Benjamin Asdell
CEE/MAE M20
UID: 005114992 **Homework 7** October 15, 2020

1 Euler-Bernoulli Beam Bending

1. Introduction

   The objective of this problem is to simulate the stress experienced by a simply supported beam under a point load. The program calculates the vertical displacement of the beam at certain discretized nodes along the beam and plots the displacement over the length of the beam. It then finds the location and magnitude of the maximum displacement due to the load, including a calculation of the error from the mathematical maximum.

2. Models and Methods

   According to the assignment, the deflection y in an Euler-Bernoulli beam can be calculated from the following equation:

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

In Eq. 1, E represents the elastic modulus of the material, in this case steel with a value of $E = 2 * 10^11$ Pa. The moment of inertia I of the beam is given by the following equation:

$$I = \frac{a^4 - b^4}{12} \tag{2}$$

where a represents the outer width of the beam and b represents the inner width. In this problem, $a = 0.075$ m and $b = 0.075$ m. The $M(x)$ is called the bending moment; it is a calculation of the stress of the beam relative to the beam's length L, the point force applied P, and the location of the point force d compared to the location being examined x.

$$M(x) = \begin{cases} -\frac{Px(L-d)}{L} & 0 \leq x \leq d \\ -\frac{Pd(L-x)}{L} & d < x \leq L \end{cases} \tag{3}$$

There are boundary conditions specified by the assignment to simplify the nature of the problem and the smoothness of the calculation. These conditions are that the displacements at the endpoints of the beam, $y_{x=0}$ and $y_{x=L}$, are both 0. This means that regardless of where the force is applied on the beam, the endpoints are secured in a fashion that inhibits any bending. To find the displacements along the beam, the beam is discretized into a finite amount of nodes representing a point along the beam. A matrix equation $Ay = b$ is then created to relate the displacements at each of these nodes to the second order central difference equation in Eq. 4.

$$y_{k-1} - 2y_k + y_{k+1} = \Delta x^2 \frac{M(x)}{EI} \tag{4}$$

Since the central difference expression on the left side of Eq. 4 utilizes the previous and next values of the displacment at each node, an A matrix can be built using the coefficients of the left hand side that will be multiplied by the y vector in MATLAB. Since we also have the boundary conditions for the first and last entries in the b vector, matrix multiplication and MATLAB's built-in matrix algebra provides the y vector easily. From this y vector, the maximum displacement can be extracted in addition to its location along the beam. The assignment also asks for the percent error relative to the mathematical calculation of the maximum displacement, which are given in the following equations:

$$y_{theo} = \frac{Pc(L^2 - c^2)^{1.5}}{9\sqrt{3}EIL}, c = min(d, L - d) \tag{5}$$

$$error = |\frac{y_{theo} - y_{max}}{y_{theo}}| * 100 \tag{6}$$

To examine the changes in bending along the beam and visually see the displacement caused by the bending stress, the displacement y vector is plotted as a function of the discretized x nodes along the beam.

3. Calculations and Results

With 20 nodes and a distance $d = 0.15$ m from the left hand edge of the beam, the following plot is generated:

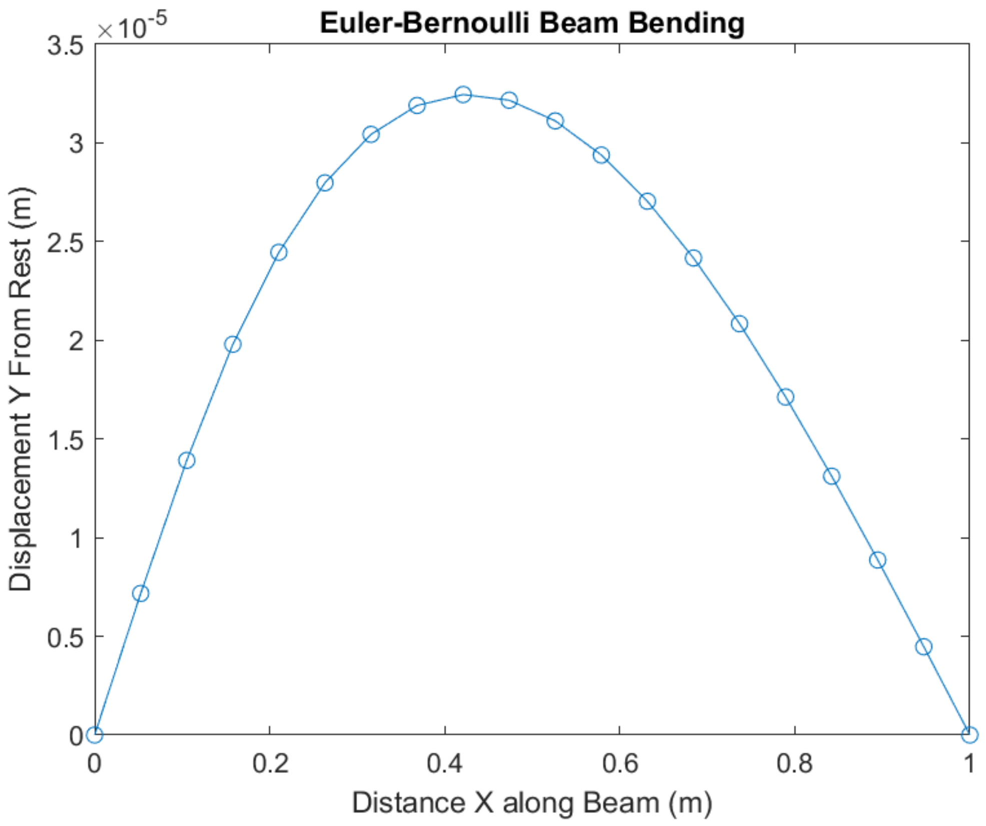


Figure 1: Visibly, the maximum displacement occurs somewhere near 0.4 m from the left edge of the beam. Note that this maximum displacement and in general the higher displacement values are not centered around the location of the point force.

The above plot with the original dimensions given by the assignment shows the displacement grows to a maximum displacement and returns to a zero value at the endpoints. To clarify, this graph only shows the magnitude of the displacement, as the point force is oriented downward on the top face of the beam and therefore the bending of the beam would occur in the negative vertical direction. The maximum displacement as determined from the plot and data is $3.245 * 10^{-5}$ m, and occurs at the 9th node, or $x = 0.4211$. The error in this determination compared to the theoretical maximum displacement is 0.2492%. This is a relatively insignificant amount of error, and therefore this plot can be concluded to be a relatively accurate model of the bending stress. The next plot discretizes the beam even further into 100 nodes.

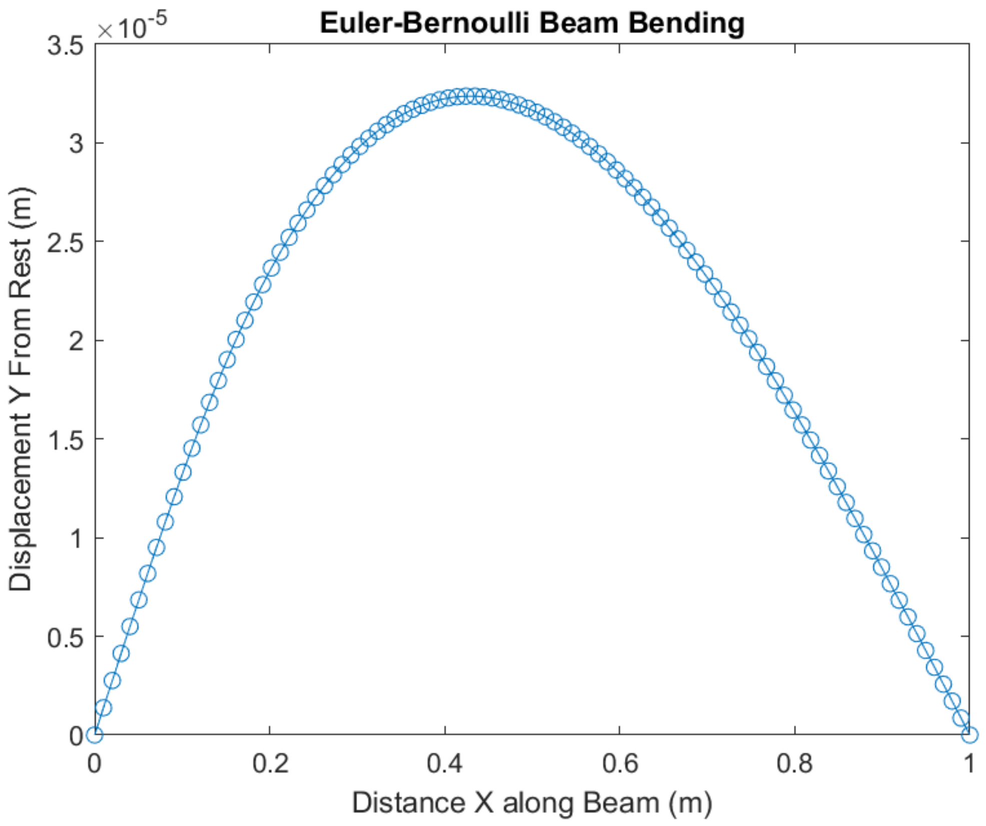


Figure 2: By visual comparison, this plot has the same shape as the first plot. It is geometrically smoother, with more nodes meaning there is less distance between each displacement calculation.

While it may be more difficult to visually determine the maximum displacement and its location from this specific plot, calculating those values from this data set of 100 nodes will lead to a more accurate answer. The maximum displacement from this plot is $3.2369 * 10^{-5}$ m and occurs at the 43rd node, or $x = 0.4242$ m. The error in this particular calculation is only 0.0014%. This is not only drastically less error than the original plot, but so insignificant that this model is very nearly an exact representation of the bending of the beam. The final plot below is an example of less discretized nodes, in this case only 5 nodes.

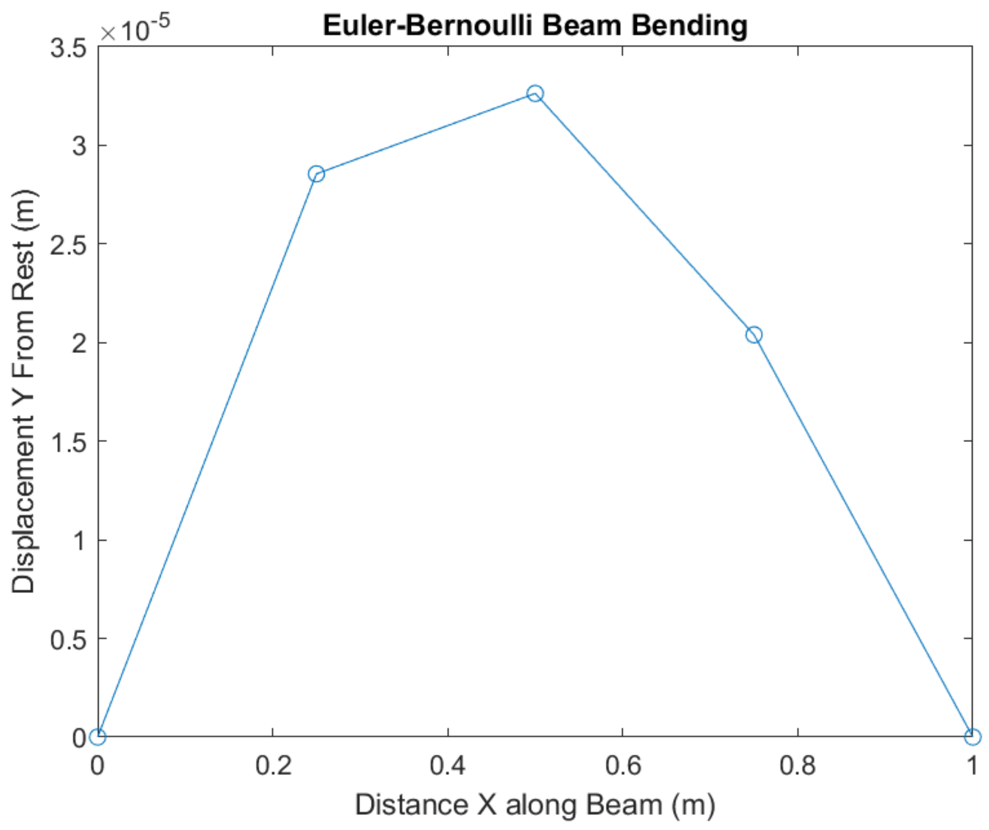


Figure 3: It is clear that this plot is not an accurate representation of the bending of the beam because of the sharp edges of the graph. It does maintain somewhat the same shape as the first two plots, but the roughness of the plot is evident and has an impact on the accuracy of the maximum displacment calculation.

In this plot, the maximum displacement is calculated to be $3.2632 * 10^{-5}$ and occurs at $x = 0.5$ m. The error in this calculation is 0.811%, which is significantly higher than both of the previous errors, but still at an acceptably low value. This model can still be reasonably used to find the maximum displacement; however, the x location at which this displacement occurs is very inaccurate considering the values found in the previous two plots.

Nodes	Error(%)
3	0.811
5	0.811
20	0.2492
100	0.0014

4. Discussion

This problem is a demonstration that MATLAB is very useful in breaking down physical experiments and simulating the results of physical interactions. There are a few interesting points to make about the functionality of the program and its utility in handling physical problems like this. First, as one can see from the table above, the error between this method and the theoretical maximum displacement is relatively low at all numbers of discretized nodes. While the error does decrease significantly as the number of nodes increase, all node values are less than 1%. Secondly, building off the first point, this program completes its task in a very short amount of time. Timing the program with an increasing number of nodes only produced an increase in time on a scale of hundredths of seconds, which may be significant in computer time but not that significant at all in programmer time. In fact, it took several orders of magnitude higher discretization in order for the program to take longer than one second. Using the program with 100 nodes on the beam, it is possible to determine a range of x values between which the maximum y displacement will occur regardless of the location of the point force d. This range is $x_{min} = 0.4242$ to $x_max0.5758$ m. Of course, these values are determined from the 100 node simulation and increasing the number of nodes will most likely increase the accuracy of these values. However, this range provides insight into the location of the most bending. In practical applications, this range allows architects or engineers to place supports within this range of x along the beam in order to strengthen the structural integrity of any object experiencing a point force.

2 The Game of Life

1. Introduction

   The goal of this problem is to simulate the infamous John Conway's Game of Life over many generations. The program first generates a grid of specified size with a probability that any given cell on the grid has a 20% chance of being alive. Then, the program simulates each generation according to the rules of the Game of Life and generates a video of the grid over time in addition to a plot of the living cell population over time.

2. Models and Methods

   Conway's Game of Life can be represented on 2D array within MATLAB. A dead cell is represented by a 0 value at that array index and a living cell is therefore represented by a 1. The rules of the game of life are as follows:

   (a) A living cell with either 2 or 3 living neighbors is also alive in the next generation.

   (b) A living cell with less than 2 (isolation) or greater than 3 living neighbors (overpopulation) dies during the next generation.

   (c) A dead cell with exactly 3 living neighbors becomes alive during the next generation.

   Using these rules, the program generates a new grid based on the previous one for each generation. A video showing a smooth transition between each generation from the first to the last is then generated using MATLAB's VideoWriter. Then, a plot of the proportion of living cells out of the total cells in each generation over time is generated.
   According to the assignment, a variation of the Game of Life is addressed in this program that induces a 1% chance that any given cell that should be alive in the next generationn dies prematurely. The same video and plot are generated for this variation. The code submitted for this assignment only includes the functionality for this variation.

3. Calculations and Results

   Without premature death introduced into the simulation, the plot for the original Game of Life is generated:

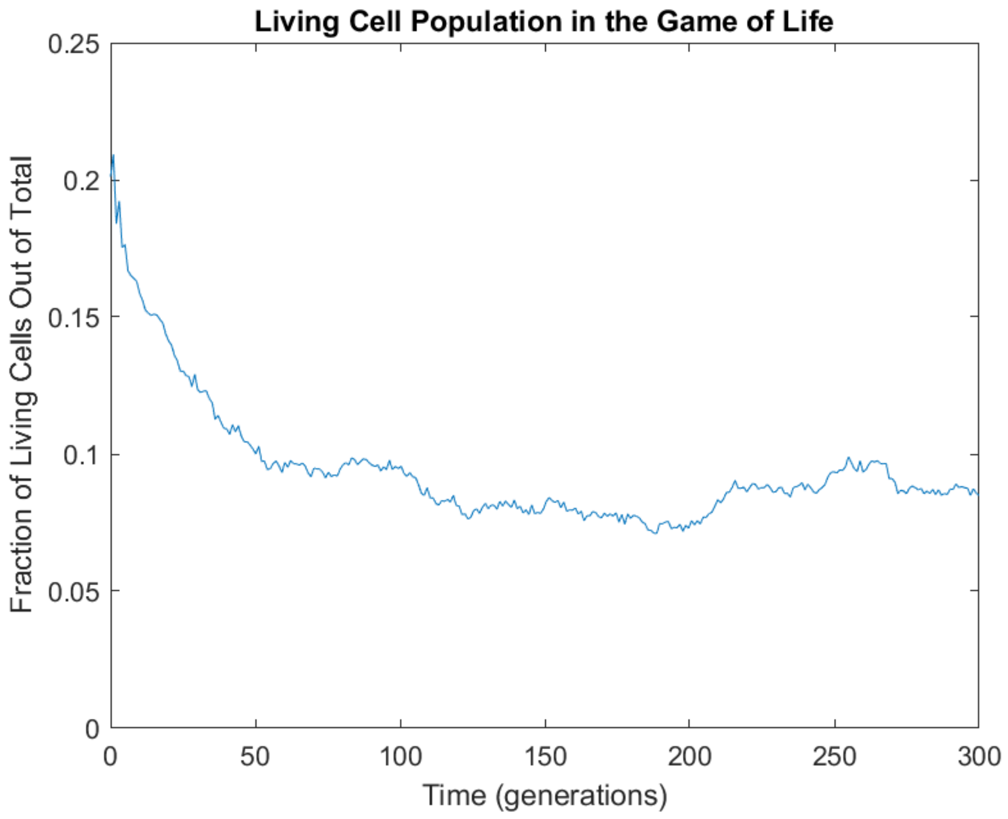
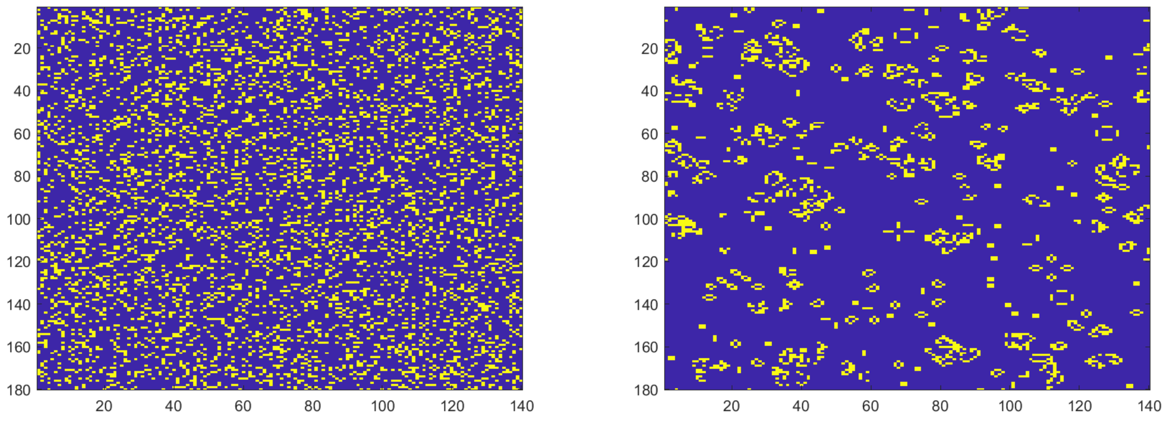


Figure 4: The living population decreases over time, but seems to stabilize around the 150th generation. Despite this, there is significant deviation even after this stabilization point.



(a) Initial (b) Final

Figure 5: The initial (left) and final (right) distributions of living cells. Yellow cells represents living cells and purple cells represent dead cells.

The next of figures show the variation of Conway's Game of Life. Remember that in this variation, cells that should be alive will die with a 1% chance. The plots are as follows:

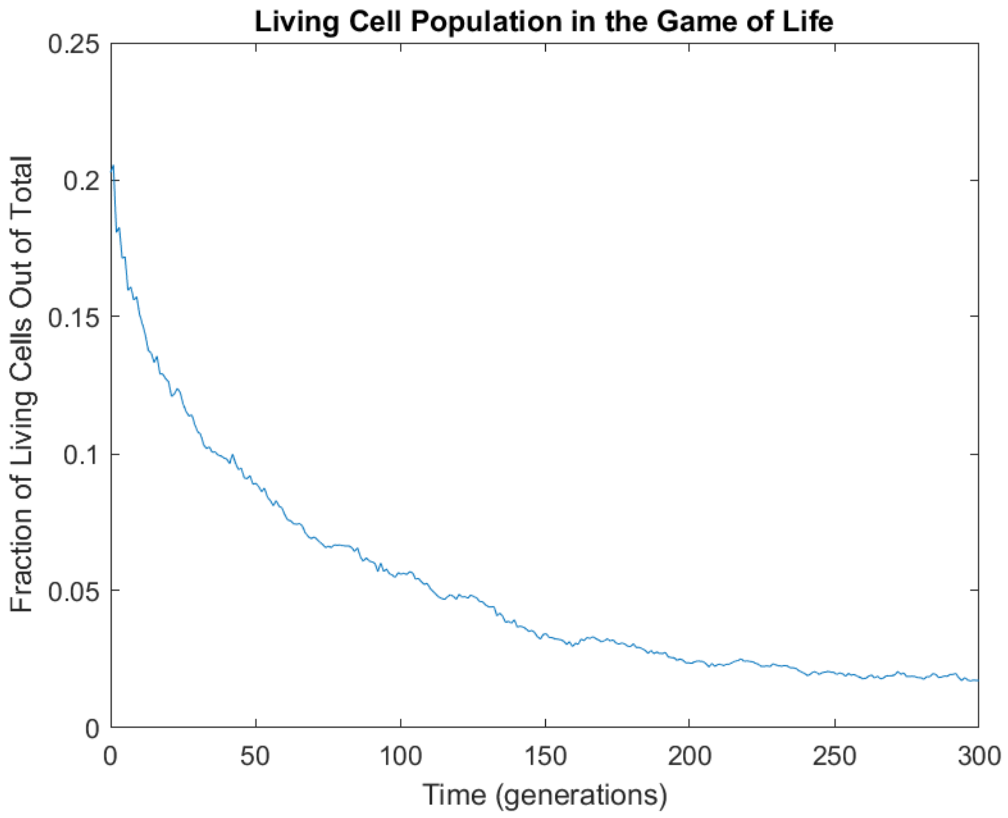
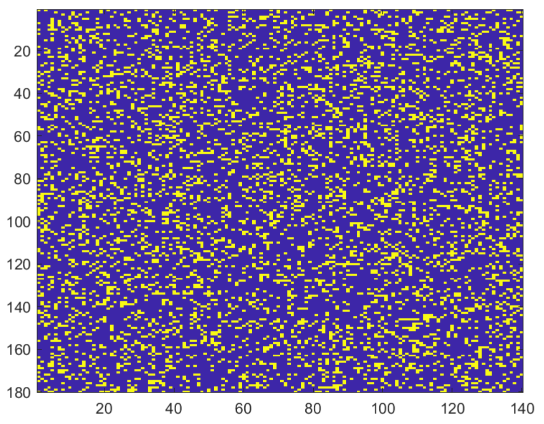
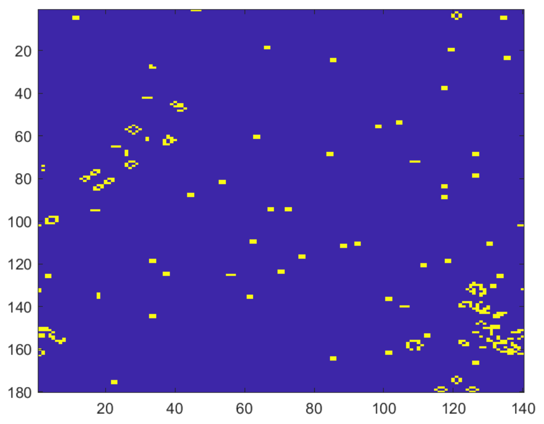


Figure 6: Compared to the previous plot, the living cell population stabilizes around 250 generations. The effect of premature death on the cells is very apparent, as the final living cell population after 300 generations is significantly lower than the original simulation. The population does not seem to stabilize until at least the 250th generation.



(a) Initial



(b) Final

Figure 7: The initial (left) and final (right) distributions of living cells. Yellow cells represents living cells and purple cells represent dead cells. This particular set of plots represents the variation with 1% of dying prematurely.

4. Discussion

This program is an excellent example of how MATLAB deals with compelex mathematical problems. The Game of Life is a simulation of simplistic cell behavior. MATLAB allows the user to view each generation of the Game of Life over time. As expected, the premature death variation causes a loss of total living cell population over time. Without premature death, the living cell population appears to stabilize around the same value as mentioned above. Visibly, the final grid for the premature death simulation is far more sparse in terms of living cells compared to the original simulation. This program was extremely interesting to code and view the result, as the videos generated really seemed to come alive. It is very apparent that introducing premature death into the system drastically reduced the final living cell population as the pockets of cells that stayed stagnant in the original simulation were killed off by random chance over the course of the 300 generations.