Chaos and Fractals

Chaos and Fractals

Mark McClure
University of North Carolina at Asheville

October 27, 2021

Preface

This is an introductory text on the basics of chaotic dynamics and fractal geometry. The intended audience includes undergraduate students of mathematics or other technical disciplines with a strong background in calculus and some exposure (or willingness to explore) more advanced topics such as the basics of real analysis, the complex plane, and linear algebra.

Topics in the text include real iterative dynamics (experimentation, cobweb plots, bifurcation, and chaos), complex iterative dynamics (Julia sets, the Mandelbrot set, and the iteration of higher order polynomials), fractal geometry (self-similarity, iterated function systems and fractal dimension).

Another important aspect of this text is its emphasis on computation. We introduce Python code that runs live in the online version to generate images of many of the sets that we'll meet. Such code is generally quite simple using basic constructs such as function definition, conditionals, and iterative loops.

This text was written with PreTeXt which makes groovy things like live Python code and nicely formatted online and print versions easy.

Contents

Chapter 1

Introduction

One of the central ideas in both chaos theory and fractal geometry is iteration - i.e., the repitition of a process over and over again. It might seem a bit odd to base so much of our discussion on a process that doesn't neccessarily appear to be of fundamental importance to the unintiated. Thus, in this introduction, we start with a motivation for iteration - namely, Newton's method for solving equations. From that natural beginning, we'll meet many of the important ideas that we'll study further in text.

1.1 Surprise in Newton's method

Newton's method is a technique to estimate roots of a differentiable function. Invented by Newton in 1669, it remains a stalwart tool in numerical analysis. When used as intended it's remarkably efficient and stable. After a little experimentation, Newton's method yields some surprises that are very nice illustrations of chaos.

1.1.1 The basics of Newton's method

Let's begin with a look at the example that Newton himself used to illustrate his method. Let $f(x) = x^3 - 2x - 5$. The graph of f shown in Figure 1.1.1 seems to indicate that f has a root just to the right of $x = 2$.

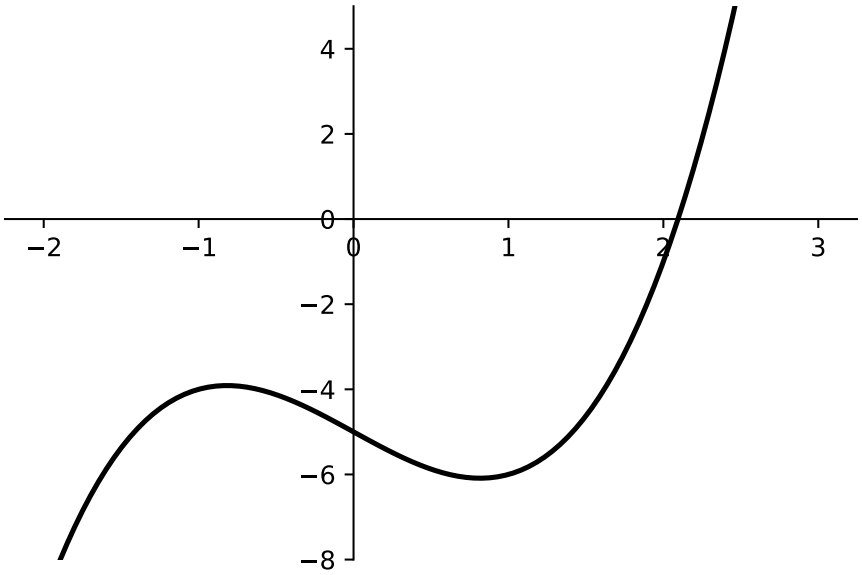

Figure 1.1.1 Newton's example function for his method

Newton figured that, if the root of f is a little bigger than 2, then he could write it as $2 + \Delta x$. He then plugged this into f to get

$$\begin{aligned} f(2+\Delta x) &= (2+\Delta x)^3 - 2(2+\Delta x) - 5 \\ &= 8 + 3 \times 2^2 \Delta x + 3 \times 2\Delta x^2 + \Delta x^3 - 4 - 2\Delta x - 5 \\ &= -1 + 10\Delta x + 6\Delta x^2 + \Delta x^3 \\ &\approx -1 + 10\Delta x. \end{aligned}$$

In that last step, since Δx is small, he figures that higher powers of Δx are negligibly small. Thus, he figures that $-1 + 10\Delta x \approx 0$ so that $\Delta x \approx 1/10$.

The point is that, if 2 is a good guess at a root of f, then $2 + 1/10 = 2.1$ should be an even better guess. A glance at the graph seems to verify this. Of course, we could then repeat the process using 2.1 as the initial guess. We should get an *even better* estimate. The process can be repeated as many times as desired. This is the basis of *iteration*.

1.1.2 Newton's method in the real domain

Let's take a more general look at Newton's method. The problem is to estimate a root of a real, differentiable function f, i.e. a value of x such that $f(x) = 0$. Suppose also that we have an initial guess x_0 for the actual value of the root, which we denote r. Often, the value of x_0 will be based on a graph. Since f is differentiable, we can estimate r with the root of the tangent line approximation to f at x_0; let's call this point x_1. When x_0 is close to r, we often find that x_1 is even closer. This process is illustrated in Figure 1.1.2 where it indeed appears that x_1 is much closer to r than x_0.

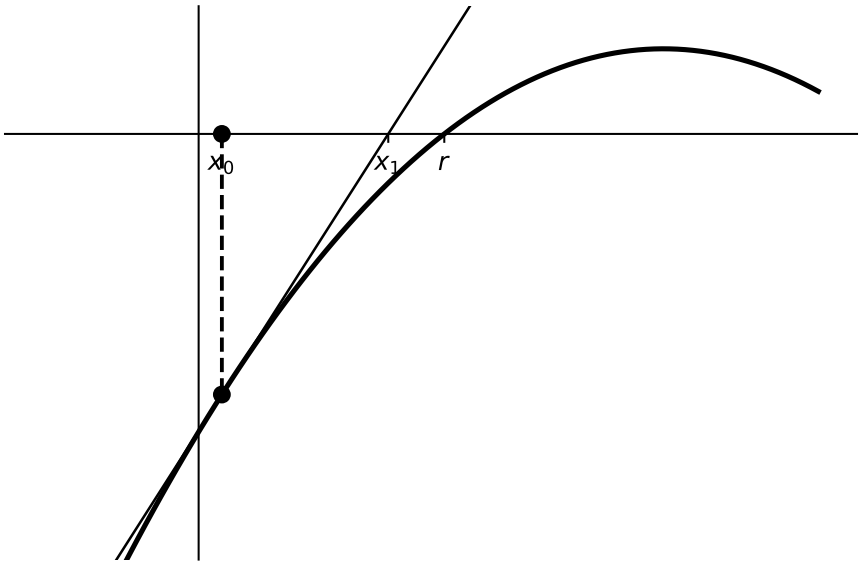

Figure 1.1.2 One step in Newton's method

We now find a formula for x_1 in terms of the given information. Recall that x_1 is the root of the tangent line approximation to f at x_0; let's call this tangent line approximation ℓ. Thus,

$$\ell(x) = f(x_0) + f'(x_0)(x - x_0)$$

and $\ell(x_1) = 0$. Thus, we must simply solve

$$f(x_0) + f'(x_0)(x - x_0) = 0$$

for x to get $x_1 = x_0 - f(x_0)/f'(x_0)$.

Now, of course, x_1 is again a point that is close r. Thus, we can repeat the process with x_1 as the guess. The new, better estimate will then be

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}.$$

The process can then repeat. Thus, we can define a sequence (x_n) *recursively* by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

This process is called *iteration* and the sequence it generates often converges to the root r.

1.1.2.1 Examples

We now present several examples to illustrate the variety of things that can happen when we apply Newton's method. Throughout, we have a function f and we defined the corresponding Newton's method iteration function

$$N(x) = x - \frac{f(x)}{f'(x)}.$$

We then iterate the function N from some starting point or, perhaps, from several starting points.

Example 1.1.3 We start with $f(x) = x^2 - 2$. Of course, f has two roots, namely $\pm\sqrt{2}$. Thus, we might think, of the application of Newton's method to f as a tool to find good approximations to $\sqrt{2}$.

First, we compute N:

$$\begin{aligned} N(x) &= x - f(x)/f'(x) = x - \frac{x^2-2}{2x} \\ &= x - \left(\frac{x^2}{2x} - \frac{2}{2x}\right) = x - \left(\frac{x}{2} - \frac{1}{x}\right) = \frac{x}{2} + \frac{1}{x}. \end{aligned}$$

Now, suppose that $x_0 = 1$. Then,

$$\begin{aligned} x_1 &= N(1) = \frac{1}{2} + \frac{1}{1} = \frac{3}{2} \\ x_2 &= N(3/2) = \frac{3/2}{2} + \frac{1}{3/2} = \frac{17}{12} \\ x_3 &= N(17/12) = \frac{17/12}{2} + \frac{1}{17/12} = \frac{577}{408} \end{aligned}$$

Note that

$$(577/408)^2 = 332929/166464 = 2 + 1/166464$$

so that third iterate is quite close to $\sqrt{2}$.

Note that we've obtained a *rational* approximation to $\sqrt{2}$. At the same time, it's clear that it would be nice to perform these computations on a computer. In that context, we might generate a *decimal* approximation to $\sqrt{2}$. Here's how this process might go in Sage:

```
f(x) = x^2 - 2
N(x) = x - f(x)/diff(f(x),x)
xi = 2.0
for i in range(7):
    xi = N(xi)
    print(xi)
```

```
1.50000000000000
1.41666666666667
1.41421568627451
1.41421356237469
1.41421356237310
1.41421356237309
1.41421356237310
```

Note how quickly the process has converged to 12 digits of precision.

Of course, f has two roots. How can we choose x_0 so that the process converges to $-\sqrt{2}$? You'll explore this question computationally in Exercise 1.3.1. It's worth noting, though, that a little geometric understanding can go a long way. Figure 1.1.4, for example, shows us that if we start with a number x_0 between zero and $\sqrt{2}$, then x_1 will be larger than $\sqrt{2}$. The same picture shows us that any number larger than $\sqrt{2}$ leads to a sequence that converges to $\sqrt{2}$.

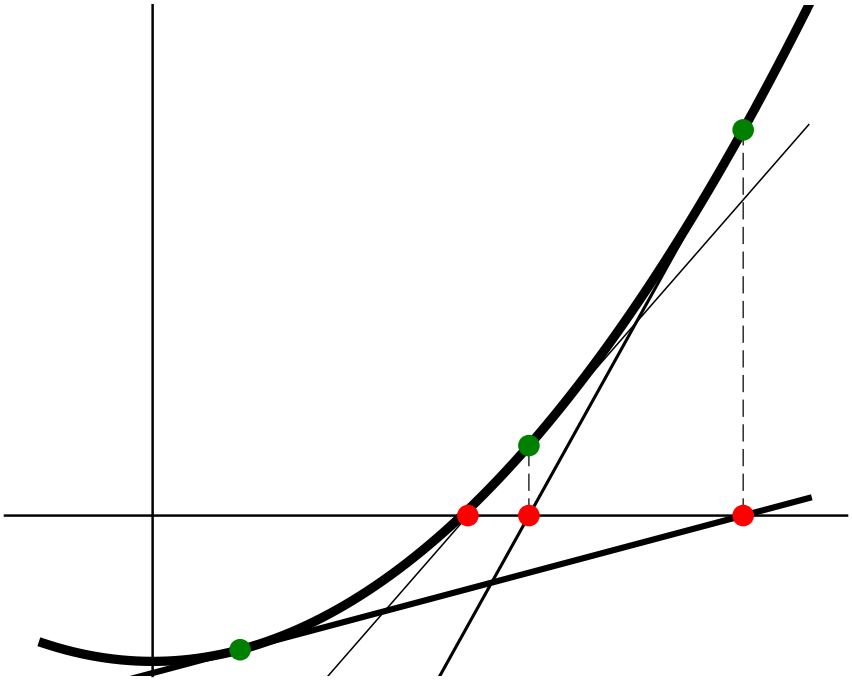

Figure 1.1.4 Three steps in Newton's method for $f(x) = x^2 - 2$

□

Example 1.1.5 We now take a look at $f(x) = x^2 + 3$. A simple look at the graph of f shows that it doesn't even hit the x-axis; thus, f has no roots. It's not at all clear what to expect from Newton's method.

A simple computation shows that the Newton's method iteration function is

$$N(x) = \frac{x}{2} - \frac{3}{2x}.$$

Note that $N(1) = -1$ and $N(-1) = 1$. In the general context of iteration that we'll consider later, we'll say that the points 1 and -1 lie on an orbit of period 2 under iteration of the function N.

Sequences of other points seem more complicated so we turn to the computer. Suppose that we change the initial seed $x_0 = 1$ just a little tiny bit and iterate with Sage.

```
f(x) = x^2 + 3
N(x) = x - f(x)/diff(f(x),x)
xi = 2.0
for i in range(100):
    xi = N(xi)
    print(xi)
```

```
Long crazy list of random-looking numbers
```

Well, there appears to be no particular pattern in the numbers. In fact, if we generate 1000 iterates an plot those that lie within 10 units of the origin on a number line, we get Figure 1.1.6. This is our first illustration of chaotic behavior. Not just because we see points spread all throughout the interval but also because we appeared to have a stable orbit when $x_0 = 1$. Why should the behavior be so different when we change that initial seed to $x_0 = 0.99$?

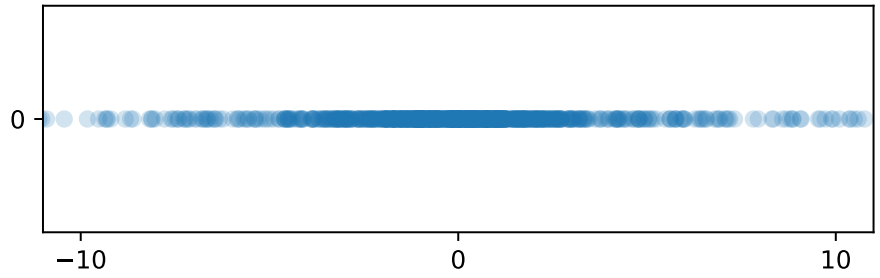

Figure 1.1.6 Chaotic behavior from Newton's method

□

Example 1.1.7 Newton's original example had one real root, Example 1.1.3 had two real roots and Example 1.1.5 had no real roots. Let's take a look at an example with *lots* real roots, namely $f(x) = \cos(x)$.

Generally, the closer your initial seed is to a root, the more likely the sequence starting from that seed is to converge to that root. What happens, though, if we start some place that's not so close to a root? What if we start close to the maximum - near zero? Let's investigate in code.

```
seed(1)
N(x) = x + cot(x)
for j in range(10):
    xi = random()/10
    for i in range(8):
        xi = N(xi)
    print([xi,cos(xi)])
```

```
23.5619449019235, 8.57871740039736e-16]
[14.1371669411541, 5.51091059616309e-16]
[87432.0943457307, -3.22999223447169e-12]
[32.9867228626928, -4.90477700295530e-16]
[67.5442420521806, -4.40934745756706e-15]
[108.384946548848, 2.44867461765812e-15]
[58.1194640914112, 4.89239739022353e-16]
[29.8451302091030, 6.12942380210265e-16]
[36.1283155162826, -3.18470065841971e-15]
[14.1371669411541, 5.51091059616309e-16]
```

OK, let's pick this code apart. The inner for loop looks like so:

```
xi = random()/10
for i in range(8):
    xi = N(xi)
```

Thus, xi is set to be a random number between 0 and 0.1; the for loop then iterates the Newton's method function in for the cosine from that initial seed 8 times. The outer for loop simply performs this experiment 10 times. After each run of the experiment, we print the resulting xi - along with the value of the cosine at that point, to check that we're indeed close to a root of the cosine. It's striking that we get 9 different results over 10 runs even though the starting points are so close to one another.

Figure 1.1.8 gives some clue as to what's going on. Recall that we can envision a Newton step for a function f from a point x_i by drawing the line that is tangent to the graph of f at the point $(x_i, f(x_i))$. The value of x_i is then the point of intersection of this line with the x-axis. Because the slope at

the maximum is zero, the value of this point of intersection is very sensitive to small changes. In fact, there are infinitely many roots of the cosine any one of which could be hit by some initial seed in this tiny interval.

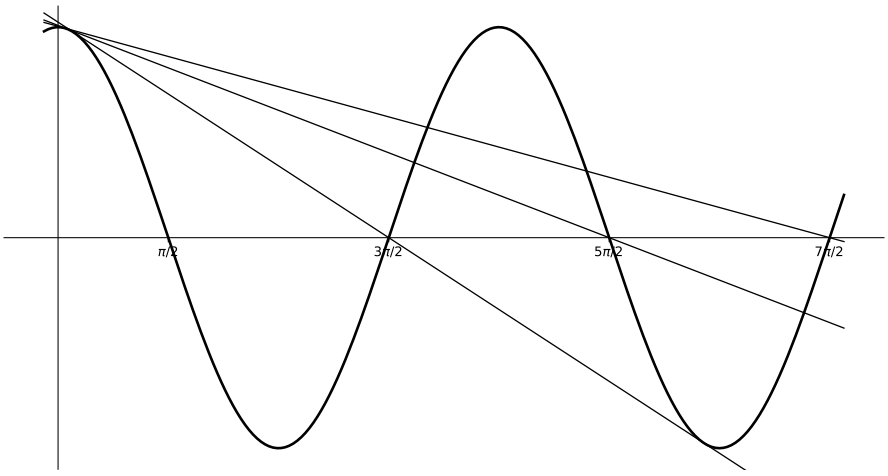

Figure 1.1.8 Initial Newton steps for the cosine

□

1.1.3 Newton's method in the complex domain

Let's move now to the complex domain, where even crazier things can happen. To understand this stuff, of course, you'll need a basic understanding of complex numbers but it's really not a daunting amount of information. You'll need to know that a complex number z has the form

$$z = x + iy,$$

where x and y are real numbers (the real and imaginary parts of z) and i is the imaginary unit (thus $i^2 = -1$). You'll also need to know (or accept) that you can do arithmetic with complex numbers and plot them in a plane, called the complex plane. We'll discuss complex variables in a more detail when we jump into complex dynamics more completely.

1.1.3.1 Cayley's question

In the 1870s, the prolific British mathematician Arthur Cayley (whose name is all over abstract algebra) posed the following interesting question: Given a Suppose that p is a complex polynomial and z_0 is an initial seed that we might use for Newton's method. Generally, p will have several roots scattered throughout the complex plane. Is it possible to tell to which of those roots (if any) Newton's method will converge to when we start at z_0?

Generally, the closer an initial seed is to a root, the more likely that seed will lead to a sequence that converges to the root. It might be reasonable to guess that the process always converges to the root that is closest to the initial seed. Cayley proved that this is true for quadratic functions but was commented that the general question is much more difficult for cubics. With the power of the computer, there is a very colorful way to explore the question

Algorithm 1.1.9 Coloring basins of attraction for Newton's method.

1. *Given: A function* $f : \mathbb{C} \to \mathbb{C}$.
2. *Choose a rectangle* R *in the complex plane with sides parallel to the real and imaginary axes.*
   - *The points in* R *represent potential initial seeds when applying Newton's method to* f.
3. *Discretize the rectangle into points of the form*
$$z_{0,jk} = (a + j\Delta x) + (b + k\Delta y)k.$$
4. *For each point* $z_{0,jk}$ *perform Newton's iteration to generate a sequence* $(z_{n,jk})_n$ *until one of two things happen:*
   - $|f(z_{n,jk})| < \varepsilon$, *where* ε *is some pre-specified small number. We assume that the process has converged to a root; color the initial seed* $z_{0,jk}$ *according to which root the process converged to and shade the initial seed according to how many iterates this process took.*
   - *The iteration count exceeds some pre-specified bailout; we color the initial seed* $z_{0,jk}$ *black.*

Example 1.1.10 Let $p(z) = z^2 + 3$. We already played with this function in the real case back in Example 1.1.5 and the computer indicated that there might be chaos on the real line. When we allow complex inputs, though, there are two roots - one at $\sqrt{3}i$ (above the real axis) and one at $-\sqrt{3}i$ (below the real axis). If the process always converges to the root that is closest to the initial seed, then seeds in the upper half plane should converge to $\sqrt{3}i$ while seeds in the lower half plane should converge to $-\sqrt{3}i$. We don't have the tools to prove this yet, but we can investigate with Algorithm 1.1.9. The result is shown in Figure 1.1.11.

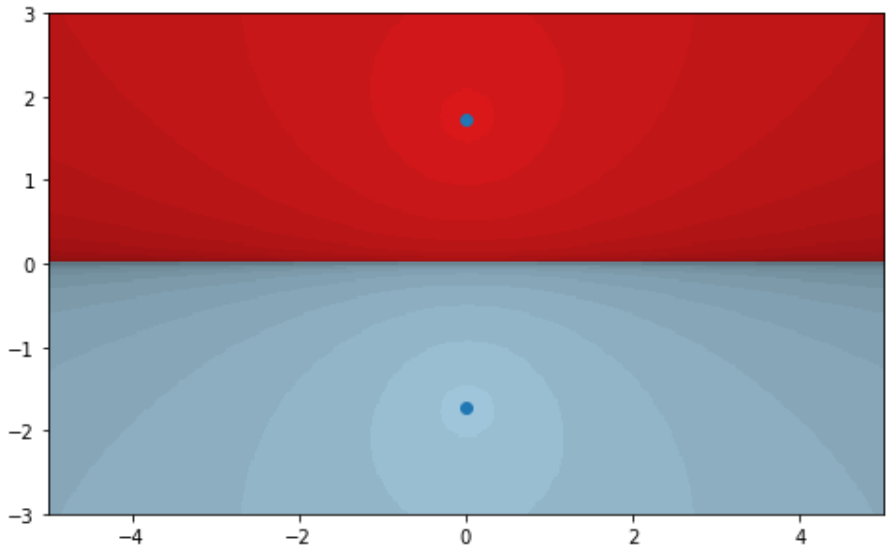

Figure 1.1.11 Attractive basins for $f(z) = z^2 + 3$

□

Example 1.1.12 Let $p(z) = z^3 - 1$. As a complex, cubic polynomial, we expect p to have three roots. It's easy to see that $z = 1$ is a root. This means that $z - 1$ is a factor, which makes it easier to find that

$$p(z) = (z - 1)(z^2 + z + 1).$$

We can then apply the quadratic formula to find that the other two roots are

$$\frac{-1 \pm \sqrt{-3}}{2} = -\frac{1}{2} \pm \frac{\sqrt{3}}{2} i.$$

Note that all three roots can be clearly seen in Figure 1.1.13, which shows the result of Algorithm 1.1.9.

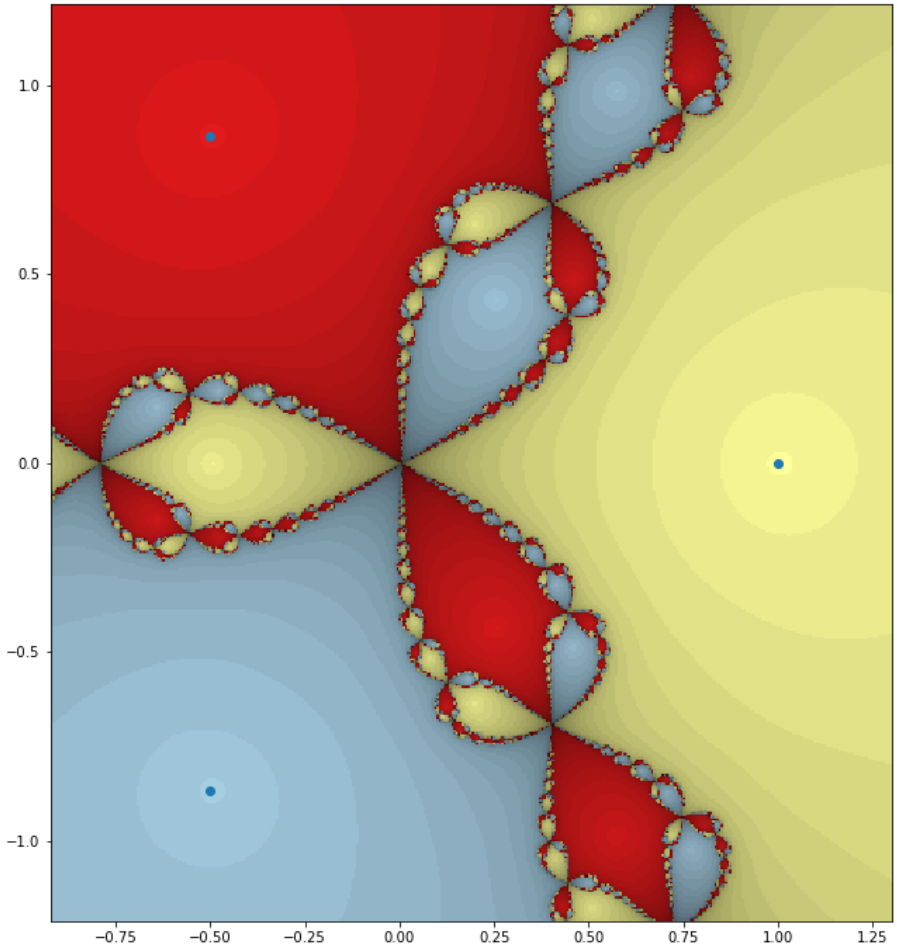

Figure 1.1.13 Attractive basins for $f(z) = z^2 + 3$

□

Figure 1.1.13 is just an incredible picture. We can clearly see the three roots of p and, indeed, if we start close enough to a root we do converge to that root. The boundary between the basins seems to be incredibly complicated, though.

As a final example, let's take a look at the complex cosine.

Example 1.1.14 Let $f(z) = \cos(z)$. Of course, we played with the real cosine back in Example 1.1.7. There, we saw that in any tiny interval containing zero, an initial seed might converge to any of infinitely many roots. In the complex

case, the basins of attraction are shown in Figure 1.1.15, and a zoom is shown in Figure 1.1.16.

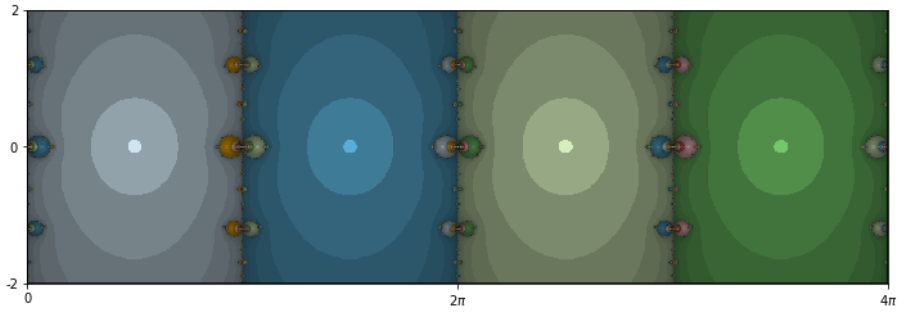

Figure 1.1.15 Attractive basins for the cosine

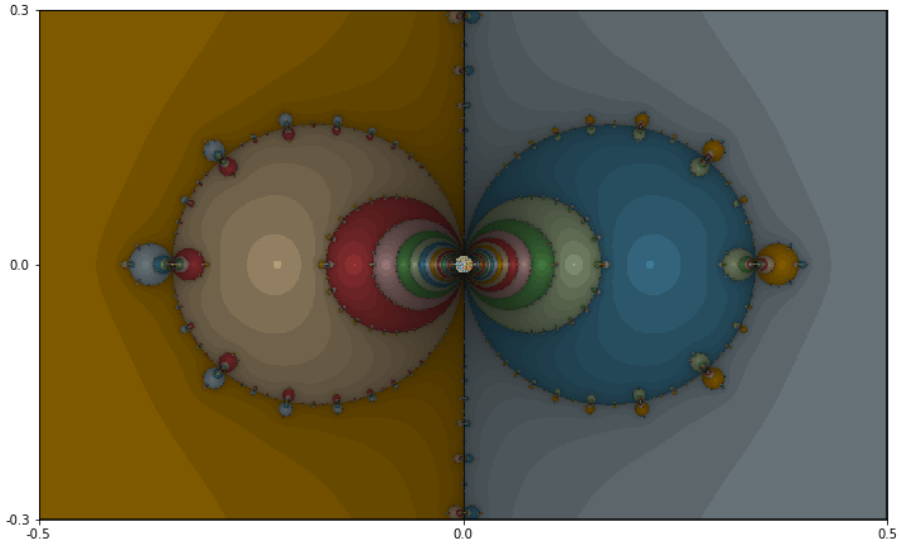

Figure 1.1.16 A zoom into the basins of the cosine

Two more incredible pictures! Note the periodicity in Figure 1.1.15. Each root seems to have its own column of attraction. The boundaries between the basins are again very complicated. In the zoom of Figure 1.1.16, t looks like there's an infinite sequence of circles collapsing on either side of the origin. This agrees with our observations in the real case of Example 1.1.7. □

Generating pictures for the Newton's method is great fun. There is an online tool that allows you to generate the basins of attraction for an (almost) arbitrary polynomial here:https://marksmath.org/visualization/complex_newton/.

1.2 The scope of chaos

The previous section introduced you a few fascinating ideas and images (functional iteration in real and complex variables and infinitely convoluted shapes that have become knows as fractal) - all in the context of Newton's method. The purpose of this short section is to indicate and illustrate a few other areas in which chaos and fractals occur.

1.2.1 The iteration of real functions

We were introduced to the idea of real iteration back in Subsection 1.1.2 but there's no reason the function we iterate needs to come from Newton's method. Figure 1.2.1 illustrates iteration using a geometric cool called a *cobweb plot*. In part (a) we see the iteration of $N(x) = x/2 + 1/x$, that arises in Newton's method. In part (b) we the iteration of $f(x) = 3.1x(1-x)$. This is an example of a *logistic* function and really has nothing directly to do with Newton's method.

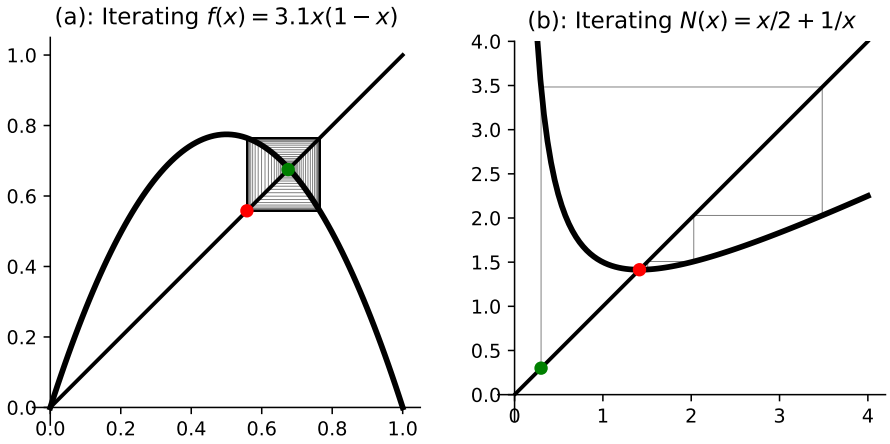

Figure 1.2.1 Two cobwebs that arises in real iteration

The logistic function above is a member of a *family* of functions - the logistic family. These functions have the form $f_r(x) = rx(1-x)$. When studying a family of functions like this we are often interested in the various types of behavior that might arise. We can explore these possibilites with a tool called a *bifurcation diagram*. The bifurcation diagram for the logistic family is shown in Figure 1.2.2.

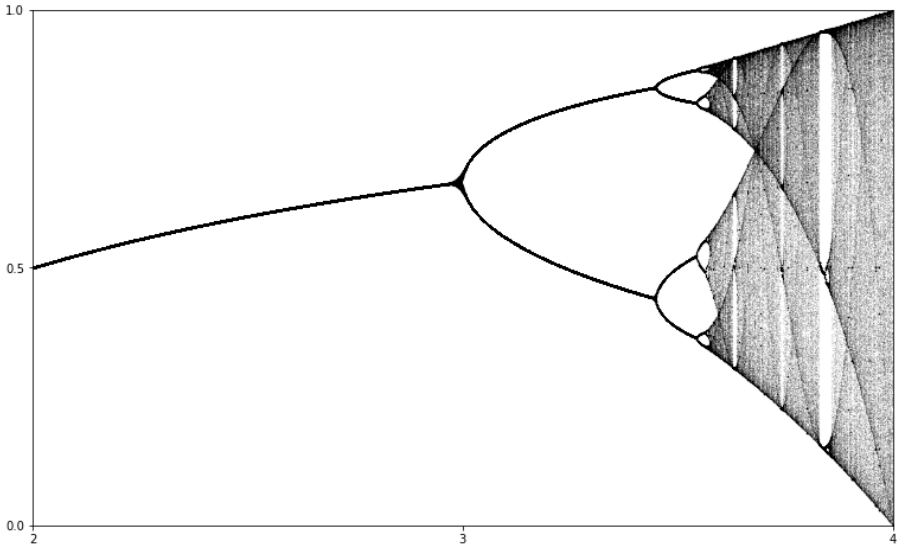

Figure 1.2.2 The bifurcation diagram for the logistic family

1.2.2 The iteration of complex functions

We were introduced to the idea of complex iteration back in Subsection 1.1.3 and, again, there's no reason the function we iterate needs to come from Newton's method. In the context of complex iteration we'll meet Julia sets, a few of which are shown in Figure 1.2.3

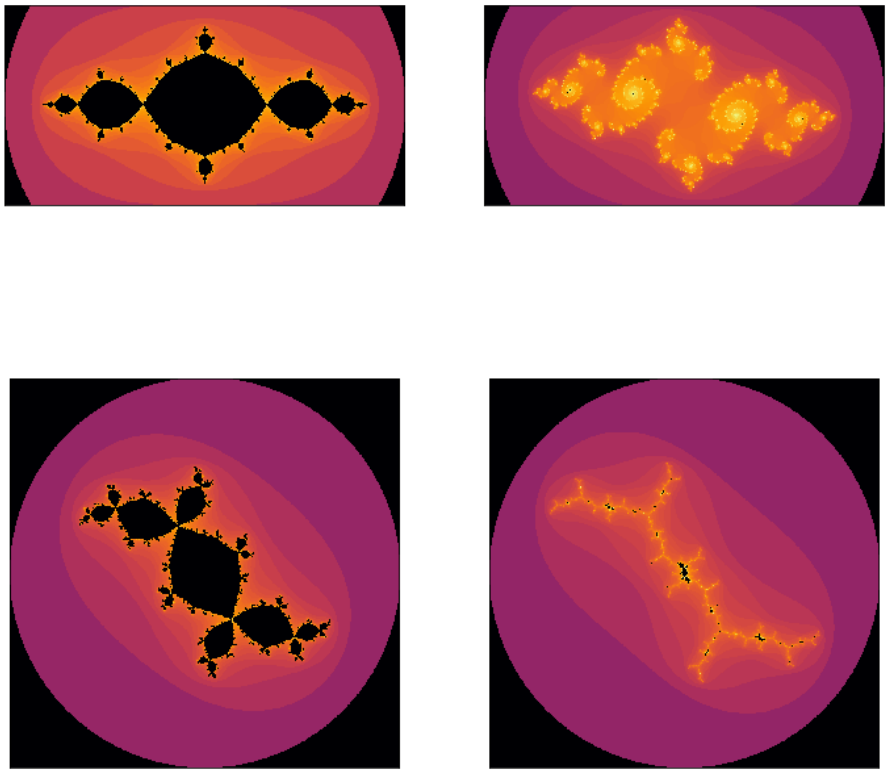

Figure 1.2.3 Some Julia sets

The Julia sets shown in Figure 1.2.3 all arise from the iteration of functions in the quadratic family - i.e. from the iteration of functions of the form $f(z) = z^2+c$. It is in this context that the famous Mandelbrot set shown in Figure 1.2.4 arises.

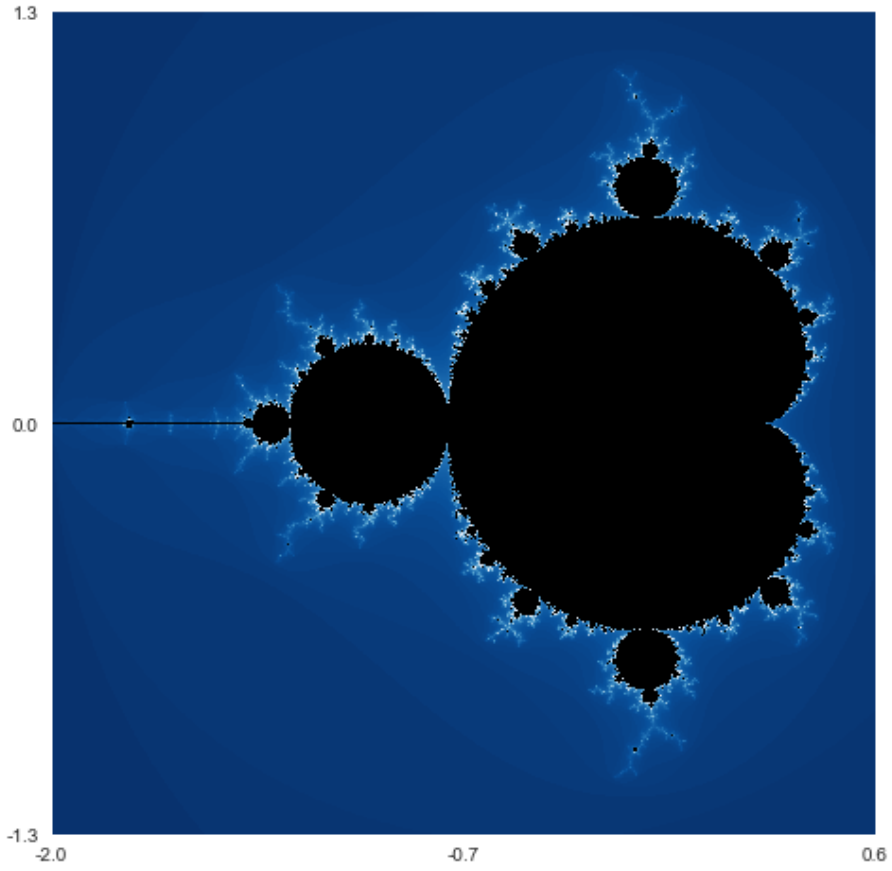

Figure 1.2.4 The Mandelbrot set

1.2.3 Geometric iteration and fractal geometry

When studying fractal geometry, we might iterate a function that maps sets to other sets. Such an example is shown in Figure 1.2.5. The limiting figure, called the Sierpinski triangle is shown in Figure 1.2.6.

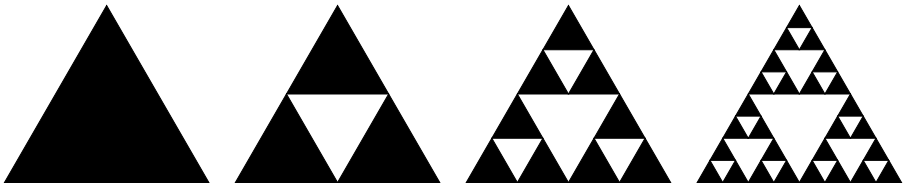

Figure 1.2.5 Geometric iteration

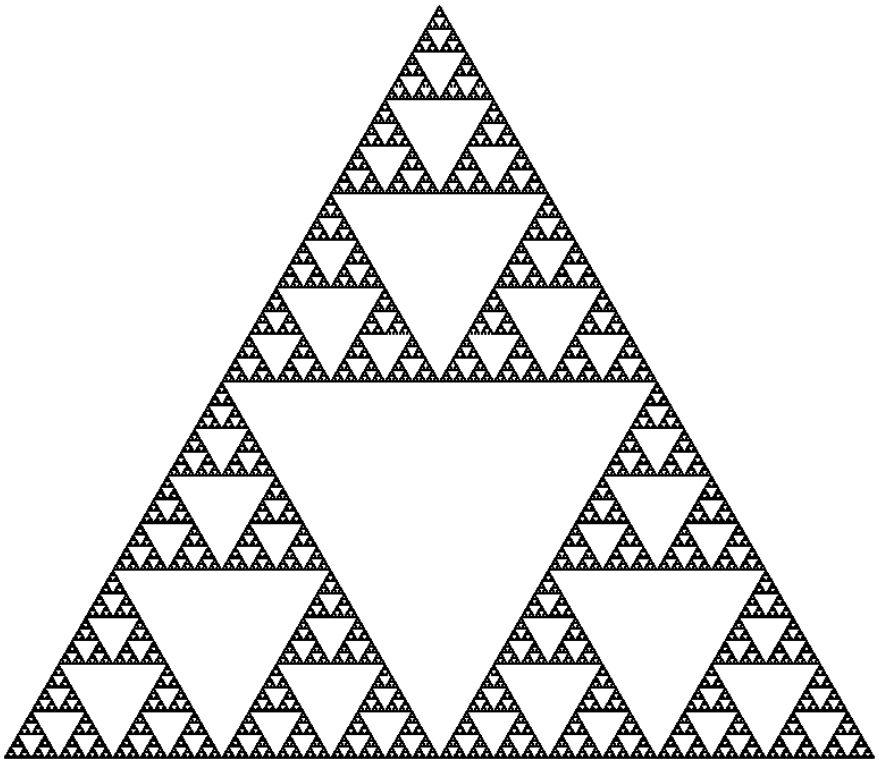

Figure 1.2.6 The Sierpinski triangle

The specific type of function that we iterate in this context is called an *iterated function system* Some amazingly intricate images can be constructed using an iterated function system. Figure 1.2.7 shows the Barnsely fern, which is described by a list of just four functions.

Figure 1.2.7 The Barnsley fern

1.3 Exercises

This set of exercises will be mostly experimental. So, fire up your favorite computational environment. There are lots of potential choices but this text will generally present examples using Sage.

1. Continuing with the example of $f(x) = x^2 - 2$ explored in Example 1.1.3, compute ten Newton iterations for several values of x_0. Be sure to choose both positive and negative values and values that are both large and small in magnitude.

2. In the previous exercise, what happens when $x_0 = 0$? Draw a graph to illustrate the situation.

3. Let f be a quadractic function that has two, distinct, real roots but that is otherwise arbitrary. Using a geometrical understanding of the real Newton's method, show why an initial seed x_0 always leads to a sequence that converges to the closer of the two roots of f.

4. Let's modify Newton's original example just a little bit to consider

$$f(x) = x^3 - 2x - 2.$$

(a) Compute the corresponding Newton's method iteration function, N.

(b) Iterate N from the initial point $x_0 = 0$. What behavior do you see?

(c) Iterate N from several initial points x_0 close to zero. Now, what behavior do you see?

5. Figure 1.3.1 shows the graph of the function

$$f(x) = \frac{1}{3}x(x+1)(x-3)(x^2-2).$$

The green dots represent points on the graph with x-coordinates that we might consider as initial seeds for Newton's method.

(a) Suppose we start at the green dot whose x coordinate is just slightly larger than 1. To which root do you think the process will converge?

(b) Suppose we start at the green dot whose x coordinate is between 2 and 3. To which root do you think the process will converge?

(c) Find a specific value of the initial seed x_0 between 2 and 3 with the property that the process converges to the smallest root of the function.

(d) Find a specific value of the initial seed x_0 between 2 and 3 with the property that the process converges to the value 1.

6. Launch the interactive tool for generating the basins of attraction of Newton's method for polynomials here: https://marksmath.org/visualization/complex_newton/.

Now, use the tool to generate images for the following polynomials and answer any additional questions that are asked.

(a) $f(z) = z^4 - 1$

i. What are the four roots of the function? Where do they fit into the picture?

ii. Click on the picture. How do you interpret the line that is drawn?

(b) $f(z) = (z^2 - 1)(z - 10)$

i. What initial step should you take to enter your input?

ii. What are the roots of the function? How could you account for this when generating the picture?

(c) $f(z) = z^3 - 2z - 2$

i. You should see some black regions. What's up with that?

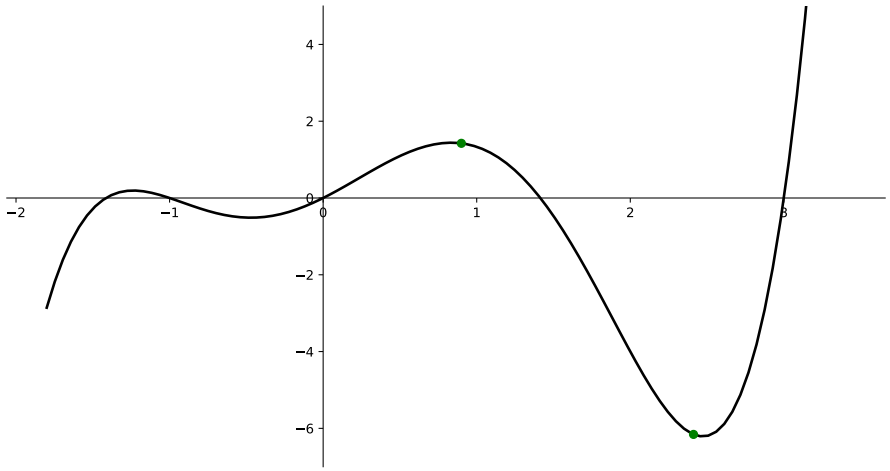

Figure 1.3.1 The graph of the function for problem 5

Chapter 2

The iteration of real functions

Introduction. In this chapter, we take a close look at discrete, real dynamics. That is, we study the iteration of functions $f : \mathbb{R} \to \mathbb{R}$. This concept was introduced intuitively in Subsection 1.1.2.

2.1 Basic notions

We begin with some of the most fundamental definitions and examples. While these definitions are stated for real functions, many of them extend quite easily to other contexts.

Definition 2.1.1 Let $x_0 \in \mathbb{R}$ be an initial point and define a sequence (x_n) recursively by $x_{n+1} = f(x_n)$. This sequence is called *the orbit* of x_0 under iteration of f. ◊

Some orbits don't move; they are fixed.

Definition 2.1.2 A point $x_0 \in \mathbb{R}$ is a *fixed point* of f if $f(x_0) = x_0$. ◊

Sometimes an orbit might return to the original starting point.

Definition 2.1.3 Suppose that the orbit (x_n) satisfies

$$x_0 \to x_1 \to x_2 \cdots \to x_{n-1} \to x_0$$

and $x_n = x_0$. Such an orbit is called a *periodic orbit* and the points themselves are called *periodic points*. If $x_k \neq x_0$ for $k = 1, 2, \ldots, n-1$, then n is called the *period* of the orbit. ◊

Note that a fixed point is a periodic point with period one.

Sometimes, the orbit of a non-periodic point might land on a periodic orbit.

Definition 2.1.4 If the zeroth term x_0 of an orbit (x_n) is not periodic but x_n is periodic for some n, then x_0 and its orbit are called *pre-periodic*. ◊

Example 2.1.5 Let $f(x) = 2x+1$. We could compute a few terms of the orbit of $x_0 = 1$ by direct comutation:

$$1 \to 3 \to 7 \to 14 \to 31 \to \cdots$$

It's not too hard to guess that a general formula for n^{th} term might be $x_n = 2^{n+1} - 1$. You should check this for the first few values of n.

It's easy to check that $x_0 = -1$ is a fixed point. It seems unlikely that there are other fixed points or periodic orbits of any period. □

Example 2.1.6 Let $f(x) = x^2 - 1$. Then zero is a periodic point and one is a pre-periodic point, as the reader may easily verify.

To find a fixed point, we can simply set $f(x) = x$ and solve the resulting equation. In this case, we get

$$x^2 - 1 = x \text{ or } x^2 - x - 1 = 0.$$

We can then apply the quadratic formula to find that

$$x = \frac{1 \pm \sqrt{5}}{2}$$

are both fixed. □

Often, it helps to express these ideas in terms of composition of functions. We denote the n fold composition of a function with itself by f^n. That is, $f^2 = f \circ f$ and $f^n = f \circ f^{n-1}$. (Be careful note to confuse this with raising a function to a power.) A more complete understanding of periodicity arises from the study of the functions f^n. For example, a point x_0 has period n iff $f^n(x_0) = x_0$ but $f^k(x_0) \neq x_0$ for $k = 1, 2, \ldots, n-1$.

Checkpoint 2.1.7 Let $f(x) = 2x^2 - 3x - 6$.

1. Find the first three terms of the orbit of $x_0 = 1$.
2. Find all fixed points of f.
3. Write down an equation that any point of period 2 should satisfy.

2.2 Computer experimentation

It doesn't take long to realize that a little computer power will help develop intution much better than, say, doing a lot of arithmetic by hand. We'll often use Sage for basic exploration of iterative dynamics. Here are a few examples.

2.2.1 Computing orbits

Suppose $f(x) = \frac{1}{5}x^2 - 2x + 6$. Here's how to compute the first five iterates of $x_0 = 1$ under f using Sage:

```
f(x) = (1/5)*x^2 - 2*x + 6
xi = 1
for i in range(5):
    xi = f(xi)
    print(xi)
```

```
21/5
141/125
312381/78125
36639701661/30517578125
1810059539710884397142 1/4656612873077392578125
```

Note that the computation is exact. That is, we get rational numbers like 21/5, rather than decimal approximations like 4.2. Often we would prefer decimal approximations. Let's set xi = 1.0 in the code block above and compute 100 iterates.

```
f(x) = (1/5)*x^2 - 2*x + 6
xi = 1.0
for i in range(100):
    xi = f(xi)
    print(xi)
```

```
4.20000000000000
1.12800000000000
3.99847680000000
1.20060974402765
3.88707326343553
...
3.61804229612085
1.38196141906219
3.61804063463090
1.38196233750627
3.61803930544461
1.38196307225920
```

Sure looks like we've found a pattern! Perhaps, it's an orbit of period 2?

2.2.2 Finding periodic orbits

Continuing with the example of the last sub-section, suppose we'd like to find all orbits of period 2 for $f(x) = \frac{1}{5}x^2 - 2x + 6$. I guess we should let $F(x) = f^2(x)$ and then find all fixed points of F or, equivalently, find all roots $F(x) - x$. We can automate this procedure like so:

```
f(x) = (1/5)*x^2 - 2*x + 6
F(x) = f(f(x))
(F(x)-x).roots(ring=RR)
```

```
[(1.38196601125011, 1),
 (2.37652461702020, 1),
 (3.61803398874990, 1),
 (12.6234753829798, 1)]
```

Each root is returned as a (value,multiplicity) pair. Note that the 1.38 and 3.618 agree with our prior computation; those form the orbit of period 2. The other points are fixed points. Of course, those should be the points of intersection with the line $y = x$. We can illustrate that with Sage too:

```
f(x) = (1/5)*x^2 - 2*x + 6
plot([x,f(x)], [x,-4,14]) +
    point([(2.37,2.37),(12.62,12.62)])
```

Sometimes we might be interested in a somewhat higher interval, in which case it might make sense to use the computer to do the iteration for us. For example, here's how to find the fourth iterate of $f(x) = x^2 - 2$:

```
f(x) = x^2 - 2
F(x) = x
for i in range(4):
    F(x) = F(f(x))
F(x)
```

```
(((x^2 - 2)^2 - 2)^2 - 2)^2 - 2
```

With that in hand, we can find the orbits of period 4.

```
f(x) = x^2 - 2
F(x) = x
for i in range(4):
    F(x) = F(f(x))
(F(x)-x).roots(ring=RR)
```

```
[(-1.96594619936780, 1),
 (-1.95629520146761, 1),
 (-1.70043427145923, 1),
 (-1.61803398874989, 1),
 (-1.20526927275851, 1),
 (-1.00000000000000, 1),
 (-0.547325980144166, 1),
 (-0.209056926535307, 1),
 (0.184536718926604, 1),
 (0.618033988749895, 1),
 (0.891476711553077, 1),
 (1.33826121271772, 1),
 (1.47801783444132, 1),
 (1.82709091528520, 1),
 (1.86494445880871, 1),
 (2.00000000000000, 1)]
```

Checkpoint 2.2.1 Refering to the last example?

- What is the orbit of the first point `-1.96594619936780` in the list?
- What do you make of the last point `2.0000000000` in the list?

Checkpoint 2.2.2 Find an orbit of period three. Given the function $f(x) = 7x^2 + 14x + \frac{23}{4}$,

1. Iterate f from the starting point $x_0 = 1.0$ twenty times. What does the long term behavior appear to be?
2. Iterate f from the starting point $x_0 = -1.0$ two hundred times. What does the long term behavior appear to be?
3. Write down an equation that an orbit of period 3 should satisfy and solve that equation using Sage.

2.3 Graphical analysis

There is an efficient geometric tool to visualize functional iteration. The basic idea is simple: Suppose we graph the function f together with the line $y = x$. If those two graphs intersect; that point of intersection is a fixed point. Now, suppose we're on the line at the point (x_i, x_i). If we move vertically to the graph of the function, we preserve the x coordinate but change the y coordinate to $f(x_i)$. Thus, we arrive at the point $(x_i, f(x_i)) = (x_i, x_{i+1})$. If we then move horizontally back to the line $y = x$ we now preserve the y coordinate but change the x coordinate so that the x and y coordinates are the same. Thus, we arrive at the point (x_{i+1}, x_{i+1}).

In summary: The process of moving vertically from a point on the line $y = x$ to the graph of f and back to the line horizontally is a geometric representation of one application of the function f. This step is illustrated in Figure 2.3.1(a). Repeated application of this process represents repeated application of f, i.e.

iteration. This is illustrated in Figure 2.3.1(b). Note that the orbit appears to be attractive.

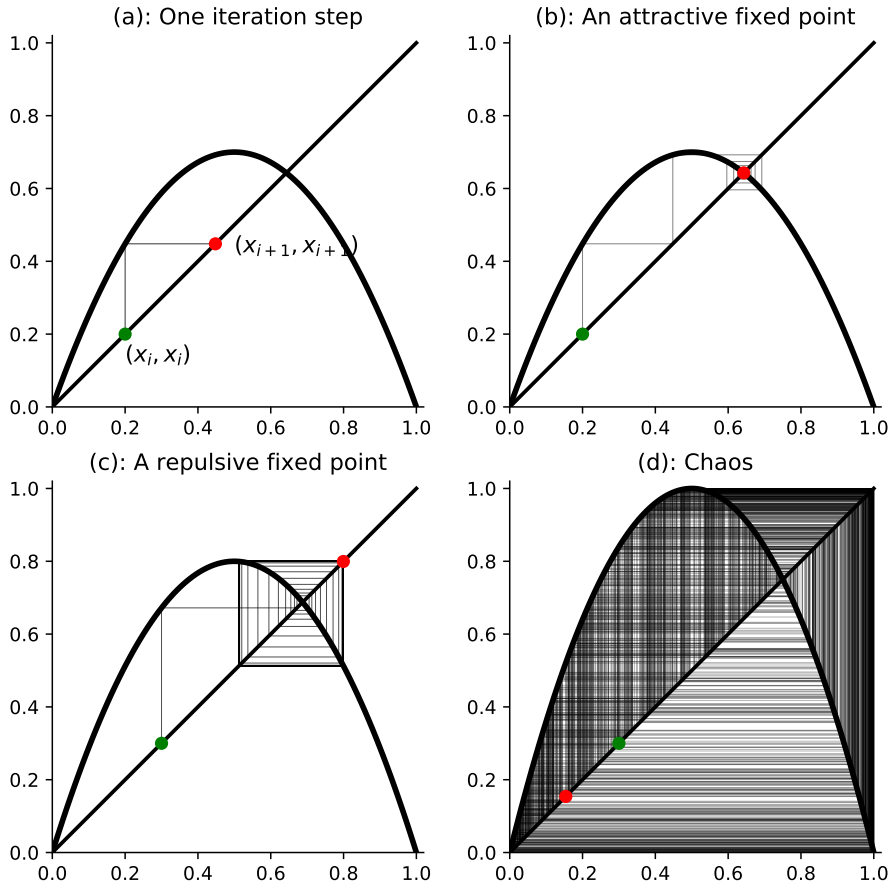

Figure 2.3.1 Some cobweb plots

It turns out that the process is quite sensitive to the slope of the function at the point of intersection. A slightly steeper function is shown in Figure 2.3.1(c); we notice that the fixed point now appears to be repelling. Finally, figure Figure 2.3.1(d) illustrates the fact that all hell can break loose.

Playing with cobweb plots is a great way to get a feel for the possibilites that arise in discrete dynamical systems. Here's an interactive tool that we'll use for this purpose:https://marksmath.org/visualization/cobwebs/

Checkpoint 2.3.2 Plug the function $f(x) = \frac{1}{5}x^2 - 2x + 6$ that we played with in Section 2.2 into the cobweb tool. Can you see why we couldn't help but find the periodic behavior?

2.4 Classification of fixed points

The cobweb plots in the previous section illustrate that the slope of the function at the point where it crosses the fixed point might play a role in the behavior of the iterates near that fixed point. We explore that further here. First, we explore the simplest situation - functions with constant slope.

Example 2.4.1 Linear iteration. Suppose that f is a linear function: $f(x) = ax$. It's easy to see that the origin $x = 0$ is a fixed point of f. Show that

any non-zero initial point x_0 moves away from the origin under the iteration of f whenever $|a| > 1$ but moves towards the origin under iteration of f if $|a| < 1$.

Solution. This is easy, once we recognize that there is a closed form for the n^{th} iterate of f, namely $f^n(x) = a^n x$. Note that cobweb plots for these functions are shown in Figure 2.4.2.

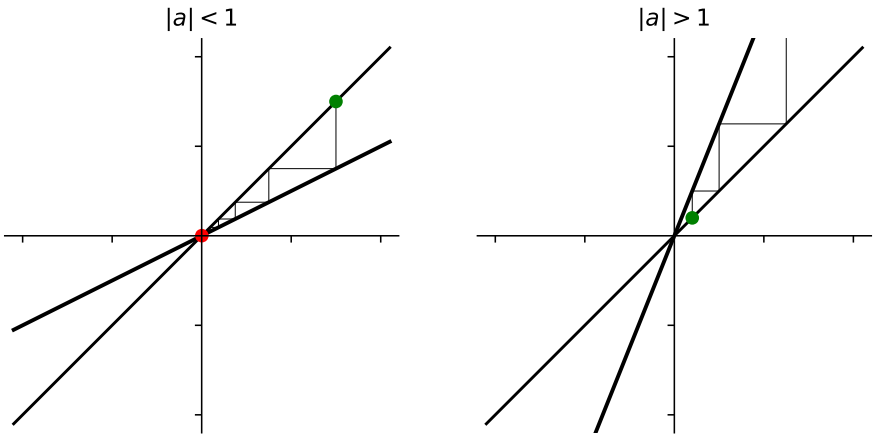

Figure 2.4.2 Some linear cobweb plots

□

Checkpoint 2.4.3 Affine iteration. Suppose, that f is an *affine function*, which just means that it has the form $f(x) = ax + b$, where $a \neq 0$. Suppose also that $x_0 \in \mathbb{R}$ and let's consider the iterates $x_{n+1} = f(x_n)$

1. Show that f has a unique fixed point iff $a \neq 1$. What if $a = 1$?
2. Suppose that $|a| < 1$. Show that the sequence of iterates converges to the fixed point of f.
3. Suppose that $|a| > 1$. Show that the sequence of iterates diverges.
4. What happens if $a = -1$?

Example Example 2.4.1 and exercise Checkpoint 2.4.3 together classify the dynamical behavior of first order polynomials completely and show that their behavior is fairly simple. For that reason, we focus on polynomials of degree two and higher. Already in the quadratic case, we can find much more complicated and interesting behavior. Motivated by the behavior we see in linear and affine functions, we make the following defintion.

Definition 2.4.4 Let $f : \mathbb{R} \to \mathbb{R}$ be continuously differentiable and suppose that $x_0 \in \mathbb{R}$ is a fixed point of f. Then we classify x_0 as

1. *attractive*, if $0 < |f'(x_0)| < 1$,
2. *super-attractive*, if $f'(x_0) = 0$,
3. *repulsive* or *repelling*, if $|f'(x_0)| > 1$, or
4. *neutral*, if $|f'(x_0)| = 1$,

The number, $f'(x_0)$ is called the *multiplier* for the fixed point. If, in the attractive case, the multiplier is zero, we say that x_0 is *super-attractive*. ♢

The following theorem justifies this notation.

Theorem 2.4.5 *Let $f : \mathbb{R} \to \mathbb{R}$ be continuously differentiable and suppose that $x_0 \in \mathbb{R}$ is a fixed point of f.*

1. *If x_0 is an attractive or super-attractive fixed point for f, then there is an $\varepsilon > 0$ such that the orbit of x under iteration of f tends to x_0 for every x such that $|x - x_0| < \varepsilon$.*

2. *If x_0 is a repelling fixed point for f, then there is an $\varepsilon > 0$ such that the orbit of x under iteration of f tends (initially) away from x_0 for every x such that $|x - x_0| < \varepsilon$.*

Proof. We prove part one; the second part is similar. Since $|f'(x_0)| < 1$ and f' is continuous, we may choose an $\varepsilon > 0$ and a postive number $r < 1$ such that $|f'(x)| < r$ for all x such that $|x - x_0| < \varepsilon$. Then, given x such that $|x - x_0| < \varepsilon$, we can apply the Mean Value Theorem to obtain a number c such that

$$|f(x) - x_0| = |f(x) - f(x_0)| = |f'(c)||x - x_0| \leq r\varepsilon.$$

By induction, we can show that

$$|f^n(x) - x_0| \leq r^n\varepsilon.$$

The result follows, since $r^n\varepsilon \to 0$ as $n \to \infty$. ■

From the proof, we see that $x_n \to x_0$ exponentially and that the magnitude of $|f'(x_0)|$ dictates the base of that exponential. When $f'(x_0) = 0$, the rate is faster than exponential.

Example 2.4.6 The function $f(x) = 4.8\,x^2(1 - x)$ is graphed in Figure 2.4.7, along with the line $y = x$. The points of intersection are fixed points and, from left to right, they are super-attractive, repulsive, and attractive. The reader should consider the appearance of a cobweb plot for initial values starting near each of those fixed points.

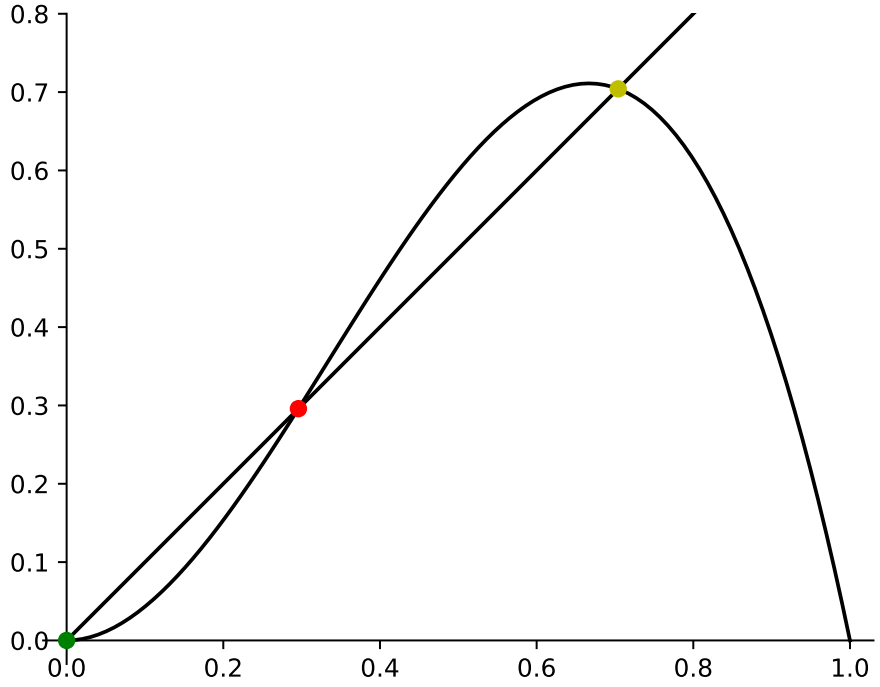

Figure 2.4.7 Three types of fixed points

□

The behavior of iterates near a neutral fixed point can be more varied.

Checkpoint 2.4.8 Dynamical behavior near neutral fixed points. For each of the following scenarios, find an example of a function $f : \mathbb{R} \to \mathbb{R}$ and a fixed point x_0 of f satisfying that scenario.

1. There is an $\varepsilon > 0$ such that the orbit of x tends to x_0 for all x such that $|x - x_0| < \varepsilon$.
2. There is an $\varepsilon > 0$ such that the orbit of x tends initially away from x_0 for all x such that $|x - x_0| < \varepsilon$.
3. There is an $\varepsilon > 0$ such that the orbit of x tends to x_0 for all x such that $0 < x - x_0 < \varepsilon$ but the orbit of x tends initially away from x_0 for all x such that $0 < x_0 - x < \varepsilon$.
4. There is an $\varepsilon > 0$ such that the orbit of x tends to x_0 for all x such that $0 < x - x_0 < \varepsilon$ but the orbit of x tends initially away from x_0 for all x such that $0 < x_0 - x < \varepsilon$.

2.5 Classification of periodic orbits

As mentioned right after the example on periodicty 2.1.6, a periodic point for f of period n is a fixed point of f^n. Treating the points of a periodic orbit this way allows us to extend the classification as fixed points to periodic orbits.

Definition 2.5.1 Let $f : \mathbb{R} \to \mathbb{R}$ be continuously differentiable and suppose that $x_0 \in \mathbb{R}$ is a periodic point of f with period n. Let $F = f^n$. We classify x_0 and its orbit as

1. *attractive*, if $|F'(x_0)| < 1$,
2. *super-attractive*, if $F'(x_0) = 0$,
3. *repulsive* or *repelling*, if $|F'(x_0)| > 1$, or
4. *neutral*, if $|F'(x_0)| = 1$,

The number $F'(x_0)$ is called the *multiplier* of the orbit. If, in the attractive case, the multiplier is zero, we say that the orbit is *super-attractive*. ◊

There is a nice characterization of the multiplier of an orbit that allows us to compute it without explicitly computing a formula for f^n.

Lemma 2.5.2 *Suppose that*

$$x_0 \to x_1 \to x_2 \to \cdots \to x_{n-1} \to x_0$$

is an orbit of period n for $f : \mathbb{R} \to \mathbb{R}$. Then the multiplier of the orbit is

$$f'(x_0)f'(x_1)\cdots f'(x_{n-1}).$$

Proof. First note that for an $n = 2$, we can apply the chain rule to obtain

$$\frac{d}{dx}f^2(x) = \frac{d}{dx}f(f(x)) = f'(f(x))f'(x).$$

Thus, if $x_0 \to x_1 \to x_0$ is an orbit of period two and we evaluate that equation at x_0, we obtain

$$\frac{d}{dx}f^2(x)\bigg|_{x=x_0} = f'(x_1)f'(x_0).$$

The result for orbits longer than two can be proven by induction, since

$$\frac{d}{dx}f^n(x) = \frac{d}{dx}f(f^{n-1}(x)) = f'(f^{n-1}(x))\frac{d}{dx}f^{n-1}(x).$$

■

Note that the only way the product in Lemma 2.5.2 is zero, is if one of the terms is zero. This yields the following corollary.

Corollary 2.5.3 *A periodic orbit is super-attracting if and only if it contains a critical point.*

Example 2.5.4 Let $f(x) = x^2 - 1$. Note that $f(0) = -1$ and $f(-1) = 0$ so that $0 \to 1 \to 0$ forms an orbit of period 2. To see if this orbit is attractive, we examine

$$F(x) = f(f(x)) = (x^2 - 1)^2 - 1 = x^4 - x^2.$$

Note that $F'(0) = 0$ and $F'(-1) = 0$; thus, the orbit is super-attractive.

The plots of f and f^2, together with $y = x$, are shown in Figure 2.5.5. Note that f has two fixed points shown in red. They can found by solving the equation $x^2 - 1 = x$ and they are both repulsive under iteration of f. The two super-attractive orbits of f^2 are shown in green.

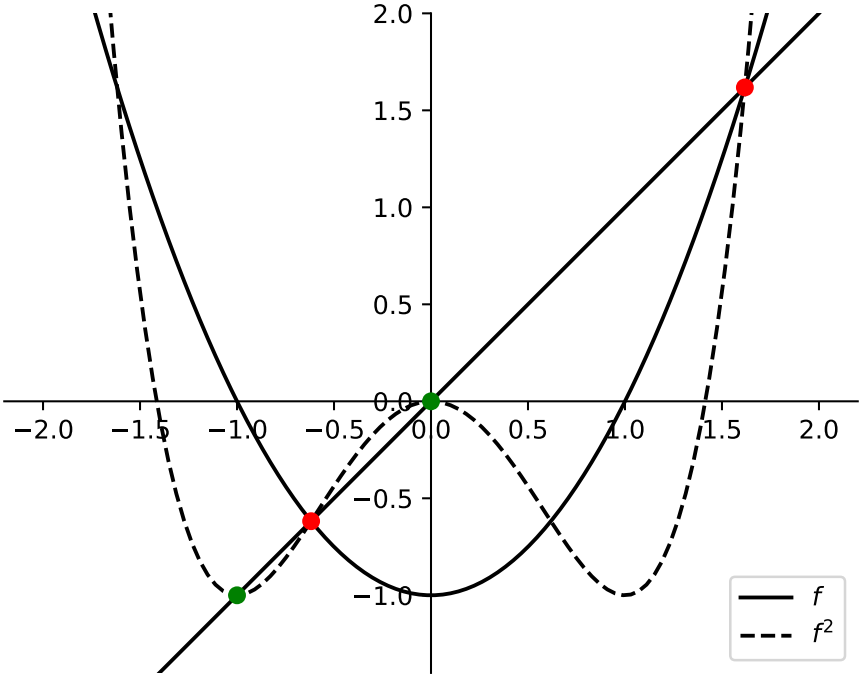

Figure 2.5.5 An attractive orbit of period two

□

2.6 Critical orbits

A *critical point* of f is simply a point where the derivative of f is zero and the orbit of a critical point is called a *critical orbit*. As it turns out, critical orbits have an outsized influence on the global behavior of the dynamics of f - a fact due largely to the following theorem:

Theorem 2.6.1 *If $f : \mathbb{C} \to \mathbb{C}$ has an attractive or super-attractive orbit, then that orbit must attract at least one critical point.*

Note that this is really a theorem of complex dynamics. There is an analogous statement for real dynamics but its statement is much more complicated and takes us a bit farther astray than we want or need. This is a great example of complex analysis being, in some ways, more elegant than real analysis.

using Theorem 2.6.1 makes it easy to find attractive orbits.

Example 2.6.2 Let $f_c(x) = x^2 - 1.625$. Find *all* attractive or super-attractive orbits of f.

Since $f'(x) = 0$, f has exactly one critical point, namely $x = 0$. Thus we can find any attractive behavior by iterating from zero.

```
xi = 0
for i in range(1000):
    xi = xi^2 - 1.625
    if i>=994:
        print(xi)
```

```
-0.00576937193306759
-1.62496671434750
1.01551682273730
-0.593725582737533
-1.27248993240298
-0.00576937193306759
```

Sure looks like we've found an orbit of period 5. Furthermore, this is necessarily the *only* periodic oribit. □

It's worth pointing out that it is essential that we consider all complex critical points. It's easy to see, for example, that zero is an attractive fixed point for $f(x) = \frac{1}{2}x + x^3$. The derivative of f is $f'(x) = \frac{1}{2} + 3x^2$ which has no real roots. There are two *complex* roots of f', though, that are attracted to the origin.

Nontheless, Theorem 2.6.1 is a powerful tool for understanding real iteration when the critical points are real and we'll use this to powerful effect in the next section.

2.7 Parameterized families of functions

Rather than explore the behavior of a single function at a time, we can introduce a parameter and explore the range of behavior that arises in a whole family of functions.

2.7.1 Examples of parameterized families

Two important examples of parameterized families of functions are

1. *The quadratic family*: $f_c(x) = x^2 + c$
2. *The logistic family*: $f_\lambda(x) = \lambda x(1 - x)$

The cobweb plots shown back in Figure 2.3.1 are all chosen from the logistic family with $\lambda = 2.8$, $\lambda = 3.2$, and $\lambda = 4$. Even in those three pictures with graphs that look so very similar, we see three different types of behavior: an attractive fixed point, an attractive orbit of period two, and chaos (which can be given a very technical meaning).

Figure 2.7.1 shows some cobweb plots for the quadratic family of functions. Note that the behavior we see is very similar to the behavior we see for the logistic family - a fact that will become more understandable once we study conjugacy in section Section 2.8

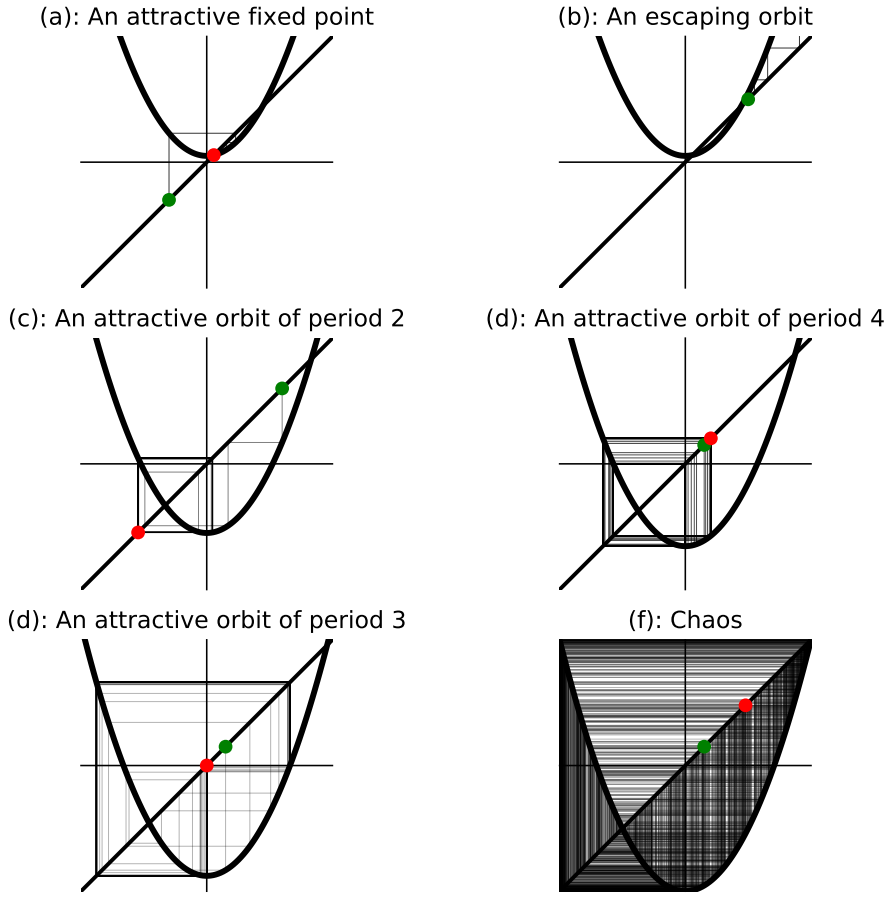

Figure 2.7.1 Some cobweb plots for the quadratic family

2.7.2 The bifurcation diagram

A fabulous illustration of the types of behavior that can arise in a family of functions indexed by a single real parameter and each with a single critical point can be generated as follows: For each value of the parameter, compute a large number points of the orbit of the critical point (maybe 1000 iterates). Since we're interested in long term behavior, rather than any transient behavior, discard the first few iterates (maybe 100). Then, plot the remaining points in a vertical column at the horizontal position indicated by the parameter.

If we do this systematically for the quadratic family, plotting the columns to generate the bifurcation diagram, we get Figure 2.7.2

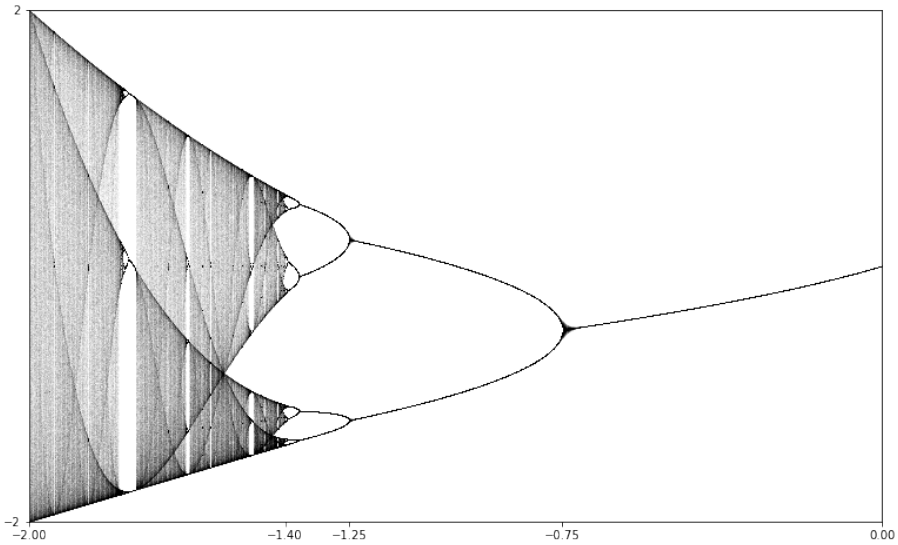

Figure 2.7.2 The bifurcation diagram for the quadratic family

We can interpret this diagram as follows:

- For $-0.75 < c < 0$, there is an attractive fixed point.
- For $-1.25 < c < -0.75$, there is an attractive orbit of period 2.
- As c passes from just above -0.75 to just below -0.75, the dynamics of f_c undergo a *bifurcation*.
- For c just a little less than -1.25, there is an attractive orbit of period four. This orbit bifurcates soon into an attractive orbit of period 8. It appears that this behavior continues as c decreases.
- For c somewhere around $c \approx -1.4$, the period doubling appears to stop and we get more complicated behavior.

Generally, a bifurcation occurs at a parameter value $c = c_0$ if the global dynamical behavior of the function f_c undergoes some qualitative change as c passes through c_0. There are number of different types of bifurcations that can occur, depending on the nature of the qualitative behavior under consideration. The bifurcations that are evident in Figure 2.7.2 in the range $-1.4 < c < 0$ are called *period doubling bifurcations*.

2.7.3 The period doubling cascade

Let's work towards a deeper, theoretical understanding of the period doubling that we see in the bifurcation diagram of Figure 2.7.2. Again, we are dealing with the family of functions $f_c(x) = x^2 + c$. For c just a bit larger than -0.75 it appears that we have an attractive fixed point while, for c just a bit smaller than -0.75, it appears that we have an attracting orbit of period two. Why, exactly does this happen?

First, let's explore the fixed points of f_c; we can find them by solving $f_c(x) = x$:

$$x^2 + c = x \iff x^2 - x + c = 0.$$

Applying the quadratic formula, we find

$$x = \frac{1 \pm \sqrt{1 - 4c}}{2}.$$

For $c < 1/4$, we have two real fixed points but a glance at the graphs from Figure 2.7.1 shows that it's the smaller of these two fixed points we're interested in. Of course, $f'(x) = 2x$, so the value of the derivative at the smaller fixed point is $1 - \sqrt{1-4c}$. Plugging $c = -3/4$ into this formula, we find that this is -1. For c slightly larger than $-3/4$, this is bigger than -1 and for c slightly smaller than $-3/4$, this is smaller than -1. This explains why we have an attractive fixed point for c slightly larger than $-3/4$ that is no longer attractive once c passes below $-3/4$.

Now, we ask - why does the attractive orbit of period two appear as the attractive fixed point disappears? To see this, we consider the function

$$F_c(x) = f_c \circ f_c(x) = (x^2 + c)^2 + c = x^4 + 2cx^2 + (c^2 + c).$$

We are interested in the fixed points, thus we must solve

$$x^4 + 2cx^2 + (c^2 + c) = x \text{ or } x^4 + 2cx^2 - x + (c^2 + c) = 0. \tag{2.7.1}$$

Here is an observation that helps us factor this polynomial: Any point that is fixed by f_c must also be fixed by F_c. Thus, we expect $x^2 + c - x$ to be a factor of the polynomial in (2.7.1). Using this, we find that

$$x^4 + 2cx^2 - x + (c^2 + c) = (x^2 - x + c)(x^2 + x + c + 1).$$

We can then apply the quadratic formula to get the two new fixed points of F_c, namely

$$x = \frac{-1 \pm \sqrt{1 - 4(c+1)}}{2} = \frac{-1 \pm \sqrt{-(3+4c)}}{2}.$$

These two points form an orbit of period two for f_c. Since $f_c'(x) = 2x$ we can multiply those points by two and multiply the results to get the multiplier for the orbit. The result is:

$$(-1 + \sqrt{-(3+4c)})(-1 - \sqrt{-(3+4c)}) = 4 + 4c.$$

When $c = -3/4$, the multiplier is 1. For c a little less than $-3/4$, the multiplier is a little less than one. Hence the orbit has become attractive.

A nice way to visualize this is to plot f_c^2 together with f_c and $y = x$ on the same set of axes for a few different choices of c. This is shown in Figure 2.7.3 where we can see exactly how The fixed point went from attractive to repulsive while an attractive orbit of period two showed up as c passed below -0.75.

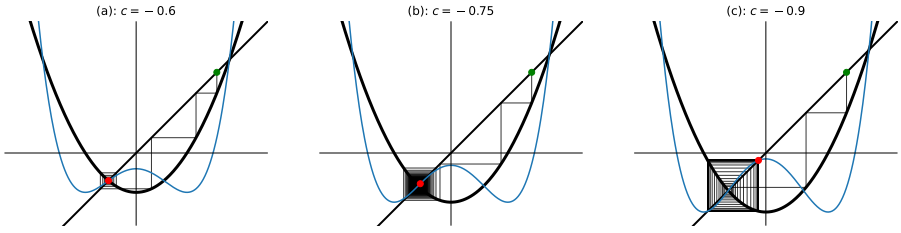

Figure 2.7.3 Bifurcation

Note that our cobweb tool allows you to plot iterates of f on the plot to assist in this type of analysis.

Checkpoint 2.7.4 Consider the logistic family $f_\lambda(x) = \lambda x(1 - x)$.

1. Using our cobweb tool

2.7.4 The critical curves

A close look at Figure 2.7.2 seems to indicate the presence of curves lending structure to the diagram. These are the so-called *critical curves*.

Definition 2.7.5 Let f_α be a parameterized family of functions, each with a single critical point c_α. The n^{th} critical curve is defined by $f_\alpha^n(c_\alpha)$, which is a polynomial in α ◊

For the quadratic family $f_c(x) = x^2 + c$, the n^{th} critical curve is the polynomial $f_c^n(0)$, which is a polynomial in c. The first few critical polynomials are

$$\begin{aligned} f_c^0(0) &= 0 \\ f_c^1(0) &= c \\ f_c^2(0) &= c^2 + c \\ f_c^3(0) &= (c^2 + c)^2 + c \\ f_c^4(0) &= ((c^2 + c)^2 + c)^2 + c \end{aligned}$$

If we plot a few of these on top of the bifurcation diagram, we get Figure 2.7.6.

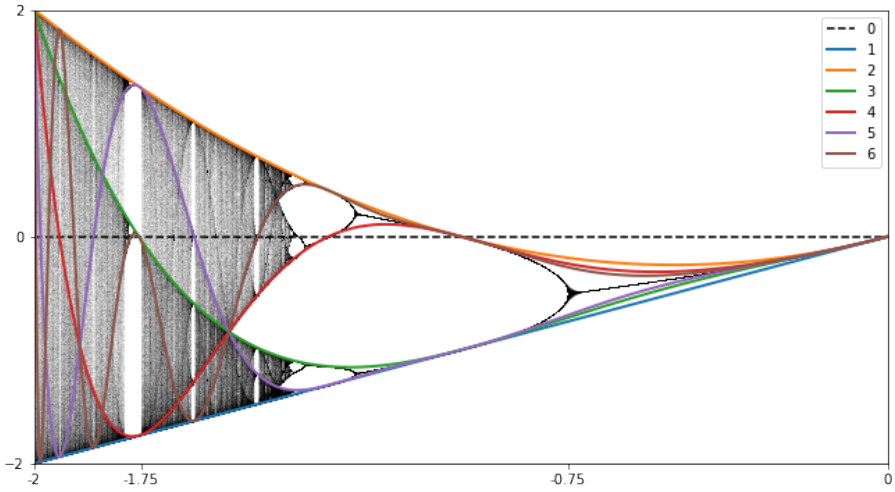

Figure 2.7.6

Figure 2.7.7 Gives an indication of how the critical curves form. The dashed parallel lines on either side of the horizontal c-axis define a region in the plane of the bifurcation diagram. The image of those lines under application of f_c^n are shown for $n = 1, 2, 3$ in the figure, together with the corresponding critical curve. As we see, the region bound between the original horizontal lines is mapped by f_c^n to a much smaller region on one side of the n^{th} critical curve. Thus, there is a higher density of points on one side of the each critical curve.

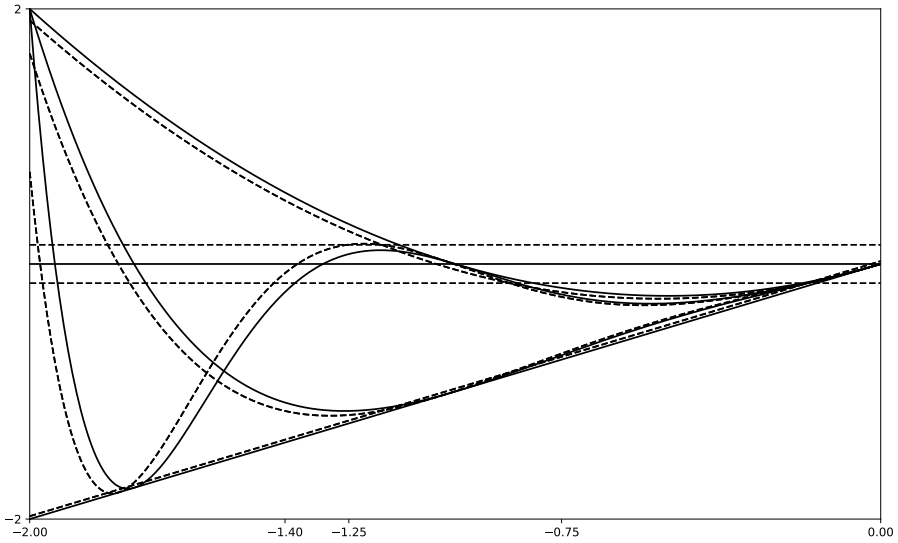

Figure 2.7.7

2.7.5 Periodic windows

Amongst the chaos apparent within the bifurcation diagram, there appear to be a number *periodic windows* - a range of parameter values where an attractive periodic orbit dominates the behavior. The largest one appears in Figure 2.7.6 just to the left of $c = -1.75$ where the third (green) critical curve crosses the horizontal c-axis. This is the so-called *period three window*.

Note that the critical curves seem to cross the horizontal c-axis at exactly the locations of the periodic windows. Corollary 2.5.3 explains why. Note that $f_c^n(0)$ crosses the horizontal axis exactly when $f_c^n(0) = 0$. Thus, 0 is periodic for f_c with a period that divides n.

When $n = 3$, for example, this leads to the equation

$$(c^2 + c)^2 + c = 0.$$

We can find all real roots of this polynomial using the following Sage code:

```
f(c,x) = x^2 + cx
F(c,x) = x
for i in range(3):
    F(c,x) = F(c,f(c,x))
F(c,0).roots(ring=RR)
```

```
[(-1.75487766624669, 1), (0.000000000000000, 1)]
```

The value of about -1.75 agrees with our observations. The value of zero occurs because zero is a fixed point of f_0 and, therefore, of f_0^3 as well.

An alternative way to shed light on the sudden occurrence of the period three window is to use cobwebplots as shown in Figure 2.7.8. In that figure on the left, we see the cobweb plot for $c = -1.6$ together with the 3^{rd} iterate $f_{-1.6}^3$ of f. The key observation is that the graph of $f_{-1.6}^3$ completely misses the line $y = x$. Thus, there is no orbit of period three for $c = -1.6$. By the time $c = -1.76$ on the right figure, there is such an orbit.

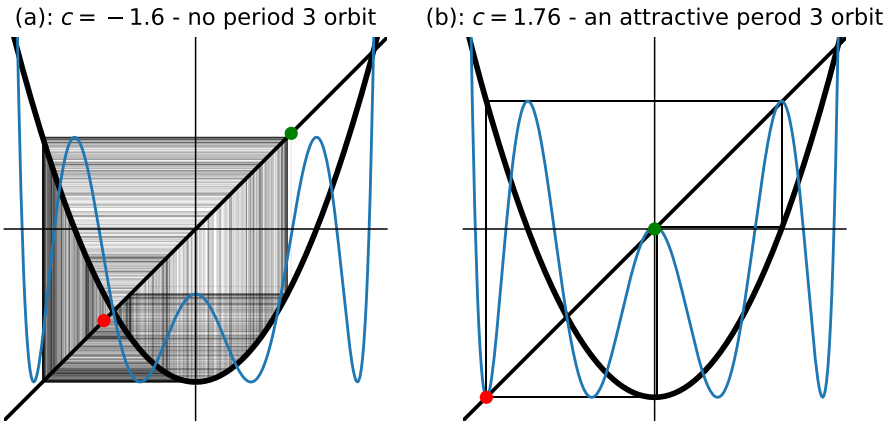

Figure 2.7.8 The occurence of the period 3 window

Note that our interactive cobweb tool can be used to explore this further: https://marksmath.org/visualization/cobwebs/

2.8 Conjugacy

Figure 2.3.1 and Figure 2.7.1 show that the iterative behavior of the logistic family and the quadratic family are very similar. In a sense, they are identical. We make that notion precise in this section.

Definition 2.8.1 Conjugacy. Let S and T be sets and suppose that $f : S \to S$ and $g : T \to T$. We say that f is *semi-conjugate* to g if there is a surjective function $\varphi : T \to S$ such that

$$f \circ \varphi = \varphi \circ g.$$

The function φ is called a *semi-conjugacy*. In the case that φ is bijective, then we say that φ is a conjugacy and that f and g are conjugate. ◊

A geo-symbolic way to remember the semi-conjugation formula is in the form of a commutative diagram:

$$\begin{array}{ccc} T & \xrightarrow{g} & T \\ \downarrow{\scriptstyle \varphi} & & \downarrow{\scriptstyle \varphi} \\ S & \xrightarrow{f} & S \end{array}$$

The formula states if you follow the arrows from the upper left to the lower right in either direction, you get the same result.

An immediate consequence of the definition of conjugacy is

$$f^2 \circ \varphi = f \circ f \circ \varphi = f \circ \varphi \circ g = \varphi \circ g \circ g = \varphi \circ g^2$$

and, by induction

$$f^n \circ \varphi = \varphi \circ g^n.$$

As a result, if (t_i) is an orbit of g, then $(\varphi(t_i))$ is an orbit of f.

Generally, the nicer φ is, the closer the relationship between the dynamics of f and the dynamics of g. If φ is bijective, then the relationship is quite close. If φ is continuous with continuous inverse, then topological properties of the orbits will be preserved. If S and T are sets of real or complex numbers and $\varphi(x) = ax + b$, then an orbit of one function will be geometrically similar to an orbit of the other. The dynamical systems are truly identical, up to a scaling.

Example 2.8.2 Show that $f(x) = x^2 - 1$ is conjugate to $g(x) = \frac{1}{2}x^2 + 2x - 2$ via the conjugacy $\varphi(x) = \frac{1}{2}x + 1$.

Solution. We simply compute

$$f(\varphi(x)) = \left(\frac{1}{2}x + 1\right)^2 - 1 = \frac{1}{4}x^2 + x$$
$$\varphi(g(x)) = \frac{1}{2}\left(\frac{1}{2}x^2 + 2x - 2\right) + 1 = \frac{1}{4}x^2 + x.$$

Figure 2.8.3 illustrates the similarity between the two functions.

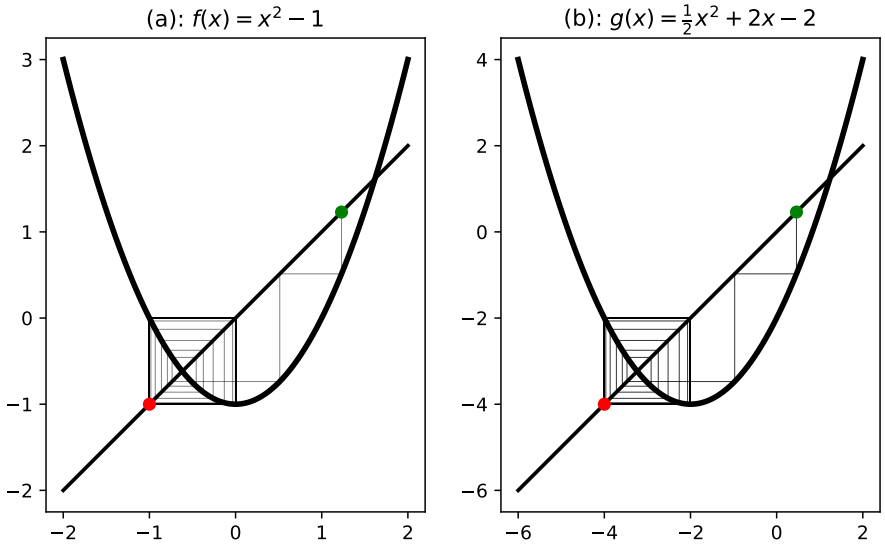

Figure 2.8.3 Cobweb plots for conjugate functions

□

If you suspect that f is conjugate to g via a conjugacy of the form $\varphi(x) = ax + b$, then you can find that conjugacy by setting $f(\varphi(x)) = \varphi(g(x))$. If you compare coefficients, you should get a system of equations that you can solve for a and b yielding the conjugacy.

Checkpoint 2.8.4 Find a conjugacy of the form $\varphi(x) = ax + b$ from $f(x) = x^2 - 2$ to $g(x) = 4x(1 - x)$.

Exercise Checkpoint 2.8.4 can be generalized. In fact, the quadratic family for $-2 \leq c \leq 1/4$ is identical to the logistic family for $1 \leq \lambda \leq 4$.

Checkpoint 2.8.5 Show that $f(x) = x^2 + (2\lambda - \lambda^2)/4$ is conjugate to $g(x) = \lambda x(1 - x)$ via the conjugacy $\varphi(x) = -\lambda x + \lambda/2$.

2.9 The doubling map and chaos

A glance at the cobweb plots of $f(x) = x^2 - 2$ and $g(x) = 4x(1 - x)$ shows that they both exhibit very complicated behavior. In fact, they are chaotic in a perfectly quantitative sense. In this section, we'll introduce the doubling map, which is (in a sense) the prototypical chaotic map. After seeing why it's chaotic, we'll show that it's conjugate to $f(x) = x^2 - 2$, implying that it too is chaotic.

2.9.1 The doubling map

Let H denote the half-open, half-closed unit interval:

$$H = [0, 1) = \{x \in \mathbb{R} : 0 \leq x < 1\}.$$

The doubling map d is the function $d : H \to H$ defined by

$$d(x) = 2x \bmod 1 = \begin{cases} 2x & 0 \leq x \leq 1/2 \\ 2x - 1 & 1/2 \leq x \leq 1 \end{cases}.$$

A graph of the doubling map together with a typical cobweb plot starting at an irrational number is shown in Figure 2.9.1

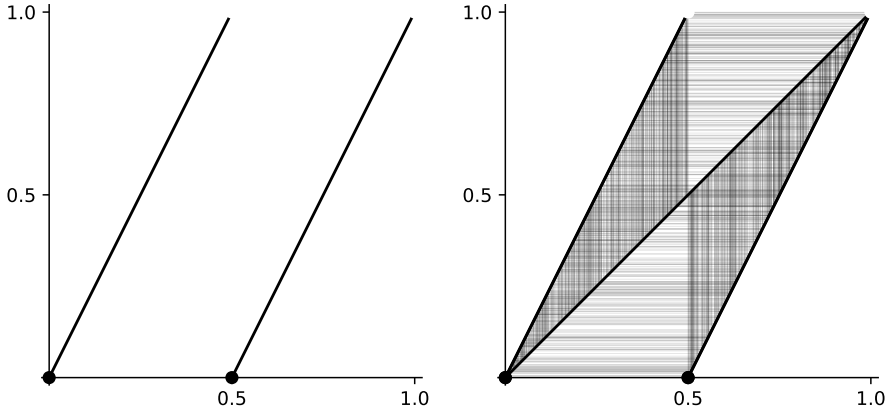

Figure 2.9.1 The doubling map

As it turns out, the doubling map is particularly easy to analyze if we consider its effect on the binary representation of a number. Suppose that $x \in H$ has binary representation

$$x = 0_{\dot{2}} b_1 b_2 b_3 b_4 b_5 \cdots,$$

where each b_i is a zero or a one. (In computer parlance, the b_is are called the *bits* of the number.)

Now, the effect of d on the binary representation of x is simple:

$$d(x) = 0_{\dot{2}} b_2 b_3 b_4 b_5 \cdots.$$

That is, the effect of d is to simply shift the bits of x to the left, discarding the bit that shifted into the ones place.

Of course, some numbers have multiple binary representations. For example,

$$0_{\dot{2}}1 = 0_{\dot{2}}0\overline{1} = \frac{1}{2},$$

The doubling map, though, is defined independently of the binary representation of the number. Thus, we would expect that the shift operation, when applied to different representations of the same number, should lead to the same result. For example, the shift operation when applied to the two representations of $1/2$ above yield.

$$0_{\dot{2}}\overline{0} = 0_{\dot{2}}\overline{1} = 0 \mod 1.$$

This characterization of the doubling map makes it very easy to find orbits with specific properties. Suppose, for example, we want an orbit of period 3. Simply pick (almost) any number of the form

$$x = 0_{\dot{2}}\overline{b_1 b_2 b_3}$$

The only caveat is that we can't have all b_is the same for that would lead to either zero (which is fixed) or one (which is not in H). As a concrete example,

$$x = 0_{\dot{2}}\overline{001} = \sum_{k=1}^{\infty} \frac{1}{8^k} = \frac{1}{7}$$

has period 3. In fact, it's easy to verify that

$$\frac{1}{7} \to \frac{2}{7} \to \frac{4}{7} \to \frac{8}{7} = \frac{1}{7} \bmod 1$$

under the doubling map.

Another nice feature of this representation is that there is a simple correspondence between the binary expansion of a number and its position in the unit interval. Every number with a binary expansion starting with a zero lies in the left half of the unit interval, while every number starting with a one lies in the right half. The first two bits of a number specify in which quarter of the interval the number lies; the first three bits specify in which eighth of the unit interval the number lies, as shown in Figure 2.9.2

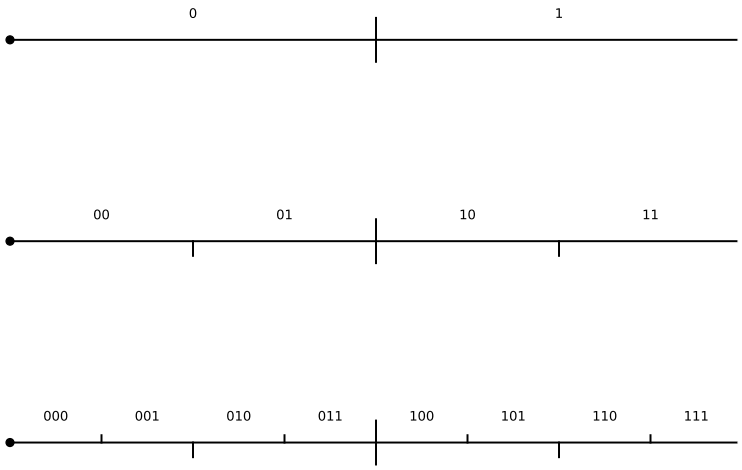

Figure 2.9.2 Dyadic intervals

More generally, given $n \in \mathbb{N}$, we can break the unit interval up into n pieces with length $1/2^n$ and endpoints $i/2^n$ for $i = 0, 1, \ldots, 2^n$. These are called *dyadic intervals* and their endpoints (number of the form $1/2^n$) are called dyadic rationals. The first n bits of a number specify in which n^{th} level dyadic interval that number lies. In fact, the left hand endpoint of a dyadic interval has a terminating binary expansion which tells you exactly the first n bits of all the points in that interval.

Now, suppose that

$$x_1 = 0_{\dot{2}} b_1 b_2 \cdots b_n b_{n+1} b'_{n+2} \cdots \quad \text{and} \quad x_2 = 0_{\dot{2}} b_1 b_2 \cdots b_n b'_{n+1} b'_{n+2} \cdots .$$

Thus, the binary expansions of x_1 and x_2 agree up to at least the n^{th} spot but potentially disagree after that. Then, our geometric understanding of dyadic intervals allows us to easily see that,

$$|x_1 - x_2| \leq \frac{1}{2^n}.$$

Of course, there's also a simple algebraic proof of this fact, based on the fact that the bits cancel for $k \leq n$

$$\begin{aligned} |x_1 - x_2| &= \left| \sum_{k=n+1}^{\infty} \frac{b_k - b'_k}{2^k} \right| \\ &\leq \sum_{k=n+1}^{\infty} \frac{|b_k - b'_k|}{2^k} \leq \sum_{k=n+1}^{\infty} \frac{1}{2^k} = \frac{1}{2^n}. \end{aligned}$$

2.9.2 Chaos

We can now prove three claims about the doubling map that, together, assert that the doubling map displays some of the essential features of chaos. First, we'll need to state and prove a lemma.

Lemma 2.9.3 *Suppose that*

$$x = 0_{\dot{2}} b_1 b_2 b_3 \cdots \quad \textit{and} \quad y = 0_{\dot{2}} b'_1 b'_2 b'_3 \cdots$$

are elements of H that satisfy $b_1 \neq b'_1$ but $b_2 = b'_2$. Then $|x - y| \geq 1/4$.

Proof. Computing the difference using the binary representations, taking into account that the terms disagree in the first spot and agree in the second, and finally applying the reverse triangle inequality, we get

$$\begin{aligned} |x - y| &= \left| \sum_{i=1}^{\infty} \frac{b_i - b'_i}{2^n} \right| = \left| \pm\frac{1}{2} + \sum_{i=3}^{\infty} \frac{b_i - b'_i}{2^n} \right| \\ &\geq \left| \left| \pm\frac{1}{2} \right| - \left| \sum_{i=3}^{\infty} \frac{b_i - b'_i}{2^n} \right| \right| \geq \left| \frac{1}{2} - \frac{1}{4} \right| = \frac{1}{4}. \end{aligned}$$

■

A geometric interpretation of this lemma is as follows. The fact that the two points disagree in the first spot means that they cannot lie in the same half of H. The fact that they do agree in the second spot means that they lie in the same quarter relative to their half, as shown in Figure 2.9.4. Clearly, any two such points cannot be within $1/4$ of one another.

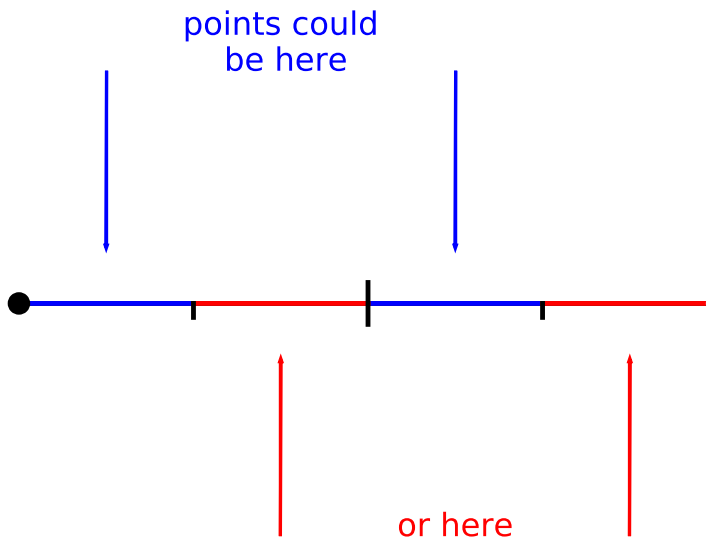

Figure 2.9.4 Possible positions of points in lemma Lemma 2.9.3

Claim 2.9.5 Sensitive dependence on initial conditions. *For every $x \in H$ and for every $\varepsilon > 0$, there is some $y \in H$ and an $n \in \mathbb{N}$ such that $|x - y| < \varepsilon$ yet $|d^n(x) - d^n(y)| \geq 1/4$.*

Proof. Choose $n \in \mathbb{N}$ large enough so that $1/2^n < \varepsilon$. Now suppose that $x \in H$ has binary expansion

$$x = 0_{\dot{2}} b_1 b_2 \cdots b_n b_{n+1} b_{n+2} \cdots .$$

Define $y \in H$ so that

$$y = 0_{\dot{2}} b_1 b_2 \cdots b_n (1 - b_{n+1}) b_{n+2} \cdots .$$

That is, the bits of y agree with those of x in the first n spots, disagree with x in the $(n+1)^{\text{st}}$ spot, and finally agree with x again in the $(n+2)^{\text{nd}}$ spot.

Then, the numbers $d^n(x)$ and $d^n(y)$ satisfy the hypotheses of lemma Lemma 2.9.3, thus $|d^n(x) - d^n(y)| \geq 1/4$. ■

Claim 2.9.6 Denseness of periodic orbits. *For every open interval $I \subset H$, there is some periodic orbit with an element in I.*

Proof. Let $x \in I$ and choose $n \in \mathbb{N}$ large enough so that

$$(x, x + 1/2^n) \subset I.$$

Now suppose that

$$x = 0_{\dot{2}} b_1 b_2 \cdots b_n \cdots .$$

Then,

$$\hat{x} = 0_{\dot{2}} \overline{b_1 b_2 \cdots b_n}$$

is a periodic point in I. ■

Claim 2.9.7 A dense orbit. *There is a point $x \in H$ with the property that, for every open interval $I \subset H$, there is some iterate of x in I.*
Proof. We'll define x by specifying its binary expansion. We begin by writing down all possible *finite* binary strings:

$$0,\ 1,\ \ 00,\ 01,\ 10,\ 11,\ \ 000,\ 001,\ 010,\ 011,\ 100,\ 101,\ 110,\ 111, \ldots$$

We then concatenate these to obtain the binary representation of x

$$x = 0_{\dot{2}}0\,1\,00\,01\,10\,11\,000\,001\,010\,011\,100\,101\,110\,111\ldots$$

Now, let $I \subset H$ be an open interval. We claim that there is some iterate of x in I. To see that, let L denote the length of I and choose $n \in \mathbb{N}$ large enough so that

$$\frac{1}{2^n} < \frac{1}{2}L.$$

Let i be the smallest integer such that $i/2^n \in I$. Note that we must also have $(i+1)/2^n \in I$. Thus, the dyadic interval $[i/2^n, (i+1)/2^n)$ is wholly contained in I and the first n bits of every point in that interval agree with $i/2^n$. So, let

$$\frac{i}{2^n} = 0_{\dot{2}}b_1b_2\cdots b_n$$

and note that, by construction, the string $b_1b_2\cdots b_n$ appears somewhere in the binary expansion of x. Thus, we can apply the doubling function to the point x some number, say m, times to obtain

$$d^m(x) = 0_{\dot{2}}b_1b_2\cdots b_n\cdots .$$

The number $d^m(x)$ is then an iterate of x that lies in I. ■

While there is no truly universally accepted definition of chaos, claims Claim 2.9.5, Claim 2.9.6, and Claim 2.9.7 are generally agreed to express some of the essential features of chaos. We might think of them as representing:

- Instability,
- Structure, and
- Indecomposability

2.9.3 A chaotic quadratic

Let $f(x) = x^2 - 2$. We now show that f is semi-conjugate to the doubling map d under the semi-conjugacy $\varphi(x) = 2\cos(2\pi x)$. As a result, φ maps all the orbit types that d has to an orbit of f with similar properties. Thus, f is chaotic.

Claim 2.9.8 *The map $f(x) = x^2 - 2$ is semi-conjugate to the doubling map $d(x) = 2x \bmod 1$ under the semi-conjugacy $\varphi(x) = 2\cos(2\pi x)$.*
Proof. We must simply show that $f \circ \varphi = \varphi \circ d$, so let's compute. First,

$$f(\varphi(x)) = 2(2\cos(2\pi x))^2 - 2 = 4\cos^2(2\pi x) - 2.$$

Well, that was easy. The next part is a little trickier - we just need to apply a couple of trig identities and use the fact that we can drop the mods inside the squared trig functions due to the symmetries of those functions.

$$\begin{aligned}\varphi(d(x)) &= 2\cos(2\pi(2x \bmod 1)) \\ &= 2(\cos^2(\pi(2x \bmod 1)) - \sin^2(\pi(2x \bmod 1)))\end{aligned}$$

$$
\begin{aligned}
&= 2(\cos^2(2\pi x) - \sin^2(2\pi x)) \\
&= 2(\cos^2(2\pi x) - (1 - \cos^2(2\pi x))) \\
&= 2(2\cos^2(2\pi x) - 1) \\
&= 4\cos^2(2\pi x) - 2
\end{aligned}
$$

■

Again, the key fact about semi-conjugacy is that φ maps orbits of d to orbits of f. Thus, since d has a dense orbit f too has a dense orbit. Here's a concrete example illustrating this idea.

Example 2.9.9 Find a point of period 11 for the chaotic quadratic $f(x) = x^2 - 2$.

Solution. First, it's easy to find an orbit of period 11 for the doubling map. One example is

$$0_{\dot{2}}00000000001 = \sum_{k=1}^{\infty} \frac{1}{2^{11k}} = \frac{1}{2047}.$$

The point behind conjugacy is that $\varphi(1/2047) = 2\cos(2\pi/2047)$ will be a point of period 11 for f. The reader is advised to check this numerically! □

2.10 Tent maps and the Cantor set

To this point, we've focused on functions that preserve an interval. In the logistic family $f_\lambda(x) = \lambda x(1-x)$, for example, we've only considered values of λ that satisfy $0 \le \lambda \le 4$ so that $f_\lambda : [0,1] \to [0,1]$. In this section, we allow for the possibility that $\lambda > 4$. As we'll see, there is no longer an invariant interval, but rather an invariant set that we'll come to call a *Cantor set*.

2.10.1 Tent maps

For each positive number a, we define the tent map T_a on $[0,1]$ by

$$T_a(x) = \begin{cases} a\,x & 0 \le x \le 1/2 \\ a\,(1-x) & 1/2 \le x \le 1. \end{cases}$$

The graphs of several tent maps are shown in Figure 2.10.1. Note that when $a < 2$, the function T_a maps $[0,1]$ *into* itself and that when $a = 2$, the function T_a maps $[0,1]$ *onto* itself. When $a > 2$, the unit interval is no longer invariant

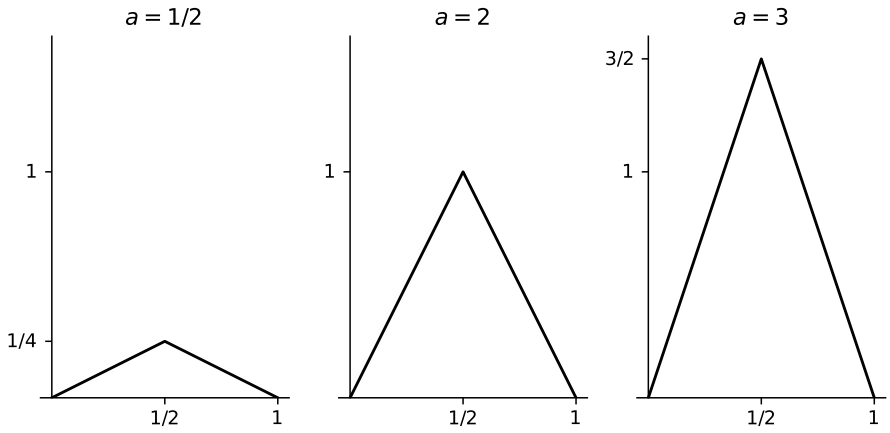

Figure 2.10.1 The graphs of several tent maps

Of course, we are interested in iterating T_a for various values of a. Figure 2.10.2 shows cobweb plots of tent maps for several choices of a in the interval $(0, 2]$. It appears that T_a should have a single attractive fixed point at the origin when $a < 1/2$ and that T_a should be chaotic on some interval when $a > 1/2$.

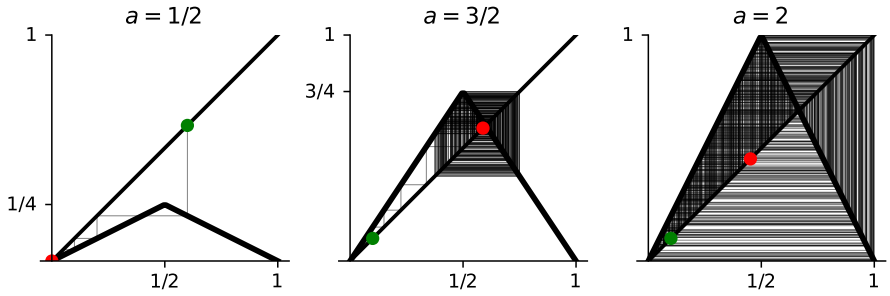

Figure 2.10.2 Cobweb plots for several tent maps

It's not too difficult to *prove* that T_2 is chaotic on the unit interval.

Claim 2.10.3 Chaotic tent maps. *The doubling map d is semi-conjugate to the tent map T_2 under the semi-conjugacy T_2 itself. Thus, T_2 is chaotic on the unit interval.First, to be clear, we are claiming that*

$$T_2 \circ T_2 = T_2 \circ d.$$

Diagrammatically:

$$\begin{array}{ccc} H & \xrightarrow{d} & H \\ \downarrow{\scriptstyle T_2} & & \downarrow{\scriptstyle T_2} \\ I & \xrightarrow{T_2} & I \end{array}$$

One quick way to check if there's any sense to this is to simply graph both expressions; you should get something like Figure 2.10.4. You can do this using Desmos like so: https://www.desmos.com/calculator/nhmcgixtmz.

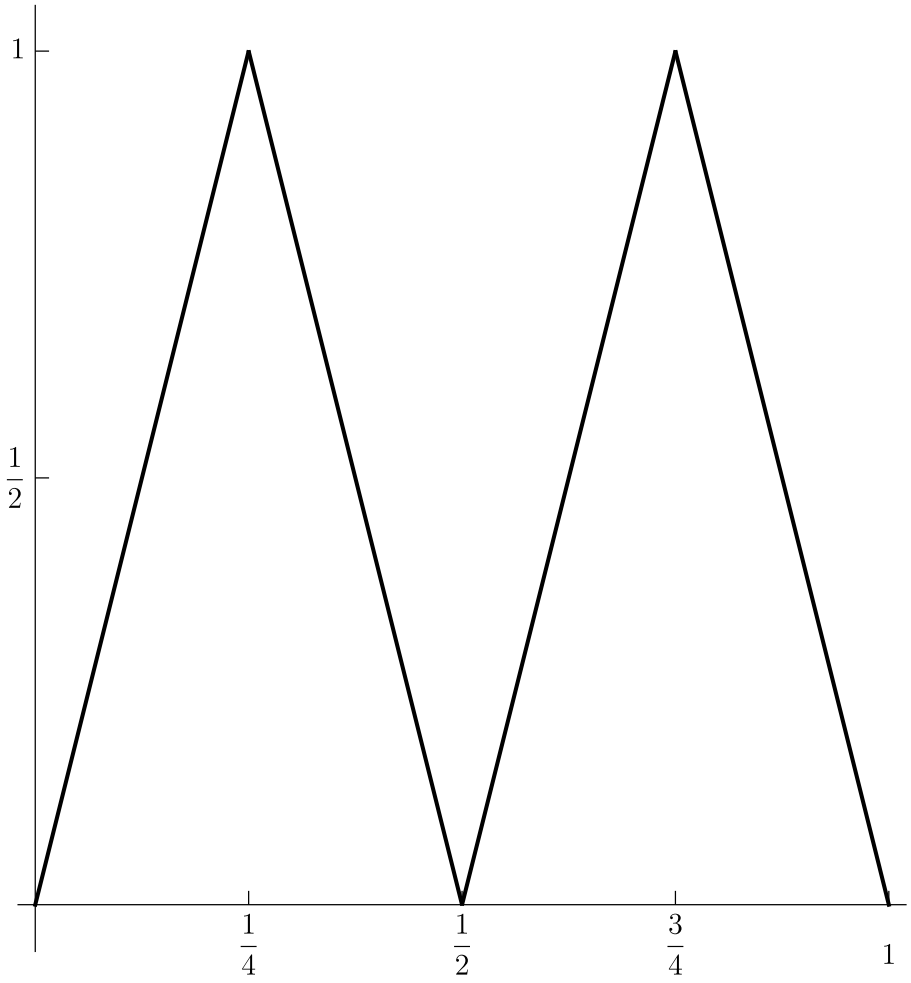

Figure 2.10.4 *The common graph of $T_2 \circ T_2$ and $T_2 \circ d$*

The graph even reveals the common value we should shoot for, namely

$$T_2 \circ T_2(x) = T_2 \circ d(x) = \begin{cases} 4x & 0 \leq x \leq \frac{1}{4} \\ 4\left(\frac{1}{2} - x\right) & \frac{1}{4} \leq x \leq \frac{1}{2} \\ 4\left(x - \frac{1}{2}\right) & \frac{1}{2} \leq x \leq \frac{3}{4} \\ 4(1-x) & \frac{3}{4} \leq x \leq 1 \end{cases}.$$

To do so, we simply check that the definitions work out over each subinterval in the piecewise definition. Over the third subinterval $1/2 \leq x \leq 3/4$, for example, we have

$$T_2 \circ d(x) = 2(2x-1) = 4x - 1$$

and

$$T_2 \circ T_2(x) = 2(1 - 2(1-x)) = 4x - 1.$$

When checking these inequalities, simply check where the output of the inner function lies to ensure that you apply the correct formula for the outer function.

Just as with the chaotic quadratic, Claim 2.10.3 implies that the tent map T_2 is chaotic. You can apply the conjugacy function T_2 itself to a periodic point of d to find a period point of T_2. For example, $T_2(0.\overline{001}) = 2/7$ is a period three point for T_2, as you can easily check.

Finally, it's worth noting that we can study T_2 by examining its effect on binary expansions.

Checkpoint 2.10.5 The tent map applied to binary expansions. Show that

$$T_2(0_{\dot{2}}b_1b_2b_3b_4\cdots) = \begin{cases} 0_{\dot{2}}b_2b_3b_4\cdots & b_1 = 0 \\ 0_{\dot{2}}(1-b_2)(1-b_3)(1-b_4)\cdots & b_1 = 1 \end{cases}.$$

In other words, the map shifts the digits (like the doubling map) when the first bit is zero but shifts bits *and* swaps the remaining bits when the first bit is one.

2.10.2 Escaping orbits

We now begin our investigation of the *tall tent map* T_3, whose graph is shown in Figure 2.10.6. We see there the critical issue that the unit interval is no longer invariant. As we see, the open interval $(1/3, 2/3)$ maps to values larger than 1.

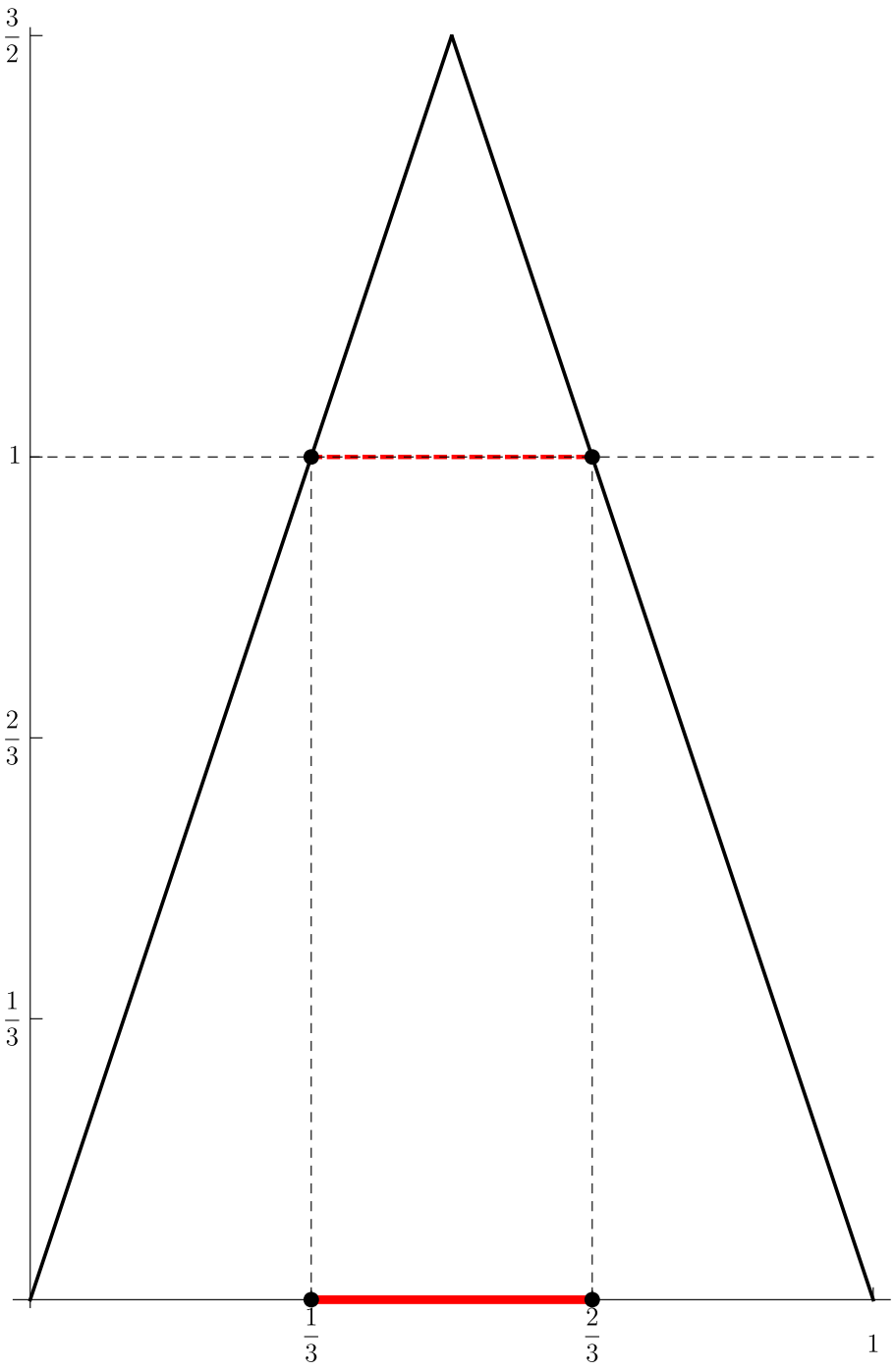

Figure 2.10.6 The tall tent map

As it turns out, there *is* a very important invariant set that leads us naturally to the study of fractal geometry. To find it, we'll continue to iterate T_3 to see what other points might escape. To that end, the graphs of T_3^2 and T_3^3 are shown in

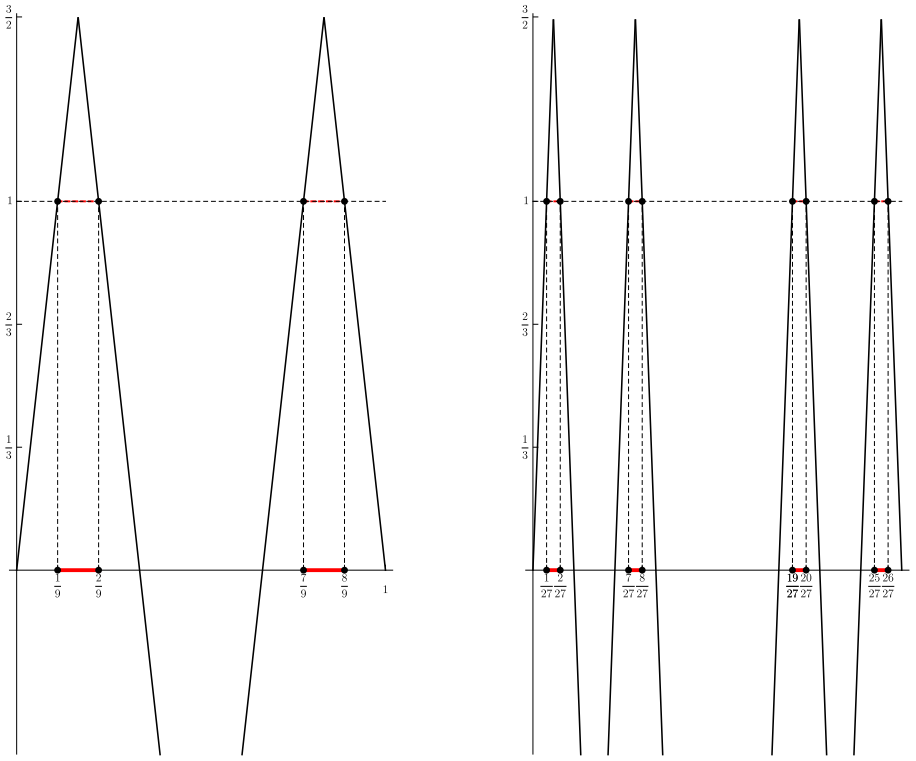

Figure 2.10.7 The graphs of T_3^2 and T_3^3

Note that the middle third has escaped from each of the closed intervals $[0, 1/3]$ and $[1/3, 1]$ that remained after the first iteration, leaving us with four intervals

$$[0, 1/9], [2/9, 1/3], [2/3, 7/9], \text{ and } [7/9, 1].$$

A similar process happens after the next step and, generally, after the n^{th} step. Let us denote the set of all intervals that remain after n iterations by C_n. Then C_0 is the unit interval and

$$C_1 = [0, 1/3] \bigcup [2/3, 1].$$

Generally, C_n consists of 2^n intervals of length $1/3^n$ and the invariant set for T_3 is exactly

$$C = \bigcap_{n=0}^{\infty} C_n.$$

It might not seem like there's much to C but it turns out to be a very rich set and the protypical fractal, as we'll meet in the next chapter. It's called the *Cantor set*.

2.11 A few notes on computation

Many of the results in these notes have been illustrated on the computer and some of the exercises require a computational approach. Whenever using the computer, it is always wise to examine the results critically. Here's a simple numerical example where things clearly go awry. In it, we are iterating the function $f(x) = x^2 - 9.1x + 1$ from the fixed point $x_0 = 0.1$. We should generate just a constant sequence.

```
x = 0.1
for i in range(20):
    x = x**2 - 9.1*x+ 1
    print(x)
```

```
# Output:
0.09999999999999998
0.1000000000000002
0.0999999999999982
0.10000000000001608
0.09999999999985698
0.10000000000127285
0.0999999999886717
0.10000000010082188
0.0999999991026852
0.10000000798610176
0.09999992892369436
0.10000063257912528
0.09999437004618517
0.1000501066206484
0.09955405358690272
0.10396912194476915
0.06469056862056699
0.41550069522129274
-2.608415498784386
31.540412453236513
```

Uh-oh!

It must be understood that this is a simple consequence of the nature of floating point arithmetic. Part of the issue is that the decimal number 0.1 or 1/10 is not exactly representable in binary. In fact,

$$\frac{1}{10} = 0_{\hat{2}}0\overline{0011} = \frac{1}{2}\sum_{k=1}^{\infty}\frac{3}{16^k}.$$

Thus, the computer *must* introduce round-off error in the computation. Furthermore, 0.1 is a repelling fixed point of the function. Thus, that round-off error is magnified with each iteration. Our study of dynamics has illuminated a critical issue in numerical computation!

In the terminology of numerical analysis, computation near an attractive fixed point is stable while computation near a repelling fixed point is unstable. Generally speaking, stable computation is trustworthy while unstable computation is not. The implication for the pictures that we see here is that illustration of attractive behavior should be just fine. In Figure 2.3.1, for example, images (b) and (c) illustrate attraction to a fixed point that not only involves stable computation but also agrees with our theoretical development. We are happy with those figures. The cobweb plot shown in Figure 2.3.1 (d), however, should frankly be viewed with some suspicion.

The same can be said for the bifurcation diagram in Figure 2.7.2. In much of that image, we see a gray smear indicating chaos. How can we trust that? Well, first, theory tells us that there really *is* chaos. That is, there are orbits that are dense in some interval for many c values. Furthermore, much of the image shows attractive regions and we can be confident in that portion.

In fact, in many of the images that we will generate later - Julia sets, the Mandelbrot set, and similar images - the stable region dominates. Thus, we can be confident in overall image because the unstable region is the complement of

the stable region. We might not be confident in computations involving some particular point, but we can be confident in the overall picture. (This will, perhaps, be more clear as we move into complex dynamics.)

Nonetheless, sometimes we want to experiment with genuinely unstable dynamics. One way to improve our confidence in these kinds of computations is to use high precision numbers. Consider, for example, the cobweb plot of the doubling map shown in Figure 2.9.1. A naive approach to generate the first few terms of an orbit associated with the doubling map might be as follows:

```
import numpy as np
x = 1/np.pi
for i in range(55):
    x = 2*x%1
    print(x)
```

```
# Truncated output
0.636619772368
0.273239544735
0.54647908947
...
0.375
0.75
0.5
0.0
0.0
```

We've reached the fixed point zero and now we're stuck! Even if we iterate 1000 times, we'll generate a cobweb plot that looks like Figure 2.11.1

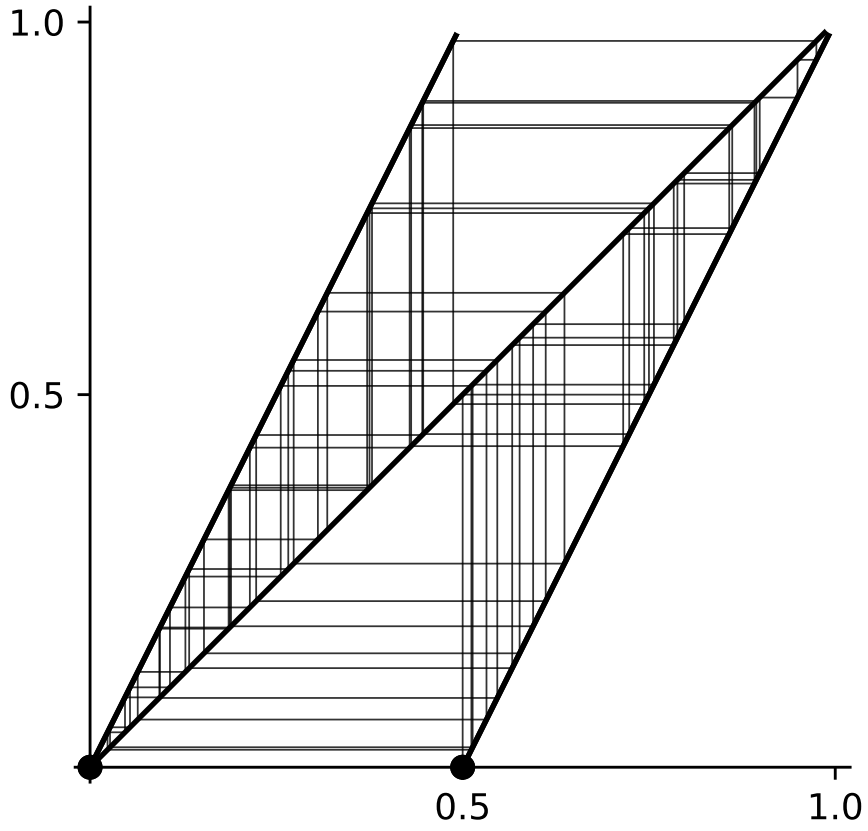

Figure 2.11.1 An inaccurate cobweb plot

The cobweb plot shown in Figure 2.9.1 was generated using the mpmath multi-precision library for Python with code that looked something like so:

```
from mpmath import mp
mp.prec = 1000
x = 1/mp.pi
for i in range(1000):
    x = 2*x%1
    print(float(x))
```

```
# Truncated output
0.6366197723675813
0.27323954473516265
0.5464790894703253
...
0.375
0.75
0.5
0.0
0.0
```

While the truncated output looks the same, note that this was after 1000 iterates. This behavior makes perfect sense if you understand that the doubling map loses one bit of precision with every iterate.

2.12 Exercises

1. Let $f : \mathbb{R} \to \mathbb{R}$ be continuously differentiable. We say that x_0 is a simple root of f if $f(x_0) = 0$ and $f'(x_0) \neq 0$. Show that if x_0 is a simple root of f, then x_0 is a super-attracting fixed point of the Newton's method iteration function N for f.

2. Find an example of a continuously differentiable function $f : \mathbb{R} \to \mathbb{R}$ that attracts no critical point.

Hint. Draw a graph. Of course, you can't violate theorem Theorem 2.6.1.

3. Let $f(x) = x^2 - 4x + 5$. Show that f has a super-attractive orbit of period 2.

4. Let $f(x) = 3x^2 - 6x + 3.415$. Find all attractive orbits of f.

5. Find a value of c such that $f_c(x) = x^2 + c$ is affinely conjugate to $g(x) = (x - 1)(x + 2)$. Show that both functions have neutral fixed points.

6. Find an orbit of period 11 for the function $g(x) = 4x(1 - x)$

7. We wish to find a number $x_0 \in H = [0, 1)$ whose orbit is dense in H under iteration of $g(x) = 4x(1 - x)$.

(a) Outline a strategy for finding x_0.

(b) Find a decimal approximation to x_0 that is valid to 10 decimal places.

Chapter 3

Self-similarity

Introduction. At the end of the last chapter, we met a strange set called the Cantor Set. In this chapter, we'll see that the Cantor set is the prototypical member of a broad class of sets called the self-similar sets. Intuitively, a self-similar set is one that is composed of smaller copies of itself. To prepare for our exploration of these sets, we'll begin with another look at the Cantor set itself.

3.1 Another look at the Cantor set

In the last chapter we met (in an as yet uncompleted section) a strange set called the Cantor set. As it turns out, this is the protypical self-similar set so let's take a look a closer look.

Cantor constructed his set in the 1880's to help him understand a problem in Fourier series. While the set seemed unnatural to mathematicians of the time, it has become a central example in real analysis. Cantor's construction is as follows. Start with the unit interval $I = [0, 1]$, the set of all real numbers between 0 and 1 inclusive. Remove the open middle third $\left(\frac{1}{3}, \frac{2}{3}\right)$ of the interval I to obtain the two intervals $I_1 = \left[0, \frac{1}{3}\right]$ and $I_2 = \left[\frac{2}{3}, 1\right]$. Then remove the open middle thirds of the intervals I_1 and I_2 to obtain the intervals $I_{1,1} = \left[0, \frac{1}{9}\right]$, $I_{1,2} = \left[\frac{2}{9}, \frac{1}{3}\right]$, $I_{2,1} = \left[\frac{2}{3}, \frac{7}{9}\right]$, and $I_{2,2} = \left[\frac{8}{9}, 1\right]$. Repeating this process inductively, we obtain 2^n intervals of length $1/3^n$ at the n^{th} stage. The cantor set C consists of all those points in I which are never removed at any stage. More precisely, if C_n denotes the union of all of the intervals left after the n^{th} stage of the construction, then

$$C = \bigcap_{n=1}^{\infty} C_n.$$

This process is illustrated in figure Figure 3.1.1.

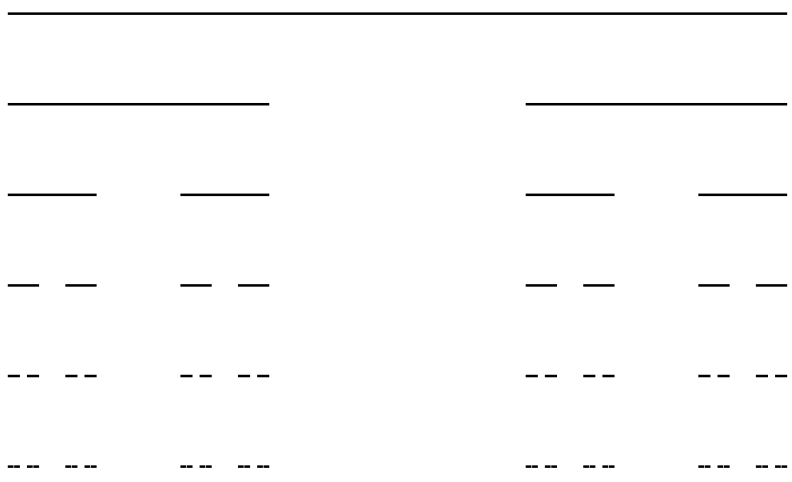

Figure 3.1.1 Construction of the Cantor set

It's clear that C should be self-similar, since the effect of the construction on the intervals I_1 and I_2 is the same as the effect on the whole interval I, but on a smaller scale. Thus C consists of two copies of itself scaled by the factor $1/3$.

The Cantor set has many non-intuitive properties. In some sense, it seems very small; if we were to assign a "length" to it, that length would have to be zero. Indeed, by it's very construction it is contained in 2^n intervals of length $1/3^n$. Thus the length of C_n is $2^n/3^n$ which tends to zero as $n \to \infty$. Since C is contained in C_n for all n, the length of C must be zero. It might even appear that there is nothing left in C after tossing so much out of the original interval I. In reality, the Cantor set is a very rich set with infinitely many points. Recall that only open intervals are removed during the construction. Thus all of the infinitely many endpoints remain. For example, $1/3$, $2/3$, and $80/81$ are all in C. There are still many more points in C, however.

There is a general technique for finding points of the Cantor set. The first stage in the construction consists of the two intervals I_0 on the left and I_2 on the right. Choose one and discard the other. Now the interval we chose, say I_0 for concreteness, contains two disjoint intervals, $I_{0,0}$ and $I_{0,2}$, in the next stage of the construction. Choose one of those and discard the other. If we continue this process inductively, we obtain a nested sequence of closed intervals which will collapse down to a point in the Cantor set. In this way, each sequence of zeros and twos forms an address for a point in the Cantor set; there is a one-to-one correspondence between the points and the addresses.

The addressing scheme for the Cantor set allows us to prove a number of interesting properties. To see that there are points in the Cantor set that are not endpoints of any of the removed intervals, simply note that the those endpoints correspond exactly to addresses that end in all zeros or all twos, but there are many other possible addresses - $(0, 2, 0, 2, 0, 2, \ldots)$ comes to mind.

From a more advanced perspective, the addressing scheme allows us to show that the Cantor set is uncountable, since the set of addresses certainly is.

To find specific points in the Cantor set, we might regard the addresses as expansions in base 3, also called *ternary expansions*. To see this note that everything in the first third of the unit inveral can be written with a leading 0 while everything in the last third of the unit interval can be written with a leading 2. Each of those breaks up in a similar manner, as shown in

Figure 3.1.2.

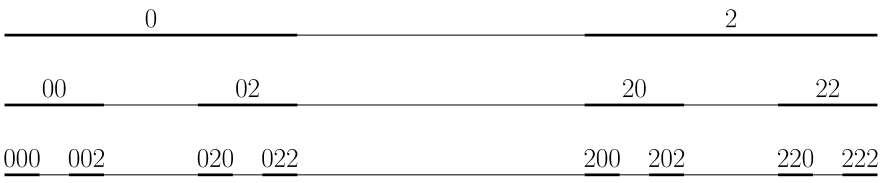

Figure 3.1.2 Addresses for the Cantor set

Thus, we now see that the Cantor set is exactly the set of all numbers in the unit interval that can be written using only zeros and twos in its ternary expansion. Furthermore, one particular point in the Cantor set that is *not* one of the removed endpoints is

$$\begin{aligned} 0_{\overline{3}}\overline{02} &= \sum_{n=1}^{\infty} \frac{2}{3^{2n}} = 2\sum_{n=1}^{\infty} \left(\frac{1}{9}\right)^n \\ &= 2\frac{1/9}{1-1/9} = 2\frac{1/9}{8/9} = 2 \times \frac{1}{8} = 1/4. \end{aligned}$$

Checkpoint 3.1.3 Use repeating blocks of length three to find three different numbers in the Cantor set that are not endpoints of any of the removed intervals.

A major question that we will address later in the book asks, "What is the dimension of the Cantor set?" Certainly, it is too small to be considered a one dimensional set; it is just a scattering of points along the unit interval with length zero. It is uncountable, however; perhaps it is too large to be considered as zero dimensional. We will develop a notion of "fractal dimension" that quantitatively captures this in-betweeness.

3.2 The Sierpinski gasket

A similar process can be used to construct the Sierpinski gasket, also called the Sierpinski triangle. We start with a closed, filled in equilateral triangle. The line segments joining the midpoints of the sides of this triangle divide it into four equilateral sub-triangles. We can discard the one in the center, keep the others and then repeat the process on the remaining triangles. This process is illustrated in figure Figure 3.2.1.

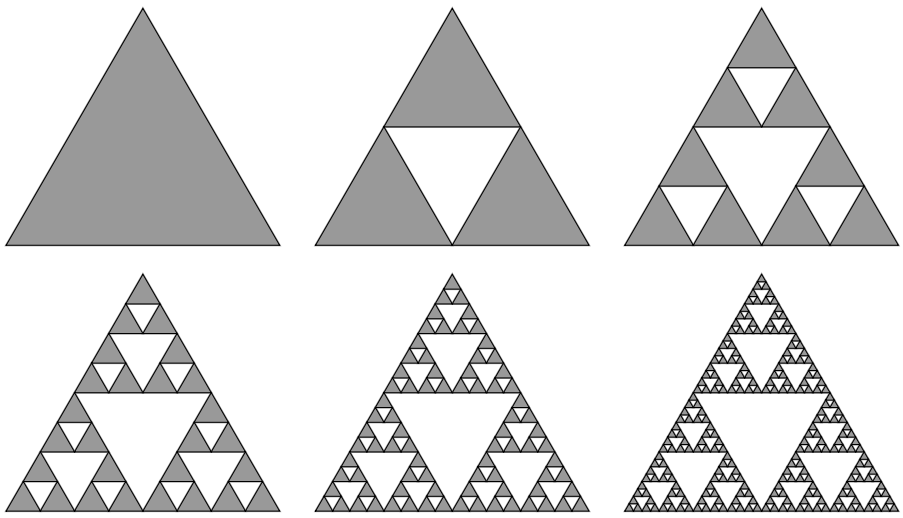

Figure 3.2.1 Construction of the Sierpinski gasket

Many of the comments regarding the Cantor set are applicable to the Sierpinski gasket as well. It is a self-similar set consisting of 3 copies of itself, each scaled by the factor 1/2. Its dimensional properties are, in a sense, between dimension one and dimension two.

3.3 Iterated function systems

We now provide a careful mathematical definition of one of the central tools of fractal geometry - the *iterated function system* or *IFS*. The idea behind the IFS technique is to focus on how the parts of a set might fit together to create the whole. Thus, we focus on what makes two sets similar; we do this with the notion of a similarity. An IFS is a collection of similarities and a self-similar set is the attractor of an IFS. Thus, we actually need to make several definitions, which we will illustrate with examples afterwards.

3.3.1 IFS definitions

We assume we are working in $\mathbb{R}^n$, n-dimensional Euclidean space. The dimension n will almost always be 2 in this text, although $n = 1$ might be the natural setting for the Cantor set. Our sets will generally be subsets of the plane. Given x in $\mathbb{R}^n$, $|x|$ will refer to the distance from x to the origin. Thus, $|x - y|$ represents the distance from x to y.

Definition 3.3.1 A function $f : \mathbb{R}^n \to \mathbb{R}^n$ will be called a *contraction* if there is a real number r such that $0 < r < 1$ and $|f(x) - f(y)| \leq r|x - y|$ for all x, y in $\mathbb{R}^n$. If $|f(x) - f(y)| = r|x - y|$ for all x, y in $\mathbb{R}^n$, then f is called a *similarity* and the number r is called its *contraction ratio.* $\Diamond$

Definition 3.3.2 An iterated function system on $\mathbb{R}^n$, or IFS, is a non-empty, finite collection of contractions of $\mathbb{R}^n$. We will usually express an IFS in the form

$$\{f_i\}_{i=1}^{m},$$

where m is the number of functions in the IFS. $\Diamond$

We will often be interested in the case where all functions in the IFS are similarities, but that's not a requirement not is it a hypothesis in the following theorem.

Theorem 3.3.3 Existence of invariant sets. *For any IFS*

$$\{f_i\}_{i=1}^m,$$

there is a unique non-empty, closed, bounded subset E of $\mathbb{R}^n$ such that

$$E = \bigcup_{i=1}^{m} f_i(E). \tag{3.3.1}$$

The set E defined is called the invariant set *or* attractor *of the IFS. If the IFS consists entirely of contractive similarities, then E is called* self-similar.

3.3.2 Interpretation

We will make frequent use of Theorem 3.3.3, so let's make sure we understand what it is saying. First, the notation $f_i(E)$ refers to the image of the set E under the action of the function f_i. For the general function $f : \mathbb{R}^n \to \mathbb{R}^n$ and set $E \subset \mathbb{R}^n$,

$$f(E) = \{f(x) : x \in E\}.$$

If the function f happens to be a similarity, then application of f transforms E into a set which is geometrically similar to E. This idea is illustrated in figure Figure 3.3.4.

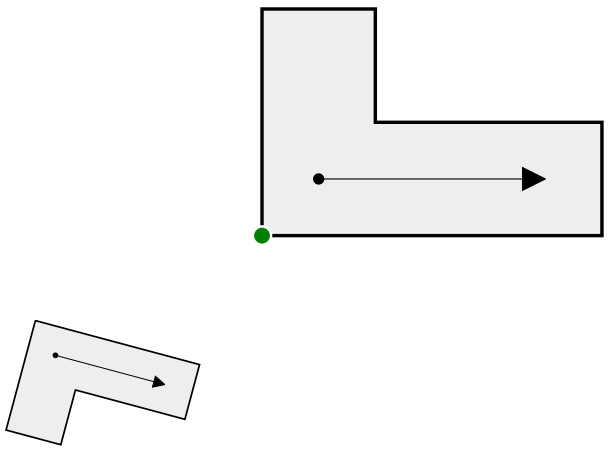

Figure 3.3.4 The action of a contractive similarity on a set E in the plane

If a set is self-similar, then it can be decomposed into several parts - each of which is geometrically similar to the whole. The decomposition of the Sierpinski triangle, for example, is shown if Figure 3.3.5

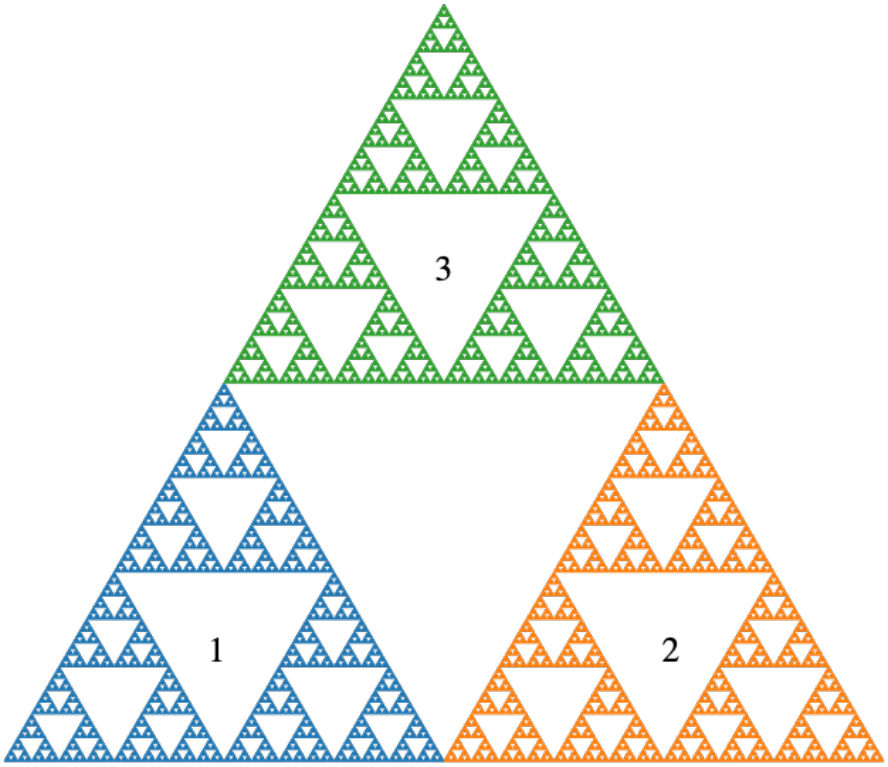

Figure 3.3.5 Decomposition of the Sierpinski triangle

3.4 Applying iterated function systems

We turn now to the question of how to generate self-similar images. Ultimately, we will consider several algorithms for this purpose. It turns out they are all related to theoretical questions presented in the next chapter. For example, we will present a constructive proof that any IFS yields a unique closed, bounded invariant set. This construction is essentially our first algorithm, which we call the basic deterministic algorithm.

Given an iterated function system $\{f_i\}_{i=1}^m$, we can define a function T which maps the collection of closed, bounded subsets of $\mathbb{R}^n$ to itself by

$$T(F) = \bigcup_{i=1}^{m} f_i(F). \tag{3.4.1}$$

We can use the function T to generate the invariant set using *iteration*, an important theme in fractal geometry. To iterate a function f which maps a set X into itself, start with a some point x_0 in X, set $x_1 = f(x_0)$, and for larger natural numbers n set $x_n = f(x_{n-1})$. Equivalently, we can write $x_n = f^n(x_0)$, the result of n-fold composition of f applied to x_0. In the current situation, the set X is the set of all non-empty, closed, bounded subsets of $\mathbb{R}^n$ and the function is the transformation T defined in equation (3.4.1). It turns out that if we start with an arbitrary non-empty, closed, bounded subset of $\mathbb{R}^n$ and iterate the function T, then the generated sequence of sets converges in a natural sense to the invariant set of the IFS. We'll examine this statement more carefully in the next chapter, but in this chapter we'll demonstrate the statement visually through experimentation.

Figure Figure 3.2.1, for example, was generated in exactly the manner described above. The initial set F was taken to be the solid equilateral triangle with vertices $(0,0), (1,0)$, and $\left(1/2, \sqrt{3}/2\right)$. The transformation T is obtained by applying equation (3.4.1) to the Sierpinski IFS. The subsequent images correspond to application of T up to 5 times.

The previous example might be misleading, since the relationship between the initial approximation and the fractal attractor is so close. Indeed, the choice of initial approximation has no effect on the final attractor, although the images illustrating convergence can be affected. Figure Figure 3.4.1, for example, illustrates the sequence of sets generated by applying this same IFS to the initial approximation consisting of the single point at the origin. Successive approximations fill the set out more and more completely. Figure Figure 3.4.2 illustrates the algorithm taking the initial set to be the shape from figure Figure 3.3.4.

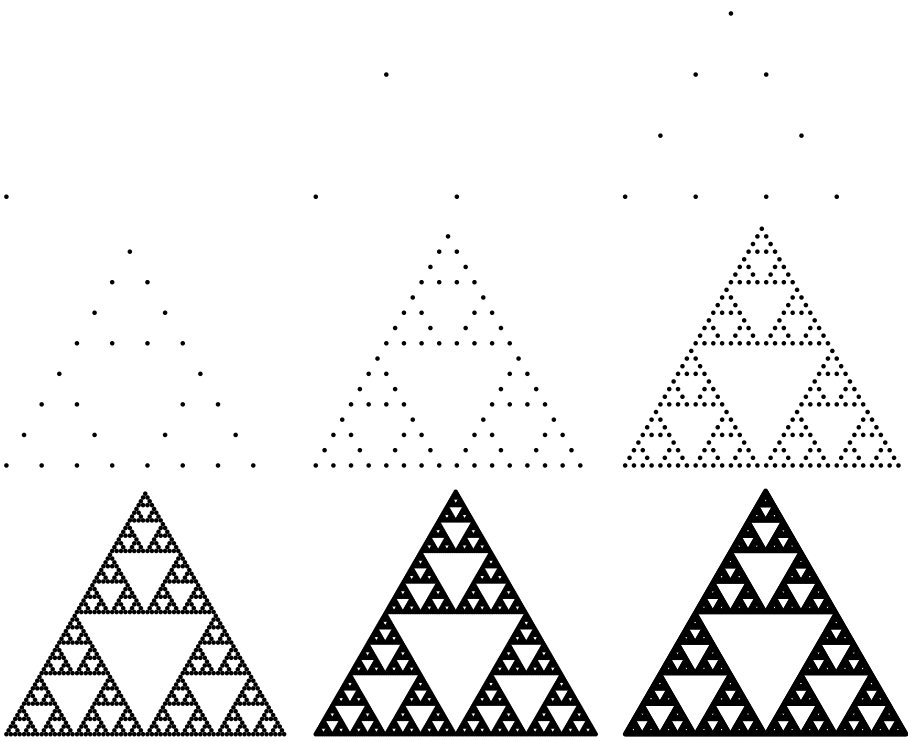

Figure 3.4.1 Approximating the Sierpinski gasket with finite sets of points.

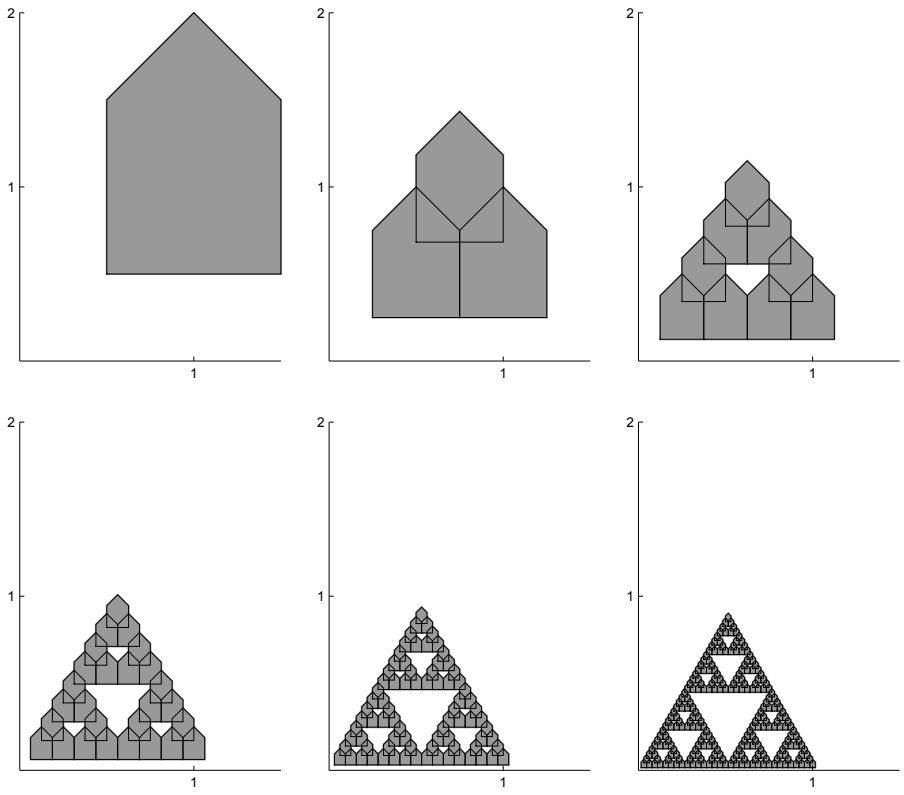

Figure 3.4.2 Approximating the Sierpinski gasket with an arbitrary set

The point of this demonstration bears repeating: the iterated function system determines the self-similar set and the initial approximation has no effect on the final outcome.

3.5 Similarity transformations

A solid understanding of similarity transformations is quite important, if you want to explore self-similar sets and generate your own examples. Thus, in this section, we're going to take a closer look at similarity transformations in both mathematical notation and code.

3.5.1 The basic types of similarity transformations

There are four basic types of similarity transformations, namely:

- Rotation through the angle θ about the point (x_0, y_0), which can be represented
  - mathematically by $R_\theta(x_0, y_0)$ and
  - in code by `rotate(t,[x0,y0])`.
- Scaling by the factor r about the point (x_0, y_0), which can be represented
  - mathematically by $S_r(x_0, y_0)$ and
  - in code by `scale(r,[x0,y0])`.
- Translation by the vector $\langle a, b\rangle$ which can be represented

    - mathematically by $T_{\langle a,b\rangle}$ and
    - in code by `shift([a,b])`.

- Reflection about a vector perpendicular to $\langle a, b\rangle$ containing the point (x_0, y_0), which can be represented
    - mathematically by $F_{\langle a,b\rangle}(x_0, y_0)$ and
    - in code by `reflect([a,b],[x0,y0])`.

These are "basic" in the sense that

- It's easy to see that each type is a similarity,
- Any similarity can be expressed as a composition of these types, and

In this subsection, we'll take a look at these four types of similarity transformations. In particular, we'll examine the effect of these transformations on the set show in Figure 3.5.1. For each of these, we'll also examine a Javascript code snippet that might represent the transformation.

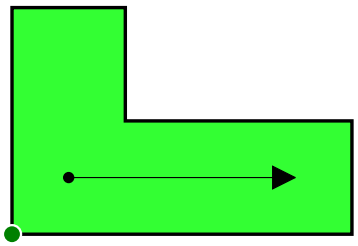

Figure 3.5.1 A set in the plane

3.5.1.1 Rotation

Rotation rotates a set about a point.

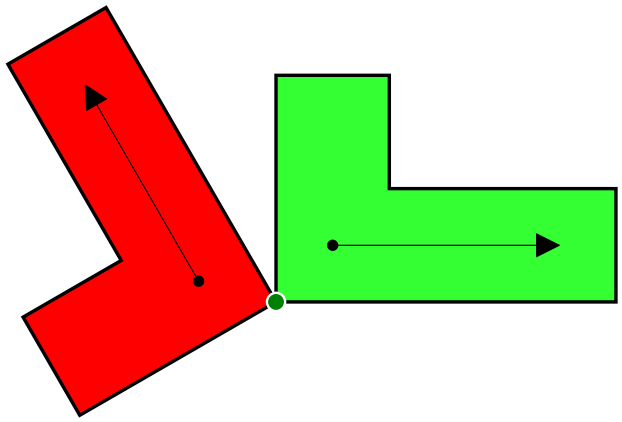

Figure 3.5.2 Rotation of a set in the plane

Mathematically, the specific rotation above is $R_{120^\circ}(0,0)$ and in code it's

```
rotate(120*degree,[0,0])
```

Note that the `[0,0]` is optional, since the rotation will be about the origin by default.

3.5.1.2 Scaling

Scaling scales a set about a point.

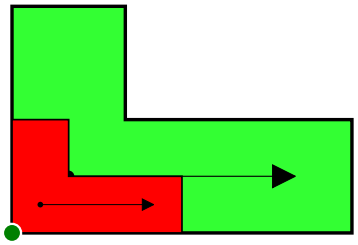

Figure 3.5.3 Scaling of a set in the plane

Mathematically, the specific scaling above is $S_{1/2}(0,0)$ and in code it's

```
scale(0.5,[0,0])
```

Note that the `[0,0]` is optional, since the scaling will be about the origin by default.

3.5.1.3 Translation

Translation translates a set by a vector.

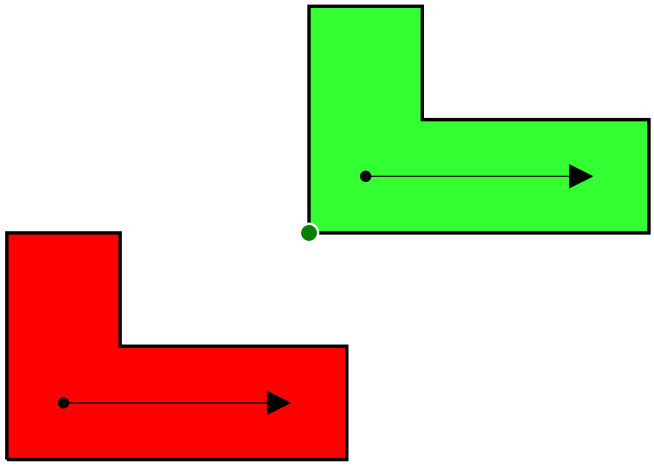

Figure 3.5.4 Translation of a set in the plane

Mathematically, the specific translation above is $T_{\langle -4,-3\rangle}$ and in code it's `shift([-4,-3])`

3.5.1.4 Reflection

Reflection reflects a set across a line.

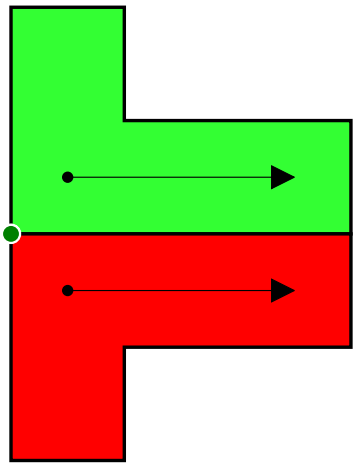

Figure 3.5.5 Reflection of a set in the plane

Mathematically, the specific reflection above is $F_{\langle 0,1\rangle}(0,0)$ and in code it's

`reflect([0,1],[0,0])`

Note that the [0,0] is optional, since the point is assumed to be the origin by default.

3.5.2 Composition

Again, we can express *any* similarity as a composition of functions chosen from the four basic types of similarity transformations. Consider, as an example, Figure 3.5.6. The larger green set can be reflected, scaled, rotated, and shifted to land on the smaller red set. Hit the "Start" button to begin a step-by-step illustration of that process.

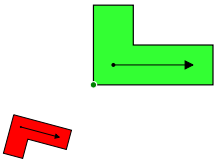

Figure 3.5.6 A more complicated similarity

In order, the sequence of basic transformations that lead to the final final transformation is:

1. The reflection $F_{\langle 0,1\rangle}(0,0)$ followed by
2. the scaling $S_{1/2}(0,0)$ followed by
3. the rotation $R_{-15^\circ}(0,0)$ followed by
4. the translation $T_{\langle -8,-3\rangle}$.

The final similarity transformation can therefore be expressed as the composition

$$T_{\langle -8,-3\rangle} \circ R_{-15^\circ}(0,0) \circ S_{1/2}(0,0) \circ F_{\langle 0,1\rangle}(0,0).$$

Note that the application of the functions happens from the inside out, i.e. from left to right.

3.5.3 `AffineFunction` object

As already metioned, the four types of similarity transformations can be represented on a computer:

- `rotate(t,[x0,y0])`
- `scale(r,[x0,y0])`
- `shift([a,b])`
- `reflect([a,b],[x0,y0])`

These representations refer to concrete Javascript objects that instances of an `AffineFunction` class. As objects, they know certain things about themselves. In computer parlance, we would say they have certain properties and methods, including:

- `is_similarity`: a Boolean indicating whether the function is a similarity or not. This should be true for all of these functions!

- `is_contractive`: a Boolean indicating whether the function is a contraction or not.
- `norm`: the scaling factor.
- `f`: a function that maps `[x,y]` pairs to other `[x,y]` pairs
- `compose`: a function that accepts another `AffineFunction` and returns the composition.

That last item `compose` is of particular importance, if we want to represent *arbitrary* similarity transformations.

3.6 Tools for generating self-similar sets

Mathematically, an iterated function system is a finite collection of contractions. Now that we have a solid understanding of `AffineFunction` objects, we can define an `IFS` (on the computer) to be a list of such objects. Just an `AffineFunction` is an object that knows certain things about itself, we might expect an `IFS` to be an object that stores some relevant information. In this section we provide pointers to two Javascript based tools that implement `IteratedFunctionSystems` using exactly this approach.

3.6.1 The IFS Visualizer

The IFS Visualizer provides a simple interface, implemented in a web page, that makes it pretty easy to generate images of self-similar sets. An image of the IFS Visualizer is shown in figure Figure 3.6.1 and you can find the IFS Visualizer itself at this URL: https://www.marksmath.org/visualization/IFS_Visualizer/.

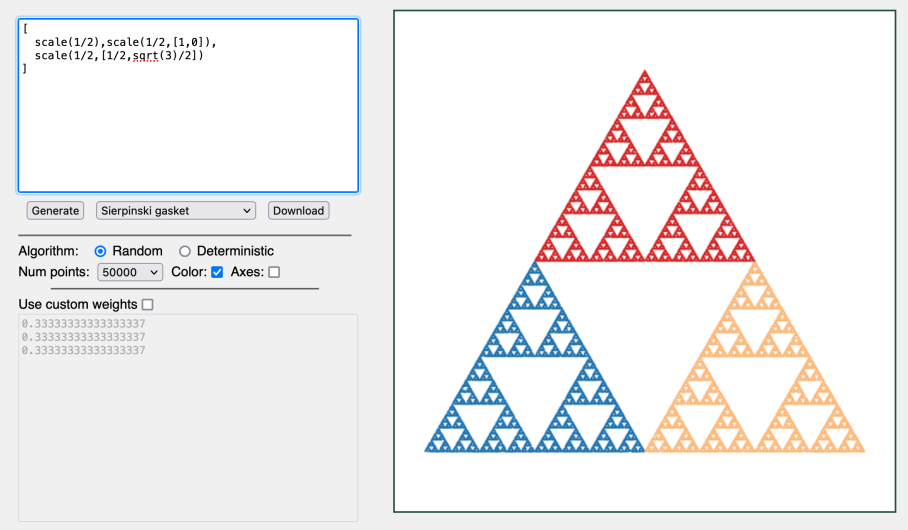

Figure 3.6.1 The IFS Visualizer

To use the IFS Visualizer, simply enter a list of `AffineFunctions` in the box in the upper left and hit the `Generate` button. There is a menu of built in examples to help get you started. The example shown in Figure 3.6.1 is the Sierpinski triangle, whose `IFS` is entered:

```
[
  scale(1/2),scale(1/2,[1,0]),
```

```
    scale(1/2,[1/2,sqrt(3)/2])
  ]
```

There are several other options that affect the way the image is drawn. We'll investigate those further as we discuss algorithms for generating self-similar sets soon.

3.6.2 IFS tools on Observable

As conventient as the IFS Visualizer is, the lack of a full-fledged programming environment does impose some limitations. A much more feature rich programming environment can be found on Observable, which provides an online, notebook based environment for creating visualizations using Javascript. The same `AffineFunction` and `IteratedFunctionSystem` classes that power the IFS Visualizer are implemented there, which makes generating a self-similar set as easy as follows:

```
// Load the packages
import {
  IteratedFunctionSystem,
  scale
} from "@mcmcclur/iteratedfunctionsystem-class"

// And use them
new IteratedFunctionSystem([
  scale(0.5),
  scale(0.5, [1, 0]),
  scale(0.5, [0, 1])
]).render_stochastic({ colors: true })
```

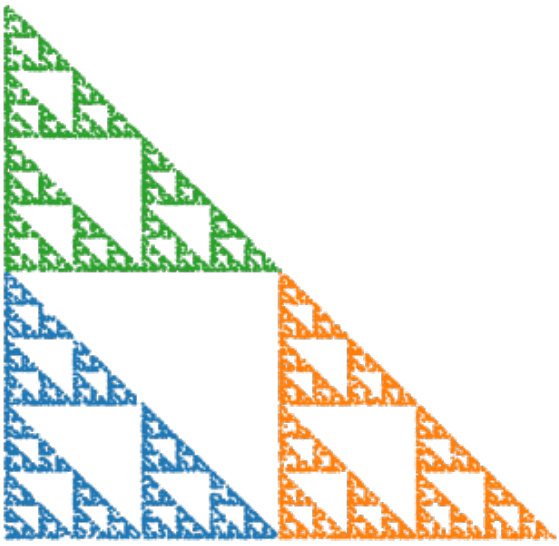

Figure 3.6.2 Simple Sierpinski triangle

There's a whole lot more we can do, since we're working in a fullfledged programming environment. Figure 3.6.3, for example shows a variation on Sierpinski's theme based on a hexagon called the hexagasket.

```
{
  let IFS = [];
  let vertices = [];
  for (let i = 0; i & 6; i++) {
```

```
      let v = [Math.cos((i * Math.PI) / 3), Math.sin((i *
          Math.PI) / 3)];
      vertices.push(v);
      IFS.push(scale(1 / 3, v));
    }
    IFS = new IteratedFunctionSystem(IFS);
    return IFS.render_deterministic({
      max_depth: 4,
      init: vertices,
      fill_colors: true,
      image_width: 800
    });
  }
```

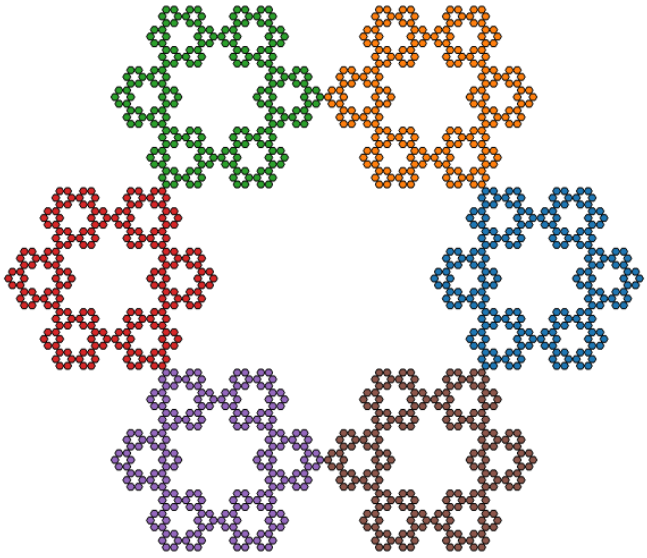

Figure 3.6.3 The hexagasket

With these tools in hand, we'll explore many more examples and discuss the algorithms used to generate them throughout the rest of this chapter. We refer the reader who is immediately curious about aspects of software, to the following Observable notebook: https://observablehq.com/@mcmcclur/iterated-function-systems.

3.7 More examples

In this section, we'll the tools we've developed to generate a few more interesting examples of self-similar sets. As the examples get more complicated, we'll find that we need refinements in our techniques to generate them.

3.7.1 Generalized Sierpinski triangles

Any three, non-collinear points in the plane determine a triangle. If we generate an iterated function system using contractions about those points with the contraction ratio 1/2, we generate a Sierpinski like construction based on the resulting triangle.

We can generate three random points and the corresponding generailized Sierpinski triangle on Observable like so:

```
new IteratedFunctionSystem(
  d3.range(3)
    .map(() => scale(1 / 2, [d3.randomUniform()(),
        d3.randomUniform()()]))
).render_stochastic({ colors: true })
```

The resulting looks a lot like Figure 3.7.1. Full code can be found in this Observable notebook: https://observablehq.com/@mcmcclur/generalized-sierpinski-triangles

Figure 3.7.1 A generalized Sierpinski triangle

3.7.2 The Sierpinski Carpet

The Sierpinski carpet consists of 8 copies itself scaled by the factor 1/3. The IFS can be expressed used the following code:

```
let vertices = [
  [-1, -1], [0, -1], [1, -1], [1, 0],
  [1, 1], [0, 1], [-1, 1], [-1, 0]];
let carpetIFS = new IteratedFunctionSystem(
  vertices.map((pt) => scale(1 / 3, pt)));
```

An approximation to the set is shown in Figure 3.7.2 basd on code here: https://observablehq.com/@mcmcclur/iterated-function-systems#cell-510

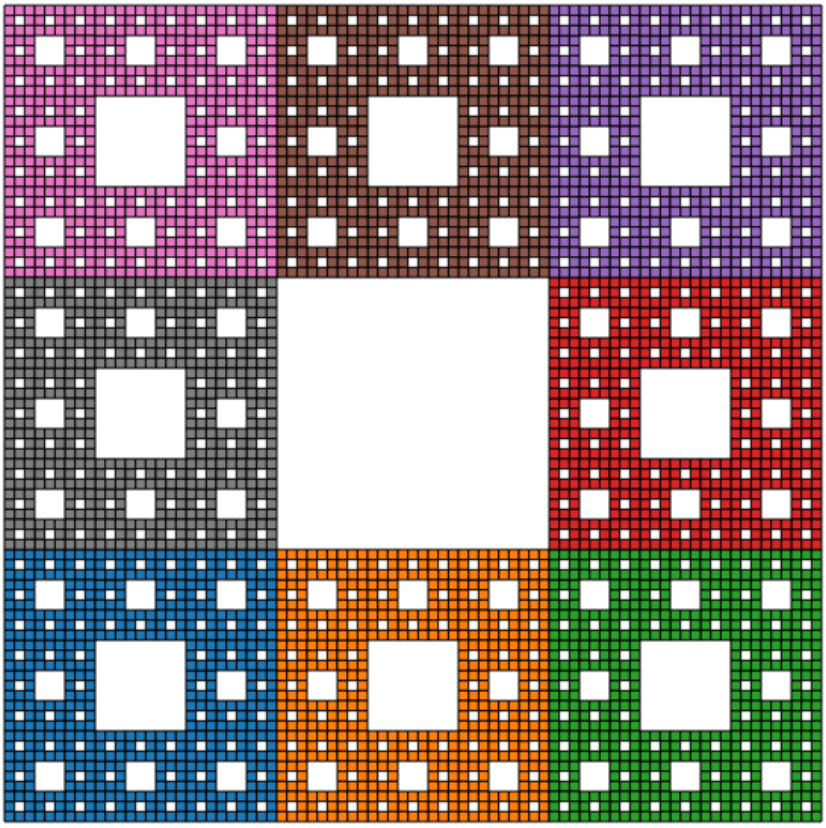

Figure 3.7.2 The Sierpinski carpet

3.7.3 The Koch curve

3.7.4 The Z-curve

3.7.5 The Sierpinski pedal triangle

3.7.6 The Z-curve

3.7.7 The Koch snowflake

3.7.8 The twindragon

Chapter 4

Fractal Dimension

Fractal geometry is concerned with the study of geometrically complicated objects. The concept of "fractal dimension"' is a quantitative measure of this complexity. The Cantor set, for example, is a set between dimensions. On one hand it seems small enough to be of dimension zero, but on the other hand it is a much richer set than what one might think of as a zero dimensional set. Fractal dimension quantifies its place in this spectrum.

There are many notions of fractal dimension - Hausdorff, similarity, box-counting, and packing dimensions are just a few. In the serious study of fractal geometry it is important to understand the relationships between these ideas. In particular, we would like to know conditions guaranteeing equality of two or more definitions. Perhaps one definition is of theoretical importance, while the other is easier to calculate.

We will focus on the similarity dimension and the box-counting dimension. The box-counting dimension is broadly applicable and widely used, but can be difficult to calculate. The similarity dimension is of much more restricted applicability, but easy to calculate when appropriate. Fortunately, it turns out that these concepts are equivalent on suitably chosen sets.

4.1 Quantifying dimension

Both of the definitions of dimension that we'll consider (similarity dimension and box-couting dimension) are generalizations of a very simple idea. We attempt to quantify the dimension of some very simple sets in a way that generalizes to more complicated sets. The simple sets we consider are the unit interval $[0,1]$, the unit square $[0,1]^2$, and the unit cube $[0,1]^3$, which should clearly have dimensions 1, 2, and 3 respectively. Each of these can be decomposed into some number N_r of copies of itself when scaled by certain factors r. The following table shows the values of N_r for various choices of r and for each of these simple objects.

Table 4.1.1 Box counts when decomposing simple objects

	$[0,1]$	$[0,1]^2$	$[0,1]^3$
$r = 1/2$	$N_r = 2 = 2^1$	$N_r = 4 = 2^2$	$N_r = 8 = 2^3$
$r = 1/3$	$N_r = 3 = 3^1$	$N_r = 9 = 3^2$	$N_r = 27 = 3^3$
$r = 1/5$	$N_r = 5 = 5^1$	$N_r = 25 = 3^2$	$N_r = 125 = 5^3$

The decomposition is illustrated for $r = 1/3$ in Figure 4.1.2

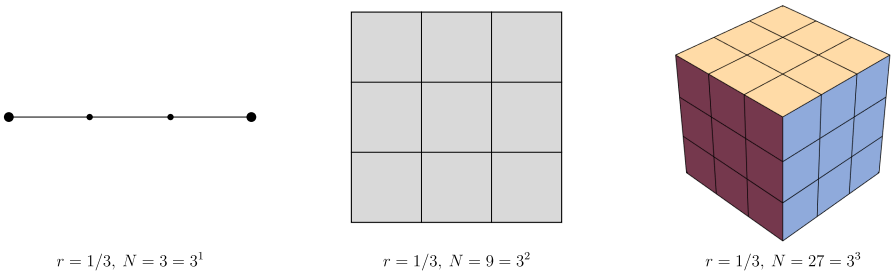

Figure 4.1.2 Simple decompositions for $r = 1/3$

Note that no matter the scaling factor or set, the number of pieces N_r, the scaling factor r, and the dimension d are related by $N_r = (1/r)^d$. This motivates the following definition: If $E \subset \mathbb{R}^n$ can be decomposed into N_r copies of itself scaled by the factor r, then

$$\dim(E) = \frac{\log N_r}{\log 1/r} \tag{4.1.1}$$

Many sets constructed via iterated function systems can be analyzed via equation (4.1.1). For example, the Cantor set is composed of two copies of itself scaled by the factor 1/3. Thus it's fractal dimension is $\log(2)/\log(3)$. The fractal dimension of the Sierpinski gasket is $\log(3)/\log(2)$ and the fractal dimension of the Koch curve is $\log(4)/\log(3)$.

4.2 Similarity dimension of an IFS

Equation (4.1.1) is not quite general enough to compute the fractal dimension of all self-similar sets, since iterated function systems need not have all contraction ratios equal to one another. For example, equation (4.1.1) cannot compute the dimension of the z-curve. There is an important generalization of equation (4.1.1) which defines the dimension associated with any IFS. We will assume that all iterated function systems consist of pure similarities for the remainder of this chapter.

Definition 4.2.1 Similarity dimension. Let $\{f_i\}_{i=1}^m$ be a fixed IFS of similarities and let $\{r_i\}_{i=1}^m$ be the list of associated similarity ratios. Define a function $\Phi : [0, \infty) \to \mathbb{R}$ by

$$\Phi(s) = r_1^s + \cdots + r_m^s.$$

Note that Φ is continuous, strictly decreasing, $\Phi(0) = m$, and $\lim_{s\to\infty} \Phi(s) = 0$. Thus there is a unique positive number s such that $\Phi(s) = 1$. This unique value of s is defined to be the similarity dimension of the IFS. ◊

We can see that this definition agrees with that given by equation (4.1.1), when applicable. If $r_i = r$ for each $i = 1, \ldots, m$, then the similarity dimension is the unique s such that

$$\sum_{i=1}^{m} r^s = mr^s = 1,$$

which has solution $\frac{\log m}{\log 1/r}$.

Example 4.2.2 Suppose that r_1 and r_2 are positive numbers satisfying $r_1 + r_2 \leq 1$. We define an iterated function system $\{f_1, f_2\}$ on $\mathbb{R}$ by setting $f_1(x) =$

$r_1 x$ and $f_2(x) = r_2 x + (1 - r_2)$. Note that f_1 contracts the unit interval by the factor $r_1$towards 0, while f_2 contracts the unit interval by the factor r_2 towards 1. If $r_1 + r_2 = 1$, then the invariant set is the unit interval and the dimension is 1. If $r_1 + r_2 < 1$, then this IFS generalizes the Cantor construction. The dimension of this IFS is the unique solution to the equation $r_1^s + r_2^s = 1$. For example if $r_1 = 1/2$ and $r_2 = 1/4$, we obtain

$$\frac{1}{2^s} + \frac{1}{4^s} = 1 \text{or} s = \frac{\log \frac{1+\sqrt{5}}{2}}{\log 2}.$$

This set is shown in Figure 4.2.3. □

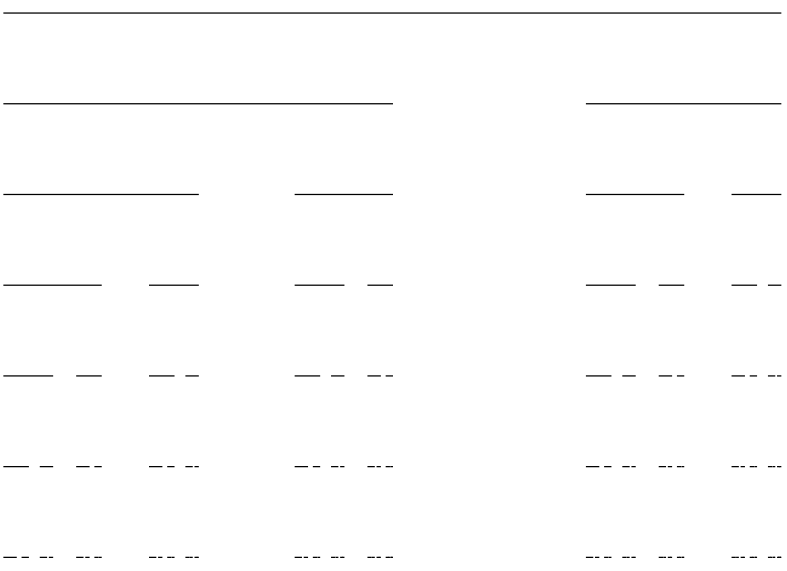

Figure 4.2.3 A two-scale Cantor set

Not all equations of this form can be solved explicitly. For example, if $r_1 = 1/2$ and $r_2 = 1/3$ then the similarity dimension is the unique solution to $1/2^s + 1/3^s = 1$. While this equation cannot be explicitly solved, it does uniquely characterize the dimension. Furthermore, numerical algorithms can be used to approximate the dimension to a high degree of accuracy. In this case, the dimension can be estimated by the following command.

Another example is provided by the z-curve, which has similarity ratio list $\{1/3, \sqrt{2}/6, 1/3, \sqrt{2}/6, 1/3\}$. It's dimension is given by the equation

$$\frac{3}{3^s} + 2\left(\frac{\sqrt{2}}{6}\right)^s = 1.$$

The solution is approximately 1.32038.

If $r_1 + r_2 > 1$, then the solution to $r_1^s + r_2^s = 1$ will satisfy $s > 1$. This indicates a potential problem with similarity dimension since we don't want to assign number larger than one to be the dimension of a subset of $\mathbb{R}$.

4.3 Box-counting dimension

As mentioned earlier, there are many definitions of fractal dimension, all with their own strengths and weaknesses. A major strength of the similarity dimen-

sion is that it is very easily computed. A major weakness of the similarity dimension is that it is very restrictive; there are many sets (even very simple sets) which are simply not self-similar. Another weakness is that many people find it to be non-intuitive when compared to the simple definition of equation (4.1.1).

The box-counting dimension has properties that are somewhat complementary to those of the similarity dimension. In particular, it is much more broadly applicable, although harder to compute. In addition, we can show that it yields the right value for most simple sets and it is a more intuitively direct generalization of equation (4.1.1).

4.3.1 Definition of box-counting dimension

Throughout this section, E will denote a non-empty, closed, bounded subset of $\mathbb{R}^n$. We wish to associate a notion of dimension with sets at that generality. To do so, we need to think about ways to decompose E into smaller sets, that aren't necessarily similar to the whole.

Definition 4.3.1 Box covering. For $\varepsilon > 0$, the ε-mesh for $\mathbb{R}^n$ is the grid of closed cubes of side length ε with one corner at the origin and sides parallel to the coordinate axes. For $n = 2$, this can be visualized as fine graph paper and we might call an ε-mesh cube a square, in that situation. Define $N_\varepsilon(E)$ to be the smallest number of ε-mesh cubes whose union contains E. ◊

A covering of a smooth curve in the plane by ε-mesh squares for two values of ε is shown in Figure 4.3.2.

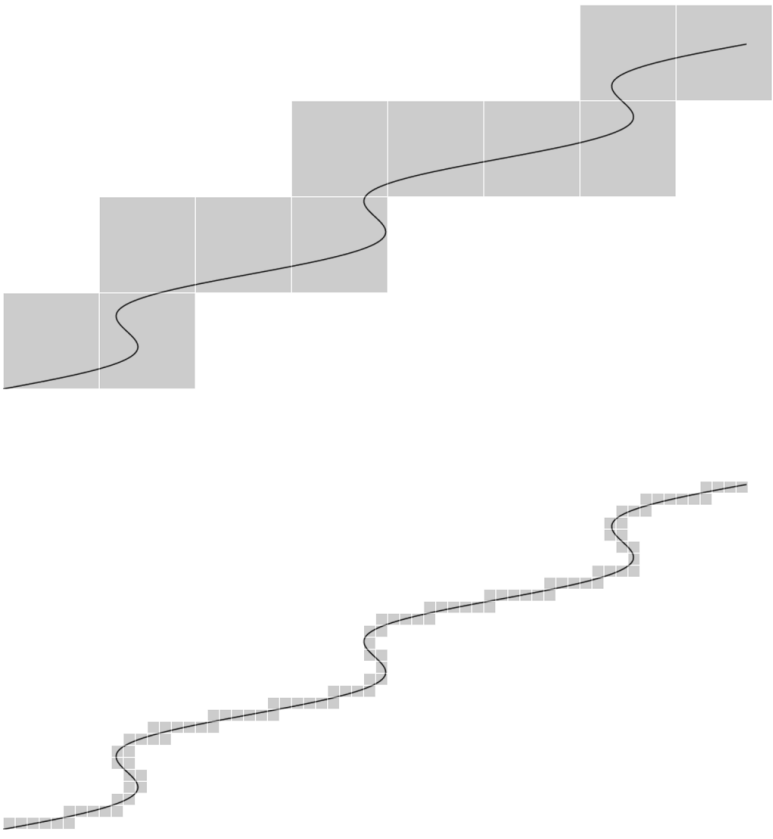

Figure 4.3.2 Coarse and fine box coverings of a smooth curve

Now, we might think of $N_\varepsilon(E)$ as analogous to N_r in equation (4.1.1). That leads us to the expression

$$\frac{\log N_\varepsilon(E)}{\log 1/\varepsilon}.$$

Definition 4.3.3 Box-counting dimension. The box-counting dimension $\dim(E)$ of a non-empty, bounded subset E of $\mathbb{R}^n$ is defined by

$$\dim(E) = \lim_{\varepsilon \to 0^+} \frac{\log N_\varepsilon(E)}{\log 1/\varepsilon},$$

provided this limit exists. ◊

In fact, we can argue that this should give us the correct dimension of 1 whenever E is a smooth curve with length L as follows: A glance at Figure 4.3.2 indicates that, for small ε, we expect that

$$N_\varepsilon(E)\varepsilon \approx L.$$

Taking the logarithm of both sides, we find that

$$\log(N_\varepsilon(E)) + \log(\varepsilon) \approx \log(L)$$

so that

$$\frac{\log(N_\varepsilon(E))}{\log(\varepsilon)} = \frac{\log(L) - \log(\varepsilon)}{\log(\varepsilon)} \to -1$$

as $\varepsilon \searrow 0$.

More generally, if

$$N_\varepsilon(E)\varepsilon^d \approx M,$$

where M is some measure of size dependent on dimension d (for example, M would be area when $d = 2$), then

$$\frac{\log(N_\varepsilon(E))}{\log(\varepsilon)} = \frac{\log(L) - d\log(\varepsilon)}{\log(\varepsilon)} \to -d$$

as $\varepsilon \searrow 0$.

4.3.2 Practical computation

While the box-couting dimension is certainly of theoretical importance, it is also of great interest in the natural sciences because it can be estimated fairly easily in a wide variety of situations. We discuss the practical aspects of that computation here.

Before beginning, we should consider that our input is likely to be some sort of raster image approximating some real world object. To be concrete, let's address Mandelbrot's famous question How long is the coast of Britain? This coast, along with a box covering of 241 squares is shown in Figure 4.3.4.

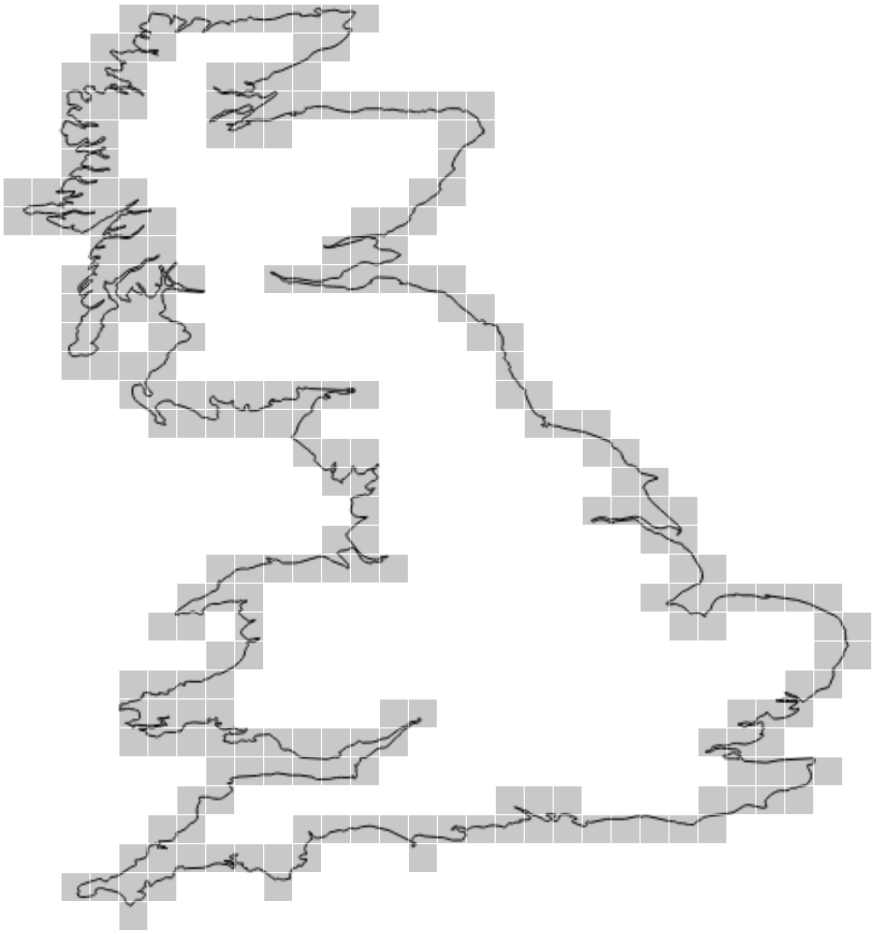

Figure 4.3.4 Box count for the coast of Britain

Given this type of input, working straight from the definition to compute $\log(241)/\log(\text{size})$ is not likely to be fruitful. For one thing, what's size? It doesn't seem like it should matter since dimension should certainly be independent of scale but it certainly would impact our specific estimate. Another issue is that we certainly *can't* compute a limit as $\varepsilon \searrow 0$ directly, as we have only a finite model. The data for Figure 4.3.4 consists of 3707 lat/lon pairs obtained from Natural Earth.

Another imporant issue is *regularity*. While self-similar sets can be very convoluted, they also tend to display a great deal regularity. From the standpoint of box-counting dimension, we might expect the relationsip

$$N_\varepsilon(E)\varepsilon^d \approx M,$$

to hold over a broad range of values.

Here's an approach that accounts for all of these issues. We rewrite the dimension relation in the last equation as

$$\log N_\varepsilon(E) \approx \log M - d \log \varepsilon$$

If we suppose that this relationship is good over a wide range of values, then we see that this is the equation of a line indicating the functional dependence of $\log N_\varepsilon(E)$ on $\log \varepsilon$. Furthermore, the dimension arises as the negative slope of that line. Thus, we might use the computer to generate some box-count data

of N_ε as a function of ε, create a loglog plot of that data, and then compute a regression line for the plot. The negative slope should tell us the dimension.

One more question that arises is - how do we best determine values of ε for our samples? We'd like to be efficient, yet still be sure that we capture enough data to accurately compute the dimension. While we don't quite have the tools yet, we'll show a bit later in Lemma 4.3.14 that we can choose a sequence of εs that converge down to zero exponentially. An easy choice is therefore $\varepsilon_n = 2^{-n}$.

Example 4.3.5 Dimension of the British coastline. You can find an implementation of the box-counting algorithm as described here in this Observable notebook; the British coasline example is included there, among several others. The data for that example is shown in Table 4.3.6. Using that data, we can generate Figure 4.3.7 and compute the regression line to be

$$\log N_\varepsilon \approx 8.707 - 1.289 \log \varepsilon.$$

Table 4.3.6 Box counts for the British coastline

r	1	2	4	8	16	32	64	128	256	512
N_r	3294	2099	1119	542	241	103	39	14	4	1

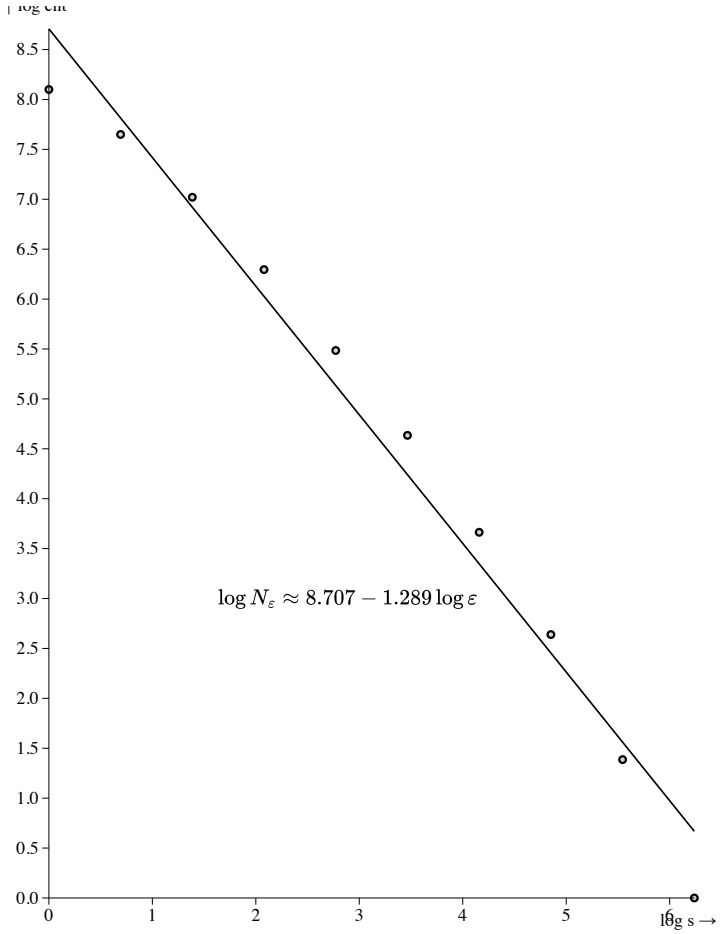

Figure 4.3.7 Regression line for the British coastline

□

4.3.3 Alternative characterizations of box-counting dimension

There are alternative ways to break a more or less arbitrary set into smaller parts that lead to alternative characterization of the resulting dimension. After the limit, the numerical values are the same but the alternative characterizations lead to proofs of nice theoretical properties.

Definition 4.3.8 Diameter. The diameter of a non-empty, closed, bounded set $E \subset \mathbb{R}^n$ is simply the maximum possible distance between any two elements of E, i.e.

$$\operatorname{diam}(E) = \max_{x,y \in E} |x - y|.$$

◊

Definition 4.3.9 **ε-cover.** Given a non-empty, closed, bounded set $E \subset \mathbb{R}^2$ and an $\varepsilon > 0$, an ε-cover of E is a collection of closed, bounded sets of diameter at most ε whose union contains E. We let $C_\varepsilon(E)$ denote the smallest possible number of elements in an ε-cover of E. ◊

Definition 4.3.10 **ε-packing.** Given $\varepsilon > 0$, an ε-packing of E is a collection of closed, disjoint balls of radius ε with centers in E. We let $P_\varepsilon(E)$ denote the maximum possible number of balls in an ε-packing of E. ◊

These geometric ideas are illustrated in Figure 4.3.11. In Figure 4.3.11(a), we see a finite collection of points. Figure 4.3.11(b) illustrates a covering of those points, Figure 4.3.11(c) illustrates a packing of the points, and Figure 4.3.11(d) illustrates a box-covering of the points.

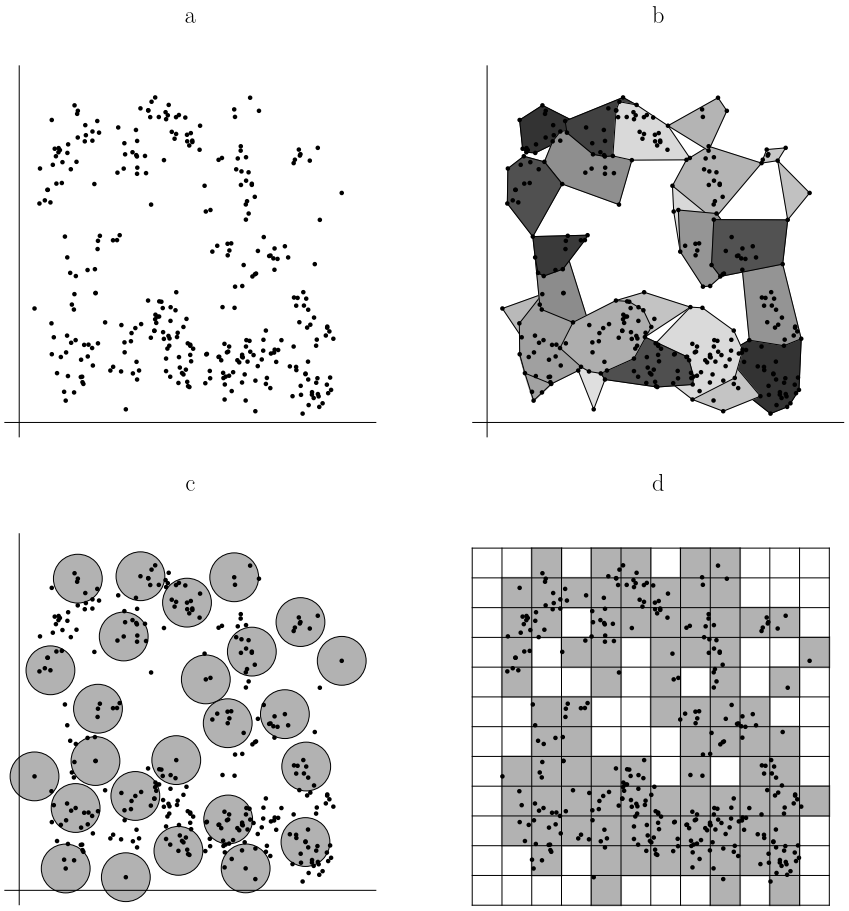

Figure 4.3.11 A covering, packing, and box covering of a finite set.

We could use any of Definition 4.3.1, Definition 4.3.9, or Definition 4.3.10 to define a notion of dimension. Thus, for the time being, let us write

$$\dim_b(E) = \lim_{\varepsilon \to 0^+} \frac{\log N_\varepsilon(E)}{\log 1/\varepsilon},$$

$$\dim_c(E) = \lim_{\varepsilon \to 0^+} \frac{\log C_\varepsilon(E)}{\log 1/\varepsilon}, \text{ and}$$

$$\dim_p(E) = \lim_{\varepsilon \to 0^+} \frac{\log P_\varepsilon(E)}{\log 1/\varepsilon}.$$

We'll call these the box-dimension, covering-dimension, and the packing-dimension repsectively.

As it turns out, these definitions are all equivalent. The significance of this fact is that it gives us a lot of flexibility in how we compute dimension. For many sets, $N_\varepsilon(E)$ is the easiest definition to work with. In theoretical situations, $C_\varepsilon(E)$ or $P_\varepsilon(E)$ might be more natural.

Lemma 4.3.12 *IF E is a non-empty, bounded subset of $\mathbb{R}^n$, then* $\dim_b(E) = \dim_c(E) = \dim_p(E)$.

Proof. Assume first that the packing dimension is well defined. In other words

$$\lim_{\varepsilon \to 0^+} \frac{\log P_\varepsilon(E)}{\log 1/\varepsilon}$$

exists. We will show that the covering dimension is also well defined and that $\dim_c(E) = \dim_p(E)$. Suppose we have an ε-packing of E that is maximal in the sense that no more closed ε-balls centered in E can be added without intersecting one of the balls in the packing. Then any point of E is within a distance at most 2ε from some center in the packing; otherwise we could add another ε-ball to the packing. Thus this ε-packing induces a 4ε-cover of E obtained by doubling the radius of any ball in the packing as illustrated in figure Figure 4.3.13. In symbols, $C_{4\varepsilon}(E) \leq P_\varepsilon(E)$. It follows that

$$\frac{\log C_{4\varepsilon}(E)}{\log \frac{1}{4\varepsilon}} \leq \frac{\log P_\varepsilon(E)}{\log \frac{1}{4\varepsilon}} = \frac{\log P_\varepsilon(E)}{\log \frac{1}{4} + \log \frac{1}{\varepsilon}}.$$

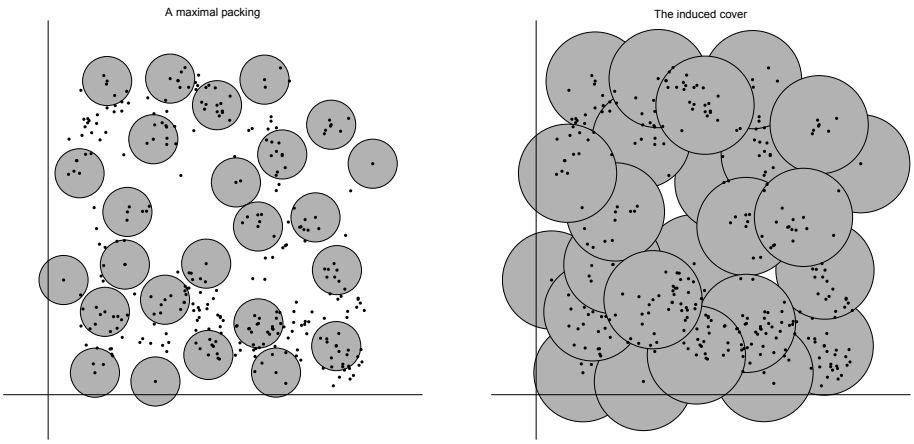

Figure 4.3.13 A maximal packing and the induced cover.

Next, the centers of the balls of any ε-packing of E are separated by more than 2ε. Thus for any ε-cover of E, different sets are needed for each center of any ε-packing. It follows that $P_\varepsilon(E) \leq C_\varepsilon(E)$ for every ε. Thus we now have

$$\frac{\log P_{4\varepsilon}(E)}{\log \frac{1}{4\varepsilon}} \leq \frac{\log C_{4\varepsilon}(E)}{\log \frac{1}{4\varepsilon}} \leq \frac{\log P_\varepsilon(E)}{\log \frac{1}{4} + \log \frac{1}{\varepsilon}}.$$

The expressions on the left and the right both approach $\dim_p(E)$ as $\varepsilon \to 0^+$. Thus, by the squeeze theorem, the expression in the middle approaches this same value as $\varepsilon \to 0^+$. Of course, this middle limit defines the covering dimension. Thus, the covering dimension exists and $\dim_c(E) = \dim_p(E)$.

If we assume that the covering dimension exists, then a similar argument shows that the packing dimension exists using the inequality

$$C_{4\varepsilon}(E) \leq P_\varepsilon(E) \leq C_\varepsilon(E).$$

One can also show that $\dim_c(E) = \dim_b(E)$ using the inequality

$$C_{\sqrt{n}\varepsilon}(E) \leq N_\varepsilon(E) \leq 2^n C_\varepsilon(E).$$

The details are left as an exercise. ■

In light of Lemma 4.3.12, we will drop the subscripts and refer to the dimension computed using any one of $N_\varepsilon(E)$, $C_\varepsilon(E)$, or $P_\varepsilon(E)$ as the box-counting dimension.

Before looking at an example, we prove another lemma, which simplifies computation considerably.

Lemma 4.3.14 *Let $(\varepsilon_k)_k$ be a sequence which strictly decreases to zero and for every k satisfies $\varepsilon_{k+1} \geq c\varepsilon_k$, where $c \in (0,1)$ is fixed, and suppose that*

$$\lim_{k\to\infty} \frac{\log C_{\varepsilon_k}(E)}{\log 1/\varepsilon_k} = d.$$

Then $\dim(E)$ is well defined and $\dim(E) = d$.

Proof. Given any $\varepsilon > 0$, there is a unique value of k such that $\varepsilon_{k+1} \leq \varepsilon < \varepsilon_k$. Assuming this relationship between ε and k, we have

$$\frac{\log C_\varepsilon(E)}{\log 1/\varepsilon} \leq \frac{\log C_{\varepsilon_{k+1}}(E)}{\log 1/\varepsilon_k} = \frac{\log C_{\varepsilon_{k+1}}(E)}{\log \frac{1}{\varepsilon_{k+1}} + \log \frac{\varepsilon_{k+1}}{\varepsilon_k}} \leq \frac{\log C_{\varepsilon_{k+1}}(E)}{\log \frac{1}{\varepsilon_{k+1}} + \log c}$$

$$\frac{\log C_\varepsilon(E)}{\log 1/\varepsilon} \geq \frac{\log C_{\varepsilon_k}(E)}{\log 1/\varepsilon_{k+1}} = \frac{\log C_{\varepsilon_k}(E)}{\log \frac{1}{\varepsilon_k} + \log \frac{\varepsilon_k}{\varepsilon_{k+1}}} \geq \frac{\log C_{\varepsilon_k}(E)}{\log \frac{1}{\varepsilon_k} + \log \frac{1}{c}}.$$

Taking these two inequalities together we have

$$\frac{\log C_{\varepsilon_k}(E)}{\log \frac{1}{\varepsilon_k} + \log \frac{1}{c}} \leq \frac{\log C_\varepsilon(E)}{\log 1/\varepsilon} \leq \frac{\log C_{\varepsilon_{k+1}}(E)}{\log \frac{1}{\varepsilon_{k+1}} + \log c}.$$

Now as $k \to \infty$, the expressions on either side of the inequality both approach d. Furthermore, $\varepsilon \to 0^+$ as $k \to \infty$. Thus, by the squeeze theorem,

$$\lim_{\varepsilon\to 0^+} \frac{\log C_\varepsilon(E)}{\log 1/\varepsilon} = d$$

and this value is $\dim(E)$ by definition. ■

Of particular interest are the geometric sequences $\varepsilon_k = c^k$ where $c \in (0,1)$. Note that similar results hold for $N_\varepsilon(E)$ and $P_\varepsilon(E)$.

Lemma 4.3.14 makes it easy to compute the box-counting dimension of the Cantor set. If E is the Cantor set, then $N_{3^{-k}}(E) = 2^k$. Thus,

$$\dim(E) = \lim_{k\to\infty} \frac{\log 2^k}{\log 3^k} = \frac{\log 2}{\log 3}.$$

It has turned out that the box-counting dimension of the Cantor set is the same as its similarity dimension. As we will see in the next section, this is not by chance. This example should not leave the impression that box-counting dimension is easy to compute. An attempt to compute the box-counting dimension of the generalized Cantor sets should make this clear.

4.4 Comparing fractal dimensions

We have now defined two notions of dimension. Box-counting dimension is broadly defined and well motivated. Similarity dimension is much more restrictive, but very easy to compute when applicable. The main goal in this section is to show that these definitions agree under a reasonable assumption. The advantage of this is two-fold. Similarity dimension can be used to compute the (usually more difficult) box-counting dimension of a self-similar set. Furthermore, we can assume that similarity dimension is a reasonable definition of dimension as it agrees with the box-counting dimension.

We first illustrate the need for an additional assumption concerning the similarity dimension Recall the definition of the generalized Cantor set, where $f_1(x) = r_1 x$ and $f_2(x) = r_2 x + (1 - r_2)$ define an IFS on $\mathbb{R}$. If $r_1 = r_2 = 3/4$, then the similarity dimension of the IFS is $\log(2)/\log(4/3) \approx 2.4 > 1$. This is clearly nonsense since the invariant set is precisely the unit interval which has dimension 1. The problem is that the two images of $[0, 1]$ under the action of the IFS have a significant amount of overlap, thus the IFS does not generate efficient covers of the invariant set. There is an assumption we can place on an IFS which limits this type of overlap.

Definition 4.4.1 Strong open set condition. An IFS $\{f_i\}_{i=1}^m$ on $\mathbb{R}^n$ with invariant set E is said to satisfy the *strong open set condition* if there is an open set U in $\mathbb{R}^n$ such that $U \cap E \neq \emptyset$ and

$$U \supset \bigcup_{i=1}^{m} f_i(U)$$

with this union disjoint. $\Diamond$

Note that if a generalized Cantor set satisfies $r_1 + r_2 \leq 1$, then the strong open set condition is satisfied by taking U to be the open unit interval. If $r_1 + r_2 > 1$, then the strong open set condition is no longer satisfied.

Theorem 4.4.2 *Let E be the invariant set of an IFS with similarity dimension s. If the IFS satisfies the strong open set condition, then* $\dim(E) = s$.

Proof. To prove the theorem we need to develop some useful notation associated with an IFS $\{f_i\}_{i=1}^m$ with ratio list $\{r_i\}_{i=1}^m$, invariant set E, and similarity dimension s. A string with symbols chosen from $\{1, \ldots, m\}$ is simply a finite or infinite sequence with values in $\{1, \ldots, m\}$. A finite string α will be denoted $\alpha = i_1 \cdots i_k$. Finite strings can be concatenated to form longer strings. Thus if $\alpha = i_1 \cdots i_k$ and $\beta = j_1 \cdots j_l$, then $\alpha\beta = i_1 \cdots i_k j_1 \cdots j_l$. Every string $\alpha = i_1 \cdots i_k$ has one parent $\alpha^- = i_1 \cdots i_{k-1}$. Given a positive integer k, let J_k denote the set of all strings of length k with values chosen from the set $\{1, \ldots, m\}$. Let $J_* = \bigcup_{k=1}^{\infty} J_k$ denote the set of all such finite strings. Given $\alpha = i_1 \cdots i_k \in J_k$, let $f_\alpha = f_{i_1} \circ \cdots \circ f_{i_k}$and $r_\alpha = r_{i_1} \cdots \mathrm{r}_{i_k}$. Then J_k induces a k^{th} level decomposition of E given by

$$E = \bigcup_{\alpha \in J_k} f_\alpha(E).$$

The sets J_k are examples of *cross-sections* of J_*, but J_k does not necessarily induce a useful decomposition of J_* with respect to box-counting dimension as the sizes of the sets $f_\alpha(E)$ may vary greatly. A more general type of cross-section may be defined as follows. Let J_ω denote the set of all infinite strings with symbols chosen from $\{1, \ldots, m\}$. Given $\sigma = i_1 i_2 \cdots \in J_\omega$, let $\omega|_k$ denote the element of J_k whose symbols are the first k symbols of σ. Given $\alpha \in J_k$,

let $[\alpha] = \{\sigma \in J_\omega : \sigma|_k = \alpha\}$ denote the set of all infinite strings whose first k symbols agree with α. A cross-section of J_* is a finite set $J \subset J_*$ such that $\cup_{\alpha\in J}[\alpha] = J_*$ with this union disjoint. Note that a cross-section J of J_* defines a decomposition of E given by

$$E = \bigcup_{\alpha\in J} f_\alpha(E).$$

Furthermore, if $\alpha \in J_*$, then

$$\sum_{j=1}^{m} r_{\alpha j}^s = r_\alpha^s \sum_{j=1}^{m} r_j^s = r_\alpha^s.$$

It follows by induction that $\sum_{\alpha\in J} r_\alpha^s = 1$ for any cross-section J of J_*.

Now for $\varepsilon > 0$, define J_ε by

$$J_\varepsilon = \{\alpha \in J_* : r_\alpha \mathrm{diam}(E) \le \varepsilon < r_{\alpha^-}\mathrm{diam}(E)\}.$$

Note that if $\sigma \in J_\omega$, then $\left(r_{\sigma|_k}\right)_k$ is a sequence of positive numbers which is strictly decreasing to zero. Thus there is a unique value of k with $\sigma|_k \in J_\varepsilon$ so J_ε defines a cross-section of J_*. Furthermore, the decomposition of E induced by J_ε has diameters comparable to ε. In particular, if $r = \min\{r_i\}_{i=1}^m$, then

$$r\varepsilon < \mathrm{diam}\left(f_\alpha(E)\right) \le \varepsilon$$

for all $\alpha \in J_\varepsilon$.

Finally, $\#(J)$ will denote the number of elements in a cross-section. We are now ready to prove our theorem on the comparison of dimensions.

Our strategy is to find constants M_1 and M_2 so that

$$s + \frac{\log M_1}{\log 1/\varepsilon} \le \frac{\log P_\varepsilon(E)}{\log 1/\varepsilon} \le \frac{\log C_\varepsilon(E)}{\log 1/\varepsilon} \le s + \frac{\log M_2}{\log 1/\varepsilon}, \tag{4.4.1}$$

for then the squeeze theorem applies. Note that the second inequality follows from the fact that $P_\varepsilon(E) \le C_\varepsilon(E)$ which was established in the proof of our lemma comparing packings and coverings.

The last inequality in (4.4.1) is equivalent to $\varepsilon^s C_\varepsilon(E) \le M_2$. We will show that this is true for $M_2 = r^{-s}\mathrm{diam}(E)^s$, where $r = \min\{r_i\}_{i=1}^m$ as above. Recall that $C_\varepsilon(E) \le \#(J_\varepsilon)$ and $r\varepsilon < r_\alpha \mathrm{diam}(E)$. Thus

$$(r\varepsilon)^s C_\varepsilon(E) \le \sum_{\alpha\in J_\varepsilon} \left(r_\alpha \mathrm{diam}(E)\right)^s = \mathrm{diam}(E)^s$$

and $\varepsilon^s C_\varepsilon(E) \le r^{-s}\mathrm{diam}(E)^s$.

Proof of the first inequality of (4.4.1) is somewhat more difficult and will require the use of the strong open set condition. We will show that there is a constant $M_1 > 0$ such that $(r\varepsilon)^s P_{r\varepsilon}(E) \ge M_1$. A simple change of variables shows that this is equivalent to $\varepsilon^s P_\varepsilon(E) \ge M_1$, which is in turn equivalent to the first inequality of equation (4.4.1). Let U be the open set specified by the strong open set condition and let $x \in U \cap E$. Choose $\delta > 0$ such that $\overline{B_\delta(x)} \subset U$. Then any cross-section J of J_* induces a packing $\left\{f_\alpha\left(\overline{B_\delta(x)}\right) : \alpha \in J\right\}$ of E. Now let

$$J_\varepsilon = \{\alpha \in J_* : r_\alpha \delta \le \varepsilon < r_{\alpha^-}\delta\}.$$

Then $r\varepsilon < r_\alpha\delta \le \varepsilon$ for all $\alpha \in J_\varepsilon$. Thus $P_{r\varepsilon}(E) \ge \#(J_\varepsilon)$ and

$$\varepsilon^s P_{r\varepsilon}(E) \geq \sum_{\alpha \in J_\varepsilon} r_\alpha^s \delta^s = \delta^s.$$

Multiplying through by r^s we obtain $(r\varepsilon)^s P_{r\varepsilon}(E) \geq (r\delta)^s \equiv M_1$. ■

-->