Mikailalsys Journal of Advanced Engineering International

e-ISSN : 3046-9694
p-ISSN : 3046-8914

Index : Harvard, Boston, Sydney University, Dimensions, Lens, Scilit, Semantic, Google, etc

https://doi.org/10.58578/MJAEI.v3i2.9408

Julia and Mandelbrot Sets of Transcendental Cosine-Function Using Picard-Thakur Iteration Method

Babawuro Zuwaira[1], Manjak N. H[2], Kwami A. M[3], Adamu M. Y[4], Ismaila O. I[5]
[1,2,3,4]Abubakar Tafawa Balewa University Bauchi, Nigeria; [5]University of Maiduguri, Nigeria
zuwairab@fchdk.edu.ng

Article Info:

Submitted:	Revised:	Accepted:	Published:
Feb 11, 2026	May 9, 2026	May 21, 2026	May 26, 2026

Abstract

This study focuses on the generation and analysis of Julia and Mandelbrot sets for transcendental functions using the Picard–Thakur iterative scheme. It aims to examine the fractal structures produced by selected transcendental functions and investigate how parameter variations influence their topology. The study applied the Picard–Thakur iteration to generate fractal patterns and analyzed the resulting structures using the escape criterion. The findings indicate that parameter tuning produces significant transformations in fractal patterns, including the emergence of symmetrical and spiral-like structures. These results demonstrate the geometric complexity and dynamical sensitivity of transcendental Julia and Mandelbrot sets under the Picard–Thakur iterative scheme. The study concludes that the Picard–Thakur iteration provides a useful computational approach for exploring the behavior of fractal sets associated with transcendental functions. This research contributes to computational mathematics and dynamical systems by offering deeper insight into parameter-dependent fractal formation, with potential relevance to applied sciences involving nonlinear and complex dynamical structures.

Keywords: Transcendental Functions; Julia Set; Mandelbrot Set; Picard–Thakur Iteration; Fractals

Introduction

Fractals are intricate mathematical structures characterized by self-similarity, where patterns repeat themselves at varying scales to create complex and visually appealing geometries. These patterns are commonly observed in nature, such as in the branching of trees, the crystalline symmetry of snowflakes, and the arrangements of fern leaves. The generation of fractals is often achieved through iterative processes**,** in which a fundamental equation or rule is repeatedly applied to produce increasingly complex and self-replicating designs. The modern study of fractals was pioneered by Benoît Mandelbrot, who was the first to use computers to generate graphical representations of complex dynamical systems based on the polynomial and rational functions originally described by French mathematician Gaston Julia in the early 20th century (Garcia et al., 2009). One of the most renowned examples of fractal generation is derived from Mandelbrot's work (1975), in which he iterated the function:$f(z) = z^2 + c$, where c is a complex constant. Julia (1918) had earlier studied similar recursive mappings. Beyond this, many other fractals have been generated using different iterative schemes. In fractal theory, fixed point theory plays a central role, as it provides a framework for understanding the iterative processes that lead to fractal structures. Fixed points are specific values that remain unchanged under repeated iterations of a function. Within fractal dynamics, these points often act as attractors, pulling nearby points toward them during iteration. The behavior of a fractal is largely governed by escape criteria, which determine whether points in the complex plane remain bounded or diverge to infinity. For Julia and Mandelbrot **sets**, the notion of "escape" is fundamental: a point in the complex plane either belongs to the set or escapes to infinity. This is often visualized using the escape-time algorithm**,** where colors are assigned to points based on how quickly they escape. Rapidly diverging points are typically assigned bright colors, while slowly diverging or bounded points receive darker hues. The upper limit of iterations plays a critical role in determining the resolution and intricacy of the resulting fractal image. Fixed point iteration schemes are widely used to generate fractals with applications across numerous scientific disciplines. For example, fractals are applied in computer science

(Barnsley, 1993), biotechnology (Brouers & Sotolongo-Costa, 2006), engineering (Hardy & Beier, 1994), and physics (Thompson, 1991). Recent studies have expanded the field significantly. Tomar et al. (2023) investigated fractals arising from complex-valued cosine function $T(z) = \cos z^k + az + c$ by generating Julia and Mandelbrot sets using a four-step fixed-point iteration method enhanced with s-convexity. Their study, implemented in MATLAB, demonstrated how iteration parameters such as α,β,γ, δ and convexity parameter s, influence fractal dimensions, symmetry, color, and computation time. When the parameters a' and c' were real, the resulting fractals were symmetric about the initial line, however, for complex values, the fractals displayed twisted, intricate geometries. Similarly, Babawuro et al. (2022) explored Julia and Mandelbrot sets generated from the Gamma and Beta functions using the Lanczos approximation. Their findings showed that the Gamma function produced chaotic, self-repeating fractals with shapes that varied significantly with parameter λ. The Mandelbrot sets displayed intricate boundaries and smaller self-similar copies, emphasizing the recursive nature of these special functions. Zou et al. (2020) employed Picard-Mann iteration to generate Julia and Mandelbrot sets, observing distinct fractal images compared to those produced with the standard Mann orbit.

Danca and Feckan (2022) introduced Julia and Mandelbrot sets of fractional order using q^{th} Caputo-like discrete fractional difference equations for $q \in (0,1)$. Likewise, Bhoria et al. (2023) examined transcendental functions using the Picard–Thakur iteration scheme, an extension of the classical Picard iteration. They defined specific escape criteria and boundedness conditions to analyze transcendental sine and exponential functions in the complex plane. Their Mathematica 13.0 algorithms produced vivid, color-coded fractals, where the parameters $\tau_1, \tau_2, \tau_3, \tau_4$ significantly affected the color variations of the Julia sets for exponential functions. When these parameters were assigned complex values, quadratic Mandelbrot sets resembled the structure of traditional Mandelbrot sets. In another notable contribution, Danca (2024) demonstrated that a Mandelbrot set of integer order can be derived as a special case of Julia sets of Caputo-like fractional order, with algorithms adapted to generate equipotential lines and external rays for fractional-order fractals. Unlike integer-order cases, some of these lines were observed to intersect.

Blenker et al. (2019) extended the study of fractals to hyperbolic numbers, finding that Julia and Mandelbrot sets in this context are structurally simpler than those generated in the complex number setting. Amobeda et al. (2016) generated Julia and Mandelbrot sets for

the Gamma function using the iterative function $f_\lambda(z) = \Gamma(z)^2 + \lambda$, showing that the fractal behaviors observed in elementary functions also emerge in the Gamma function.

Building on these foundations, this research focuses on visualizing Julia and Mandelbrot sets for a complex transcendental function using the Picard–Thakur iterative procedure. The study establishes appropriate escape radii and employs a four-step iteration scheme to generate stunning, high-resolution fractals that highlight the dynamical behaviors of these transcendental functions.

Definition 1. (Julia set) The set of all point which has a bounded orbit under *f* is refer to as filled-in Julian set of f and is denoted by B_λ .

Mathematically,

$$B_\lambda = \{z : | f_\lambda^n(z) | \neq \infty \ \ as \ \ n \to \infty\}.$$

The boundary of the filled-in Julian set denoted by J_λ is called Julian set of *f*.

Definition 2 (Mandelbrot set) The Mandelbrot set *M* of some complex map $f_\lambda(z)$, where $\lambda \in C$,

is the set of values of the complex parameter λ for which, $f_\lambda^n(z) \neq \infty$, *as* n $\longrightarrow \infty$.

Mathematically expressed as; $M = \{\lambda : \lambda \in C, \lim_{n\to\infty} f_\lambda^n(z) \neq \infty \text{ as } n \to \infty\}$.

Definition 3. (The Picard orbit) Consider f to be a non-empty set and $f : C \to C$. Picard's orbit for any initial point $z_0 \in C$ can be defined as the set of all iterates of z_0, denoted as $O(f, z_0) = \{z_n : z_n = f(z_{n-1}), n = 1,2,3,...\}$. (Devany 1992).

Picard–Thakur orbits iteration procedure: Let C be a subset of complex numbers, and $f_c : C \to C$ is a mapping. For the initial point $z_0 \in C$, the Picard–Thakur orbit is defined as

$$\begin{aligned} z_{n+1} &= f_c(x_n) \\ x_n &= (1-\alpha_1) f_c(\lambda_n) + \alpha_1 f_c(y_n) \\ y_n &= (1-\beta_1)\lambda_n + \beta_1 f_c(\lambda_n) \\ \lambda_n &= (1-\gamma_1) z_n + \gamma_1 f_c(z_n) \end{aligned}$$

Where $\alpha_1, \beta_1, \gamma_1 \ \in \ (0,1)$.

The above sequence of iterates given by the Equation above is known as the Picard–Thakur orbit, which can be written as PTO ($f_c, z_n, \alpha_1, \beta_1, \gamma_1$).

Escape Criteria for Complex Cosine Functions The escape criterion performs an essential role in generating and analyzing Julia sets as well as Mandelbrot sets and their variants. We develop the following escape criteria for the complex-valued cosine functions to explore and compare the new mutants of Julia and Mandelbrot sets via Picard Thakur iteration method.

Theorem 1

Let $f_c(z) = \cos(z^r) + bz + c$ be a complex transcendental function and

$$|z| \geq |c| > \max\left\{\left(\frac{2+|b|}{\gamma_1|a||\lambda_1|}\right)^{\frac{1}{r-1}}, \left(\frac{2+|b|}{\beta_1|a||\lambda_2|}\right)^{\frac{1}{r-1}}, \left(\frac{2+|b|}{|a|(\alpha_1|\lambda_3|-|\lambda_2|}\right)^{\frac{1}{r-1}}, \left(\frac{2+|b|}{|a||\lambda_4|}\right)^{\frac{1}{r-1}}\right\},$$

where $0 < \alpha_1, \beta_1, \lambda_1 < 1$ and $c \in C$ Define

$$\begin{aligned} z_{n+1} &= f_c(x_n) \\ x_n &= (1-\alpha_1)f_c(\lambda_n) + \alpha_1 f_c(y_n) \\ y_n &= (1-\beta_1)\lambda_n + \beta_1 f_c(\lambda_n) \\ \lambda_n &= (1-\gamma_1)z_n + \gamma_1 f_c(z_n) \end{aligned} \tag{1}$$

Then $|z_n| \to \infty$ as $n \to \infty$

Proof: for $f_c(z) = a\cos(z^r) + bz + c$ and $n = 0$,

$$\begin{aligned} |\lambda_n| &= |(1-\gamma_1)z_n + \gamma_1 f_c(z_n)| \quad \text{From equation (1)} \\ &= |(1-\gamma_1)z_n + \gamma_1 f_c(z_n)| \\ &= |(1-\gamma_1)z_n + \gamma_1(a\cos(z_n^r) + bz_n + c)| \\ &\geq |\gamma_1(a\cos(z_n^r) + bz_n) + (1-\gamma_1)z_n| - |\gamma_1 c| \\ &\geq \gamma_1|a||\cos(z_n^r)| - \gamma_1|b||z_n| - |z_n| + \gamma_1|z_n| - \gamma_1|c| \end{aligned}$$

Now, by using $|z| \geq |c|$ and taking $z_0 = z, x_0 = x, y_0 = y, \lambda_0 = \lambda$, we have

$$|\lambda| \geq \gamma_1|a||\cos z^r| - \gamma_1|b||z| - |z| + \gamma_1|z| - \gamma_1|c|$$

Since $\cos z^r = \sum(-1)^m \frac{z^{r(2m)}}{(2m)!}$

$$\Rightarrow |\cos z^r| = \left|\sum_{m=0}^{\infty}(-1)^m \frac{z^{r(2m)}}{(2m)!}\right|$$

$$\geq \left|z^r\right|\left|\sum_{m=1}^{\infty}(-1)^m \frac{z^{r(2m-1)}}{(2m)!}\right| \geq \left|z^r\right|\left|\lambda_1\right|$$

Where $|\lambda_1| \in (0,1)$ and $|\lambda_1| \leq \left|\sum_{m=1}^{\infty}(-1)^m \frac{z^{r(2m-1)}}{(2m)!}\right|$

$$|\lambda| \geq \gamma_1|a|\left|z^r\right|\left|\lambda_1\right| - \gamma_1|b||z| - |z| + \gamma_1|z| - \gamma_1|c|$$

Since $|z| \geq |c|$, we can replace $|c|$ with $|z|$ in the above inequality so that,

$|\lambda| \geq \gamma_1|a|\left|z^r\right|\left|\lambda_1\right| - \gamma_1|b||z| - |z| + \gamma_1|z| - \gamma_1|z| = \gamma_1|a|\left|z^r\right|\left|\lambda_1\right| - \gamma_1|b||z| - |z|$ to obtain

$$|\lambda| \geq \gamma_1|a|\left|z^r\right|\left|\lambda_1\right| - |b||z| - |z| \quad = |z|(\gamma_1|a|\left|z^{r-1}\right|\left|\lambda_1\right| - |b| - 1).$$

Hence, $|\lambda| \geq |z|(\gamma_1|a|\left|z^{r-1}\right|\left|\lambda_1\right| - |b| - 1)$

since $|z| > \left(\frac{2+|b|}{\gamma_1|a|\left|\lambda_1\right|}\right)^{\frac{1}{r-1}}$ implies that $\gamma_1|a|\left|z^{r-1}\right|\left|\lambda_1\right| > 2 + |b|$, and so $\gamma_1|a|\left|z^{r-1}\right|\left|\lambda_1\right| - |b| - 1 > 1$.

Therefore, $|\lambda| > |z|$ (2)

Now, in the second step we have

$$|y_n| = |(1-\beta_1)\lambda_n + \beta_1 f_c(\lambda_n)|$$

$$= \left|(1-\beta_1)\lambda + \beta_1(a\cos(\lambda^r) + b\lambda + c)\right| \text{ if we consider } y_0 = y, \lambda_0 = \lambda, n = 0$$

$$\geq \left|\beta_1(a\cos(\lambda^r) + b\lambda) + (1-\beta_1)\lambda\right| - |\beta_1 c|$$

$$\geq \left|\beta_1(a\cos(\lambda^r) + b\lambda)\right| - (1-\beta_1)|\lambda| - |\beta_1 c|$$

$$\geq \beta_1|a|\left|\lambda^r\right| - |\lambda_2| - \beta_1|b||\lambda| - |\lambda| + \beta_1|\lambda| - \beta_1|c|$$

By using $|\lambda| > |z|$ we obtain, $|y| \geq \beta_1|a|\left|z^r\right|\left|\lambda_2\right| - \beta_1|b||z| - |z| \geq \beta_1(|a|\left|z^r\right|\left|\lambda_2\right| - |b||z| - |z|)$

$$= |z|(\beta_1|a|\left|z^{r-1}\right|\left|\lambda_2\right| - |b| - 1).$$

This implies that, $\beta_1|a|\left|z^{r-1}\right|\left|\lambda_2\right| > 2 + |b|$, so $\beta_1|a|\left|z^{r-1}\right|\left|\lambda_2\right| - |b| - 1 > 1$.

Therefore, $|y| > |z|$. (3)

In third step we have

$$x_n = (1-\alpha_1)f_c(\lambda_n) + \alpha_1 f_c(y_n)$$

$$= \left|(1-\alpha_1)(a\cos(\lambda^r)+b\lambda+c)+\alpha_1(a\cos(y^r)+by+c)\right|$$

$$\geq \alpha_1\left|a\cos(y^r)+by\right|-\alpha_1|c|-(1-\alpha_1)\left|a\cos(\lambda^r)+b\lambda\right|-(1-\alpha_1)|c|$$

$$\geq \alpha_1|a|\left|\cos(y^r)\right|-\alpha_1|b||y|-\alpha_1|c|-(1-\alpha_1)|a|\left|\cos(\lambda^r)\right|-(1-\alpha_1)|b||\lambda|-(1-\alpha_1)|c|$$

$$\geq \alpha_1|a|\left|\cos(y^r)\right|-\alpha_1|b||y|-\alpha_1|c|-|a|\left|\cos(\lambda^r)\right|+\alpha_1|a|\left|\cos(\lambda^r)\right|-|b||\lambda|+\alpha_1|b||\lambda|-|c|+\alpha_1|c|$$

By neglecting $\alpha_1|a|\left|\cos(\lambda^r)\right|$ and using (2) , (3) and $|z|\geq|c|$, we obtain

$$|x| \geq \alpha_1|a|\left|z^r\right|\lambda_3|-\alpha_1|b||z|-\alpha_1|z|-|a|\left|z^r\right|\lambda_2|-|b||z|+\alpha_1|b||z|-|z|+\alpha_1|z|$$

$$= \alpha_1|a|\left|z^r\right|\lambda_3|-|a|\left|z^r\right|\lambda_2|-|b||z|-|z|$$

$$= |z|(\alpha_1|a|\left|z^{r-1}\right|\lambda_3|-|a|\left|z^{r-1}\right|\lambda_2|-|b|-1\ ).$$

since $|z|>\left(\dfrac{2+|b|}{|a|(\alpha_1|\lambda_3|-|\lambda_2|)}\right)^{\frac{1}{r-1}}$ implies that $\left|z^{r-1}\right|a|(\alpha_1|\lambda_3|-|\lambda_2|>2+|b|\ )$, so $\alpha_1|a|\left|z^{r-1}\right|\lambda_3|-|a|\left|z^{r-1}\right|\lambda_2|-|b|-1>1$.

therefore $|x|>|z|$.

Now, in the last step, for $z_{n+1}=f_c(x_n)$, we have

$$|z_1| = |f_c(x_0)|$$

$$= \left|a\cos(x^r)+bx+c\right|$$

$$\geq \left|a\cos(x^r)+bx\,|-|\,c\right|$$

$$\geq \left|a|u^r|\right|\lambda_4|-|b\,||\,u\,|-|\,z|$$

$$\geq \left|a|z^r|\right|\lambda_4|-|b\,||\,u\,|-|\,z|$$

$$\geq |z|(|a|\left|z^{r-1}\right|\lambda_4|-|b|-1)\ .$$

When applying similar arguments repeatedly, we obtain

$$|z_2| \geq |z|(|a|\left|z^{r-1}\right|\lambda_4|-|b|-1)^2$$

$$|z_3| \geq |z|(|a|\left|z^{r-1}\right|\lambda_4|-|b|-1)^3$$

$$|z_n| \geq |z|(|a|\left|z^{r-1}\right|\lambda_4|-|b|-1)^n$$

since $|z| > \left(\frac{2+|b|}{|a||\lambda_4|}\right)^{\frac{1}{r-1}}$ implies that $|a||z^{r-1}||\lambda_4| > 2+|b|$ and $|a||z^{r-1}||\lambda_4| - |b| - 1 > 1$, hence we have

$|z_n| \to \infty$ as $n \to \infty$.

Corollary 1: Assume that $|z_j| > \max\left\{|c|, \left(\frac{2+|b|}{\gamma_1|a||\lambda_1|}\right)^{\frac{1}{r-1}}, \left(\frac{2+|b|}{\beta_1|a||\lambda_2|}\right)^{\frac{1}{r-1}}, \left(\frac{2+|b|}{|a|(\alpha_1|\lambda_3|-|\lambda_2|)}\right)^{\frac{1}{r-1}}, \left(\frac{2+|b|}{|a||\lambda_4|}\right)^{\frac{1}{r-1}}\right\}$. Then $|z_{j+1}| \geq (1+\lambda_1)^k|z_j|$ and $|z_{j+1}| \to \infty$ as $k \to \infty$.

Remark 1: Because of its vibrant and understandable color gradation, the rainbow colormap which covers the entire spectrum from red to violet has been frequently employed to visualize fractals. The rainbow colormap in Python's Matplotlib is used in this study to color Mandelbrot and Julia sets, even though jet colormaps are frequently found in programs like MATLAB. The escape dynamics of points under iterative transformations are effectively illustrated by this colormap, which assigns low iteration counts cooler colors (like blue and cyan). The outcome is visually instructive and fits in nicely with methods used in current fractal research. Figure 1 provides an example, where the rainbow color scheme is used to represent the Mandelbrot set created using the transformation $f_c(z) = a\cos(z^r) + bz + c$.

Remark 2: Even though both Julia sets and Mandelbrot sets are made by repeating complex number functions and look similar with their detailed shapes, they are not the same. The main difference is how the process starts and how the value of c is used. In the Mandelbrot set (as we did in our Python code), we always start with $z_0 = 0$, and the value of c changes across the complex plane to check which values make the sequence stay within limits. But for the Julia set, the value of c is fixed, and instead we change the starting point z_0 to see which points will escape or stay. This difference makes the patterns of the sets look different. Julia sets show the behavior of each point for one c, while the Mandelbrot set gives an overview of how Julia sets behave. Our Python code shows this clearly in Algorithm 1 and 2, which we used for the Mandelbrot and Julia sets.

Algorithm: We used Algorithm 1 (Geometry of Mandelbrot set) and Algorithm 2 (Geometry of Mandelbrot set), for sketching fractals for complex cosine functions using picard thakur iteration process via PYTHON 3.12.

Algorithm 1 The Julia set

Setup:

Take a complex number $c = u + iv$.

Initialize the variables $\alpha_1, \beta_1, \gamma_1, \lambda_1, \lambda_2, \lambda_3, \lambda_4, a, b$

Consider first iteration $z_0 = x + iy$.

Iterate:

$$z_{n+1} = f_c(x_n);$$
$$x_n = (1-\alpha_1) f_c(\lambda_n) + \alpha_1 f_c(y_n);$$
$$y_n = (1-\beta_1)\lambda_n + \beta_1 f_c(\lambda_n);$$
$$\lambda_n = (1-\gamma_1) z_n + \gamma_1 f_c(z_n); n \geq 0,$$

Where $f_c(z) = a\cos(z^r) + bz + c$ $\boldsymbol{r} = 2, 3, 4, \ldots, 0 < \alpha_1, \beta_1, \gamma_1 < 1,\ 0 < \lambda_1, \lambda_2, \lambda_3, \lambda_4 \leq 1.$

Stop:

$$|\boldsymbol{z_n}| > \textbf{Escape radius} = \max\left\{|\boldsymbol{c}|, \left(\frac{2+|\boldsymbol{b}|}{\boldsymbol{\gamma_1}|\boldsymbol{a}||\boldsymbol{\lambda_1}|}\right)^{\frac{1}{r-1}}, \left(\frac{2+|\boldsymbol{b}|}{\boldsymbol{\beta_1}|\boldsymbol{a}||\boldsymbol{\lambda_2}|}\right)^{\frac{1}{r-1}}, \left(\frac{2+|\boldsymbol{b}|}{|\boldsymbol{a}|(\boldsymbol{\alpha_1}|\boldsymbol{\lambda_3}|-|\boldsymbol{\lambda_2}|)}\right)^{\frac{1}{r-1}}, \left(\frac{2+|\boldsymbol{b}|}{|\boldsymbol{a}||\boldsymbol{\lambda_4}|}\right)^{\frac{1}{r-1}}\right\}.$$

Count:

The number of iterations undertaken to escape.

Color:

Assign a color to each point based on the number of iterations needed to escape.

Algorithm 2 The Mandelbrot set

Setup:

Take a complex number $c = u + iv$.

Initialize the variables $\alpha_1, \beta_1, \gamma_1, \lambda_1, \lambda_2, \lambda_3, \lambda_4, a, b$

Consider first iteration $z_0 = c$.

Iterate:

$$z_{n+1} = f_c(x_n);$$
$$x_n = (1-\alpha_1)f_c(\lambda_n) + \alpha_1 f_c(y_n);$$
$$y_n = (1-\beta_1)\lambda_n + \beta_1 f_c(\lambda_n);$$
$$\lambda_n = (1-\gamma_1)z_n + \gamma_1 f_c(z_n); n \geq 0,$$

Where $f_c(z) = a\cos(z^r) + bz + c$ $\boldsymbol{r = 2, 3, 4, \ldots, 0 < \alpha_1, \beta_1, \gamma_1 < 1,\ 0 < \lambda_1, \lambda_2, \lambda_3, \lambda_4 \leq}$

Stop: $|z_n| >$ **Escape radius = max** $\left\{|c|,\ \left(\frac{2+|b|}{\gamma_1|a||\lambda_1|}\right)^{\frac{1}{r-1}}, \left(\frac{2+|b|}{\beta_1|a||\lambda_2|}\right)^{\frac{1}{r-1}}, \left(\frac{2+|b|}{|a|(\alpha_1|\lambda_3|-|\lambda_2|)}\right)^{\frac{1}{r-1}}, \left(\frac{2+|b|}{|a||\lambda_4|}\right)^{\frac{1}{r-1}}\right\}$.

Count:

The number of iterations undertaken to escape.

Color:

Assign a color to each point based on the number of iterations needed to escape.

The Mandelbrot set we got from the formula $f_c(z) = a\cos(z^r) + bz + c$. Shows a clear balance along the real (horizontal) axis, as seen in Figure 1. This means that if a point $c = x + iy$ is in the set, its opposite along the y-axis, $z_0 = x - iy$ is also in the set. This happens because the cosine function is an even function, means $\cos(z) = \cos(-z)$. Even though the vertical (imaginary axis) symmetry is not perfect, the shape still looks fairly balanced on both sides because of the way the cosine and real number values behave in the formula. The fractal shows neat shapes in the middle, some mirror-like balance, and repeating thread-like patterns which are all common in complex functions like this. These patterns help confirm that, the transcendental cosine function using Picard–Thakur iteration process the set still keeps its beautiful and regular structure.

Mandelbrot sets for $f_c(z) = a\cos(z^r) + bz + c$.

In this part of the work, we use the escape time method to build the Mandelbrot set using the cosine-based function, with a maximum of 10 iterations, depending on the values of the parameters used. The process for generating the Mandelbrot set is shown in Algorithm 1.

Case (1): For the function $f_c(z) = a\cos(z^r) + bz + c$. We used different sets of parameters, which are listed in Table 1.

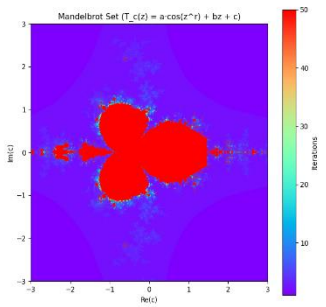

Figure 1.1

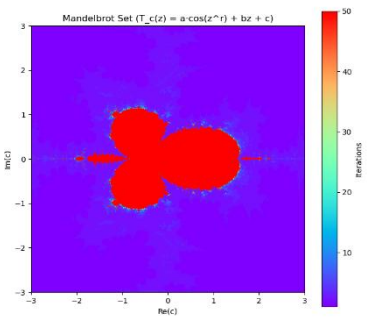

Figure 1.2

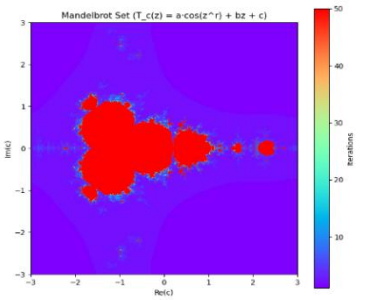

Figure 1.3

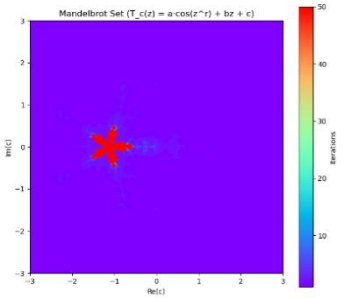

Figure 1.4

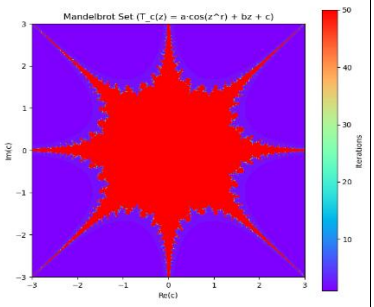

Figure 1.5

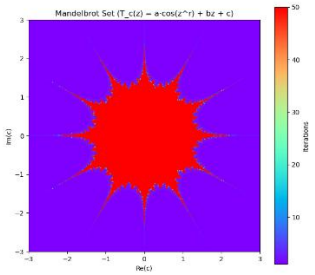

Figure 1.6

Table 1

Figures 1	a	b	r	α_1	β_1	γ_1	τ_1	τ_2	τ_3	τ_4
Figure 1.1	0.4	0.4	2	0.2	0.2	0.2	0.8	0.5	0.7	0.7
Figure 1.2	0.4	0.4	2	0.8	0.8	0.8	0.08	0.05	0.07	0.07
Figure 1.3	1	0	2	0.2	0.2	0.2	0.8	0.5	0.7	0.7
Figure 1.4	1.14	0.9	3	0.9	0.9	0.9	0.08	0.05	0.07	0.07
Figure 1.5	0.0014	0.0009	4	0.009	0.009	0.009	0.08	0.05	0.07	0.07
Figure 1.6	0.0014	0.0009	6	0.009	0.009	0.009	0.08	0.05	0.07	0.07

In Figure 1i to 1iii, we show Mandelbrot sets made from a cosine-based function using the Picard–Thakur method. For these figures, we kept the power of the function as $r = 2$r but changed the values of the parameters a, b and other Parameters $\boldsymbol{\alpha_1, \beta_1, \gamma_1, \lambda_1, \lambda_2, \lambda_3, and\ \lambda_4}$. These changes gave different shapes, but all the sets still have a balanced look along the x-axis and have two main lobes (arms). In Figure 1iv, we increased the degree to $r = 3$, and made all the feedback parameters the same. This made the shape balanced on both x- and y-axes. In Figure 1v and 1vi, we used higher powers

$r = 4\ and\ r = 6$, and the shapes became more complex. We noticed that the sets now have $2r$ arms (like spokes of a wheel), spreading out from the center. This happened because increasing the power r in the function $f_c(z) = a\cos(z^r) + bz + c$ .brings more regular repeating patterns and symmetry.

Each of these arms shows a direction where the function behaves in a steady way. The arms are spread equally around the middle, and the angle between them is about $\frac{m\pi}{r}$, where m is the number of the arm. This round symmetry comes from the cosine function, which naturally repeats, and the power r, which splits the shape into equal parts. So, for r = 4, we see 8 arms, and for r = 6, we see 12 arms, all arranged evenly like petals of a flower.

Case 2. The effect of parameter r, as given in Table 2 which are symmetrical with regard to the initial line (as can be seen in Figure 2). We notice that, as the value of the parameter r increases, the time taken to generate it decreases.

Table 2

Figures	a	b	r	α_1	β_1	γ_1	τ_1	τ_2	τ_3	τ_4
Figure 2.1	0.0014	0.0009	2	0.009	0.009	0.009	0.08	0.05	0.07	0.07
Figure 2.2	0.0014	0.0009	3	0.009	0.009	0.009	0.08	0.05	0.07	0.07
Figure 2.3	0.0014	0.0009	4	0.009	0.009	0.009	0.08	0.05	0.07	0.07
Figure 2.4	0.0014	0.0009	8	0.009	0.009	0.009	0.08	0.05	0.07	0.07
Figure 2.5	0.0014	0.0009	10	0.009	0.009	0.009	0.08	0.05	0.07	0.07
Figure 2.6	0.0014	0.0009	20	0.009	0.009	0.009	0.08	0.05	0.07	0.07

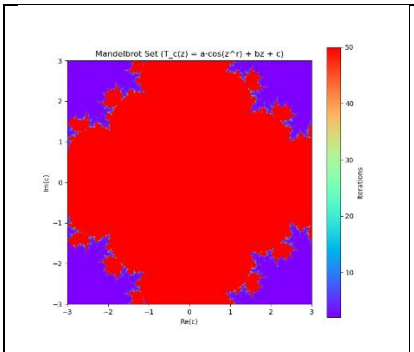

Figure 2.1

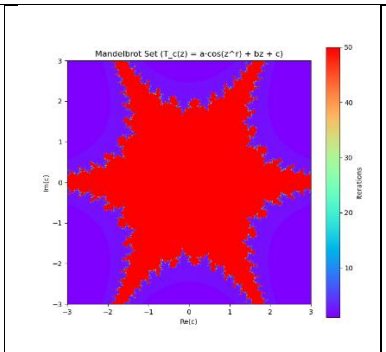

Figure 2.2

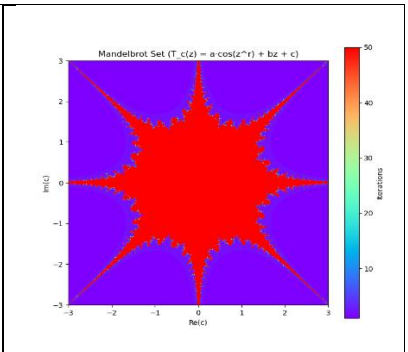

Figure 2.3

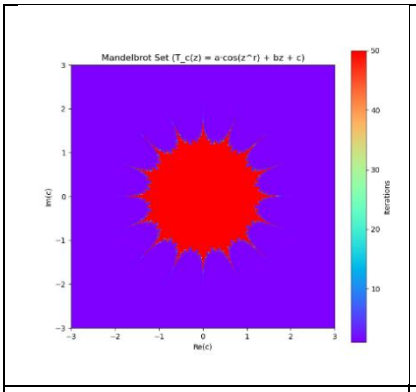

Figure 2.4

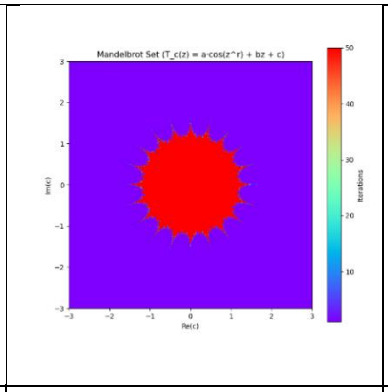

Figure 2.5

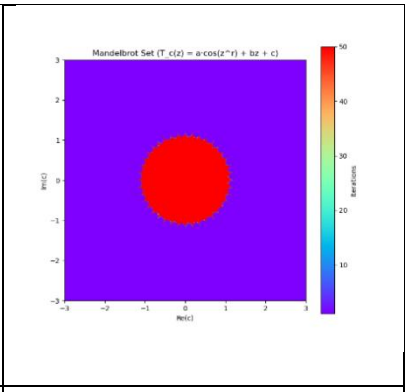

Figure 2.6

In Figure 2.1–2.3, Mandelbrot sets are generated using the cosine-based function with the Picard–Thakur method. The values of parameters a = 0.0014 and b = 0.0009 are fixed, but the power r is increased from 2 to 4. As r increases, the number of visible arms or branches in the sets also increases. For example, in Figure 2.1 with r = 2, we see 4 main lobes, while Figure 2.3 with r = 4 shows 8 clear lobes. All parameter settings like $\alpha_1, \beta_1, \gamma_1 \; and \lambda_1, \lambda_2, \lambda_3, \; to \; \lambda_4$ are kept the same across these figures to isolate the effect of r.

In Figure 2.4 to 2.6, the power r is further increased to 8, 10, and 20 respectively. This increase leads to even more complex and beautiful patterns. We can see 2**r** lobes appearing and spreading evenly around the center point, forming designs that look like stars or petals. These patterns come from the mathematical nature of the cosine function, which repeats in a circular way, and the higher power r, which divides the circle into more equal parts. So, for r = 20, there are 40 thin lobes spaced out like rays of the sun.

This shows how changing just one parameter the power r can drastically affect the shape and symmetry of the Mandelbrot set, even when all other parameters remain the same.

Case (3): When a and b are complex numbers. When complex values are assigned to the parameters 'a' and 'b', the quadratic Mandelbrot set, as shown in Figure3

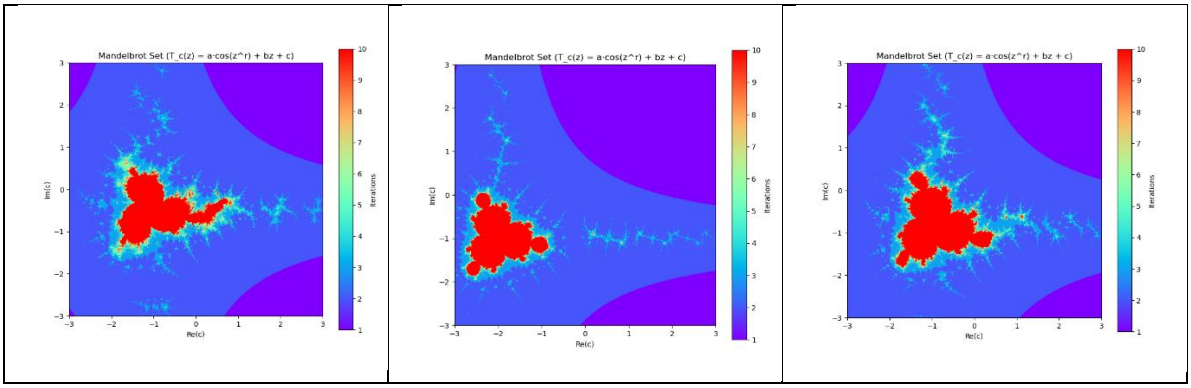

Figure 3.1	Figure 3.2	Figure 3.3
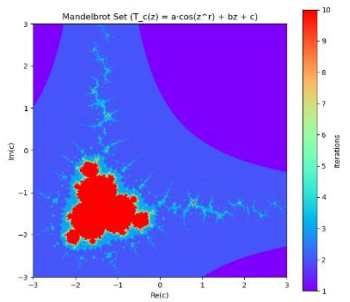	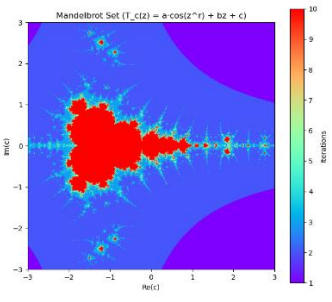	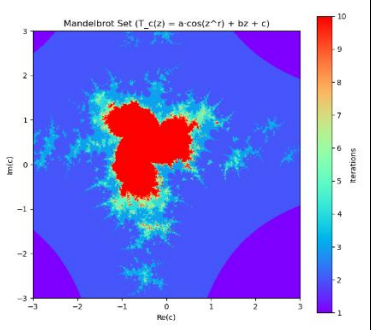
Figure 3.4	Figure 3.5	Figure 3.6

Table 3.

Figures	a	b	r	α_1	β_1	γ_1	τ_1	τ_2	τ_3	τ_4
Figure 3.1	1.10+0.50i	0.30+0.20i	2	0.2	0.2	0.2	0.8	0.5	0.7	0.7
Figure 3.2	2.00+1.00i	0.10+0.00	2	0.8	0.8	0.8	0.08	0.05	0.07	0.07
Figure 3.3	0.95+0.85i	0.15+0.10i	2	0.2	0.2	0.2	0.8	0.5	0.7	0.7
Figure 3.4	1.40+1.40i	0.05+0.05i	2	0.9	0.9	0.9	0.08	0.05	0.07	0.07
Figure 3.5	1.20i	0.00i	2	0.009	0.009	0.08	0.05	0.07	0.07	0.07
Figure 3.6	0.50-0.50i	0.30+0.30i	2	0.009	0.009	0.009	0.08	0.05	0.07	0.07

We draw Mandelbrot sets using a cosine-based function $f_c(z) = a\cos(z^r) + bz + c$, and we follow the Picard Thakur method. For these figures, we fix the power $r = 2$, but change the values of a and b which are complex numbers. In Figure 3.1 the Mandelbrot set has a unique shape with three main lobes, and the third one stretches out with a long tail. This shape is similar to Figure 3.6, though that one looks more compact without the long tail. These two use complex values for a and b, which may cause that extra lobe to form. Figures 3.2, 3.3 and 3.4 all have similar structures showing balanced and well-formed shapes. The lobes are rounded and the structure is more stable, possibly because the feedback and expansion parameters are better balanced in those cases. Figure 3.5 is very different from the rest. It has a wavy and twisted look, and it stands out because both a and b are purely imaginary. This seems to affect the symmetry, the fractal stays joined and shows clear balance along the vertical (y) axis, this comes from using imaginary numbers making the shape, with a spinning style and several arms branching out. The full list of the values used is shown in **Table 3**.

Julia set for $f_c(z) = a\cos(z^r) + bz + c$

We used the Picard–Thakur method to see how changing different parameters changes the look of Julia sets for our cosine function.

Table 4

Figures	a	b	c	r	α_1	β_1	γ_1	τ_1	τ_2	τ_3	τ_4
Figure 4.1	1	0	-0.007j	2	0.01	0.02	0.03	0.07	0.08	0.09	0.09
Figure 4.2	1	0.5	-0.007j	2	0.01	0.02	0.03	0.07	0.08	0.09	0.09
Figure 4.3	1.7	0	-0.007j	2	0.01	0.02	0.03	0.07	0.08	0.09	0.09
Figure 4.4	0.8	0.02	0.0007-0.0007j	3	0.03	0.03	0.03	0.6	0.7	0.8	0.9
Figure 4.5	0.8	0.02	0.0007-0.0007j	3	0.5	0.5	0.5	0.6	0.7	0.8	0.9
Figure 4.6	0.8	0.02	0.0007-0.0007j	3	0.9	0.9	0.9	0.6	0.7	0.8	0.9

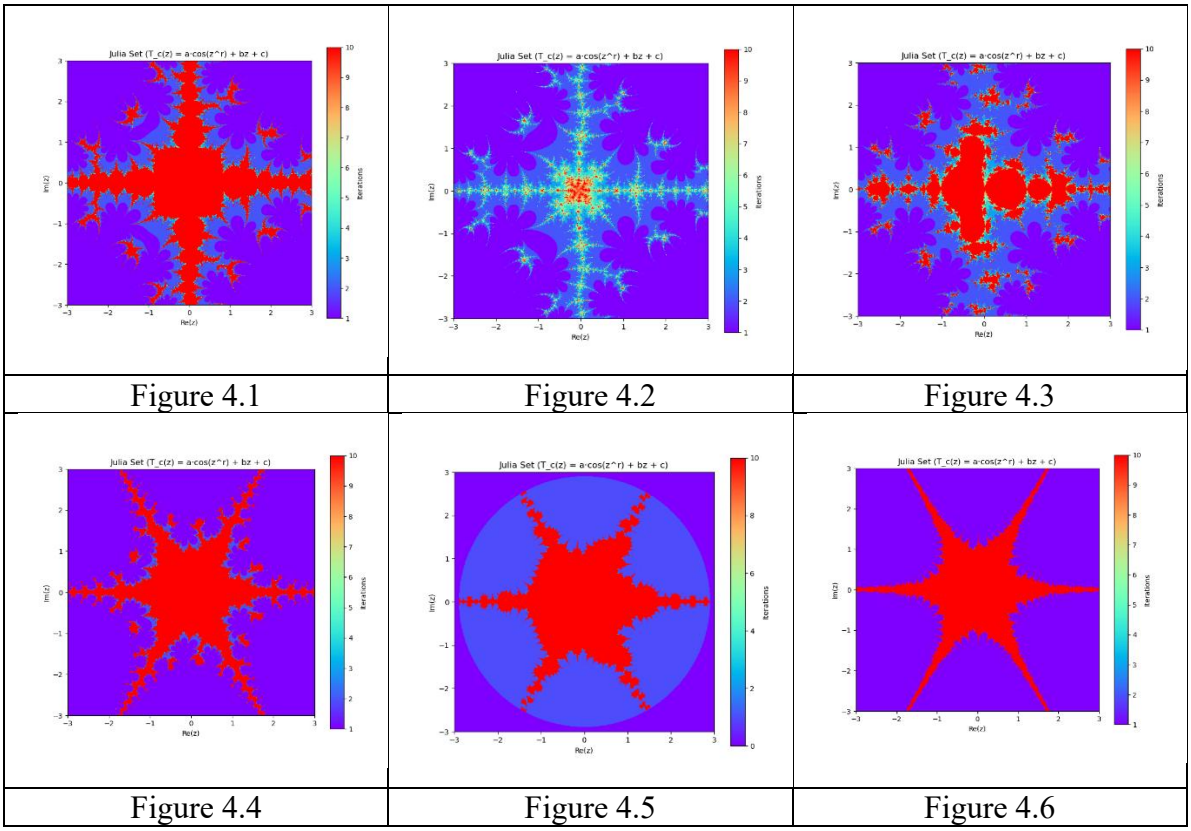

Figure 4.1 | Figure 4.2 | Figure 4.3

Figure 4.4 | Figure 4.5 | Figure 4.6

We used the Picard–Thakur method to see how changing different parameters changes the look of Julia sets for our cosine function.

For the quadratic case ($r = 2$), Figures 4.1–4.3 show what happens when we change a and b but keep the other parameters the same. In Figure 4.1 ($a = 1, b = 0$), the Julia set has four main attractors positioned at angles $\frac{mn}{\pi}$ two are symmetric across the real axis and the other two across the imaginary axis. Figure 4.2 When b is increased slightly to 0.5, the lobes open up more and the inner lines become clearer. Figure 4.3 Increasing a to 1.7 makes the pattern denser with shorter, tighter arms, meaning the iterations fold the pattern more strongly.

For the cubic case ($r = 3$), Figures 4.4–4.6 show the effect of changing $\alpha_1, \beta_1, \gamma_1$ while keeping $a = 0.8, \mathrm{b} = 0.02, \mathrm{c} = 0.0007 - 0.0007\mathrm{i}$. Figure 4.4, When these values are small ($\alpha_1 = \beta_1 = \gamma_1 = 0.03$, the Julia set has many thin, widely spaced "bunches" linked by fine threads. Medium values 0.5, Figure 4.5 reduce the number of bunches and make them more connected. High values 0.9, Figure 4.6 join the shape into a compact "starfish" with several arms placed evenly.

Table 5

Figures	a	b	c	r	α_1	β_1	γ_1	τ_1	τ_2	τ_3	τ_4
Figure 5.1	0.400	0.400	0.7474	2	0.0111	0.0056	0.0995	0.80	0.50	0.70	0.70
Figure 5.2	0.400	0.400	1.5000j	2	0.0111	0.0056	0.0995	0.80	0.50	0.70	0.70
Figure 5.3	0.400	0.400	-1.9000	2	0.0111	0.0056	0.0995	0.80	0.50	0.70	0.70
Figure5.4	0.400	0.400	0.5500+0.3800j	2	0.0111	0.0056	0.0995	0.80	0.50	0.70	0.70
Figure 5.5	0.400	0.400	0.2500+0.9000j	2	0.0111	0.0056	0.0995	0.80	0.50	0.70	0.70
Figure 5.6	0.400	0.400	-0.1500+1.100j	2	0.0111	0.0056	0.0995	0.80	0.50	0.70	0.70
Figure 5.7	0.400	0.400	-1.5800-0.2000j	2	0.0111	0.0056	0.0995	0.80	0.50	0.70	0.70
Figure 5.8	0.400	0.400	1.7800+0.2000j	2	0.0111	0.0056	0.0995	0.80	0.50	0.70	0.70
Figure 5.9	0.400	0.400	1.7000j	2	0.0111	0.0056	0.0995	0.80	0.50	0.70	0.70

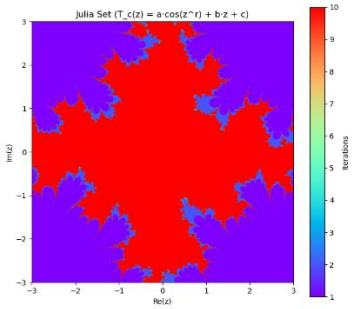

Figure 5.1

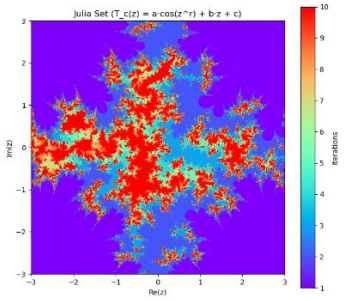

Figure 5.2

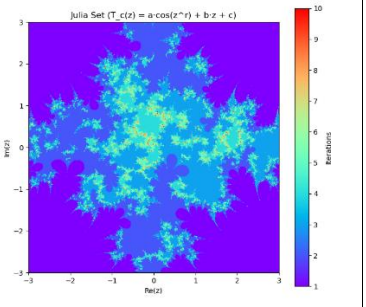

Figure 5.3

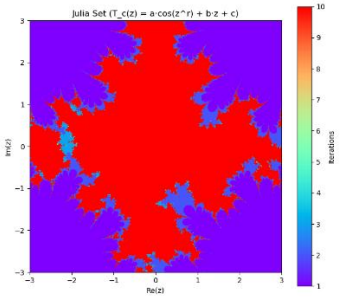

Figure 5.4

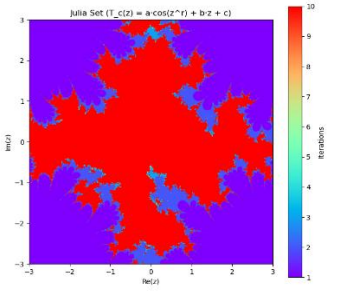

Figure 5.5

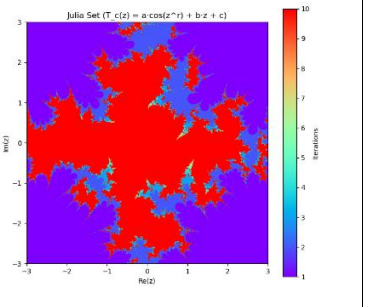

Figure 5.6

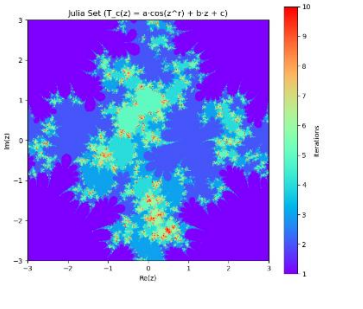

Figure 5.7

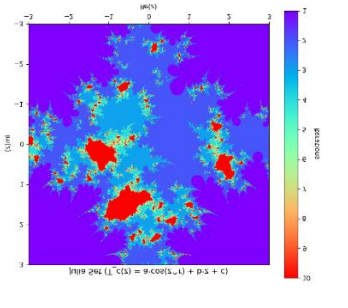

Figure 5.8

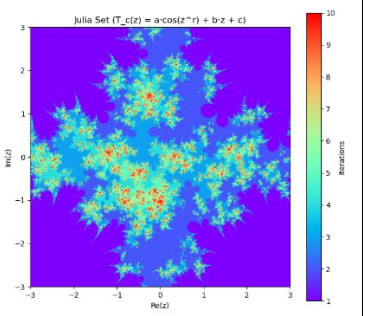

Figure 5.9

Figures 5.1–5.9 give more results for $r = 2$, with $a = b = 0.4$ but different complex values of c. A real $c = 0.7474$ in Figure 5.1 makes a smooth, classical quadratic Julia set. In Figure 5.2 a purely imaginary c stretches the pattern vertically, while Figure 5.3 a negative real c tilts and squeezes the lobes. In Figures 5.4–5.8 complex c values create many different shapes: the sets become more uneven some have open sides, while others form web-like or spiral shapes e.g., Figure 5.7. With purely imaginary $c = 1.7000i$ (Figure 5.9), the set has two vertical lobes joined by fine thread like lines.

Conclusion

The Picard–Thakur iteration scheme has proven to be a highly effective tool for exploring the complex geometries of Julia and Mandelbrot **sets** generated from transcendental functions. By systematically varying parameters within the cosine-associated function, a wide range of fractal morphologies were produced, demonstrating the sensitivity of these sets to initial conditions and iterative rules. The emergence of unique patterns, including symmetrical structures and shapes resembling natural objects, underscores the rich structural complexity inherent in transcendental dynamics. This study not only validates the flexibility and robustness of the Picard–Thakur method for analyzing complex systems but also highlights its potential applications in computational aesthetics**,** scientific modeling and chaos-based encryption systems. In conclusion, we suggest that the work can be extended to cover more transcendental functions such as $z^r + e^{c^t}$ to generate fascinating fractals for various dynamical images.

References

Amobeda, R. E., Aboiyar, T., Adee, S. O., & Ayoo, P. V. (2016). Julia and Mandelbrot sets of the gamma function using Lanczos approximation. *Applied and Computational Mathematics, 5*(2), 73–77. https://doi.org/10.11648/j.acm.20160502.16

Babawuro, Z., Manjak, N. H., Abba, A. S., Adam, H. Z., & Ibrahim, M. (2022). On Julia and Mandelbrot sets of gamma and beta functions for fractal images. *Science Forum (Journal of Pure and Applied Sciences), 22*(2), 227–233. https://bibliomed.org/?mno=127200

Barnsley, M. F. (1993). *Fractals everywhere* (2nd ed.). Academic Press.

Bhoria, A., Panwar, A., & Sajid, M. (2023). Mandelbrot and Julia sets of transcendental functions using Picard–Thakur iteration. *Fractal and Fractional, 7*(10), Article 768. https://doi.org/10.3390/fractalfract7100768

Blankers, V., Rendfrey, T., Shukert, A., & Shipman, P. D. (2019). Julia and Mandelbrot sets for dynamics over the hyperbolic numbers. *Fractal and Fractional, 3*(1), Article 6. https://doi.org/10.3390/fractalfract3010006

Brouers, F., & Sotolongo-Costa, O. (2006). Generalized fractal kinetics in complex systems: Application to biophysics and biotechnology. *Physica A: Statistical Mechanics and Its Applications, 368*(1), 165–175. https://doi.org/10.1016/j.physa.2005.12.062

Danca, M.-F. (2024). Mandelbrot set as a particular Julia set of fractional order, equipotential lines and external rays of Mandelbrot and Julia sets of fractional order. *Fractal and Fractional, 8*(1), Article 69. https://doi.org/10.3390/fractalfract8010069

Danca, M.-F., & Fečkan, M. (2023). Mandelbrot set and Julia sets of fractional order. *Nonlinear Dynamics, 111*(10), 9555–9570. https://doi.org/10.1007/s11071-023-08311-2

Hardy, H. H., & Beier, R. A. (1994). *Fractals in reservoir engineering*. World Scientific.

Julia, G. (1918). Mémoire sur l'itération des fonctions rationnelles. *Journal de Mathématiques Pures et Appliquées, 8*(1), 47–245.

Losa, G. A., Merlini, D., Nonnenmacher, T. F., & Weibel, E. R. (Eds.). (2002). *Fractals in biology and medicine* (Vol. 3). Birkhäuser.

Mandelbrot, B. B. (1975). *Les objets fractals: Forme, hasard et dimension*. Flammarion.

Thompson, A. H. (1991). Fractals in rock physics. *Annual Review of Earth and Planetary Sciences, 19*, 237–262. https://doi.org/10.1146/annurev.ea.19.050191.001321

Tomar, A., Kumar, V., Rana, U. S., & Sajid, M. (2023). Fractals as Julia and Mandelbrot sets of complex cosine functions via fixed point iterations. *Symmetry, 15*(2), Article 478. https://doi.org/10.3390/sym15020478

Zou, C., Shahid, A. A., Tassaddiq, A., Khan, A., & Ahmad, M. (2020). Mandelbrot sets and Julia sets in Picard–Mann orbit. *IEEE Access, 8*, 64411–64421. https://doi.org/10.1109/ACCESS.2020.2984689