Lab 2: Chaos Game

Lucas Rodrigues[1]

[1]Florida Atlantic University

Nonlinear Dynamic Systems , Fall 2020

Chaos Game

Experiment goals:

- Study how changing the number of points and iterations we can vastly change how a resulting plot looks.
- Experiment with repeated patterns that generate fractals
- inspect implemented code to generate the famous Sierpinski's Triangle and Barnsley's Fern.

Introduction

Why is this important?

- The field has many applications across natural sciences and pure mathematics
- Study of the code leads to greater insight on how the systems are generated and how to analyze them.

Sierpinski's Triangle

The code used to generate the triangle was:

```
def midpoint(P, Q):
    return (0.5*(P[0] + Q[0]), 0.5*(P[1] + Q[1]))
```

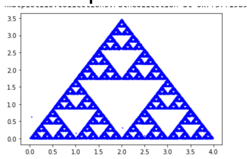
```
vertices = [(0, 0), (2, 2*np.sqrt(3)), (4, 0)]
n = 25 # Change this value and see what happens
```

```
x = [0]*n
y = [0]*n
x[0] = random()
y[0] = random()
```

```
for i in range(1, n):
    x[i], y[i] = midpoint( vertices[randint(0, 2)], (x[i-1], y[i-1]) )
```

```
plt.scatter(x, y, color = 'b', s=1)
```

As the parameter n increases, The figure slowly approaches



Barnsley's Fern

The code used to generate the fern was:

```
# Barnsley's Fern

# 1% of the time:x → 0,y → 0.16 y
# 85% of the time:x → 0.85 x + 0.04 y, y → -0.04 x + 0.05 y + 1.6
# 7% of the time:x → 0.2 x - 0.26 y, y → 0.23 x + 0.22 y + 1.6
# 7% of the time:x → -0.15 x + 0.28 y, y → 0.26 x + 0.24 y + 0.44

def pick(p):
    c = np.cumsum(p)
    return bisect(c, np.random.random() * c[-1])

p = np.array([0.01,0.07,0.07,0.85])

eq = [np.array([[0,0,0],[0,0.16,0]]),
      np.array([[0.2,-0.26,0],[0.23,0.22,1.6]]),
      np.array([[-0.15, 0.28, 0],[0.26,0.24,0.44]]),
      np.array([[0.85, 0.04, 0],[-0.04, 0.85, 1.6]])]

n = 25 # Change this value and see what happens
x = np.zeros((n,3))
x[:,2] = 1

for i in range(1,n):
    x[i,:2] = np.matmul(eq[pick(p)],x[i-1,:])

plt.figure(figsize=(10,10))
plt.scatter(x[:,0], x[:, 1], s=3, c="g", marker="s", linewidths=0)
plt.axis("equal"),plt.axis("off");
```

As the parameter n increases, The figure slowly approaches

Analysis and Conclusion

In conclusion, bounded randomness can generate very interesting images when iterated with itself, which changes and becomes more accurate as the number of iterations increase.