WXML Final Report: Shapes of Julia Sets

Mariana Smit Vega Garcia, Malik Younsi, Peter Lin, Siyuan Ni, Ryan Pachauri and Xiao Li

Winter 2017

1 Introduction

The goals of our project, shapes of Julia sets, are to find a better computational algorithm for plotting Julia set and to approximate any set of disjoint Jordan curves on the complex plane by plotting the Julia set of some polynomial. What is the Julia set? Mathematically speaking, the Julia set of a polynomial is defined by $J(P) = \{z \in C : P^m(z) \not\to \infty \text{ as } m \to \infty\}$, and the Julia set is the set of complex numbers z which do not approach infinity after infinite iterations of polynomial. Suppose that we have a quadratic polynomial $f(z) = z^2 + c$, where c is a complex parameter, then we plug initial value z_1 to the polynomial and get z_2, and plug z_2 to the polynomial and get z_3...... We keep doing this process n times as $n \to \infty$, if z_n converge to infinity, then z_0 is not in the Julia set of $f(z)$. Otherwise, it's in the Julia set. As the polynomial varies, we can produce the different shapes of Julia set.

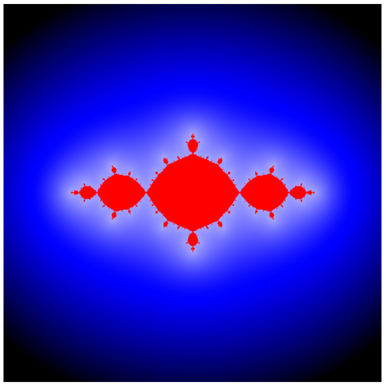

FIGURE 1. Julia Set of Polynomial $z^2 - 1$, $c = (-1, 0)$

2 Progress

2.1 Plotting Julia sets

We tried two different methods for plotting Julia sets.

2.1.1 The obvious method

The "obvious method" to determine if a complex number belongs to a Julia Set is to iterate points in the complex plane to the polynomial n times and check when $n \to \infty$, will the result converge to infinity as well.

This method is straight forward, but at the same time, the drawback of it is also easy to see: in real life, we can't iterate a point infinite times, we have to choose a certain number, for example, in our program, 100 times, and set a threshold, in our program, 10, and check after 100 times of iterations, will the magnitude of the result greater than 10. But this will bring us a new issue: there exists some number that after 100 iterations, they are still less than 10 although they actually will converge to infinity at last which means they are not part of the Julia set, but in our program, we will treat this kind of points as part of it. And this would make our plot lack of accuracy.

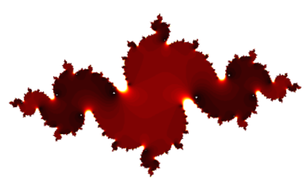

FIGURE 2. $z^2 - 0.8 + 0.156i$, $c = (-0.8, 0.156)$, by obvious method

FIGURE 3. $z^2 - 1.2 + 0.156i$, $c = (-1.2, 0.156)$, by obvious method

2.1.2 The distance estimation method

Distance estimation method (DEM) is a powerful technique for plotting Julia sets. Basically, DEM is based on various behaviors after iteration of polynomial. For each initial value of z_0 and polynomial $f(z) = z^2 + c$, we can form a sequence z_n such that $z_1 = f(z_0), z_2 = f(z_1), z_3 = f(z_2) z_n = f(z_{n-1})$, where n is the number of iteration.

The process of DEM:

The sequence z_n converges to the limit radius r, where r is a small positive real number, this means all the points of the sequence z_n are close to z_0 with a sufficiently small neighborhood. In this case, we say that z_0 is in the Julia set and we label it as 0.

The sequencez_n diverges to the limit radius r, then we compute and iterate its derivative ${z_n}' = 2 z_{n-1} {z_{n-1}}'$. If the magnitude of ${z_n}'$ is greater than or equal to defined threshold, then we say that z_0 is close to the Julia set and we label it as -1. Otherwise, we estimate the distance of z_0 by the following equation

$$d = 2 \frac{|z_n|}{|{z_n}'|} \log|z_n|$$

and label it as 1.

Then we set the different colors depend on its label.

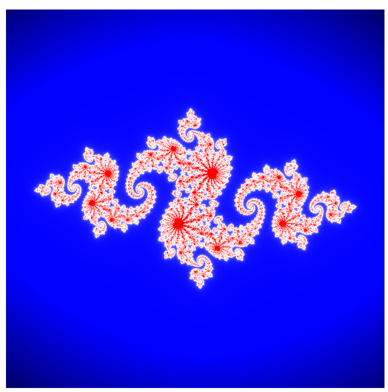

FIGURE 4. $z^2 - 0.8 + 0.156i$, $c = (-0.8, 0.156)$, by DEM

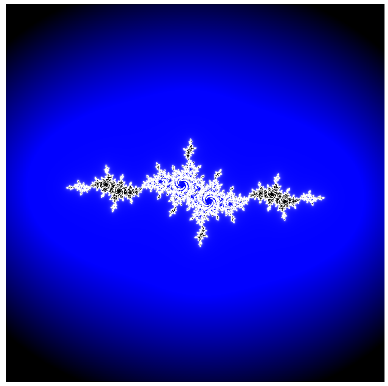

FIGURE 5. $z^2 - 1.2 + 0.156i$, $c = (-1.2, 0.156)$, by DEM

2.2 Approximation

From a paper our mentor M.Younsi co-wrote, entitled "Fekete Polynomials and Shapes of Julia Sets", we have a polynomial function:

$$P_{n,s}(z) := z\frac{e^{-ns/2}}{cap(E)^n}\prod_{j=1}^{n}(z - z_j)$$

In this function, E is the set of all points of the given Jordan curves, z_js are called leja points, and is defined as following: consider z_1 to be a random point in E, then z_2 is the point which could mamimize the value $|z_2 - z_1|$, and z_3 is the point that maximizing $|z_3 - z_1||z_3 - z_2|$, n is the number of leja points, so for z_n, we want to find the point that could maximize $|z_n - z_1||z_n - z_2|\cdots|z_n - z_{n-1}|$. $cap(E)$ is called the logarithmic capacity, it is calculated by

$$\lim_{n\to\infty}(\prod_{j=1}^{n}|z_{n+1} - z_j|)^{1/n}$$

And s is a small positive number. Here, n and s are the parameters that determine the accuracy of our approximation, the bigger the n, the smaller the s, would produce more accurate approximations, this is also make sense because when we have a bigger n, we have more leja points, so we have more understanding of the Jordan curves, hence we could produce more accurate approximations.

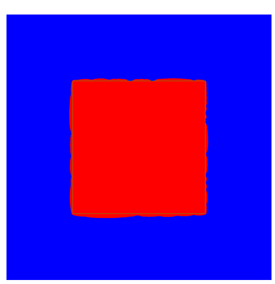

FIGURE 6. An approximation of a square with $n = 100$ and $s = 0.0100$

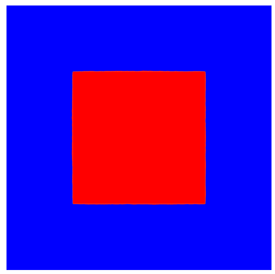

FIGURE 7.Same approximation with $n = 600$ and $s = 0.00167$

3 Future directions

3.1 Complex Shapes

We were able to successfully approximate a square; however, this square is justis just the first step, in the future, we want to approximate more complex shapes, for example, shapes with concave appearance, this require more cautious calculations and more leja points. And as mentioned earlier, we want to approximate a set of disjoint Jordan curves at the same time, but right now we can only do one. So we will keep implementing our program to make it capable to do these tasks.

Also, because of time constraint, we did all the approximations by obbious method. In the future, since we want to approximate more complex shapes and get more accurate images, we try to use distance estimation method instead.

3.2 Fractal pattern

One interesting thing about Julia Sets is that they have the fractal patterns which means when we zoom in around the edges, we should still be able to see the same shapes as the outliers appear to be. We would like to make sure that our plots will showcase this interesting effect.