Práctica 2

March 10, 2025

1 Práctica 2 : Sistemas de funciones iteradas (SFI)

Autor: Miriam Bernat Jiménez

Documento auto-generado a partir de un Jupyter Notebook

1.1 Ejercicio 1: Escribe un programa que implemente el algoritmo determinista para la obtención del fractal asociado a un sistema de funciones iteradas (SFI), teniendo en cuenta que la entrada del programa debe consistir en tres datos: SFI, número de pasos y semilla.

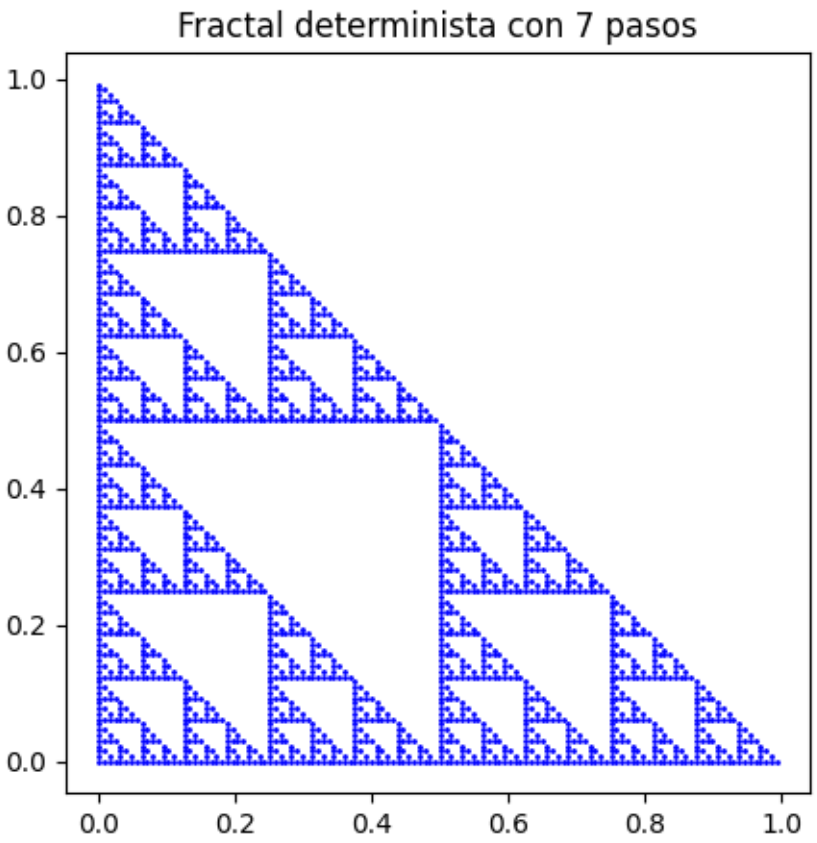
[2]:
```python
import matplotlib.pyplot as plt

# Implementamos la función para el algoritmo determinista
def fractal_determinista(sfi, pasos, semilla):
    """
    sfi: lista de funciones (SFI)
    pasos: número de iteraciones
    semilla: tupla (x0, y0) con el punto inicial
    """
    # Conjunto inicial con la semilla
    puntos = [semilla]

    for _ in range(pasos):
        nuevos_puntos = []
        # A cada punto actual le aplicamos TODAS las funciones del SFI
        for p in puntos:
            for funcion in sfi:
                nuevos_puntos.append(funcion(p))
        # Actualizamos la lista de puntos con los nuevos
        puntos = nuevos_puntos

    return puntos

# Nuestro main para ejecutar
def ejecucion(sfi, pasos, semilla):
    """
     sfi: lista de funciones (SFI)
```

```
    """
    # Ejecutamos el algoritmo determinista
    puntos_resultantes = fractal_determinista(sfi, pasos, semilla)

    # Separamos en listas las coordenadas x e y
    x_vals, y_vals = zip(*puntos_resultantes)

    # Graficamos el conjunto de puntos
    plt.figure(figsize=(5, 5))
    plt.scatter(x_vals, y_vals, s=1, color='blue')
    plt.title(f"Fractal determinista con {pasos} pasos")
    plt.axis('equal')    # Misma escala en ambos ejes
    plt.show()
```

1.2 Ejercicio 2: Pruébalo con los cdigos de los SFI de las transparencias de clase.

1.2.1 1. Sierpinski:

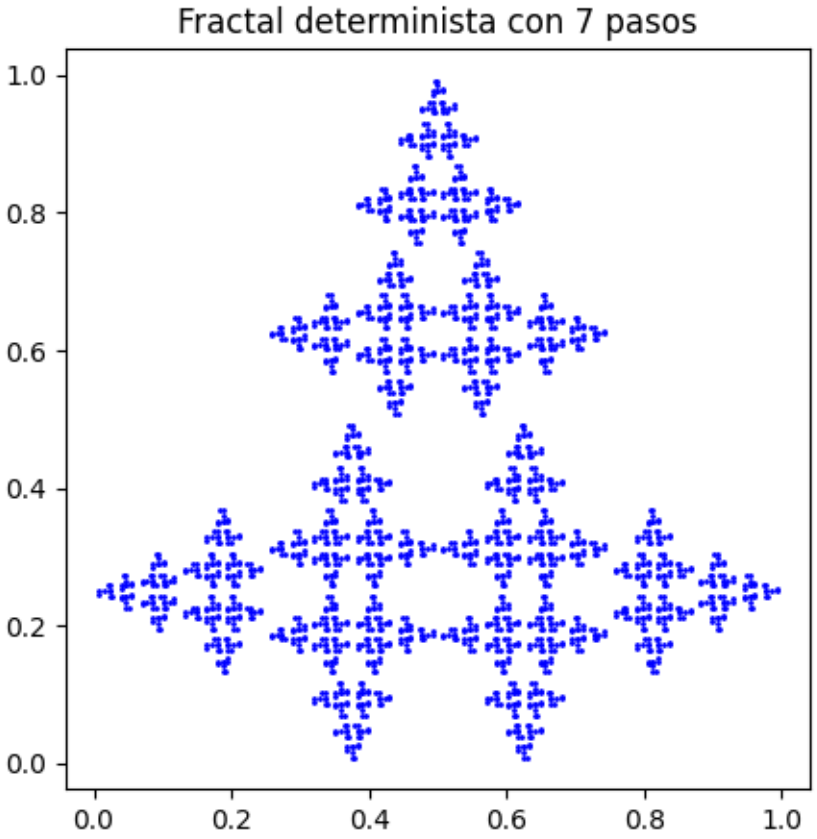
[3]:
```
# Definimos las transformaciones (f1, f2, f3) según la tabla dada
def f1(p):
    x, y = p
    return (0.5 * x, 0.5 * y)

def f2(p):
    x, y = p
    return (0.5 * x + 0.5, 0.5 * y)

def f3(p):
    x, y = p
    return (0.5 * x, 0.5 * y + 0.5)

# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3]

pasos = 7

semilla = (0,0)

ejecucion(sfi, pasos, semilla)
```



1.2.2 2. Fractal:

[4]:
```
# 1) Definimos las transformaciones (f1, f2, f3) según la tabla dada
def f1(p):
    x, y = p
    return (-0.5 * y + 0.5, 0.5 * x)

def f2(p):
    x, y = p
    return (0.5 * y + 0.5, -0.5 * x + 0.5)

def f3(p):
    x, y = p
    return (0.5 * x + 0.25, 0.5 * y + 0.5)

# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3]

pasos = 7
```

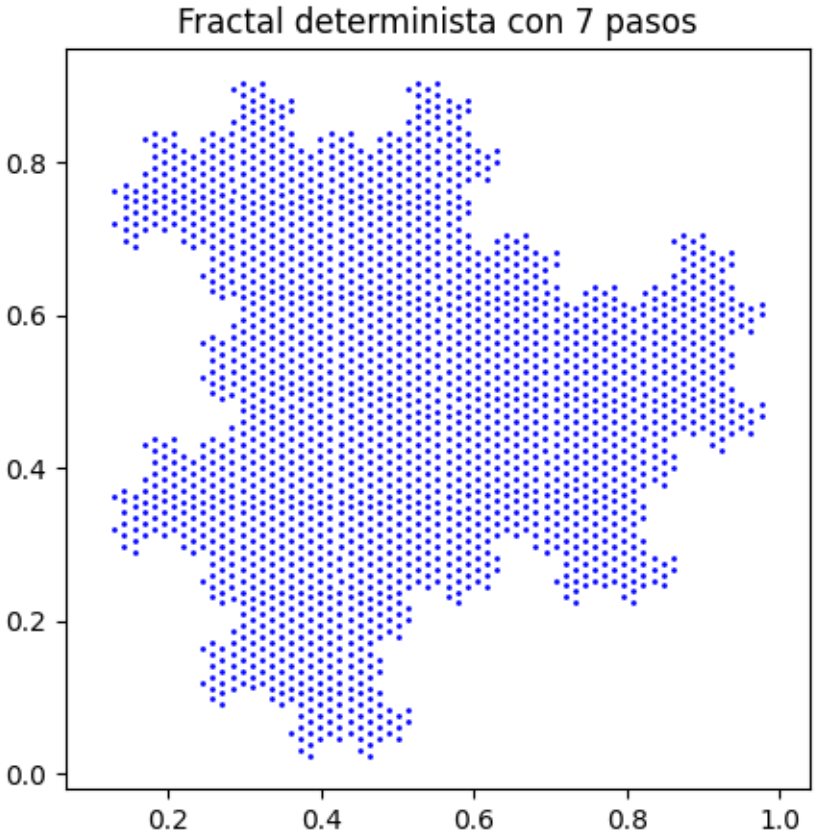
```
semilla = (0,0)

ejecucion(sfi, pasos, semilla)
```



1.2.3 3. Fractal:

[5]:
```
def f1(p):
    x, y = p
    return (0.577 * y + 0.0951, -0.577 * x + 0.5893)

def f2(p):
    x, y = p
    return (0.577 * y + 0.4413, -0.577 * x + 0.7893)

def f3(p):
    x, y = p
    return (0.577 * y + 0.0952, -0.577 * x + 0.9893)

# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3]
```

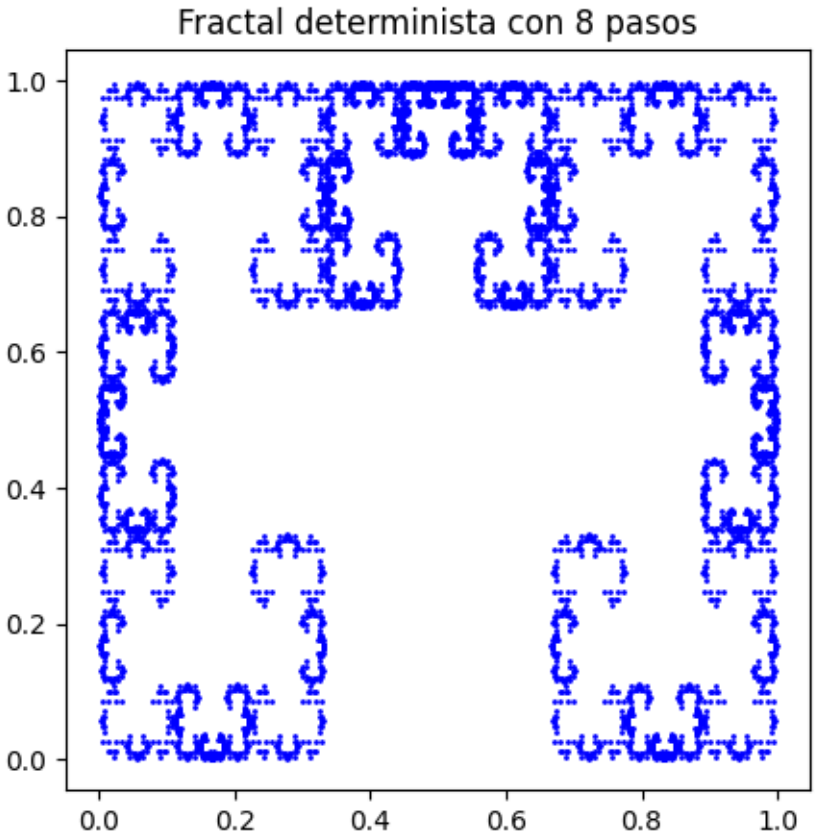
```
pasos = 7

semilla = (0,0)

ejecucion(sfi, pasos, semilla)
```



1.2.4 4. Fractal:

[6]:
```
def f1(p):
    x, y = p
    return (0.333 * x + 0.333, 0.333 * y + 0.666)

def f2(p):
    x, y = p
    return (0.333 * y + 0.666, x)

def f3(p):
    x, y = p
    return (-0.333 * y + 0.333, x)
```

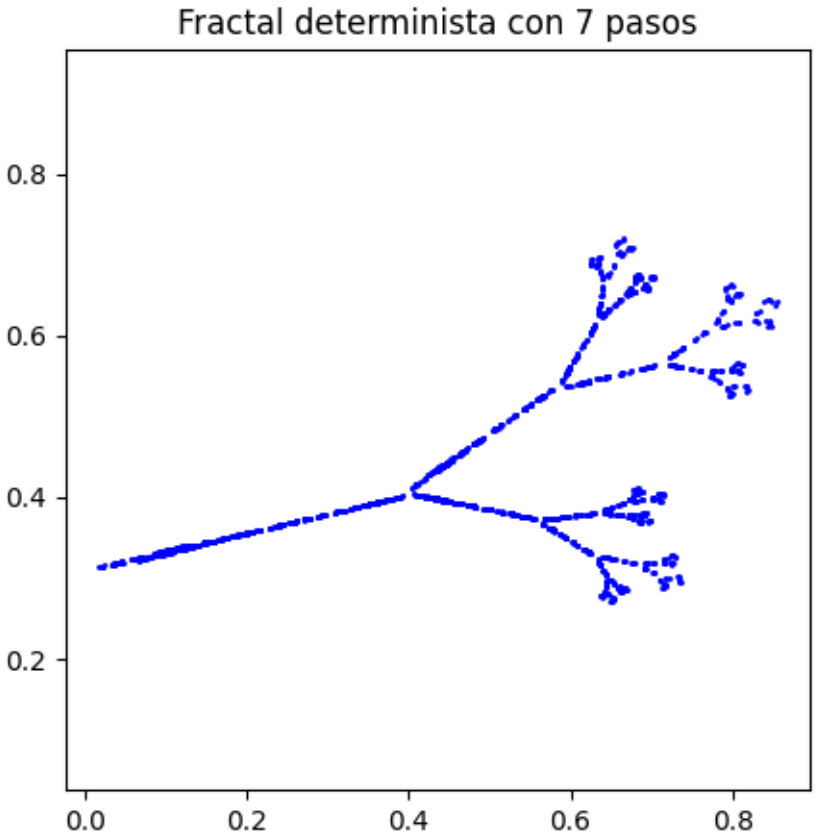
```
# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3]

pasos = 8

semilla = (0,0)

ejecucion(sfi, pasos, semilla)
```



1.2.5 5. Fractal:

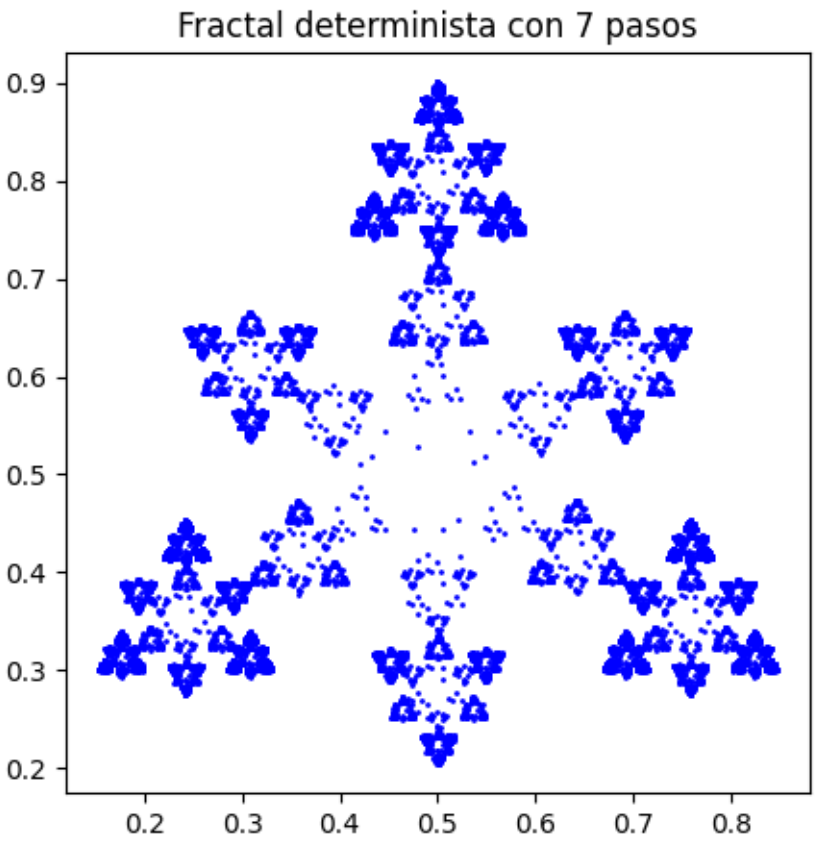
[7]:
```
def f1(p):
    x, y = p
    return (0.387 * x + 0.430 * y + 0.2560, 0.430 * x + (-0.387) * y + 0.5220)

def f2(p):
    x, y = p
```

```
    return (0.441 * x + (-0.091) * y + 0.4219, (-0.009) * x + (-0.322) * y + 0.
↪5059)

def f3(p):
    x, y = p
    return (-0.468 * x + 0.020 * y + 0.4000, (-0.113) * x + 0.015 * y + 0.4000)

# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3]

pasos = 7

semilla = (0,0)

ejecucion(sfi, pasos, semilla)
```



1.2.6 6. Fractal:

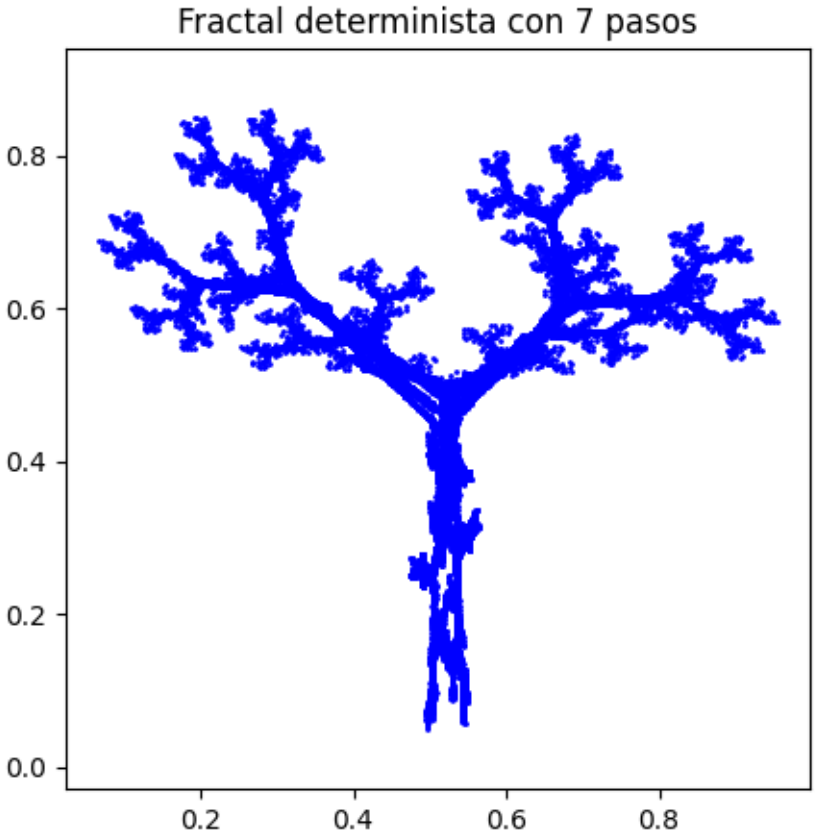
[8]:
```
def f1(p):
    x, y = p
    return (0.255 * x + 0.000 * y + 0.3726, 0.000 * x + 0.255 * y + 0.6714)

def f2(p):
    x, y = p
    return (0.255 * x + 0.000 * y + 0.1146, 0.000 * x + 0.255 * y + 0.2232)

def f3(p):
    x, y = p
    return (0.255 * x + 0.000 * y + 0.6306, 0.000 * x + 0.255 * y + 0.2232)

def f4(p):
    x, y = p
    return (0.370 * x + (-0.642) * y + 0.6356, 0.642 * x + 0.370 * y + (-0.
 ↪0061))

# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3, f4]

pasos = 7

semilla = (0,0)

ejecucion(sfi, pasos, semilla)
```



1.2.7 7. Fractal:

[9]:
```
def f1(p):
    x, y = p
    return (0.195 * x + (-0.488) * y + 0.4431, 0.344 * x + 0.443 * y + 0.2452)

def f2(p):
    x, y = p
    return (0.462 * x + 0.414 * y + 0.2511, (-0.252) * x + 0.361 * y + 0.5692)

def f3(p):
    x, y = p
    return (-0.058 * x + (-0.070) * y + 0.5976, 0.453 * x + (-0.111) * y + 0.
 ↪0969)

def f4(p):
    x, y = p
    return (-0.035 * x + 0.070 * y + 0.4884, (-0.469) * x + (-0.022) * y + 0.
 ↪5069)
```

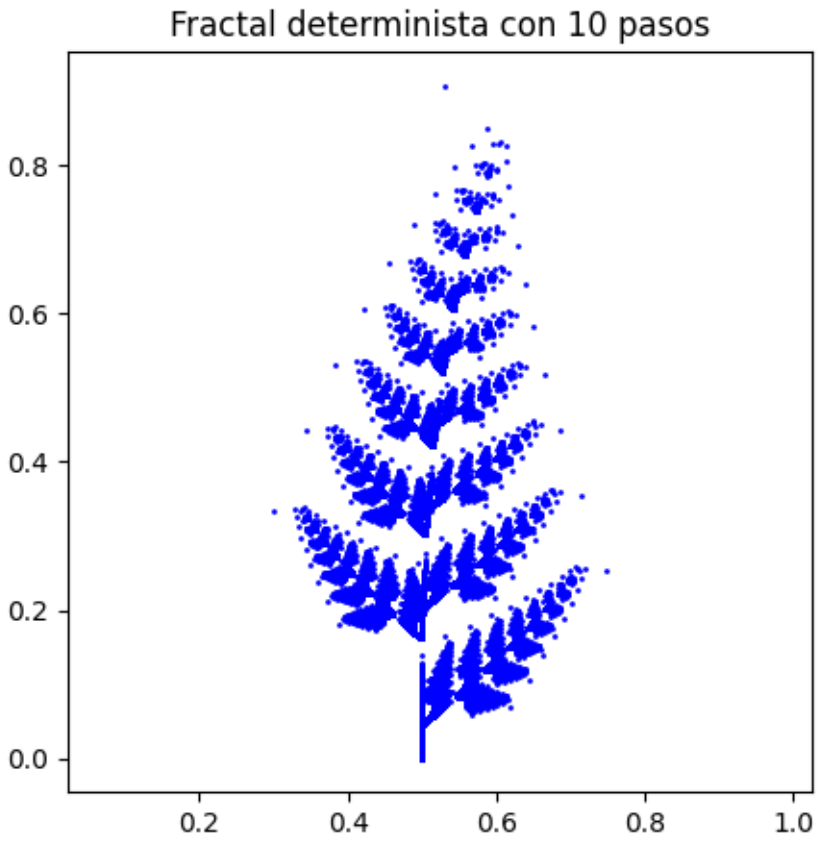
```
def f5(p):
    x, y = p
    return (-0.637 * x + 0.000 * y + 0.8562, 0.000 * x + 0.501 * y + 0.2513)

# El SFI es la lista de estas funciones
sfi = [f1, f2, f3, f4, f5]

pasos = 7

semilla = (0,0)

ejecucion(sfi, pasos, semilla)
```



1.2.8 8. El helecho de Barnsley:

[10]:
```
def f1(p):
    x, y = p
    return (0.849 * x + 0.037 * y + 0.075, (-0.037) * x + 0.849 * y + 0.1830)

def f2(p):
```

```
    x, y = p
    return (0.197 * x + (-0.226) * y + 0.400, 0.226 * x + 0.197 * y + 0.0490)

def f3(p):
    x, y = p
    return (-0.150 * x + 0.283 * y + 0.575, 0.260 * x + 0.237 * y + (-0.0840))

def f4(p):
    x, y = p
    return (0.000 * x + 0.000 * y + 0.500, 0.000 * x + 0.160 * y + 0.0000)

# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3, f4]

pasos = 10

semilla = (0,0)

ejecucion(sfi, pasos, semilla)
```



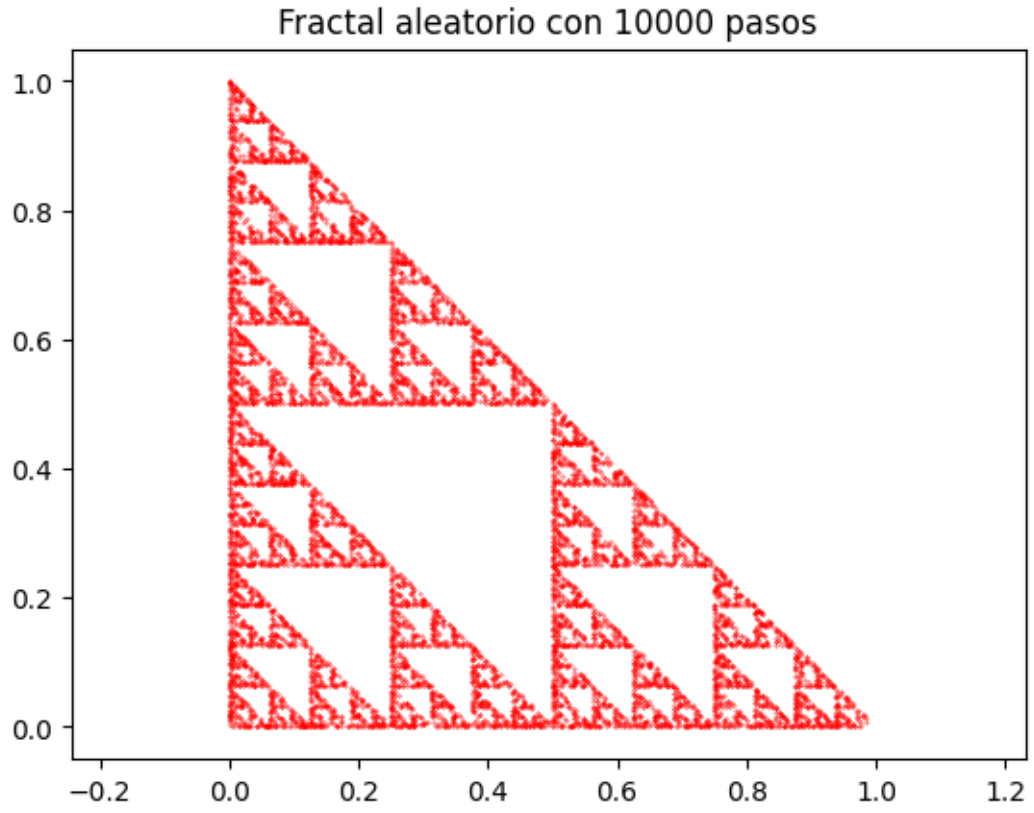
1.3 Ejercicio 3: Escribe un programa que implemente el algoritmo aleatorio.

[11]:
```python
import numpy as np
import matplotlib.pyplot as plt
import random

# Implementamos la función para el algoritmo aleatorio
def fractal_aleatorio(sfi, pasos, semilla):
    """
    sfi: Lista de funciones afines que definen el sistema de funciones iteradas␣
 ↪(SFI).
    pasos: Número de iteraciones a realizar.
    semilla: Punto inicial (x, y) desde donde comienza la iteración.
    """
    # Se inicializa el punto actual con la semilla
    punto_actual = semilla
    puntos = []  # Lista para almacenar los puntos generados

    for _ in range(pasos):
        funcion = random.choice(sfi)  # Se elige una función aleatoria del SFI
        punto_actual = funcion(punto_actual)  # Se aplica la función al punto␣
 ↪actual
        puntos.append(punto_actual)  # Se almacena el nuevo punto

    return puntos  # Se devuelve la lista de puntos generados

# Función para ejecutar el algoritmo y dibujar el resultado
def ejecucion_aleatorio(sfi, pasos, semilla):
    """
    sfi: Lista de funciones afines del sistema iterado.
    pasos: Número de iteraciones.
    semilla: Punto inicial (x, y).
    """
    # Ejecutamos el algoritmo determinista
    puntos = fractal_aleatorio(sfi, pasos, semilla)

    # Separamos en listas las coordenadas x e y
    x_vals, y_vals = zip(*puntos)

    # Se grafica el fractal
    plt.scatter(x_vals, y_vals, s=0.1, color='red')
    plt.title(f'Fractal aleatorio con {pasos} pasos')
    plt.axis('equal')
    plt.show()
```

1.4 Ejercicio 4: Pruébalo con algunos de los SFI de las transparencias de clase.

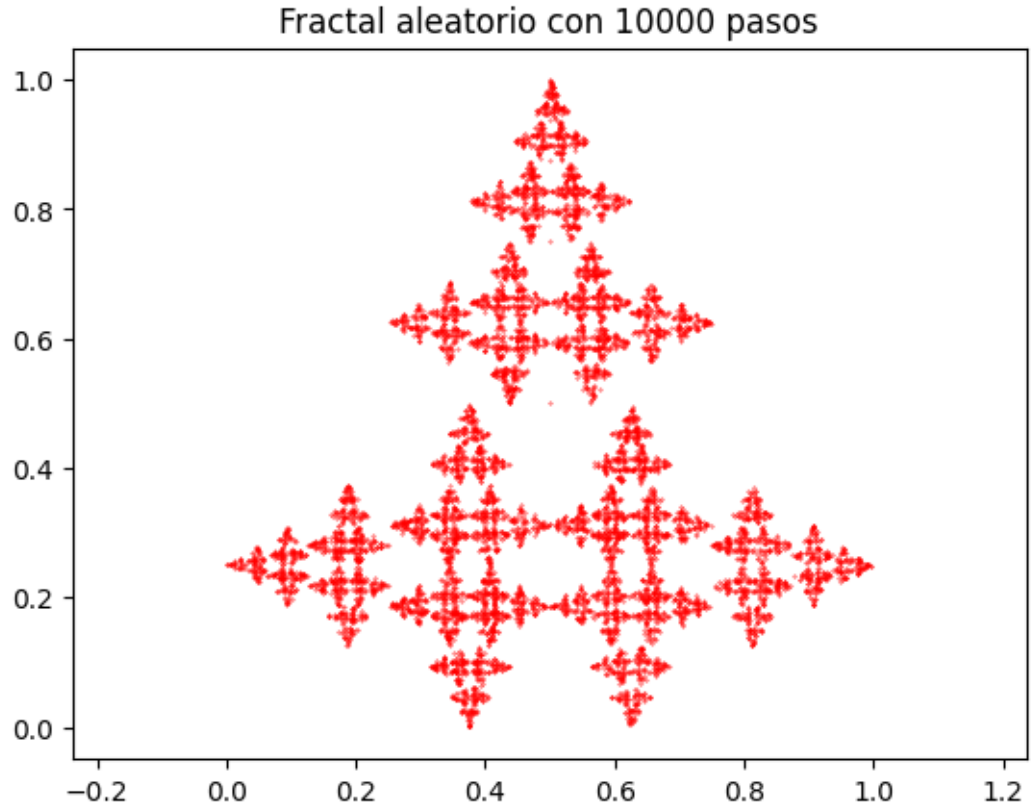
1.4.1 1. Sierpinski:

[12]:
```
# Definimos las transformaciones (f1, f2, f3) según la tabla dada
def f1(p):
    x, y = p
    return (0.5 * x, 0.5 * y)

def f2(p):
    x, y = p
    return (0.5 * x + 0.5, 0.5 * y)

def f3(p):
    x, y = p
    return (0.5 * x, 0.5 * y + 0.5)

# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3]

pasos = 10000
semilla = (0,0)

ejecucion_aleatorio(sfi, pasos, semilla)
```



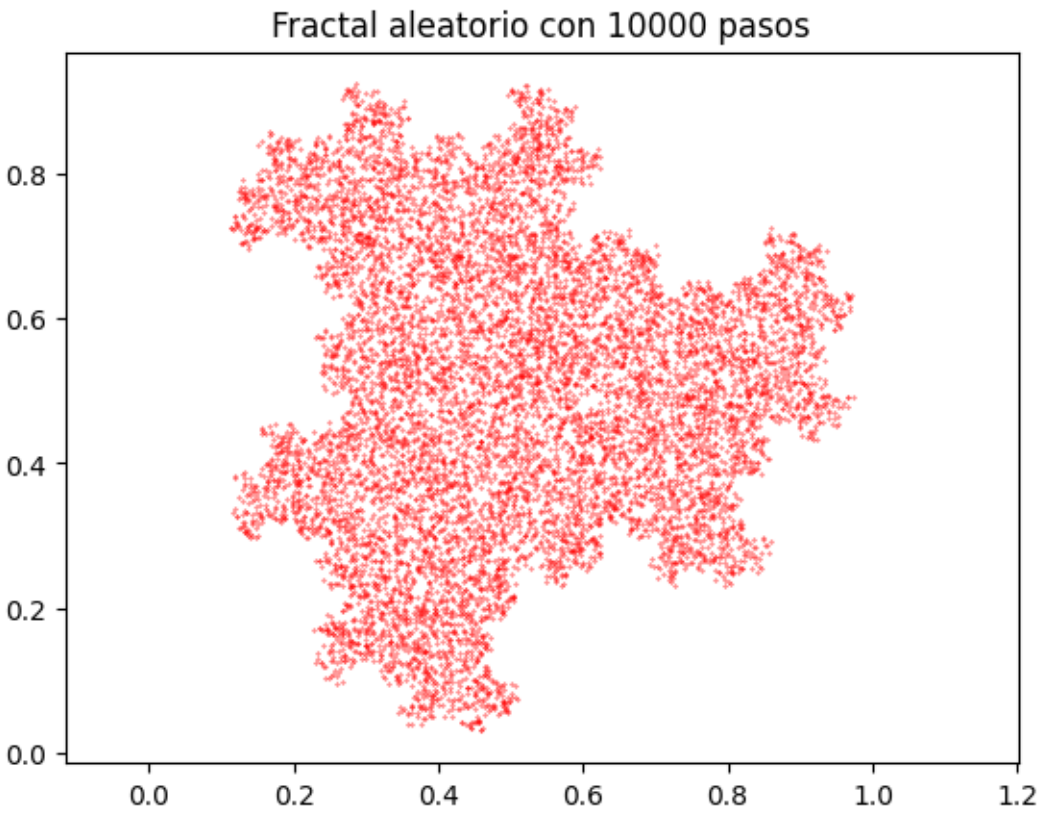
1.4.2 2. Fractal:

[13]:
```
# Definimos las transformaciones (f1, f2, f3) según la tabla dada
def f1(p):
    x, y = p
    return (-0.5 * y + 0.5, 0.5 * x)

def f2(p):
    x, y = p
    return (0.5 * y + 0.5, -0.5 * x + 0.5)

def f3(p):
    x, y = p
    return (0.5 * x + 0.25, 0.5 * y + 0.5)

# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3]
pasos = 10000
semilla = (0,0)

ejecucion_aleatorio(sfi, pasos, semilla)
```



1.4.3 3. Fractal:

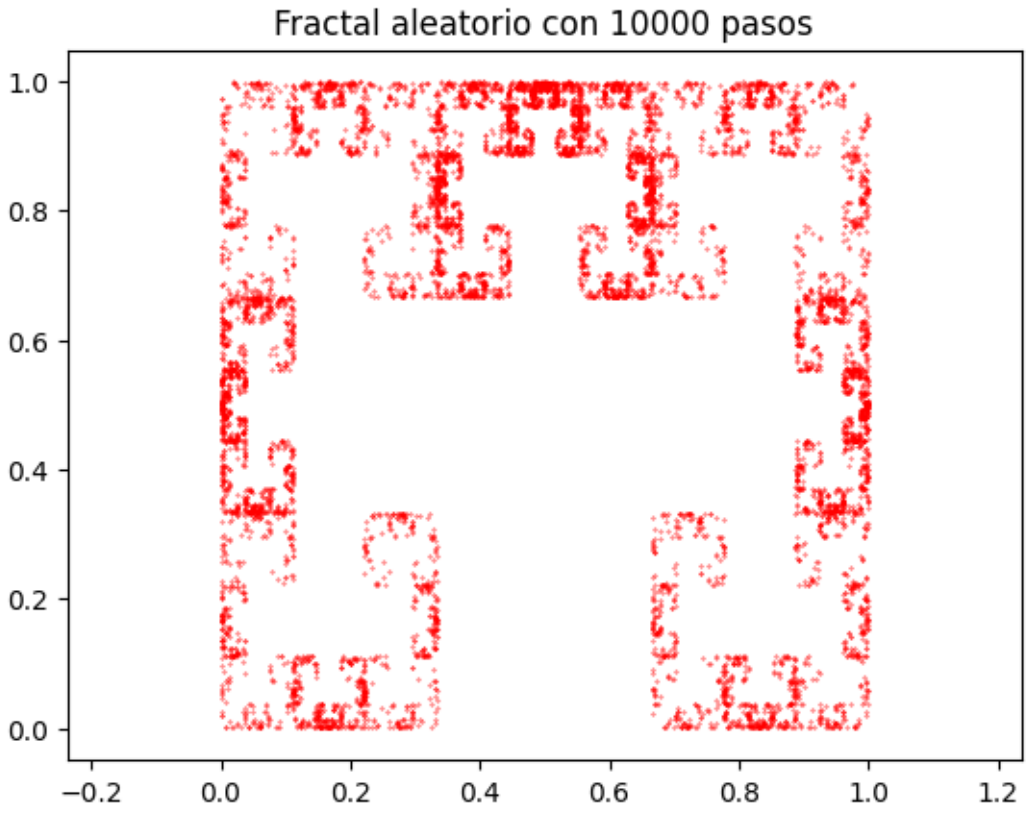
[14]:
```
def f1(p):
    x, y = p
    return (0.577 * y + 0.0951, -0.577 * x + 0.5893)

def f2(p):
    x, y = p
    return (0.577 * y + 0.4413, -0.577 * x + 0.7893)

def f3(p):
    x, y = p
    return (0.577 * y + 0.0952, -0.577 * x + 0.9893)

# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3]

pasos = 10000

semilla = (0,0)

ejecucion_aleatorio(sfi, pasos, semilla)
```



1.4.4 4. Fractal:

[15]:
```
def f1(p):
    x, y = p
    return (0.333 * x + 0.333, 0.333 * y + 0.666)

def f2(p):
    x, y = p
    return (0.333 * y + 0.666, x)

def f3(p):
    x, y = p
    return (-0.333 * y + 0.333, x)


# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3]

pasos = 10000

semilla = (0,0)
```

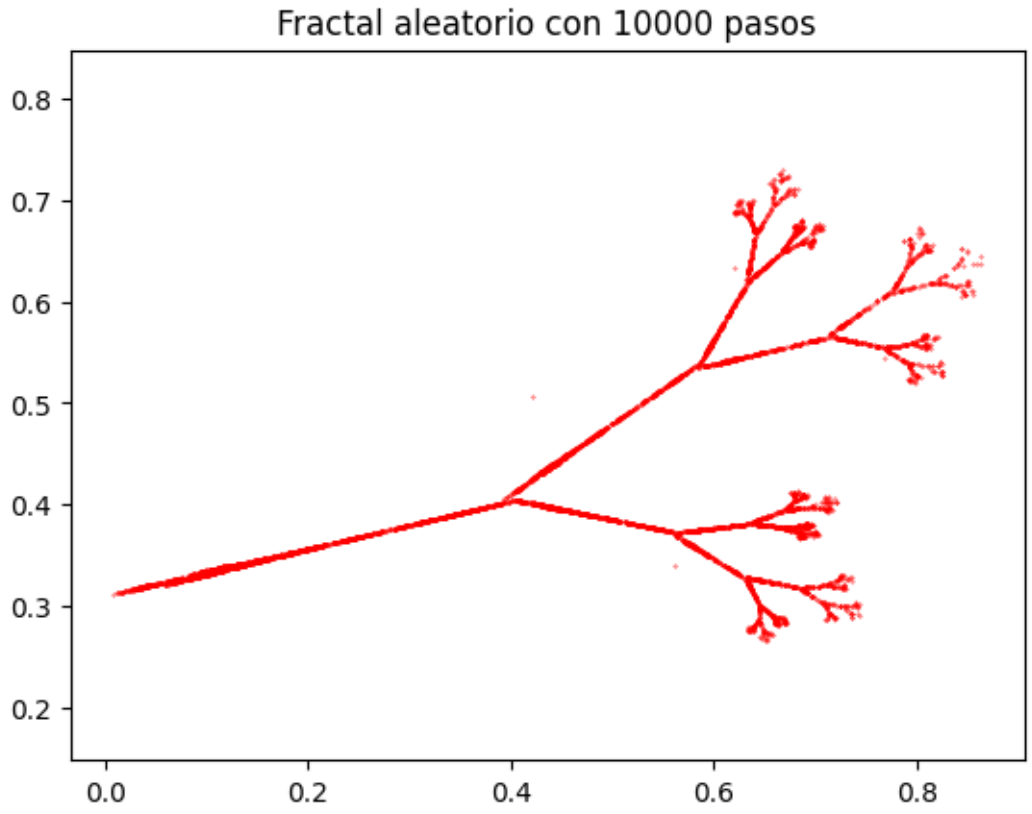
```
ejecucion_aleatorio(sfi, pasos, semilla)
```



1.4.5 5. Fractal:

[16]:
```
def f1(p):
    x, y = p
    return (0.387 * x + 0.430 * y + 0.2560, 0.430 * x + (-0.387) * y + 0.5220)

def f2(p):
    x, y = p
    return (0.441 * x + (-0.091) * y + 0.4219, (-0.009) * x + (-0.322) * y + 0.
↪5059)

def f3(p):
    x, y = p
    return (-0.468 * x + 0.020 * y + 0.4000, (-0.113) * x + 0.015 * y + 0.4000)

# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3]
```

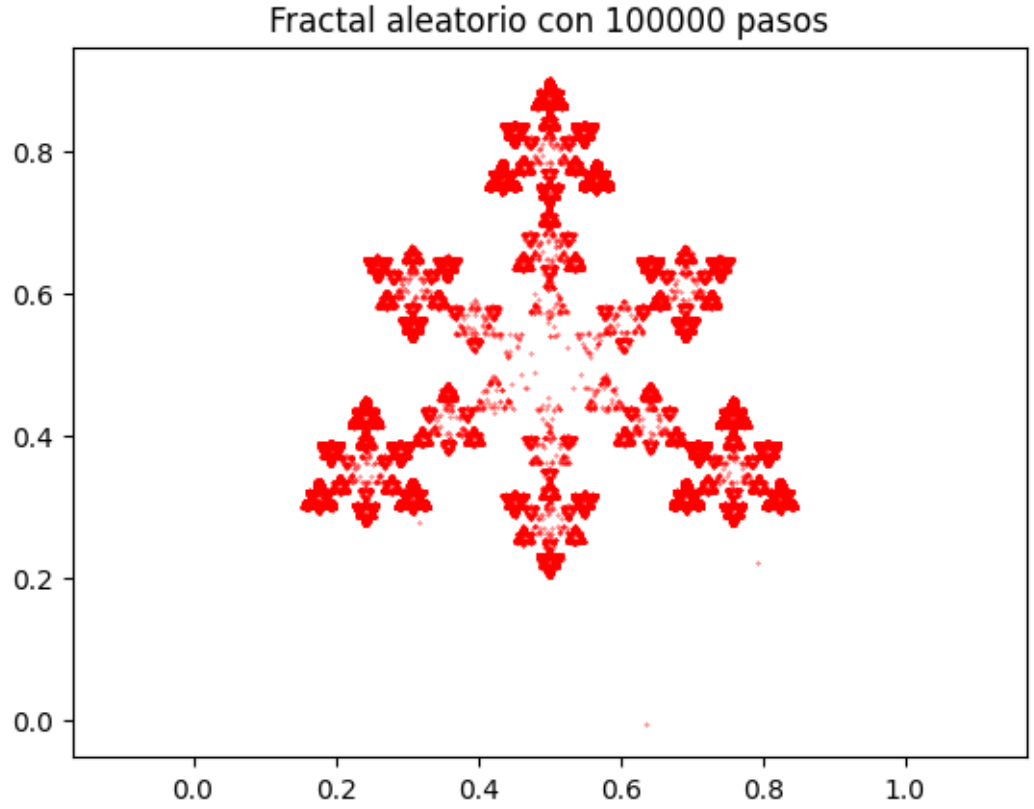
```
pasos = 10000

semilla = (0,0)

ejecucion_aleatorio(sfi, pasos, semilla)
```



1.4.6 6. Fractal:

[17]:
```
def f1(p):
    x, y = p
    return (0.255 * x + 0.000 * y + 0.3726, 0.000 * x + 0.255 * y + 0.6714)

def f2(p):
    x, y = p
    return (0.255 * x + 0.000 * y + 0.1146, 0.000 * x + 0.255 * y + 0.2232)

def f3(p):
    x, y = p
    return (0.255 * x + 0.000 * y + 0.6306, 0.000 * x + 0.255 * y + 0.2232)

def f4(p):
```

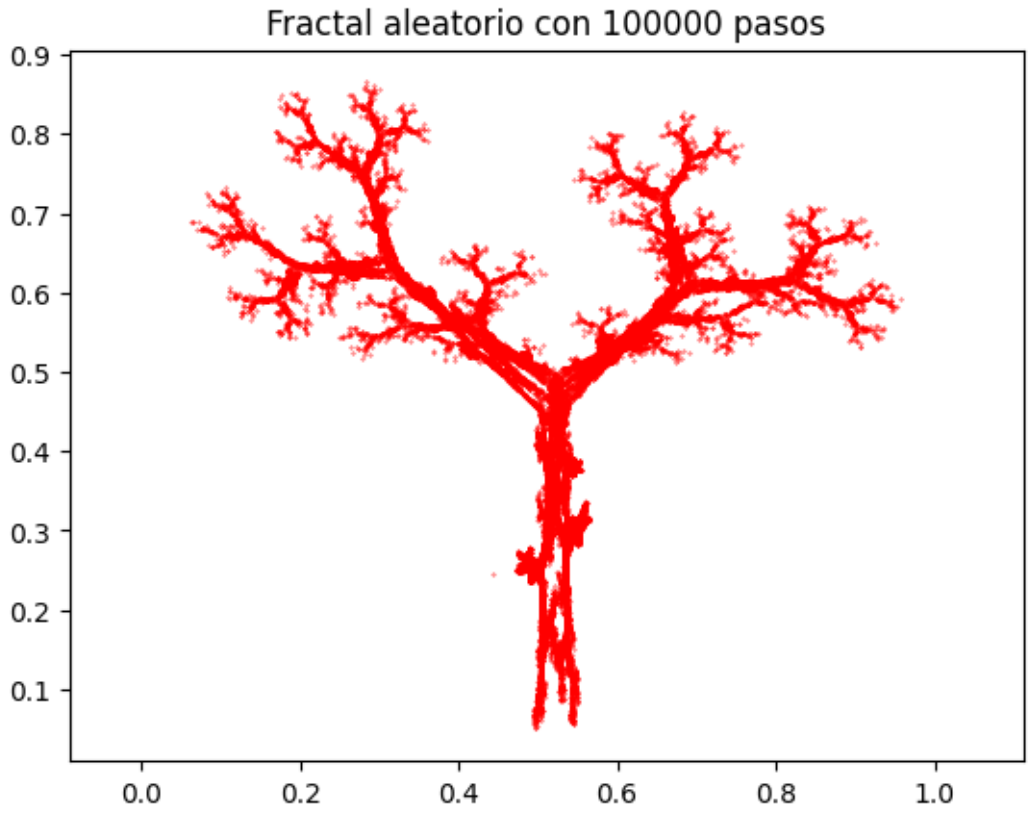
```
    x, y = p
    return (0.370 * x + (-0.642) * y + 0.6356, 0.642 * x + 0.370 * y + (-0.
↪0061))

# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3, f4]
pasos = 100000
semilla = (0,0)
ejecucion_aleatorio(sfi, pasos, semilla)
```



1.4.7 7. Fractal:

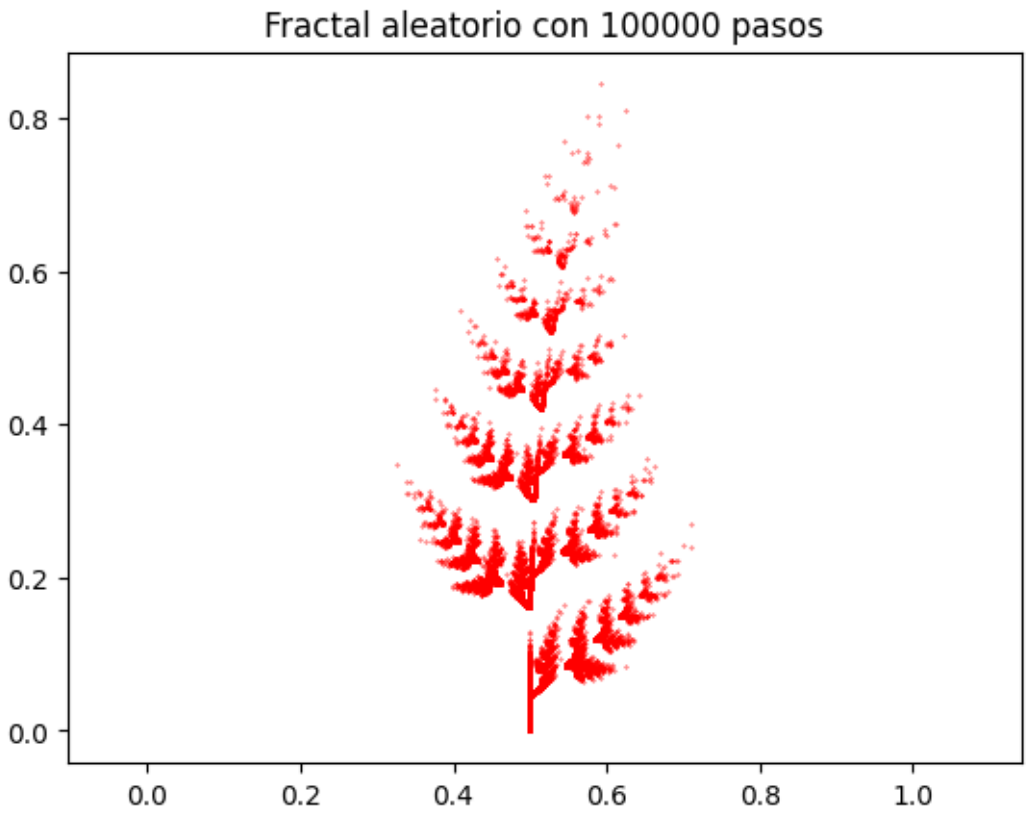
[18]:
```
def f1(p):
    x, y = p
    return (0.195 * x + (-0.488) * y + 0.4431, 0.344 * x + 0.443 * y + 0.2452)

def f2(p):
    x, y = p
    return (0.462 * x + 0.414 * y + 0.2511, (-0.252) * x + 0.361 * y + 0.5692)

def f3(p):
```

```
    x, y = p
    return (-0.058 * x + (-0.070) * y + 0.5976, 0.453 * x + (-0.111) * y + 0.
 ↪0969)

def f4(p):
    x, y = p
    return (-0.035 * x + 0.070 * y + 0.4884, (-0.469) * x + (-0.022) * y + 0.
 ↪5069)

def f5(p):
    x, y = p
    return (-0.637 * x + 0.000 * y + 0.8562, 0.000 * x + 0.501 * y + 0.2513)

# El SFI es la lista de estas funciones
sfi = [f1, f2, f3, f4, f5]

pasos = 100000

semilla = (0,0)

ejecucion_aleatorio(sfi, pasos, semilla)
```



1.4.8 8. El helecho de Barnsley:

[19]:
```
def f1(p):
    x, y = p
    return (0.849 * x + 0.037 * y + 0.075, (-0.037) * x + 0.849 * y + 0.1830)

def f2(p):
    x, y = p
    return (0.197 * x + (-0.226) * y + 0.400, 0.226 * x + 0.197 * y + 0.0490)

def f3(p):
    x, y = p
    return (-0.150 * x + 0.283 * y + 0.575, 0.260 * x + 0.237 * y + (-0.0840))

def f4(p):
    x, y = p
    return (0.000 * x + 0.000 * y + 0.500, 0.000 * x + 0.160 * y + 0.0000)

# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3, f4]

pasos = 100000

semilla = (0,0)

ejecucion_aleatorio(sfi, pasos, semilla)
```

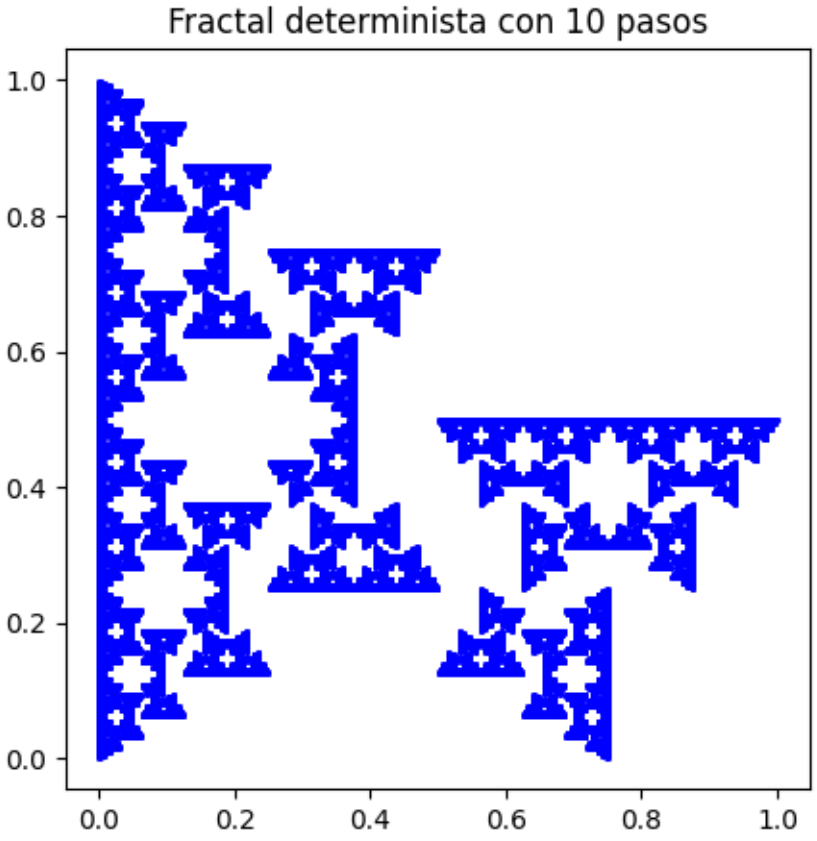


1.5 Ejercicio 5: Verifica la respuesta a los ejercicios de "problema inverso" de la hoja de problemas, comprobando que el SFI encontrado genera la imagen correspondiente.

1.5.1 Apartado a:

[20]:
```
def f1(p):
    x, y = p
    return (0.5 * x + 0.0 * y + 0.0, 0.0 * x + 0.5 * y + 0.5)

def f2(p):
    x, y = p
    return (0.5 * x + 0.0 * y + 0.0, 0.0 * x + (-0.5) * y + 0.5)

def f3(p):
    x, y = p
    return (0.0 * x + 0.5 * y + 0.5, (-0.5) * x + 0.0 * y + 0.5)


# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3]
```

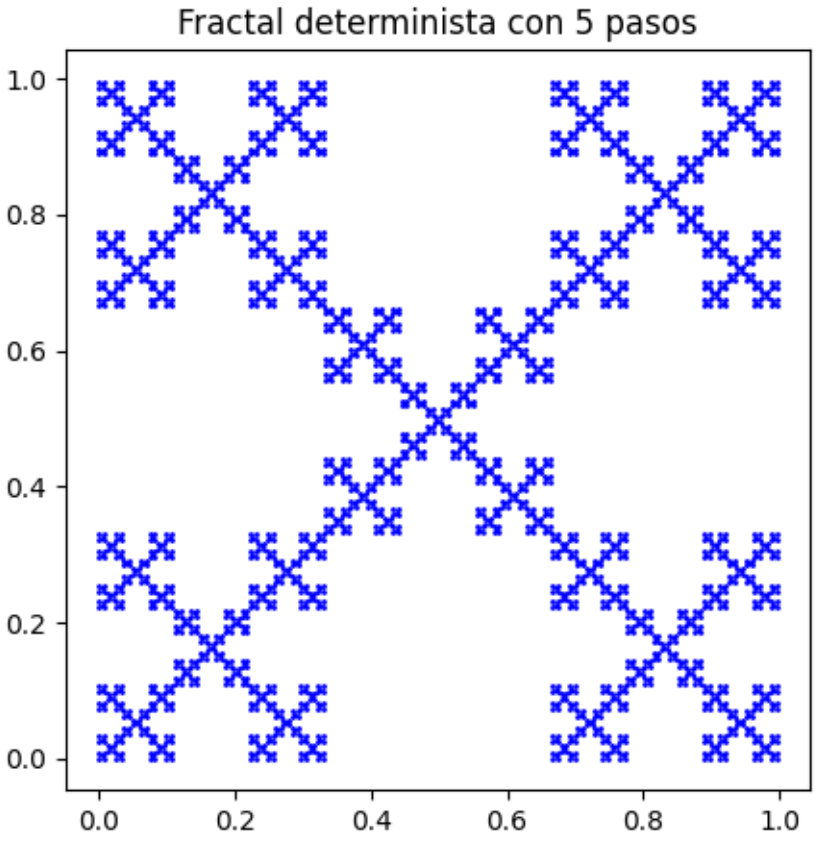
```
pasos = 10

semilla = (0,0)

ejecucion(sfi, pasos, semilla)
```



1.5.2 Apartado b:

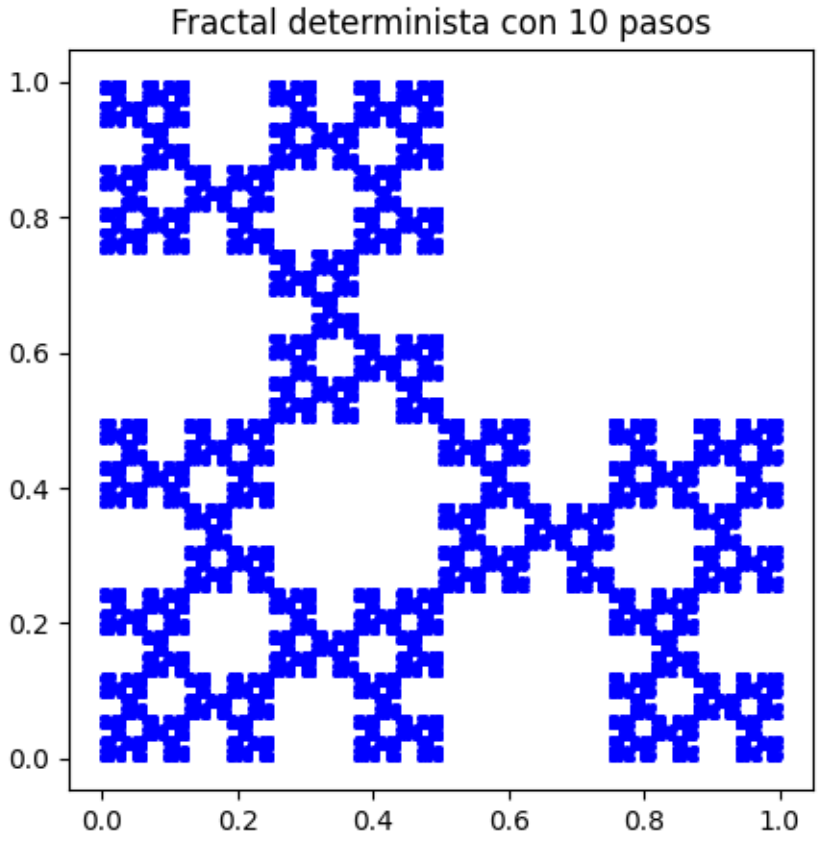
[21]:
```
def f1(p):
    x, y = p
    return (1/3 * x, 1/3 * y)

def f2(p):
    x, y = p
    return (1/3 * x , 1/3 * y + 2/3)

def f3(p):
    x, y = p
    return (1/3 * x + 1/3, 1/3 * y + 1/3)
```

```python
def f4(p):
    x, y = p
    return (1/3 * x + 2/3, 1/3 * y)

def f5(p):
    x, y = p
    return (1/3 * x + 2/3, 1/3 * y + 2/3)


# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3, f4, f5]

pasos = 5

semilla = (0,0)

ejecucion(sfi, pasos, semilla)
```



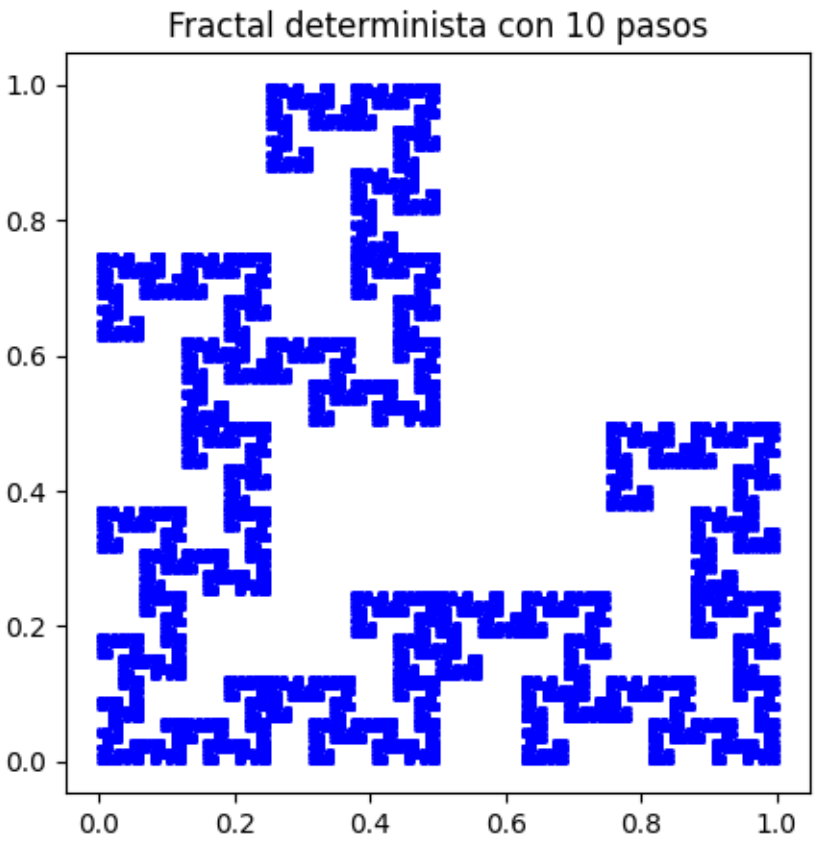
1.5.3 Apartado c:

[22]:
```
def f1(p):
    x, y = p
    return (0.5 * x, 0.5 * y)

def f2(p):
    x, y = p
    return ((-0.5) * y + 0.5, (-0.5) * x + 1)

def f3(p):
    x, y = p
    return ((-0.5) * y + 1, (-0.5) * x + 0.5)


# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3]

pasos = 10

semilla = (0,0)

ejecucion(sfi, pasos, semilla)
```



1.5.4 Apartado d:

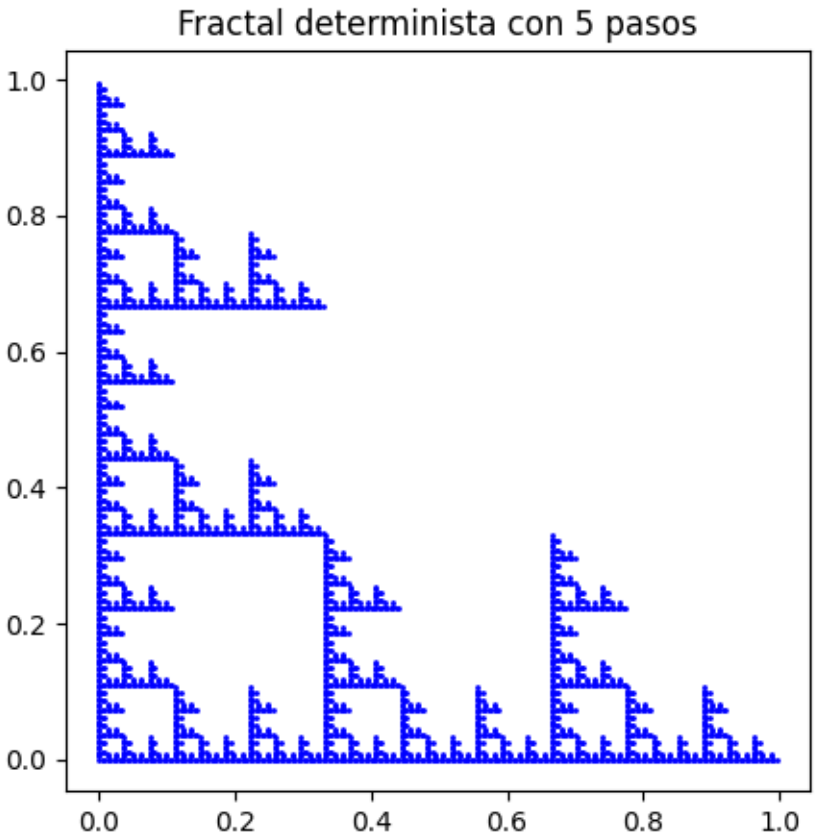
[23]:
```
def f1(p):
    x, y = p
    return (0.5 * x, 0.5 * y)

def f2(p):
    x, y = p
    return ((-0.5) * y + 1, 0.5 * x)

def f3(p):
    x, y = p
    return ((-0.5) * y + 0.5, 0.5 * x + 0.5)


# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3]

pasos = 10

semilla = (0,0)

ejecucion(sfi, pasos, semilla)
```



1.5.5 Apartado e:

[24]:
```
def f1(p):
    x, y = p
    return (1/3 * x, 1/3 * y)

def f2(p):
    x, y = p
    return (1/3 * x + 1/3, 1/3 * y)

def f3(p):
    x, y = p
    return (1/3 * x + 2/3, 1/3 * y)

def f4(p):
    x, y = p
    return (1/3 * x, 1/3 * y + 1/3)

def f5(p):
    x, y = p
```

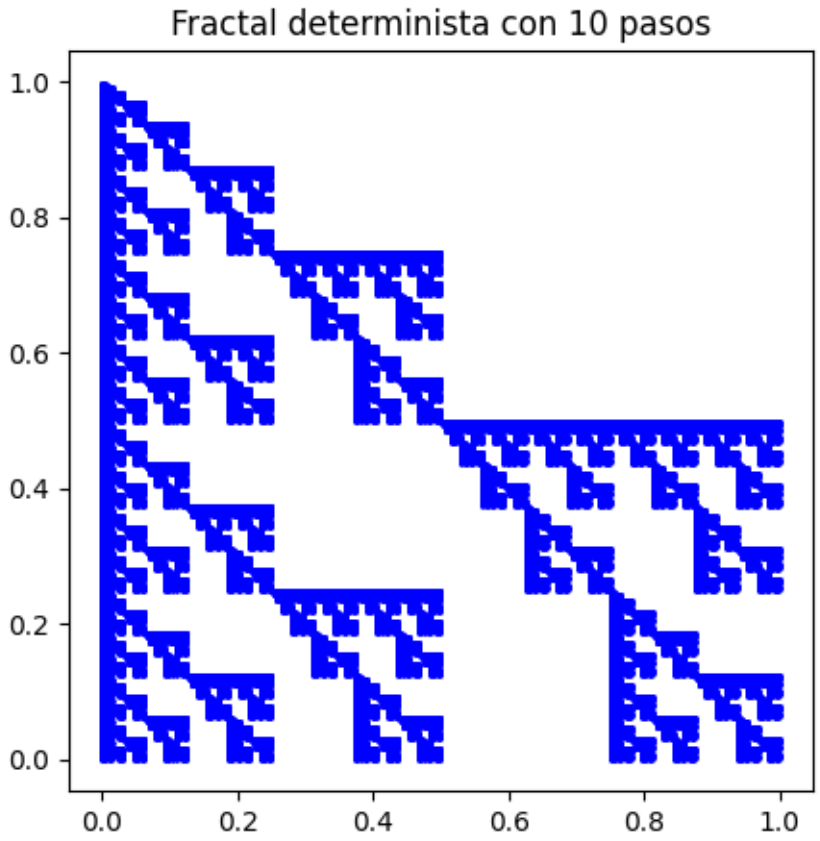
```
    return (1/3 * x, 1/3 * y + 2/3)


# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3, f4, f5]

pasos = 5

semilla = (0,0)

ejecucion(sfi, pasos, semilla)
```



1.5.6 Apartado f:

[25]:
```
def f1(p):
    x, y = p
    return (0.5 * x, 0.5 * y)

def f2(p):
    x, y = p
```

```
    return (0.5 * x, 0.5 * y + 0.5)

def f3(p):
    x, y = p
    return ((-0.5) * y + 1, (-0.5) * x + 0.5)


# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3]

pasos = 10

semilla = (0,0)

ejecucion(sfi, pasos, semilla)
```



1.5.7 Apartado g:

[26]:
```python
def f1(p):
    x, y = p
    return (0.5 * x, 0.5 * y)

def f2(p):
    x, y = p
    return (0.5 * x, (-0.5) * y + 1)

def f3(p):
    x, y = p
    return ((-0.5) * x + 1, 0.5 * y)


# El SFI es la lista de estas 3 funciones
sfi = [f1, f2, f3]

pasos = 10

semilla = (0,0)

ejecucion(sfi, pasos, semilla)
```