Systems Biology – Exercises

Week 3:Fractals

Fractals

Fractals are defined as being self-similar on all scales. This means fractals display this quality when looking at the whole set or only a small piece. As the word *similar* entails, structures do not need to be exact, but can show resemblance. These qualities can be found in nature (e.g., fern as in Figure 1), in mathematical representations (e.g. the Mandelbrot set in Equation 1 or Figure 2), and scientific data (e.g., network data).

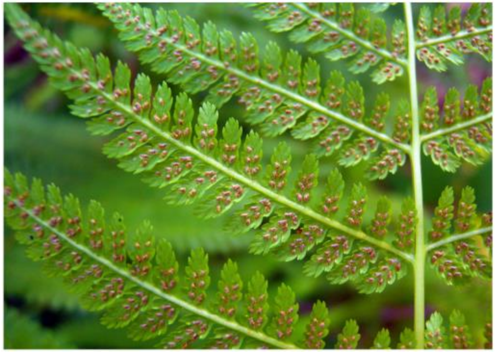


Figure 1: Example of fractals in nature: *Athyrium flix-femina*

Mandelbrot Set

- Some well-known fractal structures include: *Gosper Island, Koch Snowflake, Box Fractal, Sierpiński Sieve, Barnsley's Fern,* and *Mandelbrot Set*[1].
- Arguably the most famous out of the list above is the *Mandelbrot Set*, which we will have a closer look at today.
- **The Mandelbrot Set**: Mathematically, the Mandelbrot set is described as follows:

$$M = \left\{ c \in \mathbb{C} \,|\, f_c^{(n)}(z) \text{ is finite as } n \to \infty \right\} \qquad (1)$$

- The Mandelbrot set is based on the Julia set, both sharing the following equation:

$$Z_{n+1} = Z_n^2 + C \qquad (2)$$

- with$Z_0 = C$. In rendering the set graphically, this value is often set to 0.

1 More details at http://mathworld.wolfram.com/Fractal.html

Mandelbrot Set in Python

```
import matplotlib.pyplot as plt
import numpy as np

n=1000   # line resolution
iteration_number = 100       # number of iterations
M = np.zeros([n,n],int)
xvalues = np.linspace(-2,0.5,n)          # region spanning -2 to 0.5
yvalues = np.linspace(-1,1,n)            # region spanning -1 to 1
for u,x in enumerate(xvalues):
    for v,y in enumerate(yvalues):
        z = 0 + 0j        # initialization
        C = complex(x,y)
        for i in range(iteration_number):
            z = z*z + C
            # the cummulative absolute value of z going to infinity (bound)
            if abs(z) > 2.0:
                M[v,u] = 1
                break
plt.imshow(M,origin="lower", extent=(-2,0.5,-1,1)) # creating the image with axes label
plt.gray() # grayscale image
plt.show() # image output
```

Visualization for Mandelbrot Set

- The Output of code in the last page should look like Figure 2

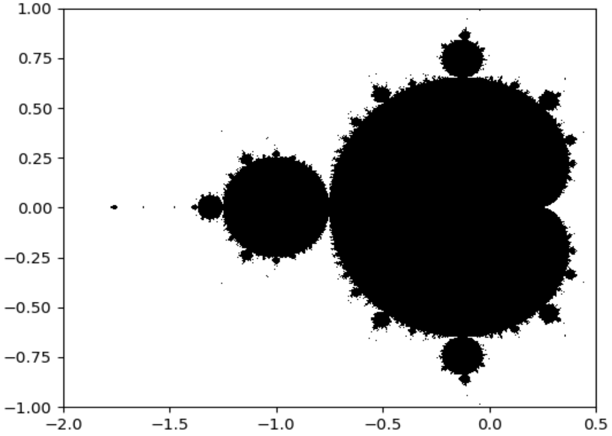
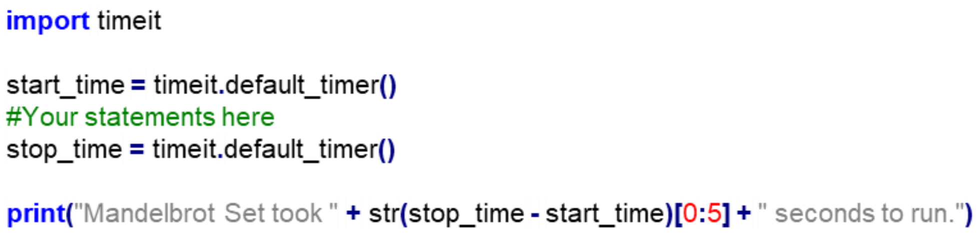


Figure 2: Mandelbrot Plot

Exercise 1

- The code above takes about 20 seconds to run on a modern high-end laptop. How long does it take for you to run it? Use the following code to test it[2]:

Code Runtime



```python
import timeit

start_time = timeit.default_timer()
#Your statements here
stop_time = timeit.default_timer()

print("Mandelbrot Set took " + str(stop_time - start_time)[0:5] + " seconds to run.")
```

2 https://docs.python.org/3/library/timeit.html

Exercise 2

- How can you influence the runtime of the code that produces the Mandelbrot set. Experiment, how the line resolution and iteration steps change your graphical Mandelbrot set.

Homework

3 http://mathworld.wolfram.com/ElephantValley.html
4 http://mathworld.wolfram.com/SeaHorseValley.html

- Due next Wednesday (17:00, 20th, Oct.), submit electronically to Manaba+R.
- File name: YourStudentID_W03_n.py (ID without hyphen, e.g., 12345678901_W03_1.py).
- Your code must include your own comments for all code sections. Go line-by-line. Comments in your program must be full sentences and reflect your understanding of the code.

Q1. Rewrite the code to display the Mandelbrot set as a function that passes the following arguments:

- line resolution
- number of iterations
- left, right, upper, and lower boundaries

Q2. Research two regions of the Mandelbrot set, the Elephant Valley[3] and the Sea Horse Valley[4]. Write a new function, which lets you choose between displaying the full Mandelbrot set, the Elephant Valley and the Sea Horse Valley.

- Q1 → YourStudentID_W03_1.py file
- Q2 → YourStudentID_W03_2.py file

Systems Biology – Exercises

Week 4: Emergent Systems