CDM

Iteration III

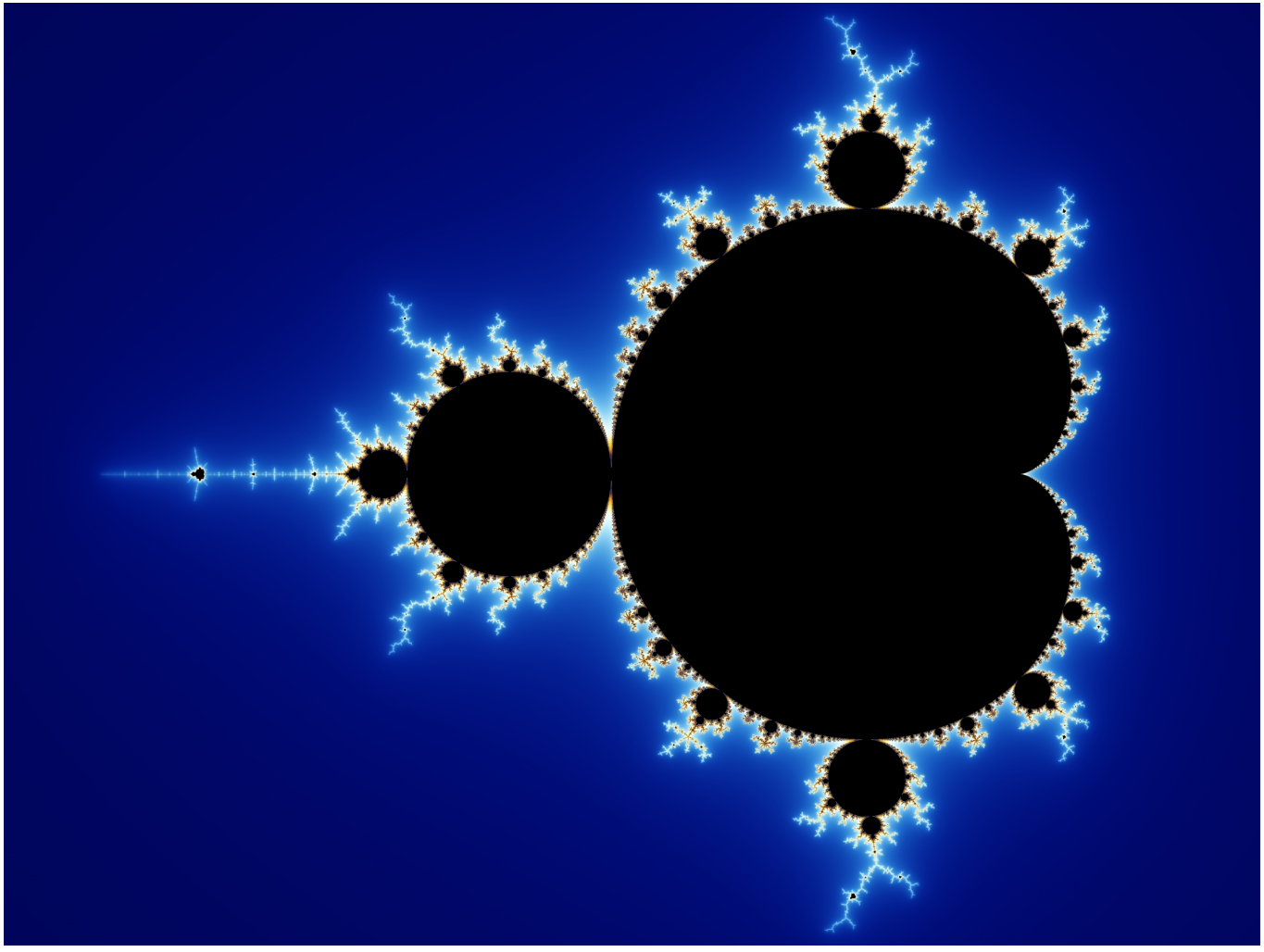
Klaus Sutner

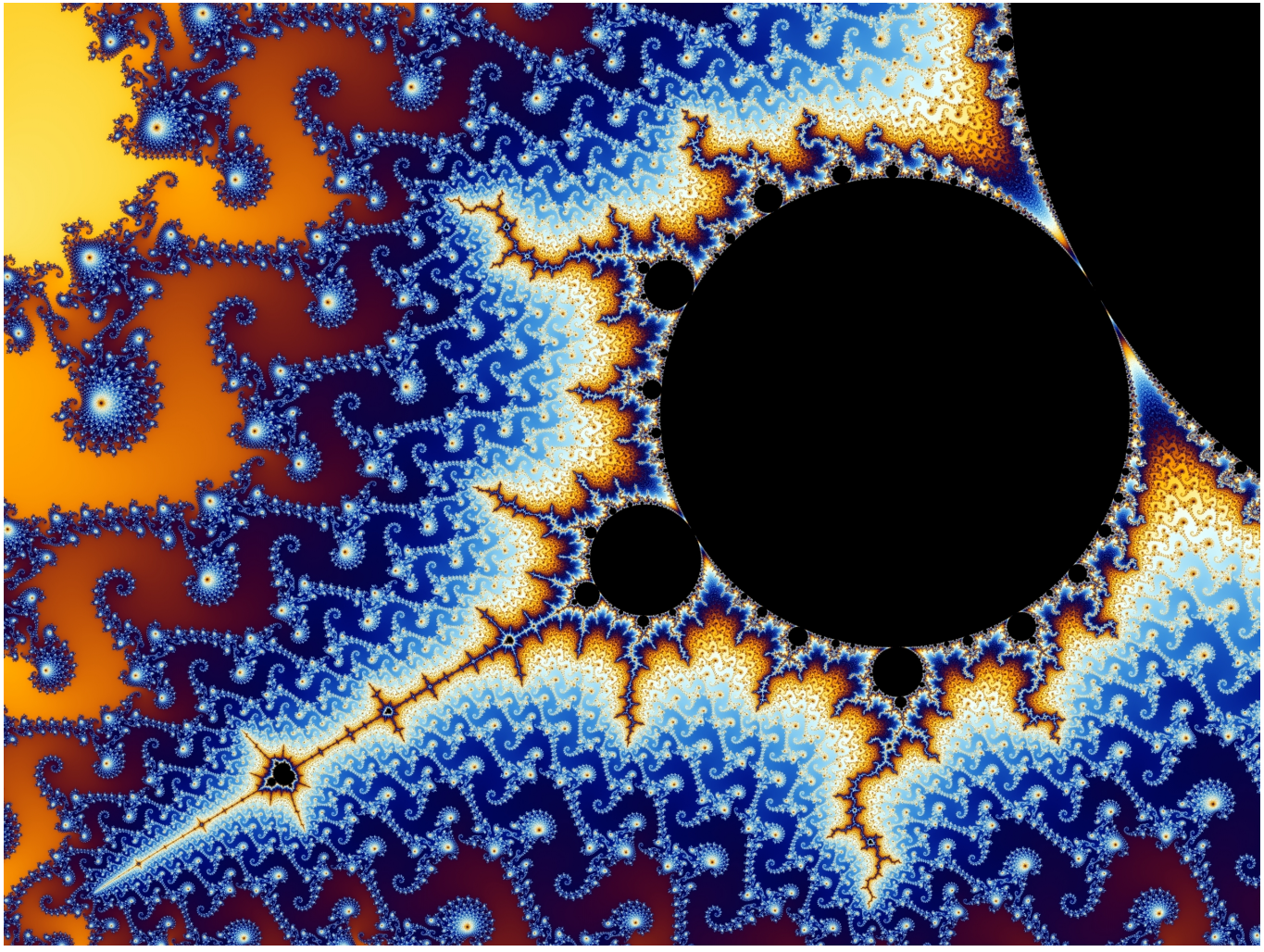
Carnegie Mellon University

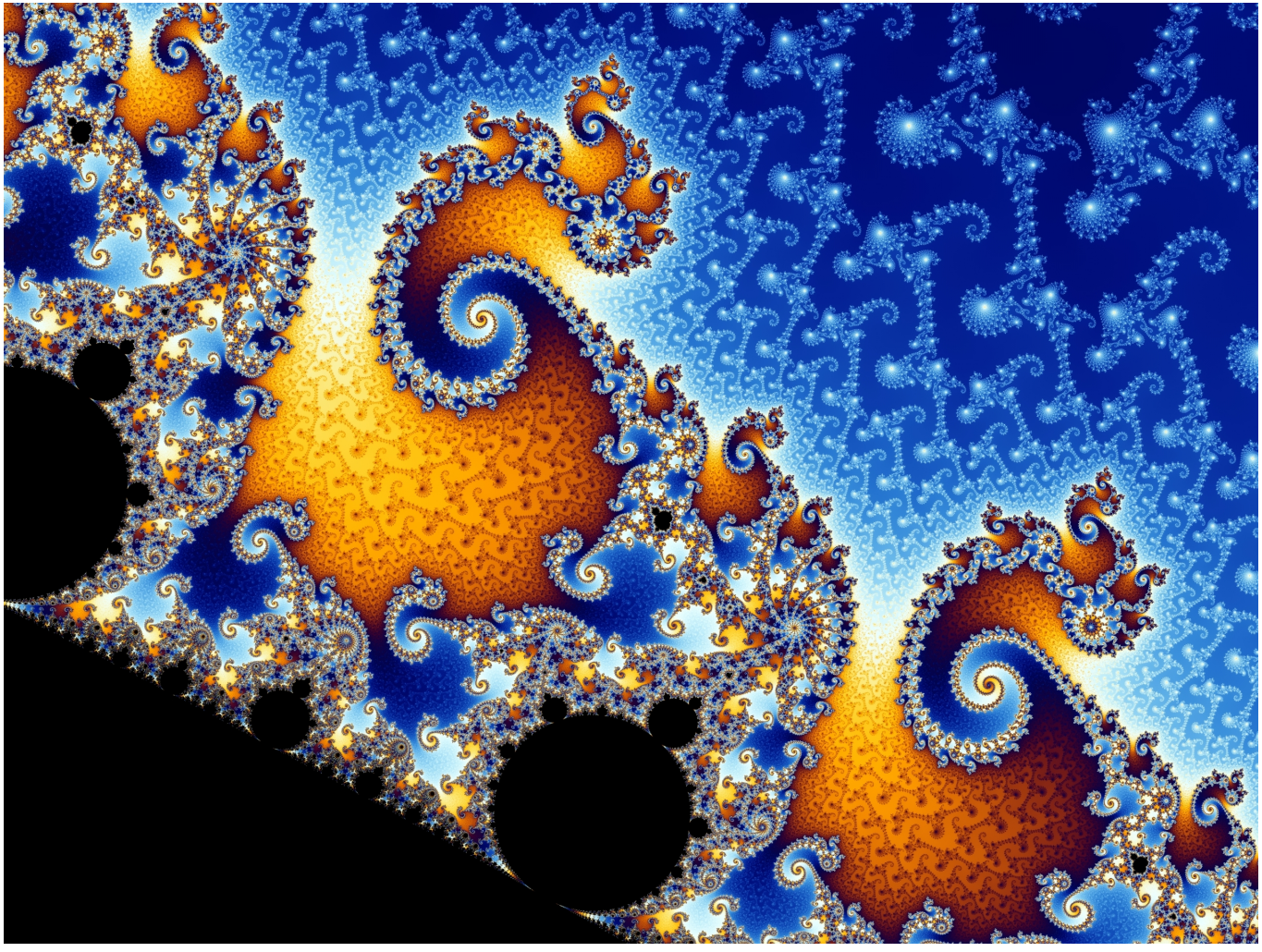
Recall that the driving idea behind recursion and iteration is self-similarity: the whole computation is highly similar to a part of itself.

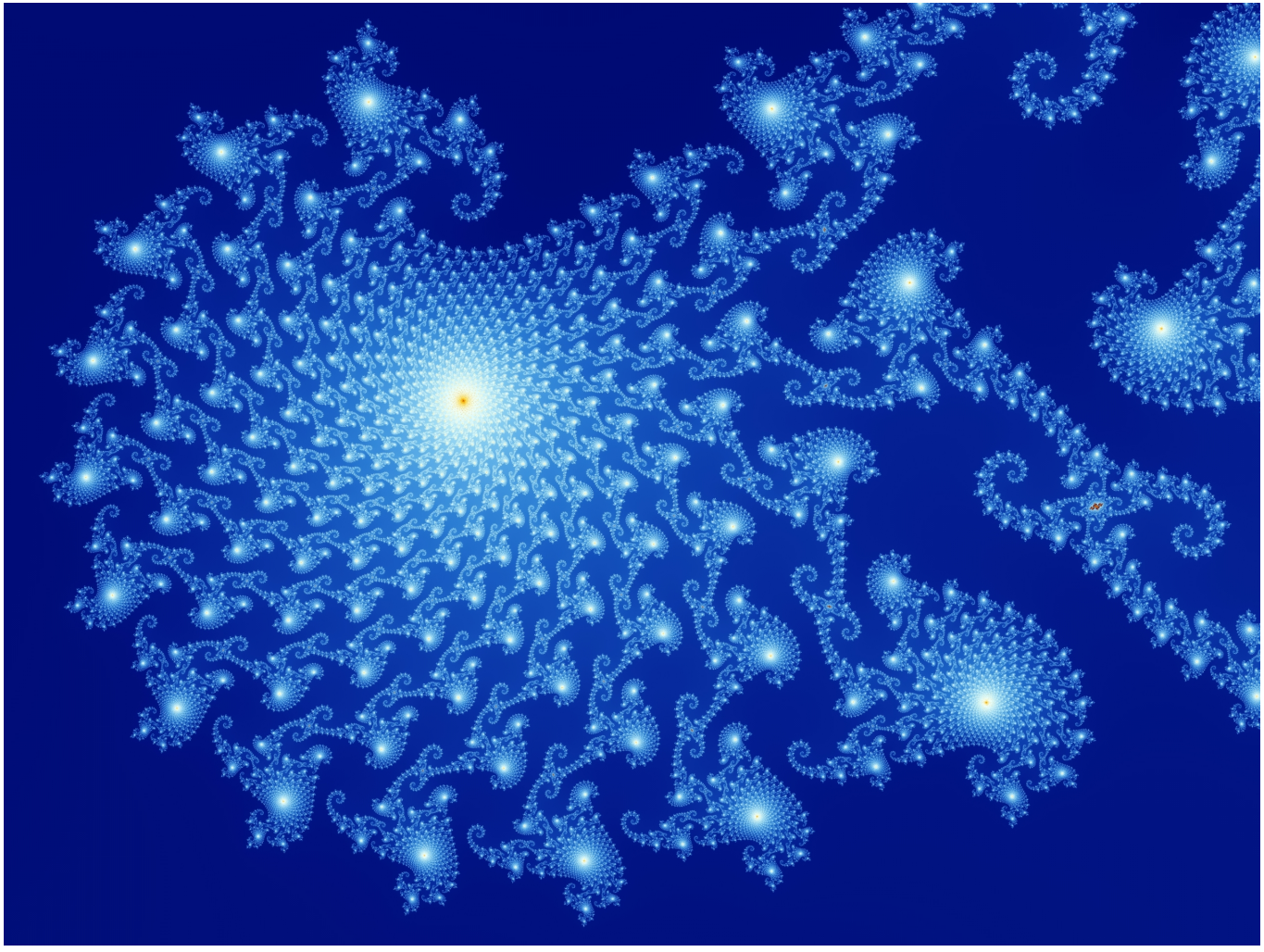
The notion of self-similarity is perhaps most compelling in geometry, there one can see how parts of a figure are similar to the whole.

Note that without computer-aided visualization none of the following pictures would exist, they cannot be drawn by hand.









Consider the second-degree complex polynomial

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

where $c \in \mathbb{C}$ is a constant.

Definition

$c \in \mathbb{C}$ belongs to the Mandelbrot set M if the orbit of 0 under p_c is bounded (as a sequence of complex numbers).

For example, $0 \in M$ but $1 \notin M$.

$c = i$ is also in: $i, -1+i, -i, -1+i, -i, -1+i, -i, -1+i, \ldots$

Likewise -1 and $-i$ are in.

And the Colors?

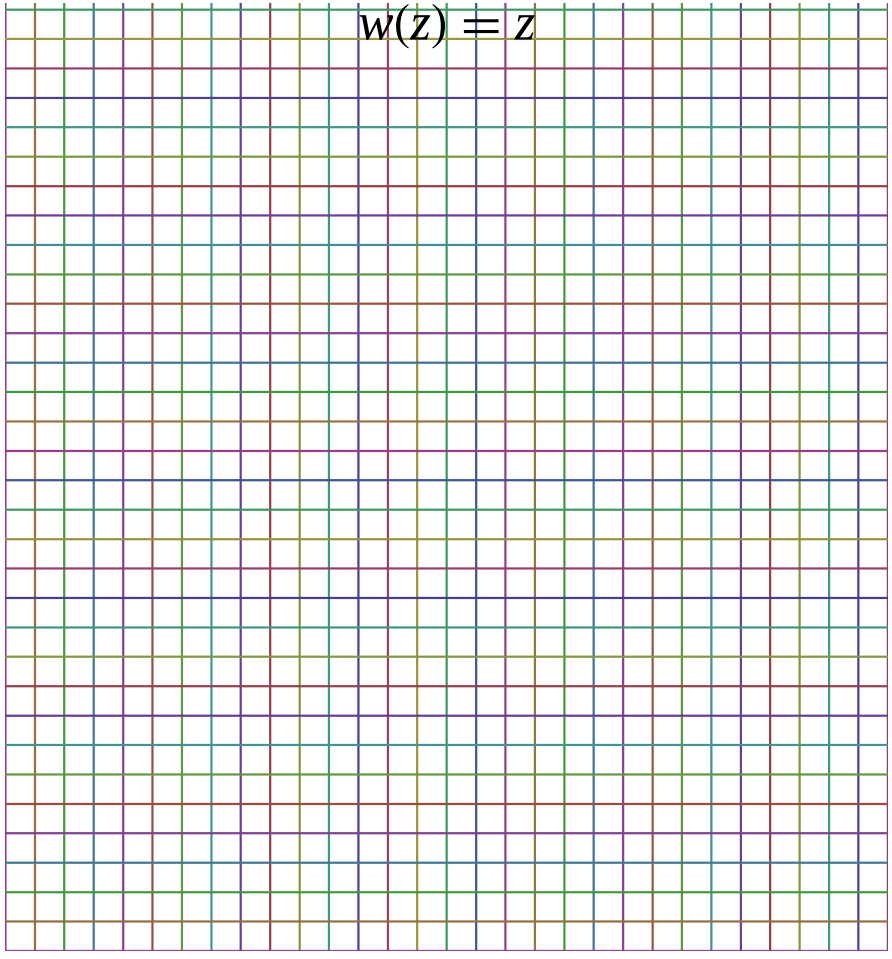
A priori, we only get a black and white picture: in M or out of M.

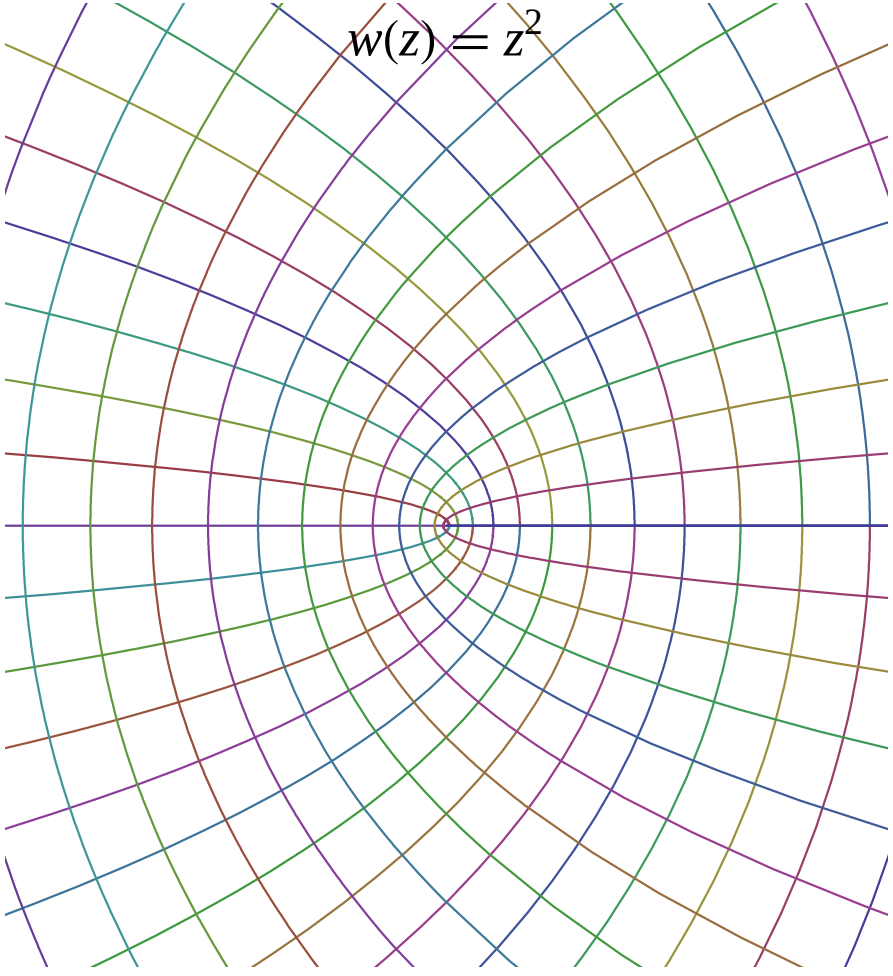
Note that it is not even clear how to do this much: there is no simple test to check if $|p_c^n(0)| \to \infty$ as $n \to \infty$.

In practice, one computes a few values $|p_c^n(0)|$ 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
http://en.wikipedia.org/wiki/Mandelbrot_set for some background.


$w(z) = z$


$w(z) = z^2$

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.

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.

Application: Reciprocal

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

Here

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

The point here is that we can express division in terms of multiplication and subtraction (simpler operations in a sense).

Example

Numerical values for $a = 1.41421356237309504880 \approx \sqrt{2}$.

0	1.00000000000000000000
1	0.58578643762690495119
2	0.68629150101523960950
3	0.70649403654862594510
4	0.70710625021159255130
5	0.70710678118614880890
6	0.70710678118654752440
7	0.70710678118654752440

Verify the result:

$$0.70710678118654752440 \times 1.41421356237309504880 = 1.00000000000000000000$$

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.

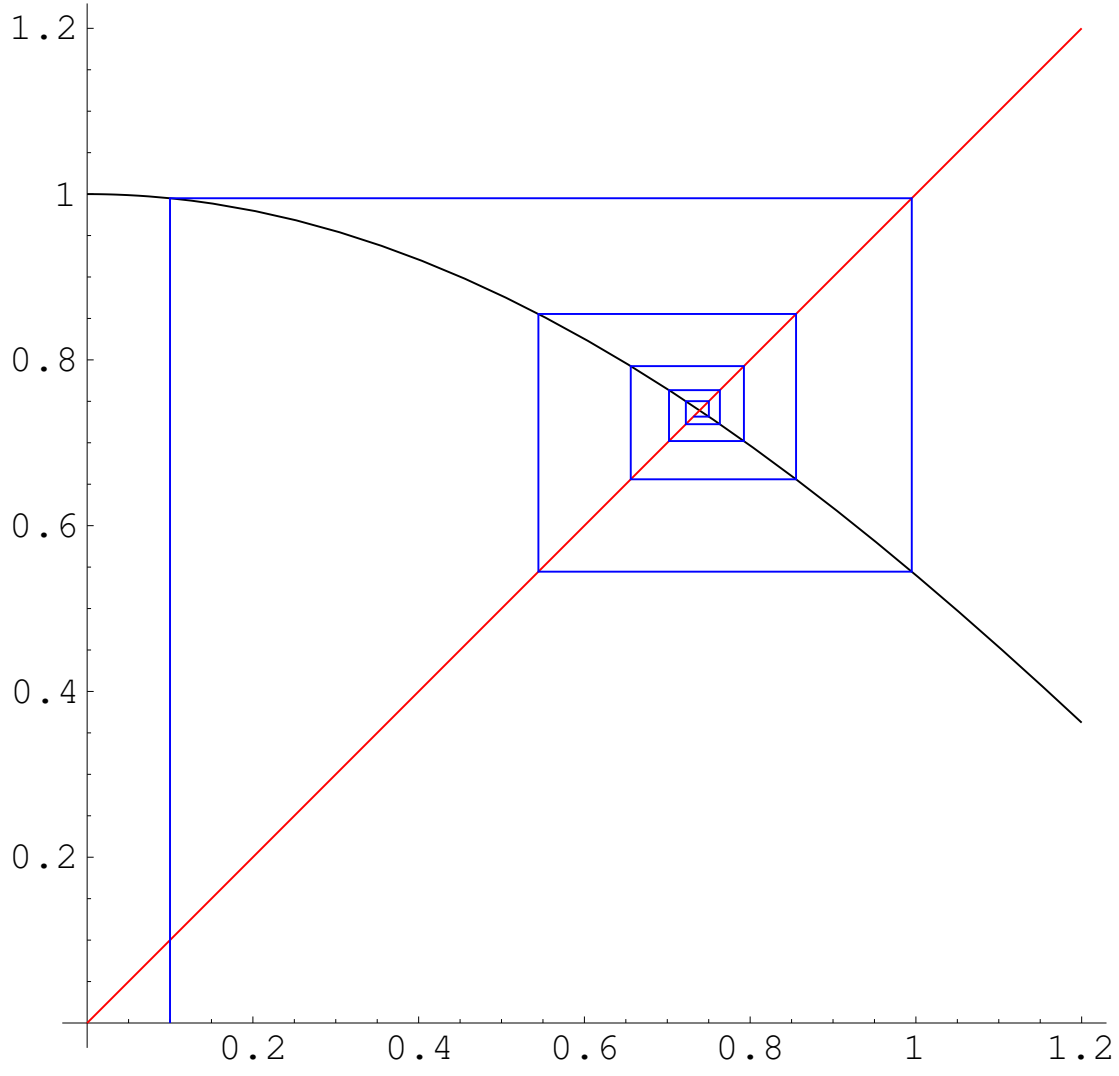
Moreover, a_n may be very close to a for reasonably small values of n so we can actually get our computational hands on a good approximation.

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:

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

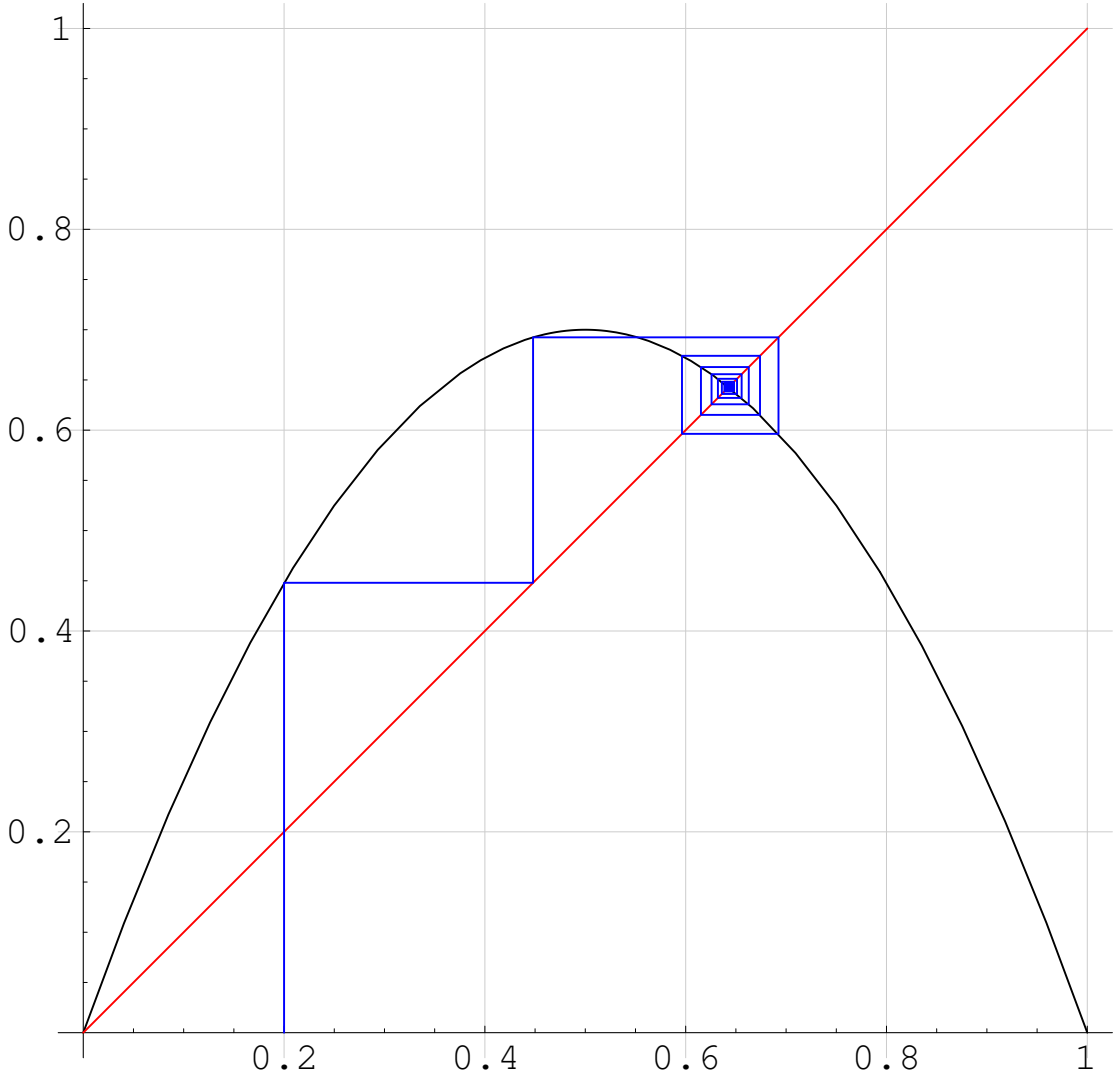
where $0 \leq p \leq 4$. Note that for these parameter values we have

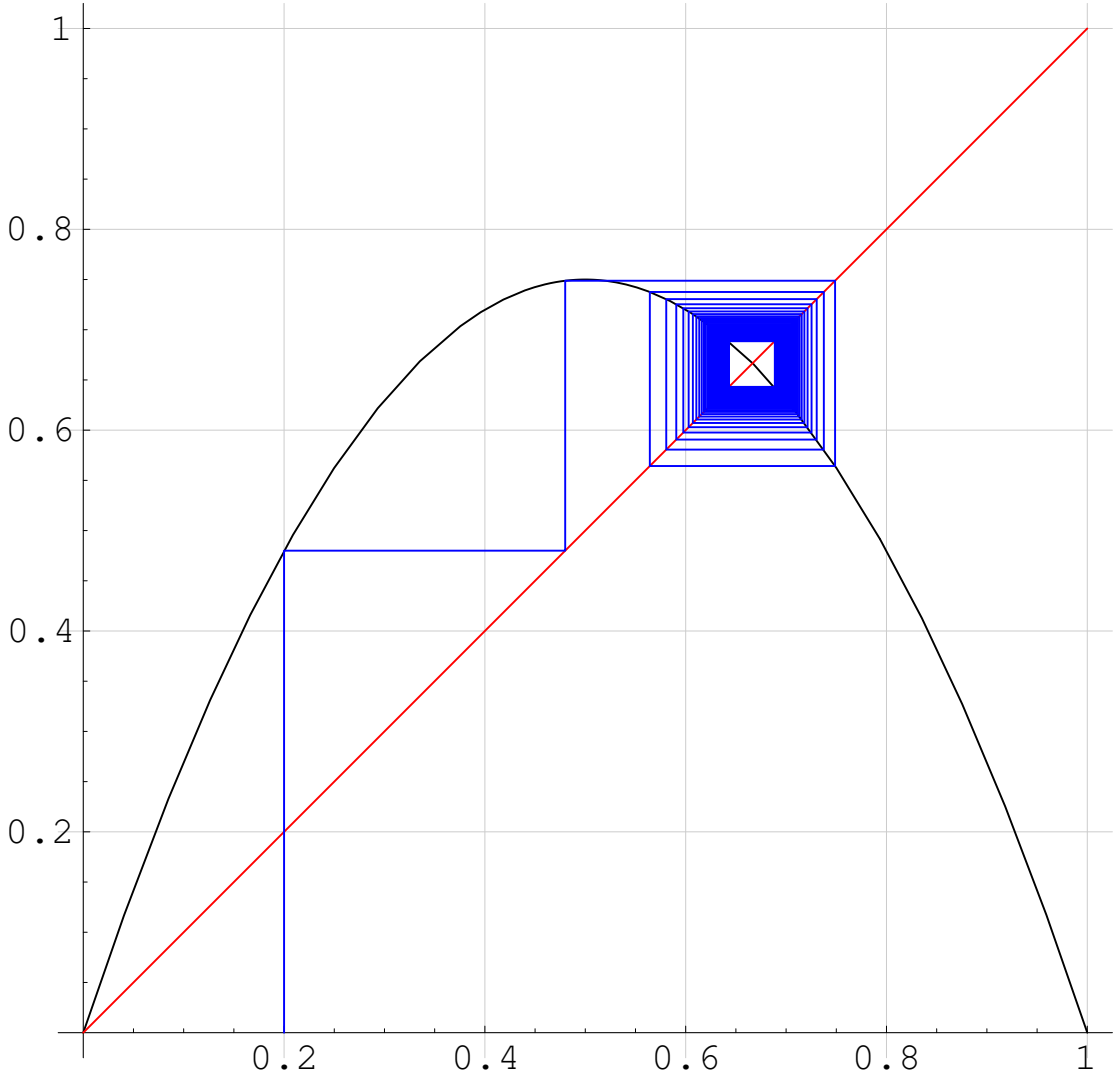
$$f_p : [0, 1] \to [0, 1]$$

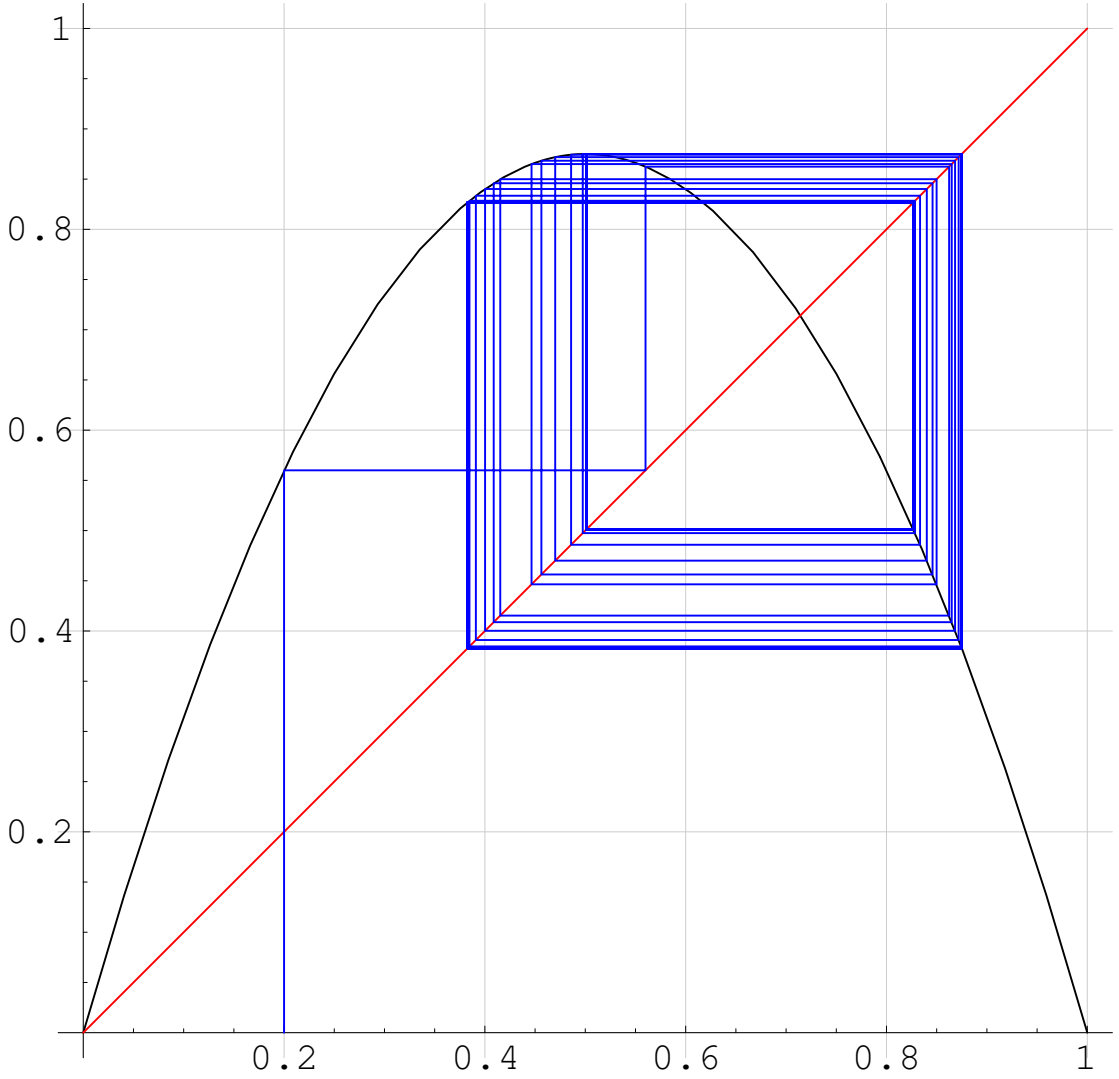
so we can actually iterate the map.

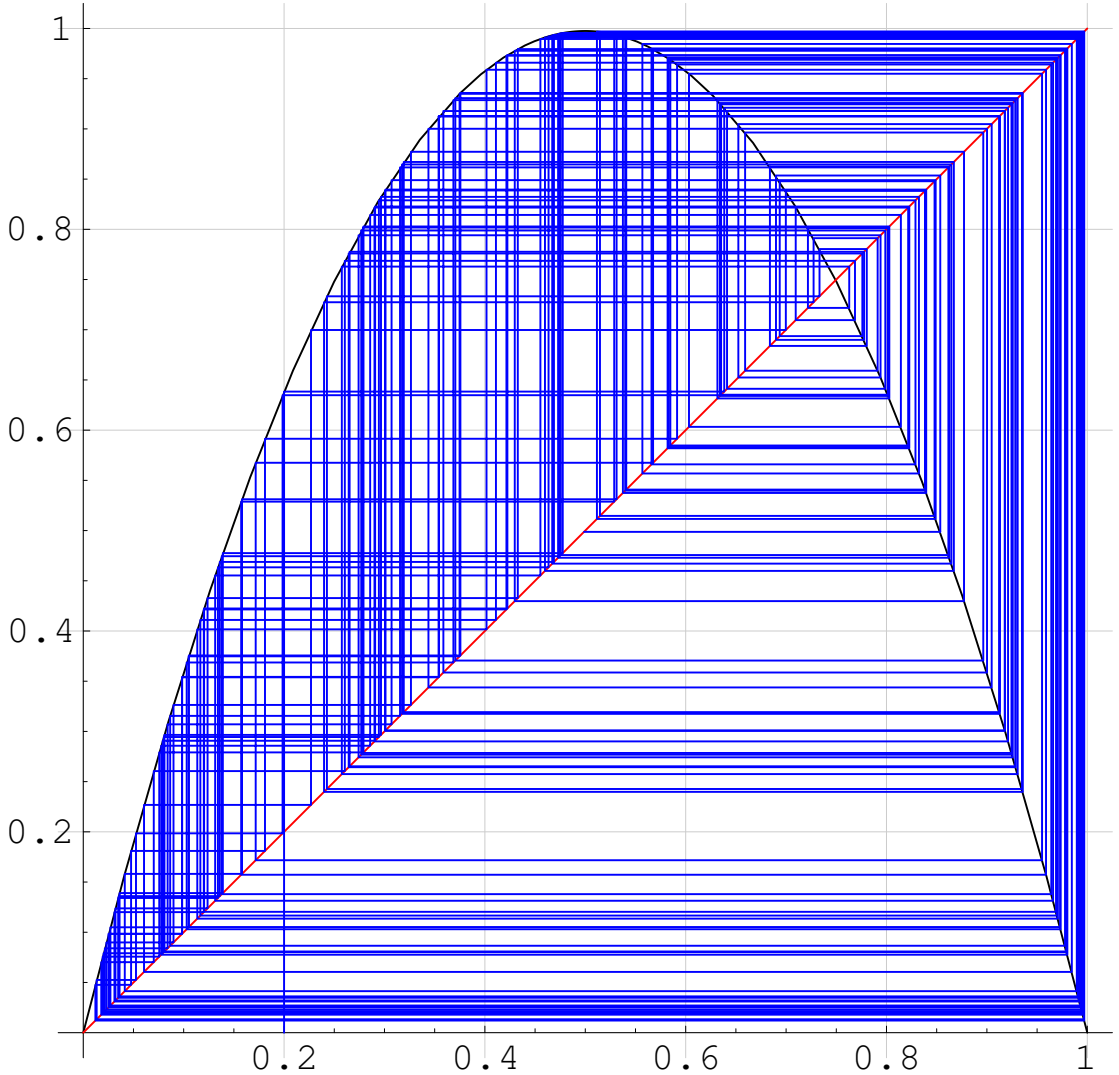
This is the so-called logistic map, a famous example in dynamics.

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

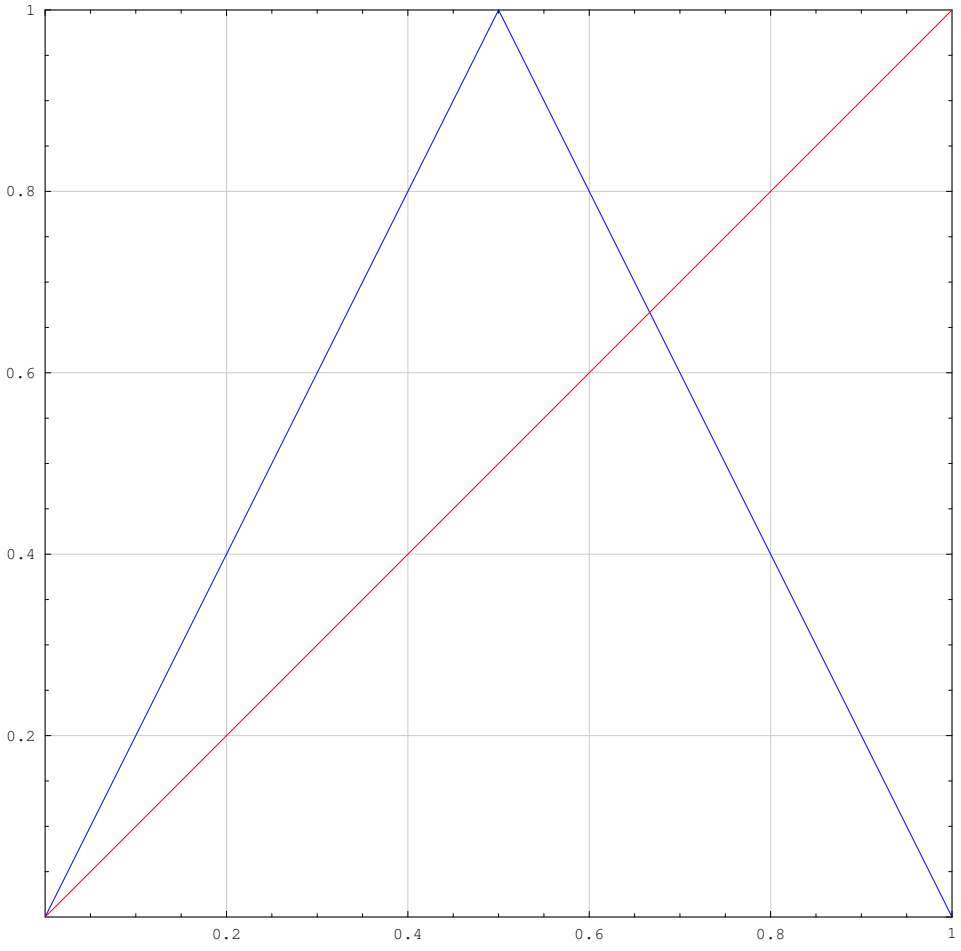

1
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1


1
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1

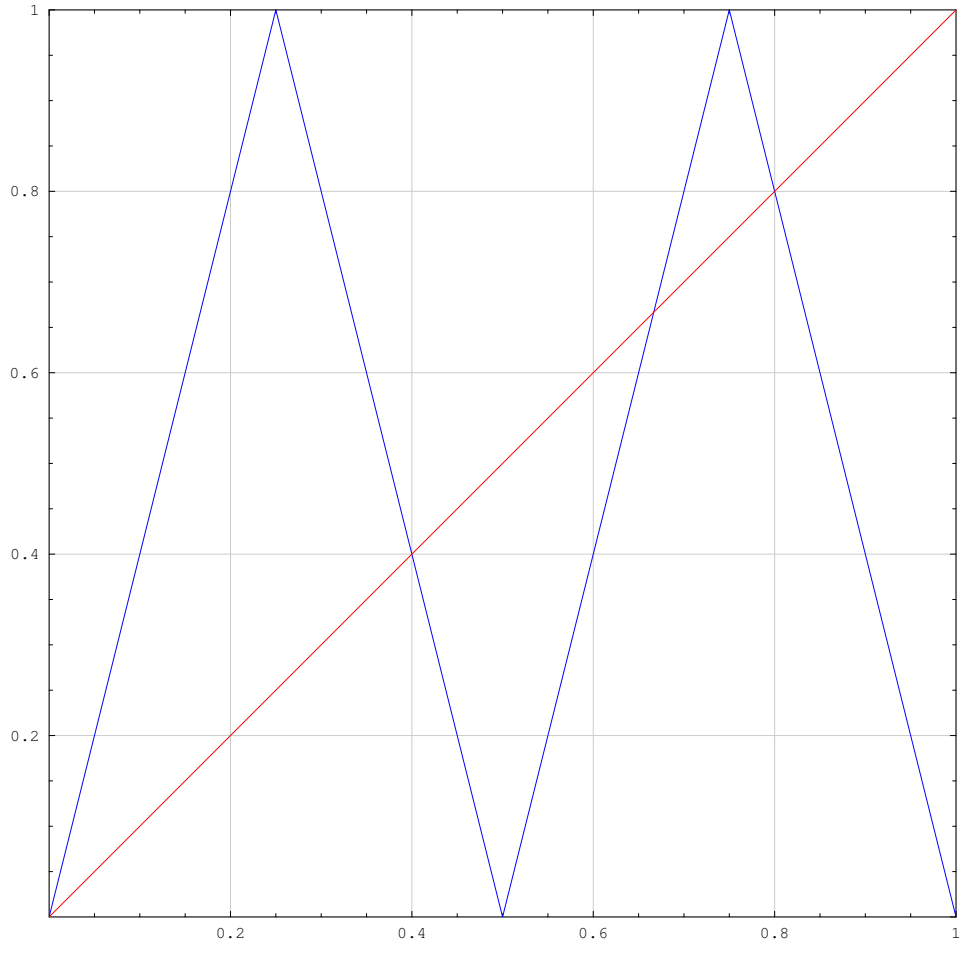

1
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1


1
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1

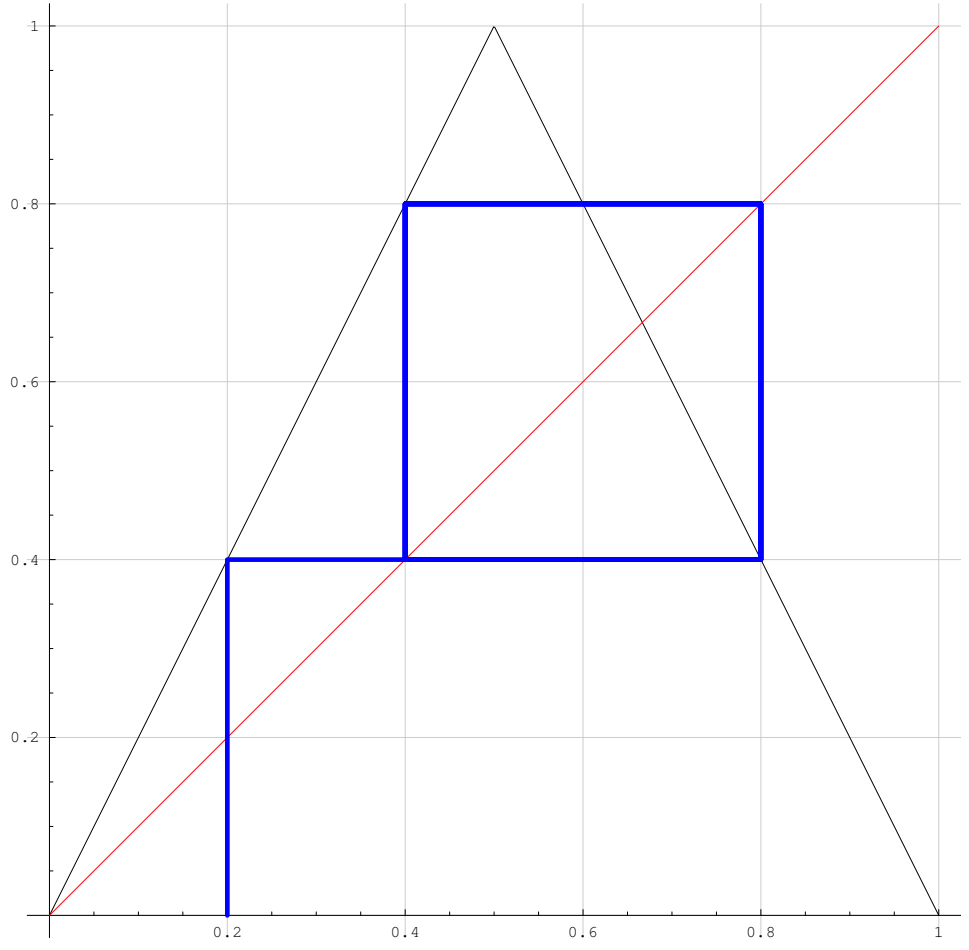
Differentiability is not necessary at all, piecewise linear will do.

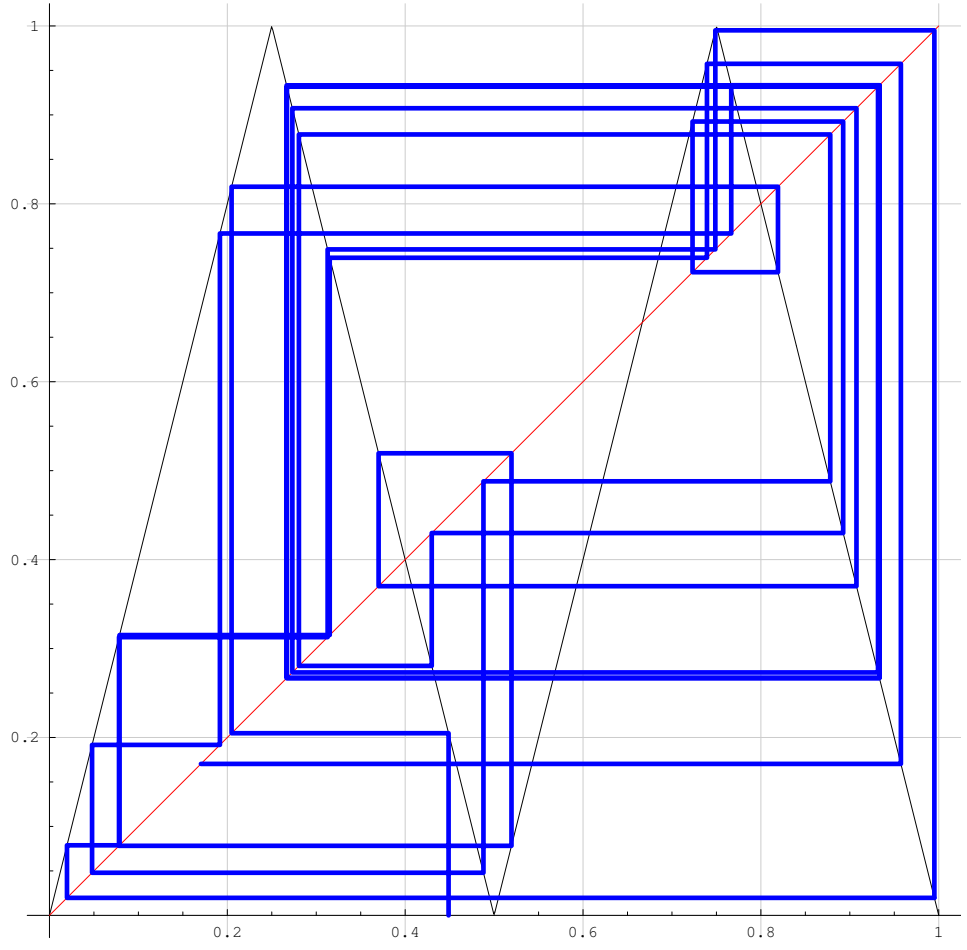


Tent Map Squared



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.

How about decidability issues related to iteration?

We have already seen two decision problems

Problem:	**Reachability Problem**
Instance:	A function $f : A \to A$, two points $a, b \in A$.
Question:	Is b in the orbit of a under f?

Problem:	**Confluence Problem**
Instance:	A function $f : A \to A$, two points $a, b \in A$.
Question:	Do the orbits of a and b under f overlap?

Actually, there are two versions: f may be fixed and only a and b change, or f may be part of the input.

When the carrier set A is finite the questions are trivially decidable and we even have clever algorithms to solve these problems based on Floyd's trick.

- In the infinite case the most interesting situation is when f is computable and when $A = \mathbb{N}$ or $A = \Sigma^\star$.
- Note that both problems are always semi-decidable in this case.
- However, in general both Reachability and Confluence are undecidable, even for fixed f.

The reason is that we can let f describe the one-step relation for register machines, coded up as on operation on suitable sequence numbers.

Then f is primitive recursive and in fact pretty simple. Note that we can safely assume that the RM erases all its registers before it halts, so it will halt iff it reaches the configuration $\langle q_H, \mathbf{0} \rangle$.

But then the question whether an given initial configuration $\langle q_0, x\mathbf{0} \rangle$ evolves to $\langle q_H, \mathbf{0} \rangle$ is undecidable.

Reachability and Confluence are clearly related:

$$\begin{aligned} \mathsf{reach}(x, y) &\to \mathsf{conf}(x, y) \\ \mathsf{conf}(x, y) &\leftrightarrow \exists\, z\, (\mathsf{reach}(x, z) \wedge \mathsf{reach}(y, z)) \end{aligned}$$

One might suspect that using one as an oracle one could solve the other, but it turns out that there is no simple computational link in general: there are cases where Reachability is undecidable but Confluence is decidable, and the other way round.

In fact, with some effort one can rig things up so that Reachability and Confluence are independent in their degree of difficulty within the class of semidecidable sets (alas, to make this precise requires a discussion of Turing degrees, see the notes).

If we use the standard coding of configurations of a Turing machine as words in

$$\Sigma^\star\, Q\, \Sigma^\star$$

then the one-step operation is very simple, it's really just word processing: we scan the word, copying until we hit $p \in Q$, then perform a small edit operation near p, and afterwards copy the rest of the word.

It is easy to see that this operation can be implemented in linear time and constant space.

Exercise

Give a careful description of the one-step operation as a string editing operation.

In essence, we are just dealing with the Halting Problem, for the umpteenth time.

But note that undecidability often is reflected in various easier versions of the problem still being difficult.

For example, consider a directed graph G on n points. There is a natural Reachability Problem for G: is there a path from a to b?

Of course, the problem is easily solvable: we can use, say, depth-first-search to check path existence in linear time.

But all standard algorithms require linear space, not just linear time. Logarithmic space suffices when we allow nondeterminism (more later) but in the deterministic case it seems to be too little.

The Collatz Conjecture is very easy to state:

Conjecture (Collatz)

All orbits of the Collatz function end in 1.

Unfortunately, though this problem is not included in the Clay challenge, some believe it to be enormously difficult.

> *Mathematics is not ready for this kind of problem.*
>
> *Paul Erdös*

Coming from one of the shining lights of 20th century discrete mathematics (Erdös number) this is discouraging.

Can We Use Computability?

Here is a desperate idea: Perhaps we could use our knowledge of computation to shed some light on this problem?

After all, the Collatz function is easily computable and we could build a small register machine that computes $C(x)$, or the stopping time function $\sigma(x)$.

Maybe we can use decidability or undecidability to tackle the problem?

Stopping Time is Computable

Recall the stopping time function $\sigma : \mathbb{N} \nrightarrow \mathbb{N}$:

$$\sigma(x) = \begin{cases} \min\big(t \mid C^t(x) = 1 \big) & \text{if } t \text{ exists,} \\ \uparrow & \text{otherwise.} \end{cases}$$

This version of σ is computable: just wrap a while-loop around the register machine computing C and add a counter. But note: the original version had ∞ as output in case of divergence; that one may not be computable).

Then the Collatz Conjecture is equivalent to σ being a total function.

Alas, it is undecidable whether a computable function is total; in fact this problem is worse than the Halting Problem, we would need access to $\emptyset''$.

So how do we turn Collatz into a decision problem?

Here is one fairly natural approach, really a version of Reachability.

Problem:	**Collatz Orbit Problem**
Instance:	A positive natural number x.
Question:	Does the orbit of x under C contain 1?

Observations:

- This problem is clearly semi-decidable.
- If the Collatz Conjecture is true, this problem is trivially decidable.
- Unfortunately, if this problem is decidable, the Collatz Conjecture may still be wrong (though the counterexamples are not terribly complicated: the set of all counterexamples is decidable).

Not too promising.

One might think that the right question to ask is this:

Problem:	**Wurzelbrunft's Collatz Problem**
Instance:	The Collatz function (or, if you prefer: a banana).
Question:	Is the Collatz Conjecture true?

Wurzelbrunft's Collatz problem is trivially decidable, albeit for entirely the wrong reasons.

The issue here is that there is only one instance.

For decidability, all we need is an algorithm that solves Fred's Collatz Problem.

No problem, here are two candidates:

- Algorithm 1: Ignore the input and output Yes.
- Algorithm 2: Ignore the input and output No.

One of those two algorithms solves this decision problem, we just don't know which.

Note that no one said that we need to be able to point out the algorithm explicitly, we just have to make sure an algorithm exists. But one of the two candidates above is guaranteed to work.

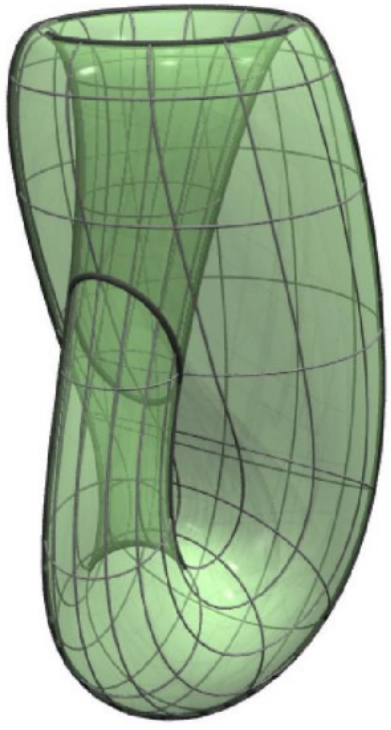
This type of argument caused a huge uproar in the mathematics community when first used by D. Hilbert in 1890 (finite basis theorem).

> "This is not mathematics, this is theology."
>
> Paul Gordon

Gordon was upset since he had found a constructive, computational proof for special case $n = 2$ in 1886, but had failed in all attempts to generalize the argument to arbitrary n.

Hilbert handled the general case not by explicitly computing a solution, but by showing that the assumption that there is no solution leads to a contradiction.

Felix Klein (of the eponymous bottle) strongly supported Hilbert's work. A while later, following suggestions of Klein and Gordon, Hilbert wrote a second paper showing how his approach can actually yield some bounds on the degree of the polynomials in question.



Gordon grudgingly concluded:

> "Theology may have its uses."

The same trick works for any decision problem with only finitely many instances $x_1, x_2, \ldots, x_n$:

This time there are 2^n algorithms that are potential solutions:

$$A_0, A_1, \ldots, A_{2^n-2}, A_{2^n-1}$$

We just may not know which of these algorithms is the right one.

But: The algorithm does exist, so the problem is decidable.

In other words, classical computability theory is ill-equipped to deal with finite decision problems: the definition just does not bite.

For problems with a single instance like the Collatz Conjecture, Riemann Hypothesis, $\mathbb{P} = \mathbb{NP}$ problem, and so on: Yes. They simply don't fit naturally into this framework.

There is an algorithm that returns the correct answer, but this is an entirely pointless observation since we have no clue which one it is.

However, in order to get useful information, we can try to exploit the fact that undecidability and unsolvability can cast a noticeable shadow. For example, Diophantine equations are not just undecidable in general, even individual equations are often very difficult to deal with.

Note that in practical computations one inevitably only deals with instances of some limited size. Here there are only finitely many instances and so, in a abstract computability sense, algorithms that deal with these are trivial.

Again, this is not a useful perspective: there is a lookup table that solves, say, Satisfiability for these size-limited instances. However, as far as realizable computation is concerned this table is just an illusion: it is

- usually absurdly large (say, larger than the physical universe), and
- for some problems we don't even know how to compute single entries in it efficiently.

It is much more productive to think of the problems as being truly infinite and develop methods that work on arbitrary inputs.

For the Collatz problem John Horton Conway, of Game-of-Life fame, found a beautiful way to show how undecidability lurks nearby.

Conway's Idea:
How about constructing infinitely many Collatz Conjectures?

More precisely, come up with a family of functions that generalize the Collatz function slightly. Then ask if for one of these functions all orbits are ultimately periodic.

The classical Collatz function will be just one in this family, so understanding the whole family would of course solve the Collatz problem.

The hard part is to come up with a nice natural class of "Collatz-like" functions. Here is Conway's approach: define

$$\mathrm{Cn}(\boldsymbol{a}, \boldsymbol{b})(q \cdot k + r) = a_r \cdot q + b_r$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are two vectors of numbers of length k and $0 \leq r < k$.

The classical Collatz function is the special case

$$k = 2, \boldsymbol{a} = (1, 6), \boldsymbol{b} = (0, 4).$$

So now we have infinitely many functions to deal with (though one of them is perhaps more interesting than all the others).

Problem:	**Conway-Collatz Problem**
Instance:	The parameters $\boldsymbol{a}$, $\boldsymbol{b}$.
Question:	Is every orbit of $\mathrm{Cn}(\boldsymbol{a}, \boldsymbol{b})$ ultimately periodic?

Theorem

The Conway-Collatz Problem is undecidable.
It remains undecidable even if all the b_i's are 0.

The theorem indicates that there is no good general way to answer questions about Collatz-like functions. So it is not entirely surprising that the classical Collatz function is also very difficult to analyze.

A famous example of a Conway function other than the classical Collatz function is the following:

$$T(2n) = 3n$$
$$T(4n+1) = 3n+1$$
$$T(4n-1) = 3n-1$$

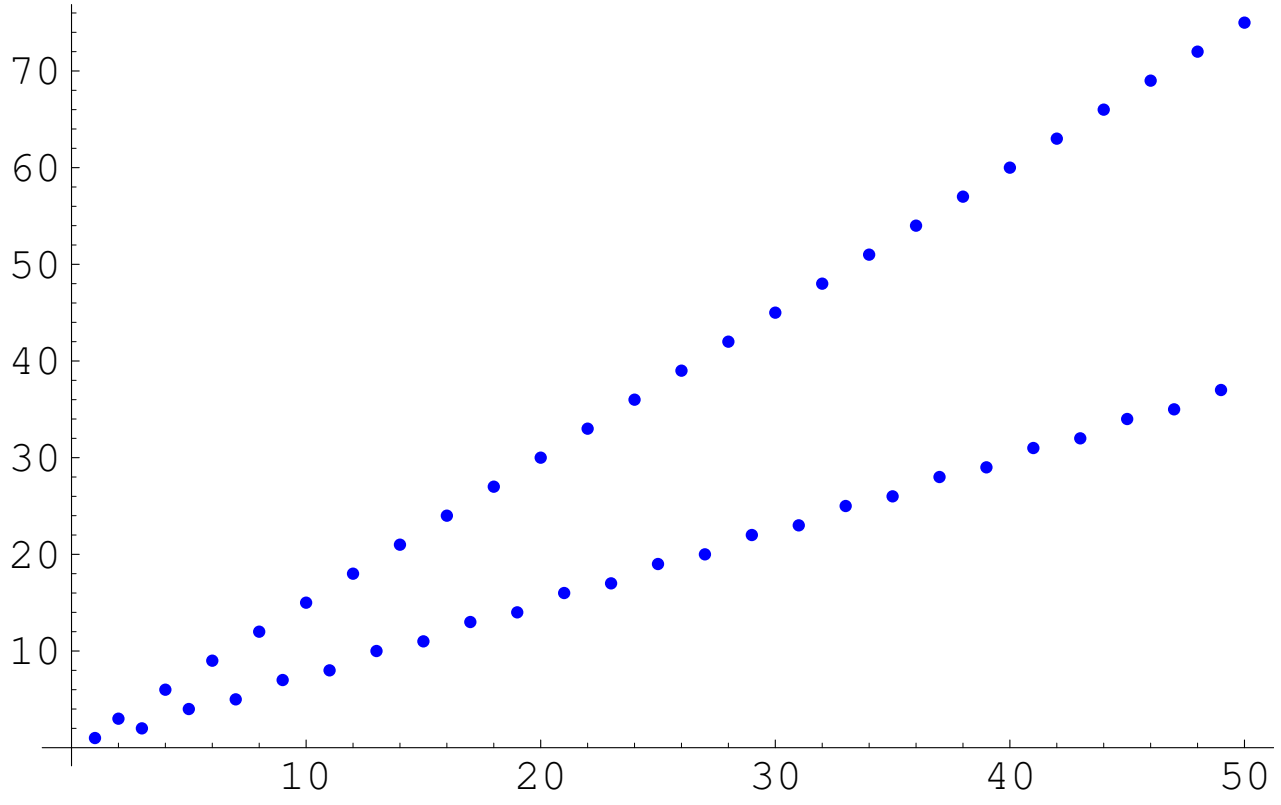
This is just $\mathsf{Cn}(6,3,6,3;0,3,1,2)$.

Proposition

$T : \mathbb{N} \to \mathbb{N}$ is a bijection.

Exercise

Prove that the T function is a bijection. Then look for cycles under T.



Looks very similar to the Collatz function.

But note that the lower line wobbles; there are really 3 linear functions here.

Some Cycles

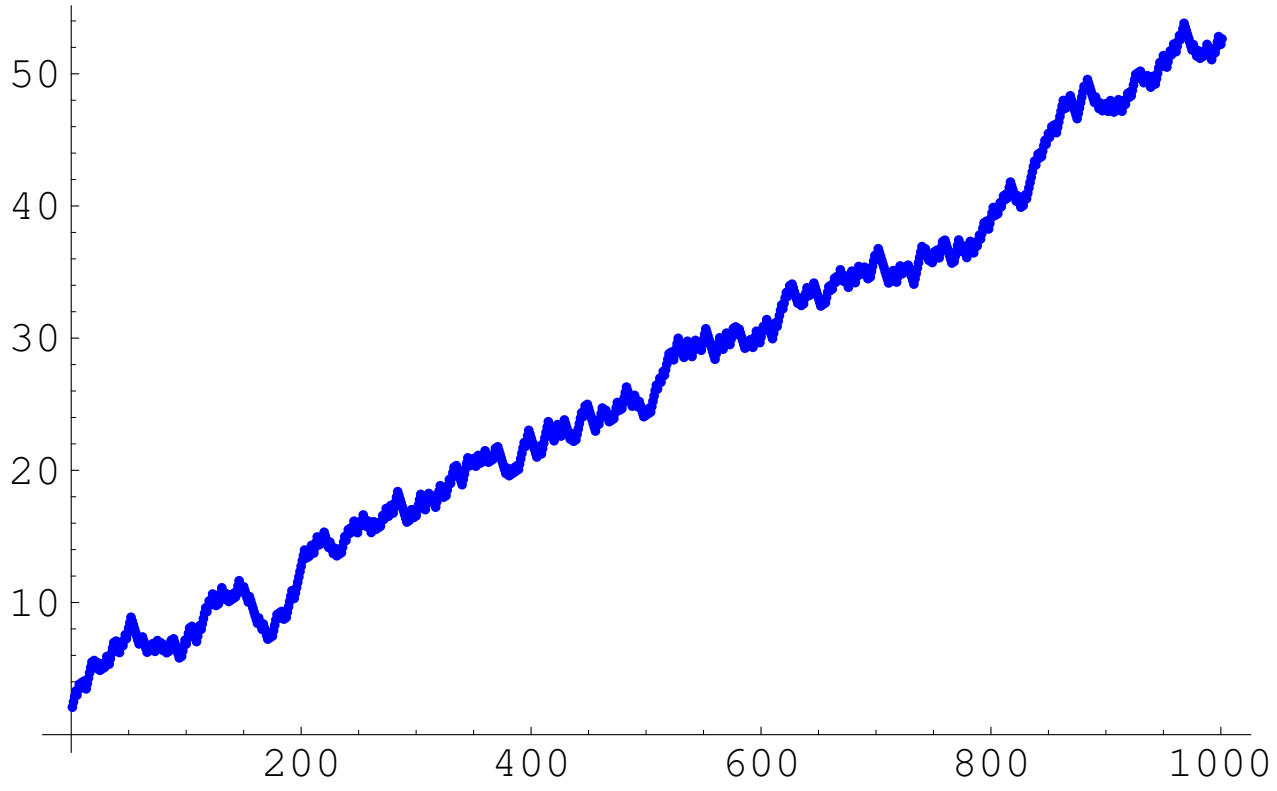
Known finite cycles are:

$$(1),$$
$$(2, 3),$$
$$(4, 6, 9, 7, 5),$$
$$(44, 66, 99, 74, 111, 83, 62, 93, 70, 105, 79, 59).$$

Open Problem:

Are there any other finite orbits?
In particular, is the orbit of 8 finite?

Orbit of 8



This is a log-plot. It seems to suggest the orbit of 8 grows without bound, but of course this is neither here nor there; maybe the values decrease after $A(100, 100)$ steps, A the Ackermann function.

The original Collatz problem is finite, so the classical notion of decidability is useless.

But there is another way one can try make sense out of the notion of a finite problem being "undecidable."

- First, one fixes some formal system $\mathcal{T}$ (which is presumably adequate to talk about the problem at hand, something like Peano arithmetic or Zermelo-Fraenkel set theory).
- Then one shows that $\mathcal{T}$ can neither prove nor refute the claim that the problem at hand has a positive solution.

Classical example: the Continuum Hypothesis or the Axiom Of Choice in set theory.

In the world of continuous functions there are a few well-known results that guarantee the existence of a fixed point.

Theorem (Brouwer)

Any continuous map from the closed unit sphere in dimension n to itself has at least one fixed point.

Theorem (Banach)

Every contraction map on a complete metric space has exactly one fixed point.

To obtain comparable results in the discrete domain requires a bit of preparatory work. We need to find the right algebraic structure (just as complete metric spaces are the right structure for Banach's theorem).

Definition

A lattice is an algebraic structure $\mathcal{A} = \langle A, \sqcap, \sqcup \rangle$ with two binary operations, referred to as meet and join that are both associative, commutative and idempotent. Moreover, the absorption law holds:

$$x \sqcap (x \sqcup y) = x \sqcup (x \sqcap y) = x$$

Example

Boolean values true and false with logical connectives “and” and “or” form a lattice.

Example

The powerset of any set with operations intersection and union form a lattice.

Example

Binary relations on a set form a lattice with intersection and union. Likewise, equivalence relations form a lattice, albeit with a different meet operation.

Example

The positive integers with $\gcd$ and lcm form a lattice.

Another way to look at lattices is to consider a poset $\langle A, \leq \rangle$.

If the poset is properly behaved, we can define binary operations

$$\inf(x, y) = \max\big(z \in A \mid z \leq x, y \big)$$
$$\sup(x, y) = \min\big(z \in A \mid x, y \leq z \big)$$

Then $\langle A, \inf, \sup \rangle$ is a lattice.

One the other hand, given a lattice we can define a partial order by

$$x \leq y \iff x \sqcap y = x.$$

This partial order has sups and infs which turn out to be exactly the join and meet operations of the lattice.

In a lattice, the join and meet operations are binary by definition. Of course, they can be generalized to a finite number of arguments, but there is a useful requirement that sups and infs should exist for arbitrary subsets of A.

Definition

A lattice is complete if every subset of the carrier set has a supremum and an infimum.

Note that every complete lattice must have a least and a largest element, usually written $\bot$ and $\top$.

Example

Every finite lattice is complete.
The infinite examples from above are all complete.
The reals with standard order form an incomplete lattice.

However, the closed interval $[0, 1] \subseteq \mathbb{R}$ with the standard order is a complete lattice.

In fact, this property is the whole point of the real numbers, it enables calculus and life on Mars. Otherwise we could live happily ever after with the rationals.

Exercise

Generalize this result to d dimensions.

Exercise

The natural numbers with divisibility form a complete lattice.

In analysis, the existence of fixed points is a difficult topic. In our setting, there is a very powerful theorem that often can be used to demonstrate the existence of fixed points.

Definition

Let L be a complete lattice and $f : L \to L$ a map. f is monotonic if for all $x, y \in L$:

$$x \leq y \quad \text{implies} \quad f(x) \leq f(y)$$

So monotonic simply means order preserving. Given a monotonic map one can always define an increasing chain

$$\bot \leq f(\bot) \leq f^2(\bot) \leq \ldots$$

Theorem (Knaster-Tarski 1955)

Let L be a complete lattice and $f : L \to L$ a monotonic map on L. Then the set of fixed points of f is a complete sublattice of L.

It follows from the theorem that fixed points exist; there may even be many of them.

Moreover, there is always

- a least fixed point μf and
- a largest fixed point νf.

To see that there is a largest fixed point define the set of all *semi-fixed-points* elements

$$I = \{\, x \in L \mid x \leq f(x) \,\}$$

Note that $\bot \in I$, $f(I) \subseteq I$ and I contains all fixed points of f.

Let $s = \sup I$ (which exists by completeness).

Then for $x \in I$ we have $x \leq s$ whence $f(x) \leq f(s)$ by monotonicity and so $x \leq f(x) \leq f(s)$. But then $f(s)$ bounds I and s itselft must be a semi-fixed-point, whence $s \in I$. But then $f(s) = s$ and we have $\nu f = s$.

The least fixed point can be gotten by duality from the largest.

It is worth while repeating the argument in the special case where $f : \mathfrak{P}(A) \to \mathfrak{P}(A)$, in which case the whole argument comes down to basic set theory.

Say, we want to demonstrate the existence of a least fixed point. This time, define

$$I = \{ X \subseteq A \mid f(X) \subseteq X \}$$

and let $B = \bigcap I$ (which exists since $A \in I$). Then $f(B) \subseteq B$: for any semi-fixed-point X, $B \subseteq X$ by definition, hence $f(B) \subseteq f(X) \subseteq X$; thus $f(B) \subseteq B$ and $B \in I$.

But I is closed under f: $f(X) \subseteq X$ implies $f(f(X)) \subseteq f(X)$. So $f(B) \in I$, and thus $B \subseteq f(B)$. But then B is a fixed point, and necessarily the least.

Another important way to obtain the least fixed point is to approximate it from below: the least fixed point of f is the sup of the chain

$$\mu f = \sup(f^n(\bot) \;\middle|\; n \geq 0)$$

At least in the case where the lattice is finite, this produces an actual algorithm.

For example, minimization of DFAs can be interpreted this way.

Suppose we have a binary relation ρ on A.

Define the following operation f for any binary relation X on A:

$$f(X) = \rho \cup I_A \cup X^{-1} \cup (X \bullet X)$$

The lattice of binary relations on A is clearly complete, and f is monotonic: $X \leq Y$ implies $f(X) \leq f(y)$.

Then $\mu f = \bigcup_{n<\omega} f^n(\emptyset)$ is the equivalential closure of ρ. The closure ordinal is ω since the chains required by transitivity are all of finite length.

Lemma

Let $f : A \to B$ and $g : B \to A$ be two arbitrary functions. Then there are partitions $A = A_1 \cup A_2$ and $B = B_1 \cup B_2$ such that $f(A_1) = B_1$ and $g(B_2) = A_2$.

Proof. We exploit the Knaster-Tarski theorem. Consider the powerset of A, clearly a complete lattice.

For any $X \subseteq A$ define

$$F(X) = A - g(B - f(X))$$

A moment's thought reveals that F is indeed monotonic and thus has a least fixed point.

But then $A_1 = \mu F$ works as required (make sure to verify this fact). □

Note that the Cantor-Bernstein theorem is a simple corollary to Banach's lemma: when f and g are both injective we have $|A_1| = |B_1|$ and $|A_2| = |B_2|$.

This argument is considerably less complicated than the standard proof of Cantor-Bernstein.

Exercise

Try to prove Banach's lemma from scratch, without reference to the Knaster-Tarski theorem. Try some special cases first, say, f maps to one point, is surjective, and so on.

Exercise

Look up some standard proofs for Cantor-Bernstein. How do they compare to the proof above?

Summary

- Iteration produces complicated behavior even in simple functions.
- It matters little whether the domain is discrete or continuous.
- Many algorithms can be construed as fixed point constructions.
- Some computational environments such as Mathematica offer a fixed point operation as a primitive.
- There is a memoryless linear time algorithm to compute transient and period of a map on a finite carrier set.
- Questions about the orbits of functions may easily be undecidable, even if the functions in question are very simple.
- There are results in the discrete realm that are similar to classical fixed point theorems in analysis, but they require a bit of machinery.