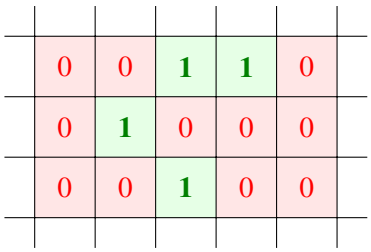
HOMEWORK 7

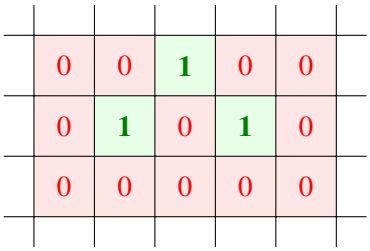
Due: Tuesday, May 22, 2018, 11:55pm

1. **The Game of Life.**

   In this problem, we'll be simulating the fate of living cells using the rules from mathematician John Conway's famous "Game of Life". Cells exist at each point on a 2D grid and can be in one of two states: alive (`1`) or dead (`0`), as shown in the figure below.



Initial Condition



After 1 Update

   In calculating the next generation, an individual cell's survival depends on the state of its 8 nearest neighbors (vertically, horizontally, and diagonally adjacent cells). Your code should enforce the following classical rules to calculate each successive generation:

   - A living cell with either 2 or 3 living neighbors survives on to the next generation.
   - A living cell with fewer than 2 or more than 3 living neighbors does not survive on to the next generation due to isolation or overcrowding, respectively.
   - A dead cell with exactly 3 live neighbors becomes a living cell in the next generation.

   You must employ periodic boundary conditions in this problem to allow the grid to "wrap around" onto itself in both the x and y-direction to avoid artificially influencing cells at the edge of the domain (in the simplified figure above, no boundary conditions were considered and non-visible cells were assumed to hold zeros). Refer to the lecture slides on how to implement these boundary conditions. To visualize the results, we'll be using the MATLAB function `imagesc`. To create an animation of your results, simply call `imagesc` for every generation and use the `drawnow` function to ensure that each timestep is displayed.

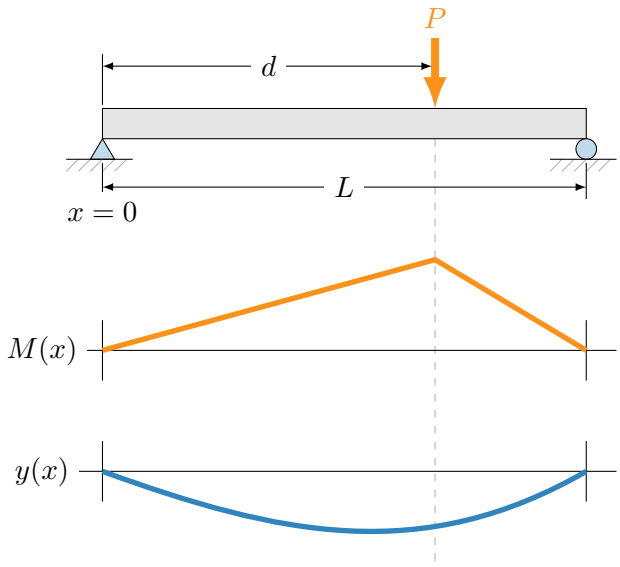
   (a) Demonstrate your method on a grid with `num_rows` $= 150$ and `num_cols` $= 200$. To initialize your game, randomly distribute a 2D array where each cell has probability 0.1 of being alive and 0.9 of being dead (i.e., your initial condition should be a random mix with roughly 10% 1's and 90% 0's). Run your simulation for 300 timesteps (generations) and present both the initial and final distributions in your report (additional images can be included, but be sure to include these two). In a separate figure, plot the number of living cells in your simulation as a function of time.

(b) Can you identify any commonly occurring cell patterns that persist over multiple generations? Include examples and images from your results. Does the total number of living cells over time follow a consistent trend? Experiment with your own birth/death rules (e.g., maybe 4 living neighbors leads to survival, maybe diagonal neighbors are weighted less, maybe you consider neighbors beyond the original 8, etc., be creative). How does your rule change affect the simulation in terms of the visualization produced and the number of living cells over time? Include results from your simulation to support these points.

Note: You are absolutely encouraged to experiment with different birth/death rules, initial conditions, and hand-tuned test cases to generate report results for this problem, but in the final version you upload to CCLE, your code *must* use the classical rules outlined above on a 150×200 grid with a randomly distributed living to dead ratio of $1 : 9$.

2. **Euler-Bernoulli Beam Bending.**

In this problem, we'll solve a system of equations to study the displacement of a simply-supported aluminum beam subjected to a single point load. The bar has length $L = 1\,\mathrm{m}$ and a constant, circular-tube cross section with outer radius $R = 0.013\,\mathrm{m}$ and inner radius $r = 0.011\,\mathrm{m}$. A force, $P = 2000\,\mathrm{N}$, is applied $0.75\,\mathrm{m}$ away from the left-hand edge.



The deflection, y, in an Euler-Bernoulli beam is related to the bending moment, M, as illustrated in the diagram above and in Equation 1.

$$EI\frac{d^2y}{dx^2} = M(x) \tag{1}$$

Where E is the modulus of elasticity ($70\,\mathrm{GPa}$ for aluminum, $1\,\mathrm{Pa} = 1\,\mathrm{N\,m^{-2}}$) and I is the moment of inertia of the cross section, $I = \frac{\pi}{4}(R^4 - r^4)$. Note that although E and I are constants for the beam, the moment varies along the length of the bar and depends on the location of the applied force as shown in Equation 2.

$$M(x) = \begin{cases} \dfrac{-P(L-d)x}{L} & \text{for } 0 \leq x \leq d \\ \dfrac{-Pd(L-x)}{L} & \text{for } d < x \leq L \end{cases} \tag{2}$$

Additionally, due to the immovable supports on either end of the beam, we will enforce the two boundary conditions shown in Equation 3.

$$y|_{x=0} = 0 \quad \text{and} \quad y|_{x=L} = 0 \tag{3}$$

(a) Build the A matrix for the system. Begin by creating 20 evenly-spaced nodes in the range $[0,\ L]$ to discretize the bar. Next, discretize Equation 1 for all the *interior* points using the central, second-derivative stencil developed in discussion. Remember to enforce the boundary conditions at both endpoints, $x = 0$ and $x = L$.

(b) Form the right-hand side vector. The behavior at interior points is a function of the bending moment, modulus of elasticity, moment of inertia, and node spacing, while nodes corresponding to endpoints are governed by the boundary conditions. Note that our sign convention dictates that a downward-pointing force be denoted with a negative sign.

(c) Solve your system of equations and plot the displacement with connected markers (e.g., `plot(..., 'o-')`) as a function of x-position along the beam. What is the maximum displacement of the beam (in meters) and where does it occur? Calculate and report the error of your maximum displacement calculation when compared with the theoretical solution below:

$$y_{\text{max}} = \frac{Pc(L^2 - c^2)^{1.5}}{9\sqrt{3}EIL} \qquad \text{where} \quad c = \min(d, L-d) \tag{4}$$

What happens to the error in the maximum displacement when you increase the number of discretization points to 100? Experiment with changing the location of the applied force, $0 \leq d \leq L$, using your 100-node simulation. Use these experiments to identify the range of x-positions that *always* contains the location of the maximum displacement; that is, identify both x_{min} and x_{max} such that any valid value of d generates a maximum displacement located between these two points on the beam (knowing *where* the location of maximum deflection will occur *regardless* of where the bending force is applied can be a useful design factor!).

Note: You are absolutely encouraged to experiment with different setups to generate results for your discussion, but in the final version you upload to CCLE, your code should use the original dimensions, 20 nodes, and a $2000\,\text{N}$ force located $0.75\,\text{m}$ away from the left-hand edge.

Using the naming convention presented in the syllabus, submit **two** separate files to the CCLE course website: (1) a .pdf of your written report and (2) a .zip file containing all of the MATLAB files written for the assignment. Remember to use good coding practices by keeping your code organized, choosing suitable variable names, and commenting where applicable. Any of your MATLAB .m files should contain a few comment lines at the top to provide the name of the script, a brief description of the function of the script, and your name and UID.