Ojas Bardiya

UID: 505145284

CEE/MAE 20

08/07/20

1. **Game of Life**
   1. **Introduction**

We simulate the John Conway’s Game of Life regarding cells in a Petri dish

with a few additional boundary conditions. We initialize the population of the Petri dish on a **150\*200** grid such that it has it a living cell to dead cell ratio of **1:9**.

* 1. **Model and Theory**

We determine if a cell survives to the next generation or not by using the 3 rules as described –

* If a living cell has **2** or **3** living neighbors, it survives onto the next generation.
* If a living cell has **< 2** or **> 3** living neighbors, it does not survive onto the next generation due to isolation or overcrowding respectively.
* If a dead cell has exactly **3** neighbors, then it becomes alive the next generation.

We denote living cells as having a numeric value of **1** and dead cells as having a numeric value of **0**. We also employ periodic boundary conditions in which we assume that the x-y grid ‘wraps around’ itself, so that each cell has exactly **8** neighbors.

* 1. **Pseudocode**

*Set up the grid with respect to the initial conditions*

*Visualize the grid in its initial condition*

*Set up the number of generations to observe*

*Initialize array to hold count of living cells in each generation*

*Iterate the Game of Life via time-steps*

*Create a new grid to hold values for the next time-step*

*Loop through the rows*

*Check for boundary conditions*

*Loop through the columns*

*Check for boundary conditions*

*Get the sum of neighbors for the current cell*

*If the cell is alive*

*Determine if it survives to the next generation*

*Else if it is dead*

*Determine if it becomes alive during the next generation*

*Plot the current grid*

*Update the current grid matrix*

*sum the total number of living cells*

*Plot the number of living cells with change in generation*

* 1. **Calculations and Results**

**A screenshot of a cell phone

Description automatically generated**

Initial set-up

A screenshot of a cell phone

Description automatically generated

After 300 time-steps

A close up of a map

Description automatically generated

No. of living cells vs generation

On running multiple simulations, we see that there is a sharp decrease in the population of living cells initially, followed by a rapid surge in their levels. The population then more or less plateaus, alternatingly increasing and decreasing.

In most cases, the population after 300 generations is usually slightly greater than the initial population of cells. If we were to draw the best-fitting line for this graph (using least-squares method, for instance) we observe a slight upward slope in most cases.

Now, if we increase the proportion of living cells, say a **1:1** ratio of dead and living cells we observe an exponential decline in the population due to overcrowding. However, if we further decrease the proportion of living cells, say to a ratio of **1:50**, we observe a rapid decline in population due to isolation.

On another note, if a living cell survives in the case it has 4 neighbors as well, we observe the following change in population of living cells over time –

A close up of a map

Description automatically generated

Clearly, this indicates a logarithmic relationship where the number of living cells quickly fill up most of the x-y grid and reach a state of equilibrium.

* 1. **Conclusion**

We observe that a proportion of living to dead cells of **1:9** leads to modest changes in population over time with living cells tending to cluster in a few areas as the number of time-steps increase.

1. **Euler-Bernoulli Beam Bending**
   1. **Introduction**

We solve a system of second-order differential equation(s) to observe the displacement of a simply supported aluminum beam subjected to a single point load. We also determine the difference between the predicted value and observed value for maximum displacement. The initial conditions are set up as follows –

Applied Force(P) = 2000

Outer Radius of beam (R) = 0.013

Inner Radius of beam (r) = 0.011

Length of beam (L) = 1

Distance of applied force (d) = 0.75

Modulus of Elasticity (E) = 70e9

* 1. **Model and Theory**

We discretize the bar by creating 20 evenly spaced nodes in the range [0, L].

We model the system using the equation –

E\*I \*d2y/dx2 = M(x)

where M(x) is given by –

for 0 <= x <= d

and

for d <= x <= L

and the inertia (I) is given by π/4(R4 – r4)

We also determine the theoretical value for maximum displacement using the following formula –

Ymax = where c = min(d, L – d)

We solve the matrix system Ay = b to find out the maximum displacement for different node coordinates.

* 1. **Pseudocode**

*Set up the initial conditions*

*Initialize the node coordinates*

*Set up a matrix A to discretize the second-order differential equation*

*Calculate the bending moment*

*Determine the solution vector b*

*Solve for y using Ay = b*

*Plot displacement vs x-position*

*Determine error between observed and theoretical results*

* 1. **Calculations and Results**

A close up of a map

Description automatically generated

The maximum displacement is **0.0381** and is observed at x = **0.5789** m.

The error observed is **0.02%** for 20 nodes.

On increasing the number of nodes to zero, the error observed is < 0.01% and the maximum displacement is **0.0380**.

For d = 0.5 (with 20 nodes), we observe that the maximum displacement is **0.0543**.

Since the displacement at d is the same as the displacement at L - d, we can say that the maximum displacement generated for any value of d will be in the range [0, 0.0543].

* 1. **Conclusion**

We conclude that a using higher number of nodes to discretize the bar results in a lower degree of error and also that xmin, xmax for the range of displacement is [0, 0.0543], with maximum displacement occurring at d = 0.5.