ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

METODOS NUMÉRICOS (ICCD412)

FRACTAL: CONJUNTO DE MANDELBROT

Integrantes:

- Mateo Sebastián Cumbal Guasgua
- Daniel Ismael Flores Espín
- Johann Vladimir Pasquel Montenegro
- Luis Rafael Tipán Tandalla

DOCENTE: Ing. Jonathan A. Zea

ESCUELA POLITÉCNICA NACIONAL

ÍNDICE DE CONTENIDOS

INDICE DE FIGURAS

1. INTRODUCCIÓN

Los fractales son estructuras geométricas que exhiben un patrón de autosimilitud, lo que significa que sus componentes se repiten casi de manera idéntica a diferentes escalas. Este fenómeno, que se origina a partir de procesos iterativos, permite describir y analizar sistemas y formas complejas que se encuentran a menudo en la naturaleza, como las ramas de un árbol, la forma de una costa o los patrones de una nube. Uno de los

ESCUELA POLITÉCNICA NACIONAL

fractales más conocidos y estudiados es el Conjunto de Mandelbrot, que ilustra gráficamente el límite entre la convergencia y la divergencia de una función iterativa em un diseño complejo.

2. OBJETIVOS

- Desarrollar un algoritmo iterativo que determine, para cada punto c en el plano complejo, cuántas iteraciones son necesarias para alcanzar la divergencia. Este cálculo es esencial para definir la estructura y el contorno del fractal.
- Se propone analizar cómo la densidad de puntos en la cuadrícula afecta el resultado visual del fractal, evaluando al mismo tiempo el rendimiento del algoritmo según la resolución utilizada. Se intentará establecer cuál es la resolución máxima que el programa puede manejar sin sacrificar la eficiencia y la precisión de los cálculos, asegurando una representación óptima y exacta de las complejas estructuras del fractal.

- Implementar una función de zoom interactivo que permita acercarse y alejarse de la imagen del fractal, de manera que cada acción de zoom redibuje la figura de forma dinámica y precisa

3. METEDOLOGÍA

3.1. Descripción de la solución.

El algoritmo creado permite generar y visualizar el Conjunto de Mandelbrot de forma eficiente, utilizando técnicas de optimización que mejoran el rendimiento en los cálculos iterativos. Para ello, se ha empleado Python junto con bibliotecas especializadas como NumPy, Matplotlib y Numba. Estas herramientas facilitan el manejo de matrices, los cálculos vectorizados, la representación gráfica del fractal y la optimización de la velocidad.

3.2. Validación de Parámetros

Antes de realizar el cálculo del fractal, es crucial establecer y validar los parámetros que

ESCUELA POLITÉCNICA NACIONAL

influyen en la precisión y calidad de la visualización. Entre estos parámetros se incluyen:

- Resolución de la imagen: Define la densidad de puntos evaluados en la cuadrícula y afecta el nivel de detalle del fractal.
- Precisión en coordenadas: Todas las operaciones geométricas utilizan *np.float64*, lo cual es esencial para mantener las estructuras fractales en profundos acercamientos.
- Número máximo de iteraciones: Establece el criterio para determinar la convergencia o divergencia de cada punto, lo que influye en el contraste y la riqueza de detalles en la imagen generada.
- Umbral de escape: Se define generalmente como $|Z| > 2$, criterio que indica que un punto ha divergido.

3.3. Desarrollo de Matemático

Definición del Conjunto de Mandebrot

El conjunto de Mandelbrot es un fractal definido en el plano complejo y se construye a partir de la iteración de la función cuadrática al ser una estructura geométrica auto semejante, usa un fundamento matemático en este caso se usa la formula conocida en el contexto de la teoría de fractales, describe un proceso iterativo *Z y C* son números complejos. Aquí Z_0 se inicia en 0 y *C* representa cada punto del plano complejo que se está evaluando. La iteración continúa hasta que el valor absoluto de *Z* supera 2 o se alcanza un número máximo predeterminado de iteraciones.

$$\mathrm{Z_{n+1}} = \mathrm{Z_N^2} + \mathrm{C}$$

El número de iteraciones necesarias para que $|Z|$ exceda 2, o el máximo permitido, se emplea para determinar el color correspondiente en la representación gráfica del fractal.

ESCUELA POLITÉCNICA NACIONAL

Criterio de Escape

El criterio de escape es una condición matemática utilizada para determinar si un número complejo pertenece al conjunto de Mandelbrot. Se basa en la observación de que, si la sucesión iterativa con la formula ya descrita, en este caso se vuelve infinitamente grande tendiendo ***n*** a infinito, entonces el numero complejo C no pertenece al conjunto de Mandelbrot

Este cálculo es utilizado para el criterio de escape en el caso del programa Python se implementó el uso de $|Z|^2 > 4$ para poder evitar cálculos innecesarios, eliminando la necesidad de calcular raíces cuadradas.

Para ver mejor como es la iteración, usando la formula ya vista mostraremos como es en le caso en donde diverge.

Para C = 1

$$Z_1 = 0^2 + 1 = 1.$$

$$Z_2 = 1^2 + 1 = 2.$$

$$Z_1 = 2^2 + 1 = 5.$$

$$Z_1 = 5^2 + 1 = 26 \ldots y\ así\ sucesivamente$$

Como se ve en este caso la secuencia el punto C = 1 se considera divergente.

3.4. Funciones Principales

Función Calcula iteraciones

Calcula cuántas iteraciones de la función de Mandelbrot se necesitan para determinar si un punto pertenece o no al conjunto, evaluando su convergencia.

ESCUELA POLITÉCNICA NACIONAL

```
@jit(nopython=True)
def mandelbrot(complejo, iter_max):
    z = 0j
    for i in range(iter_max):
        z = z * z + complejo
        if (z.real * z.real + z.imag * z.imag) > 4.0:
            return i
    return iter_max
```

Función Genera la matriz del fractal en paralelo.

Genera la matriz que representa el conjunto de Mandelbrot, realizando los cálculos de forma paralela para mejorar la eficiencia.

```
def generar_mandelbrot(x_min, x_max, y_min, y_max, res_x, res_y, iter_max):
    xs = np.linspace(x_min, x_max, res_x)
    ys = np.linspace(y_min, y_max, res_y)
    imagen = np.empty((res_y, res_x), dtype=np.int32)
    for i in prange(res_y):
        y = ys[i]
        for j in range(res_x):
            x = xs[j]
            imagen[i, j] = mandelbrot(complex(x, y), iter_max)
    return imagen
```

3.5. Visualización y Actualización del Fractal

Función que Gestiona animaciones combinando panning + zoom.

Controla la animación del fractal, permitiendo movimientos (panning) y acercamientos (zoom) en el conjunto para explorar diferentes áreas.

```
def animar_zoom_dinamico(limites_objetivo, pasos_totales=120, retardo=0.005):

    global x_min, x_max, y_min, y_max, cancelar_animacion

    # Centro actual y centro objetivo
    centro_actual = ((x_min + x_max) / 2, (y_min + y_max) / 2)
    centro_objetivo = (
        (limites_objetivo[0] + limites_objetivo[1]) / 2,
        (limites_objetivo[2] + limites_objetivo[3]) / 2,
```

ESCUELA POLITÉCNICA NACIONAL

```
    )

    # Distancia entre centros y ancho actual (para definir un umbral)
    dx = centro_objetivo[0] - centro_actual[0]
    dy = centro_objetivo[1] - centro_actual[1]
    distancia = np.hypot(dx, dy)
    ancho_actual = x_max - x_min

    # Si la diferencia de centros es mayor que un cierto porcentaje del ancho
actual,
    # se realiza primero una animación de panning.
    umbral_pan = ancho_actual * 0.2  # 20% del ancho actual
    if distancia > umbral_pan:
        # Se asignan un número de pasos para panning y zoom (se pueden ajustar)
        pasos_pan = int(pasos_totales * 0.5)
        pasos_zoom = pasos_totales - pasos_pan

        # Etapa de panning: se interpola el centro sin cambiar el zoom actual.
        centro_inicio = centro_actual
        for paso in range(pasos_pan):
            if cancelar_animacion:
                cancelar_animacion = False
                return
            t = suavizado_cubico((paso + 1) / pasos_pan)
            nuevo_centro_x = centro_inicio[0] + dx * t
            nuevo_centro_y = centro_inicio[1] + dy * t
            # Se mantiene el tamaño actual de la ventana
            mitad_ancho = ancho_actual / 2
            mitad_alto = (y_max - y_min) / 2
            nuevo_x_min = nuevo_centro_x - mitad_ancho
            nuevo_x_max = nuevo_centro_x + mitad_ancho
            nuevo_y_min = nuevo_centro_y - mitad_alto
            nuevo_y_max = nuevo_centro_y + mitad_alto
            x_min, x_max, y_min, y_max = (
                nuevo_x_min,
                nuevo_x_max,
                nuevo_y_min,
                nuevo_y_max,
            )
            actualizar_fractal(baja_resolucion=True)
            plt.pause(retardo)
```

ESCUELA POLITÉCNICA NACIONAL

```
        # Etapa de zoom: se interpola desde la ventana actual hasta
'limites_objetivo'.
        limites_inicio = (x_min, x_max, y_min, y_max)
        for paso in range(pasos_zoom):
            if cancelar_animacion:
                cancelar_animacion = False
                return
            t = suavizado_cubico((paso + 1) / pasos_zoom)
            nuevo_x_min = limites_inicio[0] * (1 - t) + limites_objetivo[0] * t
            nuevo_x_max = limites_inicio[1] * (1 - t) + limites_objetivo[1] * t
            nuevo_y_min = limites_inicio[2] * (1 - t) + limites_objetivo[2] * t
            nuevo_y_max = limites_inicio[3] * (1 - t) + limites_objetivo[3] * t
            x_min, x_max, y_min, y_max = (
                nuevo_x_min,
                nuevo_x_max,
                nuevo_y_min,
                nuevo_y_max,
            )
            actualizar_fractal(baja_resolucion=True)
            plt.pause(retardo)
        actualizar_fractal(baja_resolucion=False)
    else:
        # Si la diferencia es pequeña, se usa la animación estática.
        animar_secuencia_zoom(limites_objetivo, pasos=pasos_totales,
retardo=retardo)
```

Validación de Límites Numéricos

La validación de límites numéricos es un proceso implementado en este código para garantizar que los valores de los límites de visualización del conjunto de Mandelbrot no sean demasiado pequeños o grandes, evitando errores de visualización o cálculos innecesarios.

```
def corregir_limites_vista():
    global x_min, x_max, y_min, y_max
    rango_minimo = np.float64(1e-14)
    if (x_max - x_min) < rango_minimo:
        cx = (x_min + x_max) / 2
        x_min = cx - rango_minimo / 2
        x_max = cx + rango_minimo / 2
```

ESCUELA POLITÉCNICA NACIONAL

```
    if (y_max - y_min) < rango_minimo:
        cy = (y_min + y_max) / 2
        y_min = cy - rango_minimo / 2
        y_max = cy + rango_minimo / 2
```

3.6. Interacción con el Usuario

El sistema permite explorar el fractal de manera interactiva. Con las flechas, el usuario mueve la vista (panning), y con las teclas Z y X, acerca o aleja el zoom. Los botones predefinidos llevan a regiones clave, como el "Minibrot". Las animaciones se cancelan automáticamente al interactuar, asegurando respuestas rápidas. Esta interfaz es sencilla y eficiente para navegar el fractal.

```
def evento_teclado(event):
    global x_min, x_max, y_min, y_max, cancelar_animacion
    if event.key == "z":
        zoomear(0.75)
    elif event.key == "x":
        zoomear(1 / 0.75)
    elif event.key == "left":
        cancelar_animacion = True
        dx = (x_max - x_min) * 0.05
        x_min -= dx
        x_max -= dx
        actualizar_fractal(baja_resolucion=False)
    elif event.key == "right":
        cancelar_animacion = True
        dx = (x_max - x_min) * 0.05
        x_min += dx
        x_max += dx
        actualizar_fractal(baja_resolucion=False)
    elif event.key == "down":
        cancelar_animacion = True
        dy = (y_max - y_min) * 0.05
        y_min += dy
        y_max += dy
        actualizar_fractal(baja_resolucion=False)
```

ESCUELA POLITÉCNICA NACIONAL

```
    elif event.key == "up":
        cancelar_animacion = True
        dy = (y_max - y_min) * 0.05
        y_min -= dy
        y_max -= dy
        actualizar_fractal(baja_resolucion=False)
```

4. Visualización del Programa

Mendelbrot Original

Visualización el conjunto de Mandelbrot en su forma clásica, mostrando su estructura fractal característica.

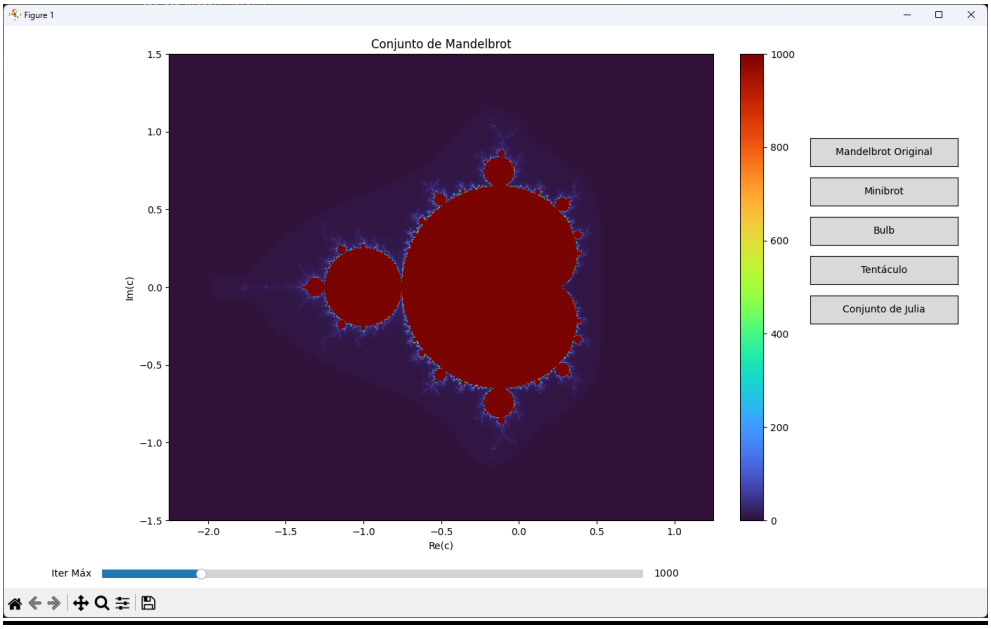


Figura 1. Criterio de escape Mendelbrot Original

Minibrot

Muestra una región más pequeña del conjunto de Mandelbrot, destacando detalles minúsculos y complejos.

ESCUELA
POLITÉCNICA
NACIONAL

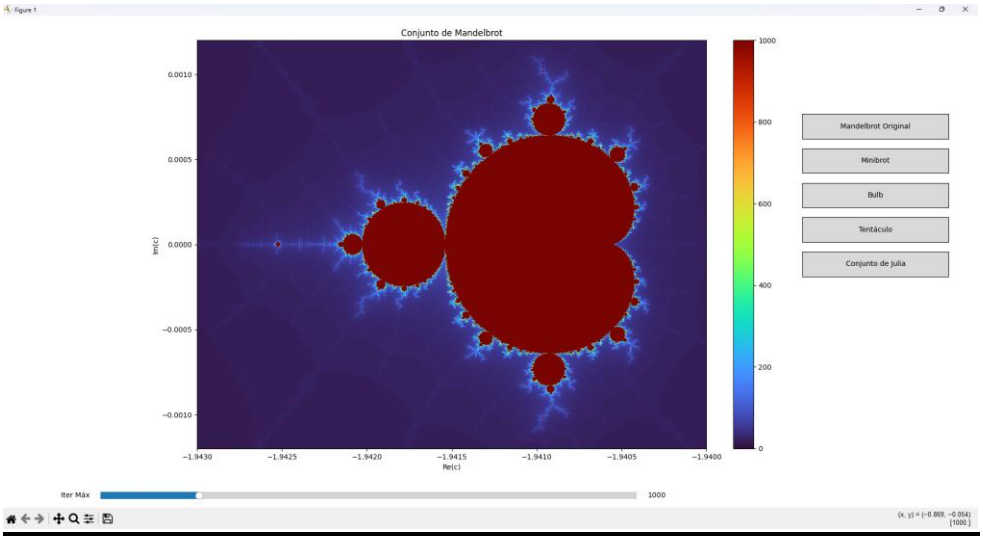


Figura 2. Visualización Minibrot.

Tentáculo

Visualiza una zona específica del conjunto de Mandelbrot que tiene una forma característica similar a un tentáculo, revelando una estructura compleja.

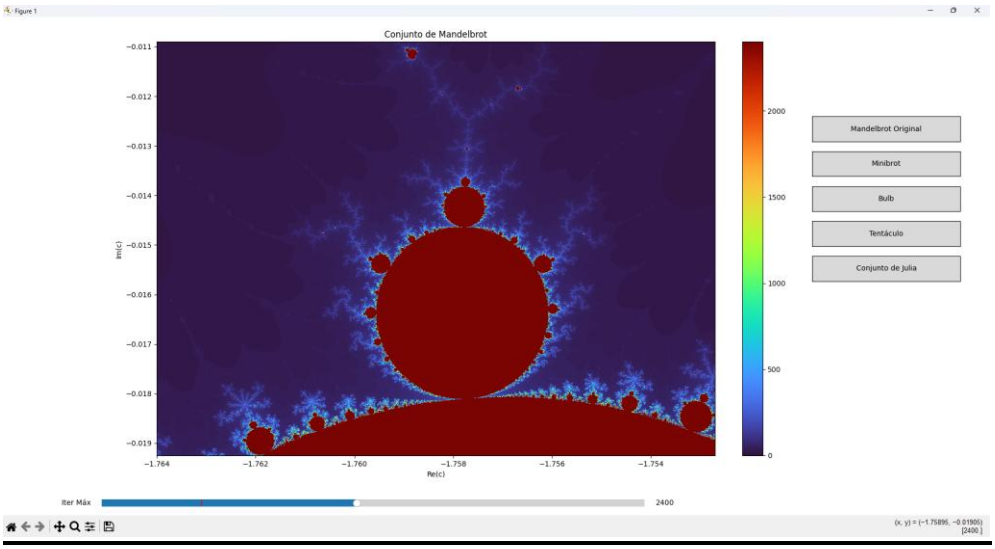


Figura 3. Visualización Tentáculo.

ESCUELA
POLITÉCNICA
NACIONAL

Conjunto de Julia

Visualiza el conjunto de Julia, que es similar al conjunto de Mandelbrot pero con un parámetro fijo, mostrando otro tipo de fractal basado en el valor elegido.

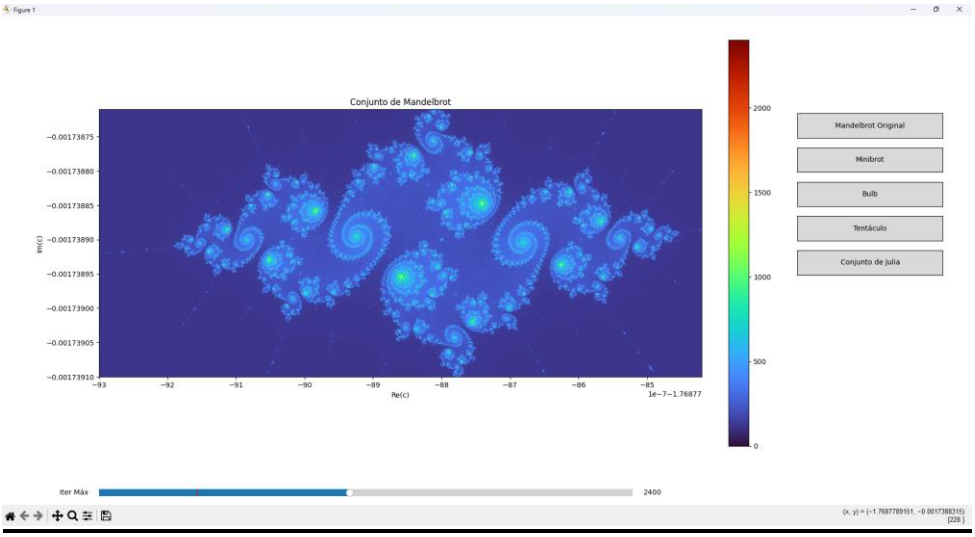
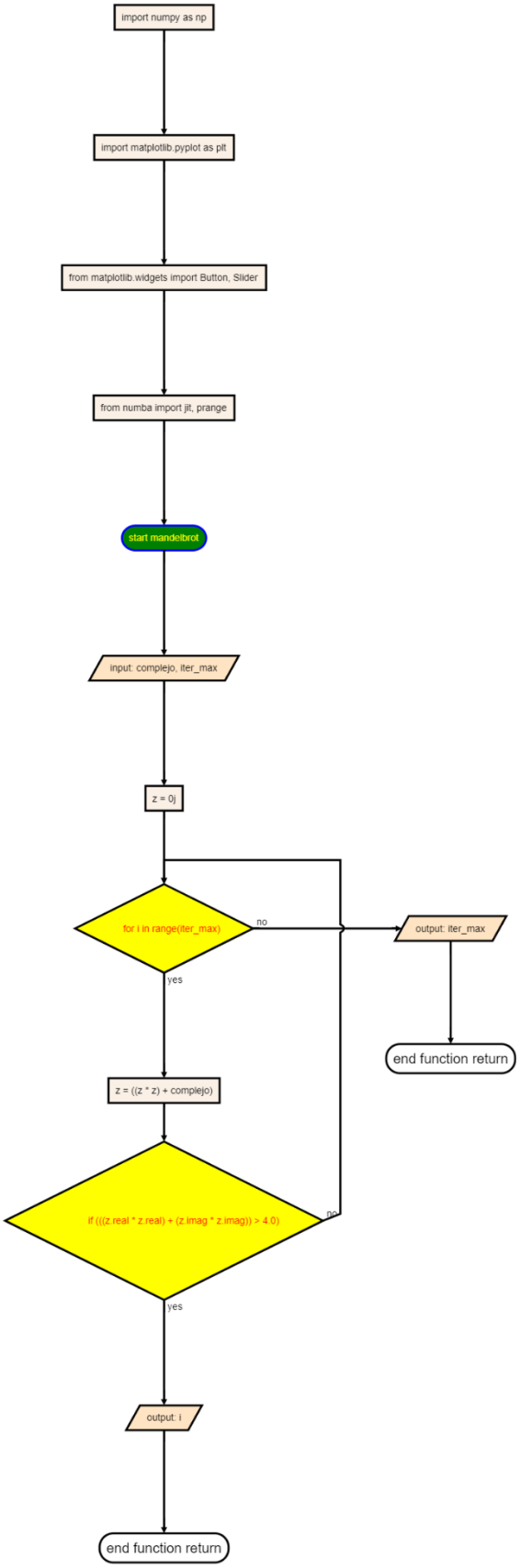


Figura 4. Visualización Conjunto de Julia

ESCUELA
POLITÉCNICA
NACIONAL

5. Diagrama de flujo.



ESCUELA POLITÉCNICA NACIONAL

6. CONCLUSIONES:

- El conjunto de Mandelbrot no solo es una estructura visualmente atractiva, sino que también tiene aplicaciones fundamentales en la matemática, la teoría del caos y el modelado de fenómenos naturales. La iteración de funciones complejas y su análisis numérico permiten comprender mejores sistemas dinámicos no lineales, destacando la importancia de los fractales en el estudio de patrones y estructuras emergentes. Este trabajo sienta las bases para futuras exploraciones en computación gráfica y optimización numérica, con potenciales aplicaciones en simulaciones científicas y modelado matemático avanzado.

- La calidad del fractal generado está directamente influenciada por la densidad de puntos evaluados en la cuadrícula, lo que resalta la importancia de equilibrar resolución y rendimiento computacional. Se implementó una función de zoom interactivo que permite explorar regiones específicas del fractal sin perder precisión, revelando detalles como el "Minibrot" y estructuras más profundas. Sin embargo, a medida que aumenta la resolución, se incrementa el costo computacional, lo que sugiere que futuras mejoras podrían incluir optimización mediante procesamiento en GPU.

- La implementación del conjunto de Mandelbrot permitió una visualización clara y precisa de su estructura fractal, demostrando su autosimilitud a diferentes escalas. La utilización de técnicas de optimización como la vectorización con NumPy y la compilación con Numba permitió mejorar la eficiencia computacional sin comprometer la calidad de la imagen generada. Además, el uso de un criterio de escape optimizado $Z^2 > 4$ evitó cálculos innecesarios, reduciendo el tiempo de procesamiento y asegurando la correcta clasificación de puntos en el plano complejo.