into another computation

$$C' = C'_0, C'_1, \ldots, C'_m$$

such that

- the result is the same, but
- $m \ll n$

Of course, this won't always be possible, but sometimes we might be able to "compress" a computation (by using a smarter algorithm).

Consider the orbit of a under the rational function (this is a clear case of abuse of a Möbius transformation)

$$f(x) = \frac{2 + 2x}{3 + x}$$

A little fumbling shows that

$$f^t(x) = \frac{2(a-1) + (a+2)x}{2a + 1 + (a-1)x} \qquad a = 4^t$$

So there is no need to iterate f, we can simply do the coefficient arithmetic.

But if you insist . . .

$$f^5(x) = \cfrac{2 + \cfrac{2\left(2 + \cfrac{2\left(2 + \cfrac{2\left(2+\frac{2(2+2x)}{3+x}\right)}{3+\frac{2+2x}{3+x}}\right)}{3+\cfrac{2+\frac{2(2+2x)}{3+x}}{3+\frac{2+2x}{3+x}}}\right)}{3+\cfrac{2+\frac{2\left(2+\frac{2(2+2x)}{3+x}\right)}{3+\frac{2+2x}{3+x}}}{3+\cfrac{2+\frac{2(2+2x)}{3+x}}{3+\frac{2+2x}{3+x}}}}}{3 + \cfrac{2\left(2 + \cfrac{2\left(2+\frac{2(2+2x)}{3+x}\right)}{3+\frac{2+2x}{3+x}}\right)}{3+\cfrac{2+\frac{2(2+2x)}{3+x}}{3+\frac{2+2x}{3+x}}}}}$$

If the function f in question is linear it can be written as

$$f(x) = M \cdot x$$

where M is a square matrix over some suitable algebraic structure. Then

$$f^t(x) = M^t \cdot x$$

and M^t can be computed in $O(\log t)$ matrix multiplications.

So this is an exponential speed-up over the standard method.

Another important case is when f is a polynomial map

$$f(x) = \sum a_i x^i$$

given by a coefficient vector $\boldsymbol{a} = (a_d, \ldots, a_1, a_0)$.

In this case the coefficient vector for $f \circ f$ can be computed explicitly by substitution. This is useful in particular when computation takes place in a quotient ring such as $R[x]/(x^n - 1)$ so that the expressions cannot blow up.

Again, an exponential speed-up over the standard method.

But Beware of Hasty Conclusions

But we cannot conclude that $f^t(x)$ can always be computed in $O(\log t)$ operations.

The reason fast exponentiation and the examples above work is that we can explicitly compute a representation of $f \circ f$, given the representation of f.

But, in general, there is no fast representation for $f \circ f$, we just have to evaluate f twice.

Just think of f as being given by an executable, a compiled piece of C code. We can wrap a loop around the executable to compute f^t, but that just evaluates f t-times, in the obvious brute-force way. No speed-up whatsoever.

Exercise

Ponder deeply. Assume the speed-up trick always works and figure out what that would mean for complexity theory.

Speaking about hasty conclusions, here is a simple inductively defined sequence of integers.

$$a_1 = 1$$
$$a_n = a_{n-1} + (a_{n-1} \bmod 2n)$$

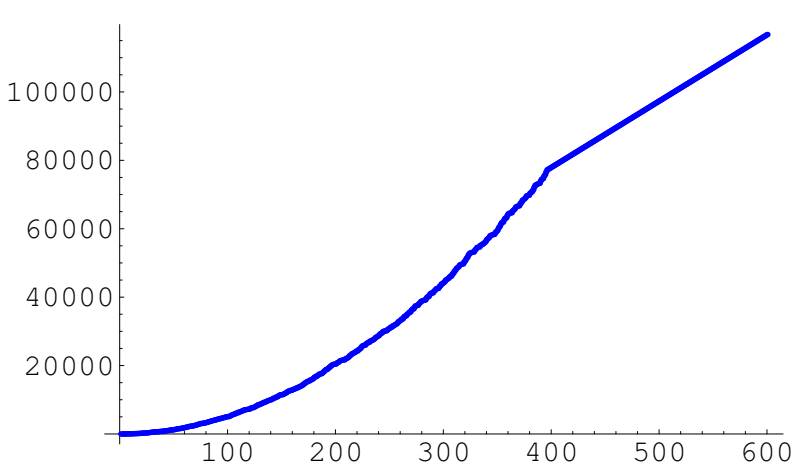
Thus, the sequence starts like so:

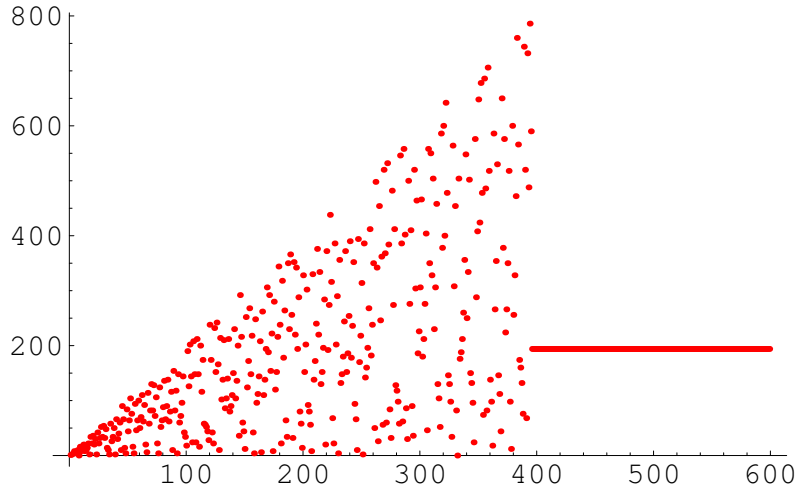
$$1, 2, 4, 8, 16, 20, 26, 36, 36, 52, 60, 72, 92, 100, 110, 124, 146, 148, 182, 204$$

This seems rather complicated. The function appears to be increasing in a somewhat complicated manner.

Alas, there is a rude surprise.

The sequence is ultimately linear: $a_{396+k} = a_{396} + k \cdot 194$ for $k \geq 0$.





The plot on the left is the sequence, on the right (in red) are the forward differences.

Exercise

Figure out why the sequence is ultimately linear.

Here is another strange integer sequence:

$$a_n = \lceil 2/(2^{1/n} - 1) \rceil - \lfloor 2n/\ln 2 \rfloor$$

This time, the sequence starts like so:

$$0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \ldots$$

and continues like this for a long, long time, for trillions of terms.

Note that it is a minor pain to compute the terms; it's not even clear that $n \mapsto a_n$ is primitive recursive [†]. At any rate, it sure looks like the sequence is constant 0. Alas

$$a_{777\,451\,915\,729\,368} = 1$$

[†] The expression looks like real arithmetic, but it can be handled with just integer arithmetic.

Iteration can be construed as a special case of primitive recursion.

$$F(0, y) = y$$
$$F(x + 1, y) = f(F(x, y))$$

Then $F(x, y) = f^x(y)$.

This is really no more than the standard bottom-up approach to computing an primitive recursive function, expressed in an elegant and concise way.

Conversely, iteration can be used to express recursion. Suppose

$$f(0, \boldsymbol{y}) = g(\boldsymbol{y})$$
$$f(x+1, \boldsymbol{y}) = h(x, f(x, \boldsymbol{y}), \boldsymbol{y})$$

Define a new function H by

$$H : \mathbb{N} \times \mathbb{N} \times \mathbb{N}^k \longrightarrow \mathbb{N} \times \mathbb{N} \times \mathbb{N}^k$$
$$H(x, z, \boldsymbol{y}) = (x+1, h(x, z, \boldsymbol{y}), \boldsymbol{y})$$

Then

$$f(x, \boldsymbol{y}) = \mathsf{snd}(H^x(0, g(\boldsymbol{y}), \boldsymbol{y}))$$

This is perhaps the most natural definition, but if we wanted to we could make H unary by coding everything up as a sequence number.

More surprisingly, suppose we have some simple basic functions such as

$$x + y \qquad x * y \qquad x \dot{-} y \qquad \mathsf{rt}(x)$$

Here $\mathsf{rt}(x)$ is the integer part of $\sqrt{x}$. These functions suffice to set up the usual coding machinery. If we add an additional operation of iteration

$$f(x) = g^x(0)$$

we can replace primitive recursion by unary iteration.

Exercise

Come up with a precise version of this statement (define a clone) and give a detailed proof.

Definition

The trajectory or orbit of $a \in A$ under f is the infinite sequence

$$\mathsf{orb}_f(a) = a, f(a), f^2(a), \ldots, f^n(a), \ldots$$

The set of all infinite sequences with elements from A is often written A^ω. Hence the we can think of the trajectory as an operation of type

$$(A \to A) \times A \to A^\omega$$

that associates a function on A and element in A with an infinite sequence over A.

Sometimes one is not interested in the actual sequence of points but rather in the set of these points:

$$\{ f^i(a) \mid i \geq 0 \}$$

While the sequence is always infinite, the underlying set may well be finite, even when the carrier set is infinite.

In a sane world one would refer to the sequences as trajectories, and use the term orbit for the underlying sets. Alas, in the literature the two notions are hopelessly mixed up.

So, when we refer to a "trajectory" we will always mean the sequence, but, bending to custom, we will use "orbit" for both.

Here is a clever definition due to Dedekind: given an endofunction f and a point a, the corresponding chain is defined to be

$$\bigcap \{ X \subseteq A \mid a \in X, f(X) \subseteq X \}$$

Thus, the chain is the least set that contains a and is closed under f. That is exactly the orbit of a under f, considered as a set.

Who cares?

Dedekind's definition does not require the natural numbers. In fact, it can be used to define them. In Dedekind's view, this means that arithmetic can be reduced to logic.

Here is how. Suppose we have a function $f : A \to A$ and a point $a \in A$ such that

- f is an injection,
- a is not in the range of f,
- A is the chain of f and a.

Dedekind calls these sets simply infinite.

We can think of a as 0 and, more generally, we can think of $f^n(a)$ as n.

So this is a way of describing the natural numbers, the smallest infinite set, without any hidden references to the naturals.

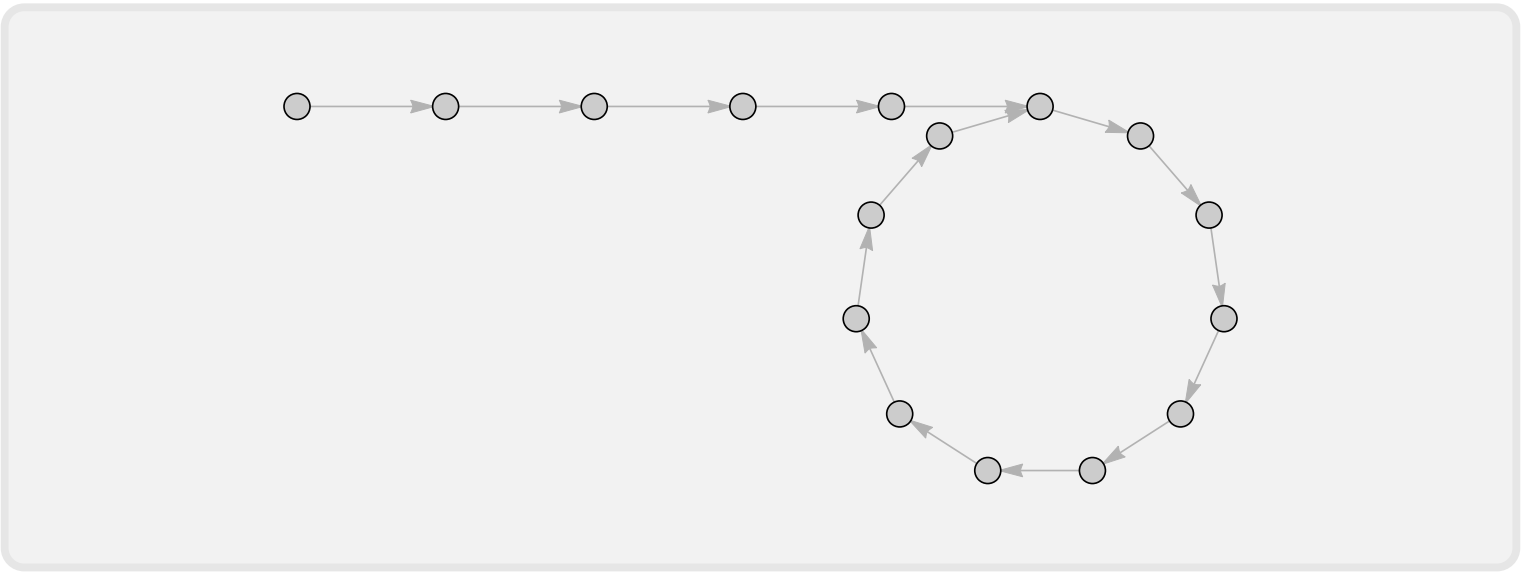
According to Dedekind, the chain C defined by f and a has the form

$$C = \bigcap \{\, X \subseteq A \mid a \in X, f(X) \subseteq X \,\}$$

But note that C is one of the X's on the right hand side. So there is some (non-vicious) circularity in this approach. Most mathematicians would not bat an eye when confronted with definitions like this one, they are totally standard.

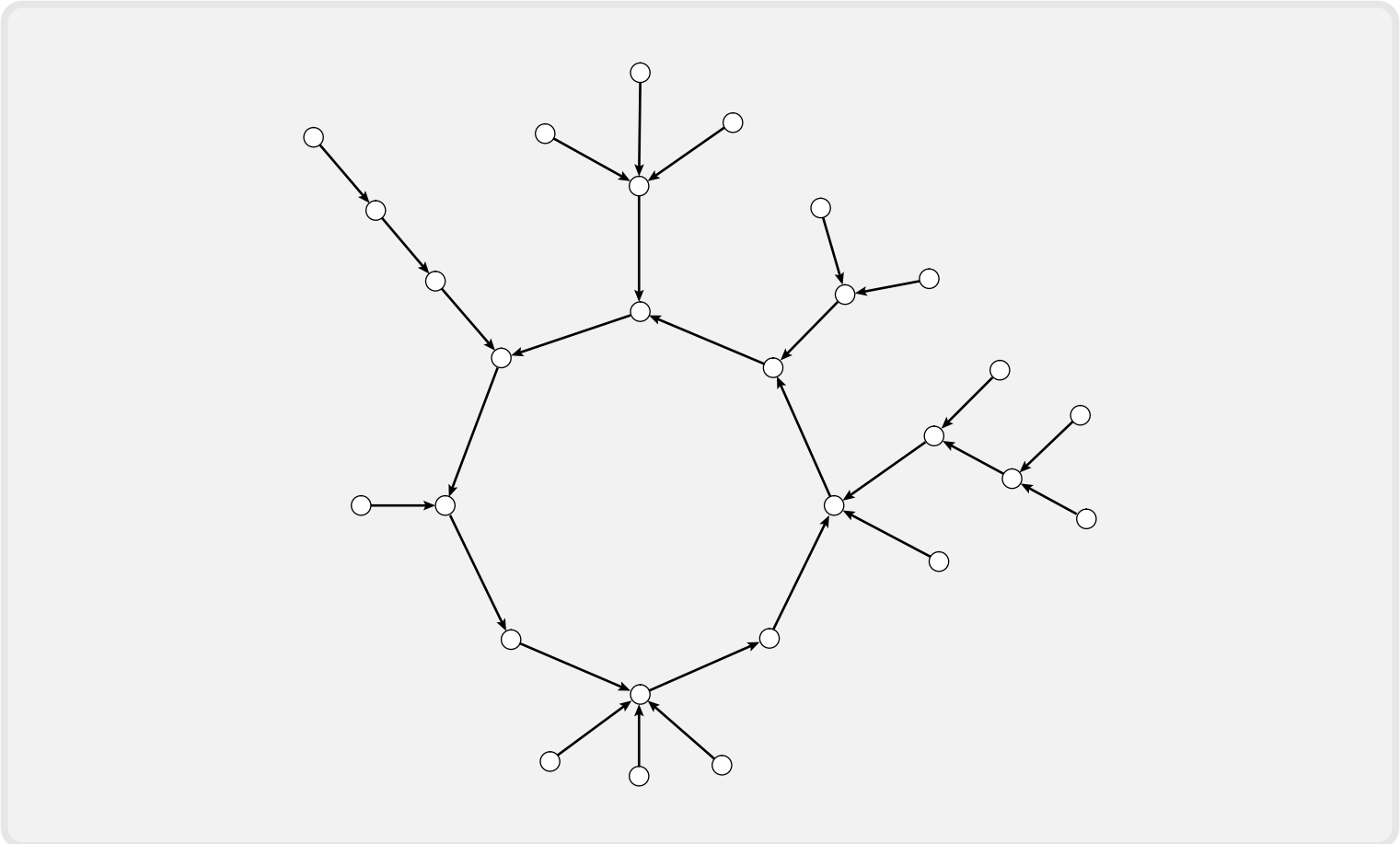
And the payoff is huge. For example, when Bernstein told Dedekind about his correct proof of the "Cantor-Schröder-Bernstein" theorem, he was shocked to hear that Dedekind had a much better proof, based on his chains.

At any rate, if the carrier set is finite, all trajectories must ultimately wrap around and all orbits must be finite:



What changes is only the length of the transient part and the length of the cycle (in the picture 5 and 11).

The lasso shows the general shape of any single orbit, but in general orbits overlap. All orbits with the same limit cycle are called a basin of attraction in dynamics.



The geometric perspective afforded by the diagram also suggests to study path-existence problems.

Definition

Let f be a function on A and $a, b \in A$ two points in A. Then point b is reachable from a if for some $i \geq 0$:

$$f^i(a) = b$$

In other words, point y belongs to the orbit of x.

Proposition

Reachability is reflexive and transitive but in general not symmetric.

Reachability is symmetric when A is finite and f injective (and therefore a permutation): each orbit then is a cycle and forms an equivalence class.

Definition

Let f be a function on A and $a, b \in A$ two points in A. Points a and b are confluent if for some $i, j \geq 0$:

$$f^i(a) = f^j(b)$$

In other words, the orbits of a and b merge, they share the same limit cycle (which may be infinite and not really a cycle).

Reachability implies confluence but not conversely. For finite carrier sets reachability is the same as confluence iff the map is a bijection.

Proposition

Confluence is an equivalence relation.

Reflexivity and symmetry are easy to see, but transitivity requires a little argument.

Let $f^i(x) = f^j(y)$ and $f^k(y) = f^l(z)$, assume $j \leq k$. Then with $d = k - j \geq 0$ we have

$$f^{i+d}(x) = f^{j+d}(y) = f^k(y) = f^l(z).$$

Each equivalence class contains exactly one cycle of f, and all the points whose orbits lead to this cycle – just as in the last picture.

1 Iteration, Trajectories and Orbits

2 **The Mandelbrot Set**

3 Calculus and Fixed Points

Consider the second-degree complex polynomial

$$p_c(z) = z^2 + c$$

where $c \in \mathbb{C}$ is a constant. We can turn this into a nice decision problem:

Problem:	**Mandelbrot Problem**
Instance:	A compllex number c.
Question:	Is the orbit of 0 under p_c bounded?

Of course, this is not a classical decision problem where the instances are required to be finitary objects, but let's ignore that.

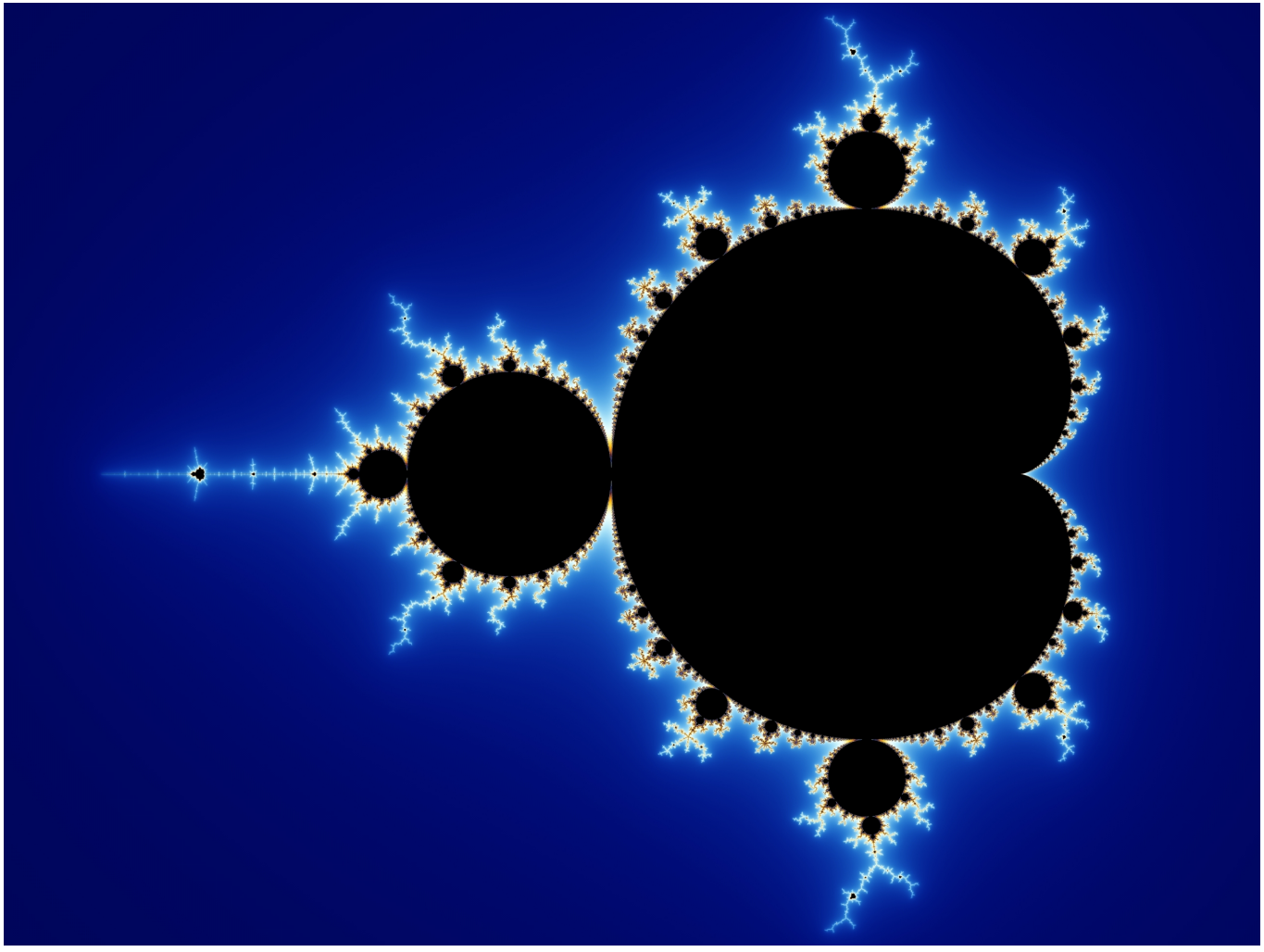
Definition

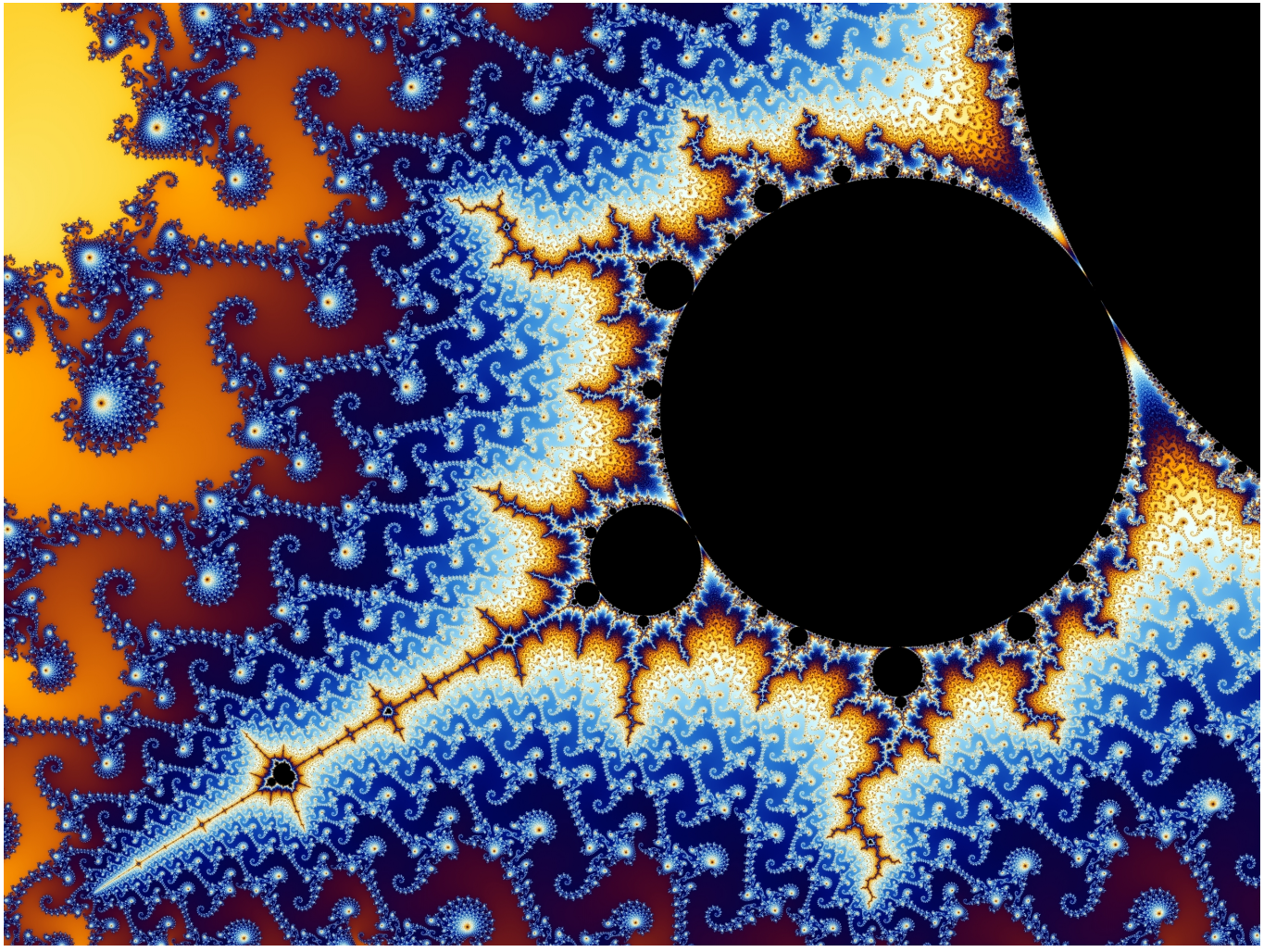
The Mandelbrot set $M \subseteq \mathbb{C}$ is the set of Yes-instances of the Mandelbrot Problem.

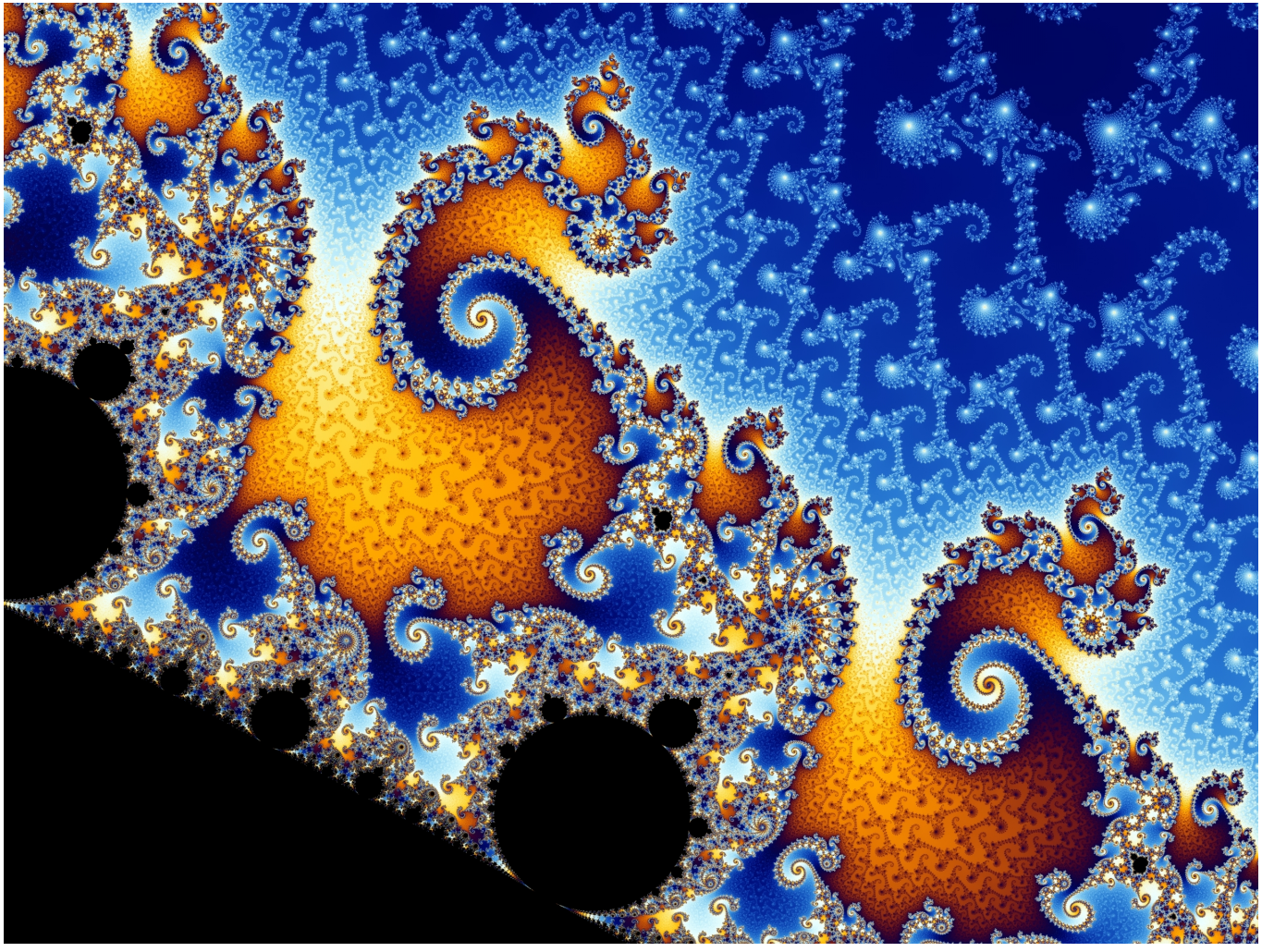
Note that it is quite tricky to determine whether a particular complex number c belongs to M or not: we can compute $p_c^t(0)$ for small values of t, but it is often unclear whether this sequence diverges.

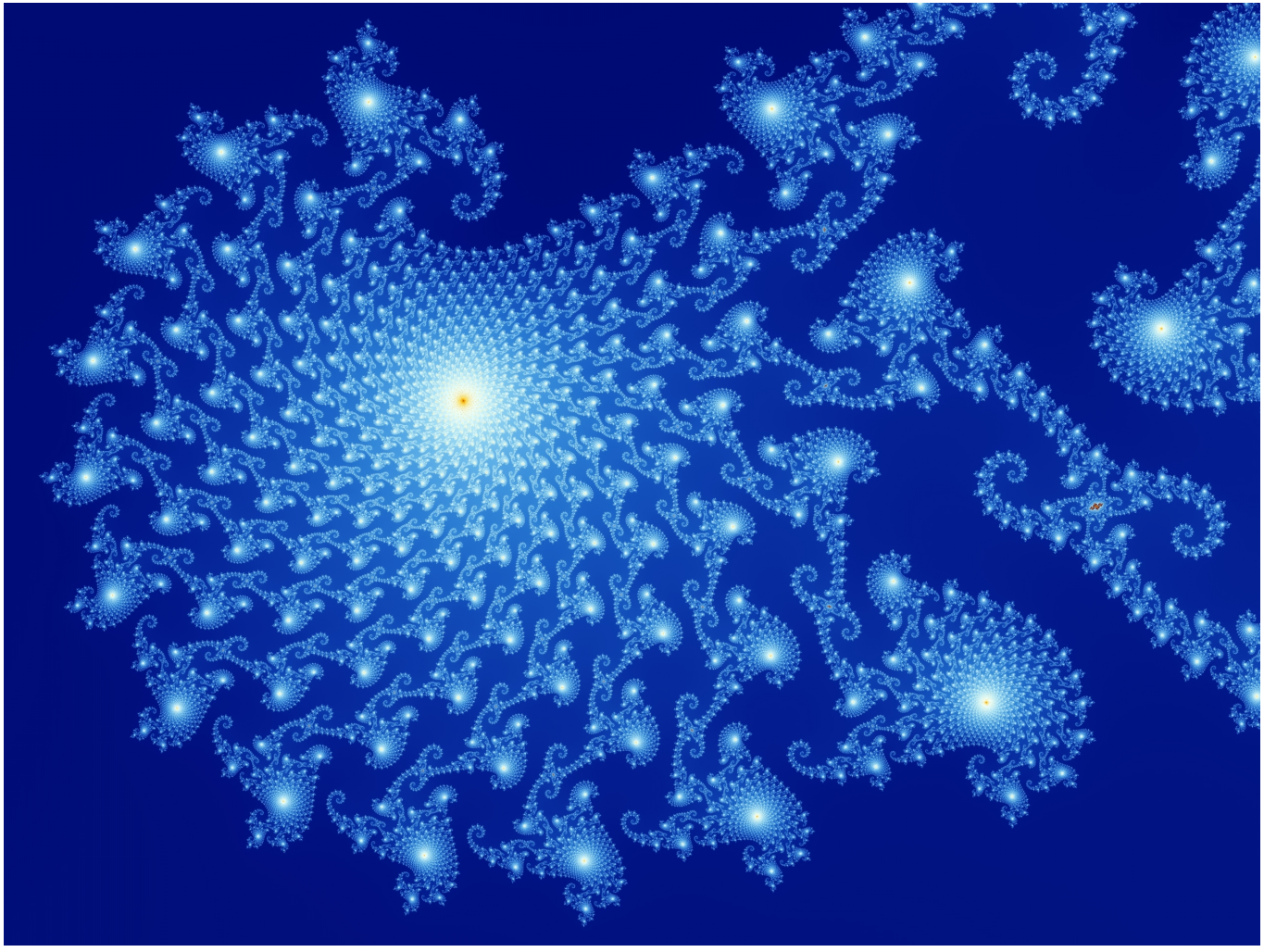
M 0, $\mathbf{i}$, $-\mathbf{i}$, $1/5$, $-11/10$, $-13/10$, $-19/10$, -2

$\overline{M}$ 1, 2, $1/2$, $1/4$, $2\mathbf{i}$, $-2\mathbf{i}$









And the Colors?

A priori, we only get a black and white picture: c is in M or not in M.

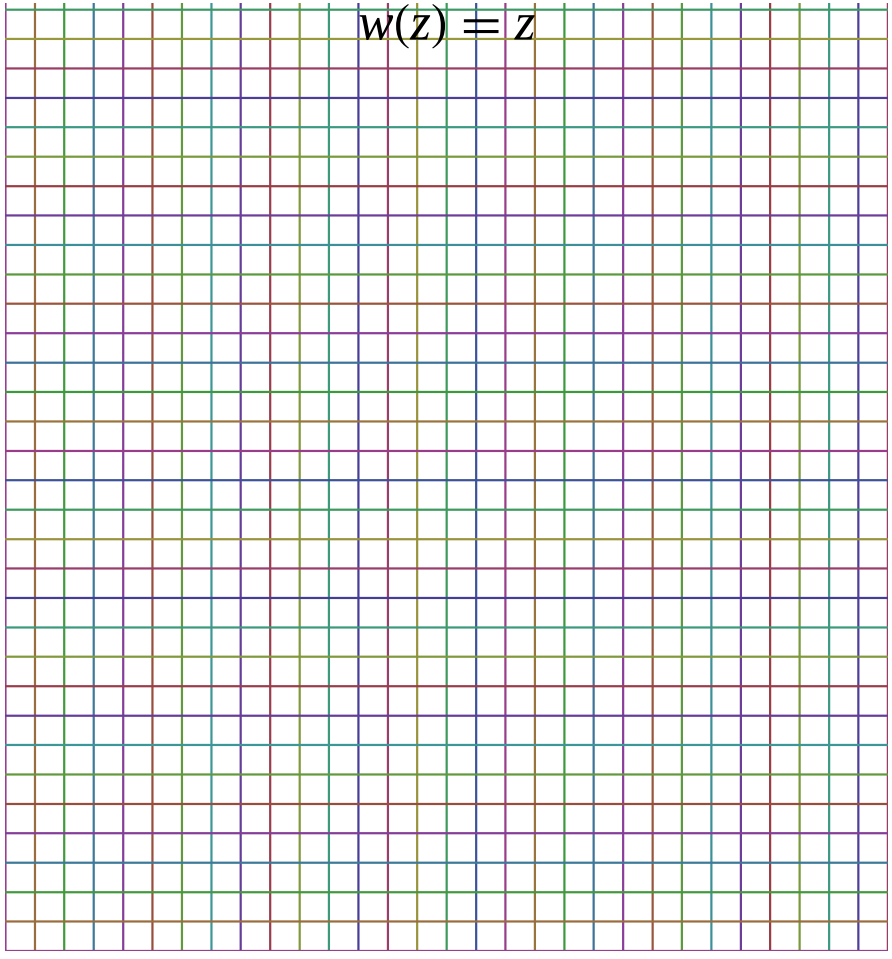
Again, it is not even clear how to do this much: there is no simple test to check if $|p_c^t(0)| \to \infty$ as $n \to \infty$.

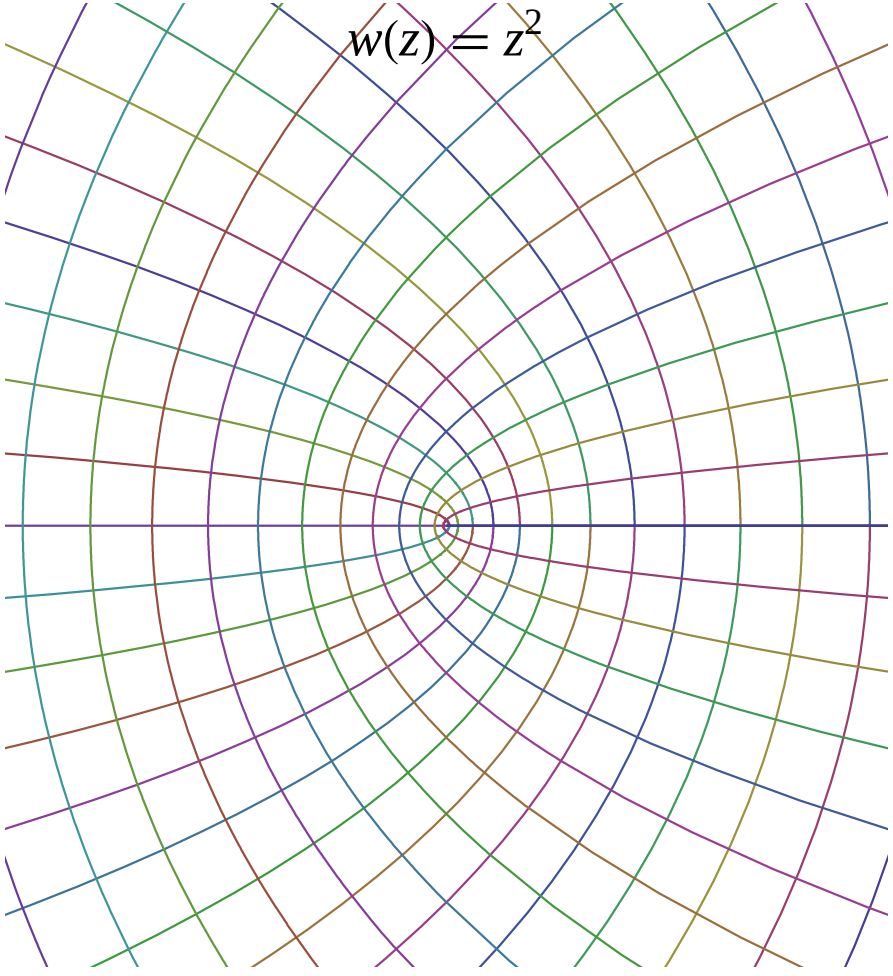
In practice, one computes a few values $|p_c^t(0)|$ and checks whether they seem to tend to infinity.

Colors can be introduced for example based on the speed of divergence.

Doing this right is somewhat of a black art, see Mandelbrot for some background.

Squaring





The symbolic orbit of 0 under $z \mapsto z^2 + c$.

$$
\begin{array}{ll}
0 & 0 \\
1 & c \\
2 & c^2 + c \\
3 & c^4 + 2c^3 + c^2 + c \\
4 & c^8 + 4c^7 + 6c^6 + 6c^5 + 5c^4 + 2c^3 + c^2 + c \\
5 & c^{16} + 8c^{15} + 28c^{14} + 60c^{13} + 94c^{12} + 116c^{11} + \\
 & 114c^{10} + 94c^9 + 69c^8 + 44c^7 + 26c^6 + 14c^5 + 5c^4 + 2c^3 + c^2 + c
\end{array}
$$

Note that the coefficients are rather wild, there is little hope to understand these polynomials.

Several mathematicians developed the foundations for structures such as the Mandelbrot set in the early 20th century, in particular Gaston Julia and Pierre Fatou. Fractal dimensions were also well understood, see the work by Felix Hausdorff and Abram Besicovitch.

Why did they not discover the Mandelbrot set?

Because they had no computers, unlike Monsieur Mandelbrot who happened to be working for IBM at Yorktown Heights.

So one of the most important developments in geometry in the 20th century was quite strongly connected to computation and visualization.

1 Iteration, Trajectories and Orbits

2 The Mandelbrot Set

3 **Calculus and Fixed Points**

Many interesting applications of fixed points lie in the continuous domain: finding a fixed point is often a good method in calculus to compute certain numbers.

A classical problem: calculate $\sqrt{2}$, for some value of 2.

By "calculate" we mean: give a method that produces as many digits in the decimal expansion of $\sqrt{2}$ as desired.

One obvious way to do this is a version of binary search: given an approximation a, b such that $a^2 < 2 < b^2$ try $(a + b)/2$.

Fine, but a bit complicated.

Consider the map

$$g : \mathbb{R}^+ \to \mathbb{R}^+$$
$$g(x) = x/2 + 1/x$$

Then the iterates $g^n(1)$ approximate $\sqrt{2}$.

Note that $\sqrt{2}$ is in fact a fixed point of g. Alas, there is a problem: it is plain false to claim that

$$\mathrm{FP}(g, 1) = \sqrt{2}$$

All the numbers in the orbit are rational, but our goal is an irrational number, which therefore cannot be a fixed point.

But wouldn't it be nice if we could set

$$a = g^{\omega}(1)$$

so that

$$g(a) = g(g^{\omega}(1)) = g^{\omega}(1) = a$$

In a way, we can: we can make sense out of $g^{\omega}(1)$ by using limits: with luck there will be some number a such that

$$|g^{n}(1) - a| \to 0 \qquad n \to \infty.$$

Of course, this does not always work.

In our case we duly have

$$\lim_{n\to\infty} g^n(1) = \sqrt{2}.$$

and $\sqrt{2}$ is indeed a fixed point of g, the orbit just never reaches this particular point.

This is no problem in real life: we can stop the iteration when $|2 - (g^n(1))^2|$ is sufficiently small.

As a matter of fact, convergence is quite rapid:

0	1.00000000000000000
1	1.50000000000000000
2	1.41666666666666518
3	1.41421568627450966
4	1.41421356237468987
5	1.41421356237309492
6	1.41421356237309492

This is Newton's Method: to find a root of $f(x) = 0$, iterate

$$g(x) = x - \frac{f(x)}{f'(x)},$$

and pray that everything works.

Obviously f needs to be differentiable here, and we would like $f'(x)$ to be sufficiently far away from 0 so that the second term does not become unreasonably large.

A typical application of Newton's Method is to determine $1/a$ in high precision computations.

Here we use functions

$$f(x) = 1/x - a$$
$$g(x) = 2x - ax^2$$

The point is that we can express division in terms of multiplication and subtraction (operations that are arguably more basic).

Numerical values for $a = 1.41421356237309504 88 \approx \sqrt{2}$.

0	1.0000000000000000000
1	0.58578643762690495 11
2	0.68629150101523960 95
3	0.70649403654862594 51
4	0.70710625021159255 13
5	0.70710678118614880 89
6	0.70710678118654752 44
7	0.70710678118654752 44

Verify the result:

$$0.7071067811865475244 \times 1.4142135623730950488 = 1.0000000000000000000$$

If quadratic convergence is not enough one can speed things up tremendously by iterating more complicated functions. For example, define $f_a(x)$ to be the rational function

$$x - \frac{\left(x^2 - a\right)\left(3x^2 + a\right)\left(3x^6 + 27x^4a + 33x^2a^2 + a^3\right)}{2x\left(5x^4 + 10x^2a + a^2\right)\left(x^4 + 10x^2a + 5a^2\right)}$$

Then f_a can be used to approximate square roots very rapidly.

$$f_2^2(1) = \frac{947411251496369334178730799209000179 37}{669920920505516376634389067131823137 72}$$

The error here is 2.2281×10^{-76}, after just 2 steps!

Of course, there is a cost: function evaluation becomes more complicated. Specifically, more costly multiplications and divisions are needed than in the plain Newton case. The hope is that this will be more than offset by the smaller number of iterations.

Since the Gerlach function $f_a(x)$ would presumably be evaluated numerous times as part of some library this is a good place to optimize by precomputing x^2, x^4 and so on.

There is also the minor problem of figuring out what these complicated functions should be in the first place.

Exercise

Determine the optimal evaluation strategy for $f_a(x)$.

Needless to say, complicated numerical methods should not be implemented by hand, they belong into a well-thought-out and well-tested library.

For example, the Boost C++ library supports a number of fast root finding methods.

```
#include <boost/math/tools/roots.hpp>

template <class F, class T>
T schroeder_iterate(F f, T guess, T min, T max, int digits);
```

Exercise

Try out the various root finding methods in Boost.

As we have seen, with luck, a fixed point for a continuous function $f : \mathbb{R} \to \mathbb{R}$ can be found by plain iteration, at least in the sense that we can produce a numerical approximation (perhaps even rapidly).

We compute $a_n = f^n(a)$ and exploit the fact that if there is a limit $a = \lim a_n$ then a is a fixed point of f.

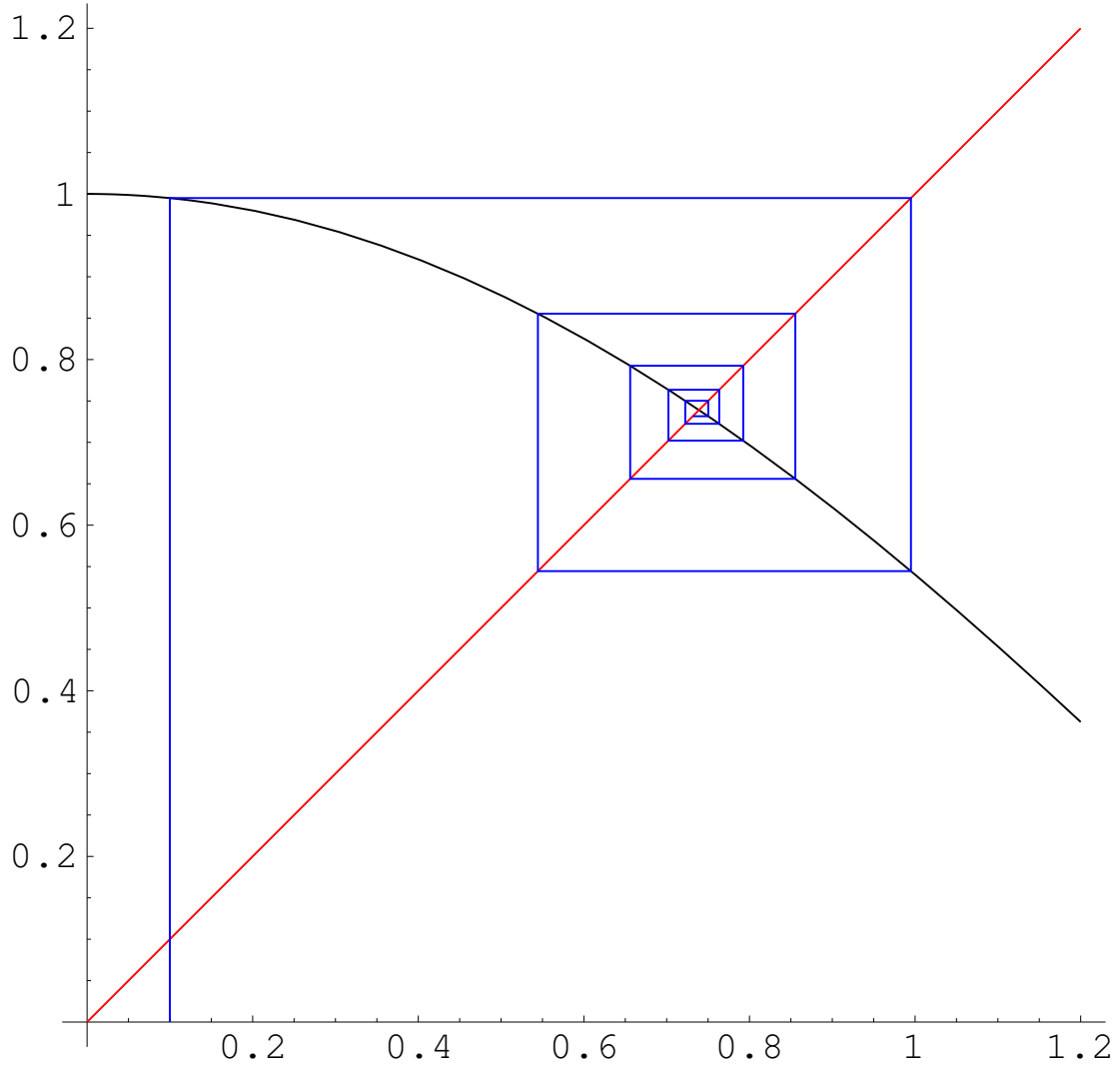
Moreover, a_n may be very close to a for reasonably small values of n so we can actually get our computational hands on a good approximation.

Alas, iteration does not always produce fixed points, even when they are easy to detect visually.

Here are some examples.

A Nice Fixed Point

$\cos x = x$



Cosine is a transcendental function, and thus relatively complicated. Alas, as the Mandelbrot set suggests, even with second order polynomials strange things can happen under iteration. Here is a real example:

$$f_p(x) = p \cdot x \cdot (1 - x)$$

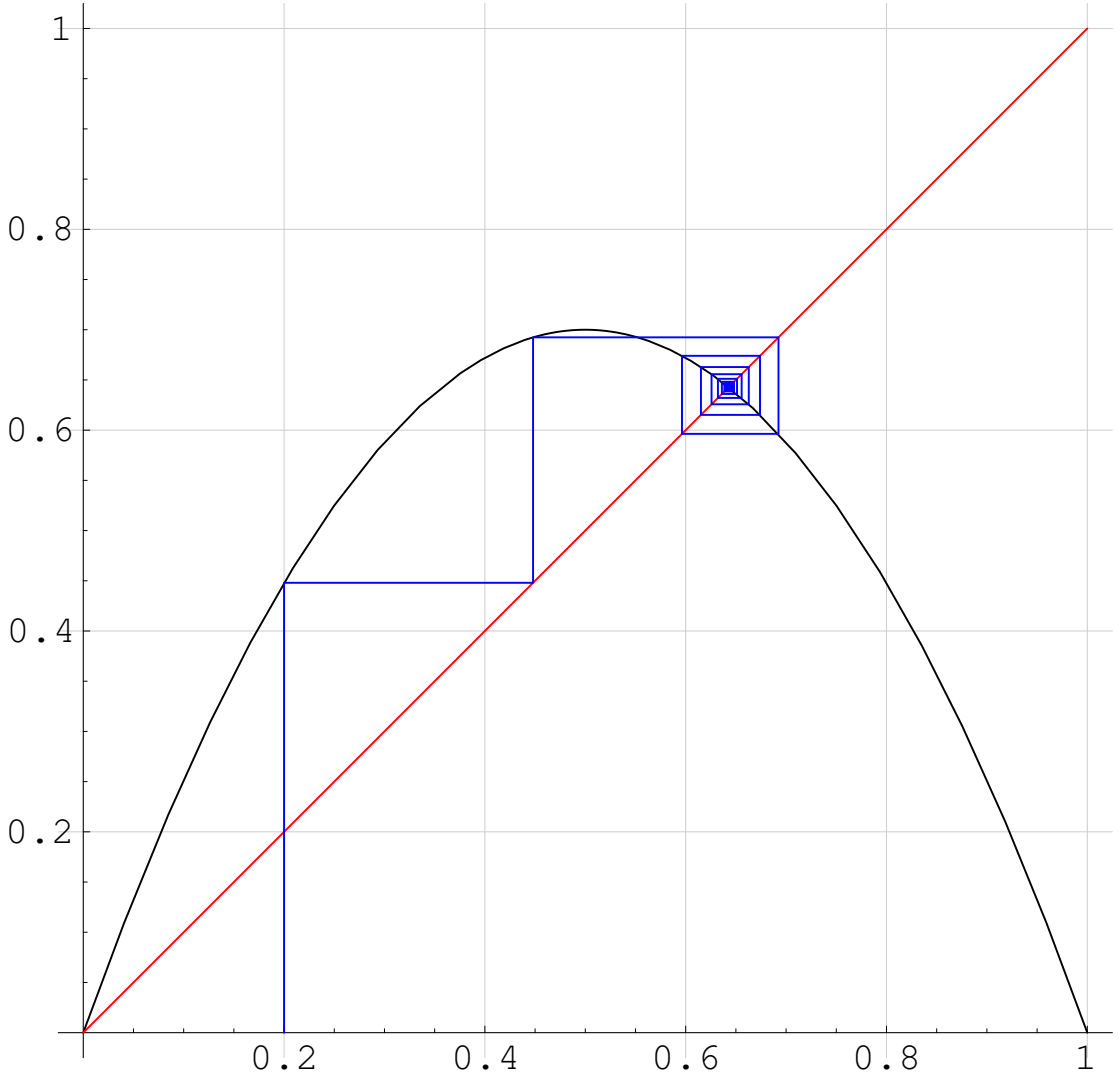
where $0 \leq p \leq 4$. Note that for these parameter values we have

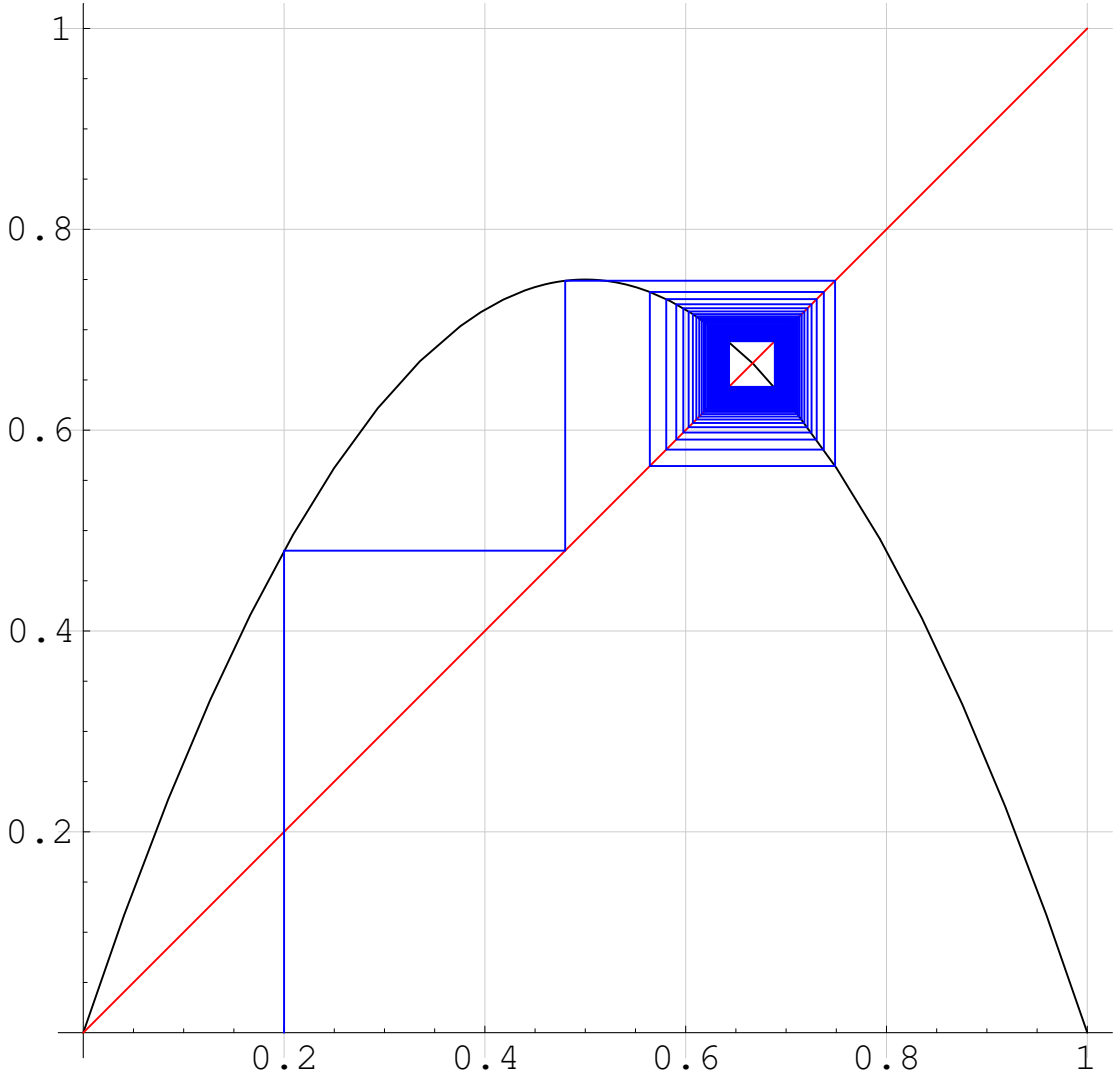
$$f_p : [0, 1] \to [0, 1]$$

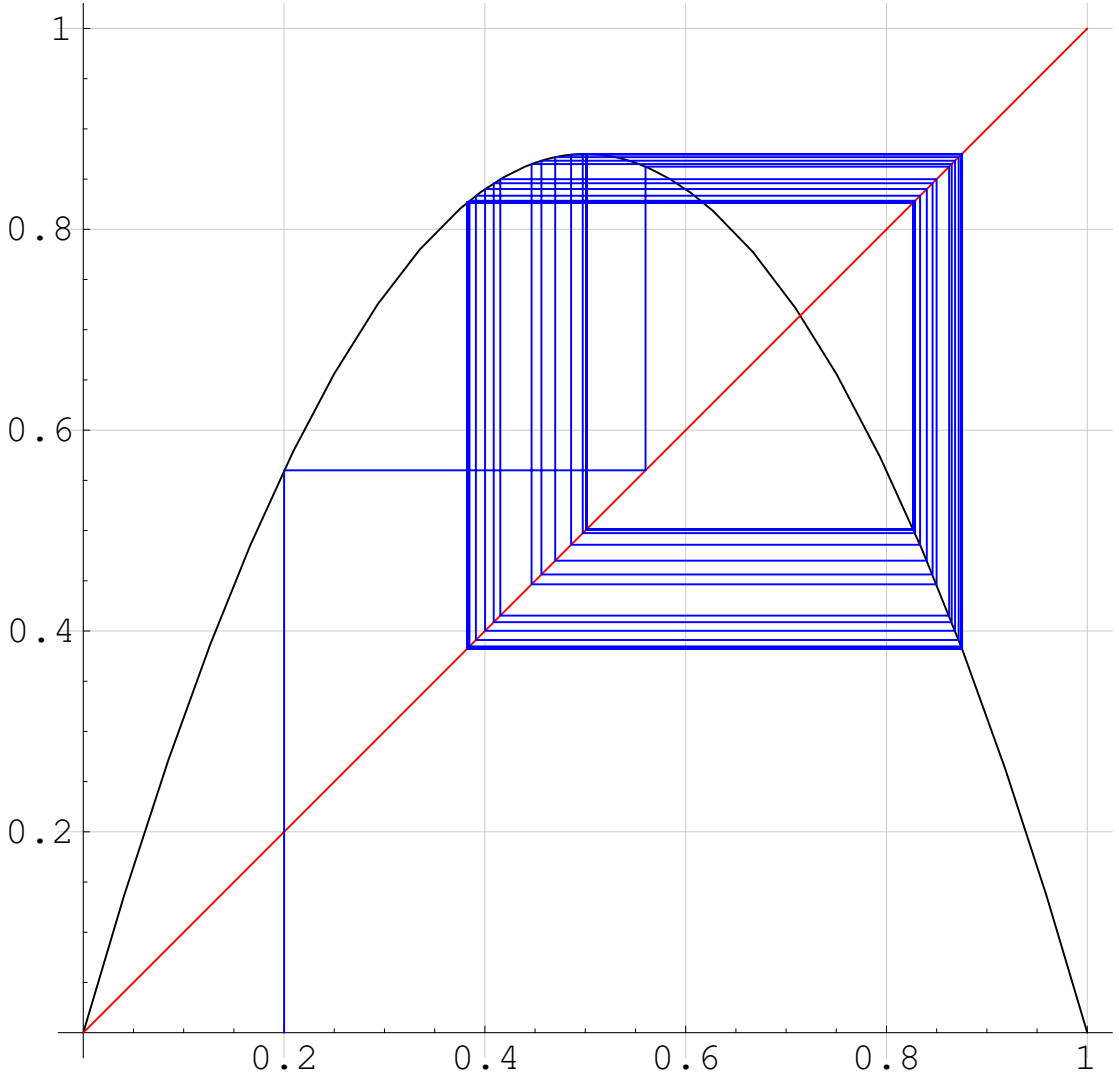
so we can actually iterate the map.

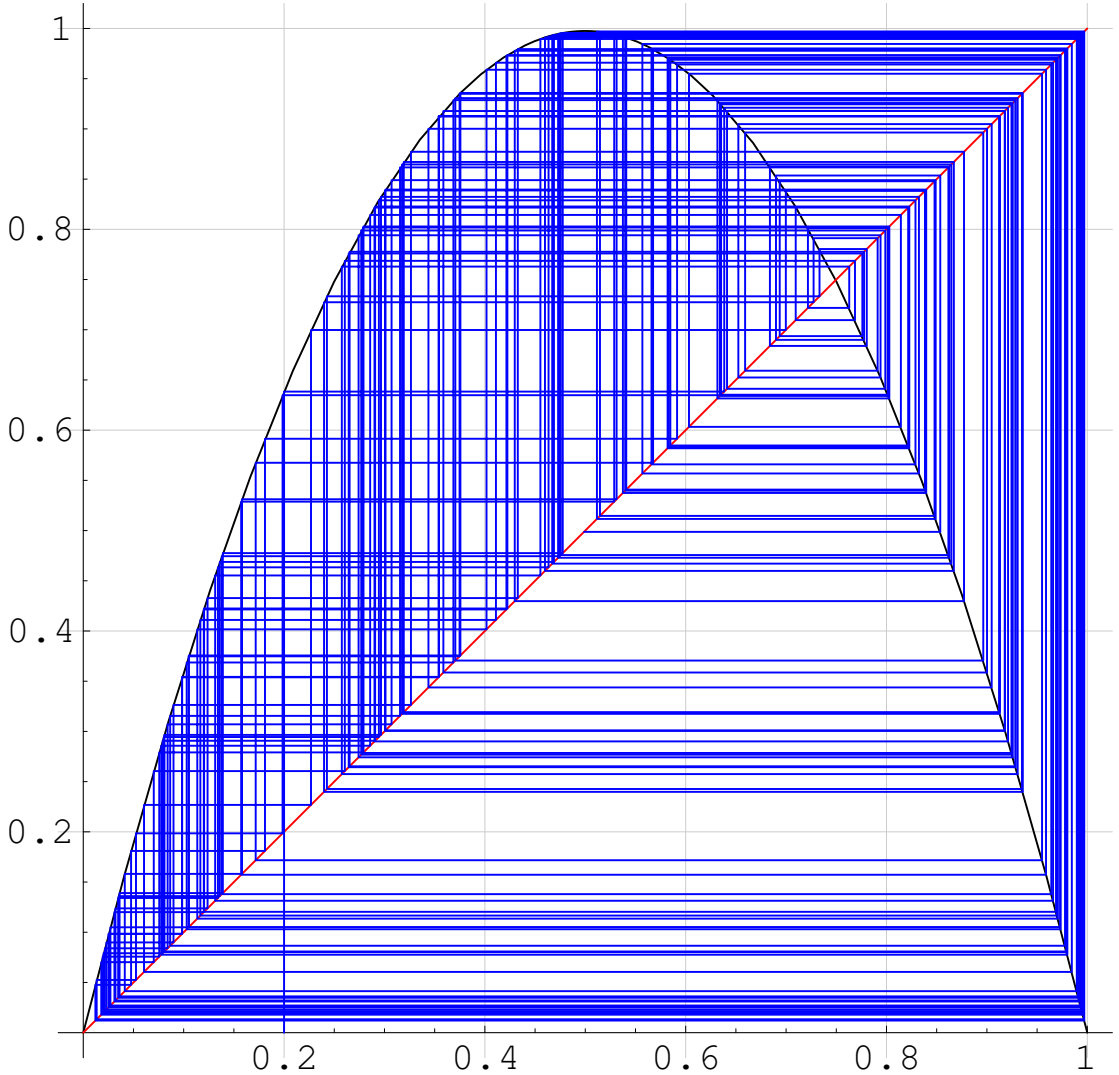
This is the so-called logistic map, a famous example in dynamics.

Its behavior depends drastically and very unexpectedly on the parameter p.

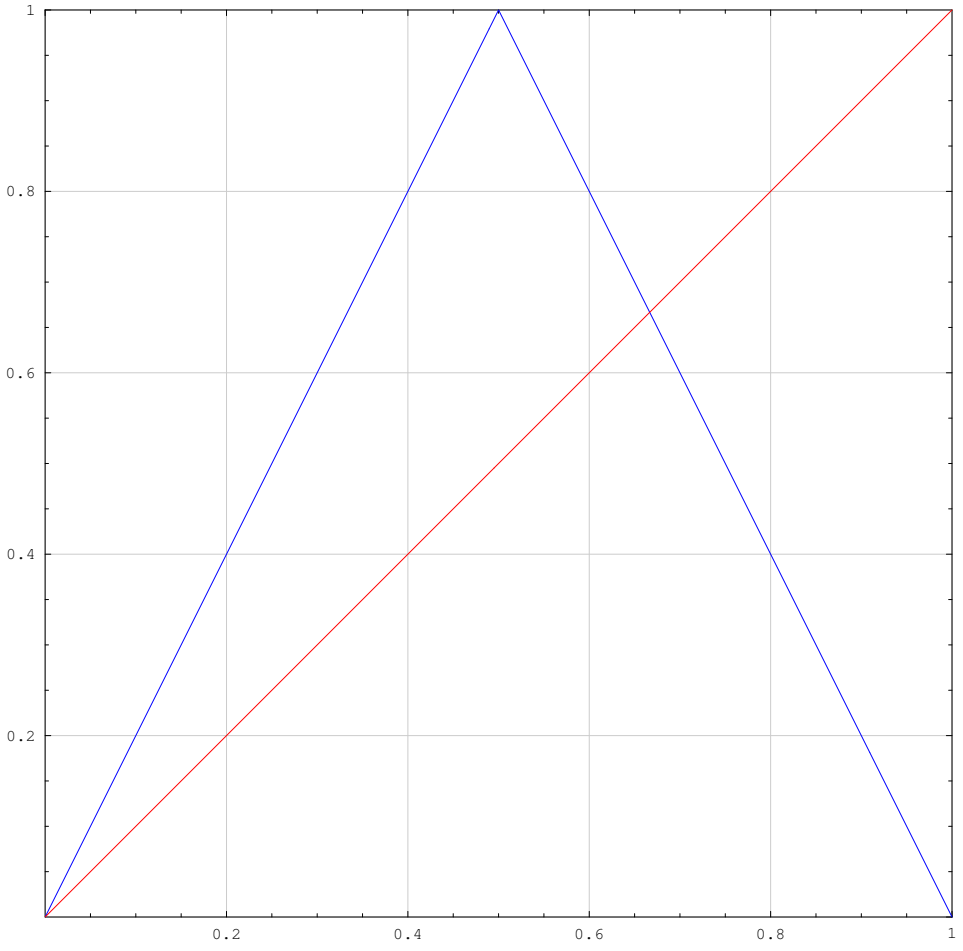

1
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1


1
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1

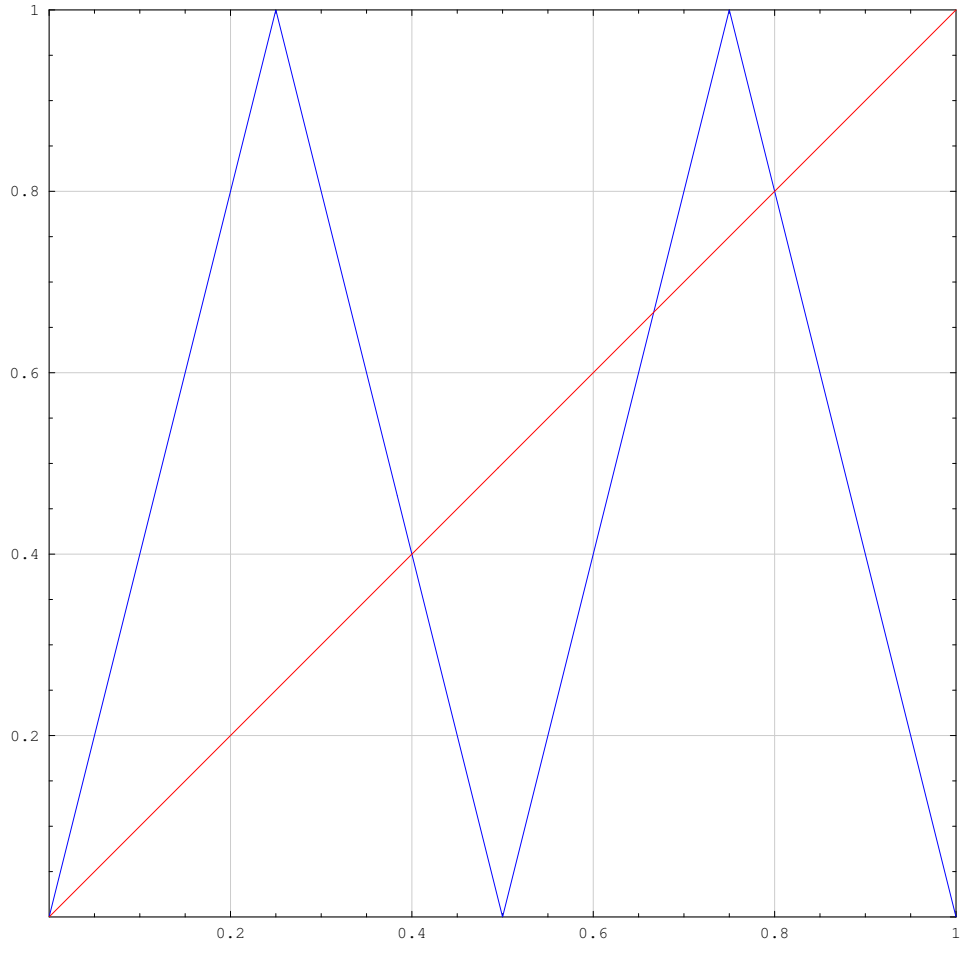

1
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1


1
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1

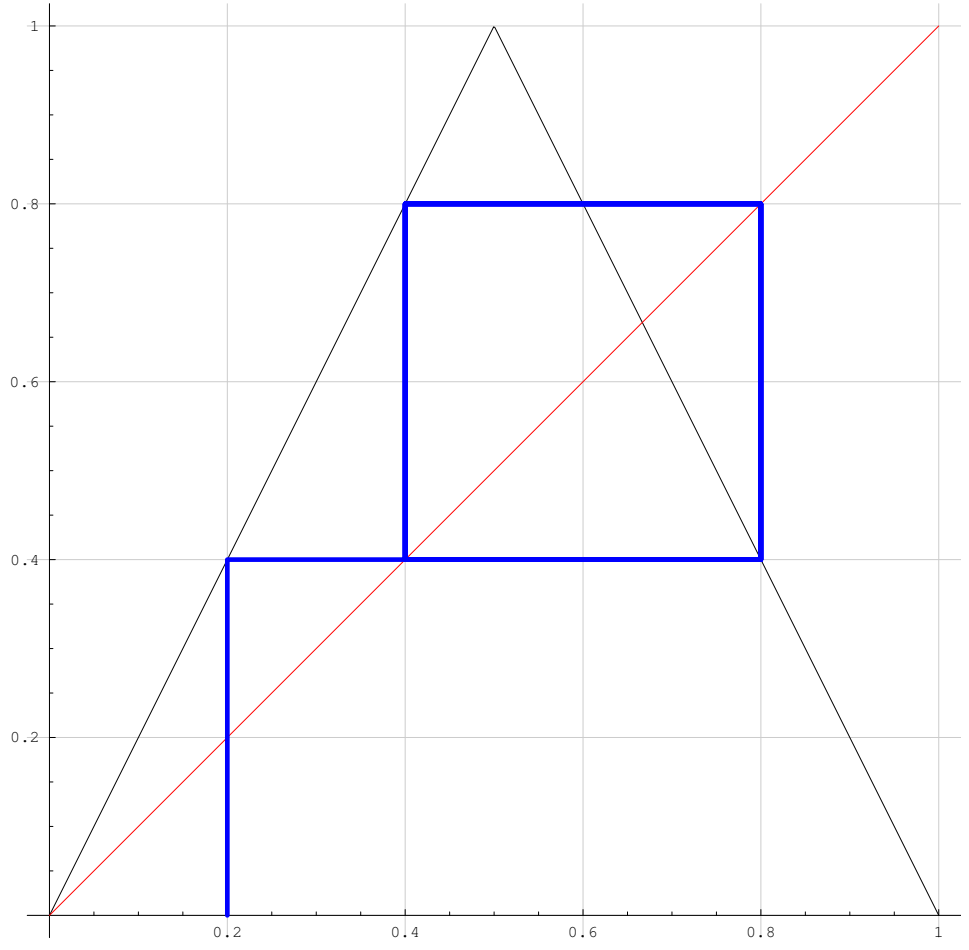
Differentiability is not necessary at all, piecewise linear will do.

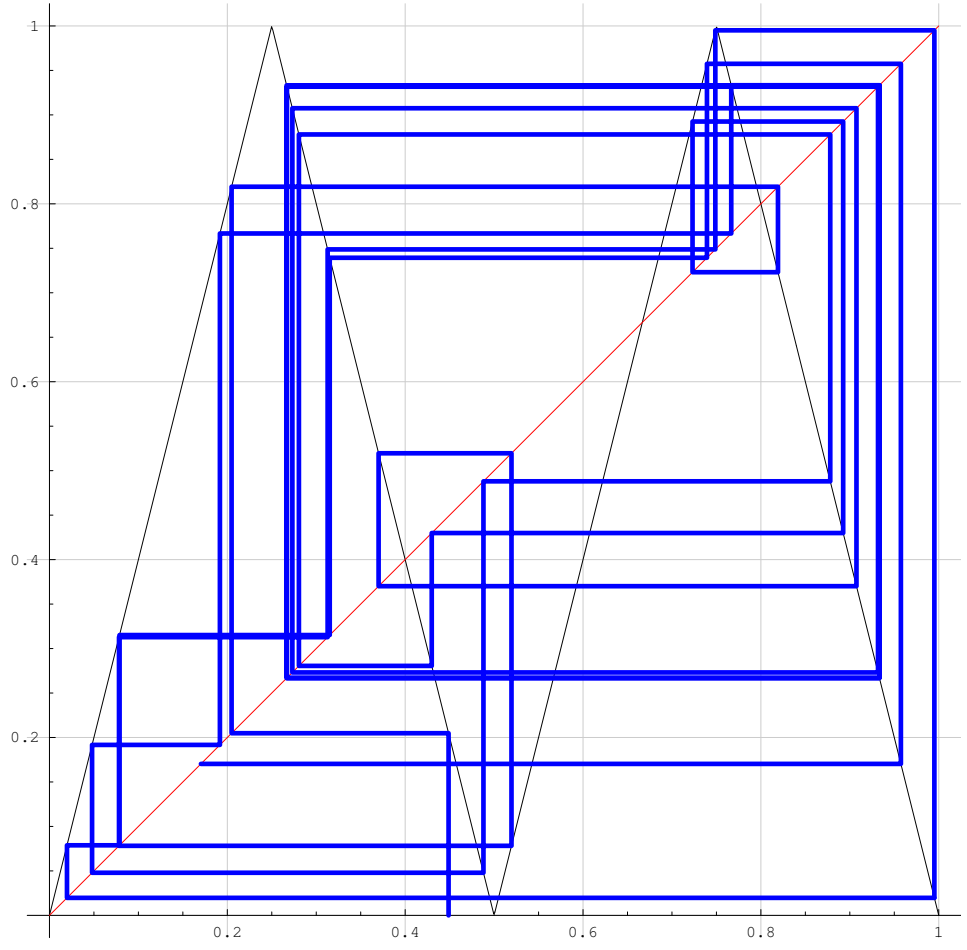


Tent Map Squared



A 2-Cycle




1
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1

Here is a remarkable theorem describing chaos in real valued functions.

Consider the following weird ordering of the natural numbers, the so-called Sarkovskii ordering (of order type $\omega^2 + \omega^{\text{op}}$):

$$3, 5, 7, 9, \ldots, 2 \cdot 3, 2 \cdot 5, \ldots, 4 \cdot 3, 4 \cdot 5, \ldots, 2^3, 2^2, 2^1, 2^0.$$

Theorem (Sharkovskii 1964)

For any continuous function $f : \mathbb{R} \to \mathbb{R}$: if f has a cycle of length α then f has a cycle of length β for all $\alpha < \beta$ in the Sharkovskii ordering.

Hence, if there is a 3-cycle, then there are cycles of any length.