Carleton University

Course: ELEC 4700 Modelling of Integrated Device

Assignment No: 3

Monte-Carlo/Finite Difference Method

Kwabena Gyasi Bawuah

101048814

Due: 03/21/2021

Table of Contents

Introduction:

This report is for ELEC 4700 in response to the call for an assignment report. The assignment was to couple the two simulators from the first 2 assignments. These particles are to be giving velocities using the Maxwell-Boltzmann distribution. Lastly, an enhancement is to put the system to test by adding a bottle neck boundary. This report will detail the results from the built simulation, observation of results, discussion of results, answers to specific questions asked in the assignment and conclusions derived. Samples of the code used to perfume these models will also be produced within the sections.

Monte-Carlo modification:

a) Using the formula E=V/d with d being the distance of the voltage difference.

With V = 0.1V and x=200nm

E = 500kN/C

Given from code:

Ex =

5.5556e+05

Ey =

0

b) Using the formula F=E*q

With E=500kN/C

Given from code:

Fx =

-8.9010e-14

Fy =

0

c) Electron Acceleration

a = F/mass

This gives a value of 8.7935x10^-13 N

d)

For each particle, the Average carrier velocities are.

For x component: 1/ (number of particles) *($\sum_{k=0}^{n}$ velocity componet in x)

For y component: 1/ (number of particles) *($\sum_{k=0}^{n}$ velocity componet in y)

The assumed electron concentration is $p=10^{15}cm^{-2}$.

Drift current density in x = (q*p/ number of particles) $\sum_{n=1}^{N}$ velocity componet in x

Drift current density in y = (q*p/ number of particles) $\sum_{n=1}^{N}$ velocity componet in y

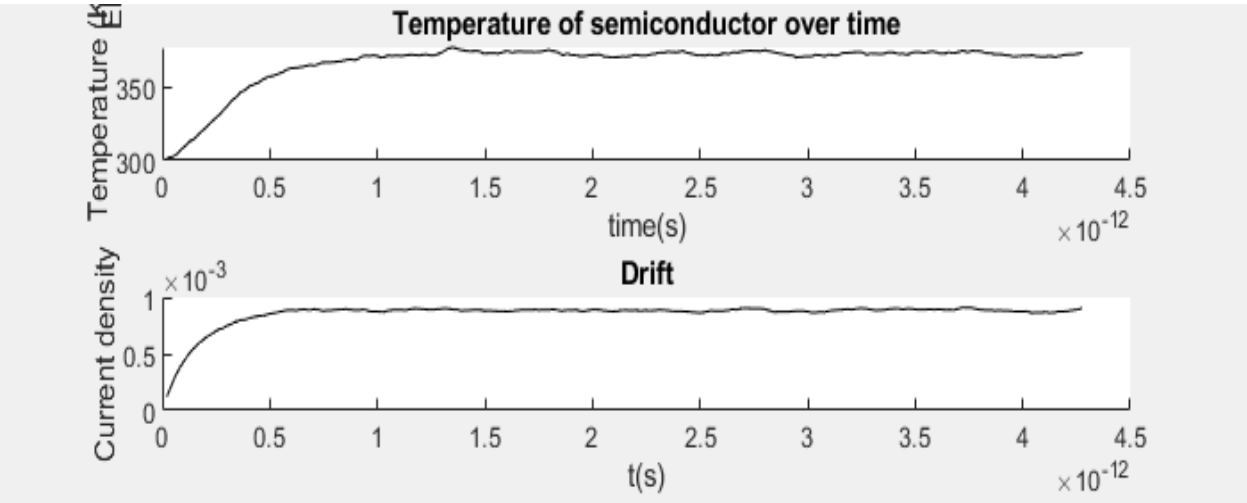
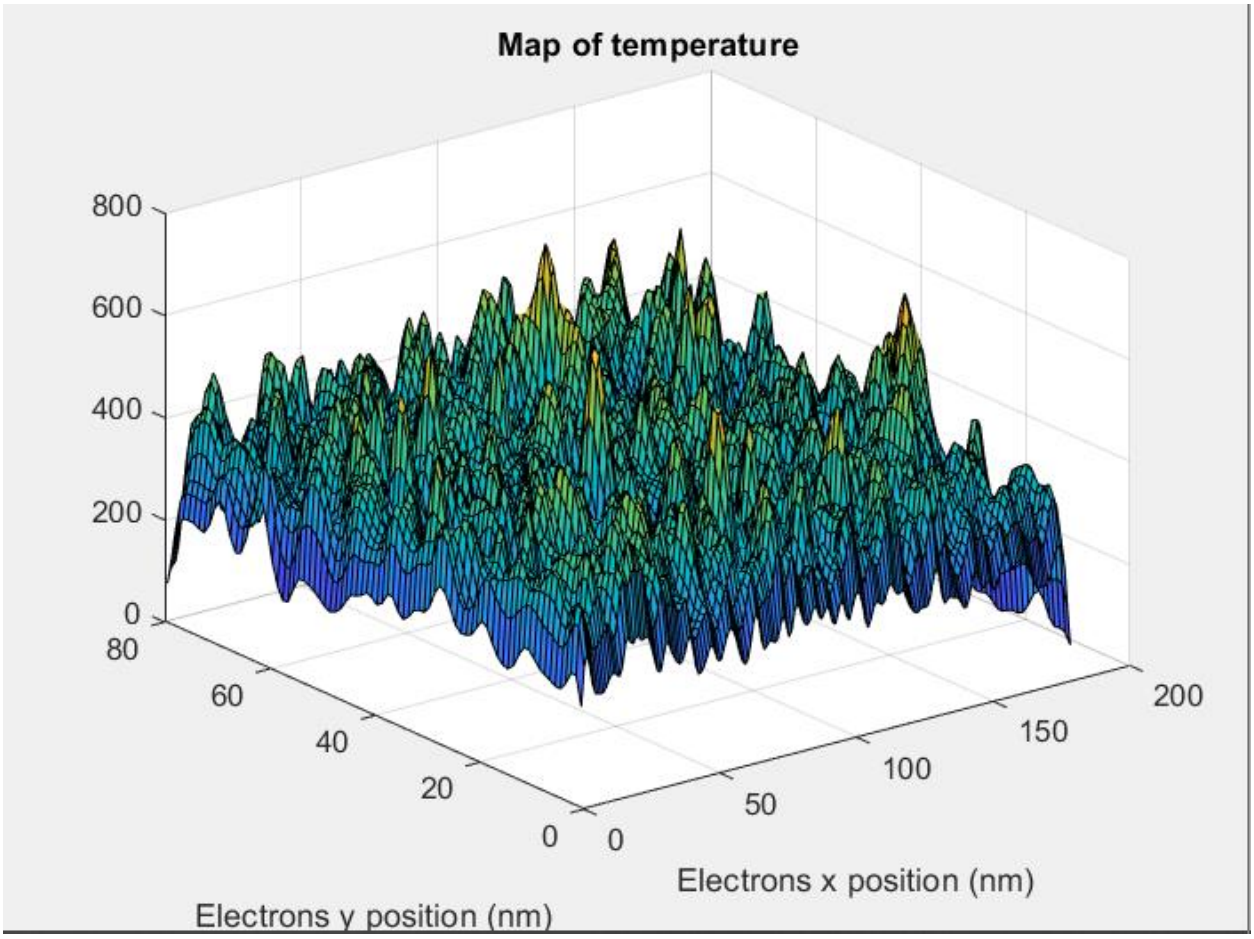


Figure 3: Current Plot

As the current begins to diffuse into the space, it starts from zero and then gets to constant current density.

e)


Map of temperature
800
600
400
200
0
80
60
40
20
0
0
50
100
150
200
Electrons x position (nm)
Electrons y position (nm)

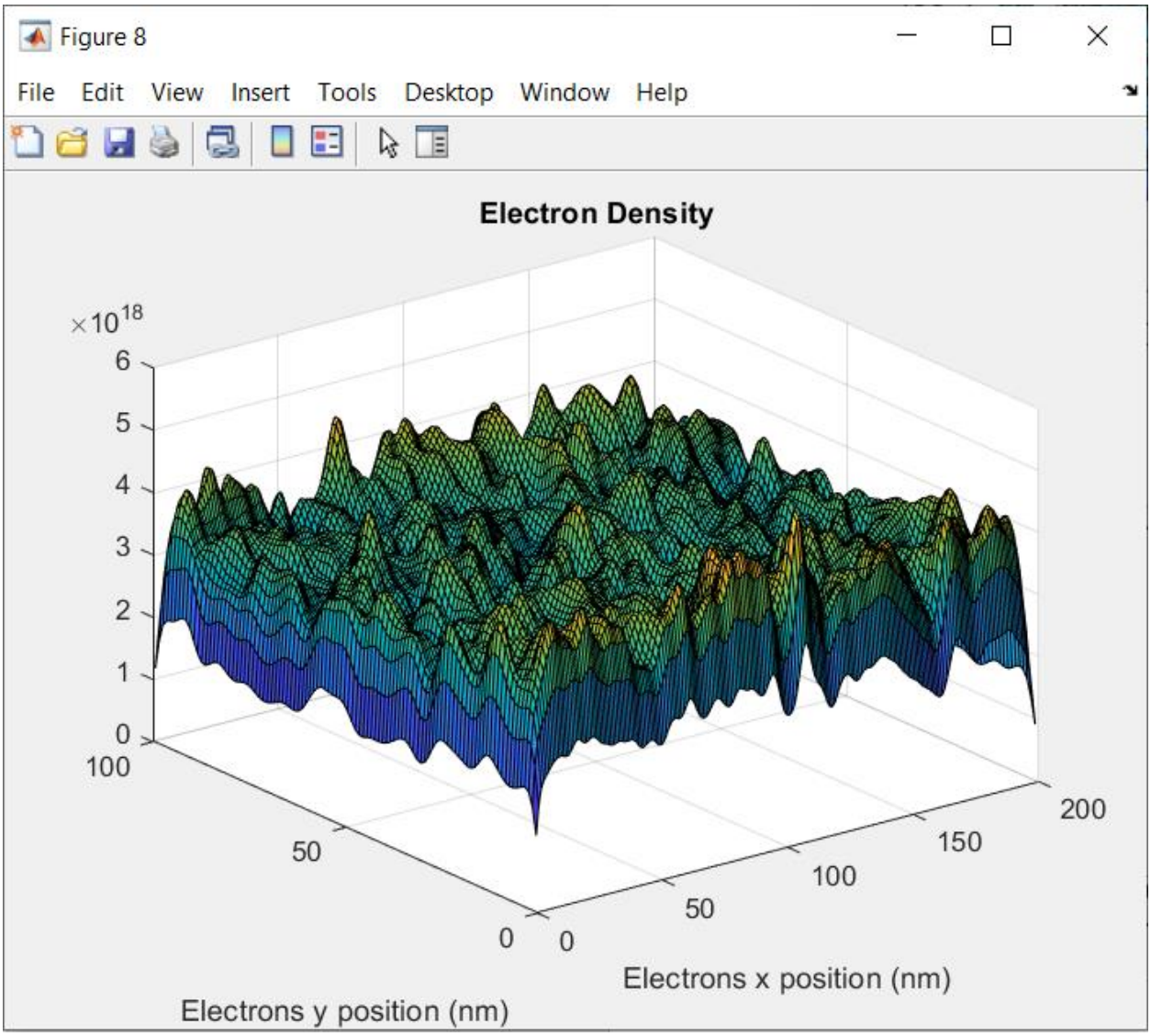


Figure 4: Sample density and temperature plots

Code Used for Q1:

```
%Assign 3
%q1
%Editting Assig 1 part 3
close all;
clc
%Kwabena Gyasi Bawuah
%101048814
%UNTITLED Summary of this function goes here
```

```
%   Detailed explanation goes here
%electron spec
 global C

    addpath ../geom2d/geom2d

    C.q_0 = 1.60217653e-19;             % electron charge
    C.hb = 1.054571596e-34;             % Dirac constant
    C.h = C.hb * 2 * pi;                    % Planck
constant
    C.m_0 = 9.10938215e-31;             % electron mass
    C.kb = 1.3806504e-23;               % Boltzmann
constant
    C.eps_0 = 8.854187817e-12;          % vacuum
permittivity
    C.mu_0 = 1.2566370614e-6;           % vacuum
permeability
    C.c = 299792458;                    % speed of light
    C.g = 9.80665; %metres (32.1740 ft) per s²

    T = 300;
    k = 1.38e-23;
    mn = 0.26*C.m_0; %effective mass
    tmn = 0.2e-12;    % Mean time between collisions

    vth = sqrt((2*C.kb*T)/mn);% Thermal velocity

    freepath = vth*tmn   % mean free path

    ConductorL = 180e-9;
    ConductorW = 80e-9;

    dpoints = 5e4;
    ecount = 15; %the number of electron to show on plot

    %electron concentratoin
    den = 1e15*100^-2;

    detaT= ConductorW/vth/100;
    sims = 1000;

%     Xpos = rand(1,ecount).*ConductorW;
%     Ypos = rand(1,ecount).*ConductorL;
    traj=zeros(sims,ecount*2);
```

```
    temp=zeros(sims, 1);

    temp(:,1)= 300;

    Pscat = 1-exp(-detaT/tmn);
    %----------------------------------------
    ProbDistr = makedist('Normal','mu', 0, 'sigma',
sqrt(C.kb*T/mn));

    Vx = 0.1;
    Vy = 0;
    dens = 1e15*100^-2;

    Ex = Vx/ConductorL
    Ey = Vy/ConductorW

    Fx = -C.q_0*Ex
    Fy = -C.q_0*Ey

    dVx = Fx*detaT/mn
    dVy = Fy*detaT/mn
    dVx = dVx.*ones(dpoints,1);
    dVy = dVy.*ones(dpoints,1);

    tspec = 0;
    bspec=0;

    boxes = 1e-9.*[80 120 0 40; 80 120 60 100];
    specularbox = [0 1];

    for i = 1: dpoints
        angle = rand*2*3.14;
        state(i,:)= [ConductorL*rand ConductorW*rand
random(ProbDistr) random(ProbDistr)];

%         if (state(i,2)>60e-9 &(state(i,1)>80e-9 &
state(i,1)<120e-9))  | (state(i,2)< 40e-9 &(state(i,1)>80e-
9 & state(i,1)<120e-9))
%             state(i,1:2) = [ConductorL*rand
ConductorW*rand];
%         end

    end
```

```
    %take plot ot of loop to stop the refresh
figure(5);
subplot(4,1,2);
%         plot(detaT*(0:i-1), temp(1:i));
%         plot(detaT*(0:i-1),temp(1:i));
tPlot = animatedline;
xlabel('time(s)');
ylabel('Temperature (K)');
title('Temperature of semiconductor over time');

figure(5);
subplot(4,1,3);
%part = sqrt(state(:,3).^2 + state(:,4).^2);
% xlim([0 7e5]);
% ylim([0 2000]);
%histogram(part);
currentPlot = animatedline ;
xlabel('t(s)');
ylabel('Current density');
title('Drift');

    for i = 1:sims
        %initialize new states
        state(:,3) = state(:,3) + dVx;
        state(:,4) = state(:,4) + dVy;
        state(:,1:2) = state(:,1:2) + detaT.*state(:,3:4);

        out = state(:,1) > ConductorL;
        state(out,1) = state(out,1) - ConductorL;

        out = state(:,1) < 0;
        state(out,1) = state(out,1) + ConductorL;

        out = state(:,2) > ConductorW;

    if(tspec)
        state(out,2) = 2*ConductorW - state(out,2);
        state(out,4) = -state(out,4);
    else
        state(out,2) = ConductorW;
        part = sqrt(state(out,3).^2 + state(out,4).^2);
        angle = rand([sum(out),1])*2*3.14;
        state(out,3) = part.*cos(angle);
        state(out,4) = -abs(part.*sin(angle));
```

```
    end

    out = state(:,2) < 0;

    if(bspec)
        state(out,2) = -state(out,2);
        state(out,4) = -state(out,4);
    else
        state(out,2) = 0;
        part = sqrt(state(out,3).^2 + state(out,4).^2);
        angle = rand([sum(out),1])*2*3.41;
        state(out,3) = part.*cos(angle);
        state(out,4) = abs(part.*sin(angle));
    end
%     for out=1: dpoints
%             if (state(out,2)>60e-9 &(state(out,1)>80e-9 &
state(out,1)<120e-9))
%                 boxNum = 1;
%             elseif (state(out,2)< 40e-9
&(state(out,1)>80e-9 & state(out,1)<120e-9))
%                 boxNum = 2;
%             else
%                 boxNum = 0;
%             end
%             while(boxNum ~= 0)
%                 XDist = 0;
%                 newx = 0;
%                 if(state(out,3) > 0)
%                     XDist = state(out,1) -
boxes(boxNum,1);
%                     newx = boxes(boxNum,1);
%                 else
%                     XDist = boxes(boxNum,2) -
state(out,1);
%                     newx = boxes(boxNum,2);
%                 end
%
%                 yDist = 0;
%                 newy = 0;
%                 if(state(out,4) > 0)
%                     yDist = state(out,2) - boxes(boxNum,
3);
%                     newy = boxes(boxNum, 3);
%                 else
```

```
%                       yDist = boxes(boxNum, 4) -
state(out,2);
%                       newy = boxes(boxNum, 4);
%                   end
%
%                   if(XDist < yDist)
%                       state(out,1) = newx;
%                       if(~specularbox(boxNum))
%                           sgn = -sign(state(out,3));
%                           part = sqrt(state(out,3).^2 +
state(out,4).^2);
%                           angle = rand()*2*3.14;
%                           state(out,3) =
sgn.*abs(part.*cos(angle));
%                           state(out,4) = part.*sin(angle);
%                       else
%                           state(out,3) = -state(out,3);
%                       end
%                   else
%                       state(out,2) = newy;
%                       if(~specularbox(boxNum))
%                           sgn = -sign(state(out,4));
%                           part = sqrt(state(out,3).^2 +
state(out,4).^2);
%                           angle = rand()*2*3.14;
%                           state(out,3) = part.*cos(angle);
%                           state(out,4) =
sgn.*abs(part.*sin(angle));
%                       else
%                           state(out,4) = -state(out,4);
%                       end
%                   end
%                    boxNum = 0;
%               end
%       end

        out = rand(dpoints, 1) < Pscat;
        state(out,3:4) = random(ProbDistr, [sum(out),2]);

        temp(i) = (sum(state(:,3).^2) +
sum(state(:,4).^2))*mn/k/2/dpoints;


        for out=1:ecount
```

```
            traj(i, (2*out):(2*out+1)) = state(out,1:2);
        end

        %getting j using density
        %add the points

        if  mod(i,5) == 0
        figure(5);
        subplot(4,1,1);
        hold off;
        plot(state(1:ecount,1)./1e-9,
state(1:ecount,2)./1e-9, 'o');
        hold on;
%         for ouut=1:size(boxes,1)
%            plot([boxes(ouut, 1) boxes(ouut, 1)
boxes(ouut, 2) boxes(ouut, 2) boxes(ouut, 1)]./1e-
9,[boxes(ouut, 3) boxes(ouut, 4) boxes(ouut, 4) boxes(ouut,
3) boxes(ouut, 3)]./1e-9, 'k-');
%         end
        xlim([0 ConductorL/1e-9]);
        ylim([0 ConductorW/1e-9]);
        xlabel('Electrons x position (nm)');
        ylabel('Electrons y position (nm)');
        title('Electrons Simulations');

        subplot(4,1,4);
        part = sqrt(state(:,3).^2 + state(:,4).^2);
        xlim([0 7e5]);
        ylim([0 2000]);
        histogram(part);
        xlabel('v(m/s)');
        ylabel('Particle count');
        title('Histogram to show particle speed');

        J(i, 1) = -C.q_0.*dens.*mean(state(:,3));
        J(i, 2) = -C.q_0.*dens.*mean(state(:,4))

        addpoints(tPlot, detaT.*i, temp(i));
        addpoints(currentPlot, detaT.*i, J(i,1));
        end
    end

    figure(6)
    hold on;
```

```
    xlim([0 ConductorL/1e-9]);
    ylim([0 ConductorW/1e-9]);
    xlabel('Electrons x position (nm)');
    ylabel('Electrons y position (nm)');
    title('Trajectories of Electrons with curl from
field');

    for i = 1: ecount
        plot(traj(:,i*2)./1e-9, traj(:,i*2+1)./1e-9, '-');
    end

%    for out=1:size(boxes,1)
%    plot([boxes(out, 1) boxes(out, 1) boxes(out, 2)
boxes(out, 2) boxes(out, 1)]./1e-9,...
%        [boxes(out, 3) boxes(out, 4) boxes(out, 4)
boxes(out, 3) boxes(out, 3)]./1e-9, 'k-');
%     end

%     figure(7)
%     part = sqrt(state(:,3).^2 + state(:,4).^2);
%     xlim([0 7e5]);
%     ylim([0 2000]);
%     histogram(part);
%     xlabel('v(m/s)');
%     ylabel('Particle count');
%     title('Histogram to show particle speed');

    dens = hist3(state(:,1:2),[200 100])';

    N = 20;
    sigma = 1.5;
    [x y]=meshgrid(round(-N/2):round(N/2), round(-
N/2):round(N/2));
    f=exp(-x.^2/(2*sigma^2)-y.^2/(2*sigma^2));
    f=f./sum(f(:));
    figure(8);
    dens = conv2(dens,f,'same');
    dens =
dens/(ConductorW./size(dens,1)*ConductorL./size(dens,2));
    surf(conv2(dens,f,'same'));
    %set(gca,'YDir','normal');
    title('Electron Density');
    xlabel('Electrons x position (nm)');
    ylabel('Electrons y position (nm)');
```

```
    tempSumX = zeros(ceil(ConductorL/1e-
9),ceil(ConductorW/1e-9));
    tempSumY = zeros(ceil(ConductorL/1e-
9),ceil(ConductorW/1e-9));
    tempSum = zeros(ceil(ConductorL/1e-
9),ceil(ConductorW/1e-9));

    for i=1:dpoints

        x = floor(state(i,1)/1e-9);
        y = floor(state(i,2)/1e-9);
        if(x==0)
            x = 1;
        end
        if(y==0)
            y= 1;
        end

        tempSumY(x,y) = tempSumY(x,y) + state(i,3)^2;
        tempSumX(x,y) = tempSumX(x,y) + state(i,4)^2;
        tempSum(x,y) = tempSum(x,y) + 1;
    end

    temp = (tempSumX + tempSumY).*mn./k./2./tempSum;
    temp(isnan(temp)) = 0;
    temp = temp';

    N = 20;
    sigma = 1.3;
    [x y]=meshgrid(round(-N/2):round(N/2), round(-
N/2):round(N/2));
    f=exp(-x.^2/(2*sigma^2)-y.^2/(2*sigma^2));
    f=f./sum(f(:));
    figure(9);
    surf(conv2(temp,f,'same'));
    %set(gca,'YDir','normal');
    title('Map of temperature');
    xlabel('Electrons x position (nm)');
    ylabel('Electrons y position (nm)');
```

Finite Difference Method:

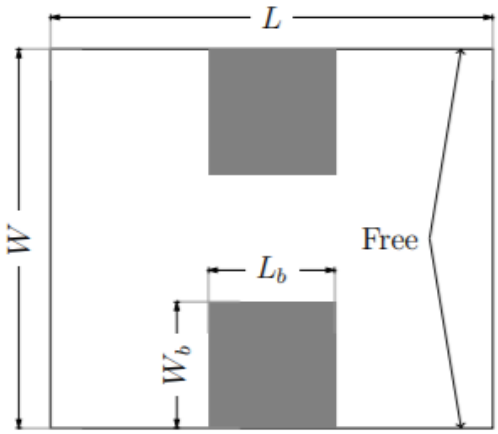


Figure 4: Rectangular region with isolated conducting sides and “bottle-neck”.

Use the Finite Difference Method to calculate the electric field and then provide a field for the Monte-Carlo bottle-neck simulation L×W shown in Figure 3 using $\nabla (\sigma x,y \nabla V) = 0$.

a)

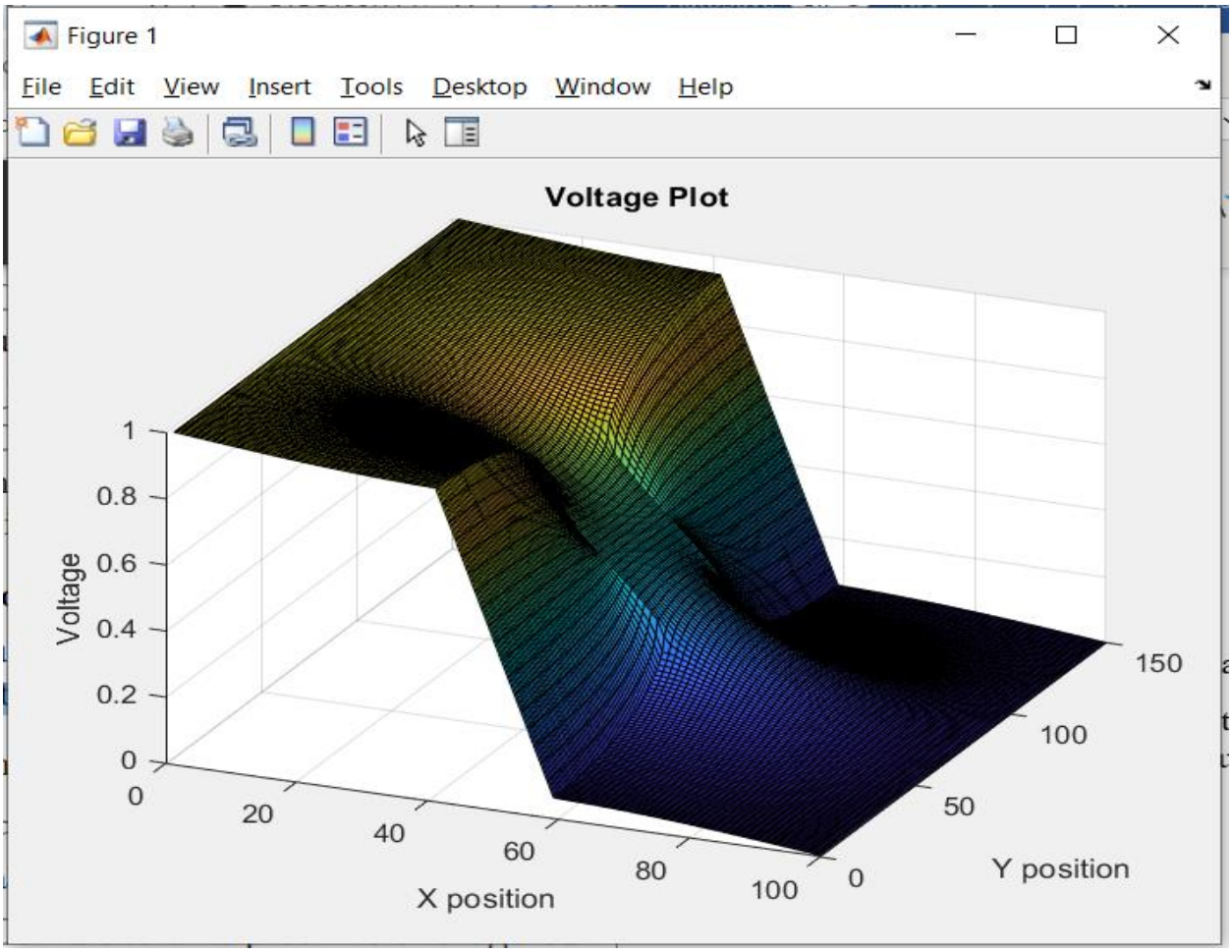


Figure 7: Surface plot of V(x,y)

b)

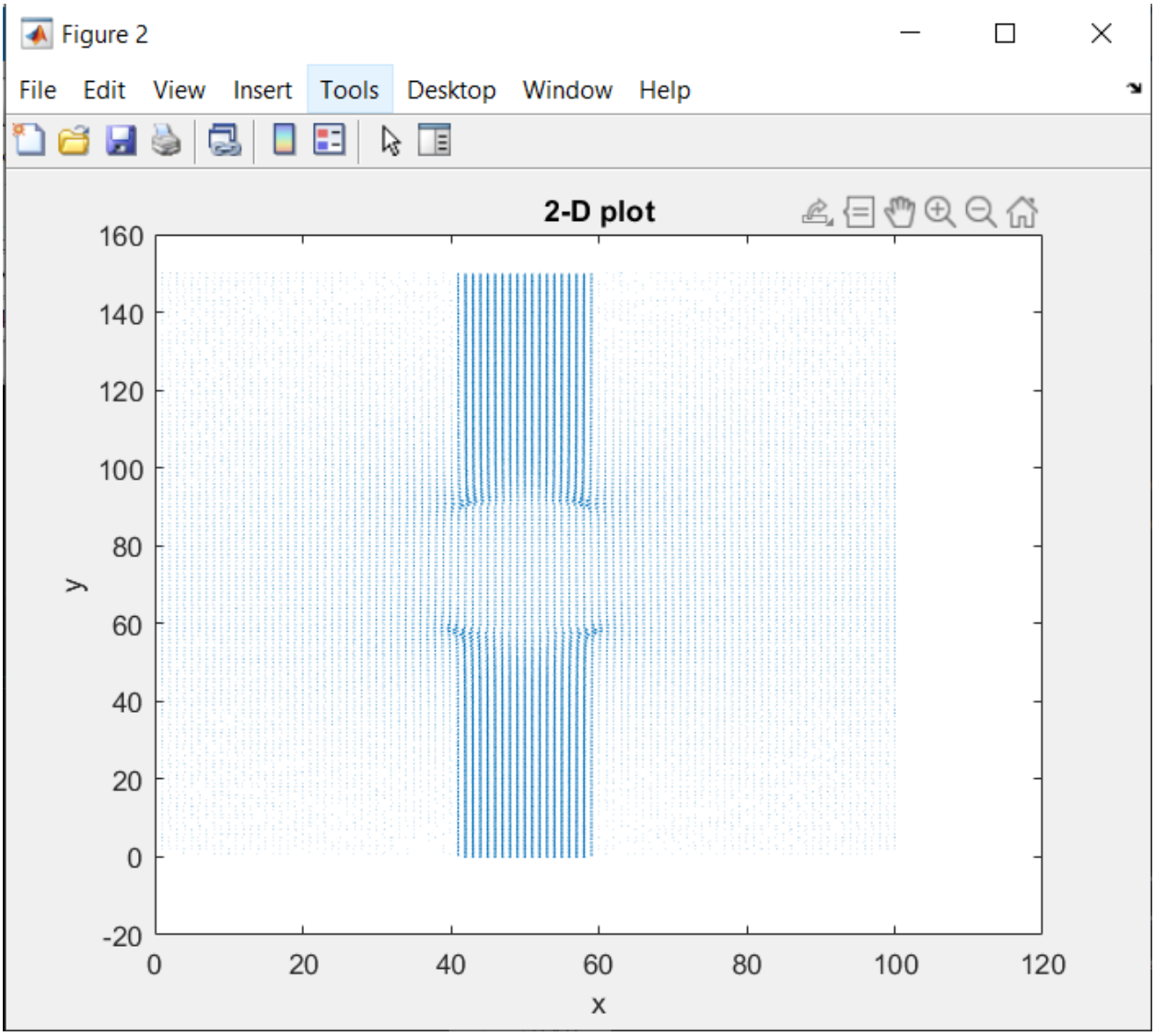


Figure 7: 2-D electric field vector plot.

c)

Was unsure of how to implement due to how I set up the boxes. Took out code because it kept crashing my program.

Code Used for Q2:

```
%Assign 3
%q2
%Kwabena Gyasi Bawuah
%101048814
```

```
% using question 1 a as background

%initiailizing the dimensions of our matrices, ensuring L
is 3/2 times W

W = 100;
L = (3/2)*W;

G = sparse(W*L);
Op = zeros(1, W*L);

boxD = L*(2/5);
boxU = W*(3/5);
boxR = L*(3/5);
boxL = W*(2/5);
%Editting Assig 2 part 2a
%sigma reference in and out the box
sigOut = 1;
sigIn = 10^-2;

% leftEdge = midX - boxL/2;
% rightEdge = midX + boxL/2;
% topEdge = midY + boxW/2;
% bottomEdge = midY - boxW/2;

box = [boxL boxD boxU boxR];


%difine the dimension with the given sigma
for i = 1:W
    for j = 1:L
        if (i > box(1) && i < box(2) && (j < box(3)||j >
box(4)))
            sigmamap(i, j) = sigIn;
        else
            sigmamap(i, j) = sigOut;
        end
    end
end

%then we fill in the whole matrix with the sigma values
defined
for i = 1:W
```

```
    for j = 1:L
        n = j + (i-1)*L;
        nxps = j + (i)*L;
        nxms = j + (i-2)*L;
        nyps = (j + 1) + (i-1)*L;
        nyms = (j - 1) + (i-1)*L;

        if (i == 1)
            G(n, :) = 0;
            G(n,n) = 1;
            Op(n) = 1;
            %already assigned above
            %sigmaMap(i,j) = sigOut;
        elseif (i == W)
            G(n, :) = 0;
            G(n,n) = 1;
            Op(n) = 0;
            %sigmaMap(i,j) = sigOut;
        elseif (j == 1)
            G(n,nxps) = (sigmamap(i+1, j) +
sigmamap(i,j))/2;
            G(n, nxms) = (sigmamap(i-1, j) +
sigmamap(i,j))/2;
            G(n, nyps) = (sigmamap(i, j+1) +
sigmamap(i,j))/2;
            G(n,n) = -(G(n,nxps)+G(n,nxms)+G(n,nyps));
        elseif (j == L)
            G(n,nxps) = (sigmamap(i+1, j) +
sigmamap(i,j))/2;
            G(n, nxms) = (sigmamap(i-1, j) +
sigmamap(i,j))/2;
            G(n, nyms) = (sigmamap(i, j-1) +
sigmamap(i,j))/2;
            G(n,n) = -(G(n,nxps)+G(n,nxms)+G(n,nyms));
        else
            G(n,nxps) = (sigmamap(i+1, j) +
sigmamap(i,j))/2;
            G(n, nxms) = (sigmamap(i-1, j) +
sigmamap(i,j))/2;
            G(n, nyps) = (sigmamap(i, j+1) +
sigmamap(i,j))/2;
            G(n, nyms) = (sigmamap(i, j-1) +
sigmamap(i,j))/2;
```

```
            G(n,n) = -
(G(n,nxps)+G(n,nxms)+G(n,nyps)+G(n,nyms));
%             G(n,n) = -3;
%             if(i>leftEdge && i<rightEdge)
%                 G(n,nxms) = sigIn;
%                 G(n,nxps) = sigIn;
%                 G(n,nyms) = sigIn;
%                 sigmaMap(i,j) = sigIn;
%             else
%                 G(n,nxms) = sigOut;
%                 G(n,nxps) = sigOut;
%                 G(n,nyms) = sigOut;
%                 sigmaMap(i,j) = sigOut;
%             end
%         elseif (j == 1)
%             G(n,n) = -3;
%             if(i>leftEdge && i<rightEdge)
%                 G(n,nxms) = sigIn;
%                 G(n,nxps) = sigIn;
%                 G(n,nyps) = sigIn;
%                 sigmaMap(i,j) = sigIn;
%             else
%                 G(n,nxms) = sigOut;
%                 G(n,nxps) = sigOut;
%                 G(n,nyps) = sigOut;
%                 sigmaMap(i,j) = sigOut;
%             end
%         else
%             G(n,n) = -4;
%             if( (j>topEdge || j<bottomEdge) && i>leftEdge
&& i<rightEdge)
%                 G(n,nxps) = sigIn;
%                 G(n,nxms) = sigIn;
%                 G(n,nyps) = sigIn;
%                 G(n,nyms) = sigIn;
%                 sigmaMap(i,j) = sigIn;
%             else
%                 G(n,nxps) = sigOut;
%                 G(n,nxms) = sigOut;
%                 G(n,nyps) = sigOut;
%                 G(n,nyms) = sigOut;
%                 sigmaMap(i,j) = sigOut;
%             end
        end
```

```
    end
end

Voltage = G\Op';
sol = zeros(L, W, 1);

for i = 1:W
    for j = 1:L
        n = j + (i-1)*L;
        sol(j,i) = Voltage(n);
    end
end

%The electric field can be derived from the surface voltage
using a
%gradient
[Ey, Ex] = gradient(sol);
%J, the current density, is calculated by multiplying sigma
and the
%electric field together. Combing the x and y matrices, a
surface plot is
%derived by surfing this matrix.
J_x = sigmamap'.*Ey;
J_y = sigmamap'.*Ex;
J = sqrt(J_x.^2 + J_y.^2);

% Sigma(x,y) Surface Plot
% figure(9)
% subplot(2,1,1);
% surf(sigmamap);
% xlabel('x');
% ylabel('y');
% zlabel('V(x,y)')
% title('Sigma Charge Density Plot');

%subplot(2,1,2);
figure(1)
surf(sol)
xlabel("X position")
ylabel("Y position")
zlabel("Voltage")
title('Voltage Plot');

%X component of electric field surface plot
```

```
% figure(3)
% subplot(2,1,1);
% surf(-Ey)
% xlabel('x');
% ylabel('y');
% zlabel('V(x,y)')
% title('Electric Field Plot for x');

%Y component of electric field surface plot
% subplot(2,1,2);
% surf(-Ex)
% xlabel('x');
% ylabel('y');
% zlabel('V(x,y)')
% title('Electric Field Plot for y');

% figure(5)
% surf(J)
% xlabel('x');
% ylabel('y');
% zlabel('V(x,y)')
% title('Current Density ?(x, y)');

figure(2)
quiver(Ex,Ey);
xlabel('x');
ylabel('y');
title("2-D plot")
```

Coupled simulations.

a)

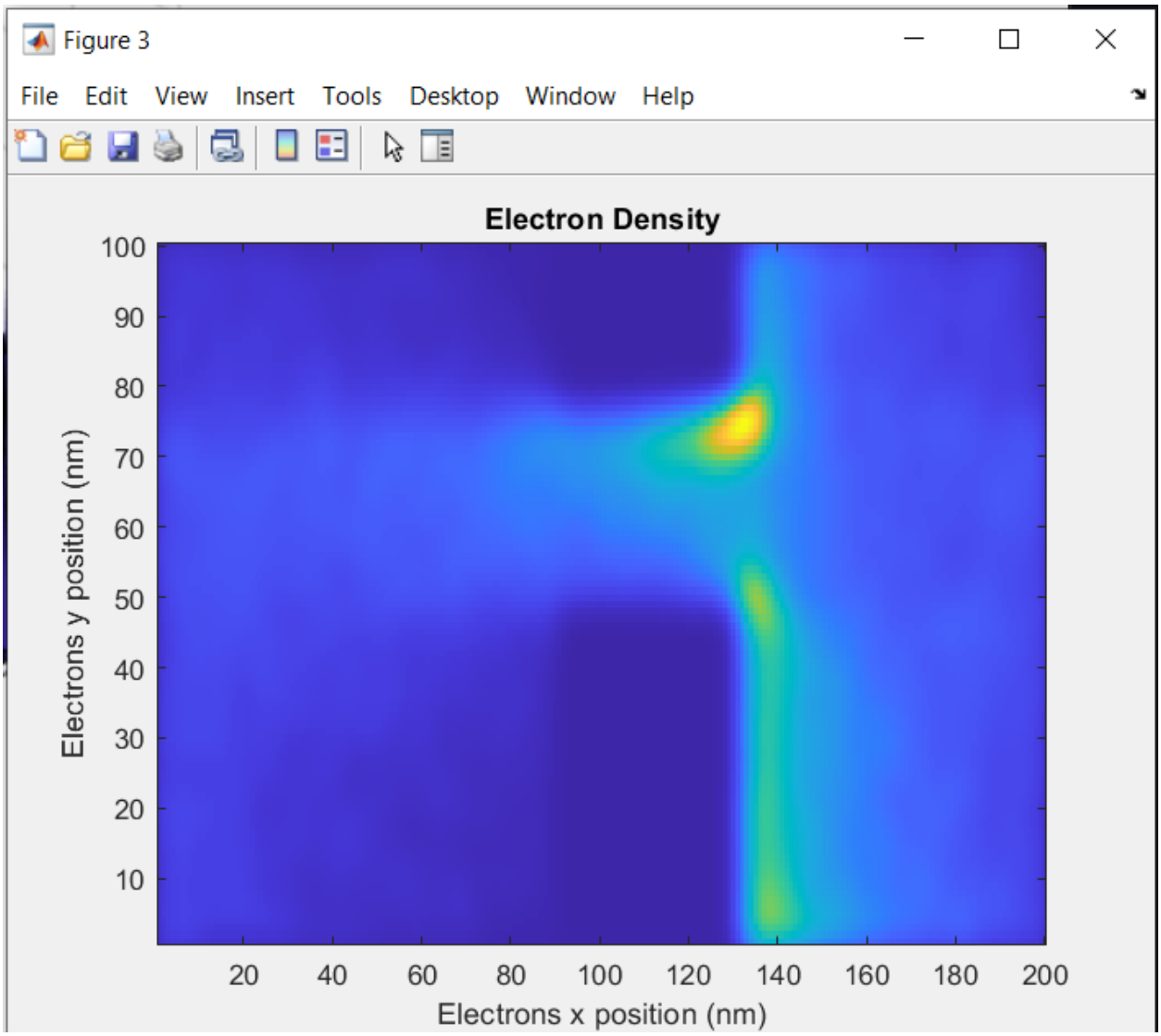


Figure 7: Density plot.

There seems to be more electrons where the voltage is applied. Now all the electrons drift through with is a characteristic of a resistance model.

b)

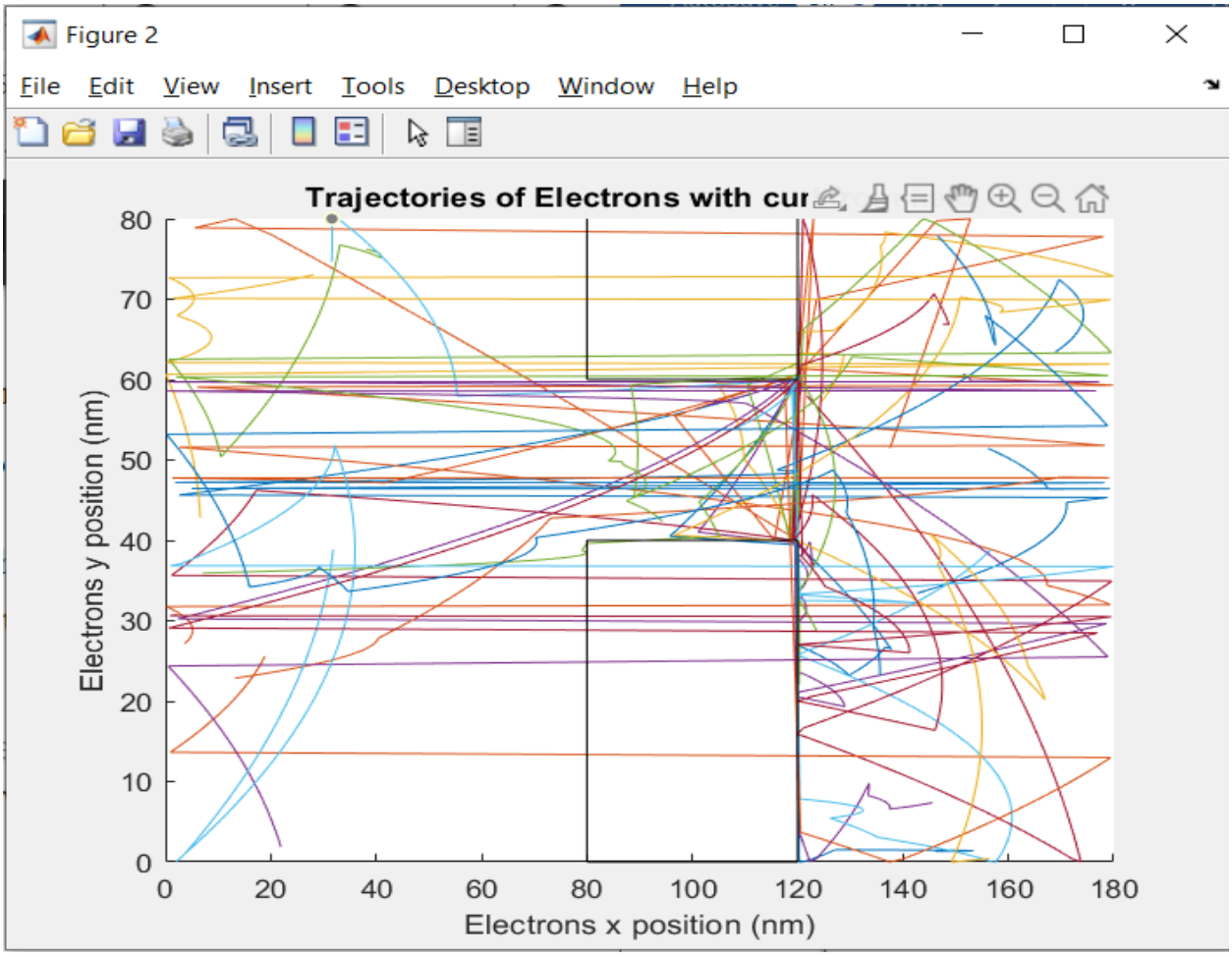


Figure 7: Plot electron trajectory.

c)

Best way to increase the accuracy of this simulation is to increase the amount of time and number of electrons. In other words, is to increase the mesh of the matrix. Getting higher resolutions into the region will make the simulation more accurate.

Code Used for Q3:

```
%Assign 3
%q3
%editting coupled
close all;
clc
%Kwabena Gyasi Bawuah
%101048814
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
```

```
%electron spec
 global C

    addpath ../geom2d/geom2d

    C.q_0 = 1.60217653e-19;              % electron charge
    C.hb = 1.054571596e-34;              % Dirac constant
    C.h = C.hb * 2 * pi;                     % Planck
constant
    C.m_0 = 9.10938215e-31;              % electron mass
    C.kb = 1.3806504e-23;                % Boltzmann
constant
    C.eps_0 = 8.854187817e-12;           % vacuum
permittivity
    C.mu_0 = 1.2566370614e-6;            % vacuum
permeability
    C.c = 299792458;                     % speed of light
    C.g = 9.80665; %metres (32.1740 ft) per s²

    T = 300;
    k = 1.38e-23;
    mn = 0.26*C.m_0; %effective mass
    tmn = 0.2e-12;    % Mean time between collisions

    vth = sqrt((2*C.kb*T)/mn);% Thermal velocity

    freepath = vth*tmn   % mean free path

    ConductorL = 180e-9;
    ConductorW = 80e-9;

    dpoints = 5e4;
    ecount = 15; %the number of electron to show on plot

    %electron concentratoin
    den = 1e15*100^-2;

    detaT= ConductorW/vth/100;
    sims = 200;

%     Xpos = rand(1,ecount).*ConductorW;
%     Ypos = rand(1,ecount).*ConductorL;
    traj=zeros(sims,ecount*2);
    temp=zeros(sims, 1);
```

```
    temp(:,1)= 300;


    MFP = vth*0.2e-12;

    Vx = 0.8;
    Vy = 0;
    dens = 1e15*100^-2;

    Ex = Vx/ConductorL
    Ey = Vy/ConductorW

    Fx = -C.q_0*Ex
    Fy = -C.q_0*Ey

    dVx = Fx*detaT/mn;
    dVy = Fy*detaT/mn;
    dVx = dVx.*ones(dpoints,1);
    dVy = dVy.*ones(dpoints,1);

    Pscat = 1-exp(-detaT/tmn);
    %---------------------------------------
    ProbDistr = makedist('Normal','mu', 0, 'sigma',
sqrt(C.kb*T/mn));

    tspec = 0;
    bspec=0;


    boxes = 1e-9.*[80 120 0 40; 80 120 60 100];
    specularbox = [0 1];

    for i = 1: dpoints
        angle = rand*2*3.14;
        state(i,:)= [ConductorL*rand ConductorW*rand
random(ProbDistr) random(ProbDistr)];

        if (state(i,2)>60e-9 &(state(i,1)>80e-9 &
state(i,1)<120e-9))  | (state(i,2)< 40e-9 &(state(i,1)>80e-
9 & state(i,1)<120e-9))
            state(i,1:2) = [ConductorL*rand
ConductorW*rand];
        end
```

```
    end

%     %take plot of loop to stop the refresh
% figure(5);
% subplot(4,1,2);
% %         plot(detaT*(0:i-1), temp(1:i));
% %         plot(detaT*(0:i-1),temp(1:i));
% tPlot = animatedline;
% xlabel('time(s)');
% ylabel('Temperature (K)');
% title('Temperature of semiconductor over time');
%
% figure(5);
% subplot(4,1,3);
% %part = sqrt(state(:,3).^2 + state(:,4).^2);
% % xlim([0 7e5]);
% % ylim([0 2000]);
% %histogram(part);
% currentPlot = animatedline ;
% xlabel('t(s)');
% ylabel('Current density');
% title('Drift');

    for i = 1:sims

        state(:,3) = state(:,3) + dVx;
        state(:,4) = state(:,4) + dVy;
        state(:,1:2) = state(:,1:2) + detaT.*state(:,3:4);

        out = state(:,1) > ConductorL;
        state(out,1) = state(out,1) - ConductorL;

        out = state(:,1) < 0;
        state(out,1) = state(out,1) + ConductorL;

        out = state(:,2) > ConductorW;

    if(tspec)
        state(out,2) = 2*ConductorW - state(out,2);
        state(out,4) = -state(out,4);
    else
        state(out,2) = ConductorW;
        part = sqrt(state(out,3).^2 + state(out,4).^2);
        angle = rand([sum(out),1])*2*3.14;
```

```
        state(out,3) = part.*cos(angle);
        state(out,4) = -abs(part.*sin(angle));
    end

    out = state(:,2) < 0;

    if(bspec)
        state(out,2) = -state(out,2);
        state(out,4) = -state(out,4);
    else
        state(out,2) = 0;
        part = sqrt(state(out,3).^2 + state(out,4).^2);
        angle = rand([sum(out),1])*2*3.41;
        state(out,3) = part.*cos(angle);
        state(out,4) = abs(part.*sin(angle));
    end
    for out=1: dpoints
            if (state(out,2)>60e-9 &(state(out,1)>80e-9 &
state(out,1)<120e-9))
                boxNum = 1;
            elseif (state(out,2)< 40e-9 &(state(out,1)>80e-
9 & state(out,1)<120e-9))
                boxNum = 2;
            else
                boxNum = 0;
            end
            while(boxNum ~= 0)
                XDist = 0;
                newx = 0;
                if(state(out,3) > 0)
                    XDist = state(out,1) - boxes(boxNum,1);
                    newx = boxes(boxNum,1);
                else
                    XDist = boxes(boxNum,2) - state(out,1);
                    newx = boxes(boxNum,2);
                end
                yDist = 0;
                newy = 0;
                if(state(out,4) > 0)
                    yDist = state(out,2) - boxes(boxNum,
3);
                    newy = boxes(boxNum, 3);
                else
```

```
                    yDist = boxes(boxNum, 4) -
state(out,2);
                    newy = boxes(boxNum, 4);
                end

                if(XDist < yDist)
                    state(out,1) = newx;
                    if(~specularbox(boxNum))
                        sgn = -sign(state(out,3));
                        part = sqrt(state(out,3).^2 +
state(out,4).^2);
                        angle = rand()*2*3.14;
                        state(out,3) =
sgn.*abs(part.*cos(angle));
                        state(out,4) = part.*sin(angle);
                    else
                        state(out,3) = -state(out,3);
                    end
                else
                    state(out,2) = newy;
                    if(~specularbox(boxNum))
                        sgn = -sign(state(out,4));
                        part = sqrt(state(out,3).^2 +
state(out,4).^2);
                        angle = rand()*2*3.14;
                        state(out,3) = part.*cos(angle);
                        state(out,4) =
sgn.*abs(part.*sin(angle));
                    else
                        state(out,4) = -state(out,4);
                    end
                end
                 boxNum = 0;
            end
    end

        out = rand(dpoints, 1) < Pscat;
        state(out,3:4) = random(ProbDistr, [sum(out),2]);

        temp(i) = (sum(state(:,3).^2) +
sum(state(:,4).^2))*mn/k/2/dpoints;


        for out=1:ecount
```

```
            traj(i, (2*out):(2*out+1)) = state(out,1:2);
        end

        %getting j using density
        %add the points

        if  mod(i,5) == 0
        figure(5);
        hold off;
        plot(state(1:ecount,1)./1e-9,
state(1:ecount,2)./1e-9, 'o');
        hold on;

        for ouut=1:size(boxes,1)
           plot([boxes(ouut, 1) boxes(ouut, 1) boxes(ouut,
2) boxes(ouut, 2) boxes(ouut, 1)]./1e-9,[boxes(ouut, 3)
boxes(ouut, 4) boxes(ouut, 4) boxes(ouut, 3) boxes(ouut,
3)]./1e-9, 'k-');
        end

        xlim([0 ConductorL/1e-9])
        ylim([0 ConductorW/1e-9])
        xlabel('Electrons x position (nm)')
        ylabel('Electrons y position (nm)')
        title('Electrons Simulations')

%         figure(6)
%         part = sqrt(state(:,3).^2 + state(:,4).^2);
%         xlim([0 7e5]);
%         ylim([0 2000]);
%         histogram(part);
%         xlabel('v(m/s)');
%         ylabel('Particle count');
%         title('Histogram to show particle speed');

        J(i, 1) = -C.q_0.*dens.*mean(state(:,3));
        J(i, 2) = -C.q_0.*dens.*mean(state(:,4))

        %not needed
%         addpoints(tPlot, detaT.*i, temp(i));
%         addpoints(currentPlot, detaT.*i, J(i,1));
        end
    end
```

```
    figure(2)
    hold on;
    xlim([0 ConductorL/1e-9]);
    ylim([0 ConductorW/1e-9]);
    xlabel('Electrons x position (nm)');
    ylabel('Electrons y position (nm)');
    title('Trajectories of Electrons with curl from
field');

    for i = 1: ecount
        plot(traj(:,i*2)./1e-9, traj(:,i*2+1)./1e-9, '-');
    end

   for out=1:size(boxes,1)
   plot([boxes(out, 1) boxes(out, 1) boxes(out, 2)
boxes(out, 2) boxes(out, 1)]./1e-9,...
       [boxes(out, 3) boxes(out, 4) boxes(out, 4)
boxes(out, 3) boxes(out, 3)]./1e-9, 'k-');
    end

%     figure(7)
%     part = sqrt(state(:,3).^2 + state(:,4).^2);
%     xlim([0 7e5]);
%     ylim([0 2000]);
%     histogram(part);
%     xlabel('v(m/s)');
%     ylabel('Particle count');
%     title('Histogram to show particle speed');

    dens = hist3(state(:,1:2),[200 100])';

    N = 20;
    sigma = 3;
    [x y]=meshgrid(round(-N/2):round(N/2), round(-
N/2):round(N/2));
    f=exp(-x.^2/(2*sigma^2)-y.^2/(2*sigma^2));
    f=f./sum(f(:));
    figure(3);
    imagesc(conv2(dens,f,'same'));
    set(gca,'YDir','normal');
    title('Electron Density');
    xlabel('Electrons x position (nm)');
    ylabel('Electrons y position (nm)');
```

```
%     tempSumX = zeros(ceil(ConductorL/1e-
9),ceil(ConductorW/1e-9));
%     tempSumY = zeros(ceil(ConductorL/1e-
9),ceil(ConductorW/1e-9));
%     tempSum = zeros(ceil(ConductorL/1e-
9),ceil(ConductorW/1e-9));
%
%     for i=1:dpoints
%
%         x = floor(state(i,1)/1e-9);
%         y = floor(state(i,2)/1e-9);
%         if(x==0)
%             x = 1;
%         end
%         if(y==0)
%             y= 1;
%         end
%
%         tempSumY(x,y) = tempSumY(x,y) + state(i,3)^2;
%         tempSumX(x,y) = tempSumX(x,y) + state(i,4)^2;
%         tempSum(x,y) = tempSum(x,y) + 1;
%     end
%
%     temp = (tempSumX + tempSumY).*mn./k./2./tempSum;
%     temp(isnan(temp)) = 0;
%     temp = temp';
%
%     N = 20;
%     sigma = 3;
%     [x y]=meshgrid(round(-N/2):round(N/2), round(-
N/2):round(N/2));
%     f=exp(-x.^2/(2*sigma^2)-y.^2/(2*sigma^2));
%     f=f./sum(f(:));
%     figure(9);
%     surf(conv2(temp,f,'same'));
%     %set(gca,'YDir','normal');
%     title('Map of temperature');
%     xlabel('Electrons x position (nm)');
%     ylabel('Electrons y position (nm)');
```

Conclusion:

The Finite Difference Method was used to solve for the current flow in certain regions. The results from observations to be as expected. All questions where also as answered with the code used provided in the sections. The codes can be put together in a MATLAB file and run for a complete simulation of the system. Problem with the boxes was unable to be fixed. This will keep being worked on.