Universidad Alfonso X El Sabio

Grado en Ingeniería Matemática

Variable Compleja y Análisis de Fourier

Teoría de fractales

Integrantes:

Cantos Burgos, Lucia
García González, Martina
González García, María
Sánchez Escribano, José Antonio
Serrano Catalina, Alejandro

1 de enero de 2025

Resumen

Este trabajo presenta un análisis exhaustivo sobre la teoría de fractales, abarcando desde sus fundamentos matemáticos hasta su implementación computacional. Se estudian las propiedades esenciales de los fractales, como la autosimilitud y la dimensión fractal, así como ejemplos clásicos como el conjunto de Mandelbrot, el método de Julia y el triángulo de Sierpinski. A través de algoritmos iterativos, se demuestra cómo estas estructuras complejas emergen de reglas simples, destacando su aplicación en fenómenos naturales y en el modelado de sistemas dinámicos.

Adicionalmente, se desarrolla una aplicación interactiva, accesible mediante un ejecutable disponible en el repositorio de GitHub, que permite la generación, visualización y personalización de fractales. La herramienta facilita la exploración de parámetros como iteraciones y constantes, ofreciendo resultados visuales que ilustran la riqueza matemática de estas estructuras. La funcionalidad de exportación como imágenes asegura su utilidad en contextos académicos y profesionales, consolidando el vínculo entre matemáticas y tecnología.

Índice

1. Introducción. Ejemplos clásicos.

Los fractales representan una intersección entre las matemáticas, la naturaleza y el arte, revelando estructuras que desafían nuestras concepciones tradicionales de la geometría. Estas formas geométricas, caracterizadas por su infinita complejidad y autosimilitud, no solo capturan patrones visuales extraordinarios, sino que también encuentran aplicaciones fundamentales en la ciencia, tecnología y diseño. Su capacidad para combinar simplicidad y sofisticación los convierte en un fenómeno digno de estudio desde múltiples perspectivas.

Posteriormente, se introducirán ejemplos clásicos como el **Conjunto de Mandelbrot**, el **Método de Julia** y el **Método de Newton**. A través de su análisis, se buscará examinar su manifestación en fenómenos naturales, así como su impacto y trascendencia en el ámbito matemático y científico.

1.1. Conjunto de Mandelbrot.

Definición. El **Conjunto de Mandelbrot** se genera a través de la iteración de la función matemática,

$$f(z) = z^2 + c \quad \text{siendo } z, c \in \mathbb{C}$$

Un punto c pertenece al conjunto si, al iniciar la iteración desde $z = 0$, la secuencia generada permanece acotada y no diverge hacia infinito.

Este fractal es conocido por su complejidad visual y su capacidad para generar patrones detallados a partir de una fórmula matemática simple, como se aprecia en la Figura 1.

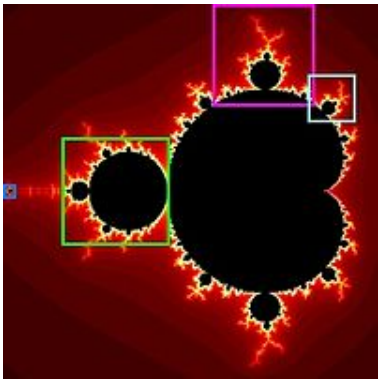

Figura 1: Conjunto de Mandelbrot.

Importancia y Relevancia. El Conjunto de Mandelbrot es un símbolo de la geometría fractal por su complejidad generada a partir de una regla matemática simple. Presenta autosimilaridad aproximada, lo que significa que, al ampliar la figura, se observan estructuras similares al conjunto completo, aunque con variaciones. Además, ha inspirado estudios en matemáticas, física y ciencias computacionales, y se ha utilizado como una herramienta visual y educativa para entender los sistemas dinámicos.

Conexiones. Este conjunto está estrechamente vinculado a los **fractales de Julia** , ya que los valores de c que pertenecen al Conjunto de Mandelbrot determinan fractales de Julia conectados, mientras que los valores fuera del conjunto generan fractales desconectados. Actúa como un "mapa"para explorar las familias

de fractales de Julia, estableciendo una conexión directa entre ambos. Asimismo, comparte similitudes con otros métodos iterativos como el **Método de Newton**, aunque con enfoques y objetivos distintos.

1.2. Método de Julia: diferentes fractales iterando potencias de Z.

Definición. El **Método de Julia** genera fractales mediante la misma fórmula del Conjunto de Mandelbrot.

$$f(z) = z^2 + c$$

Sin embargo, se caracteriza por tener una diferencia clave. En este caso, c se fija como un valor constante y se analizan los comportamientos de los puntos z en el plano complejo.

Fue desarrollado por Gaston Julia, quien estudió cómo los puntos z evolucionan bajo iteraciones sucesivas. El resultado son conjuntos de Julia conectados o desconectados, dependiendo de si c pertenece o no al Conjunto de Mandelbrot. Se puede apreciar una visualización en la Figura 2.

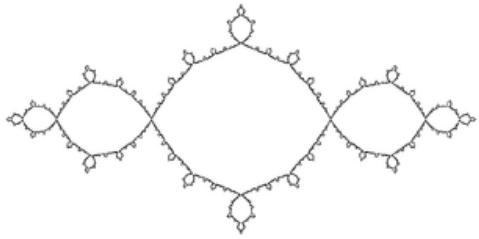

Figura 2: Método de Julia.

Importancia y Relevancia. Los fractales de Julia muestran una autosimilitud precisa, con patrones repetidos que reflejan la estructura global del fractal. Son herramientas matemáticas esenciales para analizar sistemas dinámicos y caóticos, ya que representan la estabilidad bajo iteraciones. Cada valor de c genera un fractal único, ofreciendo una alta variedad de formas visuales.

Conexiones. Existe una relación directa entre los fractales de Julia y el **Conjunto de Mandelbrot**, mencionada anteriormente. Si c pertenece al Conjunto de Mandelbrot, el fractal de Julia resultante será conectado. En caso contrario, si c no pertenece, el fractal será desconectado. Esta conexión permite la comprensión de la estructura subyacente de ambos fractales.

1.3. Método de Newton.

Definición. El **Método de Newton** es un procedimiento iterativo basado en la fórmula

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)} \quad : \text{Fórmula para la obtención de raíces de funciones.}$$

En el contexto fractal, se utiliza para visualizar cómo los puntos del plano complejo convergen hacia distintas raíces.

Cada región del plano se colorea según la velocidad y la dirección de convergencia, creando patrones fractales únicos. Se puede apreciar una visualización en la Figura 3.

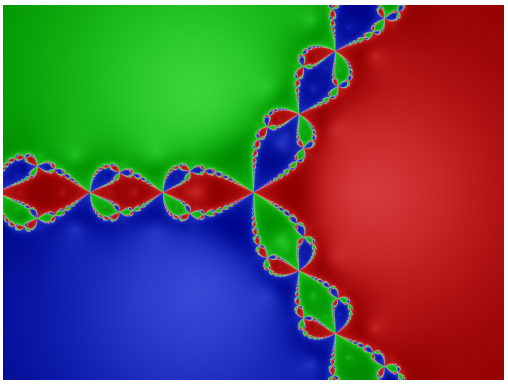

Figura 3: Método de Newton.

Importancia y Relevancia. Este método no solo es una herramienta funcional para aproximar raíces, sino que también genera patrones geométricos visualmente impactantes. Los fractales del Método de Newton ilustran las regiones de estabilidad y caos en las iteraciones, mostrando la complejidad de las dinámicas de convergencia. Es utilizado en matemáticas aplicadas y estudios computacionales para la comprensión del comportamiento de funciones complejas.

Conexiones. Comparado con el **Conjunto de Mandelbrot** y los **fractales de Julia**, el Método de Newton destaca por su enfoque práctico en la resolución de ecuaciones. Aunque su propósito principal no es artístico, sus patrones visuales comparten similitudes con otros fractales iterativos. Además, ofrece una perspectiva única sobre cómo las iteraciones producen estructuras complejas y cómo estas se relacionan con las propiedades de las funciones.

1.4. Otros ejemplos destacados de fractales.

Además de los ejemplos clásicos tratados en detalle, existen otros fractales destacados, que muestran la diversidad y riqueza de esta rama matemática. Estos se mencionan brevemente para ilustrar la variedad de aplicaciones y estructuras fractales.

- **Triángulo de Sierpinski**: Este fractal se forma dividiendo un triángulo equilátero en cuatro triángulos más pequeños y eliminando el triángulo central. Repitiendo este proceso infinitamente, se obtiene una figura con un área que tiende a cero pero con un perímetro infinito.
- **Conjunto de Cantor**: Creado dividiendo un segmento de línea en tres partes y eliminando la central, este fractal tiene dimensión fractal menor que 1. Es útil en el estudio del infinito y la teoría de medidas.
- **Esponja de Menger**: Un fractal tridimensional que se obtiene dividiendo un cubo en 27 partes y eliminando el cubo central. Tiene un volumen que tiende a cero pero una superficie infinita, lo que lo hace interesante desde una perspectiva geométrica.
- **Copos de Koch**: Comienza con un segmento de línea al que se le agrega un triángulo equilátero en su tercio central, repitiendo este proceso infinitamente. Su longitud tiende al infinito, pero el área que encierra es finita.
- **El Helecho de Barnsley**: Este fractal reproduce la apariencia de un helecho natural y es generado mediante ecuaciones iterativas simples. Es un ejemplo de cómo los fractales pueden modelar patrones en la naturaleza.

2. Características de un fractal.

2.1. Autosimilitud.

Los fractales son autosimilares, lo que significa que cada parte de ellos se asemeja al conjunto completo. Esta propiedad está presente tanto en fractales matemáticos como naturales, y es uno de los conceptos clave que los define. La autosimilitud puede clasificarse en:

- **Exacta:** Las partes del fractal son réplicas idénticas del todo. Esto ocurre en fractales generados por algoritmos matemáticos precisos, como el triángulo de Sierpinski, donde cada subestructura es una copia exacta de la figura original.
- **Estadística:** Las partes son similares al todo en promedio, pero no son copias exactas. Este tipo de autosimilitud es característica de los fractales encontrados en la naturaleza, como en la forma de las costas, las montañas o los sistemas de ríos, donde las irregularidades son consistentes pero no idénticas.

Esta propiedad permite que los fractales conserven su estructura sin importar la escala de observación. La **autosimilitud**, ya sea exacta o estadística, está vinculada a los **algoritmos recursivos**, que generan los patrones repetitivos, y a la **dimensión fractal**, clave para cuantificar su complejidad.

Ejemplos relacionados.

- **Conjunto de Mandelbrot:** Exhibe autosimilitud aproximada, donde al hacer zoom aparecen patrones similares al conjunto completo, aunque con variaciones. Su complejidad está directamente relacionada con la dimensión fractal, que mide la estructura infinita del conjunto.
- **Fractales de Julia:** Presentan autosimilitud más precisa, con patrones repetitivos y coherentes en distintas escalas. Esto se logra mediante algoritmos recursivos que reflejan la estructura global del fractal.
- **Fractales naturales:** La autosimilitud estadística es característica en patrones como costas y montañas, donde las irregularidades y la complejidad se miden a través de la dimensión fractal, que captura su estructura a distintas escalas.

2.2. Dimensión fractal y dimensión de Hausdorff-Besicovitch.

La dimensión fractal mide la complejidad geométrica de un fractal, superando las limitaciones de las dimensiones enteras de la geometría clásica. Representa cómo cambia el tamaño del fractal en función de la escala y refleja su estructura intermedia entre líneas, superficies y volúmenes.

2.2.1. Dimensión de caja (Box-counting).

Este método mide cómo el número de cajas necesarias para cubrir un fractal varía a medida que el tamaño de las cajas disminuye.

$$D = \lim_{\epsilon \to 0} \frac{\log N(\epsilon)}{\log\left(\frac{1}{\epsilon}\right)}, \quad \text{dónde } \epsilon \text{ representa el tamaño de las cajas utilizadas para cubrir el fractal.}$$

Este valor, que suele ser un número no entero, refleja la densidad y el nivel de detalle del fractal. Por ejemplo, un resultado cercano a 1 indica que el fractal tiene una estructura lineal, mientras que un valor entre 1 y 2 muestra que el fractal ocupa un espacio intermedio entre una línea y una superficie. Si el valor es

cercano a 2, implica que el fractal tiene una densidad similar a la de una superficie, aunque con complejidad infinita en sus detalles.

El número resultante también describe la forma en que el fractal llena el espacio a medida que se reduce la escala de observación. Una dimensión fractal más alta indica que el fractal es más denso y complejo, con patrones adicionales que emergen en niveles más pequeños de análisis.

Ejemplos relacionados.

- **Triángulo de Sierpinski:** El triángulo de Sierpinski es un ejemplo clásico de fractal generado por algoritmos recursivos. Su dimensión fractal, calculada con el método de caja (Box-counting), es:

$$D = \frac{\log 3}{\log 2} \approx 1{,}585.$$

- **Conjunto de Cantor:** El Conjunto de Cantor es un fractal que se genera dividiendo un segmento de línea en tres partes y eliminando la central de forma iterativa. Su dimensión fractal se calcula de manera similar y es:

$$D = \frac{\log 2}{\log 3} \approx 0{,}630.$$

  Este valor, menor que 1, indica que el Conjunto de Cantor es más denso que un conjunto de puntos discretos, pero no forma una línea continua.

- **Copo de Koch:** El copo de Koch se construye añadiendo triángulos equiláteros al tercio central de cada lado de una figura inicial, repitiendo el proceso de manera infinita. Su dimensión fractal se calcula como:

$$D = \frac{\log 4}{\log 3} \approx 1{,}261.$$

  Esto refleja su longitud infinita en un espacio finito, una característica común en fractales.

2.2.2. Dimensión de Hausdorff-Besicovitch.

Una definición más precisa y general que utiliza la suma de los diámetros elevados a una potencia d para cubrir el fractal.

$$D_H = \text{ínf}\left\{ d : \sum_{i=1}^{\infty} (\text{diámetro de la cubierta})^d < \infty \right\}.$$

Esta dimensión es especialmente útil para fractales matemáticos complejos, proporcionando un marco riguroso para describir su estructura en múltiples escalas.

Ejemplos relacionados.

- **Conjunto de Mandelbrot:** Tiene una dimensión fractal aproximada de 2, lo que indica que ocupa un espacio similar al de una superficie, aunque con complejidad infinita en sus bordes.
- **Conjunto de Cantor:** Su dimensión fractal es menor que 1, reflejando una estructura intermedia entre puntos discretos y una línea continua.

2.3. Definición por algoritmos recursivos.

Los fractales se generan mediante algoritmos recursivos, que consisten en aplicar reglas simples repetidamente para construir estructuras complejas. Matemáticamente, un fractal puede definirse como el límite de un proceso iterativo.

$$F = \bigcap_{n=1}^{\infty} F_n,$$

dónde F_n representa la figura tras n iteraciones. Este enfoque recursivo permite construir estructuras complejas a partir de pasos sencillos. A medida que $n \to \infty$, la figura converge hacia una estructura con características como autosimilitud y complejidad infinita.

Ejemplos relacionados.

- **Conjunto de Cantor:** Con cada paso, el número de segmentos restantes aumenta ($N_n = 2^n$), pero la longitud total de los segmentos disminuye ($L_n = (\frac{2}{3})^n L_0$).A pesar de que la longitud total tiende a 0 en el límite, la estructura conserva una organización infinita de puntos, mostrando cómo las iteraciones simples generan complejidad en un espacio restringido.

- **Triángulo de Sierpinski:** El resultado del proceso iterativo muestra cómo el fractal mantiene su autosimilitud exacta en todas las escalas. Con cada iteración, el número de triángulos aumenta exponencialmente ($N_n = 3^n$), mientras que la longitud de los lados disminuye ($L_n = \left(\frac{1}{2}\right)^n L_0$). Aunque el área total cubierta tiende a 0 en el límite, la estructura infinita persiste, lo que refleja cómo las iteraciones generan una complejidad creciente. El área total cubierta disminuye siguiendo la relación $A_n = \left(\frac{3}{4}\right)^n A_0$, donde A_0 es el área inicial.

  Ejemplo concreto con 4 iteraciones. En cada iteración, el triángulo se divide en cuatro subtriángulos equiláteros más pequeños, eliminando el triángulo central. Este proceso genera una estructura autosimilar que se repite indefinidamente.

  - En la iteración inicial ($n = 0$), el triángulo completo tiene un área inicial A_0 y sus tres lados son de longitud L_0.
  - En la primera iteración ($n = 1$):
    - Dividimos el triángulo inicial en cuatro triángulos más pequeños con lados de longitud $L_1 = \frac{L_0}{2}$.
    - Eliminamos el triángulo central, dejando tres triángulos.
    - El área restante es el 75 % del área inicial, ya que el triángulo eliminado tiene el 25 % del área total.

  Este proceso se repite indefinidamente, generando nuevos triángulos más pequeños y reduciendo progresivamente el área total cubierta.

  Visualización. En la Figura 4, se muestra el Triángulo de Sierpinski tras 4 iteraciones. Cada iteración evidencia cómo se reduce el área y cómo los triángulos restantes replican la estructura inicial.

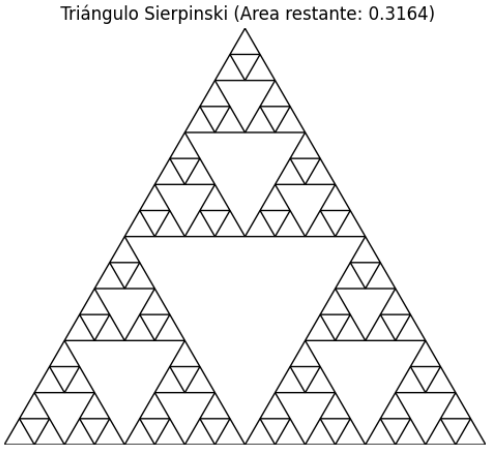

Figura 4: Triángulo de Sierpinski después de 4 iteraciones.

3. Aspectos matemáticos.

3.1. Definición.

Un fractal, desde el punto de vista matemático, es una estructura geométrica caracterizada por su autosimilitud y complejidad infinita. Su construcción se basa en procesos iterativos que aplican reglas simples repetidamente para generar formas que trascienden las dimensiones clásicas (líneas, superficies o volúmenes).

Autosimilitud: Cada parte del fractal se asemeja al conjunto completo, lo que puede clasificarse en dos tipos:

- **Exacta:** Las partes del fractal son réplicas idénticas del conjunto completo.
- **Estadística:** Las partes son similares al todo en promedio, aunque no son copias exactas. Esta propiedad es común en fractales naturales como las costas o los sistemas de ríos.

Dimensión no entera: A diferencia de las figuras geométricas clásicas (líneas: 1D, superficies: 2D, volúmenes: 3D), los fractales poseen una dimensión fraccionaria, lo que refleja su capacidad para ocupar un espacio intermedio. Por ejemplo, un fractal con dimensión 1.5 ocupa más espacio que una línea, pero menos que una superficie. Esta dimensión se calcula mediante herramientas como la dimensión de caja (Box-counting) y la dimensión de Hausdorff-Besicovitch, que describen la densidad y complejidad del fractal en diferentes escalas.
Algoritmos iterativos: Muchos fractales se construyen aplicando repetidamente reglas simples. Desde un punto de vista matemático, un fractal se define como el límite de un proceso iterativo. Si F_n representa la figura tras n iteraciones, el fractal se describe como:

$$F = \bigcap_{n=1}^{\infty} F_n.$$

Estos conceptos fundamentales permiten describir matemáticamente la complejidad y las propiedades únicas de los fractales, que van más allá de las figuras tradicionales de la geometría clásica.

Cada uno de estos conceptos, aunque mencionados aquí de manera resumida, ha sido desarrollado con mayor profundidad en los apartados anteriores, como **Autosimilitud**; **Dimesión de un fractal**; **Algoritmos recursivos**.

3.2. Dimensión Fractal.

La dimensión fractal es una medida matemática que describe la complejidad geométrica de un fractal. A diferencia de las figuras clásicas (líneas, superficies, volúmenes), que tienen dimensiones enteras como 1, 2 o 3, los fractales tienen dimensiones no enteras. Esta propiedad refleja su naturaleza intermedia: un fractal puede ocupar un espacio entre una línea y una superficie.

La dimensión fractal mide cómo cambia el "tamaño"de un fractal al observarlo a diferentes escalas. Existen varios métodos para calcularla, siendo los más comunes la **Dimensión de Caja** y la **Dimensión de Hausdorff**.

Este apartado se aborda de manera más detallada y con explicaciones ampliadas en el punto **Dimensión de un fractal**, donde se profundiza en los métodos de cálculo y en los ejemplos representativos que ilustran este concepto en diferentes contextos.

Interpretación de los resultados. La dimensión fractal de un objeto indica cómo ocupa el espacio a diferentes escalas. Los valores obtenidos se interpretan como sigue:

- **Dimensiones cercanas a 1:** Representan fractales que son esencialmente lineales, como el Conjunto de Cantor ($D \approx 0{,}630$), que ocupa un espacio intermedio entre un conjunto de puntos discretos y una línea continua.
- **Dimensiones entre 1 y 2:** Indican estructuras que ocupan un espacio mayor que una línea pero menor que una superficie, como el Triángulo de Sierpinski ($D \approx 1{,}585$) o el Copo de Koch ($D \approx 1{,}261$).
- **Dimensiones cercanas a 2:** Muestran fractales que se asemejan a superficies, aunque con bordes infinitamente complejos, como el Conjunto de Mandelbrot ($D \approx 2$).
- **Dimensiones superiores a 2:** Reflejan fractales más complejos que comienzan a llenar volúmenes, aunque estos ejemplos son menos comunes en fractales clásicos.

4. Construcción de un fractal.

La construcción de un fractal se basa en la aplicación repetida de reglas simples mediante algoritmos iterativos. Desde un punto de vista matemático, los fractales se generan definiendo un conjunto inicial y aplicando transformaciones que, tras varias iteraciones, producen una estructura que refleja propiedades como la autosimilitud y la dimensión fractal. Estos procesos permiten modelar tanto fractales matemáticos como fenómenos naturales.

Las herramientas clave para construir fractales, todas explicadas en el apartado **Características de un fractal** incluyen:

- Algoritmos recursivos: Métodos que aplican una misma regla en cada iteración.
- Autosimilitud: La estructura resultante mantiene patrones repetitivos a distintas escalas.
- Dimensión fractal: La construcción está vinculada a cómo el fractal ocupa el espacio.

4.1. Implementación práctica de fractales en Python.

La construcción de un fractal puede abordarse desde un enfoque computacional que combina conceptos matemáticos con herramientas de programación. En el proyecto, se implementa un sistema que permite generar fractales como el conjunto de Mandelbrot y los conjuntos de Julia, utilizando Python y bibliotecas como Tkinter para la interfaz gráfica, y NumPy para cálculos numéricos.

4.1.1. Construcción de fractales mediante iteraciones.

La generación de fractales se basa en iteraciones matemáticas sobre el plano complejo. Cada tipo de fractal tiene un método específico:

Conjunto de Mandelbrot. El conjunto de Mandelbrot se define mediante la iteración de la función matemática $f(z) = z^2 + c$, donde z y c son números complejos. El criterio para determinar si un punto pertenece al conjunto es si la secuencia resultante permanece acotada tras un número finito de iteraciones. En el código, este proceso se implementa evaluando cada punto del plano complejo y coloreándolo según la velocidad de divergencia.

```
def mandelbrot(c, max_iter):
    z = 0
    for n in range(max_iter):
        if abs(z) > 2:
            return n
        z = z**2 + c
    return max_iter
```

Conjuntos de Julia. Los conjuntos de Julia se generan de manera similar, pero aquí el valor c se fija y el análisis se realiza sobre puntos z en el plano complejo. Dependiendo del valor de c, el fractal resultante puede ser conectado o desconectado.

```
def julia(z, c, max_iter):
    for n in range(max_iter):
        if abs(z) > 2:
            return n
        z = z**2 + c
    return max_iter
```

Triángulo de Sierpinski. El Triángulo de Sierpinski comienza con un triángulo equilátero inicial, que se subdivide en cuatro triángulos más pequeños. En cada paso:

1. **División inicial:** Se calculan los puntos medios de los lados del triángulo.
2. **Subdivisión:** Se crean tres nuevos triángulos utilizando los puntos iniciales y los puntos medios, y se elimina el triángulo central.
3. **Repetición:** El proceso se aplica recursivamente a cada uno de los tres triángulos restantes hasta alcanzar el nivel deseado de profundidad.

La profundidad del fractal determina cuántas veces se aplica esta operación recursiva. A mayor profundidad, mayor detalle y complejidad en la figura generada.

```
def sierpinski_triangle(p1, p2, p3, depth):
    if depth == 0:
        plt.fill([p1[0], p2[0], p3[0]], [p1[1], p2[1], p3[1]], "black")
    else:
        mid1 = (p1 + p2) / 2
        mid2 = (p2 + p3) / 2
        mid3 = (p3 + p1) / 2
        sierpinski_triangle(p1, mid1, mid3, depth - 1)
        sierpinski_triangle(mid1, p2, mid2, depth - 1)
        sierpinski_triangle(mid3, mid2, p3, depth - 1)
```

4.1.2. Visualización y Parámetros de Construcción.

La representación visual de los fractales se realiza utilizando matrices bidimensionales para mapear los valores del plano complejo a píxeles en una imagen. Esto se logra con la biblioteca NumPy, que permite manejar grandes cantidades de datos de manera eficiente:

Plano Complejo. Los ejes x e y se convierten en un rango de valores complejos, definidos por los límites del zoom y la resolución deseada.

```
x = np.linspace(x_min, x_max, width)
y = np.linspace(y_min, y_max, height)
```

Coloreado. Los valores de iteración calculados para cada punto determinan el color del píxel correspondiente, creando patrones visuales. Este proceso es personalizable, permitiendo elegir esquemas de color que resalten detalles específicos del fractal.

4.1.3. Interfaz Gráfica y Personalización.

La aplicación implementada en el repositorio utiliza Tkinter para proporcionar una interfaz gráfica amigable que permite al usuario explorar y personalizar los fractales generados. El usuario puede interactuar con diferentes opciones para ajustar parámetros como el número de iteraciones, la constante c en los conjuntos de Julia, el rango del plano complejo para acercar o alejar (zoom), y los esquemas de color. Estas opciones permiten modificar la apariencia de los fractales, resaltando patrones específicos y generando representaciones visualmente atractivas que reflejan las propiedades matemáticas subyacentes.

4.2. Ejecución y Resultados.

La ejecución de la aplicación se realiza desde el archivo principal `main.py`, que inicializa la interfaz gráfica y establece los parámetros necesarios para generar fractales. Al ejecutarlo, se despliega una ventana interactiva donde el usuario puede ajustar valores como el número de iteraciones, el rango del plano complejo o las constantes del conjunto de Julia.

Cada cambio en los parámetros se refleja directamente en la visualización del fractal, permitiendo explorar su estructura y complejidad de forma interactiva. Además, la aplicación incluye la funcionalidad de exportar los fractales generados como imágenes, lo que facilita su uso en investigaciones o presentaciones. Los resultados obtenidos muestran la riqueza visual y matemática de los fractales, destacando cómo pequeñas variaciones en los parámetros pueden generar patrones completamente nuevos.

4.2.1. Aplicación interactiva.

Para mayor comodidad, en la carpeta `output` del repositorio se encuentra un archivo ejecutable (`.exe`), que permite usar la aplicación sin necesidad de instalar Python o sus dependencias. Para utilizar este archivo, es necesario descargar el repositorio desde la página oficial de GitHub:

1. Accede al repositorio desde el enlace https://github.com/mgonzalz/vc_fractales/tree/main.
2. En la página principal, haz clic en el botón verde `Code`.
3. Selecciona la opción `Download ZIP` para descargar todo el proyecto como un archivo comprimido.
4. Descomprime el archivo descargado en tu equipo.
5. Dentro de la carpeta descomprimida, navega hasta el directorio `output` y ejecuta el archivo `.exe` correspondiente.

Este ejecutable es una versión empaquetada de la aplicación, que permite disfrutar de todas las funcionalidades sin necesidad de realizar configuraciones adicionales. La interacción con los fractales y su visualización son inmediatas, proporcionando una experiencia intuitiva y eficiente.

5. Bibliografía.

[1] Gobierno de Canarias. (s.f.). *Definición de fractal*. Recuperado de https://www3.gobiernodecanarias.org/medusa/ecoblog/mrodperv/fractales/definicion-de-fractal/

[2] Freire, N. (2023). *Fractales: los patrones que se encuentran en la naturaleza*. National Geographic. Recuperado de https://www.nationalgeographic.com.es/ciencia/fractales-patrones-que-se-encuentran-naturaleza_20807

[3] Fernández Cara, E. (2018). *Fractales: bellos y sin embargo útiles*. Blog del Instituto de Matemáticas de la Universidad de Sevilla. Recuperado de https://institucional.us.es/blogimus/2018/10/fractales-bellos-y-sin-embargo-utiles/

[4] Universitat de Barcelona. (s.f.). *¿Qué es un fractal?* Recuperado de https://www.ub.edu/matefest_infofest2011/triptics/fractal.pdf

6. Anexos.

6.1. Clase Base para Fractales.

```
from abc import ABC, abstractmethod

class Fractal(ABC):
    def __init__(self, width, height, max_iter):
        self.width = width
        self.height = height
        self.max_iter = max_iter
    @abstractmethod
    def generate(self):
        pass
    @abstractmethod
    def plot(self, fractal):
        pass
```

6.2. Implementación del Triángulo de Sierpinski.

```
from ..fractal_base import Fractal
import numpy as np
import matplotlib.pyplot as plt

class Sierpinski(Fractal):
    def __init__(self, width, height, max_iter, depth):
        super().__init__(width, height, max_iter)
        self.depth = depth # Profundidad de la figura.

    def generate(self):
        fig, ax = plt.subplots()
        ax.set_aspect('equal')
        ax.axis('off')
        p1 = np.array([0, 0])
        p2 = np.array([1, 0])
        p3 = np.array([0.5, np.sqrt(3)/2])
        self._sierpinski_triangle(ax, p1, p2, p3, self.depth)
        plt.show()

    def _sierpinski_triangle(self, ax, p1, p2, p3, depth):
        if depth == 0:
            ax.fill([p1[0], p2[0], p3[0]], [p1[1], p2[1], p3[1]], "black")
        else:
            mid1 = (p1 + p2) / 2
            mid2 = (p2 + p3) / 2
            mid3 = (p3 + p1) / 2
            self._sierpinski_triangle(ax, p1, mid1, mid3, depth-1)
            self._sierpinski_triangle(ax, mid1, p2, mid2, depth-1)
            self._sierpinski_triangle(ax, mid3, mid2, p3, depth-1)
```

```
    def plot(self, fractal):
        pass  # Automaticamente generado con MatplotLib.
```

6.3. Implementación del Conjunto de Mandelbrot.

```
from ..fractal_base import Fractal
import numpy as np
import matplotlib.pyplot as plt

class Mandelbrot(Fractal):
    def __init__(self, width, height, max_iter, xmin, xmax, ymin, ymax):
        super().__init__(width, height, max_iter)
        self.xmin = xmin
        self.xmax = xmax
        self.ymin = ymin
        self.ymax = ymax

    def generate(self):
        x = np.linspace(self.xmin, self.xmax, self.width)
        y = np.linspace(self.ymin, self.ymax, self.height)
        fractal = np.zeros((self.height, self.width))

        for i in range(self.height):
            for j in range(self.width):
                c = x[j] + y[i] * 1j
                fractal[i, j] = self._mandelbrot(c)
        return fractal

    def _mandelbrot(self, c):
        z = 0
        n = 0
        while abs(z) <= 2 and n < self.max_iter:
            z = z * z + c
            n += 1
        return n

    def plot(self, fractal):
        plt.figure(figsize=(10, 10))
        plt.imshow(fractal, extent=(self.xmin, self.xmax, self.ymin, self.ymax), cmap="hot")
        plt.colorbar(label="Número de iteraciones")
        plt.title("Conjunto de Mandelbrot")
        plt.xlabel("Re (eje real)")
        plt.ylabel("Im (eje imaginario)")
        plt.show()
```

6.4. Implementación del Conjunto de Julia.

```
from ..fractal_base import Fractal
import numpy as np
import matplotlib.pyplot as plt

class Julia(Fractal):
    def __init__(self, width, height, max_iter, xmin, xmax, ymin, ymax, c):
        super().__init__(width, height, max_iter)
        self.xmin = xmin
        self.xmax = xmax
        self.ymin = ymin
        self.ymax = ymax
        self.c = c # constante de Julia.

    def generate(self):
        x = np.linspace(self.xmin, self.xmax, self.width)
        y = np.linspace(self.ymin, self.ymax, self.height)
        fractal = np.zeros((self.height, self.width))

        for i in range(self.height):
            for j in range(self.width):
                z = x[j] + y[i] * 1j
                fractal[i, j] = self._julia(z)
        return fractal

    def _julia(self, z):
        n = 0
        while abs(z) <= 2 and n < self.max_iter:
            z = z * z + self.c
            n += 1
        return n

    def plot(self, fractal):
        plt.figure(figsize=(10, 10))
        plt.imshow(fractal, extent=(self.xmin, self.xmax, self.ymin, self.ymax), cmap="hot")
        plt.colorbar(label="Número de iteraciones")
        plt.title("Conjunto de Julia")
        plt.xlabel("Re (eje real)")
        plt.ylabel("Im (eje imaginario)")
        plt.show()
```