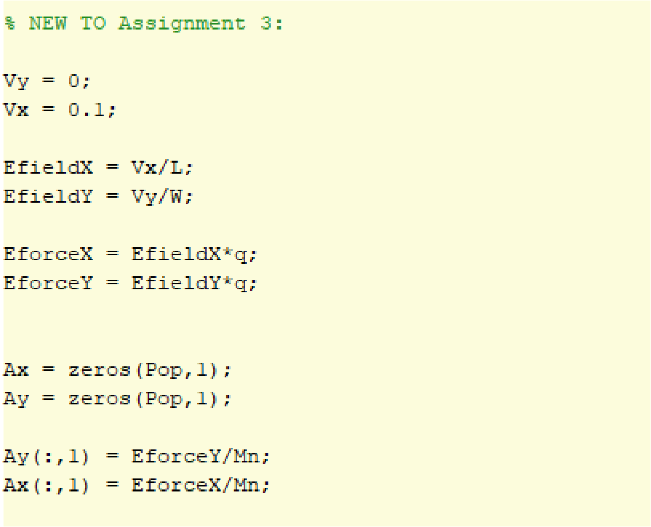
Ryan Lindsay 101038101 3/21/2021

1 Part 1

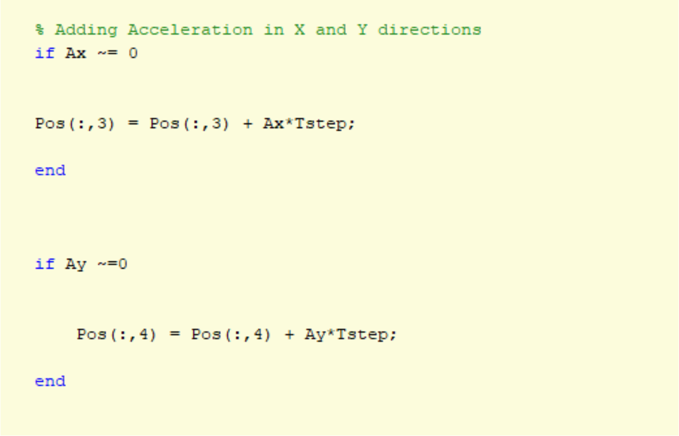
In part 1, The Monte Carlo simulation from assignment 1 was updated to include a constant voltage that is applied over the entire semiconductor in order to accelerate the electrons. In my previous assignment, I did not have the capability to simulate large numbers of electrons, because I was plotting every single one. This has been fixed, allowing me to only plot a small number of the electrons that are simulated.

The first step of the assignment is to use the provided voltage of 0.1V to develop a constant electric field across the semiconductor, accelerating the electrons in the x direction. The capability to accelerate the in y direction was also added. The following two blocks of code achieve this.



```
% NEW TO Assignment 3:

Vy = 0;
Vx = 0.1;

EfieldX = Vx/L;
EfieldY = Vy/W;

EforceX = EfieldX*q;
EforceY = EfieldY*q;


Ax = zeros(Pop,1);
Ay = zeros(Pop,1);

Ay(:,1) = EforceY/Mn;
Ax(:,1) = EforceX/Mn;
```

Using this block of code, the following plot was created.



```
% Adding Acceleration in X and Y directions
if Ax ~= 0


Pos(:,3) = Pos(:,3) + Ax*Tstep;

end



if Ay ~=0


    Pos(:,4) = Pos(:,4) + Ay*Tstep;

end
```

From these calculations the Electric field strength was found to be $5\exp(5)$ V/m. The force was found to be $8\exp(-14)$ Nm, and the acceleration was found to be $3.38\exp(17)$ m/s. Using this applied field, the 2D trajectories plot could be redone. This can be seen in the following figure. From this plot, you can see that each electron curls towards the X direction.

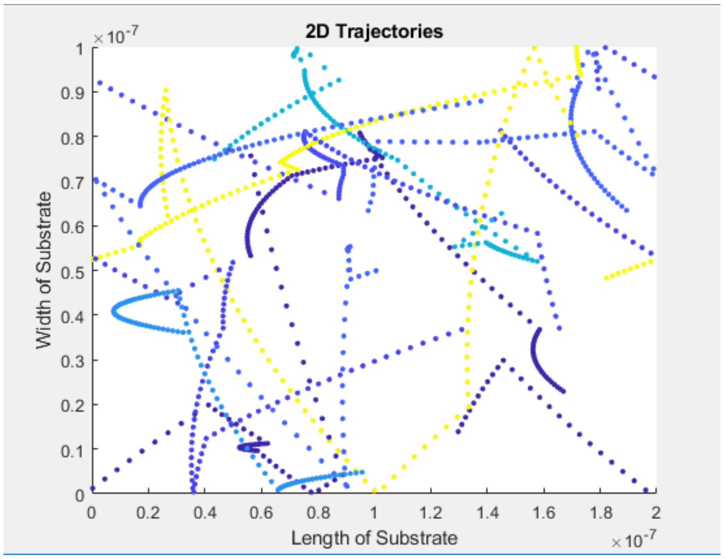
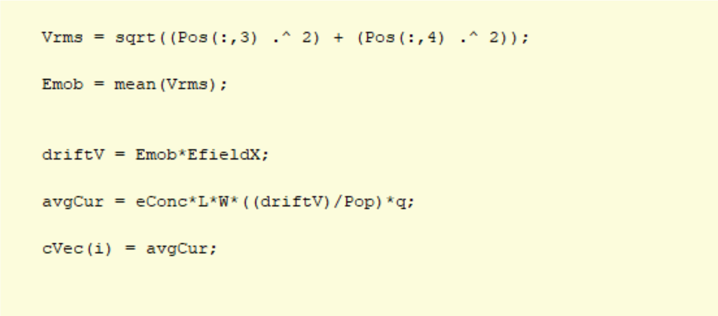


Figure 1: 2D Trajectories Plot

The next step was to find the average current in the substrate over time. The following block of code was used to do this. The proceeding figure is the average current of the time of the simulation. As time increases the current seems to plateau at around 10 pA.



```
Vrms = sqrt((Pos(:,3) .^ 2) + (Pos(:,4) .^ 2));

Emob = mean(Vrms);


driftV = Emob*EfieldX;

avgCur = eConc*L*W*((driftV)/Pop)*q;

cVec(i) = avgCur;
```

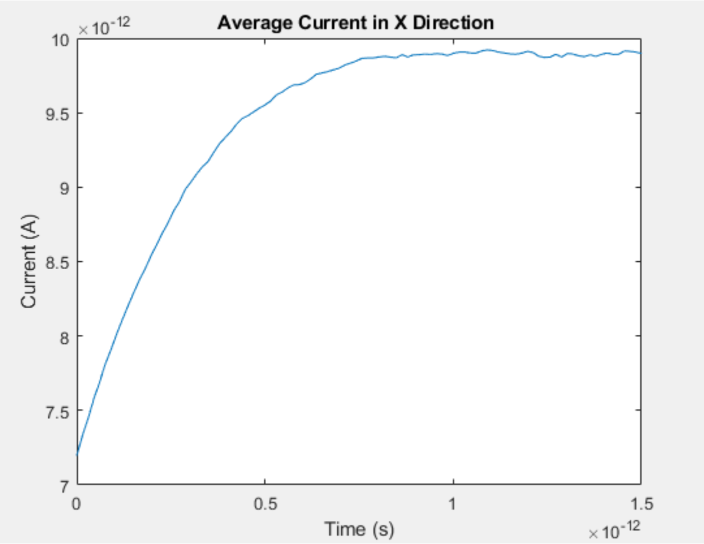


Figure 2: Current Plot

The last step was to simply plot the temperature and density plots. This code has not changed since assignment 1 so it will not be shown. These plots can be seen in the figures below respectively. While intuitively it would make sense to see an increased density and temperature on the right side of the substrate, this does not occur. This is because any electron that hits the far right boundary will automatically reappear on the left side. Therefore, there should still be a uniform distribution of electrons and temperature with the electric field.

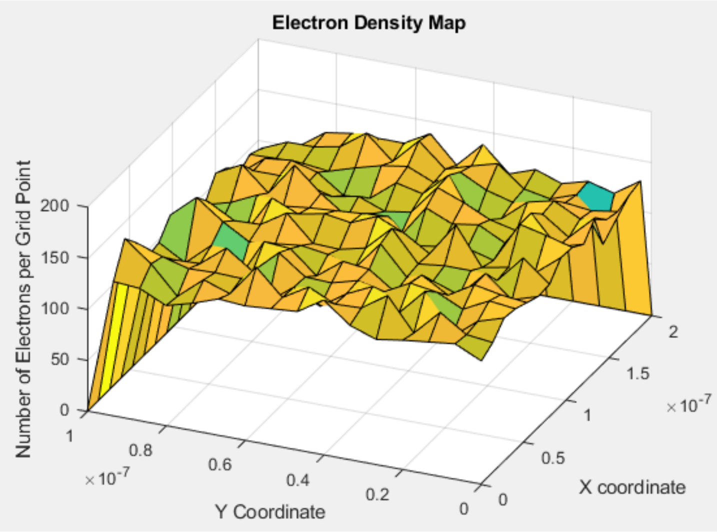


Figure 3: Electron Density Map

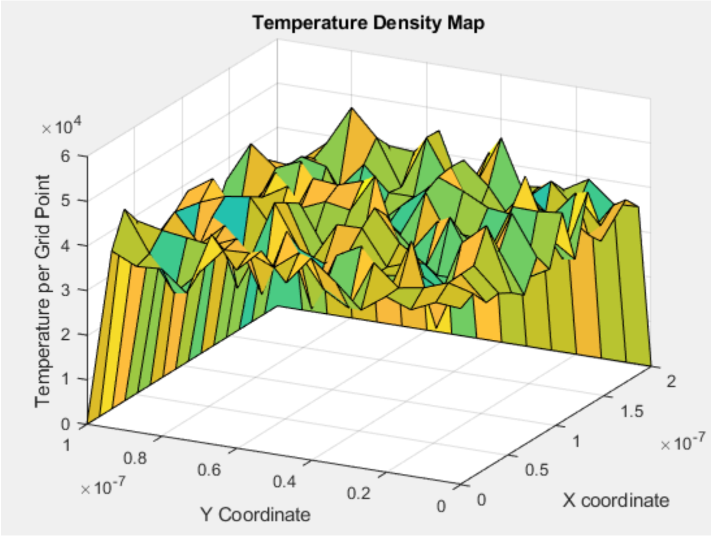
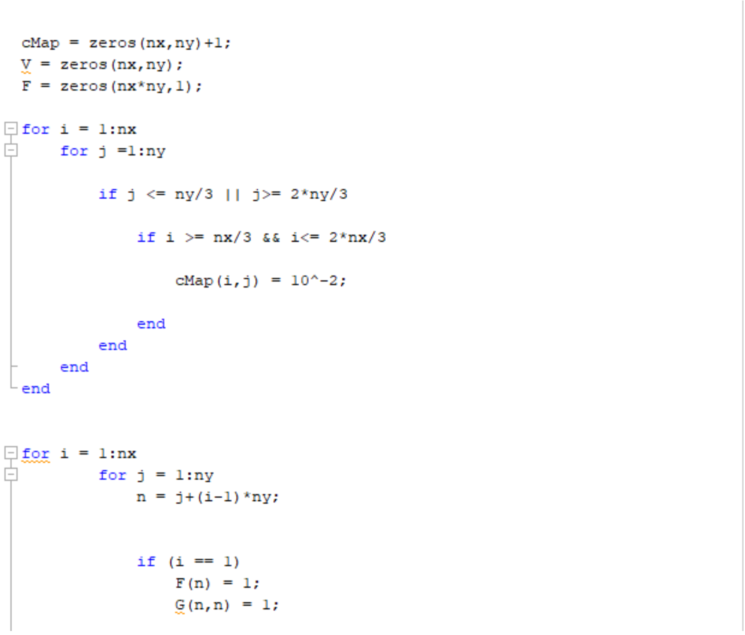


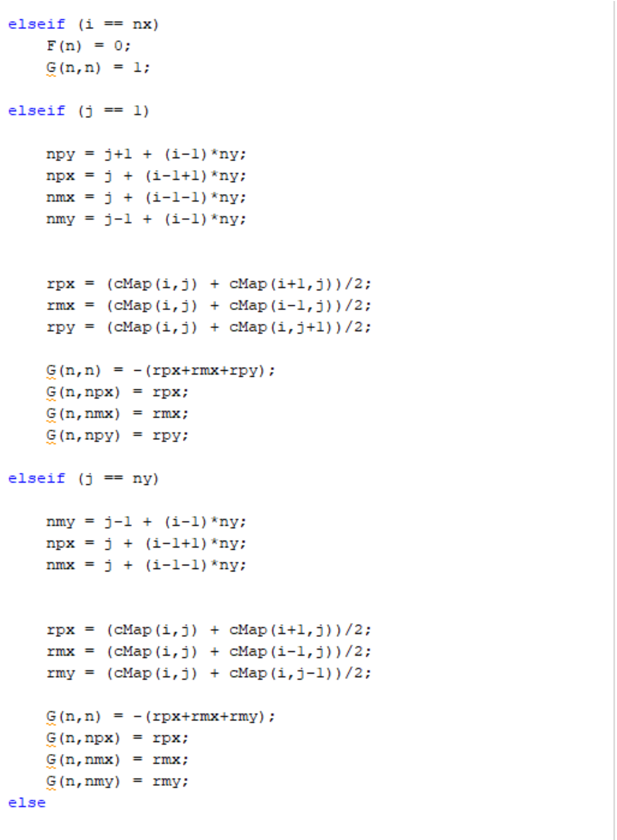
Figure 4: Temperature Density Map

2 Part 2

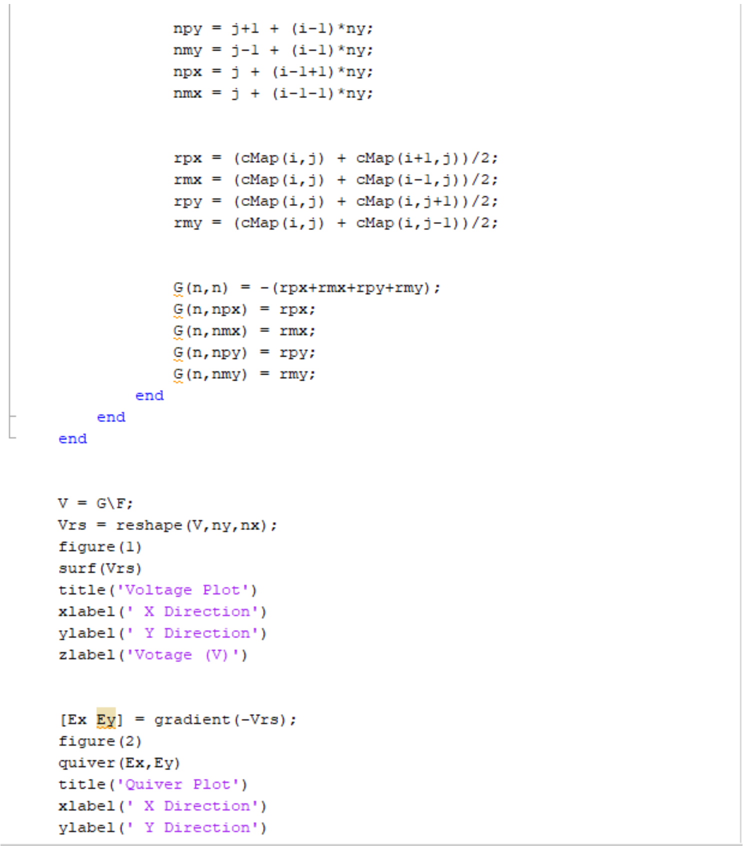
In part 2, we are using the finite difference method from assignment 2 to generate a field that we can apply to our Monte Carlo simulation. The following block of code shows the G matrix integrated with the conductivity map.



```
cMap = zeros(nx,ny)+1;
V = zeros(nx,ny);
F = zeros(nx*ny,1);

for i = 1:nx
    for j =1:ny

        if j <= ny/3 || j>= 2*ny/3

            if i >= nx/3 && i<= 2*nx/3

                cMap(i,j) = 10^-2;

            end
        end
    end
end


for i = 1:nx
        for j = 1:ny
            n = j+(i-1)*ny;


            if (i == 1)
                F(n) = 1;
                G(n,n) = 1;
```



```
            elseif (i == nx)
                F(n) = 0;
                G(n,n) = 1;

            elseif (j == 1)

                npy = j+1 + (i-1)*ny;
                npx = j + (i-1+1)*ny;
                nmx = j + (i-1-1)*ny;
                nmy = j-1 + (i-1)*ny;


                rpx = (cMap(i,j) + cMap(i+1,j))/2;
                rmx = (cMap(i,j) + cMap(i-1,j))/2;
                rpy = (cMap(i,j) + cMap(i,j+1))/2;

                G(n,n) = -(rpx+rmx+rpy);
                G(n,npx) = rpx;
                G(n,nmx) = rmx;
                G(n,npy) = rpy;

            elseif (j == ny)

                nmy = j-1 + (i-1)*ny;
                npx = j + (i-1+1)*ny;
                nmx = j + (i-1-1)*ny;


                rpx = (cMap(i,j) + cMap(i+1,j))/2;
                rmx = (cMap(i,j) + cMap(i-1,j))/2;
                rmy = (cMap(i,j) + cMap(i,j-1))/2;

                G(n,n) = -(rpx+rmx+rmy);
                G(n,npx) = rpx;
                G(n,nmx) = rmx;
                G(n,nmy) = rmy;
            else
```



```
            npy = j+1 + (i-1)*ny;
            nmy = j-1 + (i-1)*ny;
            npx = j + (i-1+1)*ny;
            nmx = j + (i-1-1)*ny;


            rpx = (cMap(i,j) + cMap(i+1,j))/2;
            rmx = (cMap(i,j) + cMap(i-1,j))/2;
            rpy = (cMap(i,j) + cMap(i,j+1))/2;
            rmy = (cMap(i,j) + cMap(i,j-1))/2;


            G(n,n) = -(rpx+rmx+rpy+rmy);
            G(n,npx) = rpx;
            G(n,nmx) = rmx;
            G(n,npy) = rpy;
            G(n,nmy) = rmy;
        end
    end
end


V = G\F;
Vrs = reshape(V,ny,nx);
figure(1)
surf(Vrs)
title('Voltage Plot')
xlabel(' X Direction')
ylabel(' Y Direction')
zlabel('Votage (V)')


[Ex Ey] = gradient(-Vrs);
figure(2)
quiver(Ex,Ey)
title('Quiver Plot')
xlabel(' X Direction')
ylabel(' Y Direction')
```

This code produced the following voltage plot and quiver plot.

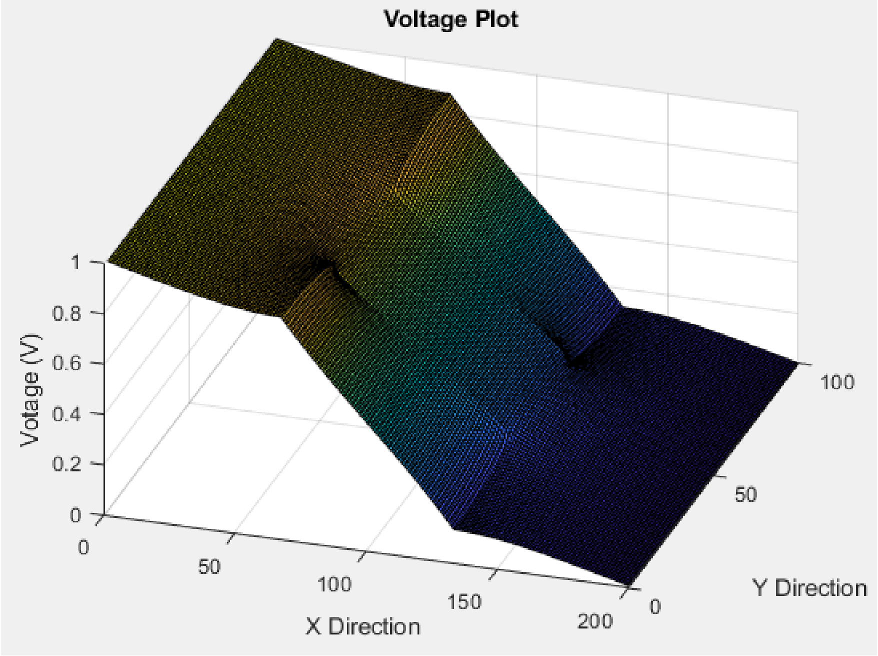


Figure 5: Voltage Plot

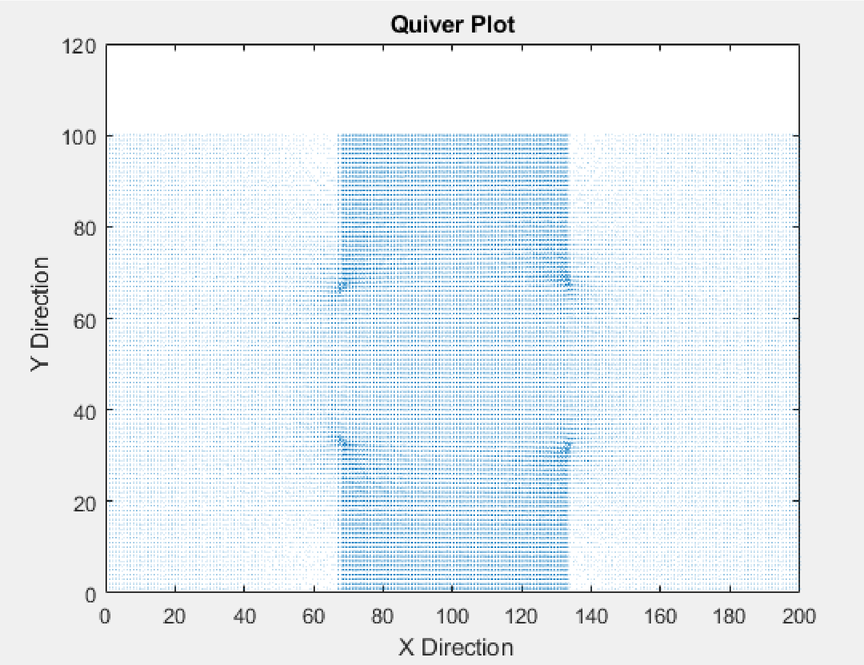
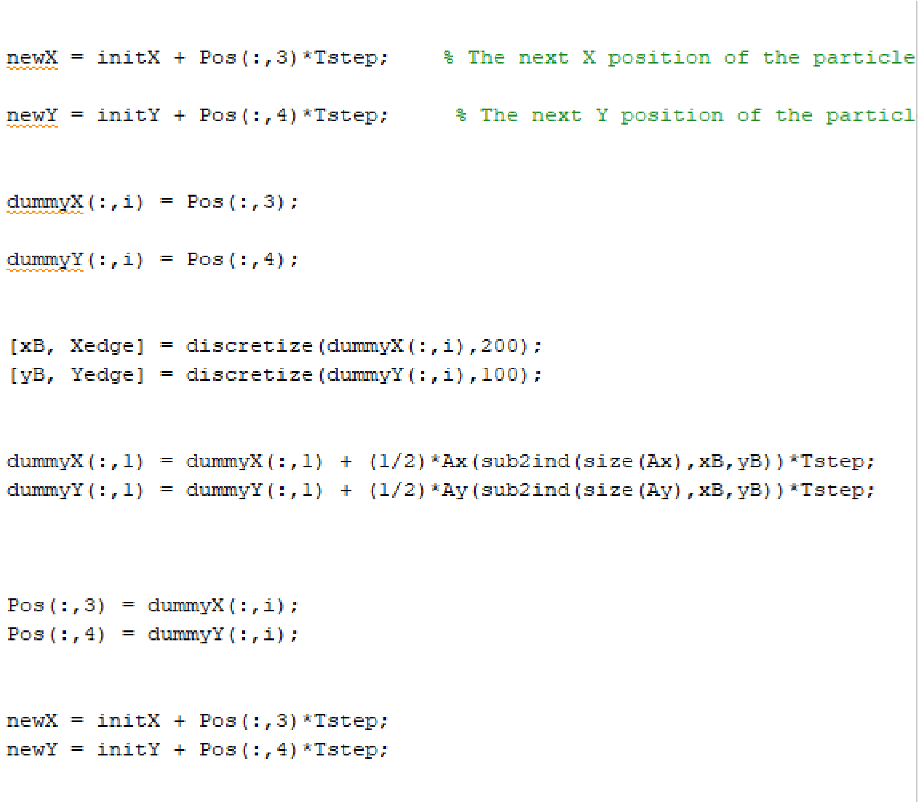


Figure 6: Quiver Plot

The next step was to applied corresponding E field found from the G matrix onto the Monte Carlo simulation from before. The following block of code attempted to do this.



```
newX = initX + Pos(:,3)*Tstep;     % The next X position of the particle

newY = initY + Pos(:,4)*Tstep;      % The next Y position of the particl


dummyX(:,i) = Pos(:,3);

dummyY(:,i) = Pos(:,4);


[xB, Xedge] = discretize(dummyX(:,i),200);
[yB, Yedge] = discretize(dummyY(:,i),100);


dummyX(:,1) = dummyX(:,1) + (1/2)*Ax(sub2ind(size(Ax),xB,yB))*Tstep;
dummyY(:,1) = dummyY(:,1) + (1/2)*Ay(sub2ind(size(Ay),xB,yB))*Tstep;



Pos(:,3) = dummyX(:,i);
Pos(:,4) = dummyY(:,i);


newX = initX + Pos(:,3)*Tstep;
newY = initY + Pos(:,4)*Tstep;
```

The 2D trajectory plot could then be plotted. I was expecting to more of a behaviour from the field, but this did not seem to occur. The plot of 1000 electrons over 1000 steps can be seen in the figure below.

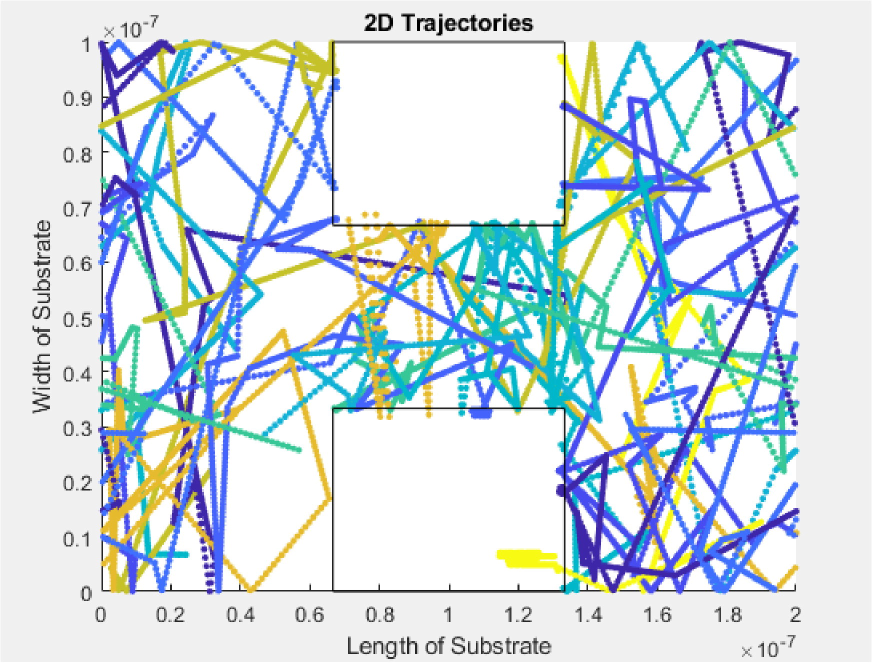


Figure 7: 2D Trajectory Plot

3 Part 3

In part 3, we will investigate the density of electrons and the average current depending on the size of the bottleneck. A constant field of 0.8V was used for this simulation. The density plot of electrons can be seen in the figure below. The electrons appear to gather at the left edge of each of the boxes. This makes sense because any electrons that ends up on the right side of the boxes will eventually end up back on the left side due to the nature of the simulation.

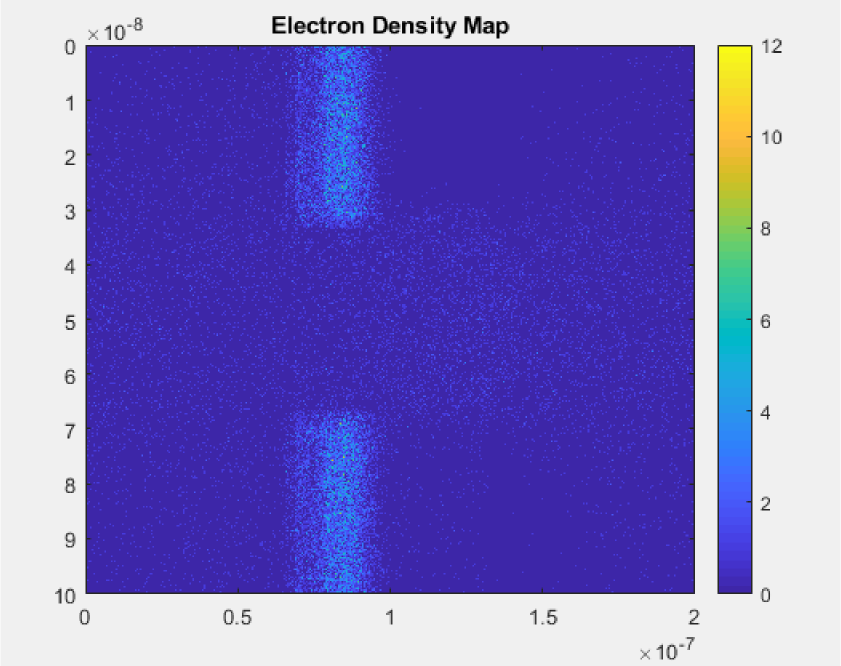


Figure 8: Density Plot

The last step was to measure the average current over different bottleneck sizes. I chose to only adjust the Y coordinate of the boxes. The average current for the standard bottleneck with 0.8V can be seen in the figure below. The standard bottleneck covers 1/3 of the substrate on each side.

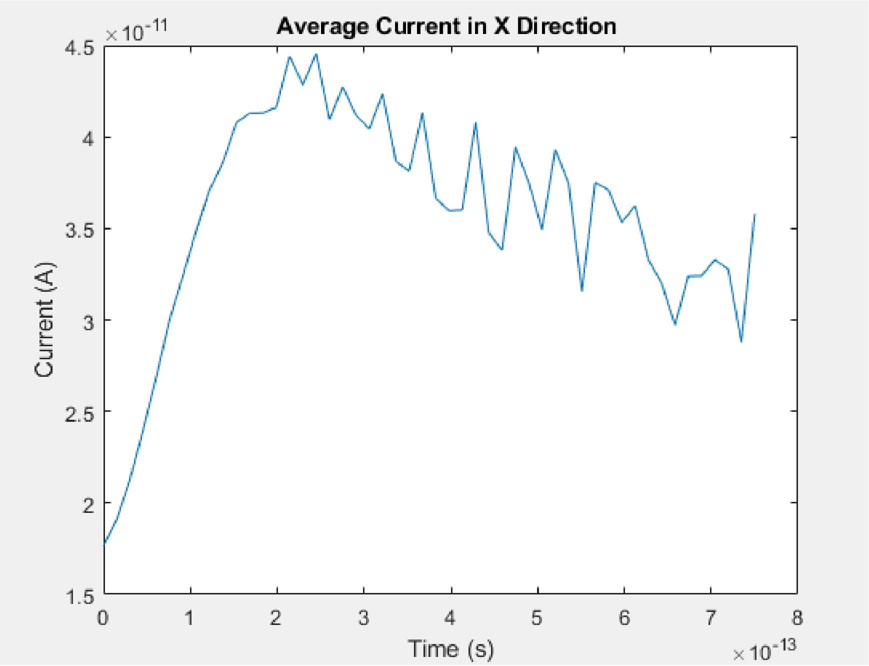


Figure 9: Standard Bottleneck Current Plot

For the next simulation, the box height on each side was reduced to 1/4 of the substrate height. The following current plot can be seen below.

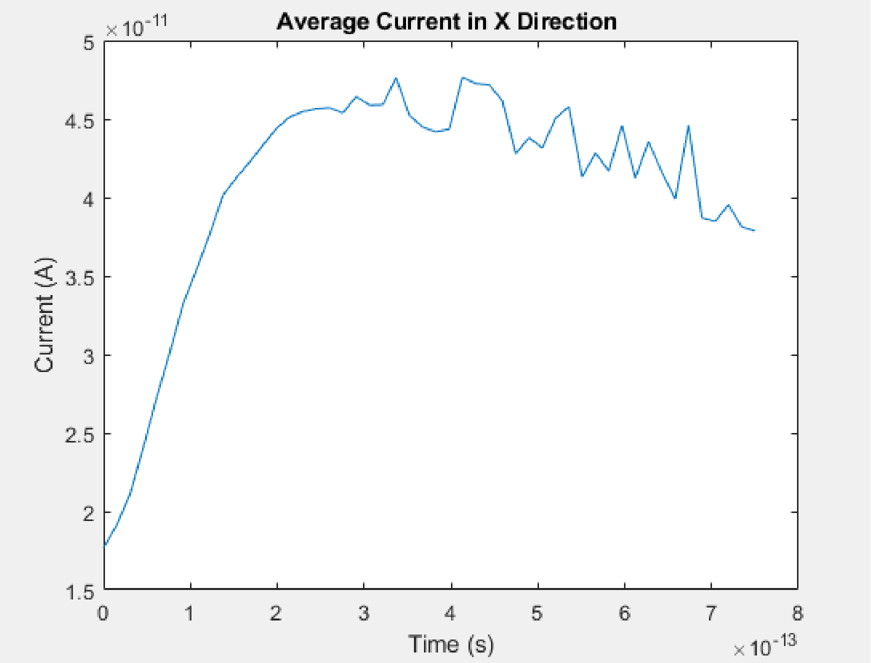


Figure 10: Small Bottleneck Current Plot

For the final simulation, the box height was increased to 3/8 of the substrate on each side. As expected, with a larger barrier, more electrons get trapped on the side and the current is reduced.

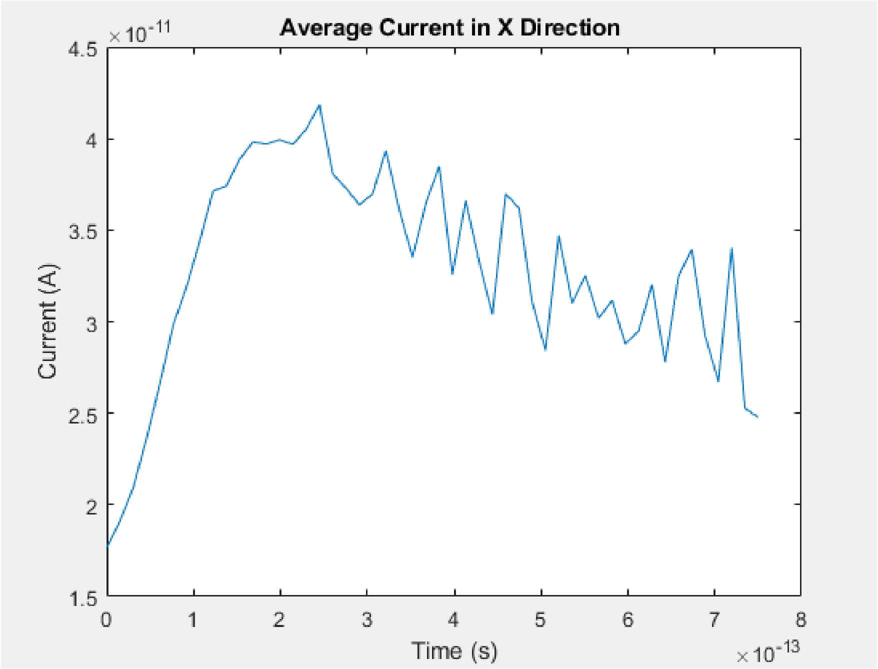


Figure 11: Large Bottleneck Current Plot

In terms of next steps, removing the infinite X direction and reflecting the electrons similar to what was done with the Y direction would be ideal. This would be a better judge of the behaviour of the electrons, as no real life material is infinite. The next step would be to create a larger substrate, allowing us to simulate more electrons at a time accurately.