Assignment 3

Table of Contents

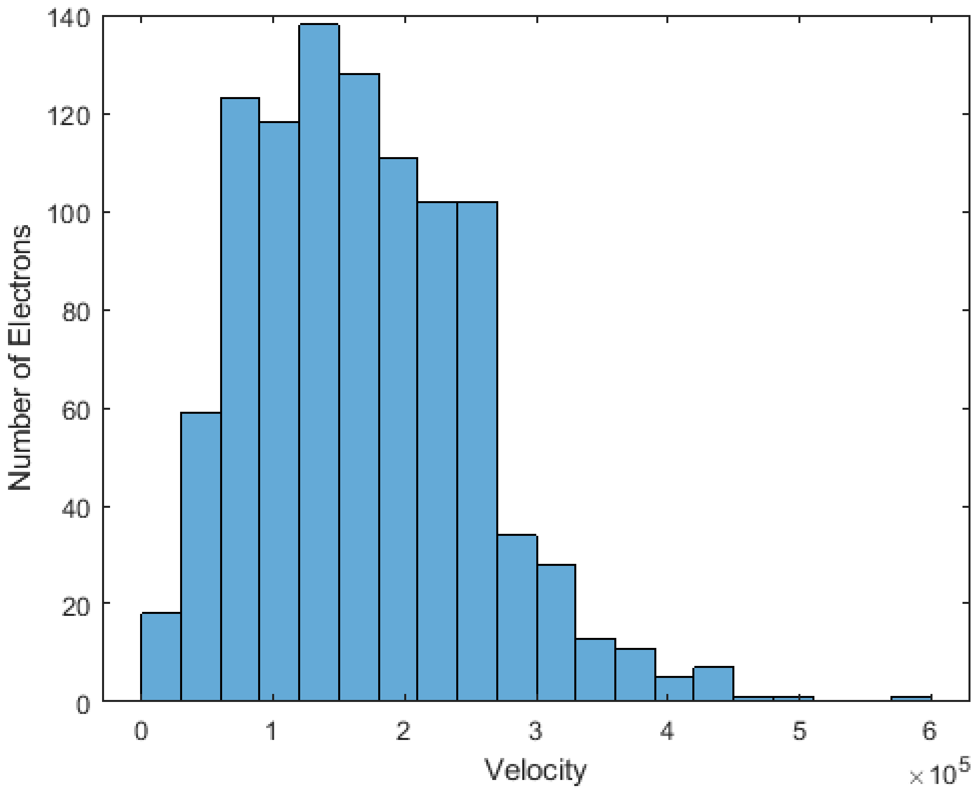
Part1

In this part of the assignment the code from assignmnet 1 that uses the Monte Carlo method to simulate random motion of electrons was modified to include the effects of having an electric feild exist within the material. In this case the simulation models a scenario where 0.1 V is apllied across the x=0 and x=L terminals of the device. Because the region we are simulating is rectangular, we can assume that the electric feilds is constant and exists only in the x-direction. Therefore, the first step in the simulation is to calculate the forece on each electron, as shown below.

```
%simulation constants
L = 200e-9; %Length of region (x axis)
W = 100e-9; %Width of region (y axis)
K = 1.3806e-23;
m = 0.26*9.1093e-31;
q = 1.60217662e-19;

%Calculate force on electrons
V = 0.1;
Fx = -q*V/L;
```

The magnitude of this force -8.0109 X 10^-14 N in the x direction. The next step in the code is to assign the position and velocities for each electron, as well as some other setup for the simulation, very much like what was done in assignmnet 1. This code is shown below.

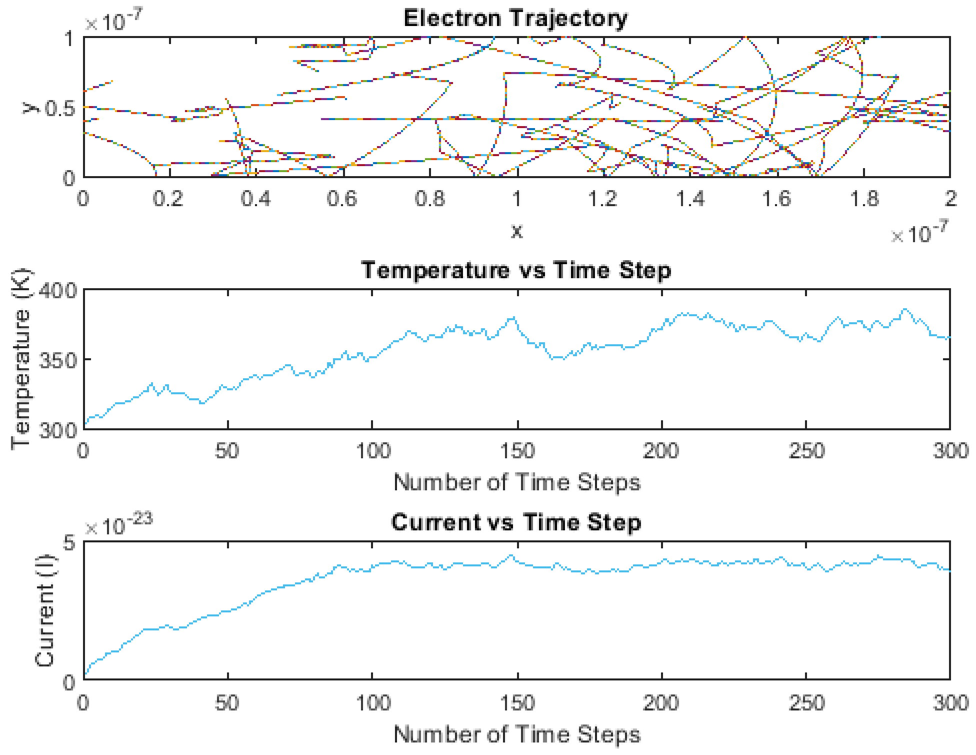
```
%Place electrons in Boundary
x = L*rand(1000,1);
y = W*rand(1000,1);

% Calculate how many electrons each "particle" in the simulation
 represents
electronConcentration = 10^15*1e4;

electronsPerParticle = electronConcentration*W*L/1000;

% Assign electron velocity based on Maxwell Boltzman Distribution
T = 300;
std = sqrt(K*T/m);
vth = sqrt(2*K*T/m);
Vx = normrnd(0,std,[1000,1]);
Vy = normrnd(0,std,[1000,1]);
V = sqrt(Vx.^2 + Vy.^2); %Now it is important to check that the
 velocicties were assigned correctly by confirming that the average
 velocity is close to the thermal velocity and by plotting a histogram
 of the velocity.
dt = 100e-9/vth/100;
figure(2)
histogram(V);
xlabel('Velocity')
ylabel('Number of Electrons')

%Initialize variables to plot
Tplot = zeros(300);
Ix = zeros(300);
```



Now the the electrons are setup, their trajectories can be calculated. The code that handles electron scattering and reflection off the simulation boundaries is exactly the same as in assignment 1. Since there is now a force acting on each electron, the code that updates the electron position had to be changed. The position and velocity in the x direction are now updated using the following kinematic equations. x2 = x1 + Vx1*dt + 0.5*Fx*dt^2/m and Vx2 = Vx1 +Fx*dt/m. Adittionaly, the current density is calculated by the product of the mean velocity, the electron density, and the electric charge, and the current is found by multiplyig the current density times the area of the region. zthe current is plotted as a function of timestep. The code for this section is shown below.

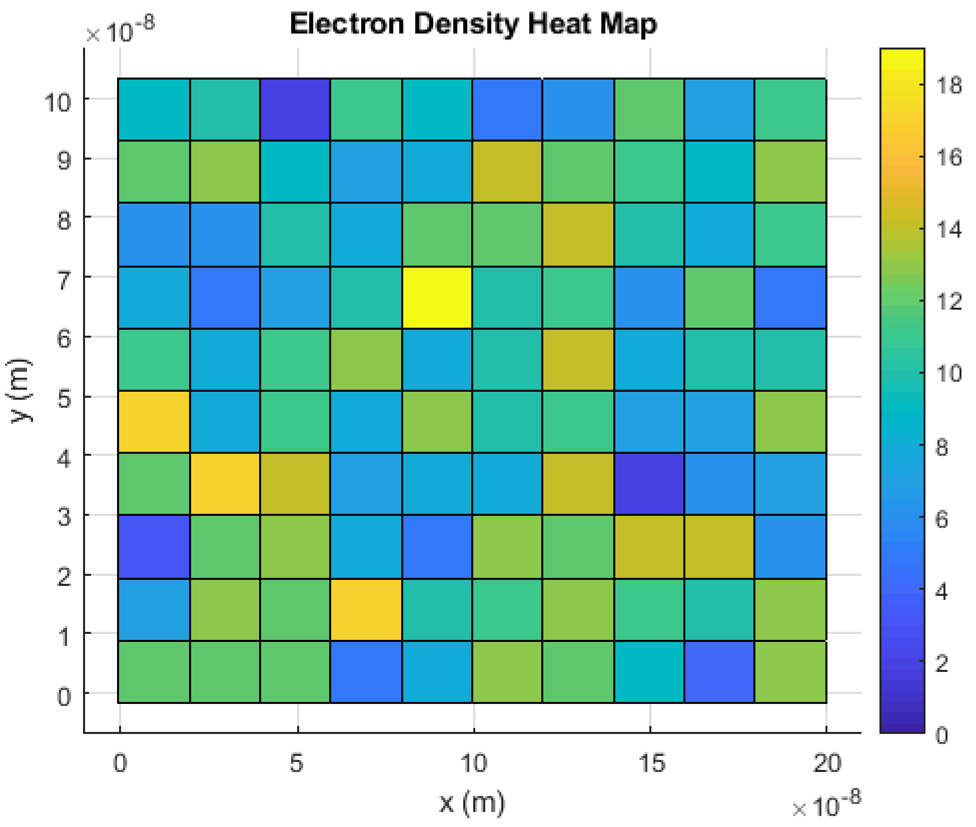
```
for i =1:300
    xold = x;
    yold = y;

    % Define region boundaries and rules for interacting with
 boundaries
```

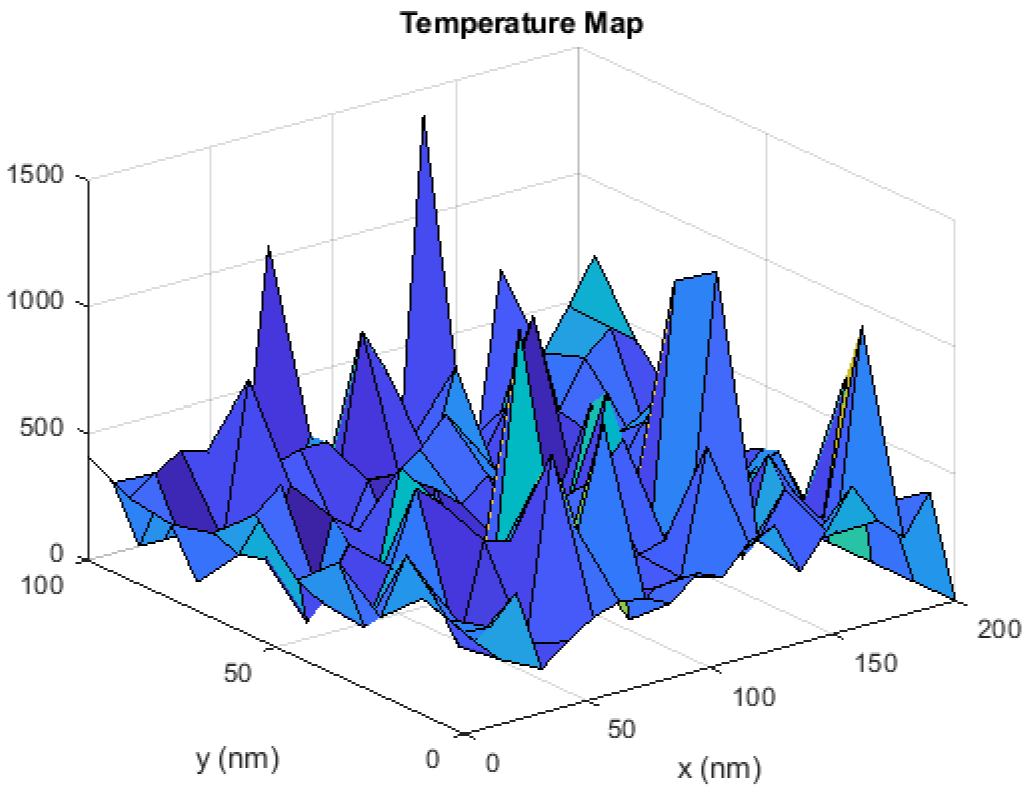
```
        xboundRight = x > L;
        xboundLeft = x < 0;
        ybound = (y > W) | (y <0);
        x(xboundRight) = x(xboundRight) - L;
        x(xboundLeft) = x(xboundLeft) + L;
        xold(xboundRight | xboundLeft) = x(xboundRight | xboundLeft);
        Vy(ybound) = -Vy(ybound);

        %Update Position
        x = x + Vx*dt + 0.5*Fx*dt^2/m;
        y = y + Vy*dt;
        Vx = Vx + Fx*dt/m;

        % Determine Witch electrons scatter and update velocity
        scatter = rand(1000,1) < (1 - exp(-dt/0.2e-12));
        Vx(scatter) = normrnd(0,std,size(Vx(scatter)));
        Vy(scatter) = normrnd(0,std,size(Vy(scatter)));

        xplot = transpose([xold(1:10) x(1:10)]);
        yplot = transpose([yold(1:10) y(1:10)]);
        Tplot(i) = (1/(2*K))*mean(Vx.^2 + Vy.^2)*m;

        %Current density and Current Calculations
        Jx = mean(Vx)*1000*electronsPerParticle*(-q);
        Ix(i) = Jx*W*L;
        figure(2)
        subplot(3,1,1);
        plot(xplot,yplot)
        xlim([0 L])
        ylim([0 W])
        title('Electron Trajectory')
        xlabel('x')
        ylabel('y')
        hold on
        subplot(3,1,2)
        plot(Tplot(1:i))
        title('Temperature vs Time Step')
        xlabel('Number of Time Steps')
        ylabel('Temperature (K)')
        hold on
        subplot(3,1,3)
        plot(Ix(1:i))
        title('Current vs Time Step')
        xlabel('Number of Time Steps')
        ylabel('Current (I)')
        hold on
        drawnow


    end
```



As shown in the current vs time plot, when the simulation is initially started, the electrons are moving in random directions with a mean drift velocity of zero. This results in an verall current of 0. The electrons are then accelerated in the negative-x direction ny the electric feild, and the current begins to increase. After a while the the current levels off and stays relativly constant because the scattering of the electrons with the lattice combined with the effects of the electric feild produce a constant drift velocity. The next step is to plot the electron density and temperatre distributions. This is shown below.

```
% Show electron distribution
figure(3);
hist3([x y],'CdataMode','auto');
view(2);
title('Electron Density');
colorbar;
xlabel('x (m)');
ylabel('y (m)');
title('Electron Density Heat Map');

temp_sum_x = zeros(20,10);
temp_sum_y = zeros(20,10);
temp_num = zeros(20,10);

for i=1:1000
 % Find which "bin" it belongs in:
 x1 = floor(x(i)/1e-8);
 y1 = floor(y(i)/1e-8);
 if(x1<=0)
 x1 = 1;
 end
 if(y1<=0)
 y1= 1;
 end
 if(y1>10)
     y1 = 10;
 end
 if(x1>20)
     x1=20;
 end
 % Add its velocity components to the cumulative count:
 temp_sum_y(x1,y1) = temp_sum_y(x1,y1) + Vy(i).^2;
 temp_sum_x(x1,y1) = temp_sum_x(x1,y1) + Vx(i).^2;
 temp_num(x1,y1) = temp_num(x1,y1) + 1;

end

temp = (temp_sum_x + temp_sum_y).*m./K./2./temp_num;
temp(isnan(temp)) = 0;
temp = transpose(temp);


figure(4)
%Plot temperature distribution
[X Y] = meshgrid(linspace(0,200,20),linspace(0,100,10));
surf(X,Y,temp)
title('Temperature Map');
xlabel('x (nm)');
ylabel('y (nm)');
zlim([0 1500]);
```


Electron Density Heat Map
×10^-8
y (m)
x (m)
×10^-8
0
5
10
15
20
0
1
2
3
4
5
6
7
8
9
10
0
2
4
6
8
10
12
14
16
18

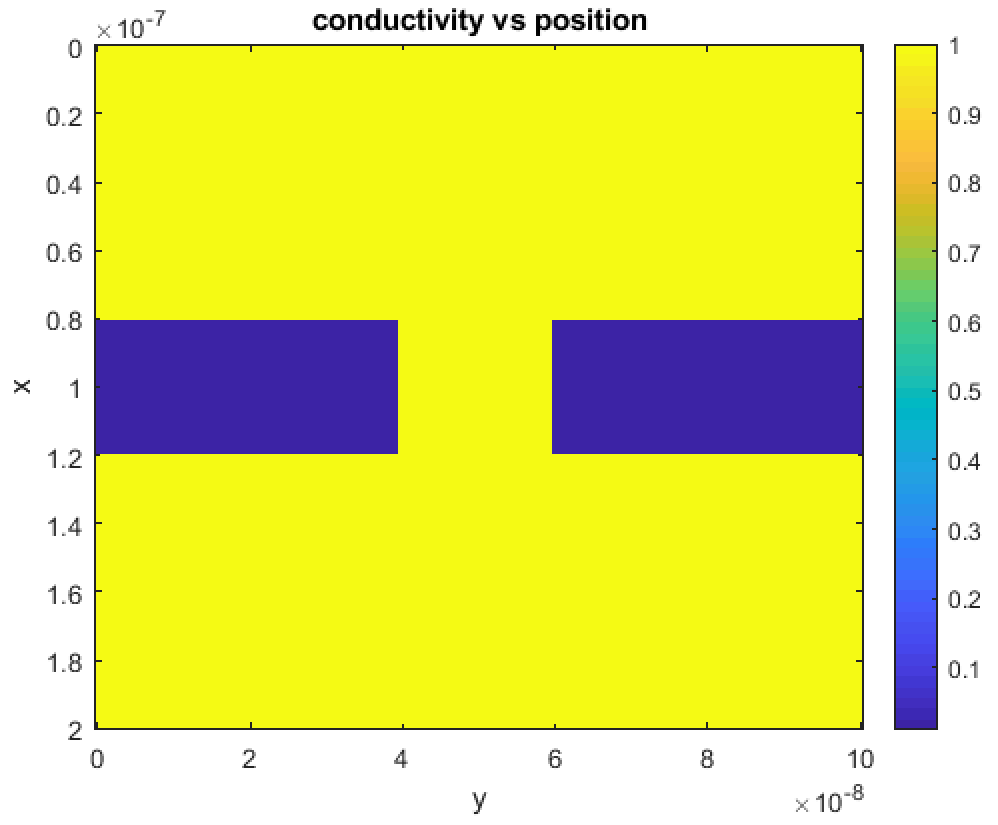

Temperature Map
1500
1000
500
0
100
50
0
y (nm)
0
50
100
150
200
x (nm)

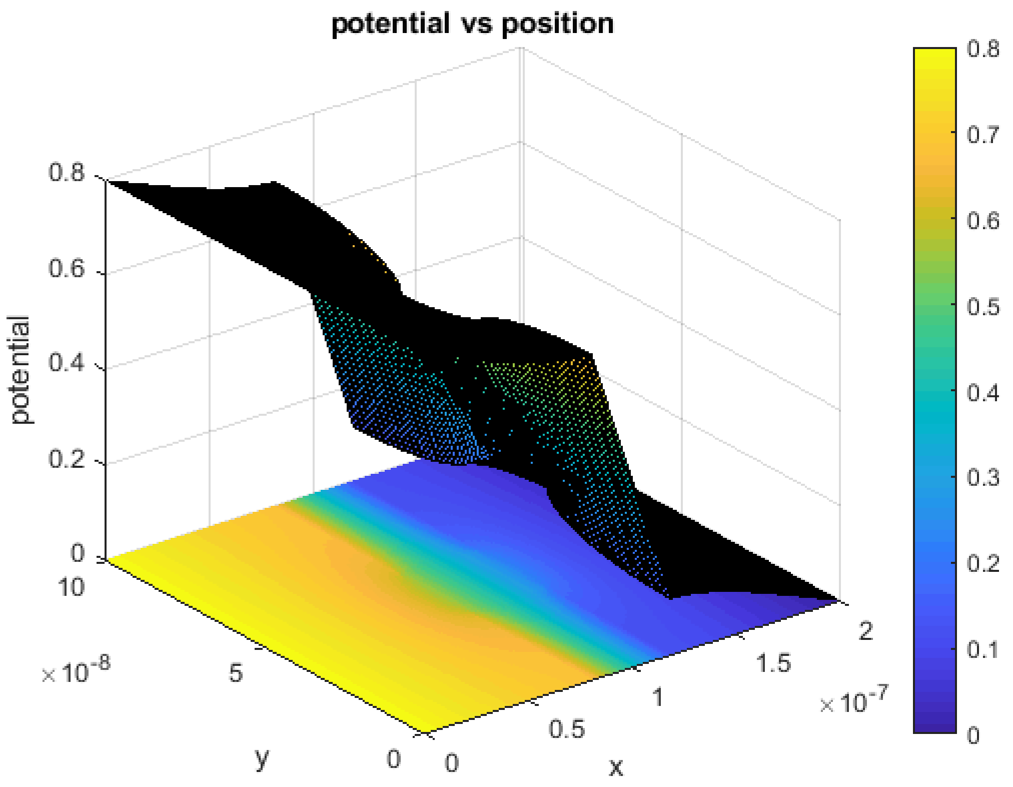
Part 2

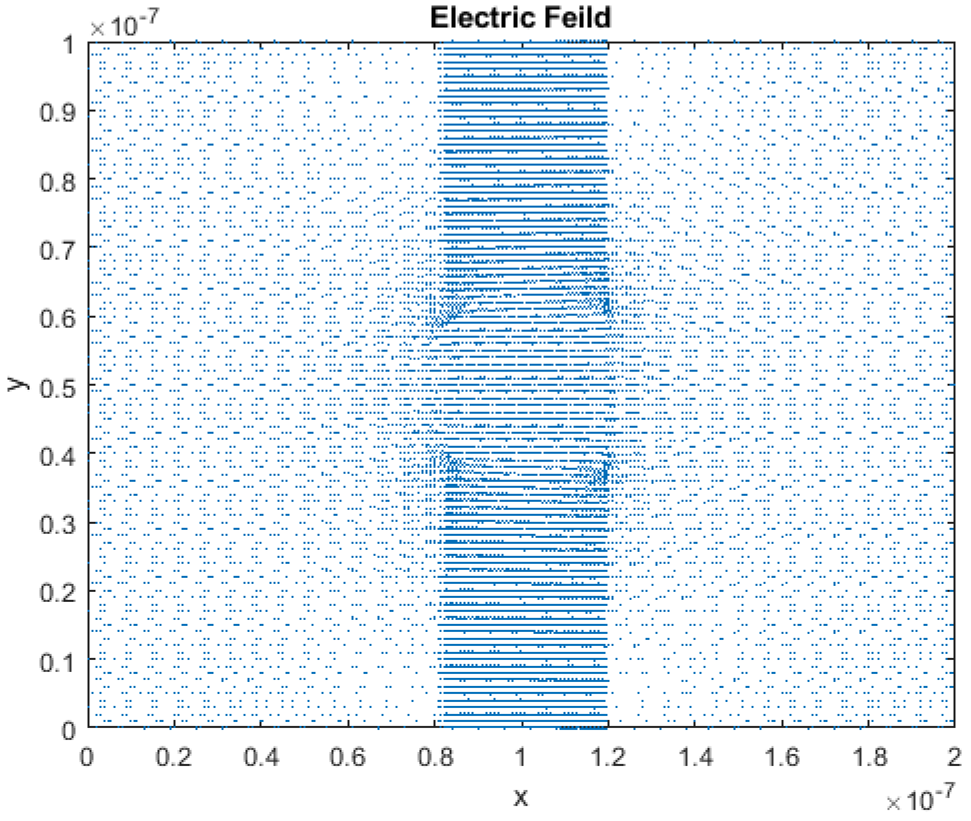
In this part of code I use the finite difference approximation to determine the electric feild inside the bottleneck region of assignment 2 This code is essentially identicle to the code from the second assignment so I won't go over the code and the results in detail but I will show them below.

```
clear all

L = 200e-9;
W = 100e-9;
Lb = 40e-9;
Wb = 40e-9;
meshspace = 1e-9;
V = 0.8;
nx = round(L/meshspace + 1);
ny = round(W/meshspace + 1);

cond1 = 1;
cond2 = 1e-2;


%This section creates a conductivity map witch assigns a conductivity
 to
%each node in the mesh. This will be used to create the G matrix
condMap = zeros(nx,ny);

for i = 1:nx
   for j = 1:ny
       if (i-1)>0.5*(L-Lb)/meshspace&&(i-1)<0.5*(L+Lb)/
meshspace&&((j-1)<Wb/meshspace||(j-1)>(W-Wb)/meshspace)
           condMap(i,j) = cond2;
       else
           condMap(i,j) = cond1;
       end
   end
end

figure(5)
imagesc([0 W],[0 L],condMap);
colorbar
xlabel('y')
ylabel('x')
title('conductivity vs position')

G = sparse(nx*ny);
B = zeros(1,nx*ny);
for i = 1:nx
    for j = 1:ny
        n = j +(i-1)*ny;

        %V=1 @ x=0 BC
        if i == 1
            G(n,n) = 1;
        B(n) = V;

    %V=0 @ x=L BC
    elseif i == nx
        G(n,n) = 1;

    %Absorbing BC @ y=0
    elseif j == 1
        nxm = j + (i-2)*ny;
        nxp = j + i*ny;
        nyp = j+1 + (i-1)*ny;

        %Resistor Values from conduction map
        rxm = (condMap(i,j) + condMap(i-1,j))/2;
        rxp = (condMap(i,j) + condMap(i+1,j))/2;
        ryp = (condMap(i,j) + condMap(i,j+1))/2;

        %node equations from resistor values
        G(n,n) = -(rxm + rxp + ryp);
        G(n,nxm) = rxm;
        G(n,nxp) = rxp;
        G(n,nyp) = ryp;

     %Absorbing BC @ y=W
    elseif j == ny
        nxm = j + (i-2)*ny;
        nxp = j + i*ny;
        nym = j-1 + (i-1)*ny;

        %Resistor Values from conduction map
        rxm = (condMap(i,j) + condMap(i-1,j))/2;
        rxp = (condMap(i,j) + condMap(i+1,j))/2;
        rym = (condMap(i,j) + condMap(i,j-1))/2;

        %node equations from resistor values
        G(n,n) = -(rxm + rxp + rym);
        G(n,nxm) = rxm;
        G(n,nxp) = rxp;
        G(n,nym) = rym;

    %internal nodes
    else
        nxm = j + (i-2)*ny;
        nxp = j + i*ny;
        nym = j-1 + (i-1)*ny;
        nyp = j+1 + (i-1)*ny;

        %Resistor Values from conduction map
        rxm = (condMap(i,j) + condMap(i-1,j))/2;
        rxp = (condMap(i,j) + condMap(i+1,j))/2;
        ryp = (condMap(i,j) + condMap(i,j+1))/2;
        rym = (condMap(i,j) + condMap(i,j-1))/2;

        %node equations from resistor values
```

```
            G(n,n) = -(rxm + rxp + rym + ryp);
            G(n,nxm) = rxm;
            G(n,nxp) = rxp;
            G(n,nym) = rym;
            G(n,nyp) = ryp;

        end
    end
end

%solving and plotting
V = G\B';
Vmap = zeros(nx,ny);
for i = 1:nx
    for j = 1:ny
        n = j +(i-1)*ny;
        Vmap(i,j) = V(n);
    end
end

[X, Y] = meshgrid(0:meshspace:L,0:meshspace:W);
figure(6)
surf(X',Y',Vmap)
colorbar
hold on
imagesc([0 L],[0 W],Vmap')
xlabel('x')
ylabel('y')
zlabel('potential')
title('potential vs position')
hold off

[Ey, Ex] = gradient(Vmap,meshspace);
Ex = -Ex;
Ey = -Ey;

figure(7)
quiver(X',Y',Ex,Ey)
xlim([0 L])
ylim([0 W])
xlabel('x')
ylabel('y')
title('Electric Feild')
```


conductivity vs position
x
y
$\times 10^{-7}$
$\times 10^{-8}$
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0
2
4
6
8
10
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1


potential vs position
potential
y
x
$\times 10^{-8}$
$\times 10^{-7}$
0.8
0.6
0.4
0.2
0
10
5
0
0
0.5
1
1.5
2
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

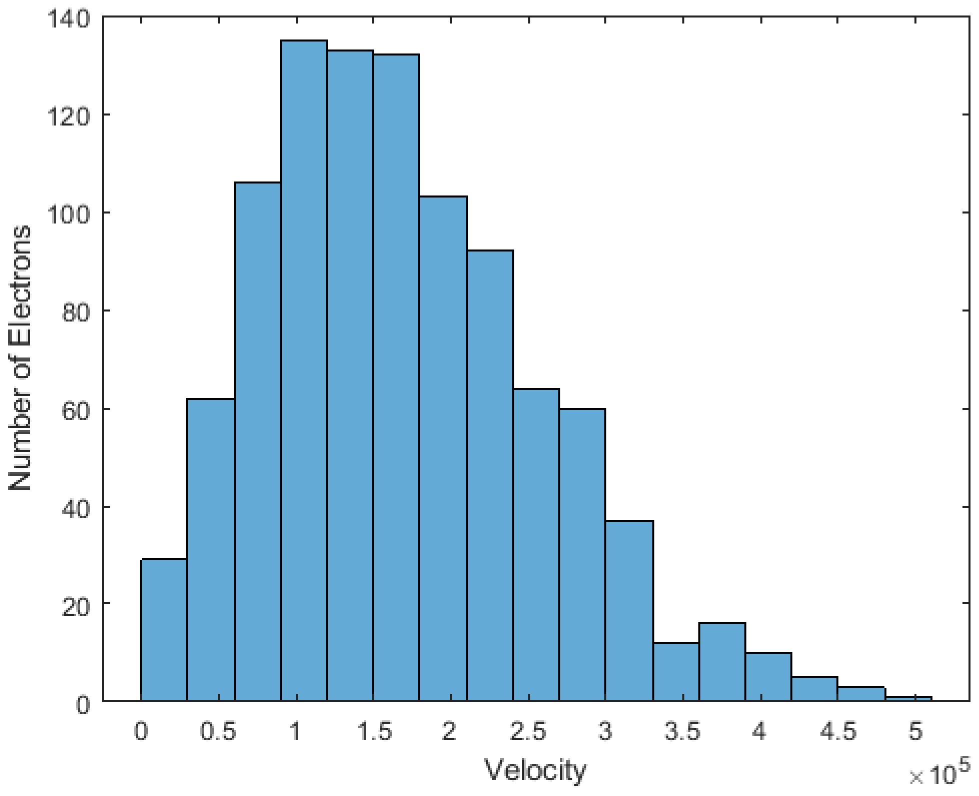


Part 3

in this part of the assignment parts 1 and 2 are mixed together. The electric feild calculated in part 2 is used in a Monte Carlo Simulation to determine the trajectories of the electrons through the bottleneck. The following code is used to set up the basic parameters for the simulation.

```
%Constants
K = 1.3806e-23;
m = 0.26*9.1093e-31;
q = 1.60217662e-19;

%Place electrons in Boundary
x = L*rand(1000,1);
y = W*rand(1000,1);

yboundSpecular = true; %Specular reflection when treu, diffuse
 otherwize;
xboundSpecular = true;
boxSpecular = true;

%Determines if any electrons are in the box and moves them out
inbox1 = x > 0.5*(L-Lb) & x < 0.5*(L+Lb) & y > (W-Wb);
inbox2 =  x > 0.5*(L-Lb) & x < 0.5*(L+Lb) & y < Wb;
x(inbox1) = x(inbox1) + ((rand() > 0.5)*2 -
 1)*Lb*rand(size(x(inbox1)));
```

```
x(inbox2) = x(inbox2) + ((rand() > 0.5)*2 -
 1)*Lb*rand(size(x(inbox2)));
y(inbox1) = y(inbox1) - 0.5*Wb*rand(size(y(inbox1)));
y(inbox2) = y(inbox2) + 0.5*Wb*rand(size(y(inbox2)));



% Assign electron velocity based on Maxwell Boltzman Distribution
T = 300;
vth = sqrt(2*K*T/m);
std = sqrt(K*T/m);
Vx = normrnd(0,std,[1000,1]);
Vy = normrnd(0,std,[1000,1]);
V = sqrt(Vx.^2 + Vy.^2);
figure(8)
histogram(V);
xlabel('Velocity')
ylabel('Number of Electrons')
Tplot = zeros(1000,1); %calculated temperature for plotting

dt = 100e-9/vth/100;
```



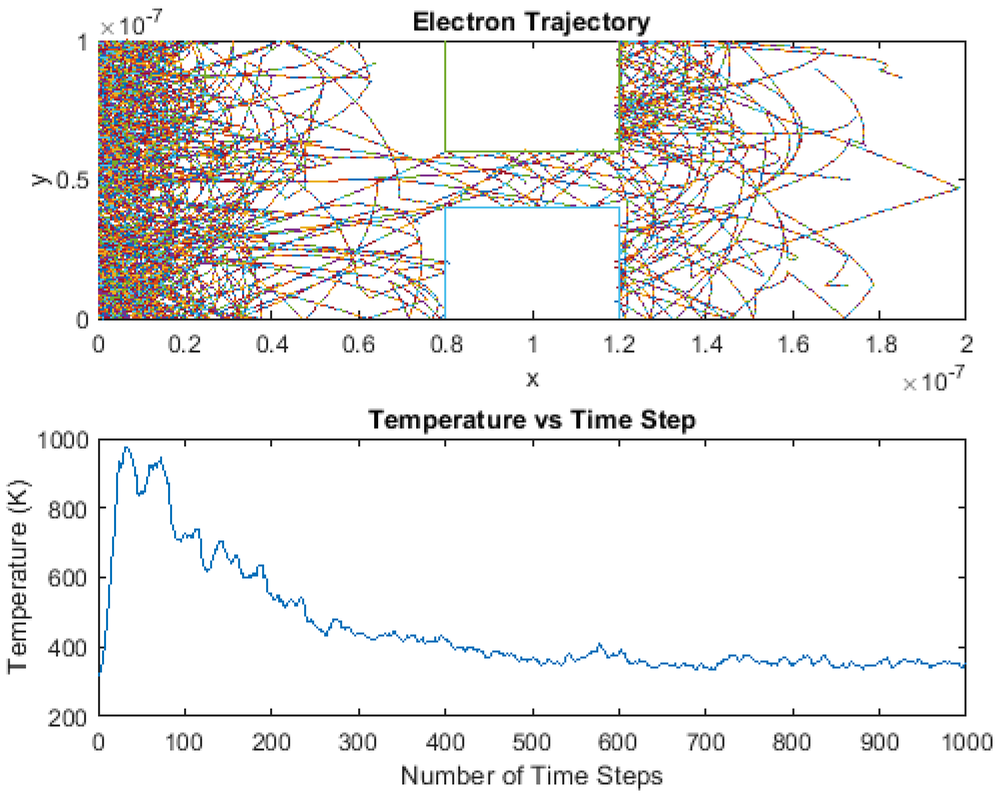
The next step is to determine the force acting on each electron. In this simulation the electric feild is only defined on discrete intervals of 1 nm. Therefore, we need to determine the electric feild at the closest mesh point for each electron then determine the force acting on the electrons. This is shown below.

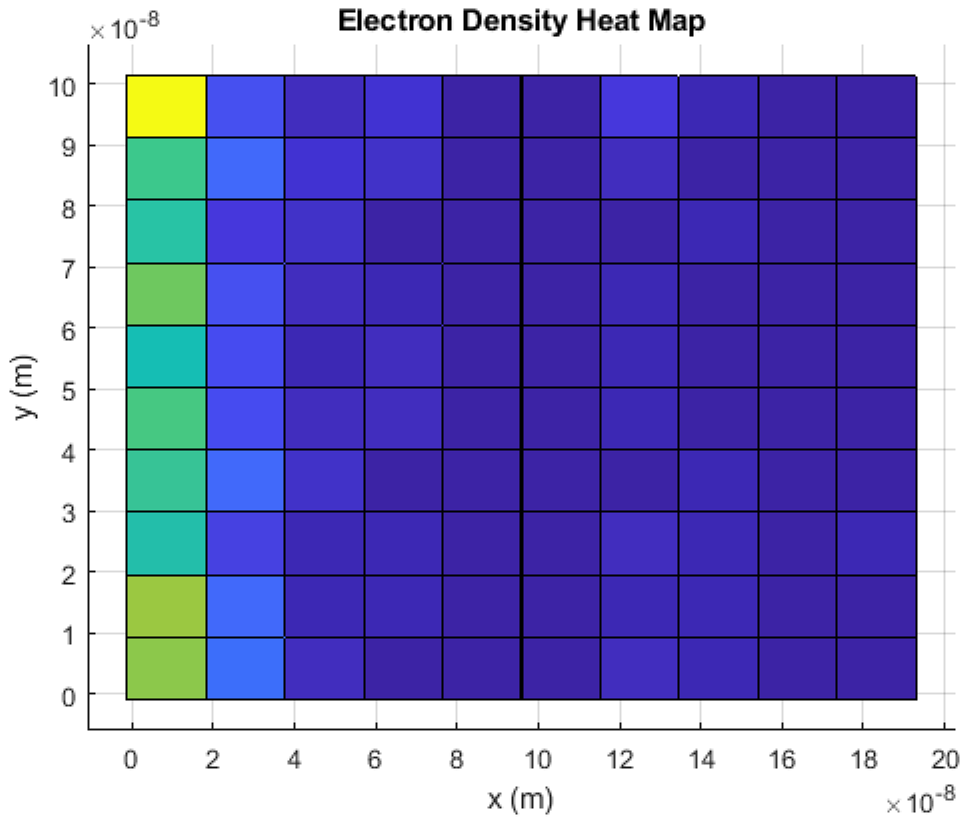
```
Fx = zeros(1000,1);
```

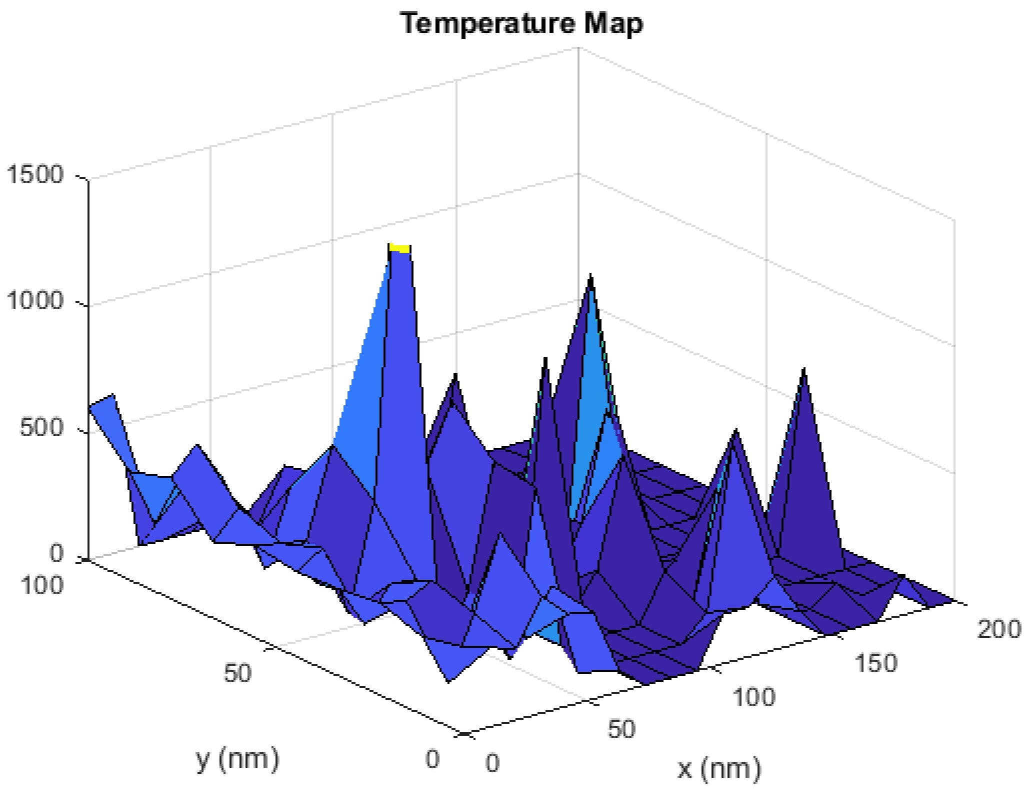
```
Fy = zeros(1000,1);
for i = 1:1000
    Fx(i) = -q*Ex(round(1e9*(x(i)+1e-9)),round(1e9*(y(i)+1e-9)));
    Fy(i) = -q*Ey(round(1e9*(x(i)+1e-9)),round(1e9*(y(i)+1e-9)));
end

% The electron trajectories are calculated using the code shown below.

Tplot = zeros(1000);
xold = x;
yold = y;
for i =1:1000
    %Defines the boundaries of the simulation as well as the boxes
    yboundTop = y > W;
    yboundBottom = y < 0;
    inbox1 = x >= 80e-9 & x <= 120e-9 & y >= 60e-9;
    inbox2 = x >= 80e-9 & x <= 120e-9 & y <= 40e-9;
    xboundRight = x > L;
    xboundLeft = x < 0;

    %Reflection off of xboundary
    if xboundSpecular
        Vx(xboundRight | xboundLeft) = - Vx(xboundRight | xboundLeft);
    else
        theta = pi*rand();
        Vx(xboundRight | xboundLeft) = V(xboundRight |
 xboundLeft)*cos(theta);
        Vy(xboundRight | xboundLeft) = V(xboundRight |
 xboundLeft)*sin(theta);
    end

    %Reflection off of y boundary
    if yboundSpecular
        Vy(yboundTop | yboundBottom) = -Vy(yboundTop | yboundBottom);
    else
        theta = pi*rand();
        Vy(yboundTop | yboundBottom) = V(yboundTop |
 yboundBottom)*cos(theta);
        Vx(yboundTop | yboundBottom) = V(yboundTop |
 yboundBottom)*sin(theta);
    end

    %Reflection off of box
    if boxSpecular
        %Reflection off of verticle face
        Vx(inbox1 & yold >= 60e-9) = -Vx(inbox1 & yold >= 60e-9);
        Vx(inbox2 & yold <= 40e-9) = -Vx(inbox2 & yold <= 40e-9);

        %Reflection off of Horizontal face
        Vy(inbox1 & yold <= 60e-9) = -Vy(inbox1 & yold <= 60e-9);
        Vy(inbox2 & yold >= 40e-9) = -Vy(inbox2 & yold >= 40e-9);
    else
        theta = pi*rand();
        %Reflection off of verticle face
        Vx(inbox1 & yold >= 60e-9) = V(inbox1 & yold >=
60e-9)*cos(theta);
        Vx(inbox2 & yold <= 40e-9) = V(inbox2 & yold <=
40e-9)*cos(theta);
        Vy(inbox1 & yold >= 60e-9) = V(inbox1 & yold >=
60e-9)*sin(theta);
        Vy(inbox2 & yold <= 40e-9) = V(inbox2 & yold <=
40e-9)*sin(theta);

        %Reflection off of Horizontal Face
        Vy(inbox1 & yold <= 60e-9) = V(inbox1 & yold <=
60e-9)*cos(theta);
        Vy(inbox2 & yold >= 40e-9) = V(inbox2 & yold >=
40e-9)*cos(theta);
        Vx(inbox1 & yold <= 60e-9) = V(inbox1 & yold <=
60e-9)*sin(theta);
        Vx(inbox2 & yold >= 40e-9) = V(inbox2 & yold >=
40e-9)*sin(theta);
    end

    %Make sure all electrons are in the proper boundary
    y(yboundTop) = 100e-9;
    y(yboundBottom) = 0;
    x(xboundRight) = 200e-9;
    x(xboundLeft) = 0;
    x(inbox1 & yold >= 60e-9 & x <= 100e-9) = 80e-9;
    x(inbox1 & yold >= 60e-9 & x > 100e-9) = 120e-9;
    x(inbox2 & yold <= 40e-9 & x <= 100e-9) =80e-9;
    x(inbox2 & yold <= 40e-9 & x >= 100e-9) =120e-9;
    y(inbox1 & yold <= 60e-9) = 60e-9;
    y(inbox2 & yold >= 60e-9) = 40e-9;


    xold = x;
    yold = y;
    x = x + Vx*dt + 0.5*Fx*dt^2/m;
    y = y + Vy*dt + 0.5*Fy*dt^2/m;
    Vx = Vx + Fx*dt/m;
    Vy = Vy + Fy*dt/m;

    %Scatter electrons randomly and reasign velocity using Maxwell
Boltzman
    %distribution
    scatter = rand(1000,1) < (1 - exp(-dt/0.2e-12));
    Vx(scatter) = normrnd(0,std,size(Vx(scatter)));
    Vy(scatter) = normrnd(0,std,size(Vy(scatter)));


    xplot = transpose([xold(1:20) x(1:20)]);
    yplot = transpose([yold(1:20) y(1:20)]);
    figure(9)
    subplot(2,1,1)
    Tplot(i) = (1/(2*K))*mean(Vx.^2 + Vy.^2)*m;
```

```
        plot(xplot,yplot)
        xlim([0 L])
        ylim([0 W])
        title('Electron Trajectory')
        xlabel('x')
        ylabel('y')
        hold on
        plot(1e-9*[80 80 120 120],1e-9*[200 60 60 200])
        plot(1e-9*[80 80 120 120],1e-9*[0 40 40 0])

        subplot(2,1,2)
        plot(Tplot(1:i))
        title('Temperature vs Time Step')
        xlabel('Number of Time Steps')
        ylabel('Temperature (K)')
        drawnow
    end

    figure(10)
    hist3([x y],'CdataMode','auto');
    view(2);
    title('Electron Density');
    xlabel('x (m)');
    ylabel('y (m)');
    title('Electron Density Heat Map');

    temp_sum_x = zeros(20,10);
    temp_sum_y = zeros(20,10);
    temp_num = zeros(20,10);

    for i=1:1000
     % Find which "bin" it belongs in:
     x1 = floor(x(i)/1e-8);
     y1 = floor(y(i)/1e-8);
     if(x1<=0)
     x1 = 1;
     end
     if(y1<=0)
     y1= 1;
     end
     if(y1>10)
         y1 = 10;
     end
     if(x1>20)
         x1=20;
     end
     % Add its velocity components to the cumulative count:
     temp_sum_y(x1,y1) = temp_sum_y(x1,y1) + Vy(i).^2;
     temp_sum_x(x1,y1) = temp_sum_x(x1,y1) + Vx(i).^2;
     temp_num(x1,y1) = temp_num(x1,y1) + 1;

    end

    temp = (temp_sum_x + temp_sum_y).*m./K./2./temp_num;
```

```
temp(isnan(temp)) = 0;
temp = transpose(temp);


[X Y] = meshgrid(linspace(0,200,20),linspace(0,100,10));
figure(11)
surf(X,Y,temp)
title('Temperature Map');
xlabel('x (nm)');
ylabel('y (nm)');
zlim([0 1500]);
```




Electron Density Heat Map
×10^-8
10
9
8
7
6
5
4
3
2
1
0
y (m)
0
2
4
6
8
10
12
14
16
18
20
x (m)
×10^-8


Temperature Map
1500
1000
500
0
100
50
0
y (nm)
0
50
100
150
200
x (nm)

Examining the density plots we see that the electronsare heavily grouped towards the left side of the device. This makes sence because this is the positive terminal of the device and the negatice electrons will be atrracted towards that terminal.

The next step in this simulation is to determine the position of each electron and then re solve Poisson's equation in order to re determine the electric feild then with this new electric feild, re calculate the position of the electrons. This process would be continue throughout the entire simulation. This would increase the accuracy of the simulation but significantly increase the simulation time.

Published with MATLAB® R2018a