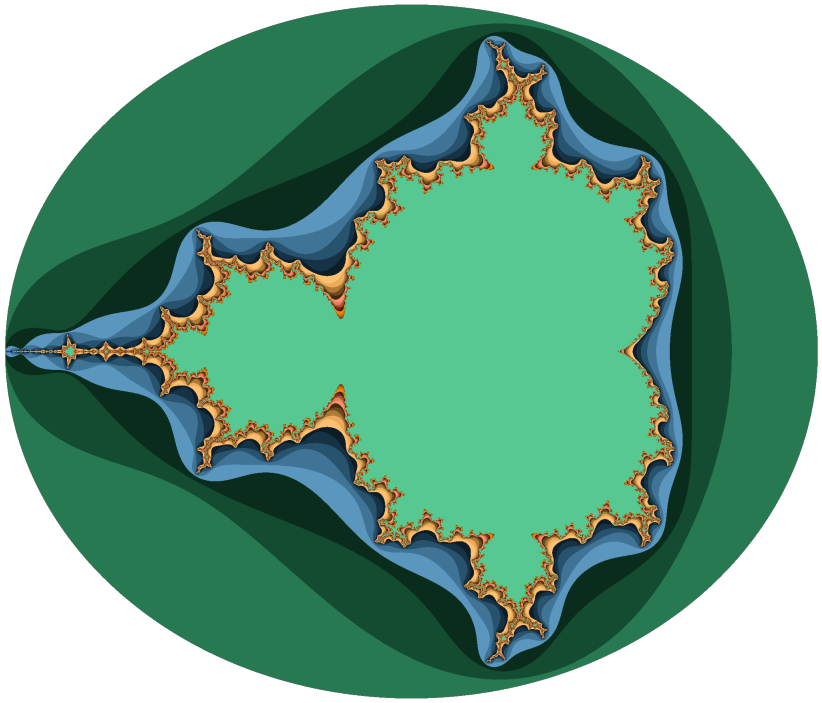
Graficación del Conjunto de Mandelbrot en Python

Andrs.ramos

March 2020



$$Z_{k+1} = Z_k^2 + C$$

Introducción

En el presente documento se aborda el problema de graficación de fractales y se procede a crear el script en python para la graficacion del fractal de Mandelbrot, así como algunas mejoras en los tiempos de ejecución y recomendaciones para hacer del código más funcional usando las librerías matplotlib y pillow.

Conjunto de Mandelbrot

El conjunto mandelbrot es un conjunto de numeros complejos C que satisfacen la condición: $\lim_{n\to\infty} ||P_n(C)||$ no diverge.

Para entender más acerca de esto se define la ecuación recursiva
$\mathrm{Z}_{k+1} = Z_k^2 + C; Z_0 = 0$
para valores de C en el plano complejo.

Dado $C_0 \in C$, sea $P_n(C_0)$ la n-ésima iteración de la ecuación recursiva en el punto C_0, entonces si la norma de P_n se establece en una serie de valores finitos o en general, que permanezca acotada, entonces el punto C_0 pertenece al conjunto de Mandelbrot

TEOREMA de acotamiento.- C pertenece al conjunto de Mandelbrot sí y sólo sí, se satisface que para toda para toda $n \geq 0$, $||P_n(C)|| \leq 2$

Algoritmos para graficación

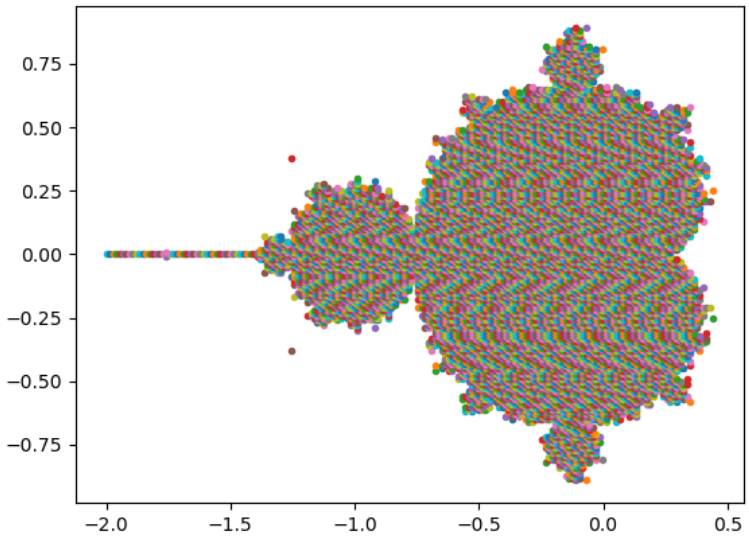
Nos enfocaremos en métodos y algoritmos que sean funcionales para la graficación a computadora del conjunto de mandelbrot.
Un algoritmo simple para graficar el fractal de mandelbrot es el siguiente:
Algoritmo I

1. Tomar un conjunto de puntos en el plano complejo, en un intervalo donde la norma de algunos de los puntos esté en la circunferencia crentrada en el origen de radio 2 (p ej $[-2, \frac{1}{2}]x[-i, i]$)

2. Iterar el punto sobre la ecuación recursiva un numero finito de veces (p ej. 20)
3. A cada paso de la iteración calcular $||Z_n||$ y si es mayor que 2 para algún punto de la iteración dejar de iterar y asignar un 0 a dicho punto; en otro caso asignar a ese punto un 1.
4. Graficar dicho punto si su valor es 1
5. Tomar un nuevo punto en el intervalo (distinto al anterior) y aplicar los pasos 1 al 5 hasta que ya no haya más puntos

Para ejemplificar ésto se muestra la figura 1, que sigue del algorimo anterior

Figure 1: Algoritmo I: 120x120 puntos



Código en python:

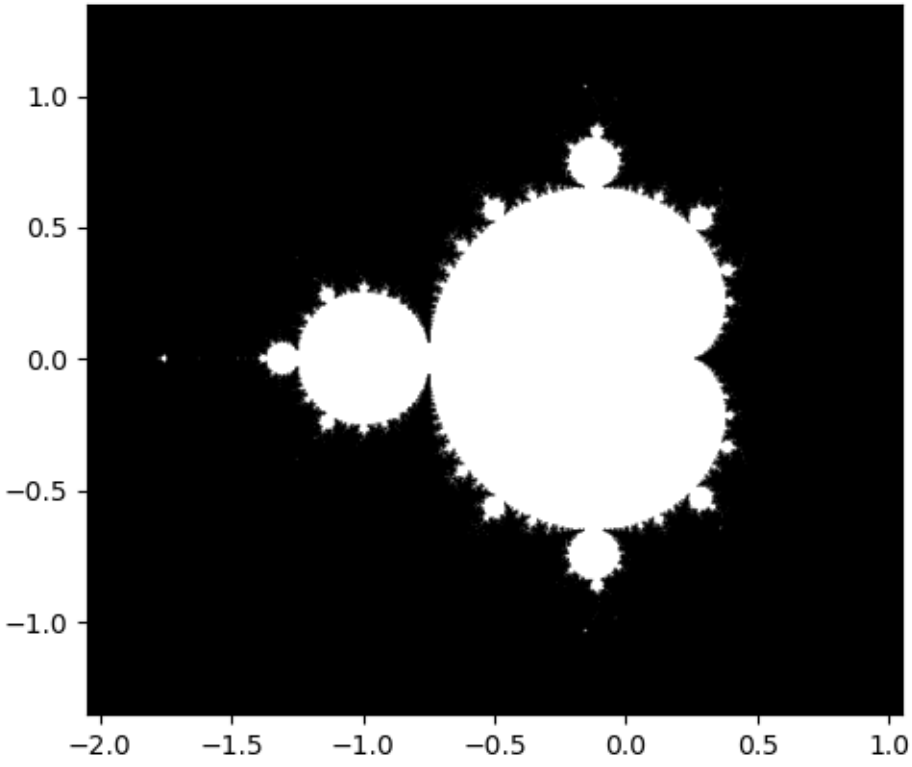
```
import matplotlib.pyplot as plt
def mandelbrot(c):
    z=c
    for i in range(20):
```

```
        z = z*z + c
        if abs(z)>2:
            return 0
    return 1
"---limites de la graficacion ---"
xlim=[-2,.7]
ylim=[-1,1]
"---Tamano de la imagen (pixeles)"
xdensity=100
ydensity=100


deltax=(xlim[1]-xlim[0])/xdensity
deltay=(ylim[1]-ylim[0])/ydensity

for i in range (ydensity):
    y0=ylim[0]+deltay*(i)
    for j in range(xdensity):
        x0=xlim[0]+deltax*(j)
        if mandelbrot(x0+y0*1j)==1:
            plt.plot(x0,y0,'.')
plt.show()
```

Realmente éste no parece una grafica competente del fractal, sino apenas una visualizacion punto por punto, no podemos incrementar demasiado la resolucion, por que el tiempo de ejecucion se eleva demasiado, ya que graficar punto por punto es una tarea que requiere bastantes recursos.
Ahora otra forma más optimizada de usar este algoritmo es con imshow de la libreria matplotlib, que requiere una matriz donde las cordenadas de los pixeles coinciden con los elementos de la matriz, es decir pasarle a las computadora los la posicion del pixel con su respectivo valor 1 o 0 y no graficar punto por punto, sino mandar los pixeles e imshow asigna un color negro al 0 y un color blanco al 1

En la figura 2 podemos ver a más detalle el fractal del conjunto de mandelbrot

Figure 2: Algoritmo I: 800x800 puntos



Código en python

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import colors
#import time as t
from numba import jit

@jit
def mandelbrot(c):
    z=c
    for i in range(50):
```

```
        z = z*z + c
        if abs(z)>2:
            return 0
    return 1

#inicio=t.time()
"------limites de graficacion y densidad de pixeles ------"
x=np.linspace(-2.05,1.05,2000)
y=np.linspace(-1.35,1.35,2000)

"----graficacion ----"
sx=np.size(x)
sy=np.size(y)
a=np.empty((sy,sx))
for i in range(sy):
    for j in range(sx):
        a[i,j]=mandelbrot(x[j]+y[i]*1j)
norm = colors.PowerNorm(0.3)
p=plt.imshow(a,cmap='Greys',norm=norm,
             extent=[-2.05,1.05,-1.35,1.35])
plt.show()
```

Bajo la misma idea, es momento de graficar sobre una paletta de color el fractal del conjunto de Mandelbrot, pero primero una breve explicación de como introducir una paleta de color a un gráfico, primero debemos tener una paleta de color no es mas que tener conjunto de colores por código, ya sea RGB, RGBA o CMYK o quizá el codigo Hex del RGB, aqui unicamente usaremos RGB y RGBA. RGB es un código para los colores que viene dado en ternas, así el primer elemento es la cantidad de rojo, el segundo la cantidad de verde, y el tercero una cantidad de azul en una escala de 0 a 255, una imagen 24-bits png es una imagen RGB donde cada pixel puede tener una gamma de 2^{24} colores, ahora el problema es saber como escoger nuestra paleta de color y como aplicarla,para eso tenemos el siguiente algoritmo
Algoritmo II

1. Tomar un conjunto de puntos en el plano complejo, en un intervalo donde la norma de algunos de los puntos esté en la circunferencia crentrada en el origen de radio 2 (p ej $[-2, \frac{1}{2}]x[-i, i]$)

2. Iterar el punto sobre la ecuación recursiva un numero finito de veces (p ej. 20)

3. A cada paso de la iteración calcular $||Z_n||$ y si es mayor que 2 para algún punto de la iteración dejar de iterar y asignar un 0 a dicho punto; en otro caso asignar a ese punto un 1.

4. Graficar dicho punto si su valor es 1

5. Tomar un nuevo punto en el intervalo (distinto al anterior) y aplicar los pasos 1 al 5 hasta que ya no haya más puntos