COMPLEX DYNAMICS & COMPUTATION

MARLIN FIGGINS

Contents

Introduction

Complex dynamics is fundamentally the study of iterated holomorphic functions. Given a holomorphic function $f : \hat{\mathbb{C}} \to \hat{\mathbb{C}}$, we want to analyze the behavior of its iterates. For a given $n \in \mathbb{N}$, we define the n-th iterate of f as

$$f^n = \underbrace{f \circ f \circ \cdots \circ f}_{n \text{ times}}.$$

Primarily, we will study the orbit of a point $z \in \mathbb{C}$ under f, $\mathscr{O}(f, z) = \mathscr{O}(z) = \{f^n(z) \mid n \in \mathbb{N}_0\}$. The general goal of this being to understand the dynamics of f i.e. its fixed and periodic points, where points tend to under iteration, and the sensitivity of the orbits of f to different initial conditions. In particular, we will take advantage of the continuity of f and the compactness of $\hat{\mathbb{C}}$ and many of our questions will pertain to the sensitivity of f to initial conditions and the recurrence of orbits, which immediately remind of the wandering and non-wandering sets. In this way, it is clear that, to some extent, we want to be able decompose our space $\hat{\mathbb{C}}$ according to the dynamics f. For the purposes of this paper, we will mainly restrict our attention to the iteration of rational functions of the Riemann sphere.

Complex dynamics as a field gained new life by early attempts to visualize the Mandelbrot set which we will define and discuss at length later. This sparked an increased use of computation and attempts to visualize some of the natural and recurrent objects in complex dynamics. Using these methods, the natural visual

Date: March 21, 2019
Last Updated: February 27, 2020.

beauty of complex dynamics becomes even more apparent and thrilling! Some may say that it is one thing to theorize and another to compute, but complex dynamics is a clear example in which both of these coexist and inform one another to fully highlight the various complexities and simplicities of the subject. In this paper, we will attempt to highlight the interplay of visualization, computation, and theory. Starting with the classical example of Möbius transformations and rational maps on the Riemann sphere $\hat{\mathbb{C}}$ and moving onto the study of polynomials in degree two and three, we will analyze and present some of the methods and algorithms used to model and visualize complex dynamical systems as well as the theory motivating them!

1. Basic Complex Dynamics

Shifting our focus to complex dynamics, we will define some notions that will be of interest in the sections to come. Many of the fundamental questions in complex dynamics concern the qualitative behavior of orbits such as whether they are periodic, dense in some set, attracted to or repelled from a point.

The first object we will concern ourselves with is the non-wandering set.

Definition 1.1. Given a holomorphic function f, the *non-wandering set of f* $\mathrm{NW}(f)$ is the set of points for which every open set U, there exists a positive integer n such that

$$f^n(U) \cap U \neq \emptyset$$

The non-wandering set of f is closed, f-invariant. This means that each neighborhood $\hat{\mathbb{C}}$ eventually returns to itelf under the dynamics of f. This is contrasted with the wandering set, the set of points for which there is a neighborhood that never returns to itself. Since the wandering set is the complement of the non-wandering set, these two sets are provide a classification the general behavior of points since $\hat{\mathbb{C}}$ can be written as a disjoint union of these two sets.

As we will see through results such as the No-Wandering-Domain theorem (Theorem 1.25), these sets do not completely describe the kinds of behaviors we're interested in the case of complex dynamics and we will develop the ideas of the Julia and Fatou sets to deal with some of the particularities of the complex case.

To begin studying these behaviors, we're going to need to develop an understanding of the *fixed points of f* i.e. points z such that $f(z) = z$.

Definition 1.2. Given a holomorphic function f and fixed point z_0, we define the *multiplier of f at z_0* to be $|f'(z_0)|$. Moreover, we call z_0

(1) *super-attracting* if $|f'(z_0)| = 0$,
(2) *attracting* if $0 < |f'(z_0)| < 1$,
(3) *indifferent* if $|f'(z_0)| = 1$,
(4) *repelling* if $|f'(z_0)| > 1$.

The multiplier of a fixed point can provide important information about the dynamics near a point. As we will see, this multiplier is an invariant under conformal conjugacies. Much of studying dynamical systems is concerned with those properties that are preserved under conjugacy. In the case of complex dynamics, we are interested in systems that conformally conjugate and those invariants.

Definition 1.3. We say that holomorphic maps f and g are *conformally conjugate* if there is some biholomorphic map φ such that

$$f \circ \varphi = \varphi \circ g. \tag{1.4}$$

Remark 1.5. Henceforth, all conjugacies are assumed to be conformal conjugacies unless stated otherwise.

Periodic orbits (including fixed points) are preserved under conjugacy as well as other dynamical properties such as transitivity, mixing, and entropy. Therefore, we can consider it to be a sort of 'change of coordinates.' In particular, we can see that φ additionally preserves multipliers.

Proposition 1.6. *If f and g are conjugate and z_0 is a fixed point of f, then z_0 and $\varphi(z_0)$ have the same multiplier.*

Proof. This is an application of the chain rule for holomorphic functions paired with the definition of conjugacy. □

In the following section, we'll apply these ideas to classify the dynamics of degree one rational functions on the Riemann Sphere.

1.1. Möbius Transformations & the Riemann Sphere.

We begin by analyzing the Möbius transformations which are the conformal automorphisms of the Riemann Sphere.

Definition 1.7. A *Möbius Transformation* is a map $f : \hat{\mathbb{C}} \to \hat{\mathbb{C}}$ of the form

$$f(z) = \frac{az+b}{cz+d}, \tag{1.8}$$

where $a, b, c, d \in \mathbb{C}$ and $ad - bc \neq 0$.

Remark 1.9. When dealing with Möbius transformations, we adopt the following rules so that f is properly an automorphism of the Riemann sphere:

(1) If $c \neq 0$, $f(-\frac{d}{c}) = \infty$ and $f(\infty) = \frac{a}{c}$.
(2) If $c = 0$, f is linear map and we define $f(\infty) = \infty$.

For any given $\lambda \neq 0$, the Möbius transformation defined by the coefficients $\lambda a, \lambda b, \lambda c, \lambda d$ is the same as the one described by a, b, c, d i.e.

$$f(z) = \frac{(\lambda a)z + (\lambda b)}{(\lambda c)z + (\lambda d)} = \frac{az+b}{cz+d}.$$

Therefore, we can always normalize, so that $ad - bc = 1$. We will only consider such Möbius transformation in what follows. In doing this, we can relate the set of Möbius transformations with the group of matrices

$$\mathbf{PSL}(2, \mathbb{C}) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \middle| \, ad - bc = 1 \text{ where } a, b, c, d \in \mathbb{C} \right\}.$$

In fact, a quick computation shows that this set of normalized Möbius transformations (under composition) is isomorphic to $\mathbf{PSL}(2, \mathbb{C})$ (under matrix multiplication) as a group. This observation is simple, yet powerful in our study of the Möbius transformations and their dynamics.

Observation 1.10. The composition of Möbius Transformation is again a Möbius transformation.

This observation helps to elucidate our problem. If we're interested in understanding the dynamics of a particular Möbius transformation on the Riemann Sphere, it is essential to understand the group of Möbius transformations themselves. Therefore, we begin by an analysis of the fixed points of a given Möbius transformation.

Lemma 1.11. *Every non-identity Möbius transformation $f(z) = \frac{az+b}{cz+d}$ has one or two fixed points.*

Proof. In the case $c = 0$, we can reduce to the linear equation $f(z) = az + b$. Clearly, f fixes ∞, so we're left to check whether $z = az + b$ has a solution. If $a = 1$, then there is no solution. Otherwise, $\frac{b}{1-a}$ is a solution. This means f has at most 2 solutions.

Otherwise $c \neq 0$. We can attempt to solve for the fixed points directly and see that the equation

$$f(z) = \frac{az+b}{cz+d} = z$$

can be rewritten as the equation for the roots of a quadratic polynomial

$$cz^2 + (d-a)z - b = 0,$$

which must have two solutions (counting multiplicity) by the Fundamental Theorem of Algebra.

□

Using the quotient rule, we can compute the derivative of a Möbius transformation f as

$$f'(z) = \frac{ad-bc}{(cz+d)^2}. \tag{1.12}$$

Our knowledge about the derivative at a fixed point gives us information about the orbits of our dynamical system. In the case of Möbius transformations, the fixed points and their multipliers allow us to completely classify their dynamics.

Theorem 1.13. *Let f be a non-constant Möbius transformation. Then if f has single fixed point z_0, every orbit of f coverages to it. Otherwise, f has two fixed points z_1, z_2 and one of the following holds:*

(1) every orbit of f converges to one of the fixed points,
(2) every orbit is finite,
(3) or forms a dense subset of some circle.

Proof. Suppose f has exactly one fixed point at ∞. Then we can write $f(z) = z+b$ where $b \neq 0$. We see that $f^n(z) = z+nb$ and that every orbit must go off to infinity. If f has a fixed point z_0, then we can conjugate using $\varphi(z) = \frac{1}{z-z_0}$ to see that

$$g(z) = \varphi \circ f \circ \varphi^{-1}(z)$$

must fix ∞. Therefore, every orbit of g goes off to ∞. This tells us that the orbit of $f = \varphi^{-1} \circ g \circ \varphi$ must approach $\varphi^{-1}(\infty) = z_0$.

If f instead has two fixed points, let them be at 0 and ∞. Therefore, we can write $f(z) = \lambda z$. This leaves us three possibilities for the parameter λ, $|\lambda| < 1, |\lambda| > 1, |\lambda| = 1$. If $|\lambda| < 1$, then $f^n(z)$ goes to 0 as n goes to infinity. Similarly, if $|\lambda| > 1$ the orbit of f goes to ∞. If $|\lambda| = 1$, then λ is either a root of unity or not. In the former case, there is some n such that $f^n(z) = \lambda^n z = z$. In the latter, $f(z)$ acts like an irrational rotation of the circle of radius $|z|$ and therefore, its orbit is dense on that circle.

Similar to before, if f has distinct fixed points z_1 and z_2, we can find a Möbius transformation φ that maps z_1 to 0 and z_2 to ∞ for example, $\varphi(z) = \frac{z-z_1}{z-z_2}$. This finishes the proof since $g = \varphi \circ f \circ \varphi^{-1}$ fixes 0 and ∞. □

This theorem completely classifies the dynamics of rational maps of degree one by conjugating any Möbius Transformation to the map $z \mapsto z + b$ or $z \mapsto \lambda z$. As one might expect, we will find that the dynamics of higher order rational functions will not behave as nicely. Because of this, visualization may become an increasing useful aid in getting an intuition for the more complicated dynamics of higher degree rational maps.

Definition 1.14. A *rational function* is a function $f : \hat{\mathbb{C}} \to \hat{\mathbb{C}}$ of the form

$$f(z) = \frac{P(z)}{Q(z)} = \frac{a_n z^n + \cdots + a_0}{b_m z^m + \cdots + b_0}, \tag{1.15}$$

where the numerator $P(z)$ and denominator $Q(z)$ are polynomials with complex coefficients and not both identically zero.

Remark 1.16. If the polynomials P and Q above are co-prime, we define the degree of f to be maximum of the degrees of P and Q as polynomials i.e. $\max(n, m)$.

Degree is a helpful notion that we can use to describe the 'complexity' of a function or, loosely, the number of different paths or branches one can take to arrive at a given point.

Proposition 1.17. *If f is a non-constant rational function of degree $d > 0$, then f is a d-fold map from $\hat{\mathbb{C}}$. That is, every point $z \in \hat{\mathbb{C}}$ has exactly d pre-images.*

If we note that the degree of rational maps multiply under composition, we can derive the following corollary about the iterates of rational maps.

Corollary 1.18. *If f is a rational map of degree d, then f^n is a rational map of degree d^n.*

This corollary suggests that the number of periodic points of period n grows exponentially in n since the rational function $f^n(z) - z$ has the same degree as f^n, meaning that the behavior of f^n is increasingly complicated and exponentially so.

With an understanding of the most basic types of holomorphic dynamical systems, we can begin to define some of the central objects of complex dynamics.

1.2. **The Fatou and Julia Sets.**

The Julia and Fatou sets give us a way of splitting the dynamics of a map f into a set of 'niceness' on which the dynamics of f are locally tame and a set of 'chaos' on which dynamics are sensitive to initial conditions. Heuristically, we will call the Fatou set the set of points on which the iterates of f are locally well behaved and the Julia set its complement. In order to make this notion rigorous, we rely on the notion of normality as introduced by Paul Montel (1912).

Definition 1.19. A family of functions $\mathcal{F}$ on X is called *normal* if every sequence of functions in $\mathcal{F}$ has a subsequence f_n that converges to a function f locally uniformly. That is, every point $z \in X$ has a neighborhood U on which $f_n\big|_U$ converges to $f\big|_U$.

This gives us a notion of how a family of functions can be well behaved. From the point of view of dynamics, the family of functions we're interested in is the collection of iterates of f.

Definition 1.20. Given a non-constant rational function f on the Riemann Sphere, the *Fatou set* $F(f)$ is the domain on which the family $\{f^n \mid n \in \mathbb{N}\}$ is normal. We define *Julia set* $J(f)$ to be the complement of the Fatou set.

From this definition, it is clear that $F(f)$ is open and that $J(f)$ is closed. Likewise, it follows that points on the Julia set are *sensitive to initial conditions* in the following sense. Any neighborhood of a point in $J(f)$ must contain points exhibiting distinct dynamical behaviors. We will see an example of this later when we discuss the dynamics of the map $z \mapsto z^2$.

Another useful lemma for understanding the structure of Julia sets is the fact that it is invariant under the map f.

Lemma 1.21. [5, Lemma 4.4] *The Julia set $J(f)$ of any holomorphic function f is fully invariant i.e. $f^{-1}(J(f)) = J(f) = f(J(f))$.*

With this knowledge, it is clear that the Julia and Fatou sets give an f-invariant decomposition of $\hat{\mathbb{C}}$ into two sets of points, one of which is sensitive to initial conditions and another which exhibits 'tame' local dynamics under f.

With this in mind, we can also relate the Julia set with the non-wandering set which was our prototype for dealing with the type of recurrence and sensitivity expected by the Julia set.

When dealing with polynomials, we will often try to compute the filled Julia set of a map f instead of just its Julia sets.

Definition 1.22. If f is a monic polynomial. the *filled Jula set* $K(f)$ is the set of points z for which the orbit $\mathscr{O}(f)$ is bounded.

This will be useful in the sense that the filled Julia set $K(f)$ is precisely the complement of the basin of attraction to ∞ for a given monic polynomial f.

Returning the idea of the non-wandering set discussed earlier, we can derive the following relationship between the Julia set and the non-wandering set.

Proposition 1.23. *The Julia set $J(f)$ is contained in the* $\mathrm{NW}(f)$.

It turns out that the reverse inclusion might not hold, the two sets are not necessarily equal. The picture is a bit more complicated as Problem 19-A of Milnor [5] shows.

Theorem 1.24. [5, Problem 19-A] *For a rational function $f\colon \hat{\mathbb{C}} \to \hat{\mathbb{C}}$, the non-wandering set* $\mathrm{NW}(f)$ *is the disjoint union of its Julia set, its attracting periodic points, and rotation domains.*

This tells us that the non-wandering set is a useful aid in our intuition and understanding the dynamics of f and the Julia set, but does not characterize it entirely.

A similar statement for the Fatou set was proved by Dennis Sullivan in 1985.

Theorem 1.25 (No-Wandering-Domain Theorem)**.** [6] *Any rational map $f\colon \hat{\mathbb{C}} \to \hat{\mathbb{C}}$ of degree greater than two has no wandering domain.*

This leads us to study the Julia and Fatou sets as separate entities entirely, which better characterize our ideas of 'locally' tame dynamics. As of now, we have defined and worked with some of the properties of the Julia set, but some natural questions are 'what do the dynamics on the Julia set actually look like?' and 'what is the the actual form of the Julia set?'. To approach these questions, we're going to rely on some computational methods.

1.3. Computing the Julia Set.

Knowing the shape of a given holomorphic function's (filled) Julia set can help gain insight into a given functions dynamics. In the case of polynomials, the filled-in Julia set gives us a way of distinguishing the points whose orbits remain bounded, localizing their dynamics. Knowing the actual shape of this set as well as its different connected components gives us an idea of how bounded orbits of points may permute or between the components of the Fatou set.

Corollary 1.26. [5, Corollary 4.13] *For any given z in the Julia set $J(f)$,*

$$\bigcup_{n=1}^{\infty} f^{-n}\{z\}$$

is dense in $J(f)$

Simply computing the Julia set by taking pre-images of a specific point is naïve in the sense that for any non-constant rational map f of degree d taking n pre-images of a point z leaves you to compute $d^n = |f^{-n}\{z\}|$ points of your Julia set. For example, if f is only of degree 2, after taking only 40 pre-images you're left to compute more than a trillion points! This further becomes an issue because there is no promise of how many iterates it will take to get close to any particular point in $J(f)$. Though there are methods adapting this idea to make it more computationally efficient, we will explore a diffrent route for computing the Julia set.

Another way to do this may be to track whether or not a given orbit of a point diverges to ∞ under a rational map f. For computational efficiency, we will instead check if the orbit escapes some bailout (or escape) radius b and find the number of iterates it takes for a given point to escape. In Python, we can implement this process as follows:

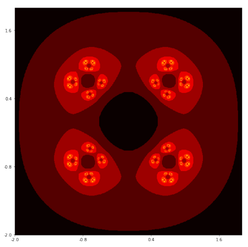
```
def julia(f, z, maxiter):  # Iterate f(z) until it escapes
    for n in range(maxiter):
        if abs(z) > 4:  # If f^n(z) escapes, return n.
            return n
        z = f(z)
    return 0
```

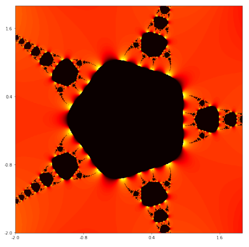
We can then apply this function to a range of pixels using the following code snippet and generate an image of the Julia set in a particular region of the plane.

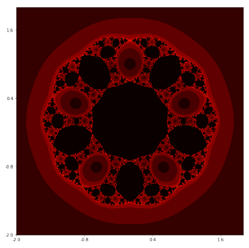
```
def julia_set(f, width=10, height=10, maxiter=200, xmin=-2, xmax=2, ymin=-1, ymax=1):

    x = np.linspace(xmin, xmax, width)   # Generate x values
    y = np.linspace(ymin, ymax, height)  # Generate y values
    escape = np.empty((width, height))

    for i in range(width):  # Loop over each z = x + yi ∈ ℂ.
```

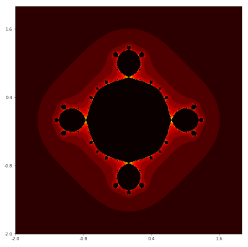
```
        for j in range(height):
            escape[i, j] = julia(f, z=x[i] + 1j*y[j], maxiter)
    return (x, y, escape)
```

This algorithm allows us to produce pictures of the Julia set for any given complex function f. Examples of such pictures are shown in Fig. 1.


(A) $z \mapsto z^2 + \frac{1}{z^2}$


(B) $z \mapsto \frac{z^5-4}{z^4}$


(C) $z \mapsto \frac{iz^5-1}{z^5-i}$


(D) $z \mapsto \frac{3}{4}z^4 - 1$

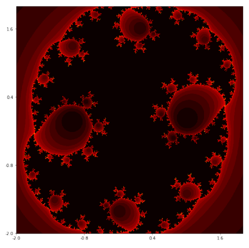

(E) $z \mapsto \frac{z^2+i}{1-z^2}$

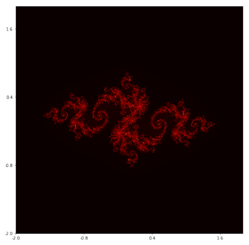
FIGURE 1. Example Julia sets computed with above code.

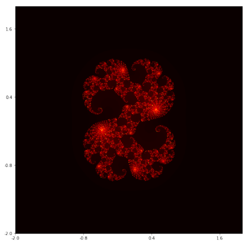
As one can see, these Julia sets can be quite complicated. This is reflected by the run time of our code which takes on average 5.95 seconds $\pm$ 331 milliseconds[1] to generate the figures like those in Fig. 1. One main cause of this being that the julia function above takes an arbitrary function f as argument which, then, must be applied repeatedly. This can lead to quite a bit of slow-down which we can alleviate by hard coding in the function we'd like to iterate with. In particular, we can replace the julia function from the code snippets above to adapt it to compute the Julia sets for the family of functions of the form $f_c(z) = z^2 + c$ where c is any complex number. We will discuss this dynamics of this family in more detail in later sections of this paper, but for now, we can use our above Julia function with the following edits to compute the Julia sets for members of this family:

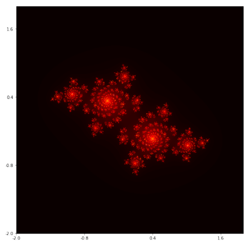
```
def julia(c, z, maxiter):  # Iterate $f_c(z)$ until it escapes
    for n in range(maxiter):
        if abs(z) > 4:  # If $f_c^n(z)$ escapes, return $n$.
            return n
        z = z*z + c
    return 0
```

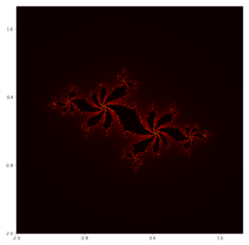
[1]All timing benchmarks in this paper are taken as a time average of 7 runs and computed using a 720×720 grid of pixels and a maximum of 500 iterates.

Examples of these Julia sets can be seen in Fig. 2. The decision to take in the complex value c as an argument instead of an arbitrary function f leads to significant speed-up in computation, each run taking 29.1 milliseconds $\pm$ 763 microseconds on average.


(A) $c = -0.79 + 0.15i$


(B) $c = 0.28 + 0.009i$


(C) $c = -0.44 + 0.59i$


(D) $c = -0.62 + 0.437i$

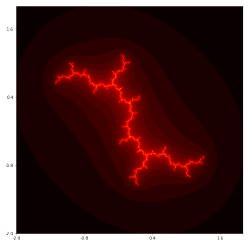

(E) $c = i$

FIGURE 2. Examples of Julia sets of $f_c(z) = z^2 + c$ for certain c values.

We can see there's a variety of structures in the Julia sets produced by this algorithm. Most of them seem to be fractal-like. Some are connected while others appear to be specks of what looks like 'dust', we will see later that connectedness of the Julia set is related to the Mandelbrot set. We will see these kinds of connections as we shift our study to the polynomials and later, the family of polynomials $f_c(z) = z^2 + c$ in the next sections.

2. Iterating Polynomials

We will begin our study of iterated complex polynomials with the most simple case of degree two polynomials. For simplicity's sake, we will start by studying $z \mapsto z^2$.

2.1. The case of $z \mapsto z^2 + c$.

Consider the squaring map $f\colon z \mapsto z^2$. For this map given any $n \in \mathbb{N}$, we can write an explicit formula for the n-th iterate of f

$$f^n(z) = z^{2n}.$$

Using this formula, we can easily deduce the dynamics of f off the unit circle.

Proposition 2.1. *Under $f(z) = z^2$, every point that is not on the unit circle is attracted to either 0 or infinity. Moreover, the Julia set $J(f)$ is the unit circle $S^1 = \{z \mid |z| = 1\}$.*

The dynamics on the Julia set S^1 of $z \mapsto z^2$ can be related to a familiar and well-studied dynamical system. We can present each point z of S^1 uniquely as $e^{2\pi i\theta}$

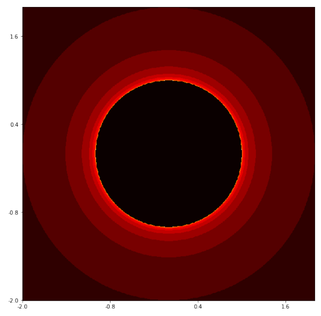


FIGURE 3. The filled Julia set of $z \mapsto z^2$.

for some $\theta \in [0, 2\pi)$. We can see that $z \mapsto z^2$ acts as $e^{2\pi i\theta} \mapsto e^{2\pi i(2\theta)}$ in S^1. This is clearly the doubling map on $S^1 = \mathbb{R}/\mathbb{Z}$ which we know to be sensitive to initial conditions. We can see that S^1 serves as the boundary for two different dynamics behaviors. Any neighborhood of a point on S^1 contains a connected component inside the unit circle which is attracted to 0 and another outside the unit circle which is attracted to ∞. This tell us that the filled Julia set $K(f)$ is precisely the closed unit disk $\overline{\mathbb{D}} = \{z \mid |z| \leq 0\}$.

Since we're interested in studying other degree two polynomials, we will shift our attention to the family of polynomials $f_c(z) = z^2 + c$ where c is any complex number. Unlike the case where $c = 0$, the dynamics of f_c are often not as simple and the Julia sets not as easily understood. Different values of c can cause f_c to have drastically different dynamics. To see this consider $c = -1, 1, i$, we can write the orbits of 0 under each f_c as sequences $(z_n)_{n=1}^{\infty} = f_c^n(0)$:

$$(z_n) = -1, 0, -1, 0, \ldots \quad \text{when } c = -1, \tag{2.2}$$

$$(z_n) = 1, 2, 5, 26, 677 \ldots \text{ when } c = 1, \tag{2.3}$$

$$(z_n) = i, -1 + i, -2i, \ldots \text{ when } c = i. \tag{2.4}$$

From this, we can see that various types of behavior are possible for different values of c. More generally one might ask what are all of the possible behaviors for the sequences $(z_n)_{n=1}^{\infty}$ defined by

$$0, z_1 = f_c(0), z_2 = f(z_1), z_3 = f(z_2), \ldots$$

Another aspect of this question is asking what happens to our family of functions $\mathcal{F}_c = \{f_c \mid c \in \mathbb{C}\}$ as we move through the parameter space of c *continuously*. Already, we can get some elementary estimates of the behavior of $z \mapsto z^2 + c$ outside a neighborhood of 0.

Proposition 2.5 (Bailout Radius for $z \mapsto z^2 + c$)**.** *For any $c \in \mathbb{C}$, if the f_c-orbit of 0 escapes the circle of radius 2, then it goes off to infinity.*

Proof. Consider the case where $|c| < 2$. Suppose $|z_n| = 2 + \epsilon$ for some n and ϵ. Then $|z_{n+1}| = |z_n^2 + c| \geq |z_n|^2 - |c| > 2 + 2\epsilon + \epsilon^2$. Repeating this argument on z_{n+1} shows the modulus must increase with every iterate, going off to infinity.

In the case $|c| > 2$, we know that $|f_c(0)| = 2 + \epsilon$. An argument similar to above shows that $|z_2| > 2 + \epsilon$ and the claim follows by induction. $\square$

2.2. **The Mandelbrot Set.**
We can then reframe our question about the orbits of 0 as follows:

Question 2.6. *For which $c \in \mathbb{C}$, does the f_c-orbit of 0 stay bounded?*

In other words, we will be finding the set of points comprising the famous *Mandelbrot set*

$$\mathcal{M} = \{c \in \mathbb{C} \mid (f_c^n(0))_{n=1}^{\infty} \text{ is bounded}\}. \tag{2.7}$$

Our earlier discussion shows that the point 1 is not in the Mandelbrot set. Additionally, Proposition 2.5 tells us that $\mathcal{M}$ is completely contained in the circle of radius two. We can also use analytic methods to find sets of points in the Mandelbrot set. By assuming a given f_c satisfies certain dynamical behavior and solving for c, we can compute sections of the Mandelbrot sets.

Example 2.8. [1, Section 1.6.] One example of a question one might ask is "for which c does f_c have an attracting fixed point in $\mathbb{C}$"?. Suppose f_c has two fixed points ζ_1, ζ_2 corresponding to the solutions of

$$f_c(z) - z = z^2 - z + c.$$

With this knowledge, we can factor and write $f_c(z) - z = (z - \zeta_1)(z - \zeta_2)$. A quick computations shows that $\zeta_1\zeta_2 = c$ and $\zeta_1 + \zeta_2 = 1$. In particular, we can take the derivative of f_c, $(f_c)'(z) = 2z$, and see that

$$(f_c)'(\zeta_1) + (f_c)'(\zeta_2) = 2(\zeta_1 + \zeta_2) = 2.$$

Considering this equality alongside Definition 1.2, it is clear that f can have at most one attracting fixed point. If we have such a fixed point ζ_*, then $|\zeta_*| < \frac{1}{2}$. Since ζ_* is a fixed point of f_c, we also have that $\zeta_* - \zeta_*^2 = c$. Therefore, we can write the set of parameters c for which f_c has an attracting fixed point as

$$C = \left\{ c = \zeta - \zeta^2 \in \mathbb{C} \,\middle|\, |\zeta| < \frac{1}{2} \right\}.$$

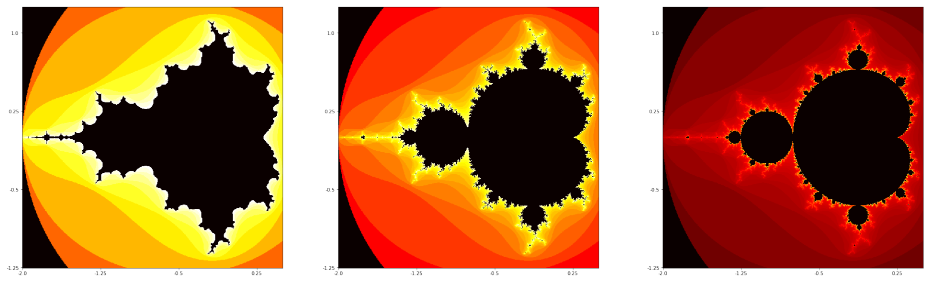
This is clearly a subset of the Mandelbrot set $\mathcal{M}$ since for every $c \in C$, the orbit of 0 under f_c approaches the attracting fixed point. As we will see this set comprises a cardioid which makes up 'the main body' of the Mandelbrot set.

After a certain point, this type of analysis becomes difficult. One, with a bit of trouble, can compute the set of parameters for which f_c has an attracting 2-cycle as $C_2 = \{c \mid |1 + c| < \frac{1}{4}\}$. Since the argument used to show this relies on the number of roots of $\frac{f_(^2 z) - z}{f(z) - z}$ being 2, using a natural generalization of this argument to directly compute C_n the set of parameters giving an n-cycle becomes difficult for $n \geq 3$. This is because the number of roots of the analogous polynomial $\frac{f_(^n z) - z}{f(z) - z}$ grows exponentially in n by Corollary 1.18. Because of this, we need to find other ways for describing and understanding the structure of the Mandelbrot set and its relation with the dynamics of the family of polynomials $\{f_c\}_{c \in \mathbb{C}}$.

One such way is to attempt visualize it using computational methods. In what follows, we will present two ways of computing the Mandelbrot set $\mathcal{M}$. The first is very similar to the algorithm for the Julia set of f_c presented before. The main difference being that instead of looping through different initial conditions z, we will be looping through parameter values c counting the number of iterates it takes to escape the circle of radius 2 and go off to infinity as per Proposition 2.5.

```
def mandelbrot_d(z, maxiter, bailout):  # Iterate $f_c(0)$ until it escapes.
    c = z
    for n in range(maxiter):
        if abs(z) > 2:  # If $|z| > 2$, orbit escapes to $\infty$.
            return n
        z = z*z + c
    return 0


def mandelbrot_dset(xmin, xmax, ymin, ymax, width, height, maxiter):

    x = np.linspace(xmin, xmax, width)
    y = np.linspace(ymin, ymax, height)
    escape = np.empty((width, height))

    for i in range(width):  # Loop over each $c = x + yi$.
        for j in range(height):
            escape[i, j] = mandelbrot_d(x[i] + 1j*y[j], maxiter)
    return (x, y, escape)
```

This algorithm allows us to produce pictures like Fig. 4. One thing that is apparent here is that the number of iterates is extremely important and using too few can lead to deceiving images. Even with these flaws, this algorithm is quite fast, taking 354 ± 2.29 milliseconds per loop on average, though one big downside is the obvious color banding caused by our usage of discrete values in coloring our images.



(A) Using 10 Iterates (B) Using 30 Iterates (C) Using 500 Iterates

FIGURE 4. Naïve attempts to visualize the Mandelbrot set can be deceiving.

The second algorithm relies on a continuous estimate of iterate count [4] used in computing the Mandelbrot set. The main difference being instead of the returning the integer iteration count n, we return

$$\eta = n + \frac{\log\log b - \log\log|z_n|}{\log 2} \quad \text{where } b \text{ is the bailout radius} \tag{2.9}$$

which gives us continuous values for our escape iterate! This equation is actually heavily related to potential theory as will see below.

In order to see the origin of (2.9), we will need the following theorem (originally by Böttcher in 1904).

Theorem 2.10. [5, Theorem 9.1] *Given a monic polynomial f of degree n, there exists a biholomorphic map φ which fixes 0 and conjugates f and z^n.*

Considering again the case of $c = 0$, we recall that the filled Julia set of f is given by the closed unit disk $\overline{\mathbb{D}}$. On the Fatou component containing ∞, $\hat{\mathbb{C}} \setminus \overline{\mathbb{D}}$, we can define the *potential function* of f_0 as

$$\psi(z) = \lim_{n\to\infty} \frac{\log|z^n|}{2^n}. \tag{2.11}$$

Using theorem of Böttcher, we can conjugate the dynamics of $z^2 + c$ to z^2 on the complements of their respective filled Julia sets.

Theorem 2.12 (Böttcher Coordinates)**.** *There exists a biholomorphic map φ such that f_c is conjugate to f_0 in a neighborhood of infinity.*

Remark 2.13. In fact, for $c \in \mathcal{M}$, our conjugacy φ can be extended to the domains $\hat{\mathbb{C}} \setminus \overline{\mathbb{D}}$ and $\hat{\mathbb{C}} \setminus K(f_c)$. This extension can be used to give another proof that the Mandelbrot set is connected.

Proof. Making the substitutions $w = \frac{1}{z}$ and $g(w) = \frac{1}{f(1/w)}$, we can conjugate the point ∞ to 0. Using Theorem 2.10, we can get a conjugacy between w^2 and $g(w)$ near zero. This corresponds to a conjugacy between z^2 and f_c near ∞. $\square$

Using that $|\varphi(f_c^n(z))| = |\varphi(z)|^{2n}$ for any large enough n, the existence of this conjugacy φ tells us that our potential function ψ is the same for every f_c though on a different domain. Therefore, for some large b, if we take the first iterate n such that $f_c^n(0) > b$, then we can find some real number η such that

$$\frac{\log|f_c^n(0)|}{2^n} = \frac{\log b}{2^\eta}. \tag{2.14}$$

Rearranging the above equation and solving for η gives us the estimate used in (2.9).

The caveat of this method is that we must pick much larger bailout radius b, so that $z^2 + c$ can begin to resemble z^2 'near infinity' in the sense of Böttcher allowing us to use our estimate. The modification of the code to include the estimate given by (2.9) can be seen in the following snippet:

```
def mandelbrot_c(z, maxiter, bailout, bailout_est):  # Iterate $f_c(0)$ until it escapes.
    c = z
    for n in range(maxiter):
        absz = abs(z)
        if absz > bailout:  # If $|z| > 2$, orbit escapes to $\infty$
            # return continuous estimate of escape iterate $n - \frac{\log\log|z_n|}{\log 2} + b_{est}$.
            return n - np.log(np.log(absz))/np.log(2) + bailout_est
        z = z*z + c
    return 0


def mandelbrot_c_set(xmin, xmax, ymin, ymax, width, height, maxiter):
```

```
bailout = 2**30
bailout_est = np.log(np.log(bailout))/np.log(2)  # $b_{est} = \frac{\log\log b}{\log 2}$

x = np.linspace(xmin, xmax, width)
y = np.linspace(ymin, ymax, height)
escape = np.empty((width, height))

for i in range(width):  # Loop over each $c = x + yi$.
    for j in range(height):
        escape[i, j] = mandelbrot_c(x[i] + 1j*y[j], maxiter, bailout, bailout_est)
return (x, y, escape)
```

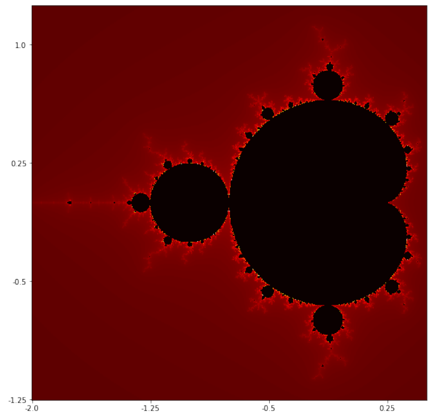
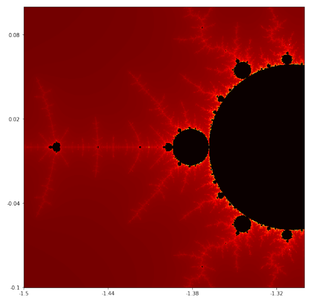


FIGURE 5. Mandelbrot Set Using Normalized Iteration Count

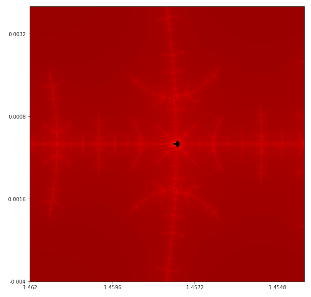
This Normalized Iteration Count algorithm produces a much 'smoother' image and allows us to see many more details of the Mandelbrot set. In particular, points for which escape takes a large number of iterates are highlighted as a lighter color and are often close to $\mathcal{M}$ itself. This feature of our continuous estimate allows us to visualize finer details of the Mandelbrot set and enables us to zoom in more confidently, though we may need to increase the maximum number of iterates to faithfully depict those c for which the orbit of 0 escapes slowly under f_c. This added clarity does not come at much cost, with this algorithm taking 383 ± 5.69 milliseconds per loop to execute on average.

Zooming allows us to see some of the (quasi-)self-similarity that $\mathcal{M}$ displays, characteristic of its fractal nature. In Fig. 6, we see that zooming in on the small island near $c = -1.457$ reveals a pattern resembling the larger set $\mathcal{M}$. This self-similarity is not limited to the appearance of mini Mandelbrot sets, but we can actually see similarities to the Julia sets of a given polynomial $z^2 + c$ and the neighborhood of c near the Mandelbrot set.

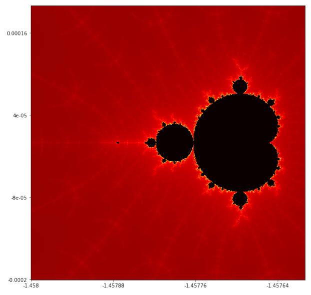
Another relationship between the Mandelbrot set and the Julia set is that for any parameter $c \in \mathcal{M}$ corresponds to f_c having a connected Julia set $J(f_c)$ i.e. $\mathcal{M}$ is the connectivity locus for the family f_c. Similarly, when $c \notin \mathcal{M}$, the the Julia



(A) Zooming onto left side.



(B) Zooming into island in middle.



(C) Zooming reveals a mini $\mathcal{M}$!

FIGURE 6. The Normalized Iteration Count algorithm allows us the clarity needed to view finer features of the Mandelbrot set.

set $J(f_c)$ is a Cantor set. Though we will not prove these statements one can see examples of this phenomenon using the various code snippets and figures above. These observations have proved to be helpful in efforts to approach the following open question.

Conjecture 2.15 (MLC Conjecture)**.** *The Mandelbrot set $\mathcal{M}$ is locally connected.*

The work of Jean-Christophe Yoccoz towards this conjecture, proving that the Mandelbrot set is locally connected at finitely renormalizable $c \in \mathcal{M}$, helps to reveal a more general relation between the Mandelbrot and Julia sets. By understanding the dynamical properties of f_c via its Julia set (establishing local connectivity of $J(f_c)$) one can move to parameter space $\mathcal{M}$ studying the properties of the Mandelbrot set itself (local connectivity at c).

We can see an application of this idea in the proof of the following theorem.

Theorem 2.16. *The Mandelbrot set $\mathcal{M}$ is compact and simply connected.*

Proof. By Proposition 2.5, $\mathcal{M}$ is precisely the set $\{c \mid |f_c^n(0)| \leq 2 \text{ for all } n \geq 0\}$. We can then decompose $\mathcal{M}$ into intersection of the sets $\mathcal{M}_n = \{c \in \mathbb{C} \mid |f_c^n(0)| \leq 2\}$. Taking any sequence of parameters $(c_i)_{i=1}^{\infty}$ in $\mathcal{M}_n$, we see that $\left|f_{c_i}^n(0)\right|$ must converge to $|f_c^n(0)|$ since f_c is continuous in c. Since the sequence $\left|f_{c_i}^n(0)\right|$ is bounded by 2, $|f_c^n(0)| \leq 2$. Therefore, each $\mathcal{M}_n$ is closed, and likewise $\mathcal{M} = \bigcap_{n=1}^{\infty} \mathcal{M}_n$ must be closed. The Mandelbrot set $\mathcal{M}$ is additionally compact since it is contained in the circle of radius 2 centered at the origin and thereby bounded.

The proof of the second statement follows theorem 8.6.1 of [2]. Suppose that $\mathbb{C} \setminus \mathcal{M}$ contains some bounded, connected component A. Since $A \subset \mathbb{C} \setminus \mathcal{M}$, we can there is some n such that

$$\sup_{c \in \overline{A}} f_c^n(0) > 2.$$

Therefore, by the maximum modulus principle, there must be some parameter γ on the boundary of A such that $f_\gamma^n(0) > 2$. Since the boundary of A must be contained in $\mathcal{M}$ because it is closed, this gives a contradiction. Therefore, there cannot be any bounded component of $\mathbb{C} \setminus \mathcal{M}$, so $\mathbb{C} \setminus \mathcal{M}$ consists of one unbounded connected component containing ∞ and is connected itself. Therefore, its complement $\mathcal{M}$ must be simply connected. □

With these tools and properties in mind, we're now in good shape to try and tackle the case of higher degree polynomials.

2.3. **Higher Degree Polynomials & the Shift Locus.**

We might try to generalize the methods in the previous section to compute one complex-dimensional slices of the shift locus.

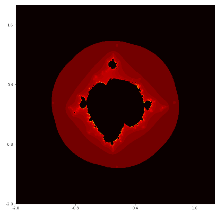
Definition 2.17. [3] Let Poly_d denote the space of monic degree d polynomials. We define the *shift locus in dimension* d to be

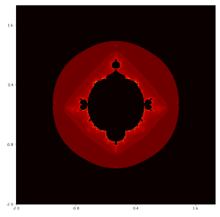
$$\mathcal{S}_d = \{f \in \mathrm{Poly}_d \mid |f^n(z_*)| \to \infty \text{ for every } z_* \in \mathbb{C} \text{ such that } f'(z_*) = 0\}. \tag{2.18}$$

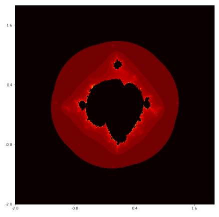
The first thing to notice is that if we identify Poly_2 with $\mathbb{C}$ via $c \leftrightarrow z^2 + c$, then we see that $\mathcal{M} = \mathbb{C} \setminus \mathcal{S}_2$ since 0 is the only critical points of $z^2 + c$. From this, we see that the Mandelbrot set is the complement of the shift locus in two dimensions. Keeping this in mind, our next question may be about how our methods in computing and understand the Mandelbrot set might generalize. In d dimensions, just finding the critical points of a given polynomial requires computing the roots of a $(d-1)$ degree polynomial. In higher dimensions, solving this equation and finding these critical points and checking the boundedness of its orbits becomes increasingly difficult from a computational standpoint. Luckily, in the case of $d = 3$, the problem is still easily tractable due to the quadratic formula.

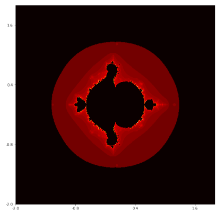
In what follows, we're going to restrict to the normal family of degree 3 polynomials given by $f_{a,b}(z) = z^3 - 3a^2 z + b$. We can identify this space with $\mathbb{C}^2$ and use this to generate one complex-dimensional slices of $\mathcal{S}_3$. Iterating this family of polynomials at its critical points $\pm a$ can be done with the following code:

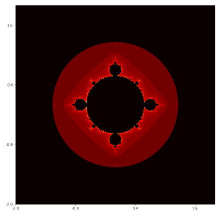
```
def Shift_Locus_3(a, b, maxiter=500, bailout=4):  # Shift Locus in case d = 3

    Twoa3 = 2*a**3  # 2a^3
    Threea2 = 3*a**2  # 3a^2

    z1 = Twoa3+b  # first iterate starting from a
    z2 = -1*Twoa3 + b  # first iterate starting from -a

    for n in range(maxiter):
```

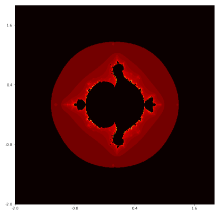

(A) $b = -0.2 + 0.2i$

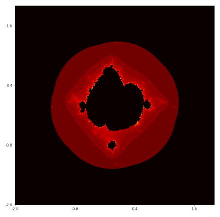

(B) $b = 0.2i$

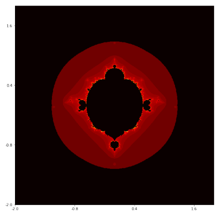

(C) $b = 0.2 + 0.2i$


(D) $b = -0.2$


(E) $b = 0$


(F) $b = 0.2$


(G) $b = -0.2 - 0.2i$


(H) $b = -0.2$

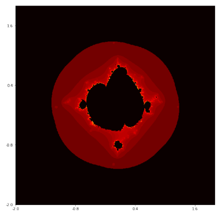
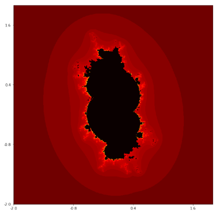
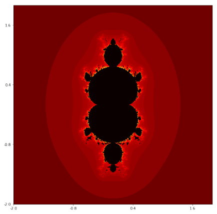

(I) $b = -0.2 + 0.2i$

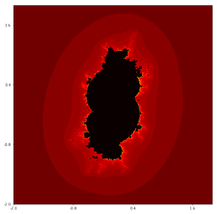
FIGURE 7. a-Parameter Space using $z \mapsto z^3 - a^2 z + b$

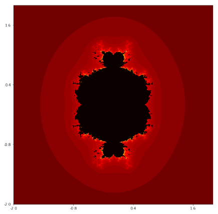
```
    absz1 = abs(z1)
    absz2 = abs(z2)

    if min(absz1, absz2) > bailout:  # If both z_1, z_2 escape bailout radius
        return n
    # Otherwise, iterate again using f(z) = z^3 - 3a^2z + b.
    z1 = z1**3-z1*Threea2 + b
    z2 = z2**3-3*z2*Threea2 + b

return 0
```

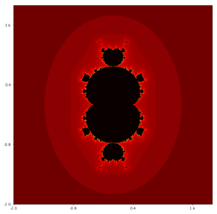
This code allows us to produce slices of $\mathcal{S}_3$ in either the a parameter space for a fixed b or in b parameter space for a fixed a by finding points which are in the complement of the shift locus $\mathcal{S}_3$. Either or is easily implemented similarly to the Mandelbrot or Julia set code presented early. Using this algorithm, we can produces images like those of Fig. 8 and Fig. 7. In a way, this is the most naïve way of computing the shift locus, borrowing heavily from our discrete escape iterate count algorithm for computing the Mandelbrot set. The naïvety is reflected in the execution time with it taking 2.88 seconds ± 158 milliseconds to generate a slice of the a-parameter space, and 4.73 seconds ± 116 milliseconds for b-parameter space on average.

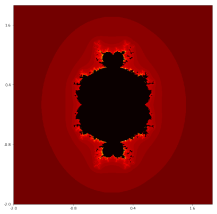

(A) $a = -0.2 + 0.2i$

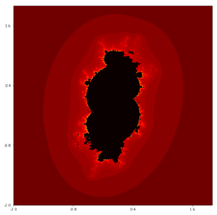

(B) $a = 0.2i$

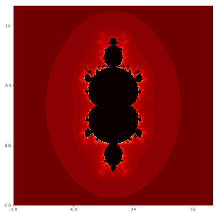

(C) $a = 0.2 + 0.2i$


(D) $a = -0.2$


(E) $a = 0$


(F) $a = 0.2$


(G) $a = -0.2 - 0.2i$


(H) $a = -0.2$

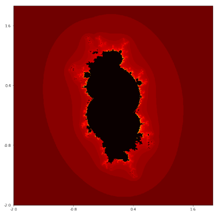

(I) $a = -0.2 + 0.2i$

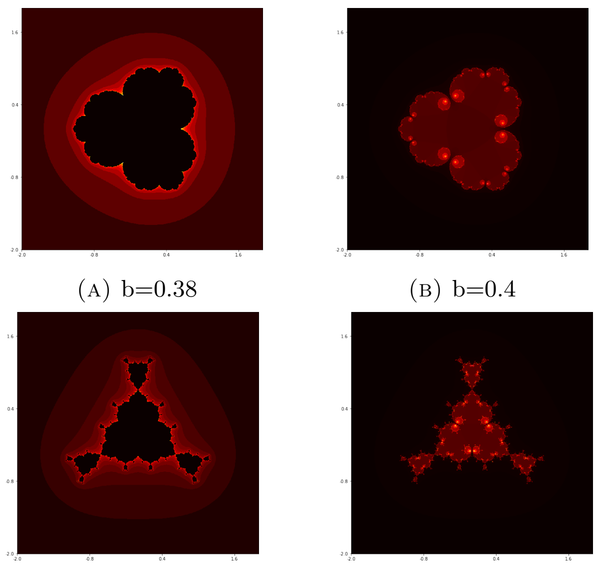
FIGURE 8. b-Parameter Space using $z \mapsto z^3 - a^2z + b$

In attempting to generalize this code to higher dimensions, calculating the critical points becomes an increasingly demanding computationally.

One might ask whether there is a similar relationship between the Julia sets of the family $z^3 - a^2z + b$ and $\mathbb{C}^2 \setminus \mathcal{S}_3$, similar to one between $z^2 + c$ and $\mathcal{M}$. We can make simple modifications to the julia code we presented in Section 1.3 to adjust it to the family of functions above:

```
def julia(a, b, z, maxiter):  # Iterate $f_{a,b}(z) = z^3 - a^2z + b$ until it escapes
    Threea2 = 3*a**2  # Pre-allocate $3a^2$.
    for n in range(maxiter):
        if abs(z) > 4:  # If $f_{a,b}(z)$ escapes, return $n$.
            return n
        z = z**3 - Threea2*z + b
    return 0
```

For reference, this algorithm takes 154 ± 2.12 milliseconds on average to execute. Using this updated julia function to compute the Julia sets for the above family may help provide some preliminary visual intuition for the relationship between the complement of the shift locus and the Julia set before attempting to investigate the possibility of such similarities using more advanced methods, may they be computational or otherwise mathematical in nature.



(A) b=0.38 (B) b=0.4

(C) b=1.08i (D) b=1.09i

FIGURE 9. Julia Sets for $f_{0,b}$ near the boundary of $\mathcal{S}_3$.

With this idea in mind, we may recall that the Mandelbrot set $\mathbb{C} \setminus \mathcal{S}_2$ is the locus of connectivity for the family of polynomials $z^2 + c$. Partially motivated by the results of our computations in Fig. 8 and Fig. 9, we end on the following question.

Question 2.19. *For which pairs of parameters $(a, b) \in \mathbb{C}^2$ are the Julia sets $J(f_{a,b})$ connected.*

Influenced by our earlier computational methods and the accompanying images, our intuition tells us that this set may be the complement of the shift locus $\mathbb{C}^2 \setminus \mathcal{S}_3$ though proving this would require additional analysis. In this way, we see that computation can help to provide visual and numerical aid in understanding mathematical problems, especially so in the case of complex dynamics.

References

[1] A.F. Beardon, *Iteration of Rational Functions: Complex Analytic Dynamical Systems*, Graduate Texts in Mathematics, Springer New York, 2000.

[2] Michael Brin and Garrett Stuck, *Introduction to Dynamical Systems*, Cambridge University Press, Cambridge, 2002.

[3] Laura De Marco, *Combinatorics and topology of the shift locus* (2011).

[4] Francisco Garcia, Angel Fernandez, Javier Barrallo, and Luis Martin, *Coloring Dynamical Systems in the Complex Plane* (2009).

[5] John Milnor, *Dynamics in One Complex Variable. (AM-160): (AM-160) - Third Edition*, Annals of Mathematics Studies, Princeton University Press, 2011.

[6] Dennis Sullivan, *Quasiconformal homeomorphisms and dynamics II: Structural stability implies hyperbolicity for Kleinian groups*, Acta mathematica **155** (1985), no. 1, 243–260.