Girls Talk Math
Dynamical Systems
Patterns and Fractals

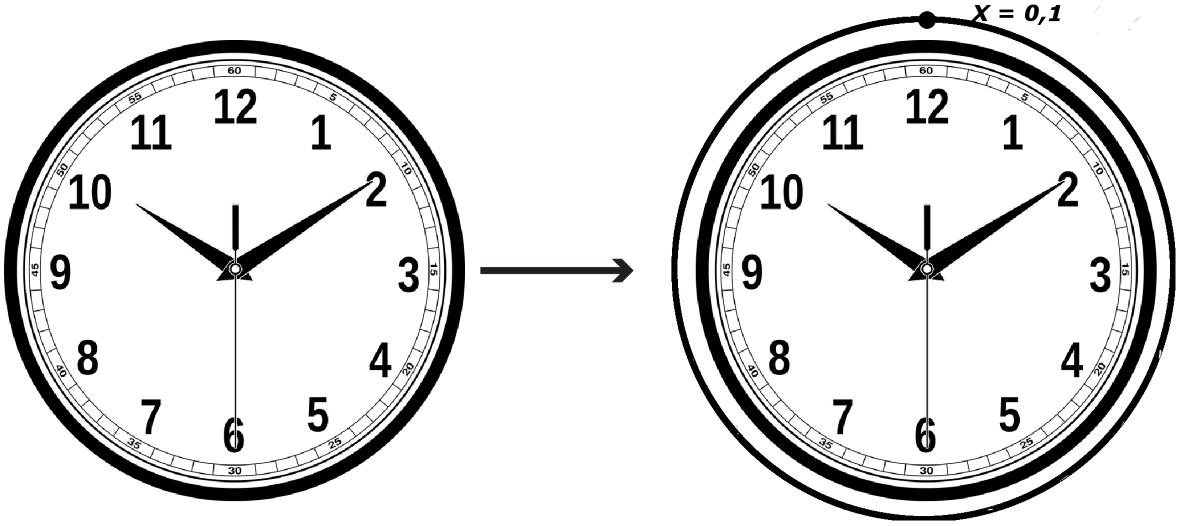
1 Introduction

All around us, we can see patterns form naturally. For example, the weather has some patterns — we know it to be warmer during the day and cooler at night. Other patterns could be the presence of pizza at lunch or the number of students with colds in your classroom over time. However, some things, like weather, are very complex and can evolve into the irregular and sometimes unpredictable behavior called **chaos**. Alternatively, we can also see even more predictable patterns in clocks where the clock ticks through the hours $1, 2, \dots 12$, and then the sequence of numbers start at 1 again (then $2, 3, \dots$ etc.). This organized, regular behavior is called **periodic**. What we would like to do as mathematicians is to assign mathematics to describe the behavior whether periodic or chaotic. **Dynamical Systems** is the mathematical modeling for the changes in something as it moves through time.

Contents

2 Mod Maps and Quadratic Functions Exercises

One way we can represent the clocks feature of restarting numbers using the modulus map. Where do we start our numbering (a.k.a. enumeration) with clocks? Where do they begin and end? Well, since the clock displays 12 hours in a circular manner, once we hit 12, we then go back to 1 rather than moving on to 13. Since the clock has the same display every 12 hours, adding 12 hours to the current time yields the current time (ignoring am/pm). In this sense 12 is the same as 0. If it reads 5 o'clock now, what time will it read in 134 hours? Adding any multiple of 12 won't change the time, so we want to find how many times 12 goes into 134 and calculate the remainder. We see $134 = 12 \cdot 11 + 2$, so in 134 hours the clock will read $5+2 = 7$ o'clock. This is an example of the modulus map in action. Here we are taking the number 134 modulus 12. We can say $134 = 2 \pmod{12}$, which is read as "134 equals 2 modulus 12".



Of particular interest to our studies will be evaluating numbers modulus 1. Think of the circle of the clock being renumbered so that the values start at 0 and as you round back to the initial 0 point you get close to 1. Then as you return to the starting point where you would reach 1, you instead go back to 0. Thus the value at this point on the circle is $1 = 0$ modulus 1 (just like how 12 was in some sense 0 when calculating a number modulus 12). We can represent this function mathematically as the modulus using the equation,

$$M(x) = x \pmod{1}.$$

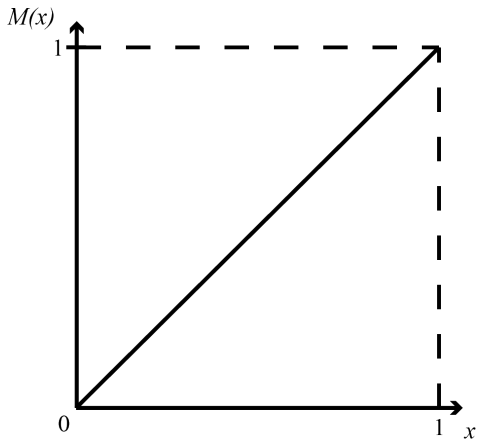


Figure 1: This is a plot of the modulus 1 function.

Check out the examples and complete the following exercises.

Example: For $M(x) = x \pmod 1$,

1. $M(0.3) = 0.3 \pmod 1 = 0.3$,
2. $M(3.2) = 3.2 \pmod 1 = 0.2$,
3. $M(-0.2) = -0.2 \pmod 1 = -0.2 + 1 \pmod 1 = 0.8 \pmod 1 = 0.8$.

Exercise 1: For $M(x) = x \pmod{\mathbf{2}}$,

1. $M(0.1) = 0.1 \pmod 2 =$
2. $M(5.7) =$
3. $M(-2.5) =$

Mathematically, the modulus is defined to be the **remainder**, r, after dividing one number, x, by another, n. Such that $x = nq + r$ and q is the **quotient**.

Python Exercise 0: This packet will involve programming with PYTHON. To learn what you need to about PYTHON or to review the necessary syntax we will use the Jupyter python notebooks:

`https://github.com/girlstalkmath-umd/patterns-and-fractals`

Click on the link 'Python_Basics.ipynb.' You can choose between two methods of running these notebooks.

1. **Run the notebook on Google Colab.**
   This approach is the most straightforward approach. Open the notebook by clicking on the 'Open in Colab' button seen on top of the code. You will need a Google account to run this. If you do not see the 'Open in Colab' button you can also go to `https://colab.research.google.com`, click the 'GitHub' tab and enter the URL above. This will allow you to open the notebook directly in Colab.

2. **Install python on your computer.**
   This approach is more advanced but might be preferred if you intend to do more coding in python. Go to: `https://www.anaconda.com/distribution/` and choose your operating systems (Windows, Mac, Linux). Download the Python 3.7 version and following the installation instructions. You can check that python is installed by typing `python -V` into a terminal window. This will give you the version of python installed. Now download 'Python_Basics.ipynb' and navigate in the terminal to that directory. In the terminal, type `jupyter notebook Python_Basics.ipynb` and press enter. A browser page should pop up with the notebook and you can get started.

Read the text and when you get to a cell (light gray page-width box), hit the play button that appears when you hover your mouse in the upper left corner of the cell. This executes the code in the cell. There are exercises in this tutorial for you to complete by adding code. Once you have completed the tutorial, in a new cell, you can check your answers for the modulus exercises. The command for mod is '`%`' so 0.1 (mod 2) is `0.1%2`. There is also the command '`divmod(x,n)`' where $x \pmod{n}$ is the function we want to evaluate. This example produces the quotient and remainder: `(q,r)`. You can then check `x = n*q+r`.

Example:

```
>>> x = 0.1
>>> n = 2
>>> (q, r) = divmod(x,n)
>>> n*q+r
0.1
>>> n*q+r == x
True
```

The '`==`' is a logic operator. It is used to see if two things are equal and outputs a '`1`' or '`True`' if its true of '`0`' or '`False`' if it's false.

Try out these PYTHON operations for the previous exercises. Compare your results.

2.1 Iterating Forward

In the case of discrete, integer values of time, (i.e. $n = 0, 1, 2, 3 \ldots$). We can model the pattern of our periodic function by the time it takes to go from a current time n to the next time $n + 1$,

$$M(x_n) = x_{n+1}$$

So that if we initialize at a value x_0 and plug $x = x_0$ into $M(x)$ to get $M(x_0)$ then it outputs x_1 which is the value at the next time. Then x_2 value is output from applying $M(x)$ to x_1 value, and so on. In this fashion we can **iterate forward** in time using a function and sometimes produce a pattern. Consider the quadratic function $f(x) = x^2$. Let's initialize it with $x_0 = 2$. Then we can iterate forward,

$$\begin{aligned} f(2) &= 4 \\ f(4) &= 16 \\ f(16) &= 256 \\ &\vdots \end{aligned}$$

The sequence of iterates produce $(2, 4, 16, 256, \ldots)$ — values that continue to increase quickly to infinity with no numbers repeating. In Python Exercise 0, we learned how to use a '`for`' loop. Next, we have an exercise to show how you can iterate forward using code by creating a programing function '`nestList`' that produces our iterative list using the iterative '`for`' loop.

Python Exercise 1: Let's take a moment to study the quadratic function using computation tools. Navigate to our github to find the Jupyter PYTHON notebooks and select 'Iterations.ipynb.' In this notebook, you do not have to change anything to make it run. Follow the instructions, running the cells one-by-one. Play around with initial values to see how the numbers behave when iterating forward.

Look at iterating through the modulus functions, $M(x) = 2x \pmod 1$, starting with initial condition $x_0 = 1/3$ and plugging $x = x_0 = 1/3$ into $M(x)$,

$$\begin{aligned} M\left(\frac{1}{3}\right) &= 2\left(\frac{1}{3}\right) \pmod 1 = \frac{2}{3} \pmod 1 = \frac{2}{3} \\ M\left(\frac{2}{3}\right) &= 2\left(\frac{2}{3}\right) \pmod 1 = \frac{4}{3} \pmod 1 = \frac{1}{3} \\ M\left(\frac{1}{3}\right) &= \frac{2}{3} \\ M\left(\frac{2}{3}\right) &= \frac{1}{3} \\ &\vdots \end{aligned}$$

For that initial condition, this iterate produce the sequence $(1/3, 2/3, 1/3, 2/3, \ldots)$ — values that have a periodic pattern to them. This sequence of numbers is called an orbit, and the fact that they repeat a pattern in time makes them a **periodic orbit**. For this example, a sequence of two numbers, $(1/3, 2/3)$, repeat which makes it a **period-2 orbit**.

In some cases, you can achieve the opposite, and get the past from the present. To to **iterate backwards**, we need to define the **inverse function** to go from a current time $n+1$ to the last time n,

$$x_n = M^{-1}(x_{n+1}). \tag{1}$$

The inverse function can be described as another function that "reverses" the function. It requires for $M(x)$ to have the properties (i) every y there is at least one x such that $y = M(x)$ and (ii) for each x, there is only one distinct $y = M(x)$. Then there would exist an inverse of the function $y = M(x)$ and it would be,

$$x = M^{-1}(y).$$

Example: Do the following functions have an inverse? If so what is the inverse?

(a) $M(x_n) = x_n + 2$.

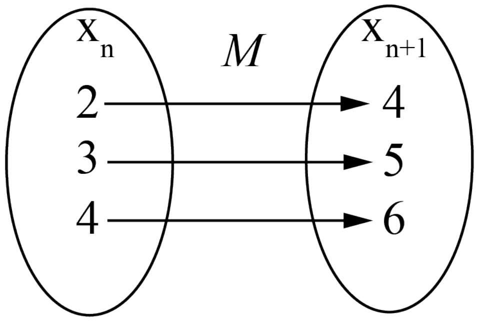


Figure 2: $M(x_n)$ has an inverse. Its inverse is $x_n = M^{-1}(x_{n+1}) = x_{n+1} - 2$.

(b) $M(x_n) = x_n^2$.

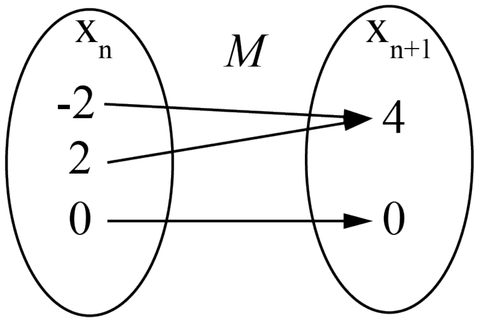


Figure 3: $M(x_n) = x_n^2$ does not have an inverse. $x_n = M^{-1}(x_{n+1}) = \pm\sqrt{x_{n+1}}$ indicates the inverse or reverse direction has two solutions.

Exercise 2: Do the following functions have an inverse? If so what is the inverse?

1. $M(x_n) = x_n - 1$
2. $M(x_n) = 2x_n$
3. $M(x_n) = 2x_n \pmod 1$
4. $M(x_n) = 2x_n + 1$
5. $M(x_n) = x_n^3$
6. $M(x_n) = x_n^4$

The modulus function (a.k.a mod map) does not have an inverse. However, given certain initial conditions, we can still predict a pattern of behavior. Why is our modulus function so special? Consider $M(x_n) = 2x_n \pmod 1$, setting $M(x_n) = x_{n+1}$, which is plotted in the figure below with x_{n+1} on the vertical axis plotted with respect to x_n on the horizontal axis

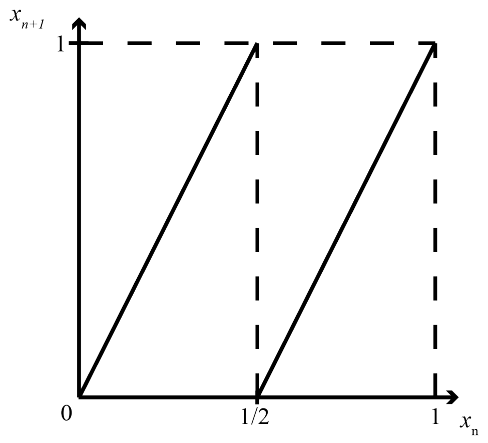
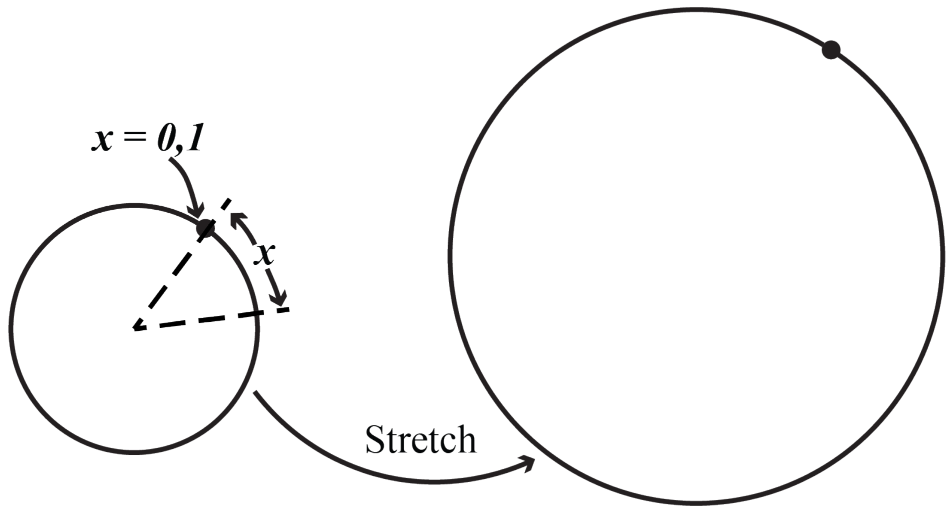
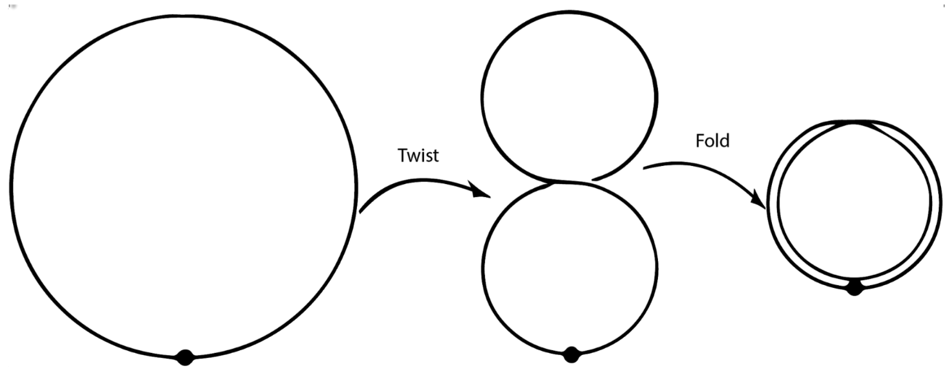


Figure 4: $M(x) = 2x_n \pmod 1$ function.

Let's go back to our circle of circumference 1 which is enumerated such that any point x on that circle has values $x \in [0, 1)$. If we take those values on the circle, $x \in [0, 1)$, and multiply them by 2 to get $2x$, we stretch the circle from its original circumference of 1 to a circumference of 2 with $x \in [0, 2)$.



Next, we apply the (mod 1) function to our stretched circle to get $2x \pmod 1$. The mod 1 function must fit the larger circle within $x \in [0, 1)$ and does so by a twisting and then folding it so the parts of the circle overlap the numbers with $x \in [0, 1)$.



Here we visualize how the mod map $M(x)$ cannot be invertible. Two values in the last folded form share the same spot, so it is not possible to figure out which is the initial value given its current position. However, there are mathematical features we can assess from a function with this behavior such as a pattern of the stretch, twist, and folding of this function. We show that it does not have an inverse, but at certain points, we can visualize the periodic behavior that we previously computed numerically.

Let's consider using substitution to apply the function to itself to get from the x_n value to the x_{n+2} value (i.e. forward two iterations),

$$\begin{aligned} x_{n+1} &= M(x_n) \\ x_{n+2} &= M(x_{n+1}) \\ \rightarrow x_{n+2} &= M(M(x_n)) \end{aligned}$$

We can see that the function applied twice to the value x_n gives the value x_{n+2}. More generally, we can denote applying the function to itself m times like,

$$\begin{aligned}
M^1(x) &= M(x)\\
M^2(x) &= M(M(x))\\
&\vdots\\
M^m(x) &= M\left(M^{m-1}(x)\right) = \underbrace{M(M(M(\ldots(M(x))}_{m \text{ times}}\ldots)))
\end{aligned}$$

So for the case above, we have $M(M(x_n)) = M^2(x_n) = x_{n+2}$. Let's consider the $M(x_n) = 2x_n \pmod 1$ function mentioned previously in Fig. 4. Applying this function to itself, we get a function that has the plot:

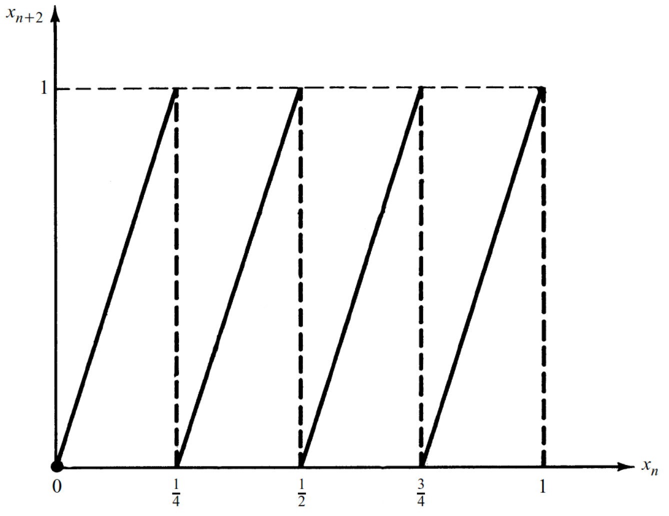


Figure 5: Plot of $M^2(x_n)$ where $M(x_n) = 2x_n$ (mod 1) function.

To understand how this figure was constructed think of applying Fig. 4 to itself. Look at Fig. 4 — all the points in the region $x_n \in (0, 1/2)$ on the x-axis is then mapped to x_{n+1} from $(0, 1)$ on the y-axis. The $x_{n+1} = (0, 1)$ are input into the function M to get x_{n+2}. This then means the plot in Fig. 4 fits in $x_n = (0, 1/2)$ of Fig. 5. You can repeat this construction for the region $x_n \in (1/2, 1)$ to get the right half. With this plot we can ask ourselves, are there any patterns that we get from applying this function twice? Are there particular values that when we apply the function twice gives us the original value — like the period-2 function which had a repeated pattern of two numbers? What we want can be written mathematically as

$$\begin{aligned}
x_n &= x_{n+2}\\
\rightarrow x_n &= M\left(M\left(x_n\right)\right) = M^2\left(x_n\right)
\end{aligned}$$

Above, we were able to find the period-2 orbit by starting at an initial condition of 1/3 or 2/3. In this case, the two points were given. How do we determine them if we aren't lucky enough to guess them? These special numbers that exhibit the repeating pattern of behavior are one type of fixed point. A point x_0 is a **fixed point** for a function $f(x)$ if $f(x_0) = x_0$. Together we will go over a method for finding the fixed points of our mod map. Let's refer to Fig. 5. We begin by drawing the diagonal line from 0 to 1 to represent the points where $x_n = x_{n+2}$. We find the fixed points when our $M^2(x)$ function intersects with this line.

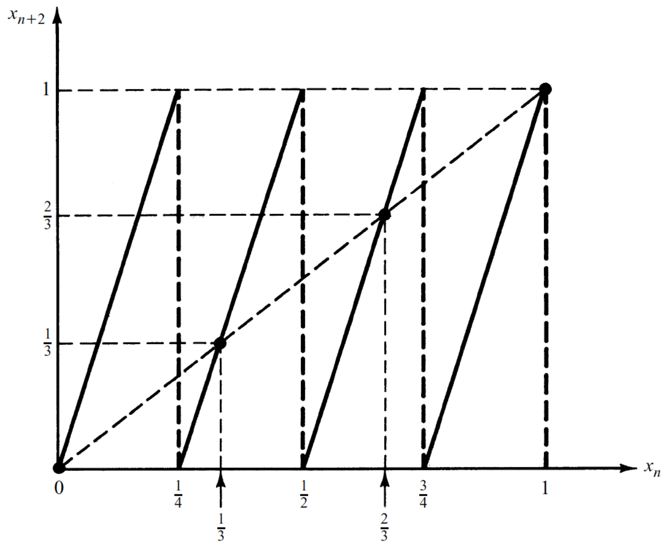


Figure 6: Visualizing the fixed points of the period-2 orbit of the $M(x_n) = 2x_n$ (mod 1) function.

Suppose we did not know those fixed points before. Using the graph above, we can see the $x_n = x_{n+2}$ line crosses a point of our function in the regions $x_n \in (1/4, 1/2)$ and $x_n \in (1/2, 3/4)$. We can compute the lines crossed by extending the lines and using the point-slope intercept formula

$$x_{n+2} = mx_n + b$$

Where b is the y-intercept and $m = \Delta y/\Delta x$ is the slope. The region $x_n \in (1/4, 1/2)$ has the line,

$$L1 : x_{n+2} = 4x_n - 1$$

Since we are finding where $L1$ and $x_{n+2} = x_n$ intersect, we can substitute $x_{n+2} = x_n$ into $L1$ and drop the n notation,

$$\begin{aligned} x_{n+2} &= 4x_n - 1 \\ x_n &= 4x_n - 1 \\ x &= 4x - 1 \end{aligned}$$

Now, solve for x.

$$\begin{aligned}
x &= 4x - 1 \\
x - x &= 4x - x - 1 \\
0 &= 3x - 1 \\
1 &= 3x \\
\frac{1}{3} &= x \rightarrow \text{which is our first fixed point!}
\end{aligned}$$

The second region has the line, $x_{n+2} = 4x_n - 2$.

<u>Exercise 3:</u>
Using the technique above, find the fixed point for the second region.

Above we described the example of a period-2 orbit for the $2x$ (mod 1) function. We discovered the fixed points by plotting the graph of the $M^2(x)$ function. However, there are many more patterns to be found from this function each with p distinct fixed points denoted $\bar{x}$ of a p-period orbit with the property,

$$\bar{x} = M^p \bar{x}.$$

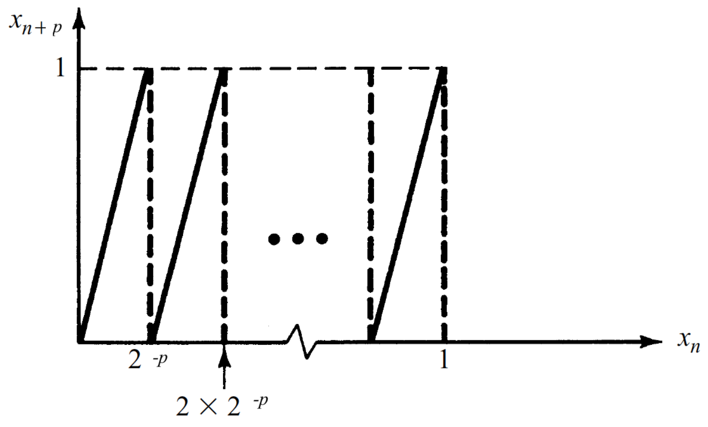
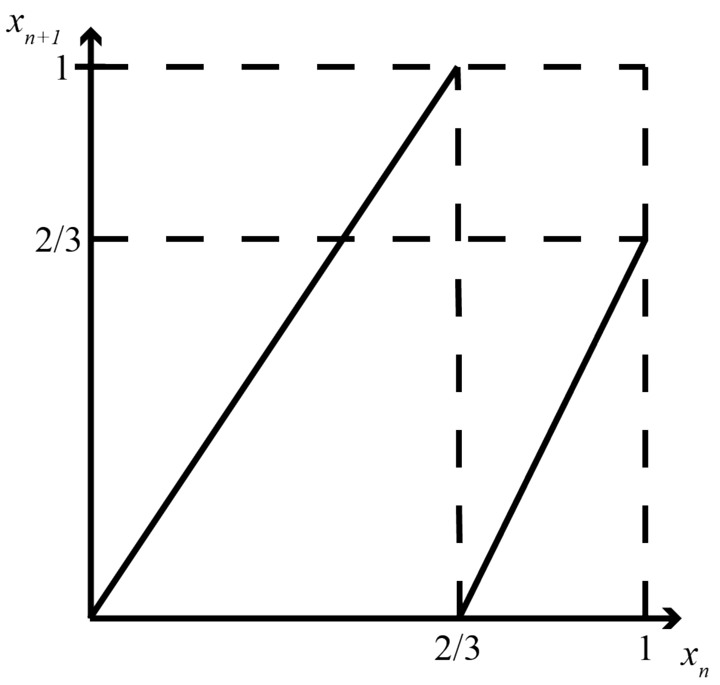


Figure 7: Plot of $M^p(x_n)$ where $M(x_n) = 2x_n$ (mod 1) function. Figure from [1].

Let's look at a slightly more irregular function than the $2x \pmod 1$ function. Consider the following function given in the diagram below.



Exercise 4:
Find the fixed points of the period-1 orbit, the period-2 orbit, and the period-3 orbit. Graph paper will help in this exercise.

1. The period-1 fixed points can be spotted on the diagram above.
2. Start by plotting x_n vs. x_{n+2}. Graph paper would be best. Check this plot with your team leader.
3. Draw the diagonal line $x_n = x_{n+2}$.
4. Find the formula for the lines that intersect.
5. For each line, substitute $x_n = x_{n+2} = x$
6. Solve for the fixed points, x.
7. Check that $x = M^2(x)$ on the plot.
8. Repeat the process but with x_n vs. x_{n+3} instead of x_n vs. x_{n+2}.

You will find there are 3 distinct period-1 orbits that all happen equal the same number by the modulus definition. There are 3 period-2 orbits, with 0 distinct period-2 orbits. Lastly, there are 3 distinct period-3 orbits.

Python Exercise 2: Now we will use computation tools to study the fixed points of the quadratic function and check our answers above. Navigate to our github to find the Jupyter PYTHON notebooks and select 'Fixed_Points.ipynb.' In this notebook, you will make changes as instructed. Follow the instructions, running the cells one-by-one. To check Exercise 4, you will have to use your new knowledge about '`if`' statements and the '`%`' for mod function.

2.2 Counting the number of fixed points

Sometimes, we like to have an idea of how many fixed points we have in our period-p orbit. In particular we might be interested in how many fixed points are in a particular period-p orbit and no other orbit. That is to say, the fixed points are *distinctly* period-p fixed points. For example, let's return to our $M(x) = 2x \pmod 1$ function. Recall that 1/3 and 2/3 were the fixed points of the period-2 orbit. When we plug one of those in $M(x)$, we get the list of numbers

$$\left(\ldots, \frac{1}{3}, \frac{2}{3}, \frac{1}{3}, \frac{2}{3}, \ldots\right)$$

and we can see the sequence of $(1/3, 2/3)$ repeat every two iterations. Note that when we plug in 0 we get

$$(\ldots, 0, 0, 0, 0, \ldots).$$

In this, we see the sequence $(0, 0)$ repeat every two iterations. It's not a very interesting period-2 orbit, but it satisfies the definition of a period-2 orbit. However, the fixed point 0 is not distinctly a period-2 orbit as it is also part of the period-1 orbit since 0 repeats every single iteration. Not only that it repeats itself every 3 iterations, 4 iterations, 5000 iterations, etc. We notice that the number 0 (which is equivalently 1) are going to be a fixed point in any period-p orbit. We will remove these fixed points from each estimate for the number of distinct fixed points in each orbit. We will show our estimate for the total number of fixed points is exact if the orbit happens to be a **prime** – meaning the only way to produce a whole number from the division of a prime is by dividing by the number 1 and itself.

Exercise 5:
Which of the following numbers are primes?

(a) 23

(b) 4

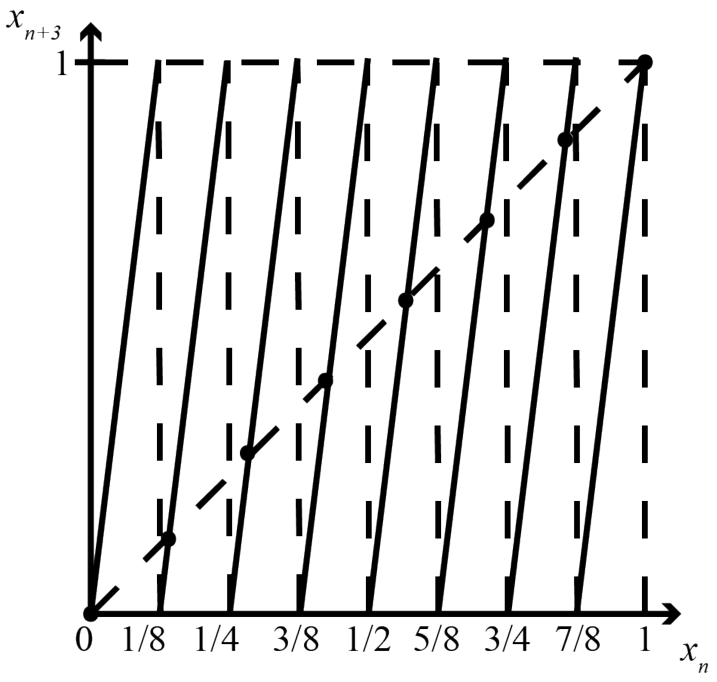
(c) 21

(d) 39

(e) 2

(f) 247

When p is a prime, the upper estimate for the number of distinct fixed points in a period-p orbit is an equality. Let's calculate how many fixed points there would be in a period-p orbit of the $M(x) = 2x \pmod 1$ function if p is prime. Recall for $M^2(x)$ we have the fixed points $(0, 1/3, 2/3, 1)$ where 0 and 1 are equivalent and not distinctly period-2. Therefore we subtract the non-distinct points from the total number of period-2 fixed points and can say we have $4 - 2 = 2$ distinct period-2 fixed points. For $M^3(x)$ we have the distinct fixed points of the period-3 orbit to be $8 - 2 = 6$. See the figure below of x_n vs. x_{n+3}.



If we continue to explore this pattern we find that there most $2^p - 2$ distinct fixed points for each period-p orbit. For prime orbits, there are exactly $2^p - 2$ distinct orbits. For numbers that aren't prime, there are fewer distinct orbits. Consider $p = 4$. There are $2^4 = 16 = 16$ total orbits. Two of those period-4 orbits are

$$\left(\ldots, \frac{8}{15}, \frac{1}{15}, \frac{2}{15}, \frac{4}{15}, \frac{8}{15}, \frac{1}{15}, \frac{2}{15}, \frac{4}{15}, \frac{8}{15}, \frac{1}{15}, \ldots\right)$$

$$\left(\ldots, \frac{1}{3}, \frac{2}{3}, \frac{1}{3}, \frac{2}{3}, \frac{1}{3}, \frac{2}{3} \ldots\right)$$

Notice in both of these we have a sequence of 4 numbers repeat in a pattern. However, in the 2nd, we can see that this is also a period-2 orbit. Since 4 can be divided by 2, it is not a prime. This is an example of how $2^p - 2$ is only an upper estimate for the number of distinct fixed points in a period-p orbit.

Exercise 6:
Answer the following questions for the $M(x) = 2x \pmod 1$ function.

1. What is the exact number of distinct period-3 fixed points?
2. What is the exact number of distinct period-4 fixed points? [Hint: $4 = 2 \times 2$. Remember to subtract the number of period-2 fixed points.]
3. What is the exact number of distinct period-6 fixed points?
4. What is the exact number of distinct period-7 fixed points?
5. What is the exact number of distinct period-8 fixed points?

3 The logistic map

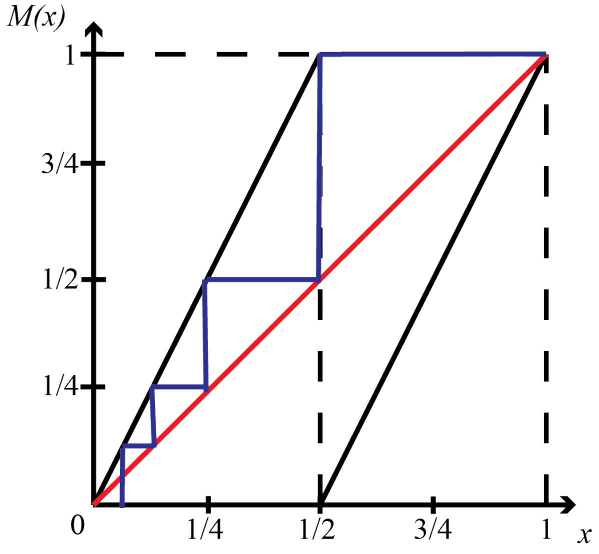
In the previous section we discussed how we can find and count the number of fixed points in the periodic set of the $M(x) = 2x \pmod 1$ function. Now we will consider what happens outside of those numbers for the uncountable amount of other values that can be initialize in the function. If we give our $M(x) = 2x \pmod 1$ function an initial condition like 1/4, we can see it produces the iterations,

$$\left(\ldots, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1, 1, 1, \ldots\right)$$

meaning by initializing the function from the point $\frac{1}{16}$ (and also the point $\frac{1}{8}$, $\frac{1}{4}$, and $\frac{1}{2}$) the sequence approaches the fixed point 0. We say that the function $M(x)$ **converges** for certain initial condition x if it approaches a constant valued fixed point. For other initial conditions, the sequence **diverges**, that is, to approach $-\infty$ or ∞. Lastly, the iterations might not diverge nor converge to a single value. They might oscillate between a few values (periodic) or exhibit no pattern at all. This latter irregular behavior we refer to as **chaos**.

3.1 Cobweb Maps

There is a way to visualize the converging and diverging from an initial condition called Cobweb Maps. Below is a cobweb map of the $M(x) = 2x \pmod 1$ function starting at initial condition $x = 1/16$.



To understand this diagram, we follow the blue line or the 'cobweb'. The black line is our $M(x) = 2x \pmod 1$ function and the red line is $M(x) = x$. Our cobweb starts at $1/16$. $M(1/16) = 1/8$ which is where the blue line first intersects the black line on the far left. The cobweb then travels right, back to the red line where $M(x) = x$. The value of 1/8 is reinitialize so then the blue line travels to intersect the black line at $M(1/8) = 1/4$. Then right back to the red line where it is reinitialized with 1/4. It then travels up to intersect the black line at $M(1/4) = 1/2$. From here is travels right to the red line. It then reinitalizes to get $M(1/2) = 1$. Then the blue line travels right to 1 which is a fixed point so it stays there.

The following exercise gives you some practice with a function we call the 'logistic map',

$$\boxed{M(x) = rx(1 - x)}$$

where r is a constant parameter we choose and x can be any real number. For this exercise we will look at $r = 2.8$ and experiment with initial conditions and ask ourselves whether the solution converges or diverges. Then increase the value of r and observe the behavior.

How can we make sense of this? Is there some kind of pattern we find in the iterations? How does that change with r? Let's check it out with our tools.

Python Exercise 3:
Navigate to our github and click on the link 'Cobweb_Plots.ipynb.' Under the "Runtime" tab, click "Run all" to have the code run. Observe the results. Now that you've run it, modify to the initial conditions and parameter in line 3 and re-run the code.

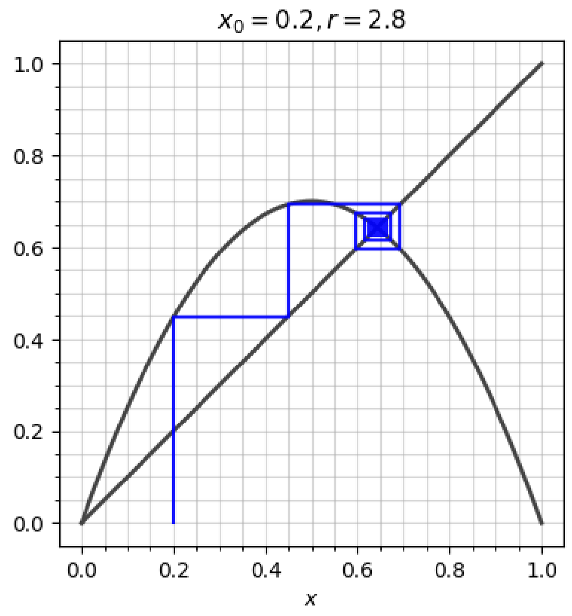
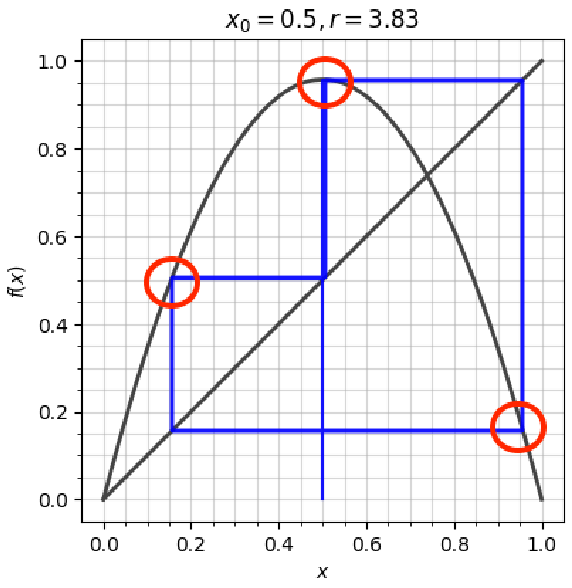
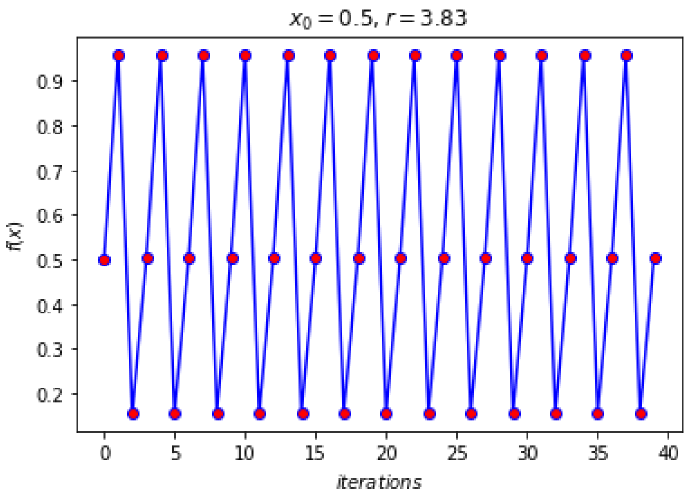


Figure 8: Cobweb plot of the logistic map.

3.2 Bifurcation Diagram

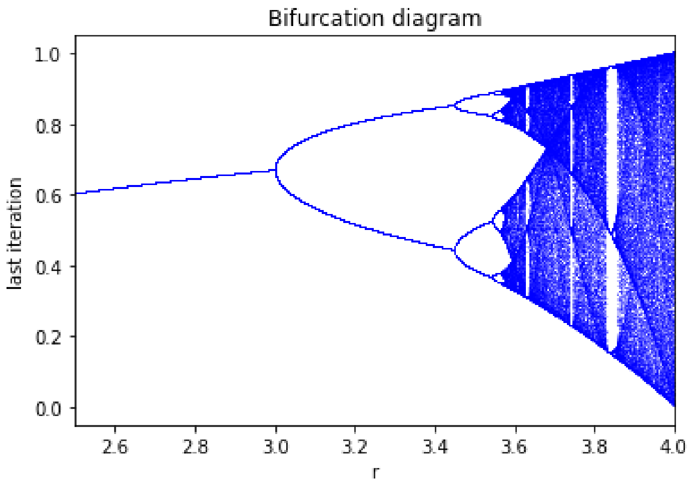
In this section, we explore the effects of changing the initial conditions and the parameter r on the iteration of the logistic map. We notice that if the initial condition is between 0 and 1 and with the same r, we see the plots replicate the same behavior after many iterations. For example, $r = 0.28$ and any initial condition, $x_0 \in (0, 1)$, will iterate to the fixed point, at around 0.64288. When we increase the parameter $r = 3.1$, we find that all initial conditions don't converge to a single value but oscillate between two fixed points as a period-2 orbit. Increasing r then produces a period-4 orbit and after a small increase a period-16 orbit, and so on until chaos. Chaos continues until we enter a window when we look at $r = 3.83$ —something changes. We see a period-3 orbit as in the figure below.





There's a more systematic way of studying many parameters of r. In this next python exercise, we run a code to generate a bifurcation diagram for the logistic map. The bifurcation diagram plots the information discussed above. Along the horizontal axis we have different values of the parameter r. The values of the fixed points in the orbits resulting from iterating with any initial value between 0 and 1 is then plotting the final iterate in the diagram.

Python Exercise 4:
Navigate to our github and click on the link 'Bifurcation_Diagram.ipynb.' Under the "Runtime" tab, click "Run all" to have the code run. Observe the results. Zoom in by changing 'xlim' in the code to look at the regions of r we studied in the last exercise.



4 Julia Sets

In the previous section, we created a bifurcation diagram for the logistic map. You notice patterns in the bifurcation diagram. That when zooming in, we can see the large scale features repeated at smaller scales. This is an example of a **fractal** which describes the self-similar structures recurring on several scales. We also notice that for certain initial conditions or parameters, functions can diverge off to $-\infty$ or ∞. Sometimes, the sequence does neither but stays in a specific range of numbers — that is to say it is **bounded**. It was natural in the modulus map for it be bounded. As for the quadratic function and the logistic map, we discovered that for certain initial conditions, the outputs are bounded under iteration. In this section we consider the quadratic function

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

where z and c are complex numbers. For different values of c we will examine iterations of this function and look at the initial conditions and the rate they converge and diverge. This will generate beautiful fractal patterns known as Julia sets.

4.1 Complex Numbers

In the previous sections, we dealt with iterating a real-valued number, x. Now, let's consider the **imaginary** number which can be defined as

$$i = \sqrt{-1}.$$

So if you were every told in math class, that the square root of a negative number has no solution, they are wrong! Well, somewhat wrong. The square root of a negative number has has no real-valued solution, and instead has an imaginary solution. Furthermore, we denote a **complex** number to be the sum of the real and the imaginary numbers by

$$z = ai + b$$

where $Im(z) = a$ is the imaginary part of z and $Re(z) = b$ is the real part of z.

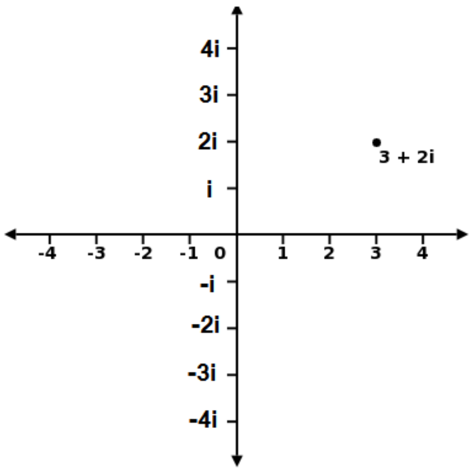


Figure 9: How we plot the complex number $3 + 2i$.

In the figure above, we can visualize complex numbers by representing the real part on the x-axis and the imaginary part on the y-axis. Therefore, we will be visualizing the complex-valued iterations of the Julia Sets and Mandelbrot Sets on a 2D plane. Before we begin with computation, let's study products and sums of complex numbers to get a better understand of the math involved in building these sets. Let's begin with the sum. The sum of complex number $a + bi$ and $c + di$ is

$$(a + bi) + (c + di) = (a + c) + (b + d)i.$$

Similarly, the difference of complex number $a + bi$ and $c + di$ is

$$(a + bi) - (c + di) = (a - c) + (b - d)i.$$

The product of two complex numbers $a + bi$ and $c + di$ is bit less intuitive. You can think of it like using the FOIL (First Outer Inner Last) rule from algebra.

$$(a + bi)(c + di) = (ac - db) + (ad + bc)i$$

The product of two complex numbers $a + bi$ and $c + di$ is bit less intuitive. Use the FOIL (First Outer Inner Last) rule from algebra and the fact that $i^2 = (\sqrt{-1})^2 = -1$ to get

$$\begin{aligned}(a + bi)(c + di) &= ac + adi + bci + dbi^2 \\ &= ac + adi + bci - db \\ &= ac - db + adi + bci \\ &= (ac - db) + (ad + bc)i\end{aligned}$$

The last definition is the norm of the complex number, $a + bi$ is defined as

$$|a + bi| = \sqrt{a^2 + b^2}.$$

Exercise 7:

Write the following in a simplified complex number form and plot them in the complex plane.

(a) $3 + 2\sqrt{-1}$

(b) $3 + \sqrt{-4}$

(c) $1 + \sqrt{-16} + \sqrt{-9}$

(d) $\frac{4+\sqrt{-16}}{2}$

(e) $\frac{9-6i}{3}$

(f) $(2 + 3i)(4 + i)$

(g) $i(1 + 3i)$

(h) $(i + 1)(i + 1)$

(i) $(i + 1)(i - 1)$

(j) $|1 + i|$

(k) $|1 - i|$

Now that we are familiar with operating with complex numbers, we will have the computer do the iterating work for us. In the next exercise, you will learn about how the Julia set it constructed by iterating a complex number through the quadratic function,

$$f(z) = z^2 + c,$$

where c is a parameter. With a fixed c, we choose different initial conditions, z_0, within a rectangular domain and iterate each. If that initial condition diverges, it is marked as black. Otherwise, when the absolute value of the iterations gets larger than 2 and $|c|$ we color it based on how many iterates it took to get theres. Different values of c produce different fractals. The syntax for formatting a complex value in python is for $z = a + bi$,

```
z = complex(a,b)
```

Python Exercise 5:

Navigate to our github and click on the link 'Julia_set.ipynb.' Read the information and run the cells one-by-one. After you create your first Julia Set, you will use a function and input different values of the parameter c to get their Julia Set.

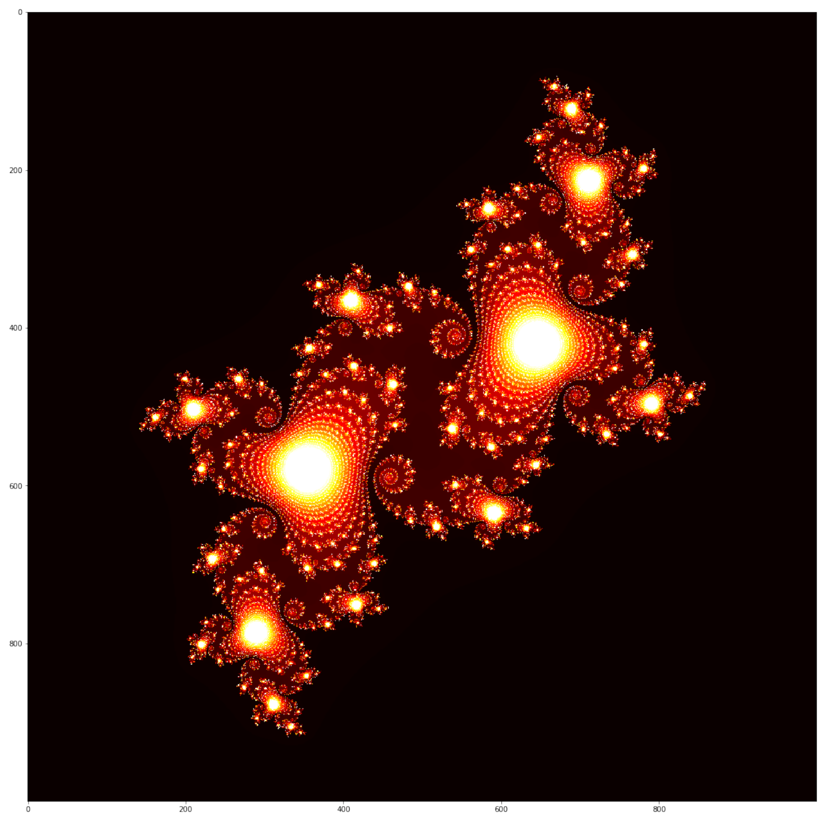


Figure 10: Plot of the Julia Set for $c = -0.1 + 0.65i$.

5 Mandelbrot Sets

In the last section we studied Julia sets for particular c over many initial conditions. In this section will study the Mandelbrot Set which varies the parameter c and starts with the initial condition 0. You may have noticed that if we iterate starting from 0 we get a pretty good idea about the dynamical properties of the function.

More specifically for $f(z) = z^2 + c$, if the orbit of 0 is attracted to a fixed point, there is only one attractive fixed point. If it is attracted to a periodic cycle, that is the only attractive periodic cycle. However, if the orbit of 0 diverges, there are no attractive cycles.

For Mandelbrot sets we want to observe the behavior of the parameter space — or how c changes. This just means that instead of changing the initial condition for the iteration, we are changing the c value and observing what happens to 0 under iteration for each parameter c. For a particular c if the orbit stays bounded under iteration, we say that c is in the Mandelbrot set (shown in color), if the orbit diverges, we say it is not in the Mandelbrot set (shown as black).

Python Exercise 6:
Navigate to our github and click on the link 'Mandelbrot_set.ipynb.' Read the information and run the cells one-by-one. Using what you learned from the Julia set algorithm, you will apply this and the new knowledge about the Mandelbrot set to complete the Mandelbrot algorithm. Once you complete the exercise you will observe the iconic Mandelbrot set.

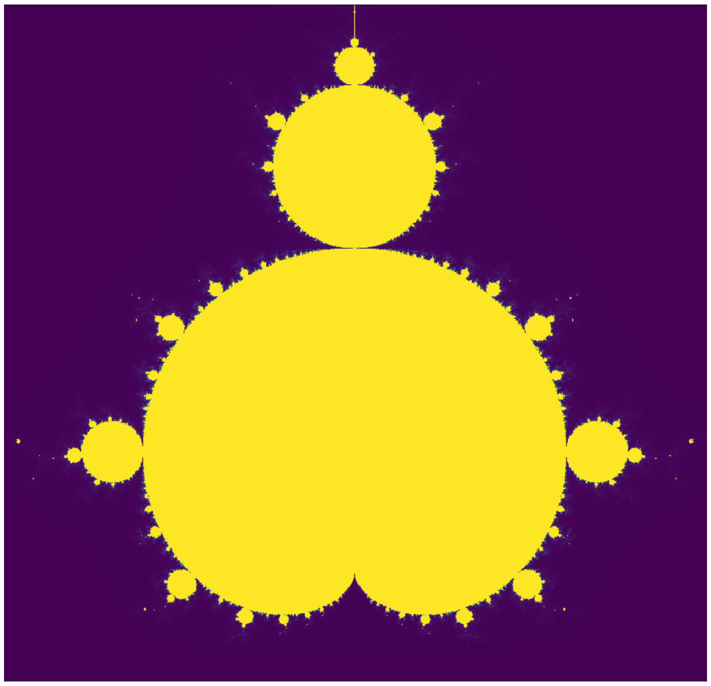


Figure 11: Plot of the Mandelbrot Set.

6 Related Videos

1. Filled Julia Set: `https://www.youtube.com/watch?v=oCkQ7WK7vuY`
2. Mandelbrot Set: `https://www.youtube.com/watch?v=NGMRB4O922I`
3. Pi and the Mandelbrot Set:
   `https://www.youtube.com/watch?v=d0vY0CKYhPY&t=21s`
4. Zooming in on a Mandelbrot Set:
   `https://www.youtube.com/watch?v=pCpLWbHVNhk`
5. Julia sets at points in the Mandelbrot set:
   `https://www.youtube.com/watch?v=dekA3GNm6Qk`

7 References

1. Ott, Edward. Chaos in dynamical systems. Cambridge university press, 2002. 2nd edition Chapter 2: One Dimensional Maps. p 24-70.
2. Wikipedia pages for Julia Sets and Mandelbrot Sets.