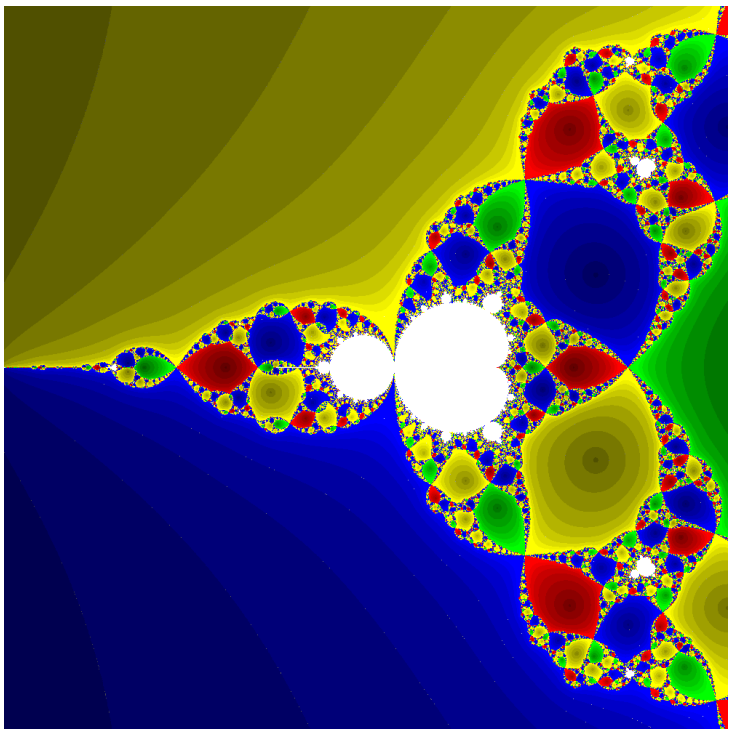
A Computational Exploration of Newton's Iteration Function for a Complex Polynomial



Julia Jansson
julia.jansson@gmail.com
Hvitfeldtska gymnasiet
Natural Science Program N3EF
19990305

under the direction of
Jenny Homann

17 mars 2017

Abstract

The focus of this report is to explore the behaviour of the Newton root-finding method on functions in the complex plane. An algorithm was designed to study the family of functions $f_a(z) = (z^2 - 1)(z^2 + a)$ where the Newton root-finding method does not converge for all initial guesses. The efficiency of Newton's method, defined as the number of iterations needed to find a root, is illustrated graphically. Since iteration of complex polynomials often gives rise to chaos, the iteration plots are fractals.

Sammanfattning

Rapporten behandlar effektiviteten av Newtons rotlösarfunktion på funktioner i det komplexa talplanet. Ett program har utvecklats för att studera hur effektiv Newtons metod är för en specifik familj av funktioner, $f_a(z) = (z^2-1)(z^2+a)$. Områden där metoden inte konvergerar har undersökts. Effektiviteten, definierad som antalet iterationer som behövs för att hitta lösningen, presenteras grafiskt. Eftersom iteration av komplexa polynom ofta ger upphov till kaos, bildas fraktaler.

Contents

1 Introduction

The butterfly effect shows us that even a flicker of a small pair of wings can lead to a hurricane. This is because small differences in initial conditions are repeated numerous times and get enlarged into something chaotic. Chaotic processes appear to be random, but they are not. Actually, chaotic systems are determined entirely by the initial conditions but they are not predictable (Lorenz, 1995).

The butterfly effect is an example of a non-linear deterministic process. It is deterministic because the future is affected by the present, and non-linear because its result is not proportional to the input. Many systems' evolution can be defined as the iteration of a non-linear function, which means repeating the non-linear function over and over. Examples of these systems are population ecology, stock-market economy and fluid dynamics (Devaney, 1992).

Likewise, chaotic behaviour occurs when a mathematical function is iterated. When a function is iterated there are certain points in the plane that are fixed points x_0 where $f(x_0) = x_0$. This means that applying the function on the point gives the same point as result. Numerical root-finding methods (equation solving) are also based on iteration. Using an initial guess, an iteration will produce a sequence of numbers that give better and better approximation of the root of the function. This is usually done by evaluating a function that is chosen for having the roots of the equation as fixed points.

In this study, fixed points will be found for the Newton iteration function, which is an efficient root-finding method (Newton, 1736). At the initial guess, which is hopefully close to the root, the function is approximated by its tangent line. The intersection of this tangent line with the x-axis is often a slightly better approximation to the root than the initial guess. The method can be iterated until a sufficiently accurate approximation of the root is obtained. Newton's method is widely used in all fields of science, since it is fast (as long as it converges it is faster than e.g. the bisection method) and can be easily generalized to high dimensional problems.

It is well-known that sometimes, iteration of a complex polynomial gives rise to chaos. For example, the French mathematician Gaston Julia iterated the complex polynomial $f_c(z) = z^2 + c$ and found beautiful figures. These are documented in the Julia set, see Figure 1. Although the Newton iteration function seems to be predictable, there are difficulties in generalizing the method to complex polynomials in order to find the roots (Cayley, 1879).

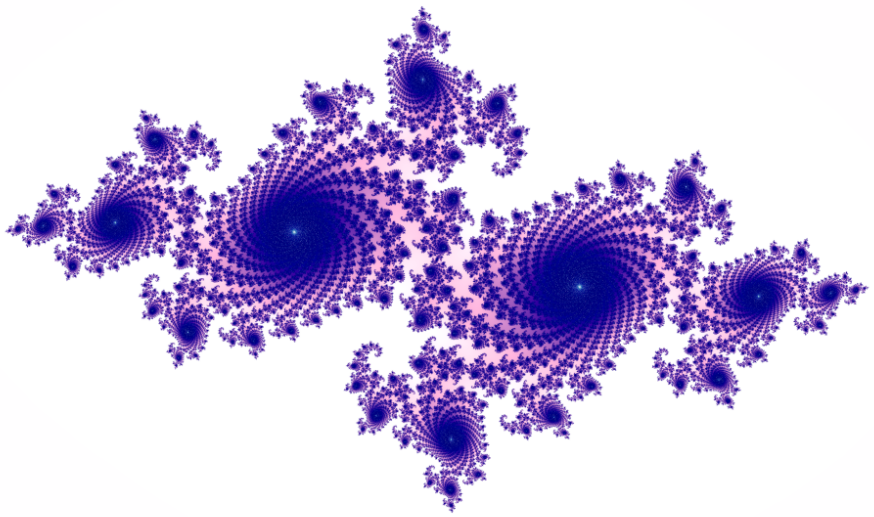


Figure 1: The Julia set displaying the complex polynomial $f_c(z) = z^2 + c$ (Eequor, 2005).

The focus in this report is to explore the behaviour of Newton's root-finding method on functions operating from and to the set of **complex** numbers, $f : \mathbb{C} \to \mathbb{C}$. What is the efficiency of the method in different regions of the complex plane? When does it converge or diverge? What are the fix-points of the function and in which areas will chaos occur?

In the following we will define the basic concepts of iteration and fixed points. Then we will describe Newton's root-finding method and show its properties. The main part of the project will be focused on designing a program to study the efficiency of Newton's method for a family of functions. Depending on the initial guesses, iterating Newton's method will give a varying efficiency. The program can be used for any function and is publicly available through github: https://github.com/juliajansson/NonlinearDynamicsAndChaos.

1.1 Basic Concepts

In this section the concepts of iteration, orbit and fixed point will be defined. First of all, iteration is something repeating over and over. So iterating a function f two times corresponds to $f(f(x))$.

Definition 1. *For an iterated function application of the function f we introduce the notation*

$$\begin{aligned} f^0(x) &= x \\ f^{n+1} &= f \circ f^n \end{aligned}$$

For example $f^1(x) = f(x)$ and $f^2(x) = f(f(x))$.

Starting from any number (called seed), we can use iteration to define a sequence of numbers, and we call this sequence an orbit.

Definition 2. *An orbit of x under f is an infinite sequence starting with x, where each new term has an extra iteration of f. The orbit $O_f(x)$ is*

$$O_f(x) = [x, f(x), f^2(x), \ldots].$$

For example, the orbit of the seed 1, under the function $g(x) = 2x$ is $[1, 2, 4, 8, 16, \ldots]$. This orbit tends to infinity. But using the seed 0, the orbit is constant: $[0, 0, \ldots]$

In general, an orbit can diverge (as in the first example), be a fixed point (as in the second example), converge to a limit or be periodic. There are other more complex orbits, that end up in chaos, and we will find these in our exploration of the Newton iteration method later. An example of a periodic orbit is the orbit of the seed 0 under the function $h(x) = 1 - x$. The orbit, $[0, 1, 0, 1, 0, 1, \ldots]$, consists of 0 and 1 repeating over again, so the period is 2. Consequently, a fixed point, has a period of 1. This means that the orbit of x consists of the same point over again: $[x, x, x, ...]$.

It is interesting to note that there are three categories of fixed points: attracting,

neutral and repelling. For an attracting fixed point, the x-values nearby move toward the attracting fixed point when repeatedly iterated (So et al., 1996). For a repelling fixed point the x-values nearby move away when repeatedly iterated. For a neutral fixed point, the values neither move towards nor from the point. The formal definition is as following.

Definition 3. *Given a fixed point x under f, x is attracting if $|f'(x)| < 1$, neutral if $|f'(x)| = 1$ and repelling if $|f'(x)| > 1$.*

For example, consider the function $g(x) = x$. It consists only of neutral fixed points since $|g'(x)| = 1$. Next, consider $h(x) = \sqrt{x}$. It has a fixed point at $x = 1$. If we take the orbits of two close points 1.1 and 0.9 we get $[1.10, 1.05, 1.02, 1.01, ...]$ and $[0.90, 0.95, 0.97, 0.99, ...]$. We see that both points seem to approach 1, it seems to be attracting. It can be shown to be attracting using the formal definition that since $h'(x)$ is equal to $\frac{1}{2\sqrt{x}}$ we have that

$$h'(1) = \frac{1}{2\sqrt{1}} = \frac{1}{2 \cdot 1} = \frac{1}{2} < 1$$

Now, as we have studied fixed points, let us move onward to critical points.

Definition 4. *A point x of $f(x)$ is critical, if the derivative, $f'(x)$, is equal to* 0.

1.2 Newton's Iteration Function

The Newton iteration function N_f is a function defined so that its iteration can be used to find the roots for a given function f with an initial guess, x (Cayley, 1879) (Yau & Ben-Israel, 1998). The expression for the function is

$$N_f(x) = x - \frac{f(x)}{f'(x)}.$$

The Newton iteration function has orbits and fixed points just like any other function. The attracting fixed points of N_f, are of great interest, since they are in one-to-one correspondence with roots of f. This is described in the theorem below, a proof of which can be found in (Devaney, 1992).

Theorem 1. *Given a function f, x_0 is a root of f if and only if x_0 is a fixed point to the Newton iterated function of f: N_f. Additionally, such a point is always attracting.*

This means that if you have found an attracting fixed point of N_f it is a root of f. (Devaney, 1992).

As an example of the critical point definition in section 1.1, we can compute the critical points of N_f by first computing the derivative

$$N_f'(x) = 1 - \left(\frac{f(x)}{f'(x)}\right)' = 1 - \left(f'(x) \cdot \frac{1}{f'(x)} + f(x) \cdot \left(\frac{1}{f'(x)}\right)'\right) =$$

$$= 1 - \left(1 + f(x) \cdot \left(\frac{-1}{f'(x)^2} \cdot f''(x)\right)\right) = \frac{f(x) \cdot f''(x)}{f'(x)^2}.$$

The critical points of N_f are the roots of its derivative, thus the points include all the roots of f and also the roots of f''.

2 Efficiency of Newton's Method

Now, with the background concepts introduced, the specific aim of the project will be described. Consider the family of functions $f_a(z) = (z^2 - 1)(z^2 + a)$ for varied values of a. This function, for a given a, has four roots, which correspond to the attracting fixed points of the Newton iteration function N_{f_a}. We will explore how efficient the Newton iteration function is for finding the roots of f_a. One of the critical points $c(a)$ of N_{f_a} are used as an initial guess for the Newton iteration function. For every complex value of a the Newton iteration function tries to approximate the roots of f_a, but it is not equally efficient everywhere. For some seeds $c(a)$ it takes more iterations to get close to a root than for other seeds.

The purpose of this project is to present a numerical exploration of how efficient Newton's method is for different $c(a)$, while a varies. This is determined by how many iterations it takes for N_{f_a} to get close to one of the roots of f_a when $c(a)$ is used as the seed for the orbit.

2.1 Roots of f_a and the Newton Iteration Function N_{f_a}

First let us find some of the relevant points related to f_a and N_{f_a}. The four roots of f_a are determined by setting $(z^2 - 1)$ or $(z^2 + a)$ equal to 0. This gives $z = \pm 1$ and $z = \pm\sqrt{a}i$. These four roots will be the attracting fixed points of N_{f_a}. Additionally, if the seed is in the vicinity of these four points, Newton's method will probably be efficient.

The Newton iteration function for f_a is

$$N_{f_a}(z) = z - \frac{f_a(z)}{f_a'(z)} = z - \frac{(z^2 - 1)(z^2 + a)}{((z^2 - 1)(z^2 + a))'}.$$

This can be simplified to

$$N_{f_a}(z) = \frac{3z^4 + z^2(a - 1) + a}{4z(z^2 + \frac{(a-1)}{2})}.$$

Remark. *This is undefined if the denominator* $4z(z^2 + \frac{(a-1)}{2})$ *is equal to* 0. *We have the roots* $z = 0$ *and* $z = \pm\sqrt{\frac{1-a}{2}}$. *Therefore the graph will approach infinity if* z *approaches* 0 *or* $\pm\sqrt{\frac{1-a}{2}}$.

2.2 The Critical Points $c(a)$ of N_{f_a}

We can calculate the critical points of N_{f_a}. They are all z such that

$$N'_{f_a}(z) = 0 \quad \longleftrightarrow \quad \frac{f_a(x) \cdot f''_a(x)}{f'_a(x)^2} = 0.$$

We have the roots of f''_a which are given by

$$f''_a(x) = 0 \quad \longleftrightarrow \quad ((z^2 - 1)(z^2 + a))'' = 0.$$

This is equivalent to

$$(z^4 + (a-1) \cdot z^2 - a)'' = (4z^3 + (a-1) \cdot 2z)' = 12z^2 + 2(a-1) = 0$$

which gives the roots $z = \pm\sqrt{\frac{1-a}{6}}$. Also, four of the critical points of N_{f_a} are the four roots of f_a. So now we have found six critical points for N_{f_a}. These are

$$\begin{aligned}
c_1(a) &= \sqrt{\frac{1-a}{6}}, \\
c_2(a) &= -\sqrt{\frac{1-a}{6}}, \\
c_3(a) &= 1, \\
c_4(a) &= -1, \\
c_5(a) &= \sqrt{a}i, \\
c_6(a) &= -\sqrt{a}i,
\end{aligned}$$

3 Numerical Algorithm

The goal with this project is to explore the efficiency of Newton's method. To accomplish this two algorithms were designed. The first algorithm plots a bitmap of how many iterations of the Newton iteration function are needed for a given function f_a in the complex plane. The second algorithm explores the family of functions f_a and shows how N_{f_a} behaves while a varies, by illustrating the efficiency of the root-finding using critical points.

In the first algorithm, a complex number z_0 is used as a seed of the orbit for N_{f_a}. If an attracting fixed point is found immediately the point is colored black. Algorithmically an attracting fixed point is considered found if the difference between the two most recent values was small enough, $\leq 10^{-6}$. These attracting fixed points correspond to the roots. Around the roots there are areas which are attracted to the fixed points, but it takes some iterations for the difference to be small enough so they are colored lighter. Areas that need a large number of iterations are colored white. The result of the first algorithm is an iteration plot which shows how efficient the Newton iteration function is in different regions of the complex plane for a function f_a. Every a corresponds to an entirely different picture, with efficiency shown for all seeds.

The second algorithm is designed to show how fast the Newton iteration function tends to attracting fixed points if critical points are used as seeds. First, one of the six critical points (c_1 to c_6) is chosen. For different complex values of a in the critical point $c(a)$, a plot is constructed to show how many iterations are needed for N_{f_a} to get the roots (attracting fixed points) to f_a. So for every complex a, the critical point $c(a)$ is used as the seed of the orbit for the N_{f_a}. The iterations are computed as in the first algorithm.

Additionally this second algorithm checks which one of the roots is found. When the difference between the two last values is small enough it saves the last value, call it z. It then checks if z is equal to one of the roots or if it is something entirely different. It is enough if the difference between z and a root is very small, because we already know it

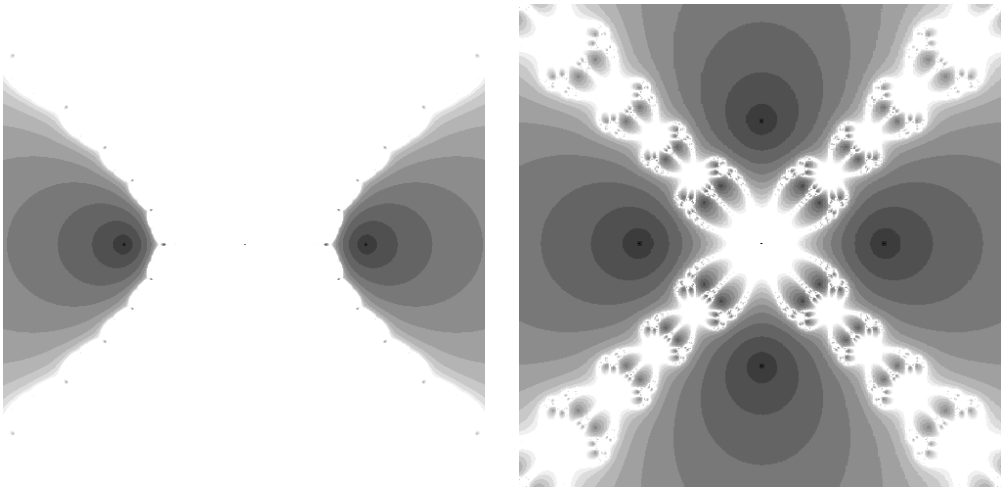
is an attracting fixed point for the Newton iteration function. So the program checks if the difference is small enough for any of the roots.

The result is represented by colors in the plot, red is $+1$, green is -1, blue is $+\sqrt{a}i$ and yellow is $-\sqrt{a}i$. If z is not close to any of the roots the point is painted white. The lightness of the color of a given point is then given the intensity of the number of iterations it takes to find the roots. For example a light blue is a point where the Newton iteration function is not too efficient but eventually finds the root $+\sqrt{a}i$. A dark red point is a point where the Newton iteration function efficiently finds the root $+1$. A white point is a point where no root is ever found and the Newton iteration function diverges.

4 Results

4.1 Onset of Chaos

The first algorithm was computed for different values of a. In Figure 2, two iteration plots show how many iterations are needed for the Newton iteration function N_{f_a} until the difference between the two final values was $\leq 10^{-6}$. The case when no iteration is needed is represented by black color and the case when many iterations are needed is represented by white color. The darker areas are where the root is found quickly using the initial guess, e.g. the initial guess is close to the root. The lighter areas are where the root is found after a large number of iterations or not at all. The range of values shown is between $z = -2 - 2i$ at the bottom left edge and $z = 2 + 2i$ at the upper right edge (where origin is in the middle), and the resolution of the plot is 500×500 pixels.



(a) The plot of the Newton iteration function $N_{f_a}(z)$ when $a = 0$.

(b) The iteration plot of the Newton iteration function $N_{f_a}(z)$ when $a = 1$.

Figure 2: Efficiency of Newton's method illustrated by the number of iterations of $N_{f_a}(z)$ for initial guesses between $z = -2 - 2i$ and $z = 2 + 2i$, for two different values of a. Dark and light areas correspond to few and many iterations, respectively.

The darkest parts of the plot are the roots, ± 1 and $\pm\sqrt{a} \cdot i$. In the case where $a = 0$ in Figure 2a the roots $\pm\sqrt{a} \cdot i$ are a double root in 0. Here we can see that the roots of $f_a(z)$ are black with grey gradients surrounding them. The large light area shows that the Newton iteration function is not always efficient. However, a small change in the initial

guess will not give rise to very different behaviour. But in the second case, corresponding to $a = 1$, around the four diagonal lines in Figure 2b the colors change more quickly from black to white and vice versa, in a pattern. So in some areas the root is found immediately and in others after a long number of iterations, and these areas are really close to each other. This means that a small change in the initial guess leads to large variations in the outcome: a signature of chaotic behaviour! Chaotic behaviour is observed also for other values of a that we tried (except $a = 0$). As an example we show the case of $a = 0.18$ in Fig. 3.

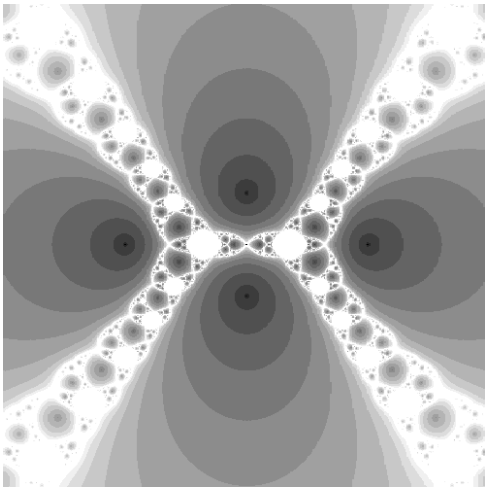


Figure 3: Efficiency of Newton's method illustrated by the number of iterations of $N_{f_a}(z)$ for initial guesses between $z = -2 - 2i$ and $z = 2 + 2i$, for $a = 0.18$. Dark and light areas correspond to few and many iterations, respectively.

4.2 Critical Points and Efficiency

Now we will show how fast the Newton iteration function tends to attracting fixed points if critical points are used as seeds. This type of plot tells how many iterations are needed for N_{f_a} to get the roots of f_a for different values of a starting from the critical point $c(a)$. In the same way as the original iteration plot it illustrates the number of iterations but also with a color for which root it is. Red is $+1$, green is -1, blue is $+\sqrt{a}i$ and yellow is $-\sqrt{a}i$.

The case of $c_1(a)$ Let us now consider the case of $c_1(a) = \sqrt{\frac{1-a}{6}}$. In Figure 4 we can see the plot of how efficient the N_{f_a} is for different values of $c_1(a)$. The range of values shown in the plot is the same as in Figure 2 but is now in the resolution of 1000×1000 pixels. A grid is added with 0.2 length, so five steps out from the center is 1. Now let us dig into the complexity of this beautiful picture.

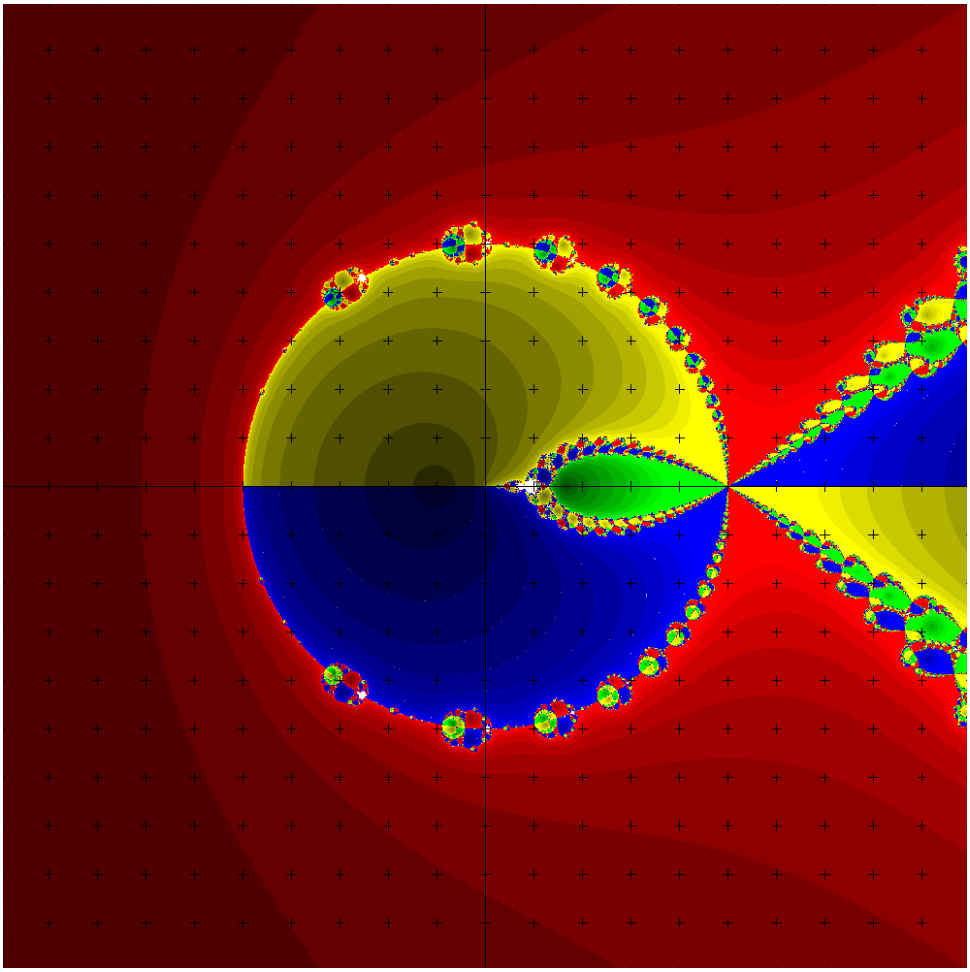


Figure 4: Efficiency of Newton's method illustrated by the number of iterations of $N_{f_a}(z)$, represented by the lightness of the color, using the critical point $c_1(a)$ as initial guess. The different colors represent the roots: red is $+1$, green is -1, blue is $+\sqrt{a}i$ and yellow is $-\sqrt{a}i$.

The middle area, roughly the unit circle, consists of a yellow half circle on the positive side and a blue half circle on the negative side. In the yellow area the root $-\sqrt{a}i$ is found and in the blue area the root $+\sqrt{a}i$ is found. In the case of the initial guess being -0.2 it has the fewest iterations. Between 0.2 and 1 there is a green area where the root -1

can be found. The most efficient initial guess in this case is a range of values between 0.2 and 0.4. Continuing from 1 to 2 in the plot we have a yellow and blue triangular area, where the efficiency increases as we move closer to 2. Surrounding all these areas is the big red area. Here the root +1 can be found. The method is increasingly efficient as we move towards the negative real axis.

There are two especially interesting areas which are in the vicinity of 0.2 and in the vicinity of 1. The Newton iteration function in 1 diverges to infinity. But if we go a small distance from 1 we can find a root. It matters which way we go, because the point 1 is close to all areas. This phenomenon can be found in other areas too, for example the small circles around the edges of the unit circle.

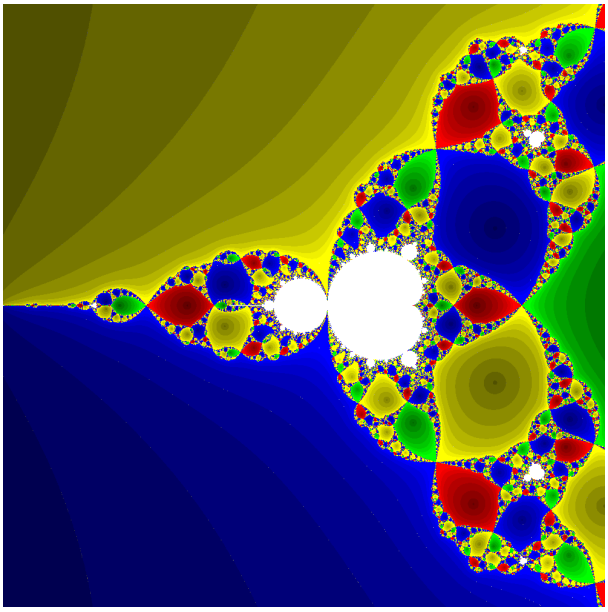


Figure 5: Zoomed in version of Figure 4, with $z = 0 - 0.15i$ at the bottom left edge and $z = 0.3 + 0.15i$ at the upper right edge.

Going from these incredibly colorful areas to the white area in the vicinity of 0.2 is not only a contrast to the eye but a change in how the Newton iteration function behaves. A zoomed in version of this can be found in Figure 5. Using values in this area as seeds does not lead to finding any roots, instead it diverges to infinity. And it looks like the famous Mandelbrot set!

The case of $c_2(a)$ Moving on to the next critical point $c_2(a) = -\sqrt{\frac{1-a}{6}}$ we can see that the geometrical shape in Figure 6 is similar to that in Figure 4. In fact, the only thing that has changed is the green and the red areas and the yellow and blue areas. So this means that where you could earlier find the root $+1$ you now find the root -1, and where you would find $+\sqrt{a}i$ you now find $-\sqrt{a}i$. This is explained by

$$c_2 = -\sqrt{\frac{1-a}{6}} = -c_1.$$

The second critical point is equal to the first critical point but with a negative sign. They have a symmetry in the same way that the roots ± 1 and $\pm\sqrt{a}i$ have. Therefore the plots are identical except for the color change. The same areas in the vicinity of 1 and 0.2 can be found.

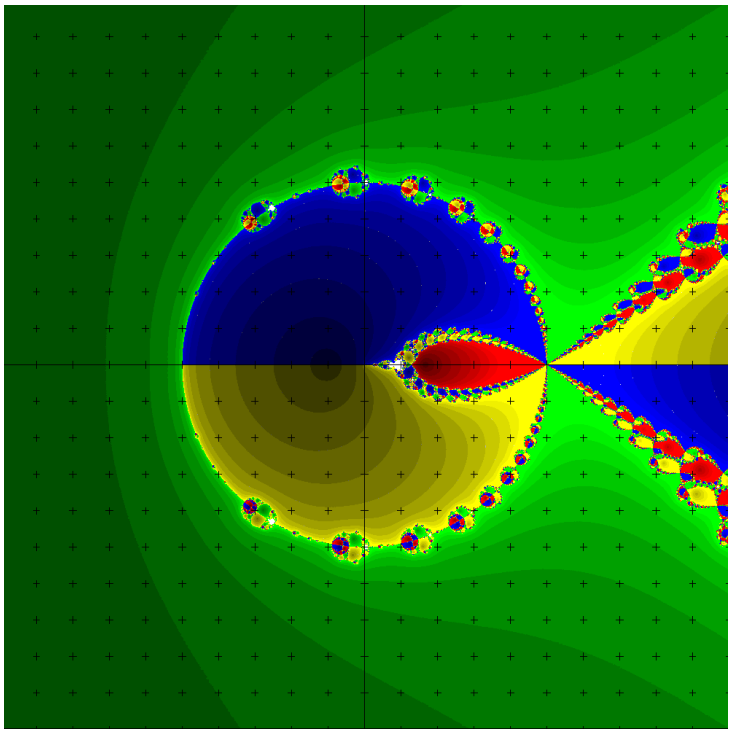


Figure 6: Efficiency of Newton's method illustrated by the number of iterations of $N_{f_a}(z)$, represented by the lightness of the color, using the critical point $c_2(a)$ as initial guess.

The case of $c_3(a)$-$c_6(a)$ These four critical points are rather simple. For $c_3(a) = 1$ the Newton iteration function finds the root $+1$ to f_a at zero iterations for every a

value considering the critical point c_3. Therefore the plot for the critical point c_3 is red everywhere. No chaos occurs, and the Newton iteration function is equally efficient for every given a value. In the same way the critical point $c_4(a) = -1$ always finds -1, $c_5(a) = \sqrt{a}i$ finds $\sqrt{a}i$ and $c_6(a) = -\sqrt{a}i$ finds $-\sqrt{a}i$. Consequently c_4 is green, c_5 is blue and c_6 is yellow everywhere.

4.3 The Efficiency of the Newton Iteration Function

We have now explored how efficient the Newton iteration function is for different a values for different critical points. The algorithms we used determined how efficient N_{f_a} was by seeing how many iterations were needed to get close to a root. The difference between two values had to be smaller than 10^{-6} for a point to be a root. But how does this given difference impact the efficiency of N_{f_a}? In Table 1 we show the efficiency for the Newton iteration function for differences ranging in exponential steps from 10^{-1} to 10^{-10}.

a	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}
$2i$	2	3	4	4	5	5	5	5	5	6
100	23	23	24	24	24	25	25	25	25	25

Table 1: The efficiency of the Newton iteration function for the a values $2i$ and 100. In the algorithm the distance between two points to be close enough for the Newton iteration function to be done are 10^{-1} to 10^{-10}.

There is not much difference between the first and last values even though the differences are ranging over 10 exponents. This means that once the Newton iteration function is close to the root, it is very efficient.

5 Discussion

We have explored how efficient N_{f_a} is for finding roots of f_a for different $c(a)$, while a varies. Using a critical point as the seed of each orbit, the efficiency is determined by how many iterations are needed to reach an attracting fixed point.

First we explored what the function would look like for a given a and how efficient N_{f_a} would be then. We see that in regions in the vicinity of the roots, the Newton iteration function is very efficient. In the "diagonal lines" where the function is as far from the roots as possible, a chaotic behaviour occurs. This has given us an image of how the Newton iteration function operates on a function with a given a, but the task was to explore how the efficiency varies using the critical point as seed for N_{f_a}'s orbit with varying a. When a varies the function f_a varies with it, and so does also the roots $\pm\sqrt{a}i$.

The critical points, c_1 and c_2, depend on a so we have made a colored iteration graph to explore the efficiency in different a-values. The graphs for c_1 and c_2 are much alike. They have only switched colors on the areas. There are areas where one of the four roots can be found, with varying efficiency. Finding $\pm\sqrt{a}i$ is best done in the vicinity of -0.2 and moving towards $+2$. Finding ± 1 is most efficient around $+0.3$ and towards -2. In some areas, for example in the vicinity of $+1$ all the four roots can be found very close to each other. In the vicinity of 0.2 however there is a completely white area where the Newton iteration function is entirely inefficient.

The other critical points, c_3 to c_6 are always efficient, because they are the roots of f_a. For these points the Newton iteration function always gives out one of the roots whatever a is, they are equally efficient everywhere.

Finally, the efficiency of the Newton iteration function was analyzed more directly by changing the accuracy used for the Newton iteration function in the algorithm. Here we can see that it is very efficient, once it is close to the root.

5.1 Error Sources

We have explored this graphically, not algebraically, and that can be a source of error. The plot was only made between $z = -2 - 2i$ and $z = 2 + 2i$, and additionally it has a resolution of 1000×1000. This means that we can only see the result in a finite number of points.

5.2 Further Research

The algorithms presented in this work can be used for plotting similar images illustrating the number of iterations needed for any function, not only the family of functions shown here. Therefore it can be used in future studies of efficiency of root-finding algorithms for arbitrary functions.

The depth of the information shown in the plots is enormous, and there are many things that could be further explored with this family of polynomials, for example the area in the vicinity of $+0.2$ in the c_1 and c_2 graphs. Why does the Mandelbrot set appear? And what happens in the areas where chaos occur?

Acknowledgements

I would like to thank Jenny Homann, my supervisor, for her help in this project. I am also grateful to Prof. Patrik Jansson for help with programming in Haskell and to Cezar Ionescu who recommended the book by Robert Devaney *A first course in chaotic dynamical systems: theory and experiment* that has inspired my project and helped me to understand the theory. Lastly, I am thankful for friends and family who have taken the time to read and give feedback on this paper.

References

Cayley, P. (1879). Desiderata and suggestions: No. 3. the Newton-Fourier imaginary problem. *American Journal of Mathematics*, *2*(1), 97–97. Retrieved from http://www.jstor.org/stable/2369201

Devaney, R. L. (1992). *A first course in chaotic dynamical systems: theory and experiment*. Perseus Books Publishing.

Eequor. (2005, December 18). Julia set. Retrieved February 18, 2017, from https://en.wikipedia.org/wiki/Julia_set#/media/File:Julia_set_(ice).png

Lorenz, E. (1995). *The essence of chaos*. Jessie and John Danz lectures. Taylor & Francis.

Newton, I. (1736). *The method of fluxions and infinite series: with its application to the geometry of curve-lines*. Eighteenth century collections online. Henry Woodfall; and sold by John Nourse.

So, P., Ott, E., Schiff, S. J., Kaplan, D. T., Sauer, T., & Grebogi, C. (1996). Detecting unstable periodic orbits in chaotic experimental data. *Physical Review Letters*, *76*(25), 4705.

Yau, L. & Ben-Israel, A. (1998). The Newton and Halley Methods for Complex Roots. *The American mathematical monthly*, *105*(9), 806–818.