Deber Seminario

Fausto Fabián Crespo Fernández

Sistema Iterado de funciones del triángulo de Sierpinsky

El triángulo de Sierpinsky es un fractal que se puede construir a partir de cualquier triángulo equilátero (https://es.wikipedia.org/wiki/Tri%C3%A1ngulo_de_Sierpinski).

El proceso para construirlo se explica en el artículo http://alerce.pntic.mec.es/rgob0004/sierpinski/fichasierpinski.pdf : se parte de un triángulo equilátero con un lado horizontal y se hallan los 3 puntos medios de los lados del triángulo, se unen estos puntos medios y se extrae ese triángulo(que es también un triángulo equilátero pero invertido) y se repite el proceso con los 3 triángulos equiláteros menores que quedan(determinados por dichos puntos medios y los vértices del triángulo original).

El término fractal fue acuñado por Mandelbrot para describir el conjunto que lleva su nombre y la palabra fractus vienen del latín (Capitulo 2 http://www.konradlorenz.edu.co/images/stories/suma_digital_matematicas/SFI%20y%20los%20Fractales.pdf).

En el capítulo 3 del artículo anterior se definen los conjunto auto semejantes mediantes 3 ejemplos: el primero es el conjunto ternario de Cantor, que se genera mediante dos contracciones $\Psi_1(x) = \frac{x}{3}$ y $\Psi_2(x) = \frac{x}{3} + \frac{2}{3}$. La primera transforma el intervalo $I = [0,1]$ a $[0, \frac{1}{3}]$

O sea $\Psi_1(I) = [0, \frac{1}{3}]$ y la segunda a $[\frac{2}{3}, 1]$ o $\Psi_2(I) = [\frac{2}{3}, 1]$ .El conjunto de Cantor E no es más que el conjunto de números reales en $= [0,1]$ y además $\Psi_1(E)\ y\ \Psi_1(E)$ no se intersecan y cubren todo E por tanto el conjunto de Cantor es auto semejante.

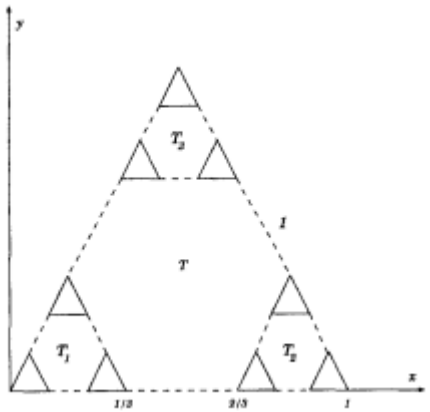
En la página 35 del artículo citado se define semejanza contractiva como una aplicación $f: X \to X\ \ con\ (X, d)$ un espacio métrico si para $x, y \in X,\ d\big(f(x), f(y)\big) \leq k\ d(x, y)$ con $0 < k < 1$ llamada constante de contracción o módulo de la contracción.

En el artículo se menciona que las aplicaciones contractivas son continuas, la composición de aplicaciones contractivas es contractiva con módulo el producto de los módulos y por tanto si f es contractiva, entonces f^n es contractiva de módulo, el módulo de f a la n.

En el capítulo 4 del artículo:
http://www.konradlorenz.edu.co/images/stories/suma_digital_matematicas/SFI%20y%20los%20Fractales.pdf se menciona los fractales son cualquier subconjunto compacto de $\mathbb{R}^n$

y se menciona que Hutchinson en 1981 estudió las propiedades de los fractales existentes y elaboro una teoría para generar los conjunto de fractales auto semejantes. También en dicho artículo, se dice que en 1985, Barnsley generalizó el método de Hutchinson ya que no solo usó semejanzas contractivas sino en general aplicaciones contractivas para generar los fractales.

En la página 43 a la 45 del artículo citado, se menciona que para el fractal triángulo de Sierpinsky $T \subset \mathbb{R}^2$ podemos poner $T = T_1 \cup T_2 \cup T_3$ donde T_i son las partes del triángulo que caen dentro de los 3 triángulos de lado 1/3.



Los T_i $1 \leq i \leq 3$ son semejantes al conjunto total T y por tanto existen semejanzas contractivas o sistema iterado $f_1, f_2\ y\ f_3$ tales que $f_i(T) = T_i$ $1 \leq i \leq 3$ que hacen que

$f_1(T) \cup f_2(T) \cup f_3(T) = T$

(Página 43 de http://www.konradlorenz.edu.co/images/stories/suma_digital_matematicas/SFI%20y%20los%20Fractales.pdf)

La aplicación f_1 es una homotecia con centro en (0,0) y razón 1/3:

$$f_1(x,y) = (\tfrac{x}{3}, \tfrac{y}{3}) \text{ o } f_1\begin{pmatrix}x\\y\end{pmatrix} = \begin{pmatrix}\frac{1}{3} & 0\\ 0 & \frac{1}{3}\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}$$

La aplicación f_2 es una homotecia con centro en (0,0) y razón 1/3 seguido de una traslación de vector (2/3,0)

$$f_2(x,y) = \left(\tfrac{x}{3}, \tfrac{y}{3}\right) + (\tfrac{2}{3}, 0) \text{ o } f_2\begin{pmatrix}x\\y\end{pmatrix} = \begin{pmatrix}\frac{1}{3} & 0\\ 0 & \frac{1}{3}\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} + \begin{pmatrix}\frac{2}{3}\\0\end{pmatrix}$$

La aplicación f_3 es una homotecia con centro en (0,0) y razón 1/3 seguida de una traslación de vector $(\frac{2}{3}\cos(60), \frac{2}{3}sen(60))$ y

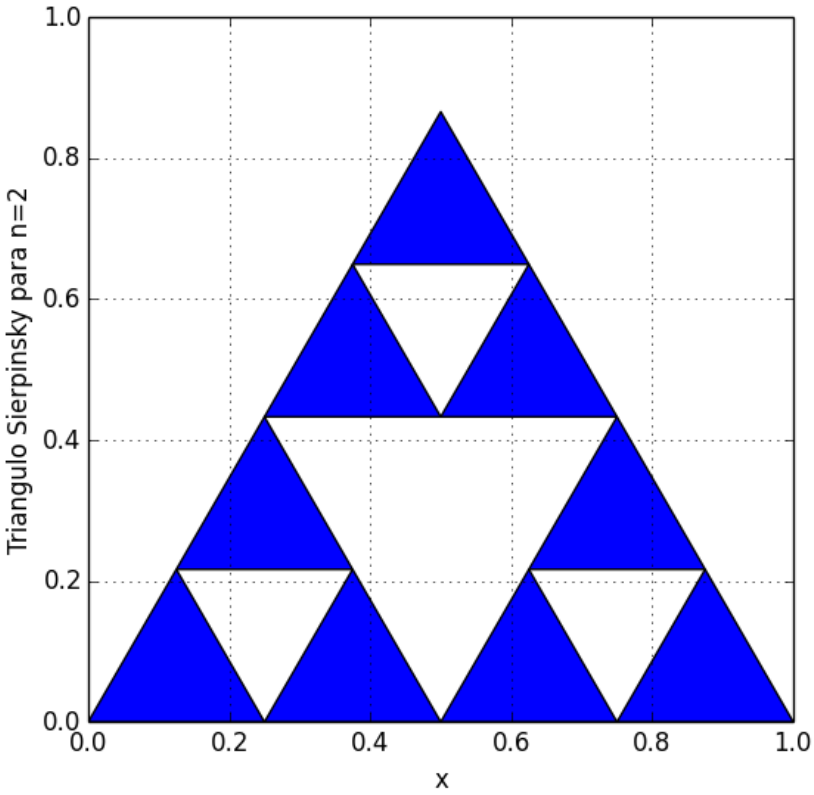
$$f_3(x,y) = \left(\tfrac{x}{3}, \tfrac{y}{3}\right) + (\tfrac{2}{3}\cos(60), \tfrac{2}{3}\text{sen}(60)) \text{ o } f_3\begin{pmatrix}x\\y\end{pmatrix} = \begin{pmatrix}\frac{1}{3} & 0\\ 0 & \frac{1}{3}\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} + \tfrac{2}{3}\begin{pmatrix}\cos(60)\\ \text{sen}(60)\end{pmatrix}$$

(página 44 de

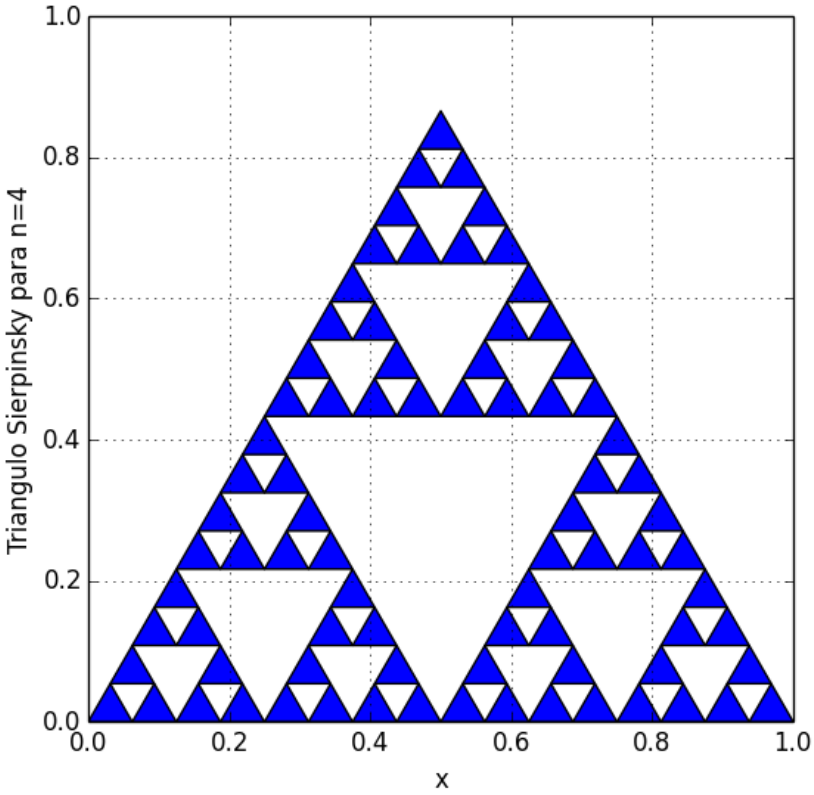
http://www.konradlorenz.edu.co/images/stories/suma_digital_matematicas/SFI%20y%20los%20Fractales.pdf)

Además en este deber se implementó un código con funciones recursivas en Python para generar el fractal triángulo de Sierpinsky.

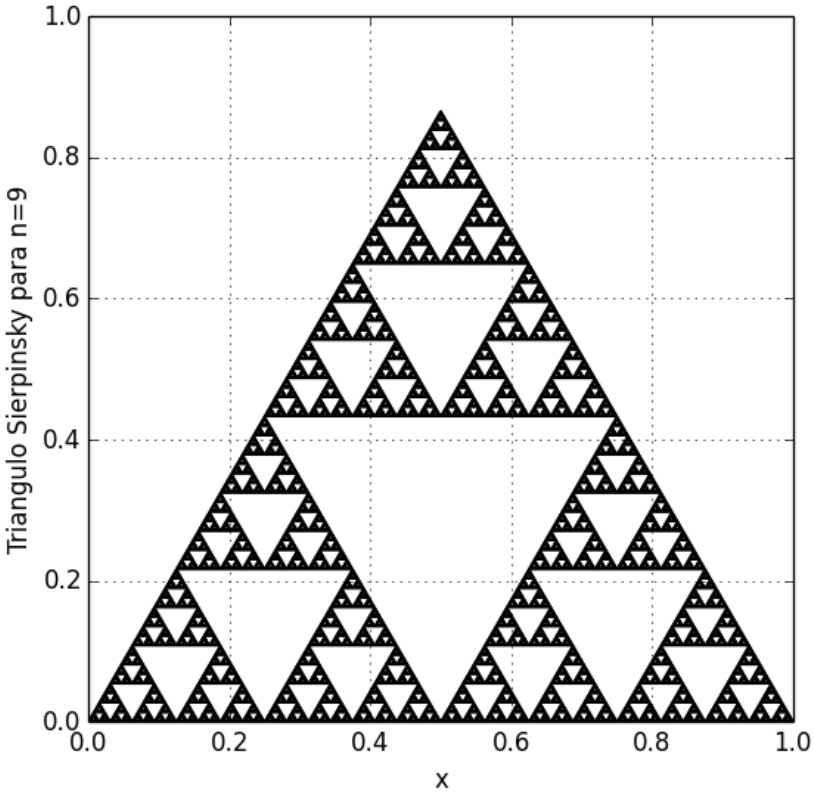
Los resultados fueron para n=2



Para n=4


Triangulo Sierpinsky para n=4
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
x

Para n=9



Código en Python:

```
import numpy as np
import matplotlib.pyplot as plt
import pylab as pl
import math
import matplotlib.pyplot as plt
import matplotlib.patches as patches



def CircunscribedCircleAroundInverteddRegularTriangle(points):
xCentro=points[2][0]
yCentro=points[2][1]+(2.0 / 3.0)*(points[0][1]-points[2][1])
radius =math.sqrt(math.pow((xCentro-points[2][0]),2)+math.pow((yCentro-
points[2][1]),2))
return [[xCentro, yCentro], radius]

def GenerateSierpinskyTriangle(points,n):
if n == 0:
return
points_triangle1 =[points[0]]
```

```
points_triangle1.append([(points[0][0] + points[1][0])/ 2, (points[0][1] + points[1][1])/ 2 ])
points_triangle1.append([(points[0][0] + points[2][0])/ 2, (points[0][1] + points[2][1])/ 2 ])
points_triangle2 =[points_triangle1[1]]
points_triangle2.append(points[1])
points_triangle2.append([(points[1][0] + points[2][0])/ 2, (points[1][1] + points[2][1])/ 2 ])
points_triangle3 =[points_triangle1[2]]
points_triangle3.append(points_triangle2[2])
points_triangle3.append(points[2])
points_center_triangle=[points_triangle1[1], points_triangle3[1], points_triangle1[2]]
[center,
radius]=CircunscribedCircleAroundInverteddRegularTriangle(points_center_triangle)

ax1 = fig1.add_subplot(111, aspect='equal')
ax1.add_patch(
patches.RegularPolygon(
center,
3,
radius,
math.pi,
facecolor ="white"
)
)
GenerateSierpinskyTriangle(points_triangle1, n-1)
GenerateSierpinskyTriangle(points_triangle2, n-1)
GenerateSierpinskyTriangle(points_triangle3, n-1)


def GenerateSierpinskyTriangle2(n):
GenerateSierpinskyTriangle([[0.0,0.0],[0.5, math.sqrt(3) / 2],[1.0,0.0]],n)
radius= math.sqrt(3) / 3

fig1 = plt.figure()
ax1 = fig1.add_subplot(111, aspect='equal')
ax1.add_patch(
patches.RegularPolygon(
(0.5 , math.sqrt(3) / 6),
3,
radius,
0
)
)
n=9
GenerateSierpinskyTriangle2(n)

plt.axis([0, 1, 0, 1])
plt.xlabel('x')
plt.ylabel('Triangulo Sierpinsky para n='+str(n))
```

```
plt.grid(True)
plt.show()
```