P2

edgardjfc

February 2022

1 Introduction

For this assignment the goal was to make a series of simulations of the game of life and produce a graph that showcases the progression of the different variables. With variable area of the board (10, 15 and 20 cells per side) and variable initial population density (0.2, 0.4, 0.6 and 0.8). This would show us how the population would progress and the chances it has of survival.

2 Code

Based on the code by Dr. E. Shaeffer [1] the following code was designed, using similar logic to produce the rule set of the game of life.

```
def makeInitialConditions(dimension, density):
    numSquares = dimension**2
    initialConditions = [1 * (random() < density) for i in range(numSquares)]
    initialGrid = np.reshape(initialConditions, (dimension,dimension))
    return initialGrid
```

With this function we can create the initial conditions for the first step of the graph. This will function as a base to calculate and produce the following steps of the game. We use it as a function so that way we can apply it multiple times for each permutation.

```
def simulate(steps, dimension, density):
    grid = makeInitialConditions(dimension, density)

    for s in range(steps):
        grid = step(grid, dimension)

    return 1 * (np.sum(grid) > 0)

def step(grid, dimension):
    nextGrid = np.zeros((dimension,dimension))
    for pos in range(dimension**2):
        x = pos % dimension
        y = pos // dimension
        surroundings = grid[max(0,x-1):min(dimension,x+2),
                            max(0,y-1):min(dimension,y+2)]
        nextGrid[x,y] = 1 * (np.sum(surroundings) - grid[x,y] == 3)
    return nextGrid
```

The ”simulate” function is used to produce a single state of the simulation given specific values, returning the state of the grid in the initial conditions. The ”step” function is used to calculate the next step that would go immediately after the previous state of the game.

3 Results

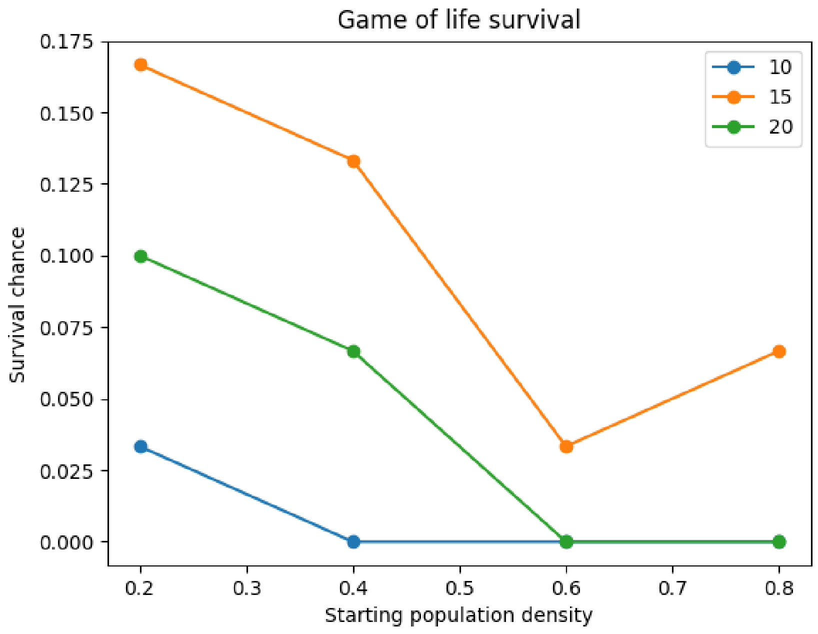

Figure 1: Plotted data

4 Interpretation

We can see in the plotted data (fig.1) how the graphs evolve along with its survival chances. We can observe how the moderate approach of having a 15 by 15 units board yields the most favorable outcome of survival in all cases. We can also observe how the smallest 10 by 10 board has the least chance of survival along the entire board. However a common trend in all of the lines is that the lowest the starting population is, the higher the chances of survival are.

5 Crystaline growth

As an alternate version of the game of life, we produced a ruleset that would allow the cells to keep on growing without any rule that would kill any of the cells. This would give us a game of life where the cells spread out in a way similar to that of a crystal forming. Starting from small seeds and spreading until it reaches the boundaries of another crystal structure.

```python
def simulate(steps, dimension, density):
    grid = makeInitialConditions(dimension, density)

    for s in range(steps):
        grid = step(grid, dimension)

    return grid

isAliveFunc = np.vectorize(lambda x: 1 * (x>0))

def step(grid, dimension):
    nextGrid = np.zeros((dimension,dimension))
    for pos in range(dimension**2):
        x = pos % dimension
        y = pos // dimension
        surroundings = grid[max(0,x-1):min(dimension,x+2),
                            max(0,y-1):min(dimension,y+2)]
        if surroundings.all():
            nextGrid[x,y] = grid[x,y]
        else:
            alivesurroundings = np.sum(isAliveFunc(surroundings))
            nextGrid[x,y] = grid[x,y] + 1*(alivesurroundings >= 3)
    return nextGrid
```

To achieve the change in the rules of the game, the ”simulate” and the ”step” function were modified. The code first searches around a cell to see if there are dead (empty) spaces, if there are, it grows into it and increases the number of the original cell. If the cell is surrounded entirely by living (occupied) cells, the growth stops.

We can see how the initial state (fig.2) developed into the final state (fig.3) and how it formed the crystal grain boundaries in between the grains.

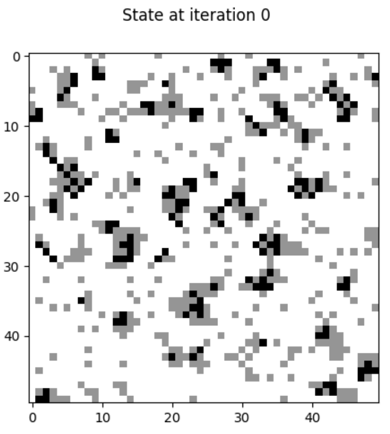

Figure 2: Innitial State

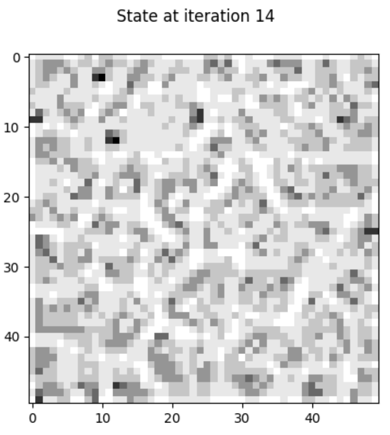

Figure 3: Final State

6 Conclusion

We can conclude that the best outcome for survival lies in having a small starting population, so the cells have space to grow into and not consume each other. Another important detail is that having a moderate size on the board can give us the best chances for survival.

We can also apply the format of the game of life for different purposes, for instance, to simulate the growth of a crystal in a 2d plane, and observe the grain boundaries.

References

[1] E. Schaeffer. *GitHub,gameOfLife.py*. URL: `https://github.com/satuelisa/Simulation/blob/master/CellularAutomata/gameOfLife.py`.