Carleton University

Modelling of Integrated Devices

ELEC 4700

Assignment 3

Author: Maharshi Gurjar
Student number: 101094330
Date submitted: March 18, 2021

Contents

List of Figures

List of Source Codes

1 Monte Carlo Simulation

1.1 Constant applied voltage

Applying a 0.1V applied voltage across the x dimension of the semiconductor region, in the Monte Carlo simulation tool results in Figure 4 below. Plotting the trajectories of the electrons shows the effects of the constant applied voltage, where the path of the electron curves. This figure also shows that without the presence of any objects in the regions, the drift current will eventually stabilize at a value.

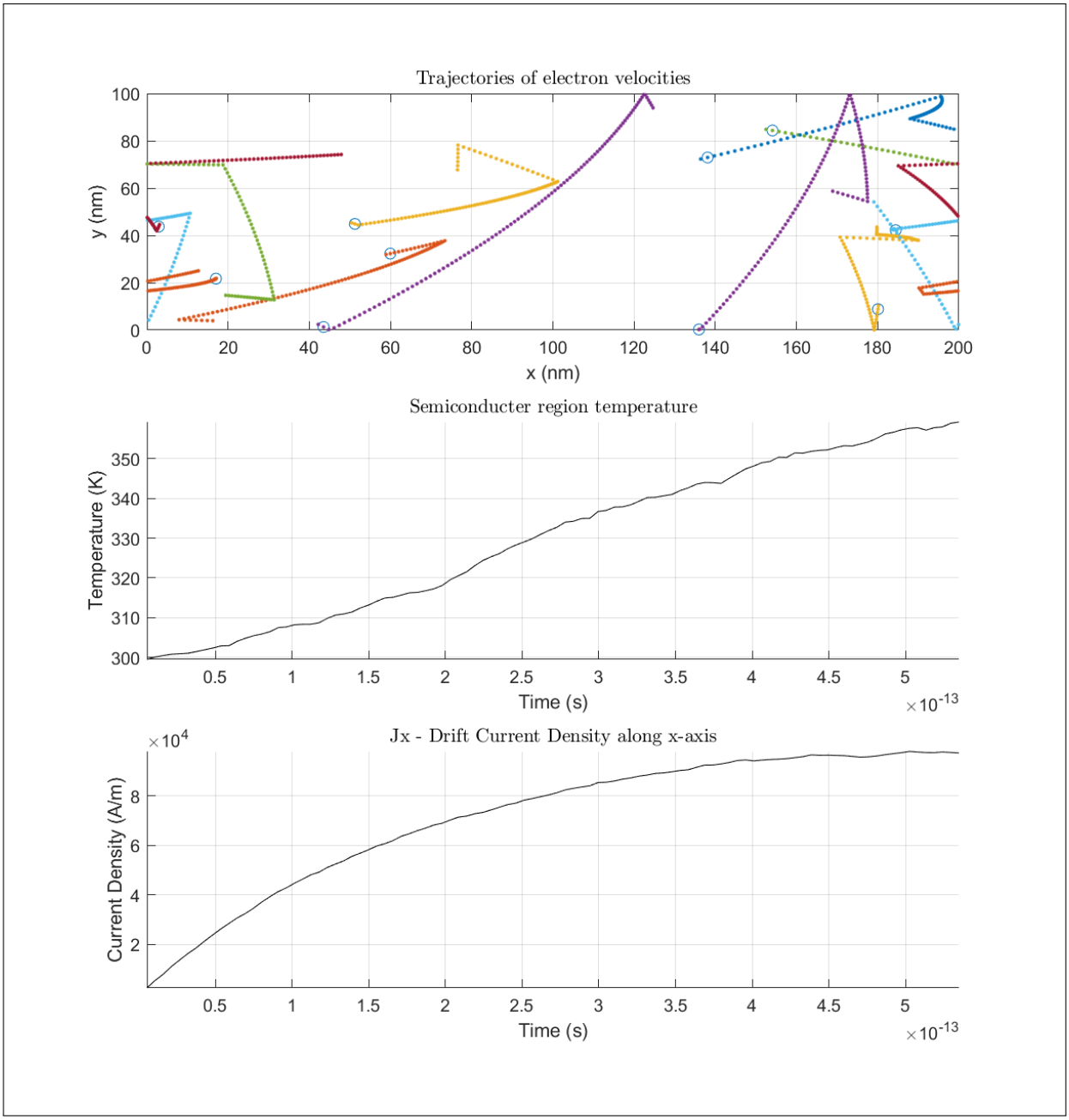

Figure 1: Monte Carlo Simulation

Force on electron The simulation was ran with a constant electric field applied only in the x-axis, therefore the force on an individual electron can be calculated as,

$$F_{ex} = Q_e * E_x = \boxed{-8.0109e^{-14}} \tag{1}$$

1.2 Electron Density

Figure 2 below shows the surface plot of the electron density, which shows the expected distribution of the densities. Without the presence of a bottleneck the electrons are approximately even distribution across the semiconductor region.

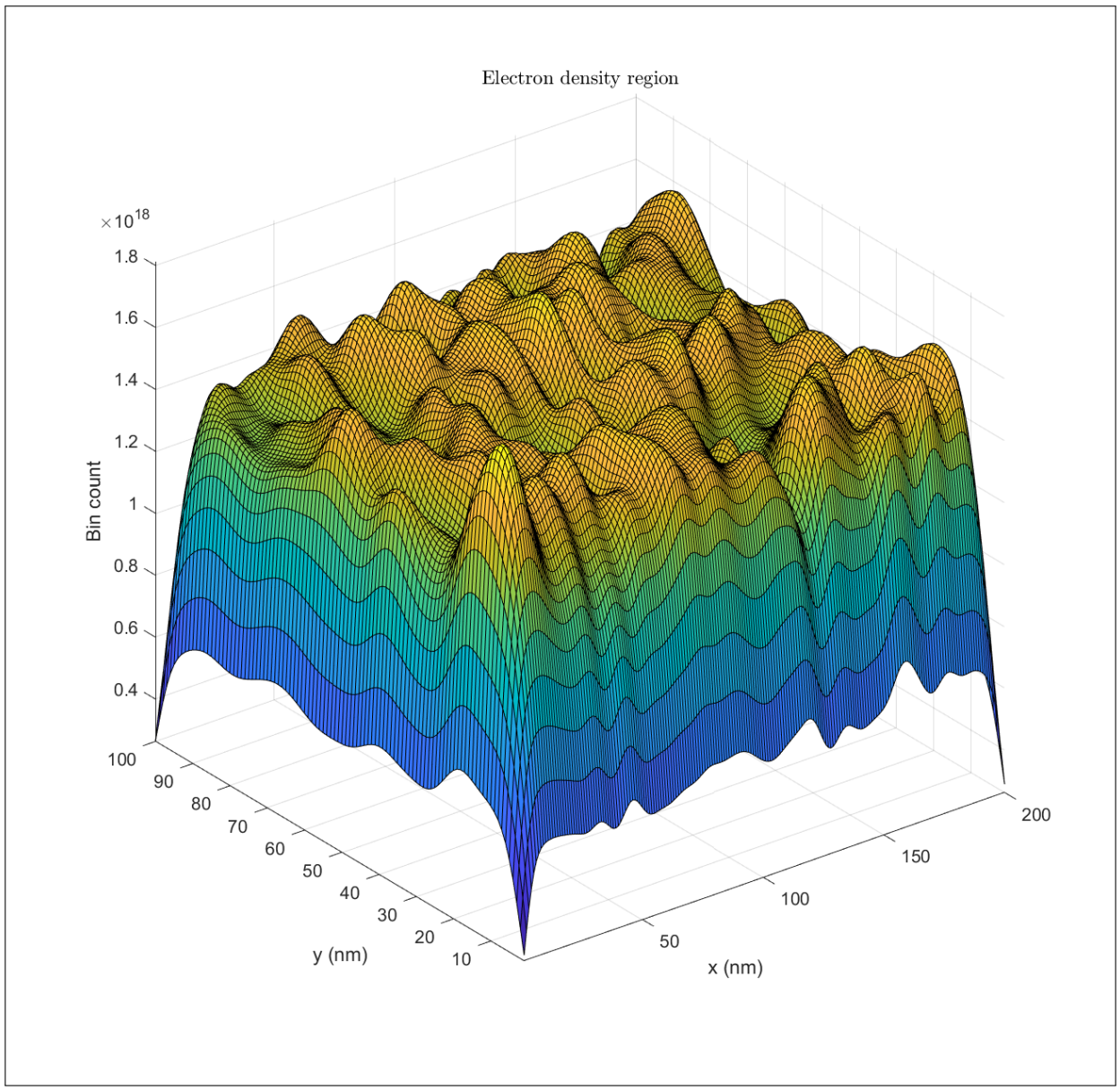

Figure 2: Electron Density at constant applied voltage

1.3 Electron Temperature

Figure 3 below shows the surface plot of the temperature in the region, which once again shows the excepted result.

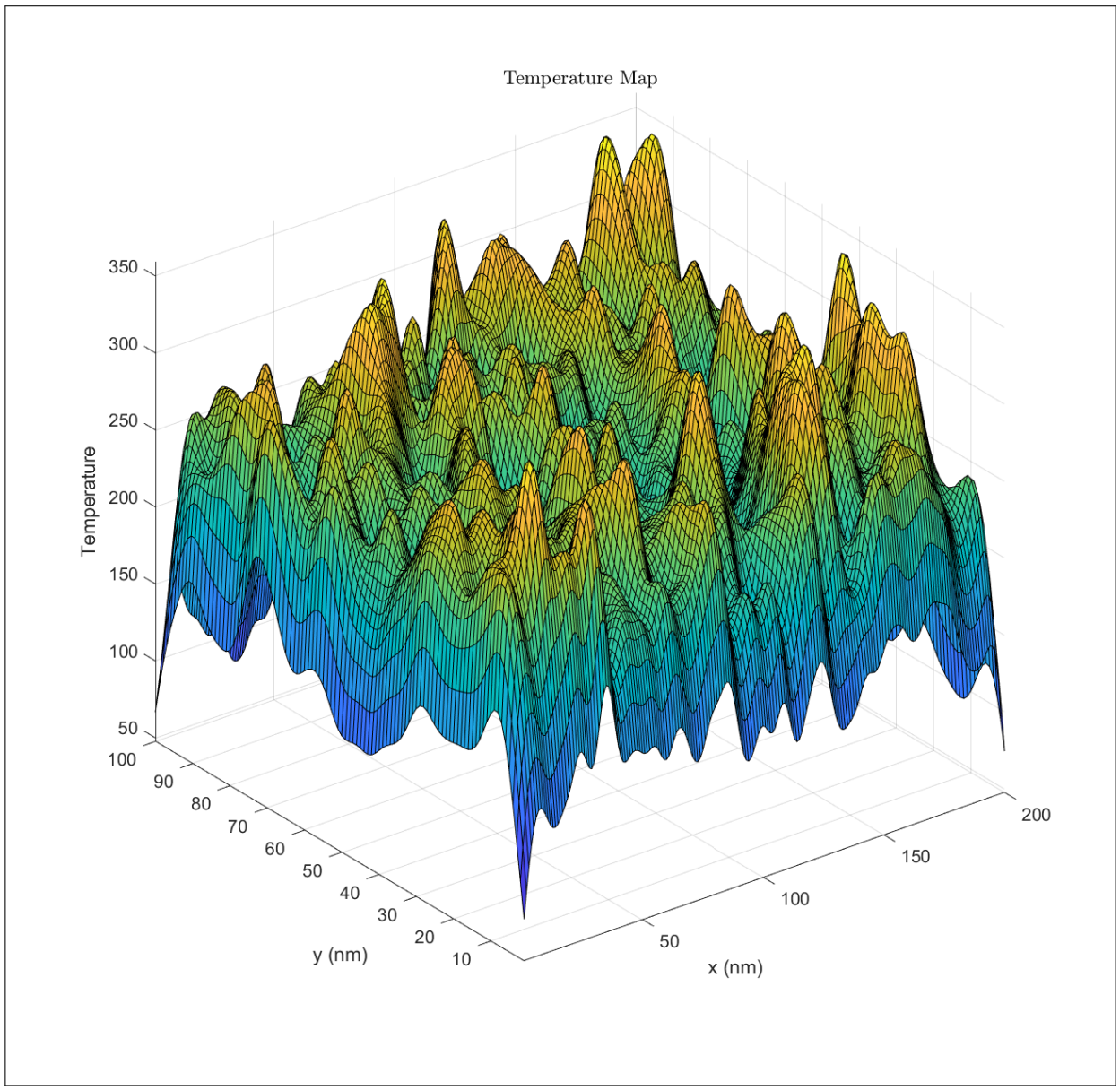

Figure 3: Electron temperature map at constant applied voltage

2 Finite Difference Method

2.1 Bottleneck

Introducing a bottleneck in the simulation, as shown below in Figure 2, the region is now simulated using the Finite Difference Method. This figure shows a blue region that is insulate (much lower sigma) then the rest of the region (yellow). This will create a bottleneck as the electrons cannot move into the bottleneck region.

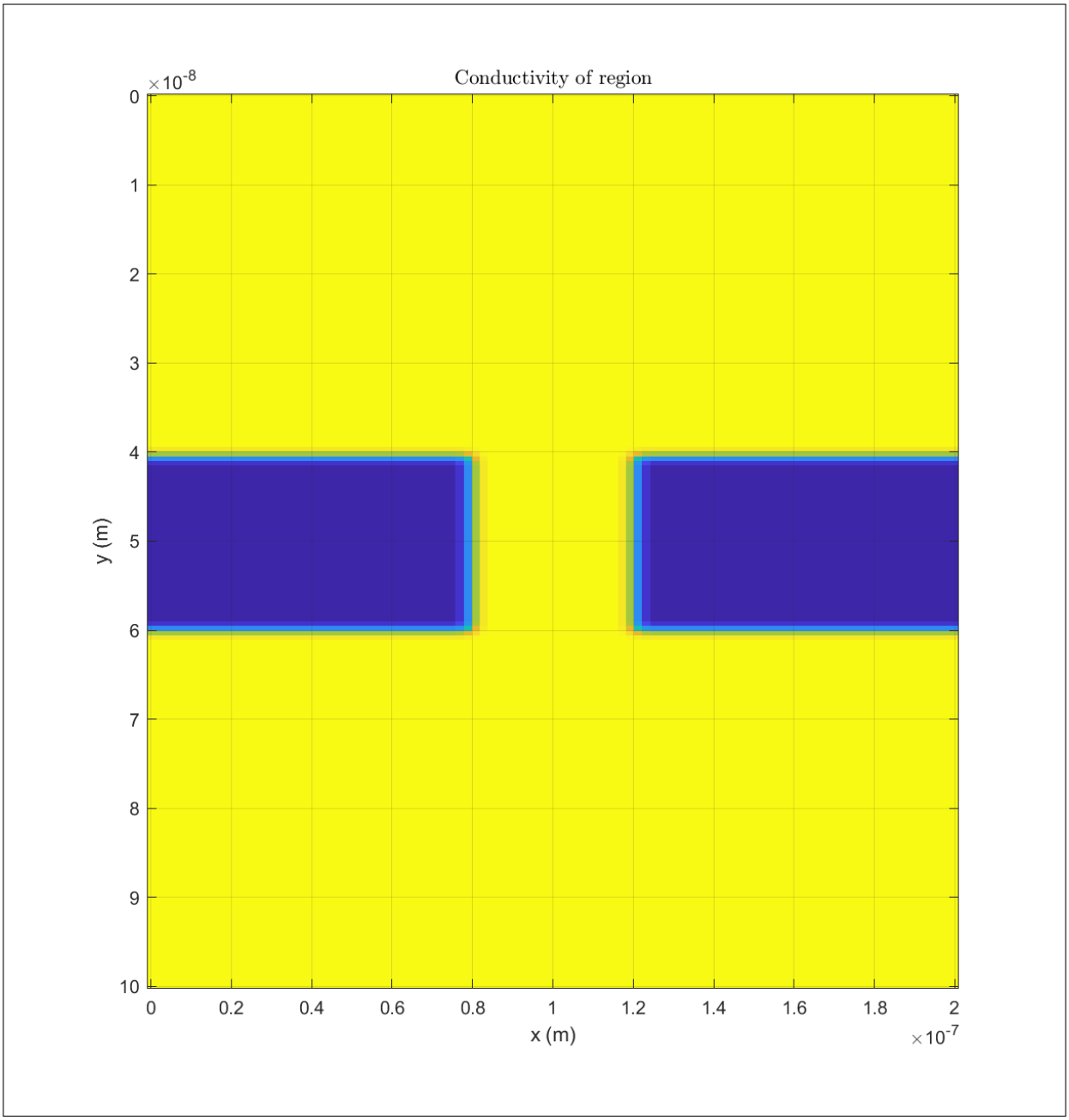

Figure 4: Conductivity map of region

2.2 Voltage Map

Figure 2 below shows the voltage map across the region including a bottleneck. This shows a result as excepted, where due to the bottleneck there is a non-linear element at the position of the bottleneck.

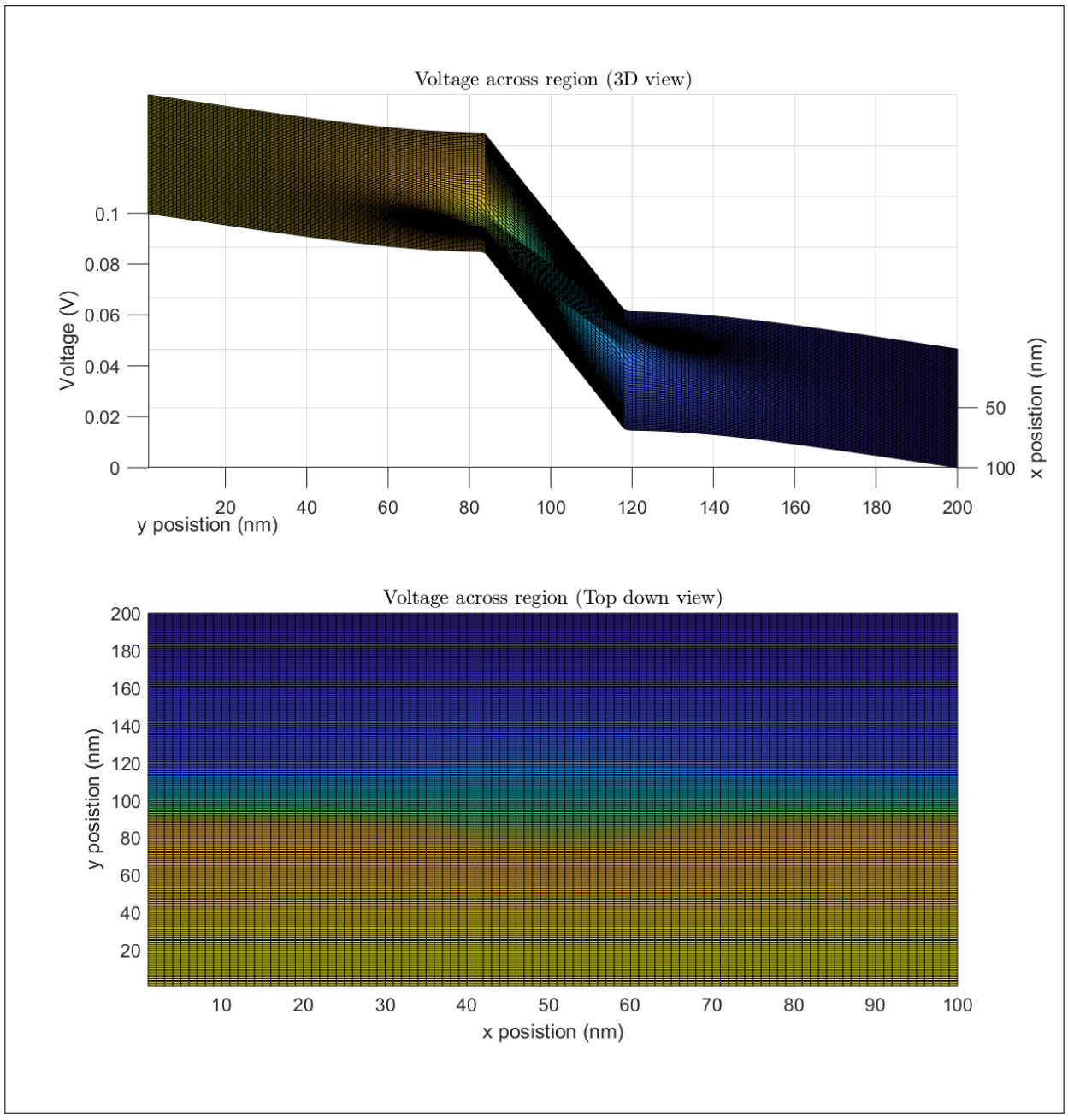

Figure 5: Voltage map using Finite Difference Method

2.3 Quiver plot

Figure 6 below shows the quiver plot across the semiconductor region including a bottleneck. This shows expected results, where the region inside of the bottle neck causes a disturbance in the linearity of the region.

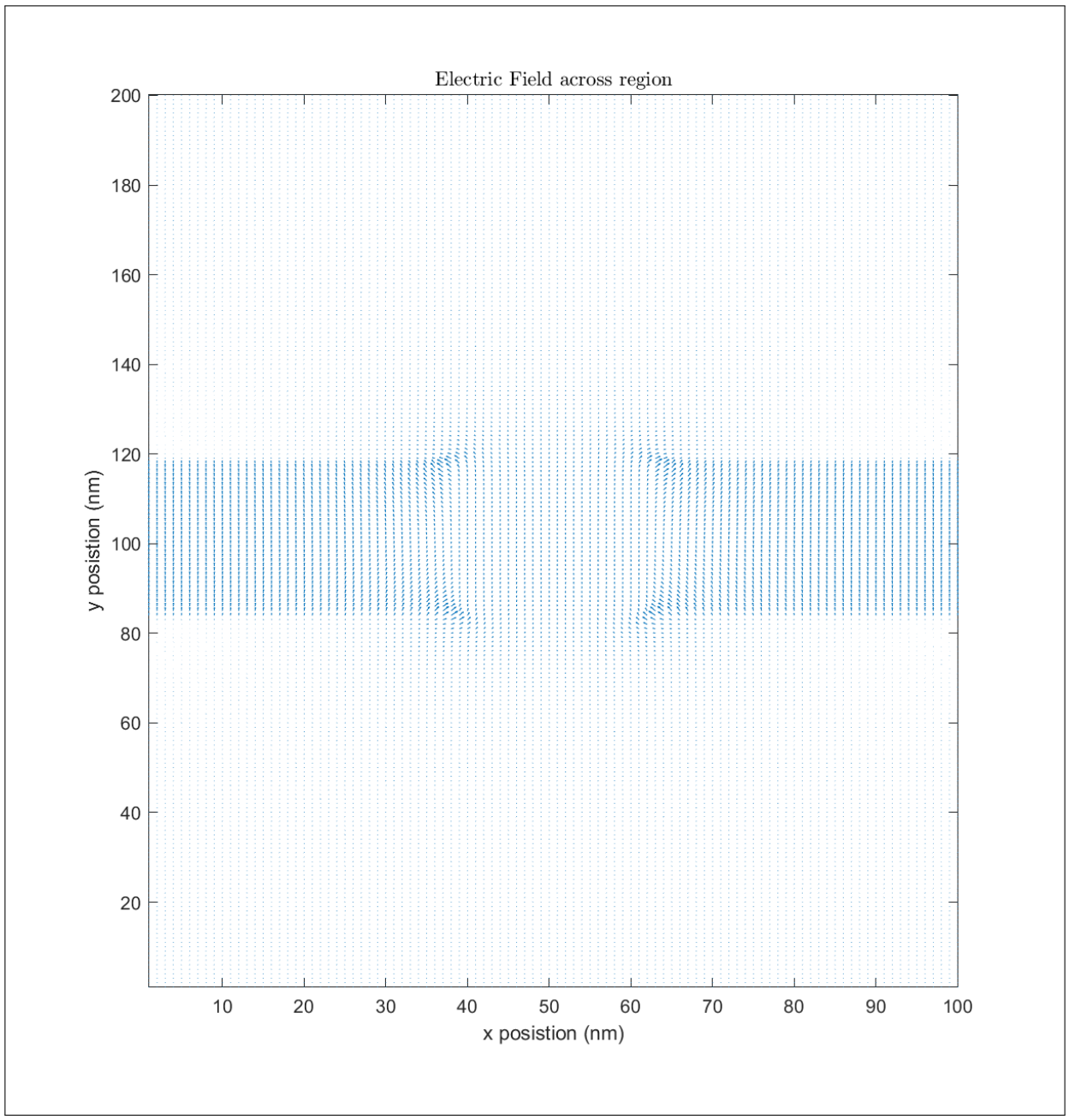

Figure 6: Quiver plot using Finite Difference Method

3 Coupled Simulations

Figure 7 below shows the results of the coupled simulation, where a constant voltage field is applied across the semiconductor region with the inclusion of a bottleneck. These results confirm the proper functionality of the simulator, where the "bending" of the electron path can be observed, as well as the "blocking" of the bottleneck.

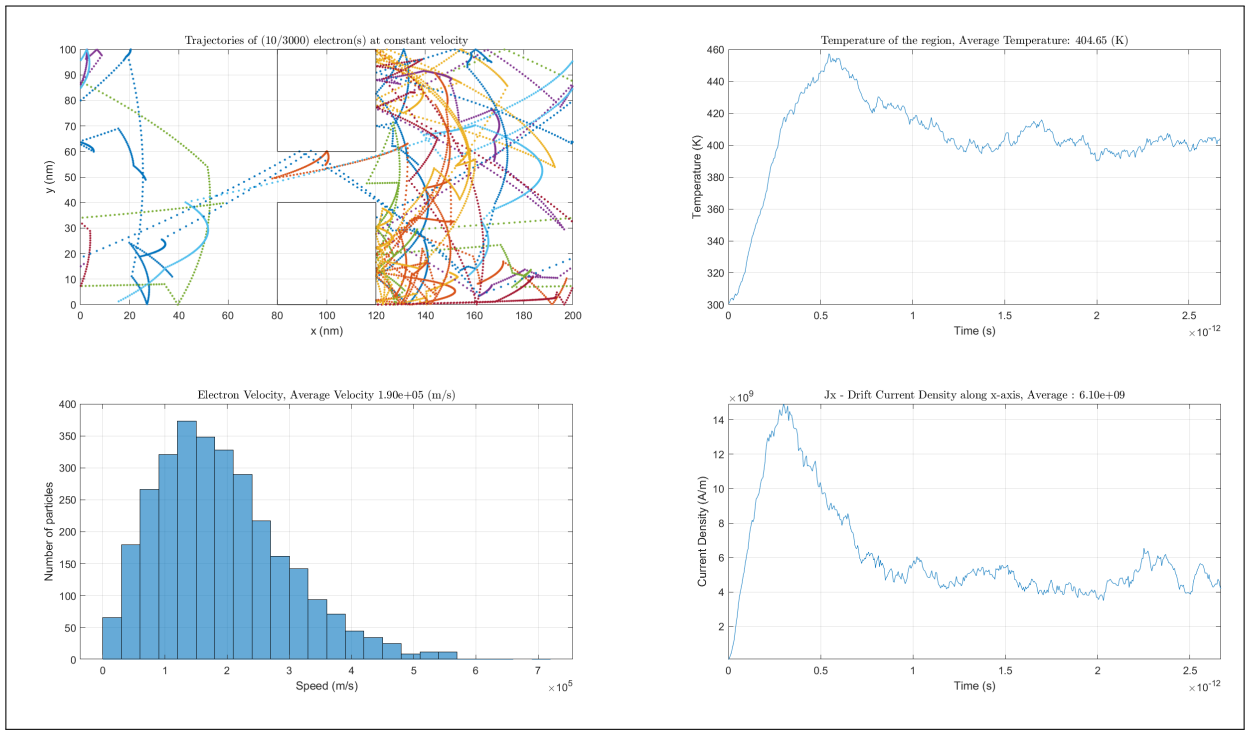

Figure 7: Coupled Simulation results

3.1 Electron Density

Figure 8 below shows the surface plot of the electron density in the semiconductor region with the bottleneck. This clearly shows the effect of the bottleneck and applied voltage, where a large concentration of electrons are stuck on one side of the bottleneck.

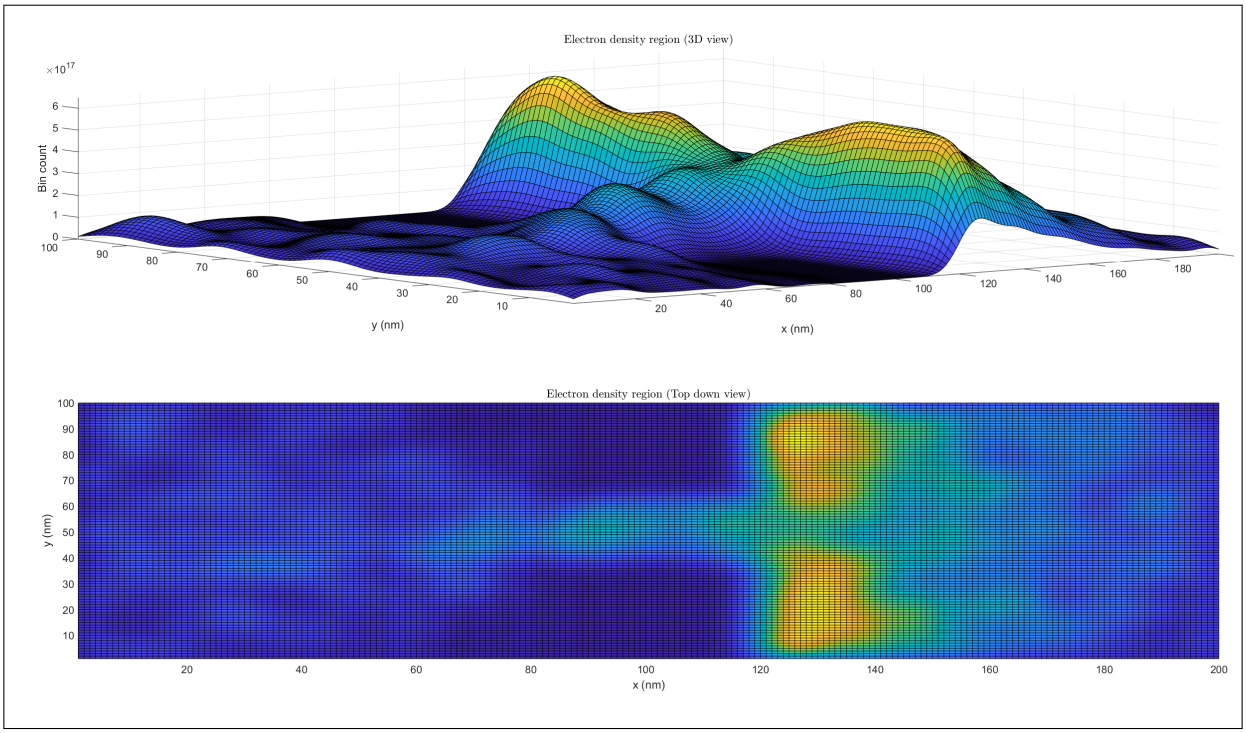

Figure 8: Electron Density using Coupled Simulations

3.2 Effect of bottleneck on drift current

Figure 9 below shows the plotted data from changing the bottleneck width, and its affects on the current density. This results makes sense, as the increased gap size allows for the easier flow of electrons, and the same visa-versa (smaller gap $\implies$ small J_x).

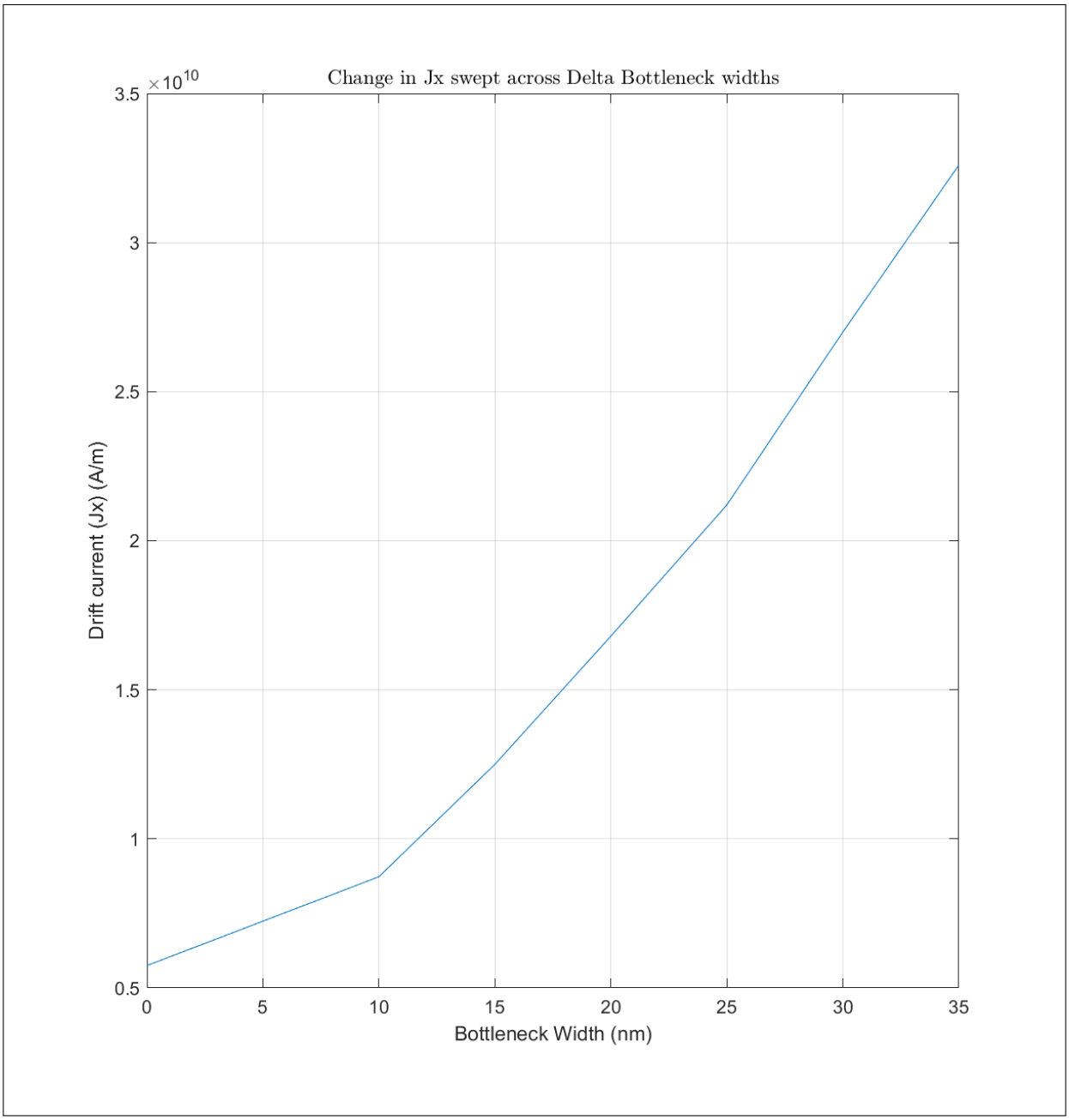

Figure 9: ΔJ_x with varying bottleneck width

3.3 Improvements/Next Step

Improvements to the accuracy of the simulator can be made by adding a third dimension (x,y, and z), increasing the number of iterations, mesh accuracy (smaller mesh size), number of electrons simulated.

4 Appendix

4.1 Part 1 Code

```
%{
% Author: Maharshi Gurjar
% ELEC 4700 - Modeling of Integrated Devices
% Assignment 3
% Part 1
%}
%%
clc; close all; clear;
set(0, 'DefaultFigureWindowStyle', 'docked')
%Define simulation envrionment and constants
M0 = 9.10938356e-31; %Rest mass of electron
Mass_n = 0.26*M0; %Effective mass of electron
Qe = -1.60217662e-19; %Electron Charge
Vx = 0.1; % Voltage along the x-axis
Vy = 0; %Voltage along the y-axis
EConcentration = 1e15 * 100^2; %Electron concentration (1/m^2)
T = 300; % Simulation envrionment temperature (K)
k = 1.38064852e-23; % Boltzmans constant
V_thermal = sqrt(2*k*T/Mass_n); %Thermal Veleocity
Height = 100e-9; % The height of the simulation environment
Length = 200e-9; % The lengthof the simulation environment
nElectrons = 30e3; % Total number of electrons to simulate
nPlotted_Electrons = 10; %Total number of electrons displayed
Time_Step = Height/V_thermal/100; % Time step of simulation
Iterations = 1000; % Number of iternations to simulate
Show_Movie = 1; %Display steps control
% The mean free path is determined by multipling the thermal velocity
... by the mean time between collisions:
MFP = V_thermal * 0.2e-12 %Mean free path
%The state of the electron  (postion and velocity) is stored in a array
... where each index refers to [x-position y-position v-in-x v-in-y]
Electron_State = zeros(nElectrons,4);
%Setting the top/bottom of the boxes specularity
Top_Specular = 1;
Bottom_Specular = 1;
%Temperature will be recorded in the array below
Temperature = zeros(Iterations,1);
%Create a scattering probability
P_Scatterieng =1 - exp(-Time_Step/0.2e-12);
%Create a distribution using the matlab makedist function
Velocity_PDF = makedist('Normal', 'mu', 0, 'sigma', sqrt(k*T/Mass_n));
%Generate a random inital population postion and velocity
%%
% The electric field compoennts (assuming uniform fields) given by:
Ex = Vx/Length;
Ey = Vy/Height;
%%
% Force on individual force given by:
Fx = Qe*Ex
Fy = Qe*Ey
%%
Temperature = zeros(Iterations,1);
J = zeros(Iterations,2);
%%
for i = 1:nElectrons
   Electron_State(i,:) = [Length*rand() Height*rand() random(Velocity_PDF) random(Velocity_PDF)];
end
%%
figure(1);
```

```
subplot(3,1,1)
plot([],[]);
axis ([0 Length/1e-9 0 Height/1e-9]);
title("Trajectories of electron velocities",'interpreter','latex');
xlabel('x (nm)');
ylabel('y (nm)');
grid on;
subplot(3,1,2)
Temp_Plot = animatedline;
title("Semiconducter region temperature",'interpreter','latex');
xlabel('Time (s)');
ylabel('Temperature (K)');
grid on;
axis tight;
subplot(3,1,3)
Current_Plot = animatedline;
title("Jx - Drift Current Density along x-axis",'interpreter','latex');
xlabel('Time (s)');
ylabel('Current Density (A/m)');
grid on;
axis tight;
%We will now move (iterate) over time, updating the positions and direction
...while plotting the state
%%
for i = 1:Iterations
    %%
    %The line below updates Vx and Vy by calculating the change in
    %veloctity under the influence of the electric field
    Electron_State(:,3) = Electron_State(:,3) + Fx*Time_Step/Mass_n;
    Electron_State(:,4) = Electron_State(:,4) + Fy*Time_Step/Mass_n;

    %The line below updates the x,y position by moving it to a new position
    ... using its current position + the velocity*(time step)
    Electron_State(:,1:2) = Electron_State(:,1:2) + Time_Step.*Electron_State(:,3:4);

    %Checking boundary conditions using Matlab matrix equations

    %Check if and move all electrons at X=200nm Bound:
    Electron_State((Electron_State(:,1)>Length),1) = Electron_State((Electron_State(:,1)>Length),1)
    ↪  - Length;

    %Check if and move all electrons at X=0nm Bound:
    Electron_State((Electron_State(:,1)<0),1) =Electron_State((Electron_State(:,1)<0),1) + Length;

    %Check (if) and move all electrons at Y Bounds and if specular or diffusive:
    if (Top_Specular == 1)
       Electron_State((Electron_State(:,2)>Height),4) =
       ↪  -1*Electron_State((Electron_State(:,2)>Height),4) ;
       Electron_State((Electron_State(:,2)>Height),2) = 2*Height -
       ↪  Electron_State((Electron_State(:,2)>Height),2);
    else
       %Electron_State((Electron_State(:,2)>Height),2) = Height;
       Electron_State((Electron_State(:,2)>Height),4) = -random(Velocity_PDF);
       Electron_State((Electron_State(:,2)>Height),3) = random(Velocity_PDF);
    end
    if (Bottom_Specular == 1)
        Electron_State((Electron_State(:,2)<0),4) = -1*Electron_State((Electron_State(:,2)<0),4) ;
        Electron_State((Electron_State(:,2)<0),2) = -Electron_State((Electron_State(:,2)<0),2);
    else
       %Electron_State((Electron_State(:,2)<0),2) = 0;
       Electron_State((Electron_State(:,2)<0),4) = random(Velocity_PDF);
       Electron_State((Electron_State(:,2)<0),3) = random(Velocity_PDF);
    end
    %%
```

```
121     %Add scattering
122      j = rand(nElectrons,1) < P_Scatterieng;
123     Electron_State(j,3:4) = random(Velocity_PDF,[sum(j),2]);
124
125     % Stores the Electron [x y] posistions in the Trajectories vector
126     ... for each different electron in a new coloum
127     for j = 1: nPlotted_Electrons
128        Trajectories_x(i,j) = Electron_State(j,1);
129        Trajectories_y(i,j) = Electron_State(j,2);
130     end
131     %To calcuatle the themal energy, Maxwell's principle of equipartion
132     ... is used,  where the final equation then becomes;
133     Temperature(i) = ( sum (Electron_State(:,3).^2) + sum(Electron_State(:,4).^2)) * Mass_n / k / 2
        ↪  / nElectrons;
134
135     %To calculate the current density
136     J(i,1) = Qe.*EConcentration.*mean(Electron_State(:,3));
137     J(i,2) = Qe.*EConcentration.*mean(Electron_State(:,4));
138     % Add the temperature and current data to the respective plots using
139     % addpoints command
140     addpoints(Temp_Plot,Time_Step.*i, Temperature(i));
141     addpoints(Current_Plot,Time_Step.*i,J(i,1));
142
143     %Shows the pathing of the electron, as well as the updating trajectory
144     if Show_Movie && mod(i,50)
145        figure(1)
146        subplot(3,1,1);
147        hold off;
148        plot(Electron_State(1:nPlotted_Electrons,1)./1e-9,Electron_State(1:nPlotted_Electrons,2)./1e
        ↪  -9,'o');
149        grid on;
150        axis([0 Length/1e-9 0 Height/1e-9]);
151        xlabel('x (nm)');
152        ylabel('y (nm)');
153        title("Trajectories of electron velocities",'interpreter','latex');
154        hold on;
155     end
156  end
157  %%
158  figure(1)
159  subplot(3,1,1)
160  hold on;
161  plot(Trajectories_x(:,1:nPlotted_Electrons)./1e-9, Trajectories_y(:,1:nPlotted_Electrons)./1e-9,'.'
     );
162  hold off;
163  axis([0 Length/1e-9 0 Height/1e-9]);
164  xlabel('x (nm)');
165  ylabel('y (nm)');
166  grid on;
167  title("Trajectories of electron velocities",'interpreter','latex');
168  %saveas(gcf,fullfile('D:\School Work\ELEC 4700\My 4700 Code\Assignment 3\Simulation
     ↪  Results','[Part1]TrajResults.png'),'png')
169
170  %%
171  Density = hist3(Electron_State(:,1:2),[200 100])';
172  N = 20;
173  sigma = 3;
174  %Creating a Gaussian filtering matrix
175  [x,y]=meshgrid(round(-N/2):round(N/2), round(-N/2):round(N/2));
176  G=exp(-x.^2/(2*sigma^2)-y.^2/(2*sigma^2));
177  G=G./sum(G(:));
178
179  figure("name","Electron Density (with Gaussian filtering)")
180  Density = conv2(Density,G,'same') / (Height./size(Density,1)*Length./size(Density,2));
```

```
surf(conv2(Density,G,'same'));
xlabel('x (nm)')
ylabel('y (nm)')
zlabel('Bin count')
axis tight
title("Electron density region",'interpreter','latex');
%saveas(gcf,fullfile('D:\School Work\ELEC 4700\My 4700 Code\Assignment 3\Simulation Results','[Part1]ElectronDensity.png'),'png')
%For the temperature density plot the electrons positions were binned
%Used the same process I did in the first assignment
Temperature_Sum_X = zeros(200,100);
Temperature_Sum_Y = zeros(200,100);
Temperature_Bin_Count = zeros(200,100);
for i=1:nElectrons
    x = floor(Electron_State(i,1)/1e-9);
    y = floor(Electron_State(i,2)/1e-9);
    if(x==0)
        x = 1;
    end
    if(y==0)
        y = 1;
    end
    Temperature_Sum_X(x,y) = Temperature_Sum_X(x,y) + Electron_State(i,3)^2;
    Temperature_Sum_Y(x,y) = Temperature_Sum_Y(x,y) + Electron_State(i,4)^2;
    Temperature_Bin_Count(x,y) = Temperature_Bin_Count(x,y) + 1;
end
% Now, with the velocities added up, calculate the temperatures:
TemperatureDensity = (Temperature_Sum_X + Temperature_Sum_Y).*Mass_n./k./2./Temperature_Bin_Count;
%If somewhere in the calculation the density beame nan then make it 0
TemperatureDensity(isnan(TemperatureDensity)) = 0;
%Transpose the matrix
TemperatureDensity = TemperatureDensity';
[x,y] = meshgrid(round(-N/2):round(N/2), round(-N/2):round(N/2));
G=exp(-x.^2/(2*sigma^2)-y.^2/(2*sigma^2));
G=G./sum(G(:));
figure("Name","Electron heat denstiy (with Guassian filtering)")
surf(conv2(TemperatureDensity,G,'same'))
set(gca,'YDir','normal');
title('Temperature Map','interpreter','latex');
xlabel('x (nm)');
ylabel('y (nm)');
zlabel('Temperature');
axis tight
%saveas(gcf,fullfile('D:\School Work\ELEC 4700\My 4700 Code\Assignment 3\Simulation Results','[Part1]ElectronHeatDensity.png'),'png')
```

Source Code 1: Part 1 - Monte Carlo Simulator

4.2 Part 2 Code

```
%{
Author : Maharshi Gurjar
Elec 4700 Assignment 3 -  Finite Difference Method
Part 2
%}
%%
clc; close all; clear;
set(0, 'DefaultFigureWindowStyle', 'docked')
%Define simulation parameters
W = 100e-9; %Width of region
L = 200e-9; %Length of region
L_Box = 40e-9; %Length of Box
W_Box = 40e-9; %Width of Box
MeshSize = 1e-9; % Preset mesh size
nx = ceil(L/MeshSize); %X bins
ny = ceil(W/MeshSize); %Y bins

Conductivity_1 = 1; %Conductivity of region
Conductitivty_2 = 1e-2; %Conductivity in box

Conductivity_Map = zeros(nx,ny);

for i = 1:nx
    for j = 1:ny
        if ((i-1)>0.5*(L-L_Box)/MeshSize && (i-1)<0.5*(L+L_Box)/MeshSize && ((j-1)<W_Box/MeshSize
        ↪  || (j-1)>(W-W_Box)/MeshSize))
            Conductivity_Map(i,j) = Conductitivty_2;
        else
            Conductivity_Map(i,j) = Conductivity_1;
        end
    end
end
% Numerial issues can happen if the derivatives are too large, thus using
% the imgaussfilt function the matrix is filtered
Conductivity_Map = imgaussfilt(Conductivity_Map,1);


figure(1);
imagesc(linspace(0,L,nx),linspace(0,W,ny),Conductivity_Map);
title('Conductivity of region','interpreter','latex')
xlabel('x (m)')
ylabel('y (m)')
view(2)
axis tight
grid on;
%saveas(gcf,fullfile('D:\School Work\ELEC 4700\My 4700 Code\Assignment 3\Simulation
↪  Results','[Part2]ConductivityMap.png'),'png')
G = sparse(ny*nx);
F = zeros(1,ny*nx);


for i = 1:(nx)
    for j = 1:(ny)
        %Create variable for node mapping
        n = j + (i-1)*ny;
        %Calculate changes in x and y
        if i == 1 % V = 1, x = 0
            G(n,n) = 1;
            F(n) = 0.1;
        elseif i == nx %V=0, x = L
            G(n,n) = 1;
```

```
283         elseif j == 1
284             xM = j + (i-2)*ny;
285             xP = j + i*ny;
286             yP = j+1 + (i-1)*ny;
287             %Resistance Values
288             rxM = (Conductivity_Map(i,j) + Conductivity_Map(i-1,j))/2;
289             rxP = (Conductivity_Map(i,j) + Conductivity_Map(i+1,j))/2;
290             ryP= (Conductivity_Map(i,j) + Conductivity_Map(i,j+1))/2;
291             %node equations
292             G(n,n) = -(rxM + rxP + ryP);
293             G(n,xM) = rxM;
294             G(n,xP) = rxP;
295             G(n,yP) = ryP;
296         elseif j == ny%BC @ y=W
297             xM = j + (i-2)*ny;
298             xP = j + i*ny;
299             nym = j-1 + (i-1)*ny;
300             %Resistance Values
301             rxM = (Conductivity_Map(i,j) + Conductivity_Map(i-1,j))/2;
302             rxP = (Conductivity_Map(i,j) + Conductivity_Map(i+1,j))/2;
303             rym = (Conductivity_Map(i,j) + Conductivity_Map(i,j-1))/2;
304             %node equations
305             G(n,n) = -(rxM + rxP + rym);
306             G(n,xM) = rxM;
307             G(n,xP) = rxP;
308             G(n,nym) = rym;
309         %internal nodes
310         else
311             xM = j + (i-2)*ny;
312             xP = j + i*ny;
313             nym = j-1 + (i-1)*ny;
314             yP = j+1 + (i-1)*ny;
315             %Resistor Values
316             rxM = (Conductivity_Map(i,j) + Conductivity_Map(i-1,j))/2;
317             rxP = (Conductivity_Map(i,j) + Conductivity_Map(i+1,j))/2;
318             ryp = (Conductivity_Map(i,j) + Conductivity_Map(i,j+1))/2;
319             rym = (Conductivity_Map(i,j) + Conductivity_Map(i,j-1))/2;
320             %node equations
321             G(n,n) = -(rxM + rxP + rym + ryp);
322             G(n,xM) = rxM;
323             G(n,xP) = rxP;
324             G(n,nym) = rym;
325             G(n,yP) = ryp;
326         end
327     end
328 end
329
330 V = G\F';
331 Voltage_Map = zeros(nx,ny);
332 for i = 1:nx
333     for j = 1:ny
334         Voltage_Map(i,j) = V(j + (i-1)*ny);
335     end
336 end
337
338 %[X , Y] = meshgrid(0:MeshSize:L,0:MeshSize:W);
339 figure(2)
340 subplot(2,1,1)
341 surf(Voltage_Map) %X',Y',
342 axis tight
343 xlabel('x posistion (nm)')
344 ylabel('y posistion (nm)')
345 zlabel('Voltage (V)')
346 title('Voltage across region (3D view)','interpreter','Latex')
```

```
347 view(90,25)
348 subplot(2,1,2)
349 surf(Voltage_Map) %X',Y',
350 axis tight
351 xlabel('x posistion (nm)')
352 ylabel('y posistion (nm)')
353 zlabel('Voltage (V)')
354 title('Voltage across region (Top down view)','interpreter','Latex')
355 view(2)
356 %saveas(gcf,fullfile('D:\School Work\ELEC 4700\My 4700 Code\Assignment 3\Simulation
    ↪ Results','[Part2]VoltageMap.png'),'png')
357
358 [Ex, Ey] = gradient(Voltage_Map,MeshSize);
359 Ex=-Ex;
360 Ey=-Ey;
361 figure(3)
362 quiver(Ex,Ey);
363 axis tight
364 xlabel('x posistion (nm)')
365 ylabel('y posistion (nm)')
366 title('Electric Field across region','interpreter','Latex')
367 %saveas(gcf,fullfile('D:\School Work\ELEC 4700\My 4700 Code\Assignment 3\Simulation
    ↪ Results','[Part2]QuiverPlot.png'),'png')
```

Source Code 2: Part 2 - Finite Difference Method Simulator

4.3 Part 3 Code

```
%{
Author : Maharshi Gurjar
Elec 4700 Assignment 3 -  Finite Difference Method
Part 3
%}
%%
clc; close all; clear;
set(0, 'DefaultFigureWindowStyle', 'docked')
%%
%Define simulation envrionment and constants
M0 = 9.10938356e-31; %Rest mass of electron
Mass_n = 0.26*M0; %Effective mass of electron
Qe = -1.60217662e-19; %Electron Charge
Vx = -0.4; % Voltage along the x-axis
Vy = 0; %Voltage along the y-axis
EConcentration = 1e15 * 100^2; %Electron concentration (1/m^2)
T = 300; % Simulation envrionment temperature (K)
k = 1.38064852e-23; % Boltzmans constant
V_thermal = sqrt(2*k*T/Mass_n); %Thermal Veleocity
Height = 100e-9; % The height of the simulation environment
Length = 200e-9; % The lengthof the simulation environment
nElectrons = 30e3; % Total number of electrons to simulate
nPlotted_Electrons = 20; %Total number of electrons displayed
Time_Step = Height/V_thermal/100; % Time step of simulation
Iterations = 500; % Number of iternations to simulate
Show_Movie = 0; %Display steps control
%%
% The mean free path is determined by multipling the thermal velocity
... by the mean time between collisions:
    MFP = V_thermal * 0.2e-12; %Mean free path
%%
%The state of the electron  (postion and velocity) is stored in a array
... where each index refers to [x-position y-position v-in-x v-in-y]
    Electron_State = zeros(nElectrons,4);
%%
%Temperature will be recorded in the array below
Temperature = zeros(Iterations,1);
%%
%Create a scattering probability
P_Scattering =1 - exp(-Time_Step/0.2e-12);
%%
%Create a distribution using the matlab makedist function
Velocity_PDF = makedist('Normal', 'mu', 0, 'sigma', sqrt(k*T/Mass_n));
%%
%Setting the top/bottom of the boudnary specularity
Top_Specular = 1;
Bottom_Specular = 1;
%Set the box characteristics (0 = diffusive)
Top_Box_Specular = 1;
Bottom_Box_Specular = 1;
%Create Box-positions [x1, x2, y1,y2]
%Box_pos = 1e-9.*[80 120 0 32.5; 80 120 67.5 100]; % [Gap size 35nm]
%Box_pos = 1e-9.*[80 120 0 35; 80 120 65 100]; % [Gap size 30nm]
%Box_pos = 1e-9.*[80 120 0 37.5; 80 120 62.5 100]; % [Gap size 25nm]
Box_pos = 1e-9.*[80 120 0 40; 80 120 60 100]; % [Gap size 20nm] [Control ]
%Box_pos = 1e-9.*[80 120 0 42.5; 80 120 57.5 100]; % [Gap size 15nm]
%Box_pos = 1e-9.*[80 120 0 45; 80 120 55 100]; % [Gap size 10nm]
%Box_pos = 1e-9.*[80 120 0 47.5; 80 120 52.5 100]; % [Gap size 5nm]
%%
% The electric field compoennts (assuming uniform fields) given by:
Ex = Vx/Length;
```

```
Ey = Vy/Height;
%%
% Force on individual force given by:
Fx = Qe*Ex;
Fy = Qe*Ey;
%%
% Current Density vector
J = zeros(Iterations,2);
%% Current plot and Temperature plot stuff
Temp_Plot = zeros(Iterations,1);
Current_Plot = zeros(Iterations,1);
%%
%Create the state of the box (specular or diffusive)
%Generate a random inital population postion and velocity
for i = 1:nElectrons
    Electron_State(i,:) = [Length*rand() Height*rand() random(Velocity_PDF) random(Velocity_PDF)];
    Electron_State( (Electron_State(:,1)>Box_pos(1,1) & Electron_State(:,1)<Box_pos(1,2) & ...
        (Electron_State(:,2)<Box_pos(1,4))) ,1) = Length*rand();
    Electron_State( (Electron_State(:,1)>Box_pos(2,1) & Electron_State(:,1)<Box_pos(2,2) & ...
        (Electron_State(:,2)>Box_pos(2,3))) ,1) = Length*rand();
end
%%
%Figure below used to test inital postion of electrons, and if movie is off
%shows the the same.
figure("Name","Electron positions")
plot(Electron_State(1:nPlotted_Electrons,1)./1e-9,Electron_State(1:nPlotted_Electrons,2)./1e-9,'o');
grid on;
axis([0 Length/1e-9 0 Height/1e-9]);
xlabel('x (nm)');
ylabel('y (nm)');
title(sprintf("Plotting (%d/%d) electron ",nPlotted_Electrons,nElectrons));
%saveas(gcf,'Part_Three_Boxes.png')
hold on;
for j=1:size(Box_pos,1)
    plot([Box_pos(j, 1) Box_pos(j, 1) Box_pos(j, 2) Box_pos(j, 2) Box_pos(j, 1)]./1e-9,...
        [Box_pos(j, 3) Box_pos(j, 4) Box_pos(j, 4) Box_pos(j, 3) Box_pos(j, 3)]./1e-9, 'k-');
end
hold off;
%%
%We will now move (iterate) over time, updating the positions and direction
...while plotting the state
    for i = 1:Iterations
    %The line below updates the x,y position by moving it to a new position
    ... using its current position + the velocity*(time step)
    Electron_Prev_State = Electron_State(:,1:2);
    Electron_State(:,3) = Electron_State(:,3) + Fx*Time_Step/Mass_n;
    Electron_State(:,4) = Electron_State(:,4) + Fy*Time_Step/Mass_n;
    Electron_State(:,1:2) = Electron_State(:,1:2) + Time_Step*Electron_State(:,3:4);
    %Checking boundary conditions using Matlab matrix equations/indexing
    %Check (if) and move all electrons at X=200nm Bound:
    Electron_State((Electron_State(:,1)>Length),1) = Electron_State((Electron_State(:,1)>Length),1)
    ↪  - Length;

    %Check (if) and move all electrons at X=0nm Bound:
    Electron_State((Electron_State(:,1)<0),1) =Electron_State((Electron_State(:,1)<0),1) + Length;

    %Check (if) and move all electrons at Y Bounds and if specular or diffusive:
    Electron_State((Electron_State(:,2)>Height & Top_Specular==1),4) =
    ↪  -1*Electron_State((Electron_State(:,2)>Height & Top_Specular),4);
    Electron_State((Electron_State(:,2)>Height & Top_Specular==1),2) = 2*Height -
    ↪  Electron_State((Electron_State(:,2)>Height & Top_Specular),2);
    Electron_State((Electron_State(:,2)>Height & Top_Specular==0),4) = -random(Velocity_PDF);
    Electron_State((Electron_State(:,2)>Height & Top_Specular==0),3) = random(Velocity_PDF);
    Electron_State((Electron_State(:,2)<0 & Bottom_Specular==1),4) =
    ↪  -Electron_State((Electron_State(:,2)<0 & Bottom_Specular==1),4);
```

```
490     Electron_State((Electron_State(:,2)<0 & Bottom_Specular==1),2) =
        ↪  -Electron_State((Electron_State(:,2)<0 & Bottom_Specular==1),2);
491     Electron_State((Electron_State(:,2)<0 & Bottom_Specular==0),4) = random(Velocity_PDF);
492     Electron_State((Electron_State(:,2)<0 & Bottom_Specular==0),3) = random(Velocity_PDF);
493     %%
494     %BOX SPECULAR CASE :
495     %Check if bottom Box is specular and if so then bounce
496     Electron_State( (Bottom_Box_Specular==1 & Electron_State(:,1)>Box_pos(1,1) &
        ↪  Electron_State(:,1)<Box_pos(1,2) & ((Electron_State(:,2)<Box_pos(1,4) &
        ↪  Electron_Prev_State(:,2)<Box_pos(1,4)))),3)...
497         =-Electron_State( (Bottom_Box_Specular==1 & Electron_State(:,1)>Box_pos(1,1) &
            ↪  Electron_State(:,1)<Box_pos(1,2) & ((Electron_State(:,2)<Box_pos(1,4) &
            ↪  Electron_Prev_State(:,2)<Box_pos(1,4)))),3);
498
499     %Check if top Box is specular and if so then bounce
500     Electron_State( (Top_Box_Specular==1 & Electron_State(:,1)>Box_pos(2,1) &
        ↪  Electron_State(:,1)<Box_pos(2,2) & ((Electron_State(:,2)>Box_pos(2,3) &
        ↪  Electron_Prev_State(:,2)>Box_pos(2,3)))),3)...
501         =-Electron_State( (Top_Box_Specular==1 & Electron_State(:,1)>Box_pos(2,1) &
            ↪  Electron_State(:,1)<Box_pos(2,2) & ((Electron_State(:,2)>Box_pos(2,3) &
            ↪  Electron_Prev_State(:,2)>Box_pos(2,3)))),3);
502
503     %Top Horizontal
504     Electron_State( (Top_Box_Specular==1 & Electron_State(:,1)<Box_pos(2,2) &
        ↪  Electron_State(:,1)>Box_pos(2,1)) & (Electron_State(:,2)>Box_pos(2,3)) &...
505         (Electron_Prev_State(:,1)>Box_pos(2,1) & Electron_Prev_State(:,1)<Box_pos(2,2)),4) =
            ↪  -Electron_State( (Top_Box_Specular==1 & Electron_State(:,1)<Box_pos(2,2) &
            ↪  Electron_State(:,1)>Box_pos(2,1)) & (Electron_State(:,2)>Box_pos(2,3)) &...
506         (Electron_Prev_State(:,1)>Box_pos(2,1) & Electron_Prev_State(:,1)<Box_pos(2,2)),4);
507     %Bottom Hoizontal
508     Electron_State( (Bottom_Box_Specular==1 & Electron_State(:,1)<Box_pos(1,2) &
        ↪  Electron_State(:,1)>Box_pos(1,1)) & (Electron_State(:,2)<Box_pos(1,4)) &...
509         (Electron_Prev_State(:,1)>Box_pos(1,1) & Electron_Prev_State(:,1)<Box_pos(1,2)),4) =
            ↪  -Electron_State( (Bottom_Box_Specular==1 & Electron_State(:,1)<Box_pos(1,2) &
            ↪  Electron_State(:,1)>Box_pos(1,1)) & (Electron_State(:,2)<Box_pos(1,4)) &...
510         (Electron_Prev_State(:,1)>Box_pos(1,1) & Electron_Prev_State(:,1)<Box_pos(1,2)),4);
511     %BOX SPECULAR CASE END :
512     %%
513     %BOX DIFFUSIVE CASE:
514     %Check if bottom Box is specular and if so then bounce
515     Electron_State( (Bottom_Box_Specular==0 & Electron_State(:,1)>Box_pos(1,1) &
        ↪  Electron_State(:,1)<Box_pos(1,2) & ((Electron_State(:,2)<Box_pos(1,4) &
        ↪  Electron_Prev_State(:,2)<Box_pos(1,4)))),3)...
516         =random(Velocity_PDF);
517     Electron_State( (Bottom_Box_Specular==0 & Electron_State(:,1)>Box_pos(1,1) &
        ↪  Electron_State(:,1)<Box_pos(1,4) & ((Electron_State(:,2)<Box_pos(1,4) &
        ↪  Electron_Prev_State(:,2)<Box_pos(1,4)))),4)...
518         =random(Velocity_PDF);
519
520     %Check if Top Box is specular and if so then bounce
521     Electron_State( (Top_Box_Specular==0 & Electron_State(:,1)>Box_pos(2,1) &
        ↪  Electron_State(:,1)<Box_pos(2,2) & ((Electron_State(:,2)>Box_pos(2,3) &
        ↪  Electron_Prev_State(:,2)>Box_pos(2,3)))),3)...
522         =random(Velocity_PDF);
523     Electron_State( (Top_Box_Specular==0 & Electron_State(:,1)>Box_pos(2,1) &
        ↪  Electron_State(:,1)<Box_pos(2,2) & ((Electron_State(:,2)>Box_pos(2,3)&
        ↪  Electron_Prev_State(:,2)>Box_pos(2,3)))),4)...
524         =random(Velocity_PDF);
525
526     %Bottom Horizontal
527     Electron_State( (Bottom_Box_Specular==0 & Electron_State(:,1)<Box_pos(1,2)&
        ↪  Electron_State(:,1)>Box_pos(1,1)) & (Electron_State(:,2)<Box_pos(1,4)) &...
528         (Electron_Prev_State(:,1)>Box_pos(1,1) & Electron_Prev_State(:,1)<Box_pos(1,2)),3) =
            ↪  random(Velocity_PDF);
```

```
529     Electron_State( (Bottom_Box_Specular==0 & Electron_State(:,1)<Box_pos(1,2) &
        ↪ Electron_State(:,4)<0 & Electron_State(:,1)>Box_pos(1,1)) &
        ↪ (Electron_State(:,2)<Box_pos(1,4)) &...
530         (Electron_Prev_State(:,1)>Box_pos(1,1) & Electron_Prev_State(:,1)<Box_pos(1,2)),4) =
            ↪ random(Velocity_PDF);
531     %Top Hoizontal
532     Electron_State( (Top_Box_Specular==0  & Electron_State(:,1)<Box_pos(2,2) &
        ↪ Electron_State(:,1)>Box_pos(2,1)) & (Electron_State(:,2)>Box_pos(2,3)) &...
533         (Electron_Prev_State(:,1)>Box_pos(2,1) & Electron_Prev_State(:,1)<Box_pos(2,2)),3) =
            ↪ random(Velocity_PDF);
534     Electron_State( (Top_Box_Specular==0 & Electron_State(:,4)>0 & Electron_State(:,1)<Box_pos(2,2)
        ↪ & Electron_State(:,1)>Box_pos(2,1)) & (Electron_State(:,2)>Box_pos(2,3)) &...
535         (Electron_Prev_State(:,1)>Box_pos(2,1) & Electron_Prev_State(:,1)<Box_pos(2,2)),4) =
            ↪ -random(Velocity_PDF);
536     %%
537     %Add scattering
538     j = rand(nElectrons,1) < P_Scattering;
539     Electron_State(j,3:4) = random(Velocity_PDF,[sum(j),2]);
540     %%
541     %Electron shifting
542     %Region 1 (Bottom  left)
543     Electron_State((Electron_State(:,1)>Box_pos(1,1) & Electron_State(:,1)<Box_pos(1,2) &
        ↪ ((Electron_State(:,2)<Box_pos(1,4) & Electron_Prev_State(:,2)<Box_pos(1,4))) &...
544         Electron_Prev_State(:,1)<Box_pos(1,1)),1) = 2*Box_pos(1,1) - Electron_State(
            ↪ (Electron_State(:,1)>Box_pos(1,1) & Electron_State(:,1)<Box_pos(1,2) &
            ↪ ((Electron_State(:,2)<Box_pos(1,4) & Electron_Prev_State(:,2)<Box_pos(1,4))) &...
545         Electron_Prev_State(:,1)<Box_pos(1,1)),1);
546     %Region 3 (Top left)
547     Electron_State((Electron_State(:,1)>Box_pos(2,1) & Electron_State(:,1)<Box_pos(2,2) &
        ↪ ((Electron_State(:,2)>Box_pos(2,3) & Electron_Prev_State(:,2)>Box_pos(2,3))) &...
548         Electron_Prev_State(:,1)<Box_pos(2,1)),1) = 2*Box_pos(2,1) - Electron_State(
            ↪ (Electron_State(:,1)>Box_pos(2,1) & Electron_State(:,1)<Box_pos(2,2) &
            ↪ ((Electron_State(:,2)>Box_pos(2,3) & Electron_Prev_State(:,2)>Box_pos(2,3))) &...
549         Electron_Prev_State(:,1)<Box_pos(2,1)),1);
550     %Region 2 (Bottom right)
551     Electron_State((Electron_State(:,1)>Box_pos(1,1) & Electron_State(:,1)<Box_pos(1,2) &
        ↪ ((Electron_State(:,2)<Box_pos(1,4) & Electron_Prev_State(:,2)<Box_pos(1,4))) &...
552         Electron_Prev_State(:,1)>Box_pos(1,2)),1) = 2*Box_pos(1,2) - Electron_State(
            ↪ (Electron_State(:,1)>Box_pos(1,1) & Electron_State(:,1)<Box_pos(1,2) &
            ↪ ((Electron_State(:,2)<Box_pos(1,4) & Electron_Prev_State(:,2)<Box_pos(1,4))) &...
553         Electron_Prev_State(:,1)>Box_pos(1,2)),1);
554     %Region 4 (Top Right)
555     Electron_State((Electron_State(:,1)>Box_pos(2,1) & Electron_State(:,1)<Box_pos(2,2) &
        ↪ ((Electron_State(:,2)>Box_pos(2,3) & Electron_Prev_State(:,2)>Box_pos(2,3))) &...
556         Electron_Prev_State(:,1)>Box_pos(2,2)),1) = 2*Box_pos(2,2) - Electron_State(
            ↪ (Electron_State(:,1)>Box_pos(2,1) & Electron_State(:,1)<Box_pos(2,2) &
            ↪ ((Electron_State(:,2)>Box_pos(2,3) & Electron_Prev_State(:,2)>Box_pos(2,3))) &...
557         Electron_Prev_State(:,1)>Box_pos(2,2)),1);
558
559     %Bottom Horizontal (going top down)
560     Electron_State( (Electron_State(:,1)<Box_pos(1,2) & Electron_State(:,1)>Box_pos(1,1) &
        ↪ (Electron_State(:,2)<Box_pos(1,4) & Electron_Prev_State(:,2)>Box_pos(1,4)) &...
561         (Electron_State(:,4)<0)),2) =  2*Box_pos(1,4) + Electron_State(
            ↪ (Electron_State(:,1)<Box_pos(1,2) & Electron_State(:,1)>Box_pos(1,1) &
            ↪ (Electron_State(:,2)<Box_pos(1,4) & Electron_Prev_State(:,2)>Box_pos(1,4)) &...
562         (Electron_State(:,4)<0)),2);
563     %Top Horizontal (going bottom up)
564     Electron_State( (Electron_State(:,1)<Box_pos(2,2) & Electron_State(:,1)>Box_pos(2,1) &
        ↪ (Electron_State(:,2)>Box_pos(2,3) & Electron_Prev_State(:,2)<Box_pos(2,3)) &...
565         (Electron_State(:,4)>0)),2) =  2*Box_pos(2,3) - Electron_State(
            ↪ (Electron_State(:,1)<Box_pos(2,2) & Electron_State(:,1)>Box_pos(2,1) &
            ↪ (Electron_State(:,2)>Box_pos(2,3) & Electron_Prev_State(:,2)<Box_pos(2,3)) &...
566         (Electron_State(:,4)>0)),2);
567     %Electron shifting end
```

```
568         %%
569         % Stores the Electron [x y] posistions in the Trajectories vector
570         ... for each different electron in a new coloum
571             for j = 1: nPlotted_Electrons
572             Trajectories_x(i,j) = Electron_State(j,1);
573             Trajectories_y(i,j) = Electron_State(j,2);
574             end
575             %To calcuatle the themal energy, Maxwell's principle of equipartion
576             ... is used,  where the final equation then becomes;
577                 Temperature(i) = ( sum (Electron_State(:,3).^2) + sum(Electron_State(:,4).^2)) * Mass_n
                    ↪  / k / 2 / nElectrons;
578             %To calculate the current density
579             J(i,1) = Qe.*EConcentration.*mean(Electron_State(:,3));
580             J(i,2) = Qe.*EConcentration.*mean(Electron_State(:,4));
581             % Add the temperature and current data to the respective plots
582             Temp_Plot(i,1) = Temperature(i);
583             Current_Plot(i,1) = J(i,1)^2 + J(i,2)^2;
584             %Shows the pathing of the electron, as well as the updating trajectory
585             if Show_Movie && mod(i,2)
586                 figure(1)
587                 plot(Electron_State(1:nPlotted_Electrons,1)./1e-9,Electron_State(1:nPlotted_Electrons,2
                    ↪  )./1e-9,'o');
588                 grid on;
589                 axis([0 Length/1e-9 0 Height/1e-9]);
590                 xlabel('x (nm)');
591                 ylabel('y (nm)');
592                 title(sprintf("Plotting (%d/%d) electron at constant
                    ↪  velocity",nPlotted_Electrons,nElectrons));
593                 hold on;
594                 for j=1:size(Box_pos,1)
595                     plot([Box_pos(j, 1) Box_pos(j, 1) Box_pos(j, 2) Box_pos(j, 2) Box_pos(j,
                        ↪  1)]./1e-9,...
596                         [Box_pos(j, 3) Box_pos(j, 4) Box_pos(j, 4) Box_pos(j, 3) Box_pos(j, 3)]./1e-9,
                            ↪  'k-');
597                 end
598                 hold off;
599                 pause(0.01)
600             end
601         end
602     %%
603     figure("name","Trajectory, temperature and speed results results")
604     subplot(2,2,1)
605     plot(Trajectories_x(:,1:nPlotted_Electrons)./1e-9, Trajectories_y(:,1:nPlotted_Electrons)./1e-9,'.'
    );
606     hold on;
607     for j=1:size(Box_pos,1)
608         plot([Box_pos(j, 1) Box_pos(j, 1) Box_pos(j, 2) Box_pos(j, 2) Box_pos(j, 1)]./1e-9,...
609             [Box_pos(j, 3) Box_pos(j, 4) Box_pos(j, 4) Box_pos(j, 3) Box_pos(j, 3)]./1e-9, 'k-');
610     end
611     hold off
612     axis([0 Length/1e-9 0 Height/1e-9]);
613     xlabel('x (nm)');
614     ylabel('y (nm)');
615     grid on;
616     title(sprintf("Trajectories of (%.2d/%.2d) electron(s) at constant
    ↪  velocity",nPlotted_Electrons,nElectrons),'interpreter','latex');
617     subplot(2,2,2)
618     plot(Time_Step*(0:Iterations-1), Temperature);
619     grid on;
620     xlim([0 Time_Step*Iterations])
621     title(sprintf("Temperature of the region, Average Temperature: %.2f
    ↪  (K)",mean(Temperature)),'interpreter','latex')
622     xlabel('Time (s)');
623     ylabel('Temperature (K)');
```

```
subplot(2,2,3)
Velocity = sqrt(Electron_State(:,3).^2 + Electron_State(:,4).^2);
histogram(Velocity);
title(sprintf("Electron Velocity, Average Velocity %.2d (m/s)",mean(Velocity)),'interpreter','latex');
xlabel("Speed (m/s)");
ylabel("Number of particles");
grid on;
subplot(2,2,4)
plot(Time_Step*(0:Iterations-1),Current_Plot)
title(sprintf("Jx - Drift Current Density along x-axis, Average : %.2d (A/m)",mean(Current_Plot)),'interpreter','latex');
xlabel('Time (s)');
ylabel('Current Density (A/m)');
grid on;
axis tight;
%saveas(gcf,fullfile('D:\School Work\ELEC 4700\My 4700 Code\Assignment 3\Simulation Results','[Part3]Trajectory_Temperature_DrifCurr.png'),'png')
%%
Density = hist3(Electron_State(:,1:2),[200 100])';
N = 20;
sigma = 3;
%Creating a Gaussian filtering matrix
[x,y]=meshgrid(round(-N/2):round(N/2), round(-N/2):round(N/2));
G=exp(-x.^2/(2*sigma^2)-y.^2/(2*sigma^2));
G=G./sum(G(:));
Density = conv2(Density,G,'same') / (Height./size(Density,1)*Length./size(Density,2));
%%
%Plot drift current
figure('Name','Drift current (x-axis) and Electron Density')
subplot(2,1,1)
surf(conv2(Density,G,'same'));
xlabel('x (nm)')
ylabel('y (nm)')
zlabel('Bin count')
axis tight
title("Electron density region (3D view)",'interpreter','latex');
%%
% Plot the electron density
subplot(2,1,2)
surf(conv2(Density,G,'same'));
xlabel('x (nm)')
ylabel('y (nm)')
zlabel('Bin count')
axis tight
title("Electron density region (Top down view)",'interpreter','latex');
view(2)
%saveas(gcf,fullfile('D:\School Work\ELEC 4700\My 4700 Code\Assignment 3\Simulation Results','[Part3]ElectronDensity.png'),'png')
```

Source Code 3: Part 3 - Coupled Simulator