COMPUTATIONAL STUDY OF CLASSICAL FRACTALS AND NEWTON BASINS OF ATTRACTION IN THE COMPLEX PLANE

Siti Sahrani[*], Salmawaty[*], Said Munzir[*], Saiful Amri[*]
[*]*Department of Mathematics, Faculty of Mathematics and Natural Sciences, University of Syiah Kuala, Banda Aceh, Indonesia, sitishrn30@gmail.com, salmawaty@usk.ac.id, smunzir@usk.ac.id, saiful_zahlen@usk.ac.id*

Email Correspondence : salmawaty@usk.ac.id

Received : August 5, 2024 Accepted : November 3, 2025 Published : December 31, 2025

Abstract: Fractal sets such as the Sierpinski Gasket, the Julia set, the Mandelbrot, and the Newton basin attraction provide rich examples of self-similar structures arising from simple iterative rules. In this work, we present a computational study of these fractals with emphasis on parameter sensitivity and basin dynamics. Using C++, we generated classical fractals (Sierpinski, Julia, and Mandelbrot) and verified their self-similar properties through coordinate transformations. For the Newton basin attraction of the polynomial $f(z) = z^n + c$, implemented in MATLAB, we analyzed the effect of the exponent n, complex constant c, and iteration depth on convergence regions. The results show that the number of basin attractions corresponds to the polynomial degree, with clear rotational symmetries, while the parameter c controls the orientation and distribution of convergence regions. Furthermore, the boundary of the Newton basin attraction was observed to align with Julia sets, highlighting the intrinsic connection between these two fractal structures. This study demonstrates how computational visualization provides analytical insight into the structure and parameter dependence of fractal systems, establishing a systematic framework for parameter-based fractal analysis that bridges classical visualization with numerical dynamics.
Keywords: Basin of attraction; fractal; Julia set; Mandelbrot set; Newton method; Sierpinski gasket.

Abstrak: Himpunan fraktal seperti segitiga Sierpinski, Julia, Mandelbrot, dan Newton basin atraksi merupakan representasi khas dari struktur *self-similarity* yang terbentuk melalui aturan iteratif sederhana. Penelitian ini menyajikan kajian komputasional terhadap fraktal-fraktal tersebut dengan fokus pada sensitivitas parameter serta dinamika basin atraksi. Algoritma fraktal klasik diimplementasikan menggunakan C++ untuk menghasilkan pola Sierpinski, Julia, dan Mandelbrot sekaligus memverifikasi sifat swasimilaritasnya melalui transformasi koordinat. Sementara itu, Newton basin atraksi dari polinomial $f(z) = z^n + c$ dianalisis menggunakan MATLAB untuk mengkaji pengaruh eksponen n, konstanta kompleks c, dan kedalaman iterasi terhadap daerah konvergensi. Hasil penelitian memperlihatkan bahwa jumlah basin atraksi sebanding dengan derajat polinomial dan menampilkan simetri rotasional yang khas, sedangkan parameter c mempengaruhi orientasi serta distribusi wilayah konvergensi. Selain itu, batas Newton basin atraksi ditemukan berimpit dengan himpunan Julia, menegaskan keterkaitan mendasar antara kedua struktur fraktal tersebut. Penelitian ini menunjukkan bahwa visualisasi komputasional mampu memberikan pemahaman analitis terhadap struktur dan pengaruh parameter pada sistem fraktal. Selain itu, penelitian ini juga

DOI: 10.22373/ekw.v11i2.25438

membangun kerangka kerja sistematis untuk analisis fraktal berbasis parameter, yang menghubungkan visualisasi klasik dengan dinamika numerik.
Kata kunci: *Basin of attraction*; fraktal; himpunan Julia; himpunan Mandelbrot; metode Newton; gasket sierpinski.

Recommended APA Citation :
Sahrani, S., Salmawaty, Munzir, S., & Amri, S. (2025). Computational Study of Classical Fractals and Newton Basins of Attraction in The Complex Plane. *Elkawnie*, *11*(2), 31-45. https://doi.org/10.22373/ekw.v11i2.25438

Introduction

Fractals are mathematical objects that exhibit self-similarity across multiple scales and are generated through simple iterative processes. They are studied both for their intrinsic aesthetic appeal and for their relevance in nonlinear dynamics, numerical methods, and scientific visualization. Classical examples include the Sierpinski triangle, Julia sets, and the Mandelbrot set, which are widely recognized as canonical illustrations of fractal geometry and iterative dynamical systems.

Recent studies have investigated fractal structures not only from a visual perspective but also for their computational and analytical properties. Julia and Mandelbrot sets, for instance, have been analyzed under parameter variation to reveal bifurcation phenomena and sensitivity to initial conditions (Peitgen et al., 2004; Devaney, 1989). Similarly, Newton basins of attraction, obtained by applying Newton's method to complex polynomials, generate intricate convergence regions that reflect the interplay between iteration dynamics and root structure (Hubbard et al., 2001; Peitgen & Richter, 1986). Beyond these classical approaches, modern work has explored deep learning techniques to accelerate basin classification (Wang et al., 2021) and higher-order root-finding schemes such as Halley's method, which produce basin geometries with novel symmetries (Amat et al., 2003).

Despite these advances, much of the literature remains descriptive, with limited focus on systematic parameter analysis. In particular, explicit computational comparisons between Newton basin attraction and Julia sets are relatively rare, and the influence of parameters such as polynomial degree, complex constant, and iteration depth on geometry of basin attraction has not been fully explored.

This study addresses these gaps by presenting a computational analysis of fractals and Newton basin attraction with emphasis on parameter sensitivity and structural behavior. Using C++, we implemented algorithms to generate the Sierpinski triangle, Julia sets, and the Mandelbrot set, verifying their self-similarity and coordinate transformation properties. In parallel, Matrix Laboratory (MATLAB) was used to investigate Newton basin attraction for the polynomial $f(z) = z^n + c$, analyzing how the degree n, parameter c, and iteration depth affect convergence regions. We further demonstrate how basin boundaries correspond to Julia sets, highlighting their intrinsic relationship within complex dynamics. Despite advances such as Artificial Intelligence (AI) based classification and

higher-order schemes (e.g., Halley's method), Newton's method remains the classical and most widely studied framework. Our analysis establishes a systematic framework for parameter-based exploration of Newton basin attraction, providing a baseline for future extensions in both theoretical and computational directions.

Methods

Sketch of Fractal

Coordinate in $\mathbb{R}^2$ vs Coordinate in Computer

In mathematics, the plot of a point $T = (x, y) \in \mathbb{R}^2$, where $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$, is represented from the origin (the intersection of the two axes): x unit to the right (or left if $x < 0$) and y units upward (or downward if $y < 0$). In contrast, the coordinate system in computer graphics adopts a different convention. The origin is located at the upper-left corner of the screen. The abscissa x' increases to the right, while the ordinate y' increases downward, and both are always nonnegative. Consequently, to visualize a point $T = (x, y) \in \mathbb{R}^2$ on computer, it must first be transformed into its counterpart $T' = (x', y') \in \mathbb{K}^2$, where $\mathbb{K}^2 = \mathbb{K} \times \mathbb{K}$ denotes the computer coordinate system.

Coordinate transformation from $\mathbb{R}^2$ ke $\mathbb{K}^2$

Before rendering graphics, two steps are required:

Define the plotting in $\mathbb{R}^2$

Select the minimum and maximum abscissa, x_m, x_M, and ordinates, y_m, y_M. Define the scales s_x and s_y for the x-axes and y-axes, respectively. The rectangular plotting area thus has dimensions $\frac{\Delta x}{s_x} \times \frac{\Delta y}{s_y}$, where $\Delta x = x_M - x_m$ and $\Delta y = y_M - y_m$.

Define the plotting region in $\mathbb{K}^2$

To achieve congruence with $\mathbb{R}^2$, set the resolution to H pixels horizontally and V pixels vertically:

$$H = u\frac{\Delta x}{s_x} \quad \text{..... (1)}$$

$$V = u\frac{\Delta y}{s_y} \quad \text{..... (2)}$$

Where u is the number of pixels for each unit. The greater the u value, the smoother the resulting image will be, but it slows down processing.

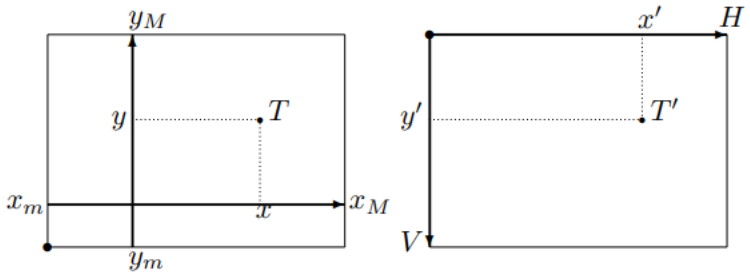

Figure 1. Coordinate Transformation

Figure 1 illustrated the mapping between the two systems. For given a point $T = (x, y) \in \mathbb{R}^2$, the transformation $\psi : \mathbb{R}^2 \to \mathbb{K}^2$ yields

$$\frac{x-x_m}{\Delta x} = \frac{x'}{H} \quad \text{..... (3)}$$

$$\frac{yM-y}{\Delta y} = \frac{y'}{V} \quad \text{..... (4)}$$

$$\psi : (x, y) \mapsto \left(\frac{H(x-x_m)}{\Delta x}, \frac{V(y_M-y)}{\Delta y}\right) \quad \text{..... (5)}$$

With the inverse mapping

$$\psi^{-1}: (x, y) \mapsto \left(x_m + \frac{x'\Delta x}{H}, y_M - \frac{y\prime\Delta y}{V}\right) \quad \text{..... (6)}$$

The transformation between the Euclidean plane $\mathbb{R}^2$ and the computer coordinate system $\mathbb{K}^2$ is therefore established via the linear mapping (5–6), ensuring consistency between mathematical formulation and graphical visualization.

Sierpinski Gasket

Sierpinski's gasket is a geometric fractal pattern constructed from a triangle $T_0T_1T_2$. To generate the gasket, an initial point G_0 is chosen inside or on the boundary of the triangle. For each iteration $i > 0$, the next point G_i is defined as the midpoint between G_{i-1} and one of the vertices T_j, where the index $j \in \{0, 1, 2\}$ is selected randomly. Repeating this process yields a sequence of points $G_1, \ldots, G_n$, which collectively form the fractal structure. As the number of iterations increases, the resulting image becomes increasingly intricate and detailed. Remarkably, although the construction is governed by random choices, the process consistently converges to the self-similar fractal known as the Sierpinski gasket, independent of the initial point G_0 or the sequence of random selections.

Julia's set and Mandelbrot's set

Julia's set is denoted by the symbol J_c and defined by a function $f(z) = z^2 + c$, where $c \in \mathbb{C}$. This set contains all the complex numbers w such that the

sequence $w_0 = w, w_1 = f(w_0), w_2 = f(w_1)$, ... is finite, i.e. for each $j \geq 0$ and $r \in R$, where r referred to as the ("radius of the boundary"), so that $|w_j| <$ r. The sequence of $w_{j\geq 0}$ is called the "orbit from w to f", or simply referred to as the orbit of w (Abbas et al., 2020).

$$J_c = \{z \in \mathbb{C} : orbit\ z\ is\ bounded\} \text{ ..... (7)}$$

The Mandelbrot set is formed in a similar way to the Julia set. It is based on the function $f(z) = z^2 + c$, where c is a constant. The only difference is that the orbit examined is always that of $0 + 0i$. If the orbit is finite, then c is considered to be part of the Mandelbrot set, denoted as $c \in M$. However, if the orbit is infinite, then c is not part of the Mandelbrot set, represented as $c \notin M$ (Farris, 2022).

$$\mathcal{M} = \{c \in \mathbb{C} : orbit\ z\ is\ bounded\} \text{ ..... (8)}$$

Fixed Points and Degrees of The Methods

Suppose that f is a rational map with a degree of 2 or more.

Definition 1. *The orbit of a point z is the set of iterates*

$$O(z) = \{\cdots, f^{-2}(z), f^{-1}(z), z, f(z), f^2(z), \cdots\}$$

If there exists a positive integer $n > 0$ such that $f^n(z) = z$, then we refer to O(z) as a periodic orbit of f with period n. In simpler terms, the orbit of z repeats itself every n iterations. Finally, if f(z) = z, z is considered a fixed point of f.

A fundamental procedure in comprehending the dynamical intricacies of the set f involves the identification and characterization of all nonrepelling periodic orbits. Specifically, fixed points, denoting orbits with a period of 1, are systematically categorized according to the following classification

Definition 2. *(Classification of fixed points): If z^* is a fixed point of the map f, then we say that z^* is*

1) Superattracting if $f'(z^*) = 0$,
2) Attracting if $0 < |f'(z^*)| < 1$,
3) Indifferent if $|f'(z^*)| = 1$,
4) Repelling if $|f'(z^*)| > 1$.

For every function f, there exists an attracting fixed point with an immediate attracting basin - a surrounding region where all iterations within it will ultimately converge to that fixed point. It is ideal for a one point iterative root-finding algorithm to have each root as a superattracting fixed point and every nonrepelling fixed point as a root. The above classification of fixed points can also be applied to periodic cycles. If there are attracting periodic cycles, then iterations near the cycle will converge to it rather than a root, for initial data in that vicinity (Geyer, 2020).

Newton Basin of Attraction

Consider a rational function $f(z)$ and its Newton function $N(z) = z - \frac{f(z)}{f'(z)}$. Let z_k^* denote a root of $f(z)$. Each root is a fixed point of the Newton function $N(z)$. The basin of attraction of z_k^* is defined as

$$W(z_k^*) = \{z \in \mathbb{C} : N^m(z) \to z_k^* \; as \; m \to \infty\}, \; \ldots\ldots\ldots (9)$$

i.e., the set of all points whose iterates under $N(z)$ converge to the root z_k^*. The Fatou set $F(f)$ of a rational function f is the set of points $\mathbb{C}$ whose orbits converge to an attractor. Conversely, the Julia set $J(f)$ is the complement of the Fatou set in the complex plane $\mathbb{C}$. Importantly, the boundary of $W(z_k^*)$ coincides with the Julia set $J(f)$ (Kanwar et al., 2023; Cordero et al., 2024). In the current system, the basin of attraction is determined using a root-finding technique, specifically the multidimensional Newton-Raphson method (Jimenez et al., 2019).

Result and Discussion

Drawing Sierpinski Gaskets

In section 2 it has been mentioned that, given a triangle, its three angular points

$$T_i = (x_i, y_i), \qquad k \in K = \{0,1,2\},$$

and given a point $G_0 = (x_i, y_i)$ inside or on that triangle, the Sierpinski Gasket is the set of points $G_1, G_2, \cdots$ where for $i = 0,1, \cdots$ we got

$$G_{i+1} = (x_{i+1}, y_{i+1}) = \left(\frac{x_i + x_k}{2}, \frac{y_i + y_k}{2}\right), \; \ldots\ldots\ldots (10)$$

Is the middle point of G_i and T_k with $k \in K$ chosen randomly. Since (6) remains true in computer coordinate systems, hence the correctness of the following function in charge of plotting Sierpinski gaskets using C++ is clear. The algorithm for drawing Sierpinski's Gasket can be seen in Algorithm 1.

Algorithm 1: Sierpinski Gasket

Input: Three corner points (x_0, y_0), (x_1, y_1), (x_2, y_2), point $G_0 = (x, y)$ and an integer $n > 0$ (number of points to be plotted)
Output: Plot points $G_1, G_2, \ldots, G_n$ $G_1, G_2, \ldots, G_n$
for $i \leftarrow 1$ **to** n **do**
    choose $k \in \{0, 1, 2\}$ randomly;
    $x \leftarrow (x + x_k)/2$; $y_i \leftarrow (y + y_k)/2$;
    plot (x, y);
end

Once implemented, the four coordinates, namely T_0, T_1, T_2, and G_0, are expressed in array form to improve efficiency. This means that the input variable n, which represents the number of iterations, is no longer required and will be ignored. Instead, the n waiver will remain in place until a button is pressed using the kbhit command.

Drawing Julia's set and Mandelbrot's set

In practice, to draw the set of Julia J_c and Mandelbrot set $\mathcal{M}$ on canvas $K = [x_m, x_M] \times [y_m, y_M]$, figure 1 on the left shows that the domain is limited to the K Argand plane. It is not feasible to check it in $\mathbb{C}$ entirely. Therefore, z will only be plotted when $z \in K \cap J_c$ for Julia's set or $z \in K \cap M$ for Mandelbrot's set. The number of terms from z orbit is infinite, but it should be limited to several terms

only, such as up to n terms. Additionally, the chosen radius j should not be too large so that the computer can perform the calculation. The larger the radius, the smoother the resulting image will be, but it takes a long time to draw. Based on the paragraph above and section 4.2, the algorithm for drawing the set of Julia J_c with $c = a + bi$ on a canvas of $[x_m, x_M] \times [y_m, y_M]$, can be written as Algorithm 2 below.

Algorithm 2: Julia Set J_{a+bi}

for each $x \in [x_m, x_M]$ **and** each $y \in [y_m, y_M]$ **do**
  $r \leftarrow x;\ s \leftarrow y;\ i \leftarrow 0$
  do
    $i \leftarrow 1 + i;\ t \leftarrow r;\ r \leftarrow r^2 - s^2 + a;\ s \leftarrow 2ts + b;$
  while $i < n$ **AND** $r^2 + s^2 < j^2$;
  if $i > n$ **then** plot(x, y);
end

The algorithm for describing the Mandelbrot set can be seen in Algorithm 3.

Algorithm 3: Mandelbrot Set

for each $x \in [x_m, x_M]$ **and** each $y \in [y_m, y_M]$ **do**
  $r \leftarrow 0;\ s \leftarrow 0;\ i \leftarrow 0$
  do
    $i \leftarrow 1 + i;\ t \leftarrow r;\ r \leftarrow r^2 - s^2 + a;\ s \leftarrow 2ts + b;$
  while $i < n$ **AND** $r^2 + s^2 < j^2$;
  if $i > n$ **then** plot(x, y);
end

The output of algorithm 1, the Sierpinski Gasket image, can be seen in Figure 3.

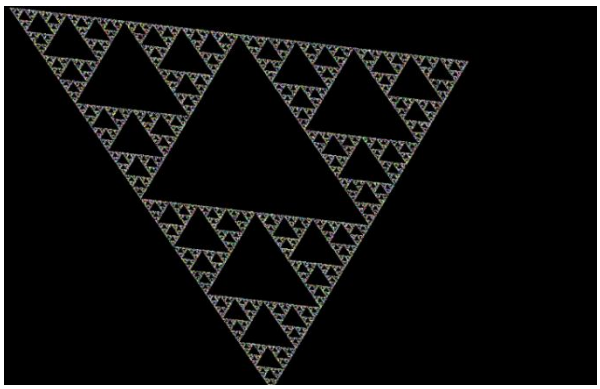

Figure 3. Output Sierpinski Gasket

The Sierpinski gasket produced by the random midpoint algorithm exhibits the characteristic triangular self-similar structure. Despite the stochastic nature of point generation, the resulting pattern consistently converges to a deterministic fractal. This highlights a fundamental property of the gasket: randomness in iteration does not affect the convergence to its ordered geometry. The output verifies the theoretical property that every iteration generates scaled copies of the initial triangle, resulting in self-similar pattern across multiple scales.

Images of Julia set for some c values as the output of the program above, can be seen in Figure 4, Figure 5, and Figure 6.

Figure 4. Julia set for $c = -0.7488 + 0.107421i$

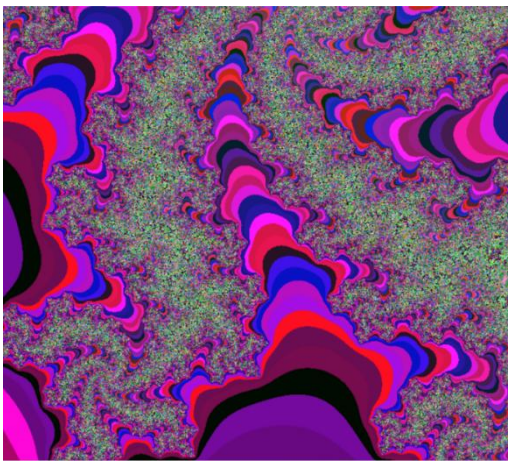

Figure 5. Julia set for $c = -0.54 + 0.54i$

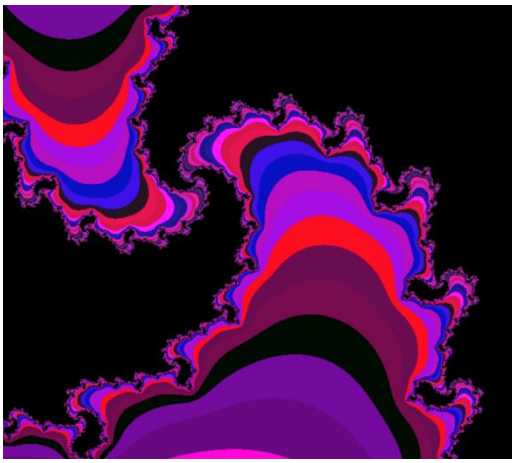

Figure 6. Julia set for $c = -0.9 + 0.2i$

The Julia sets computed for different values of c reveal the sensitivity of the geometry to parameter choice.

For $c = -0.7488 + 0.107421i$ (Figure 4), the set appears connected, forming dendritic branches with relatively smooth structures, consistent with stable orbits.

For $c = -0.54 + 0.54i$ (Figure 5), the set displays more complex branching, where some regions are connected while others begin to fragment, reflecting near the boundary of stability.

For $c = -0.9 + 0.2i$ (Figure 6), the Julia set becomes highly fragmented, with disconnected islands indicating chaotic orbit divergence.

These results illustrate the classical dichotomy between connected and disconnected Julia sets: parameters c inside the Mandelbrot set produce connected structures, while those outside yield disconnected "dust-like" patterns.

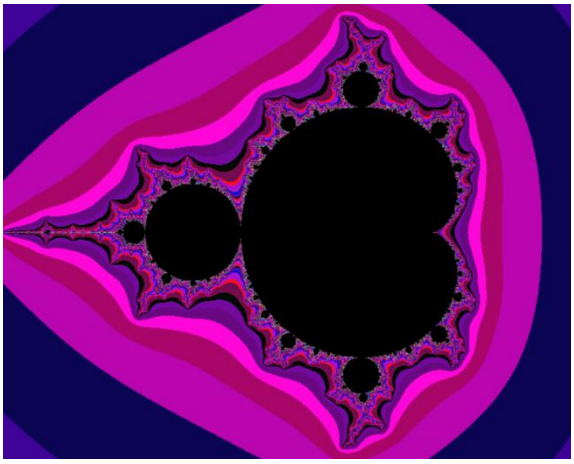

Figure 7. Mandelbrot set in the complex plane

Mandelbrot set generated in C++ (Figure 7) displays the familiar cardioid shape with attached circular "bulbs." The black region corresponds to parameters c for which the orbit of the critical point $z = 0$ remains bounded, while the exterior corresponds to divergence. The boundary of the Mandelbrot set exhibits self-similar structures across multiple scales: zooming into smaller regions reveals features that resemble the global structure. Crucially, the Mandelbrot set acts as the parameter space for Julia sets: points c inside the Mandelbrot produce connected Julia sets (Figures 4–5), while points outside yield disconnected sets (Figure 6). This computational experiment directly confirms the Julia–Mandelbrot correspondence.

Julia sets of Newton functions applied to the polynomials $f(z) = z^n + c$ and their zeros' basins of attraction.

A study conducted in Kneisl (2001), examined the basins of attraction for the attracting fixed points of the complex function $f(z) = z^3 - 1$ using the super-Newton, Cauchy's, and Halley's methods. In this paper, we use the Newton method to create the Newton basins of attraction for $f(z) = z^n + c$. Afterward, we will

examine and compare them by analyzing their basins of attraction using Newton functions for higher-order polynomials on the complex plane, with different values of c.

In this section, we will describe the steps in constructing the basin of attraction from $f(z) = z^n + c$ using Newton's function $N(z)$. These functions have been constructed to converge to the nth roots of unity. To begin, we'll set up the drawing area on the computer, by creating a grid of 1000 columns and 1000 rows, the total 1000.000 points, representing the area of the complex plane $\mathbb{R} \times \mathbb{R} = [-2, 2] \times [-2, 2]$.

The ratio of width and height of the drawing area is 1:1. Each point on the grid acts as the starting point (z_0) in the sequence $z_{k+1} = N(z_k)$. Next, calculate the distance between z_k and the roots of $f(z)$. If the distance is less than a small constant $\epsilon = 0.001$, it is concluded that the sequence converges to that particular root. The original grid point z_0 is assigned a color based on the root it converged to. If it does not converge in maximum 100 iteration, this step will proceed to the next grid point. Through this method, a specific color will be designated to the basin of attraction $W(z_n)$ associated with each root of unity z_n.

Algorithm 4 provides a way to draw basins of attraction using Newton's function, as described in the paragraph above.

Algorithm 4: Basin of Attraction using Newton Methods

Input: Function $f(z) = z^n + c, f'(z), \epsilon$ as acceptable error tolerance , m as maximum of iteration, $r(1), \ldots, r(n)$ is the exact roots of f(z)

Output: The result of plotting dots $Z_1, Z_2, \ldots, Z_n$ based on the color of roots

```
for k ← 1 to numel(x_0) do
    Z_0 ← x_0; for i ← 1 to m do
        Z_s ← Z_0 − f(z)/f'(z);
        if |Z_s − Z_0| < ϵ then break;
        Z_0 ← Z_s;
    end
    Z_k(k) ← Z_0;
end
for j ← 1 to n do
    Z_n ← |Z_k − r(n)| < ϵ;
end
Z_{n+1} ← ¬(Z_1 + Z_2 + ··· + Z_n);
Z ← (Z_1 + Z_2 + ··· + Z_{n+1});
plot(Z)
```

Next, a program computer using Matlab for drawing the basin of attraction with the iteration $N(z)$ applied to the Julia set form polynomials $f(z) = z^n + c$ with different constant values for each $n \in 2, 3, \cdots,$ can be seen in appendix II. For the output, we do a lot of investigation on variable n and constant c. In all cases, the roots of $f(z)$ are denoted by $z_{i,n}^*$ for all $i \in 1, 2, 3, \cdots$, the result of the program above is belo

Variation of n

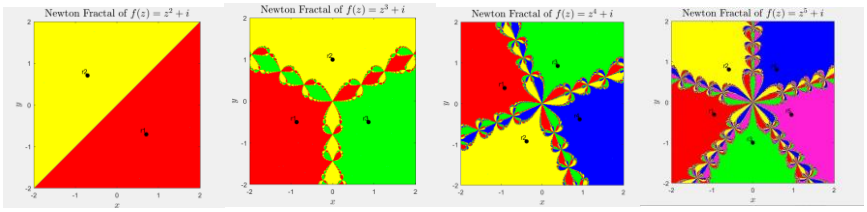

Figure 8. Newton basin of attraction $f(z) = z^n + i$, for $n = 2, 3, 4$, and 5

For the case $f(z) = z^n + i$, the number of basins of attraction that appear corresponds directly to the number of roots of the polynomial, namely n. This relation is visually confirmed in Figure 8, which illustrates the Newton basin basin attraction for $n = 2, 3, 4$ and 5.

For $n = 2$, two basin attraction emerge with a 180° rotational symmetry about the origin.

For $n = 3$, three basin attraction appear, distributed with 120° rotational symmetry.

For $n = 4$, the structure displays 90° rotational symmetry with four distinct basin attraction.

For $n = 5$, five basin attraction form a pentagonal star-like pattern with evenly spaced roots.

This outcome is consistent with the theoretical expectation: the solutions of $f(z) = 0$ are the n-th roots of $-c$, which are uniformly distributed on a circle in the complex plane, separated by an angle of $2\pi/n$. The Newton iteration inherits this rotational invariance, so the basin geometry reflects the same n-fold symmetry. Although some studies focus on the related form $z^n - c$, the symmetry properties remain equivalent up to a rotation or conjugation of the roots. Such rotational symmetries of Newton maps and their Julia sets have been discussed extensively in the literature (Hubbard et al., 2001; Campos et al., 2015; Nayak & Pal, 2022).

Additionally, the boundaries separating different basins exhibit a fractal nature, coinciding with the Julia set of the Newton function $N(z)$. This observation corroborates the theoretical result that the basin boundary of Newton's method coincides with the Julia set. The fractal geometry becomes increasingly intricate as n grows, reflecting the higher degree of the polynomial.

Variation of c

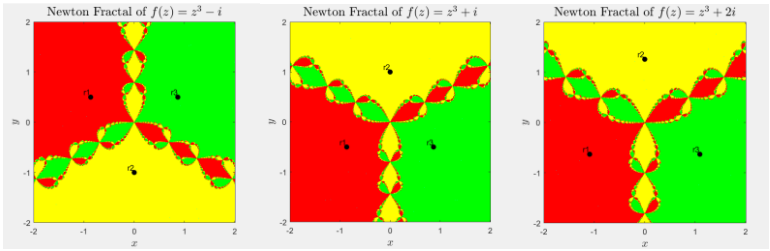

Figure 9. Newton basin of attraction $f(z) = z^3 + c$, for $c = -i, i$, and $2i$

To further investigate the role of the parameter c, we fix $n = 3$. In this case, the roots of $f(z) = z^3 + c$ are the cube roots of $-c$. Modifying c shifts the root locations in the complex plane, thereby changing the spatial distribution of basins of attraction.

For $c = -i$: the three roots are symmetrically distributed around the origin with 120° rotational symmetry, generating basins with clearly balanced patterns and fractal boundaries.

For $c = i$: the basins become the mirror image of those for $c = -i$ with respect to the real axis. This symmetry arises from the Schwarz reflection principle, which implies that the basin geometry for c and its conjugate $\bar{c}$ are reflections of each other. Thus,

$$Basin(c = i) \leftrightarrow Basin(c = -i)$$

The practical implication is that one can reduce computational effort by calculating basins for only one case, since the other can be obtained via reflection.

For $c = 2i$: the roots are displaced further along the imaginary axis, enlarging the basins and causing them to deform accordingly. Despite this deformation, the rotational symmetry of order three is preserved, reflecting the cubic structure of the polynomial.

Thus, parameter c influences both the scale and orientation of basins, while preserving the underlying rotational symmetry determined by n.

Variation Computational Parameters

Iteration Count (10 vs 50)

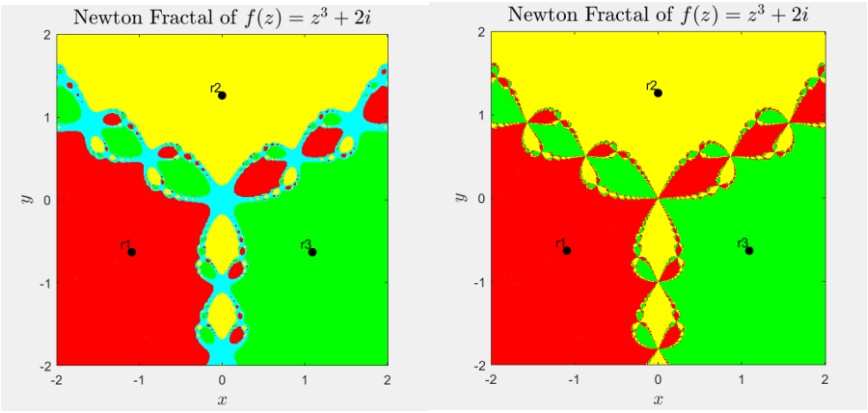

Figure 10. Newton basin of attraction for $f(z) = z^3 + 2i$ with maximum iteration count 10 (left) and 50 (right)

The number of iterations significantly affects the convergence near basin boundaries. With only 10 iterations, many points near boundaries fail to converge, producing regions with incomplete convergence, leading to blurred boundaries (Figure 10). This incomplete convergence obscures the fine fractal details. Increasing to 50 iterations eliminates most chaotic points, yielding sharper and

more accurate boundaries. However, this improvement comes at the cost of greater computational effort. The trade-off between convergence accuracy and efficiency is thus evident.

Convergence Tolerance (0.1 vs 0.01)

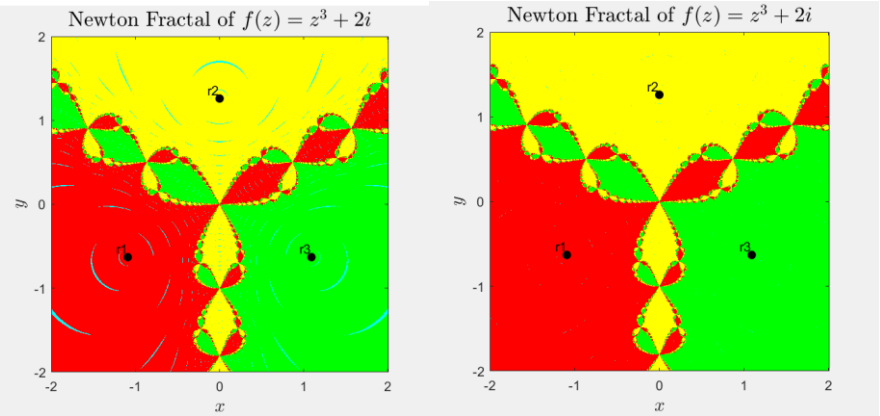

Figure 11. Newton fractal for $f(z) = z^3 + 2i$

The tolerance parameter also plays a critical role in shaping basin resolution. With a looser tolerance (0.1), iterations terminate prematurely, leading to lower precision in identifying root convergence. This results in noisy circular cyan regions around the roots, which may falsely appear chaotic. In contrast, a stricter tolerance (0.01) ensures convergence only when the iterates are sufficiently close to a root, thereby producing cleaner and more precise basin boundaries. This confirms that tolerance refinement enhances structural clarity, albeit at the expense of longer computational time.

Conclusion

This study demonstrates that C++ provides an effective framework for generating classical fractals, such as the Sierpinski triangle, Julia sets, and the Mandelbrot set, thereby verifying their self-similarity and coordinate transformation properties. In parallel, the analysis of Newton basins revealed symmetric structures that align with the theoretical distribution of polynomial roots, where the degree n dictates the number and symmetry of basins. Furthermore, the investigation confirmed the close relationship between Newton basins and Julia sets, as the basin boundaries coincide with the Julia set of the Newton function. These results establish a clear connection between numerical root-finding dynamics and fractal geometry. Finally, the study provides a baseline for further exploration of alternative iterative methods, such as Halley's or secant-based schemes, which may yield novel basin geometries and computational advantages.

Conflict of Interest

The author declares that there is no conflict of interest regarding the publication of this paper.

Acknowledgment

The author extends sincere appreciation to Dr. Rahma Zuhra, S.Si., M.Si. and Zahnur, S.Si., M.InfoTech for their constructive comments and valuable suggestions during the thesis examination, which contributed to improving the quality of this work. Finally, the author acknowledges the use of AI-based language tools (OpenAI's ChatGPT) to assist in refining the clarity, coherence, and readability of the manuscript. The intellectual content and interpretations, however, are entirely the author's own.

References

Abbas, M., Iqbal, H., & De La Sen, M. (2020). Generation of Julia and Mandelbrot Sets via Fixed Points. *Symmetry*, *12*(1), 86. https://doi.org/10.3390/sym12010086

Al-Shorman, A., Ajeel, M. S., & Al-Khaled, K. (2024). Analyzing Newton's Method for Solving Algebraic Equations with Complex Variables: Theory and Computational Analysis. *IAENG International Journal of Applied Mathematics*, *54*(6), 1038–1047.

Amat, S., Busquier, S., & Plaza, S. (2003). Dynamics of a family of third-order iterative methods that do not require the evaluation of second derivatives. *Applied Mathematics and Computation*, *146*(2–3), 435–443.

Cattani, C., Ben Mabrouk, A., & Arfaoui, S. (2022). *Fractal analysis: Basic concepts and applications* (pp. vii–viii). Singapore: World Scientific Publishing Co. Pte. Ltd.

Campos, B., Canela, J., & Vindel, P. (2022). Dynamics of Newton-like root finding methods. Numerical Algorithms, 91, 1105–1126.

Cordero, A., Ledesma, A., Maimó, J. G., & Torregrosa, J. R. (2024). Design and dynamical behavior of a fourth order family of iterative methods for solving nonlinear equations. *AIMS Mathematics*, *9*(4), 8564–8593. https://doi.org/10.3934/math.2024415

Devaney, R. L. (1989). *An introduction to chaotic dynamical systems* (2nd ed.). Addison-Wesley.

Farris, S. M. (2022). Generalized Mandelbrot sets of a family of polynomial $P_n(z) = z^n + z + c; (n \geq 2)$. *International Journal of Mathematics and Mathematical Sciences*, *2022*, 1–9. https://doi.org/10.1155/2022/4510088

Geyer, L. (2022). Classification of critically fixed anti-rational maps. *arXiv preprint arXiv:2006.10788*.

Hubbard, J. H., Schleicher, D., & Sutherland, S. (2001). *How to find all roots of complex polynomials by Newton's method*. Inventiones Mathematicae, 146(1), 1–33. https://doi.org/10.1007/s002220100162

Jnawali, J. (2021). New Modified Newton Type Iterative Methods. *Nepal Journal of Mathematical Sciences (NJMS)*, 2(1), 17–24.

Kanwar, V., Cordero, A., Torregrosa, J. R., Rajput, M., & Behl, R. (2023). A New Third-Order Family of Multiple Root-Findings Based on Exponential Fitted Curve. *Algorithms*, *16*(3), 156. https://doi.org/10.3390/a16030156

Kneisl, K. (2001). Julia sets for the super-Newton method, Cauchy's method, and Halley's method. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *11*(2), 359–370. https://doi.org/10.1063/1.1368137

Kwun, Y. C., Tanveer, M., Nazeer, W., Gdawiec, K., & Kang, S. M. (2019). Mandelbrot and Julia sets via Jungck–CR iteration With s-convexity. *IEEE Access*, *7*, 12167–12176. https://doi.org/10.1109/access.2019.2892013

Mehtre, V. V., & Singh, A. (2019). Newton–Raphson single and multiple variable methods to obtain the solutions of linear and non linear equations. *IRE Journals*, 3(6), 45–47.

Nayak, T., & Pal, S. (2024). *Julia sets of rational maps with rotational symmetries*. arXiv preprint arXiv:2402.07137. https://doi.org/10.48550/arXiv.2402.07137

Nutu, C. S., & Axinte, T. (2022). Fractals: origins and history. *Journal of Marine Technology and Environment*, *2*, 48–51. https://doi.org/10.53464/jmte.02.2022.08

Peitgen, H. O., Jürgens, H., & Saupe, D. (2012*). Fractals for the classroom: part two: Complex systems and Mandelbrot set*. Springer Science & Business Media.

Peitgen, H.-O., Jürgens, H., & Saupe, D. (2004). *Chaos and fractals: New frontiers of science* (2nd ed.). Springer.

Peitgen, H.-O., & Richter, P. H. (1986). *The beauty of fractals: Images of complex dynamical systems*. Springer.

Nwachioma, C., Perez-Cruz, J. H., Jimenez, A., Ezuma, M., & Rivera-Blas, R. (2019). A New Chaotic Oscillator—Properties, Analog Implementation, and Secure Communication Application. *IEEE Access*, *7*, 7510–7521. https://doi.org/10.1109/access.2018.2889964

Tasiu, A., Abbas, A., Alhassan, M., Umar, A., & Tasiu, R. (2020). Comparative study on some methods of handling nonlinear equations. *Anale. Seria Informatică*, *XVIII*(2), 52-53.

Wang, J., Li, X., & Sun, Y. (2021). Deep learning for classification of Newton fractal basins. *Chaos, Solitons & Fractals*, *145*, 110794.

Wiersma, A. (2016). *The Complex Dynamics of Newton's Method*. (Bachelor project in Mathematics, University of Groningen, The Netherlands).