Systems Biology – Exercises

Week 4: Emergent Systems

Emergent Systems

- The term Emergent Systems—or Emergence describes a phenomenon rather than the limitation to systems--is recently being used to describe characteristics of computer algorithms, which produce complex behavior out of simple elements. Emergence is often used in context of nature or biologically inspired algorithms, as complexity “emerges” out of putting accessible elements into an interacting compound.
- **Emergence**: In the case of fractal visualization, out of simple mathematical formulae emerge complex graphical structures. Artificial life, artificial neural networks, and swarm behavior are other examples of emergent systems.
- The following algorithm is our next step in exploring bio-inspired algorithms. It is also in preparation of the upcoming exercises covering artificial life and cellular automata.

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

# Size of the grid (universe)
universe = 50
# Alive cells = "on"
alive = 1
# Dead cells = "off"
dead = 0
# List of values "on" and "off" used to populate the initial universe
vals = [alive, dead]
# Populate the universe with random cells
grid = np.random.choice(vals, universe*universe, p=[0.1, 0.9]).reshape(universe,
universe)
# Define a function to animate the cells in the universe
# animated_universe is a void function, but placeholder values *arg,
# **kwards necessary for matplotlib initial value (framerate)
def animated_universe(framenumber, *arg, **kwargs):
    global grid
    global animatedcount
    newGrid = grid.copy()
    for i in range(universe):
        for j in range(universe):
            total = (grid[i, (j-1)%universe] +
                         grid[i, (j+1)%universe] +
                         grid[(i-1)%universe, j] + grid[(i+1)%universe, j] +
            grid[(i-1)%universe, (j-1)%universe] + grid[(i-1)%universe,
            (j+1)%universe] +
            grid[(i+1)%universe, (j-1)%universe] + grid[(i+1)%universe,
            (j+1)%universe])/alive
            # Rules for a cell dying off or staying alive
            if grid[i, j]  == alive:
                if (total < 2) or (total > 3):
                    newGrid[i, j] = dead
            else:
                if total == 3:
                    newGrid[i, j] = alive
    grid = newGrid.copy()
    mat.set_data(grid)
    return mat

fig, ax = plt.subplots()
mat = ax.matshow(grid)
# interval : number, optional
# Delay between frames in milliseconds. Defaults to 200.
# save_count : int, optional
# The number of values from frames to cache.
ani = animation.FuncAnimation(fig, animated_universe, interval=3)
# use higher interval to observe oscillator
# ani = animation.FuncAnimation(fig, animated_universe, interval=60)
plt.show()

ani.save('animation.gif', writer='imagemagick', fps=30)
```

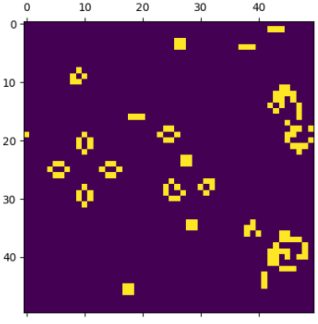
Exercise 1

- Does your code eventually come to a halt (or gets stuck)? Experiment with the initial universe size, the number of alive and dead cells.

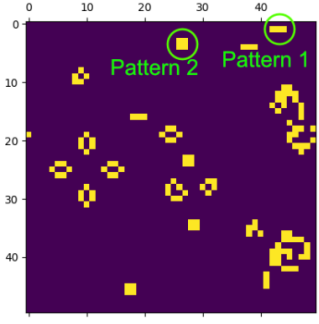
- Make note on how long it took to get to a halt state and make a screenshot.

Exercise 2

- While your code is running, make five screenshots (e.g., Figure 1) and identify at least three patterns in your graphic.



(a) Game of Life Screenshot



(b) Game of Life Screenshot - Annotated

Figure 1: Game of Life Screenshot

Patterns

- Use the following website as reference for patterns: http://www.math.com/students/wonders/life/life.html. There are, in fact, many more patterns, but manual identification would prove insurmountable (http://conwaylife.appspot.com/library).

- You can either use an image editor to write comments on the screenshot directly (Figure 1b), or refer to the coordinate system as follows:
  - Pattern 1: c(41-45,2-3): name of pattern
  - Pattern 2: c(25-27,4-6): name of pattern
  - etc.

Homework

Due next Wednesday (17:00, 27th, Oct.) electronically to manaba+R.

• File format: YourStudentID_Week04_n. py (ID without hyphen, e.g., 12345678901_Week04_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. The code can visualize the pattern “Beehive”. Select one oscillator (not the Beehive) that you find interesting and visualize it.

Q2. Expand the source code in Q1 to animate the oscillator. You can use the Python code from page 3 or code an alternative method.

Q1 → YourStudentID_Week04_1.py file, Q2 → YourStudentID_Week04_2.py file

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

universe = np.zeros((5, 6))
beehive = [[0, 1, 1, 0],
           [1, 0, 0, 1],
           [0, 1, 1, 0]]

universe[1:4, 1:5] = beehive

plt.imshow(universe, cmap='binary')
plt.show()
```

visualizing the pattern “Beehive”

Systems Biology – Exercises

Week 5: Artificial Life