The following project will investigate Mandelbrot and Julia sets, which are examples of fractals, geometric structures that has self-similarity at any scale. You will see what how to determine points in the Mandelbrot and Julia sets and then produce images of them.

Project Background

We investigate the Mandelbrot set, a set of number in the complex plane formed by iterating a complex map and determining the set of points that is bounded. Here we give some background on this. Before starting this make sure that you understand basic operations with complex numbers that were given in the course notes (weeks 2 and 3).

Complex Numbers

A few things that are important about complex numbers. Let $z = a + bi$ be a complex number then the number a is the real part and b is the imaginary part. We can think of plotting a point in the complex plane where the horizontal axis is the real part and vertical point is the imaginary axis. To multiply number $x = a + bi$ and $y = c + di$, you get

$$xy = (a + bi)(c + di) = ac + bci + adi + (bi)(di) = ac + (bc + ad)i - bd = (ac - bd)i + (bc + ad)i$$

The magnitude of a complex number is the distance the point is from the origin. It is denoted as $|z|$. If $z = (a+bi)$, then $z = \sqrt{a^2 + b^2}$. See the complex number discussion on purplemath.com for more details about this.

A complex map

A map is a fancy mathematical term for applying a function (usually repeatedly). For example, the repeated iteration of Newton's method can be thought of a map. The iteration

$$x_{n+1} = x_n^2 + c \tag{1}$$

is what we will study in this project where the x and c values are complex. For example, let let $c = i$ and $x_0 = 0$.

$$\begin{aligned}
x_1 &= x_0^2 + c = 0 + i \\
x_2 &= x_1^2 + c = i^2 + i = -1 + i \\
x_3 &= x_2^2 + c = (-1 + i)^2 + i = 1 - 2i - 1 + i = -i \\
x_4 &= x_3^2 + c = (-i)^2 + i = -1 + i
\end{aligned}$$

and since $x_4 = x_2$, then $x_5 = x_3$ and all even will equal $-1 + i$ and all odds will be $-i$.

If we repeat this with $c = 1.01i$

$$\begin{aligned}
x_1 &= x_0^2 + c = 0 + 1.01i = 1.01i \\
x_2 &= (1.01i)^2 + 1.01i = -1.0201 + 1.01i \\
x_3 &= (-1.0201 + 1.01i)^2 + 1.01i = 0.020504010000000017 - 1.0506019999999998i \\
x_4 &= -1.1033441479779194 + 0.96691689217196i \\
&\vdots \\
x_{10} &= -25.934812678057604 + 8.464736394953507i
\end{aligned}$$

Sequences

A sequence is a bunch of numbers in a particular order. We write it as

$$\{a_n\} = a_1, a_2, a_3, \ldots$$

where n is called the index. Typically a sequence does not end. For example, here are a few sequences:

$$1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$$
$$0, 1, 0, -1, 0, 1, 0, -1, \ldots$$
$$\frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \ldots$$
$$1, 2, 4, 8, \ldots$$

You can see that for any given value of c, the map given above, results in a sequence. The example with $c = i$ and $x_0 = 0$ results in the sequence:

$$0, i, -1 + i, -i, -1 + i, -i, -1 + i, \ldots$$

Often, an interesting question for sequences like this is whether or not the sequence converges or not. The limit of a sequence if it exists is the number that the sequence approaches as n (the index) goes to ∞. This notion is the same as that for limits of functions at infinity.

Just looking at the 5 sequences above, the first sequence goes to 0, the second does not converge, the 3rd goes to 1, the 4th goes to ∞ and the last does not converge. The complex one above appears to bounce between the values $-i$ and $-1 + i$ and since it isn't one particular value, this sequence does not converge.

Another possible result of a sequence is that it is bounded. We say that a sequence is **bounded above** if there is a number M such that $a_n < M$ for all n and is **bounded below** if there is a number m such that $a_n > m$ for all n. For a complex sequence, we say the sequence is bounded if there is a M such that $|a_n| < M$ for all n.

Note: the complex sequence above is bounded but does not converge.

The Mandelbrot Set

The mandelbrot set is the set of all numbers, c in the complex plane such that the sequence $\{x_n\}$ generated by the map in (1) is bounded. This is difficult to find, however we will find an approximation.

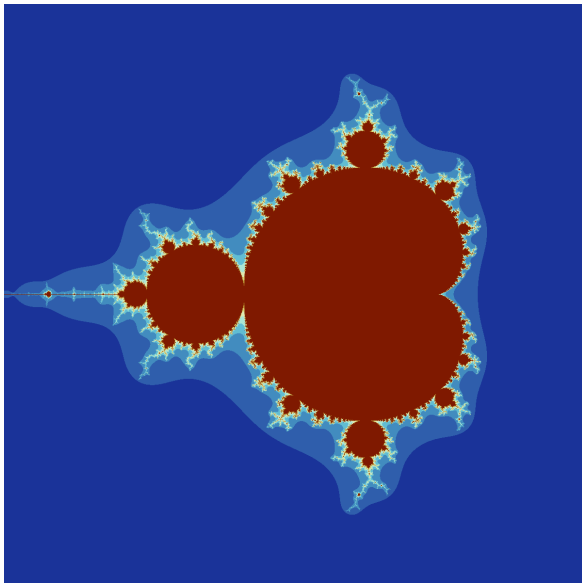
An important result is the following lemma. In short it says that if any point in the sequence is more than 2 units from the origin, then the sequence will not be bounded.

Lemma 1 *If $|x_n| > 2$ for the map in (1), then $|x_{n+1}| > |x_n|$ and as a result $|x_n| \to \infty$*

With this lemma in our pocket, we can determine if the point c is in the Mandelbrot set. The value N is a positive integer that we will use to determine if the sequence is bounded. Below we will use $N = 100$.

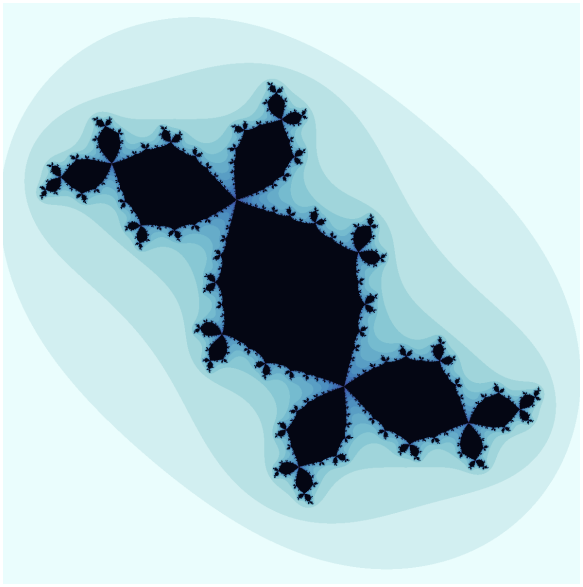
1. Let $z_0 = 0 + 0i$, $n = 0$
2. Apply $z_{n+1} = z_n^2 + c$.
3. If $|z_n| > 2$, stop the point is not in the set
4. If $n > N$, then it is probably bounded. Stop.
5. goto 2.

If we do this for a set of c in the complex plane, then we can get an array of leaving numbers. Then assigning a color to each leaving number results in a plot like:



The Julia Set

There is a related set of an iterated functions called a *Julia Set*. We still use the map $z \to z^2 + c$, however for a plot of the set, the number c is fixed and the initial point z_0 varies. For example, if $c = -0.123 + 0.745i$, then the following is generated:



and this is known as Duoady's Fractal Rabbit.

Stage 1

1. To answer the questions below, you should write a Module to do at least the following:

    - A `FractalView` type. This should contain the min and max of the view for a plot of the Mandelbrot set. These two values should be complex numbers and be the lower left and upper right corners of the plot in the complex plane. Additionally, there should be two additional positive integers which is the size of

the image in pixels. These should default to 800 wide by 600 tall if they are not provided. Store this in a module called `Fractals`.

- An `iterateFunction` function that iterates a general complex function a certain number of times. The arguments should be a function, an initial point and a number of times that it iterates. It should return a vector of complex numbers for each iterated point.
- A `leavingNumber` function that takes a complex number, c as input and returns the number of iterations to leave defined as $|z_n| > 2$. You should use a optional parameter to determine the maximum number of iterations (use 100 as the default) to take and take another optional parameter for the initial point (use `0+0im` as the default)
- An `inMandelbrot` function that takes a complex number as input and returns true or false depending on if the input is in the Mandelbrot set as explained above. Assume that the initial point is `0+0im` and upon iteration of 100 times, if $|z_{100}| < 2$ then return true.
- For each of the types and functions, you should check that the parameters are valid.

2. Write documentation for each of the structs and functions in your module.
3. Write a Unit test (as a separate file) to test that your types/functions are working as expected. (Note: think of the basics like needed parameters are positive, for example). Here are some things to consider:
   - Test the `FractalView` constructor. Ensure that all inputs are valid.
   - Test the `iterate` function with $z \to z^2 + c$ and $z_0 = 0$ and the following values $c = -i, 0.4 + 0i, -0.5 + 0.5i$ and $0.4i$. Test with $n = 5$ points.
   - Test the `leavingNumber` function for $c = -i, -1.01i, 0.4 + 0i, -0.5 + 0.5i$ and $2i$.
   - Test the `inMandelbrot` function for $c = -i, -1.01i, 0.4 + 0i, -0.5 + 0.5i$ and $2i$.

Stage 2

1. Write a Mandlebrot viewer function called `mandelbrotViewer` that takes a `FractalView` object as the only argument. To do this:
   (a) build a 2D complex array with the bounds defined by `min` and `max` as the fields of the `FractalView` object.
   (b) apply the `leavingNumber` function to each element in the array. You should have a 2D array of integers (each representing the leaving number).
   (c) plot a heat map of the array of integers. This should give a plot of the Mandelbrot set.

   Note: the tricky thing is to get the orientation correct. This occurs because matrices are oriented differently than the standard cartesian axes. Do some good testing. It is also helpful to get the aspect ratio correct. This takes more than just the `aspect_ratio` option in the `heatmap` function. The number of horizontal and vertical pixels must match the ratio of the min/max of the complex numbers.

   Additionally, you can change the coloring on the heatmap from the `Plots` package. Look at the documentation for ColorSchemes to see how to change the coloring if you'd like.
2. Make lot of fun plots of the mandelbrot set at different scalings. You can search the interweb for interesting plotting domains. Try many different scales.
3. Write a Julia set viewer function called `juliaViewer` that takes a complex constant c and a `FractalView` object as two arguments. It should be nearly identical to the `mandelbrotViewer` function except that the leaving number should be based on the intial point (input matrix) and the constant c should be the passed in argument.
4. Make a lot of fun plots of the julia sets for various values of c. Usually the interesting values are those that are close to the boundary of the mandelbrot set.