Numerical Analysis: Project I
Using Iteration Methods to Understand Fractal Geometry

Abigail Saenz, Cecilia Rodarte, Dannie Kiel, Taylor Reeder

Winter 2020

Introduction

In this project, we implemented computer programs in MATLAB that used iterative methods to generate fractal figures on the plane. Specifically, we used these methods to generate the Mandelbrot set given by the quadratic map $f_c(z) = z^2 + c$ on the complex plane.

Part I - An Introduction to Fractals

The Mandelbrot set is the set of complex values c for which the orbit of 0 under iteration of the quadratic map $f_c(z) = z^2 + c$ remains bounded in absolute value. If for a given c the $|orb(0)| > 2$, the sequence will diverge and the value c is not contained in the Mandelbrot set. A point c is in the Mandelbrot set exactly when the Julia set corresponding to that c is connected.

If c is in the Mandelbrot set, we can generate the corresponding Julia set for c using the complex quadratic polynomial $J_c(z) = z^2 + c$. Instead of computing the orbit under 0, we hold c fixed and change the value of z.

Let us take $c = 0$. For $|z| = 1$, the outcome is is the unit circle. For all $|z| < 1$, the sequence of iterates will converge to 0. If we take $|z| > 1$, the sequence of iterates will diverge. Thus, the filled Julia set of $J_c(z) = z^2 + c$ for $c = 0$ is the unit disk.

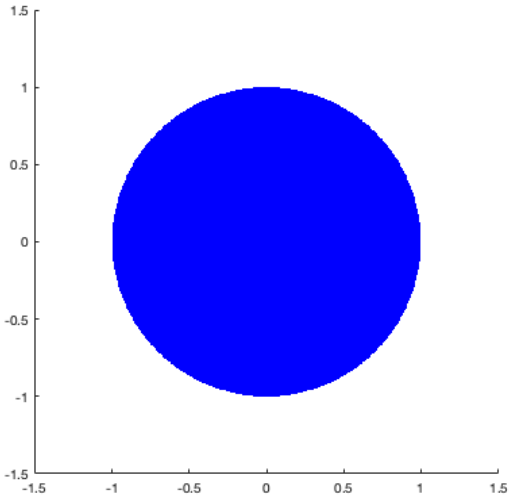
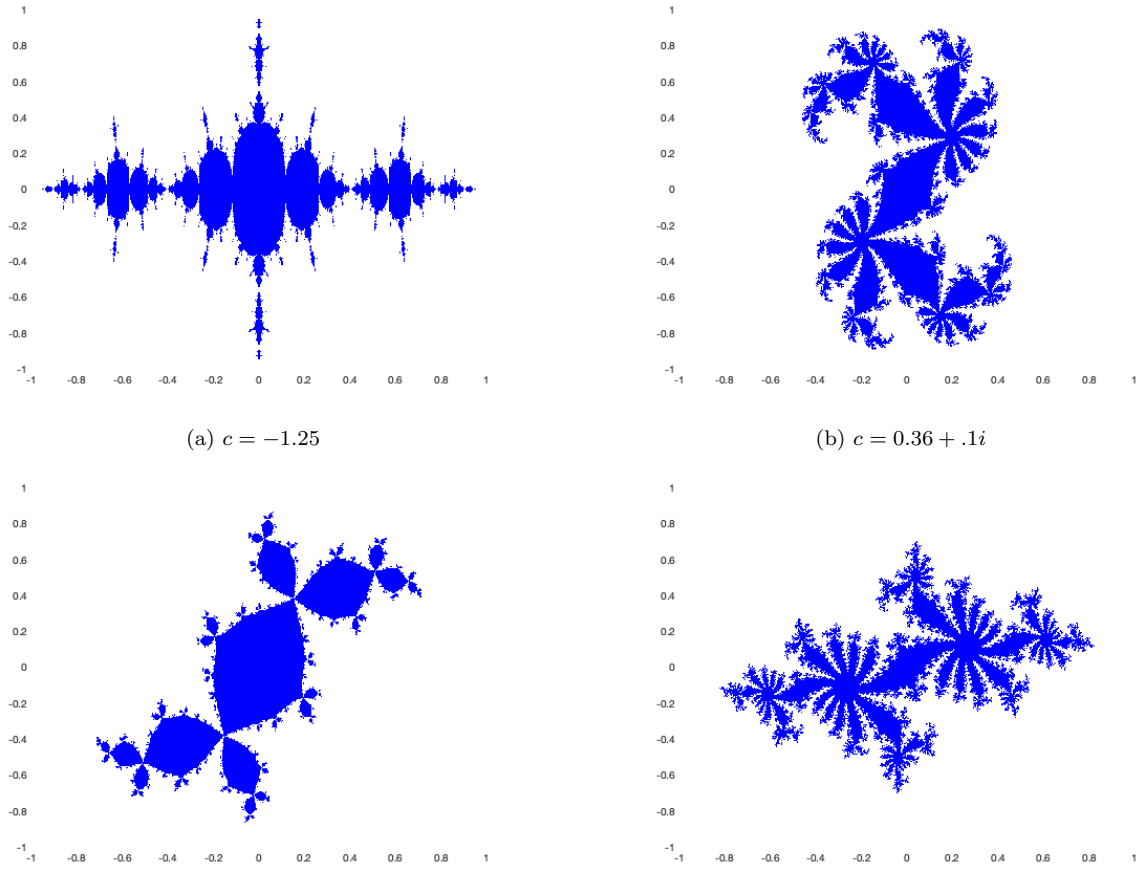


Figure 1: Filled Julia set for the value $c = 0$

Part II - Generating Examples for Various Complex Values c

Adding a nonzero constant c to the complex function $J_c(z) = z^2+c$ allows us to generate more interesting results...fractals! Different Julia sets may be generated by changing the value of c. If we fix some value of c and produce a sequence of iterates for $J_c(z) = z^2 + c$ under different values of z_0, we will obtain the Julia set for our chosen complex value.

In Greenbaum, A., Chartier, T.P. (2012), *Numerical Methods: Design, Analysis, and Computer Implementation of Algorithms*, an example script was provided for the value $c = 1.25$. We duplicated this example with the script provided. We also made our own version implementing the same algorithm and generated the Julia set for various values of c. The results are shown below.



(a) $c = -1.25$

(b) $c = 0.36 + .1i$

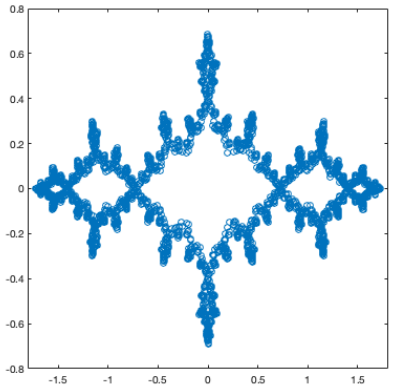
(c) $c = -0.123 - 0.745i$

(d) $c = -0.70176 - 0.2842i$

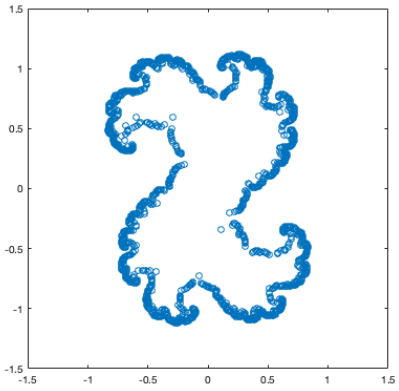
Figure 2: Julia sets for different values of the parameter c.

Part III - Constructing the Julia Set

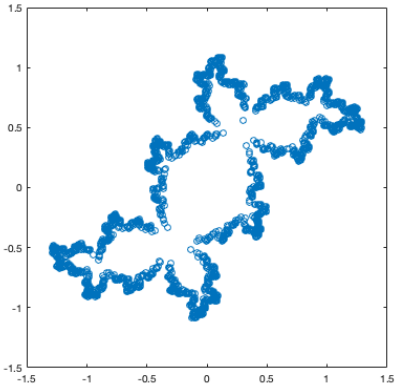
Now, we provide an alternative method for generating the Julia set: Inverse Iteration Method. Given a complex number $z = x + iy = r(\cos\theta + i\sin\theta)$ with $r = \sqrt{x^2+y^2}$ and $\theta = \arctan\frac{y}{x}$ when $x > 0$ and $\sqrt{z} = \sqrt{r\cos\frac{\theta}{2}} + i\sqrt{r\sin\frac{\theta}{2}}$, we developed an iteration method for the function $\psi = \pm\sqrt{w-c}$, where $w = J_c(z)$. For each number, we randomly picked the positive or negative value of the expression for ψ. The resulting Julia sets are provided below.



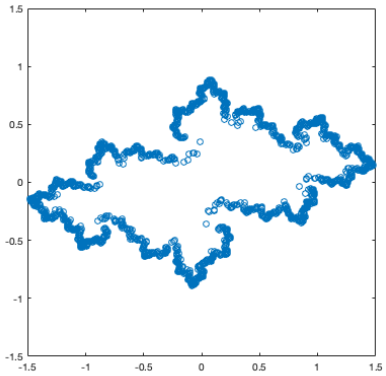
(a) $c = -1.25$



(b) $c = 0.36 + 0.1i$



(c) $c = -0.123 - 0.745i$



(d) $c = -0.70176 - 0.2842i$

Figure 3: Julia sets for different values of the parameter c using the Inverse Iteration Method.

Part IV - Computing the Fractal Dimension

A fractional dimension is classified on a scale that determines the texture of a surface. Calculations of the fractal dimension will give the researcher some insight on the smoothness of the surface. So, the fractal dimension gives a numerical value for the concept of a smooth or rough surface. For a 2D image, like the generated image for the Julia set, the fractal dimension of 2 corresponds to a polished surface, while the fractal dimension of 3 corresponds to a maximum grit. By calculating the dimension somewhere between these values, we can achieve a precise value that encapsulates whether the Julia set and disk are jagged or not.

The following process for calculating the fractal dimension is described in Bisoi A.K. and Mishra J. (2001) *On calculation of fractal dimension of images.*Given A, the fractal dimension has a dimension D defined by:

$D = \frac{log(N)}{log(1/r)}$ Where N = total number of copies of A and $\frac{N}{1/r}$ = a scaling down of A in the copy.

The overarching idea used in calculating the fractal dimension is to:

1. Take a set A (i.e. our image) and impose a grid (i, j) where boxes are size LxL (which defines $1/r$ because it sets the scale)

2. Select a particular box (k, l) in the grid and calculate its contribution to the number of colored boxes by the formula $n_r(i, j) = l - k + 1$

3. Create a count N_r of the total number of boxes with color in it based on the sum over (k, l) of contributions from each box in step (2)

4. Create a $log(N_r)$ v. $log(1/r)$ plot by using different values of r (i.e. varying L and repeating all previous steps) to create many values for N_r

5. $D = -$(slope of least square linear fit line of the plot) **note the negative here!

We implemented this algorithm in MATLAB (see script in appendix) and got the following results:

```
The fractal dimension for the complex value c = -1.250 + 0.000 is: 1.8614
The fractal dimension for the complex value c = 0.360 + 0.100 is: 2.2332
The fractal dimension for the complex value c = -0.123 + -0.745 is: 2.3809
The fractal dimension for the complex value c = -0.702 + -0.284 is: 2.4787
```

Part V - Connectivity of the Julia Set

A Julia set is connected if you can draw a line from one point contained in the set to another without ever leaving the set. If $orb(0)$ is bounded, it follows that the set is connected. Thus, the connectivity of the Julia set can be determined by computing the $orb(0)$ for a given c and testing for divergence. Because too few iterations may fail to diverge and too many iterations may be computationally expensive, we will assume that divergence occurs for $|z| > 100$.

Using our MATLAB script (see appendix), we tested for divergence for the points -1.25 and 1.25. From the previous examples, we know that $c = -1.25$ is in the Mandelbrot set and that we should expect the corresponding Julia set to be connected. We find that for the second tested value, $c = 1.25$, divergence occurs. The output of our script is given below.

For c = -1.25:

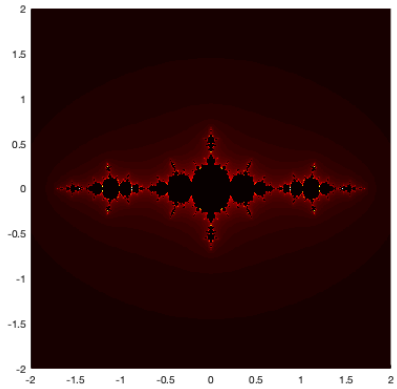
```
We assume that divergence occurs for |z| > 100.
After 101 iterations, the set failed to diverge.
We can assume that the Julia set is connected for c = -1.250.
```

For c = +1.25:

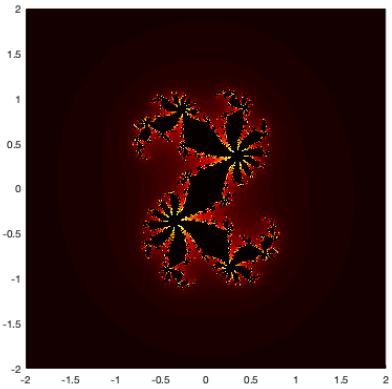
```
We assume that divergence occurs for |z| > 100.
The orbit diverged after 5 iterations.
We can conclude that the set is not connected for c = 1.250.
```

Part VI - Coloring Divergent Orbits

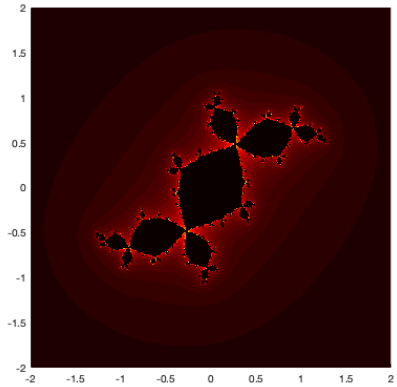
In this section, we write a MATLAB script (see appendix) that assigns a color to the divergent orbits in the Julia set. We assign colors according to the time it takes for the orbits to diverge. Two orbits that reach our baseline requirement for divergence (i.e. $|z| > 100$) at nearly the same amount of iterations will be colored similarly. Our results are shown below.



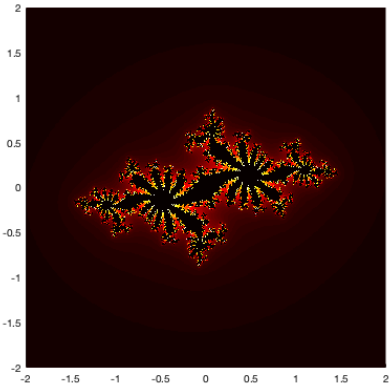
(a) $c = -1.25$



(b) $c = 0.36 + .1i$



(c) $c = -0.123 - 0.745i$



(d) $c = -0.70176 - 0.2842i$

Figure 4: Coloring divergent orbits for different values of c.

Other values of c may be tested using the same MATLAB script provided in the appendix. We expect all of the c values tested in previous part to converge because we have shown that each of these values of c are contained in the Mandelbrot set. Thus, these values of c will yield a connected Julia set when tested.

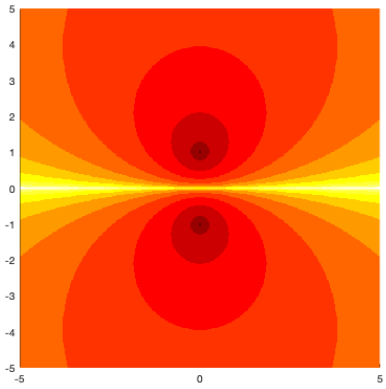
Part VII - Newton's Method in the Complex Plane

Now, we aim to find the roots of polynomials of the form $\phi(z) = z^n - 1$ for some fixed n. In order to do this, we must apply Newton's method onto the complex plane. Newton's method uses the formula $z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)}$ to generate a series of points that converges onto the root.

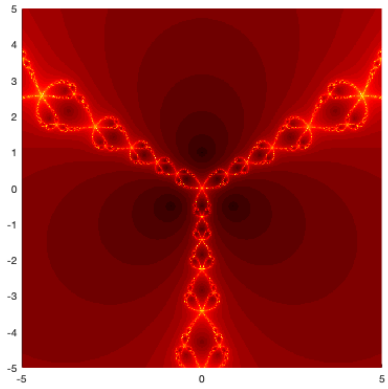
The complex plane is broken down into a real axis and an imaginary axis. When the iteration produces c as an imaginary or complex value, then we see the orbit reacts in a different pattern, but still goes to infinity but not every time. Below are the results of our script.

```
For n = 2, The root is near -1.0120, -0.0100
For n = 2, The root is near -1.0120, 0.0100
For n = 2, The root is near -0.9920, -0.0301
For n = 2, The root is near -0.9920, -0.0100
.
.
.
```

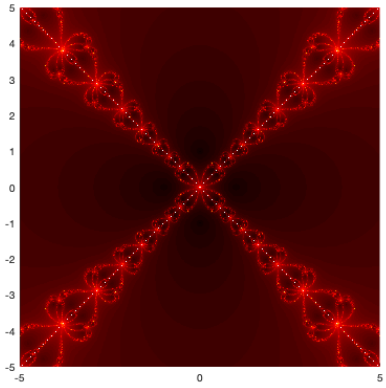
We have chosen to omit part of the output for the sake of space, but the script can be found in the appendix. Below are the images for fixed n.



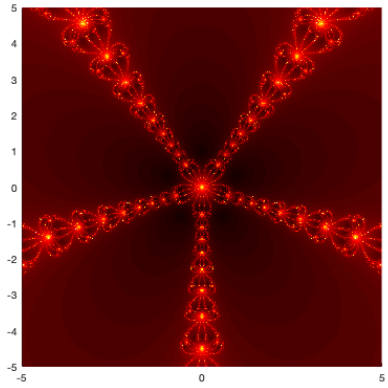
(a) $n = 2$



(b) $n = 3$



(c) $n = 4$



(d) $n = 5$

Figure 5: Iterations for the roots of the complex function $\phi(z) = z^n - 1$.

Part VIII - The Mandelbrot Set

Finally, we arrive at our concluding portion: The Mandelbrot Set. The Mandelbrot Set is the set of all c for which the iteration $\phi(z) = z^2 + c$ when beginning with $z = 0$ does not diverge. According to Yale's "The Mandelbrot Set and Julia Sets," Julia Sets can be either "connected or a dust of infinitely many points" and the Mandelbrot set is the set of c values for which the Julia set is connected. The Mandelbrot set will look similar to the Julia set because it ultimately is the same when we bound an orbit with the complex plane. So, this part builds off of the orbits in Part V and the complex plane in Part VII. We will color the set of c values according to how fast it takes for the orbit of 0 to diverge from infinity.

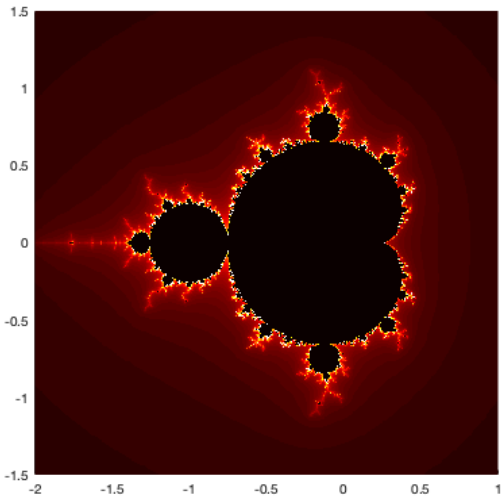


Figure 6: The Mandelbrot Set

References

Bisoi, A. K., and Mishra, J. (2001). On calculation of fractal dimension of images. Pattern Recognition Letters, 22(6-7), 631-637.

Devaney, R., and Keen, L. (1988). Chaos an fractals. The mathematics behind the computer graphics. American Mathematical Society.

Devaney, Bob. Mandelbrot Set Explorer, http://math.bu.edu/DYSYS/explorer/def.html

The Mandelbrot Set and Julia Sets. Retrieved from https://users.math.yale.edu/public_html/People/frame/Fractals/MandelSet/welcome.html

Greenbaum, Anne, and Timothy P. Chartier. Numerical Methods: Design, Analysis, and Computer Implementation of Algorithms. Princeton University Press, 2012.

Task Distribution

We met as a group every Saturday at 10 at the library to talk about task allocation and catch each other up on the progress we had made. We also kept in contact during the week via a google doc where we wrote our our discoveries as we went. Abigail Saenz and Dannie Kiel then typeset those ideas via Overleaf into the final document.

While we all discussed each part and collaborated, the work mostly was divided as follows:

PART I-III :
Script written by Abigail Saenz and Cecilia Rodarte
Write-up by Abigail Saenz and Cecilia Rodarte

PART IV :
Script by Dannie Kiel and Taylor Reeder
Write-up by Dannie Kiel and Taylor Reeder

PART V :
Script by Abigail Saenz and Cecilia Rodarte
Write-up by Abigail Saenz and Cecilia Rodarte

PART VI :
Script by Abigail Saenz
Write-up by Abigail Saenz and Dannie Kiel

PART VII :
Script by Dannie Kiel and Taylor Reeder
Write-up by Dannie Kiel and Taylor Reeder

PART VIII:
Script written by Abigail Saenz
Write-up by Abigail Saenz, Cecilia Rodarte, Dannie Kiel, and Taylor Reeder

Typeset using LaTeX by Abigail Saenz and Dannie Kiel