ELEC 4700 Assignment-3
Monte-Carlo/FiniteDifference Method

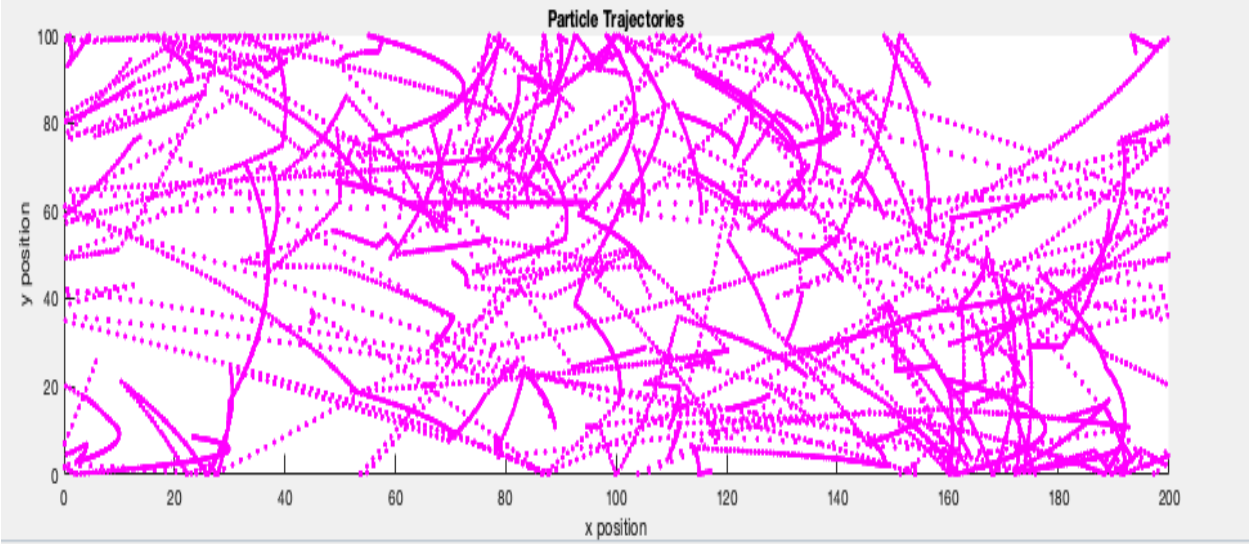
Spencer Tigere
101001717
Submitted: March 21, 2021
For: Tom Smy
Carleton University

Introduction

The purpose of this lab was to expand on labs 1 and 2 by combining the Monte Carlo simulator with the Finite Difference method to observe what happens when we introduce a field pushing the particles.

Part 1

a) The electric field of the charge is 500000.000000 V/m.
b) The force on the charge is 8.010883e-14 N.
c) The acceleration of the charge is 338234619754597888.000000 m/s^2.



d) The relationship between current density and the average carrier velocity is linear. The relationship can be expressed such as:

J = V*n*q*Ny

= (avg. carrier velocity)*(electron charge concentration)*(electron charge)*(length along y boundary)

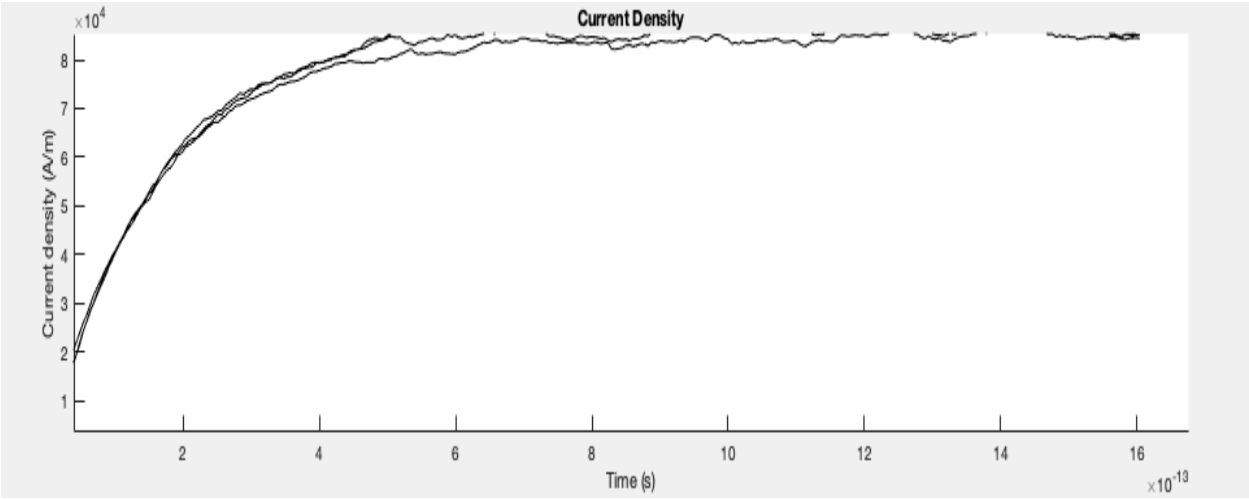


Figure 1

The current rises until it begins to saturate due to rethermalizations. Rethermalization occurs when the amount of scattering electrons begins to cancel acceleration of electrons.

e)

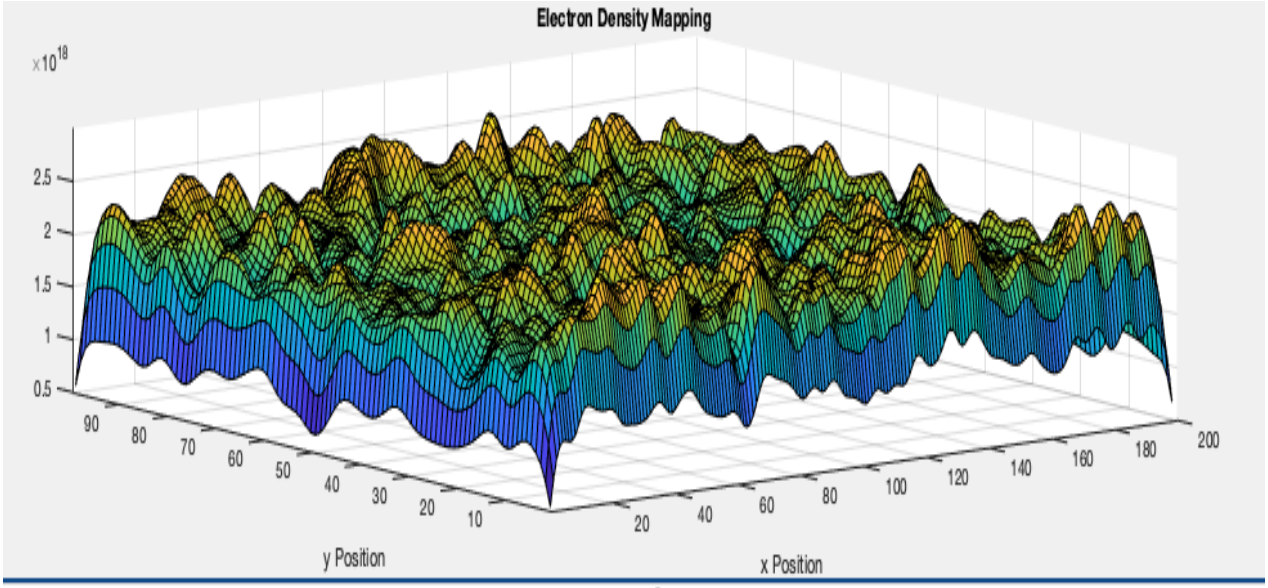


Figure 2

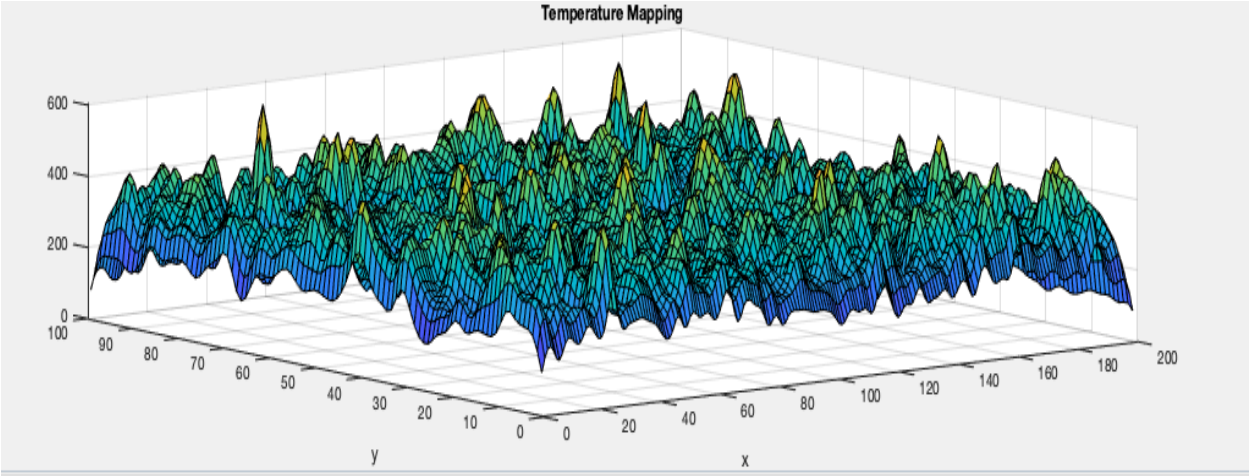


Figure 3

Part 2

a)

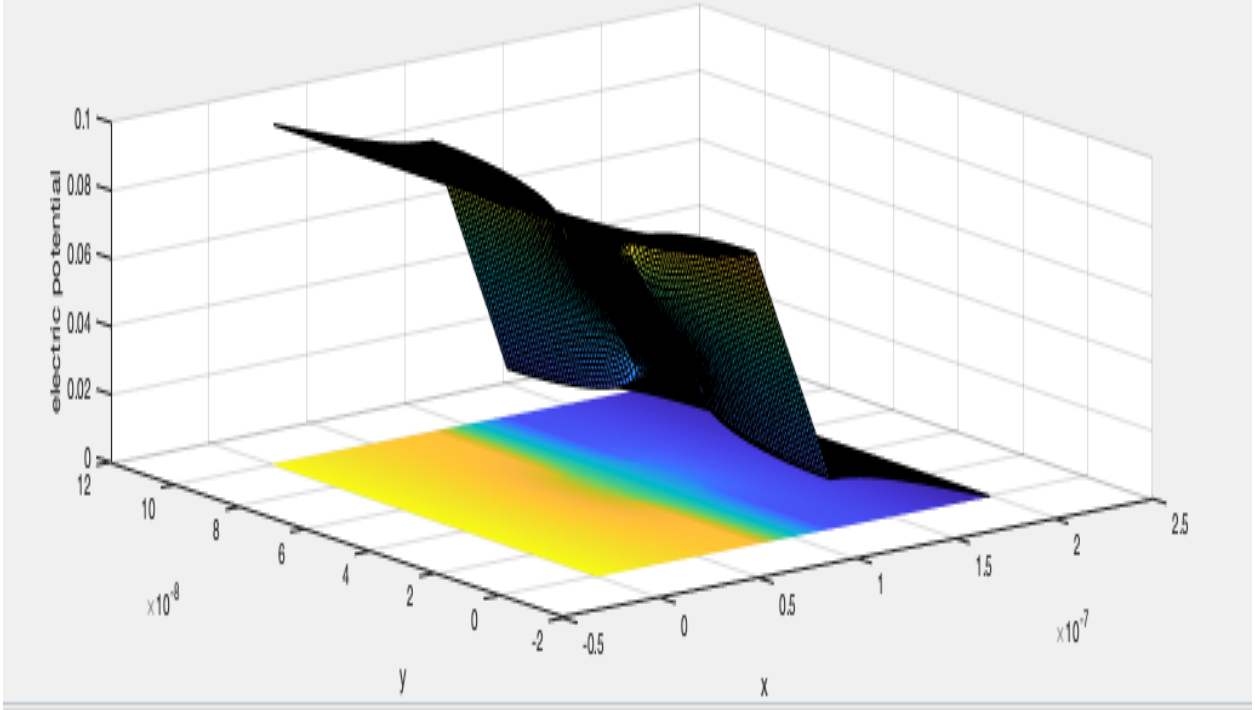


Figure 4

b)

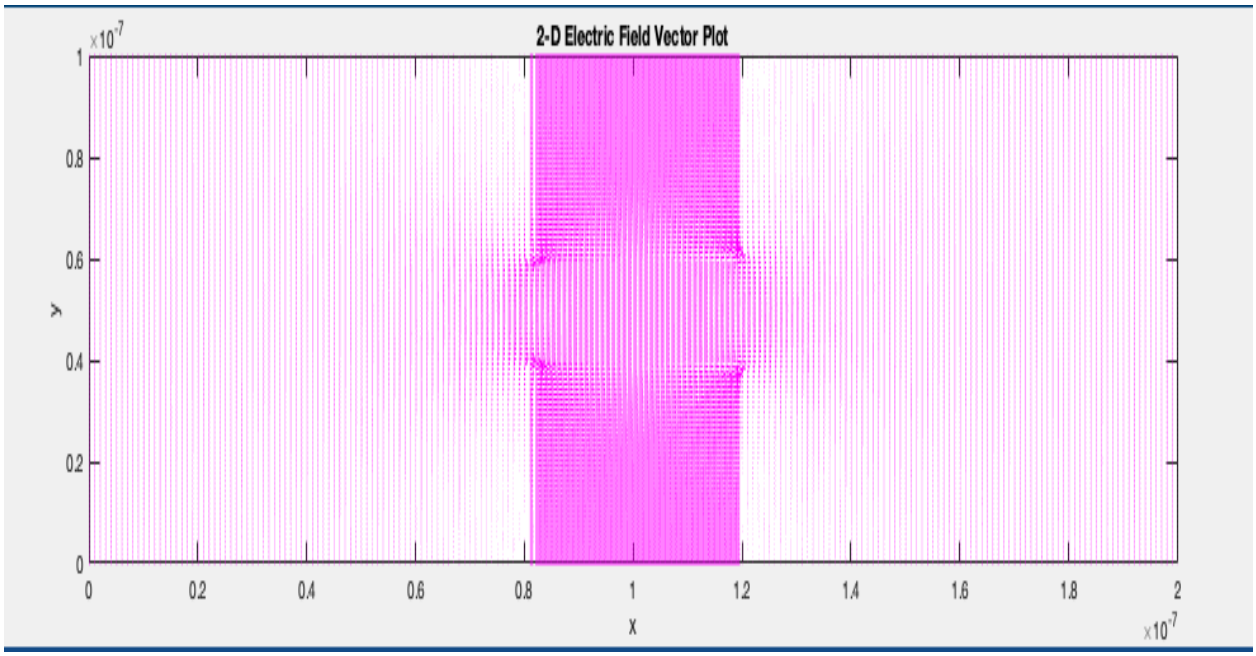


Figure 5

c)

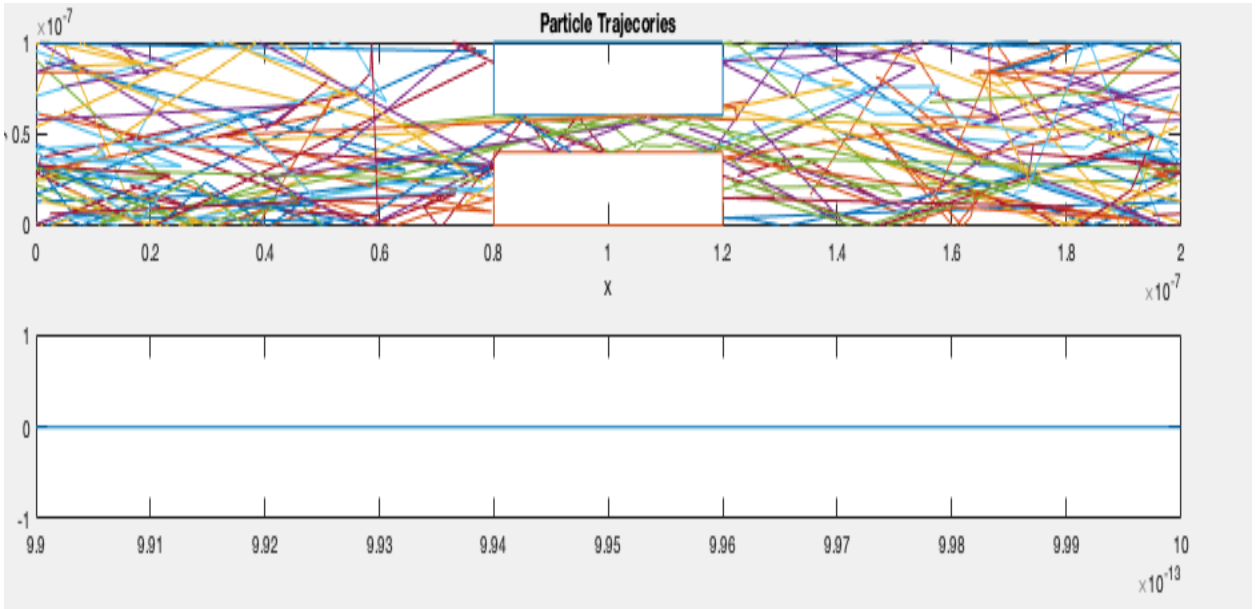


Figure 6

Part 3

a)

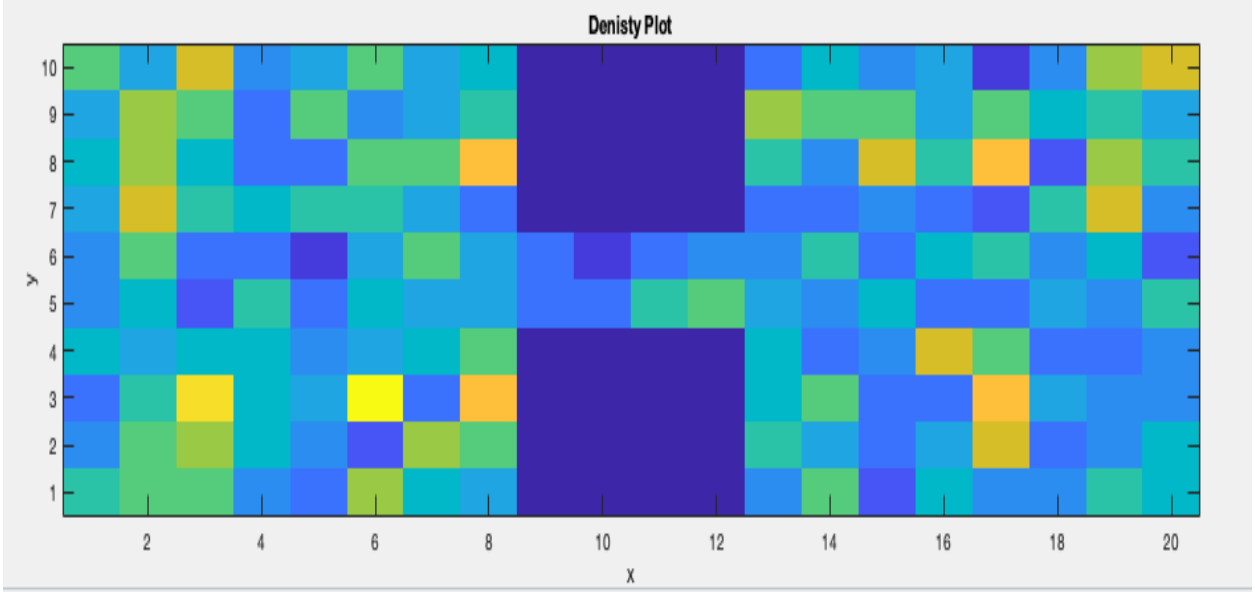


Figure 7

b) Average Current vs bottleneck width

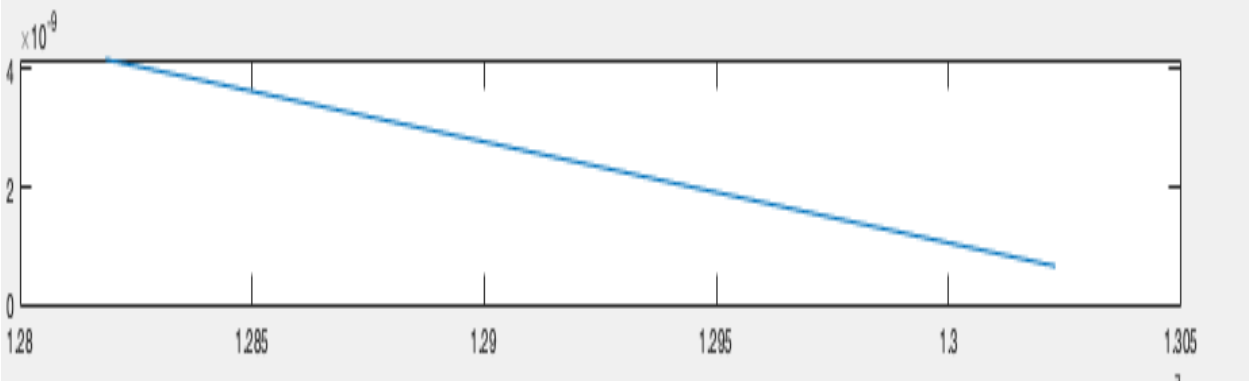


Figure 8

c) Next Step

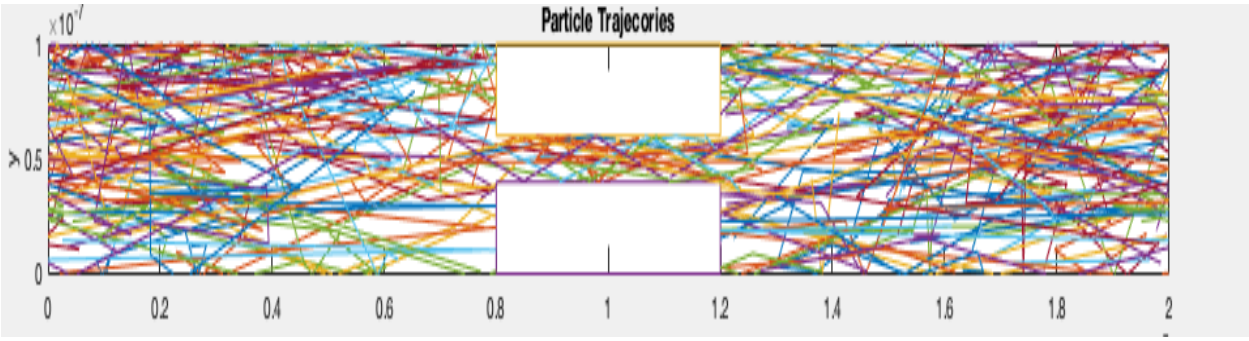


Figure 9

The next step to make the simulation more accurate is to increase the number of particles examined, or to use a finer mesh. The figure above is second execution of the particle trajectory simulation after the initial execution in figure

Conclusion

The lab was performed successfully. The simulations used in labs 1 and 2 were combined with the addition of a field. The electron behaviors and relationships were observed and noted.

Appendix

PART 1

```
%Spencer Tigere 101001717

% Part 1: Monte Carlo Simulator from Assignment 1 without
% the bottleneck from those assignments

clc
clear
set(0,'DefaultFigureWindowStyle','docked')
wid_x = 200e-9;
len_y = 100e-9;
Voltage_x = 0.1;
Voltage_y = 0;

Charge_electron = -1.60217662e-19;
electron_conc = 1e15*100^2;
m0 = 9.10938356e-31;
effective_m = 0.26*m0;
Temperature = 300;
Boltz_const = 1.38064852e-23;
Thermal_v = sqrt(2*Boltz_const*Temperature/effective_m);
mean_free_path = Thermal_v*0.2e-12;
specular_top_bound = 0;
specular_bottom_bound = 0;
time_step = len_y/Thermal_v/100;
num_iterations = 300;
size_p = 40000;
pp = 10;
pscat = 1 - exp(-time_step/0.2e-12);
vel = makedist('Normal', 'mu', 0, 'sigma', sqrt(Boltz_const*Temperature/effective_m));
Display_m = 0;

%Part 1 a)Defining Electric Field as, E = V/D.
electricfield_x = Voltage_x/wid_x;
electricfield_y = Voltage_y/len_y;
electricfield_total = electricfield_x + electricfield_y;
fprintf('The electric field of the charge is %f V/m.\n',electricfield_total);

%Part 1 b)The force on each electron is the sum of its individual components.
x_force = Charge_electron*electricfield_x;
y_force = Charge_electron*electricfield_y;
```

```
total_force = abs(x_force + y_force);
fprintf('The force on the charge is %d N.\n',total_force);

%Part 1 c)Defining acceleration as f=ma we get:
acceleration = total_force/effective_m;
fprintf('The acceleration of the charge is %f m/s^2.\n',acceleration);

%Part 1 d)Relationship between the electron drift current density and average carrier velocity
change_vx = x_force*time_step/effective_m;
change_vy = y_force*time_step/effective_m;
change_vx = change_vx.*ones(size_p,1);
change_vy = change_vy.*ones(size_p,1);
positions = zeros(size_p, 4);
traj = zeros(num_iterations, pp*2);
temporary_a = zeros(num_iterations,1);
J = zeros(num_iterations,2);

%Part 1 e)Density and Temperature maps
% Initializing the positions of the particles
for i = 1:size_p
        theta = rand*2*pi;
        positions(i,:) = [wid_x*rand len_y*rand random(vel) random(vel)];
end
temperature_plot = animatedline;
figure(2);
current_plot = animatedline;
title('Current Density');
xlabel('Time (s)');
ylabel('Current density (A/m)');

% Iterate through the simulation
for i = 1:num_iterations
        positions(:,3) = positions(:,3) + change_vx;
        positions(:,4) = positions(:,4) + change_vy;
        positions(:,1:2) = positions(:,1:2) + time_step.*positions(:,3:4);
        j = positions(:,1) > wid_x;
        positions(j,1) = positions(j,1) - wid_x;
        j = positions(:,1) < 0;
        positions(j,1) = positions(j,1) + wid_x;
        j = positions(:,2) > len_y;

        if(specular_top_bound)
        positions(j,2) = 2*len_y - positions(j,2);
        positions(j,4) = -positions(j,4);
```

```
        else
        positions(j,2) = len_y;
        v = sqrt(positions(j,3).^2 + positions(j,4).^2);
        theta = rand([sum(j),1])*2*pi;
        positions(j,3) = v.*cos(theta);
        positions(j,4) = -abs(v.*sin(theta));
        end

        j = positions(:,2) < 0;

        if(specular_bottom_bound)
        positions(j,2) = -positions(j,2);
        positions(j,4) = -positions(j,4);
        else
        positions(j,2) = 0;
        v = sqrt(positions(j,3).^2 + positions(j,4).^2);
        theta = rand([sum(j),1])*2*pi;
        positions(j,3) = v.*cos(theta);
        positions(j,4) = abs(v.*sin(theta));
        end

        j = rand(size_p, 1) < pscat;
        positions(j,3:4) = random(vel, [sum(j),2]);
        temporary_a(i) = (sum(positions(:,3).^2) +
sum(positions(:,4).^2))*effective_m/Boltz_const/2/size_p;

        for j=1:pp
        traj(i, (2*j):(2*j+1)) = positions(j, 1:2);
        end

        J(i, 1) = Charge_electron.*electron_conc.*mean(positions(:,3));
        J(i, 2) = Charge_electron.*electron_conc.*mean(positions(:,4));
        addpoints(temperature_plot, time_step.*i, temporary_a(i));
        addpoints(current_plot, time_step.*i, J(i,1));
        if(Display_m && mod(i,5) == 0)
        figure(1);
        hold off;
        plot(positions(1:pp,1)./1e-9, positions(1:pp,2)./1e-9, 'o');
        axis([0 wid_x/1e-9 0 len_y/1e-9]);
        hold on;
        title('Particle Trajectories');
        % x and y positions are in nanometers
        xlabel('x position');
        ylabel('y position');
```

```
        pause(0.05);
        end
end

figure(1);
title('Particle Trajectories');
%x and y positions are in nanometers
xlabel('x position');
ylabel('y position');
axis([0 wid_x/1e-9 0 len_y/1e-9]);
hold on;

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

electron_conc = hist3(positions(:,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(3);
electron_conc = conv2(electron_conc,f,'same');
electron_conc = electron_conc/(len_y./size(electron_conc,1)*wid_x./size(electron_conc,2));
surf(conv2(electron_conc,f,'same'));

title('Electron Density Mapping');

xlabel('x Position');
ylabel('y Position');


sum_x = zeros(ceil(wid_x/1e-9),ceil(len_y/1e-9));
sum_y = zeros(ceil(wid_x/1e-9),ceil(len_y/1e-9));
temp_num = zeros(ceil(wid_x/1e-9),ceil(len_y/1e-9));

% electron velocity
for i=1:size_p
        x = floor(positions(i,1)/1e-9);
        y = floor(positions(i,2)/1e-9);
        if(x==0)
        x = 1;
        end
```

```
        if(y==0)
        y= 1;
        end
        sum_y(x,y) = sum_y(x,y) + positions(i,3)^2;
        sum_x(x,y) = sum_x(x,y) + positions(i,4)^2;
        temp_num(x,y) = temp_num(x,y) + 1;
end

temporary_a = (sum_x + sum_y).*effective_m./Boltz_const./2./temp_num;
temporary_a(isnan(temporary_a)) = 0;
temporary_a = temporary_a';
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(4);
surf(conv2(temporary_a,f,'same'));
```

PART 2

```
clc
clear
set(0,'DefaultFigureWindowStyle','docked')


% Part 2: Use The Finite Difference Method in Assignment 2 to calculate the
% electric field and then provide a field for the Monte Carlo bottleneck
% Simulation

length_y = 200e-9;
width_x = 100e-9;
Len_box = 40e-9;
Wid_box = 40e-9;
meshspace = 1e-9;
num_x = round(length_y/meshspace + 1);
num_y = round(width_x/meshspace + 1);

conductivity_outside = 1;
conductivity_inside = 1e-2;

% Part 2 a)
conductivity_mapping = zeros(num_x,num_y);
```

```
for i = 1:num_x
  for j = 1:num_y
        if
(i-1)>0.5*(length_y-Len_box)/meshspace&&(i-1)<0.5*(length_y+Len_box)/meshspace&&((j-1)<Wid_b
ox/meshspace||(j-1)>(width_x-Wid_box)/meshspace)
        conductivity_mapping(i,j) = conductivity_inside;
        else
        conductivity_mapping(i,j) = conductivity_outside;
        end
  end
end


G_matrix = sparse(num_x*num_y);
B_matrix = zeros(1,num_x*num_y);
for i = 1:num_x
        for j = 1:num_y
        n = j +(i-1)*num_y;
        n1 = j + (i - 1) * length_y;
        nxm1 = j + ((i-1) - 1) * length_y;
        nxp1 = j + ((i+1) - 1) * length_y;
        nym1 = (j-1) + (i - 1) * length_y;
        nyp1 = (j+1) + (i - 1) * length_y;

        if i == 1
        n1 = j + (i - 1) * length_y;
        nxm1 = j + ((i-1) - 1) * length_y;
        nxp1 = j + ((i+1) - 1) * length_y;
        nym1 = (j-1) + (i - 1) * length_y;
        nyp1 = (j+1) + (i - 1) * length_y;
        G_matrix(n,n) = 1;
        B_matrix(n) = 0.1;

        elseif i == num_x
        n1 = j + (i - 1) * length_y;
        nxm1 = j + ((i-1) - 1) * length_y;
        nxp1 = j + ((i+1) - 1) * length_y;
        nym1 = (j-1) + (i - 1) * length_y;
        nyp1 = (j+1) + (i - 1) * length_y;
        G_matrix(n,n) = 1;


        elseif j == 1
```

```
n1 = j + (i - 1) * length_y;
nxm1 = j + ((i-1) - 1) * length_y;
nxp1 = j + ((i+1) - 1) * length_y;
nym1 = (j-1) + (i - 1) * length_y;
nyp1 = (j+1) + (i - 1) * length_y;
nxm = j + (i-2)*num_y;
nxp = j + i*num_y;
nyp = j+1 + (i-1)*num_y;
rxm = (conductivity_mapping(i,j) + conductivity_mapping(i-1,j))/2;
rxp = (conductivity_mapping(i,j) + conductivity_mapping(i+1,j))/2;
ryp = (conductivity_mapping(i,j) + conductivity_mapping(i,j+1))/2;

G_matrix(n,n) = -(rxm + rxp + ryp);
G_matrix(n,nxm) = rxm;
G_matrix(n,nxp) = rxp;
G_matrix(n,nyp) = ryp;


elseif j == num_y
nxm = j + (i-2)*num_y;
nxp = j + i*num_y;
nym = j-1 + (i-1)*num_y;
n1 = j + (i - 1) * length_y;
nxm1 = j + ((i-1) - 1) * length_y;
nxp1 = j + ((i+1) - 1) * length_y;
nym1 = (j-1) + (i - 1) * length_y;
nyp1 = (j+1) + (i - 1) * length_y;

rxm = (conductivity_mapping(i,j) + conductivity_mapping(i-1,j))/2;
rxp = (conductivity_mapping(i,j) + conductivity_mapping(i+1,j))/2;
rym = (conductivity_mapping(i,j) + conductivity_mapping(i,j-1))/2;

G_matrix(n,n) = -(rxm + rxp + rym);
G_matrix(n,nxm) = rxm;
G_matrix(n,nxp) = rxp;
G_matrix(n,nym) = rym;

else
nxm = j + (i-2)*num_y;
nxp = j + i*num_y;
nym = j-1 + (i-1)*num_y;
nyp = j+1 + (i-1)*num_y;
n1 = j + (i - 1) * length_y;
nxm1 = j + ((i-1) - 1) * length_y;
```

```
        nxp1 = j + ((i+1) - 1) * length_y;
        nym1 = (j-1) + (i - 1) * length_y;
        nyp1 = (j+1) + (i - 1) * length_y;
        rxm = (conductivity_mapping(i,j) + conductivity_mapping(i-1,j))/2;
        rxp = (conductivity_mapping(i,j) + conductivity_mapping(i+1,j))/2;
        ryp = (conductivity_mapping(i,j) + conductivity_mapping(i,j+1))/2;
        rym = (conductivity_mapping(i,j) + conductivity_mapping(i,j-1))/2;

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

        end
        end
end


V = G_matrix\B_matrix';
Voltage_map = zeros(num_x,num_y);
for i = 1:num_x
        for j = 1:num_y
        n = j +(i-1)*num_y;
        Voltage_map(i,j) = V(n);
        end
end

[X, Y] = meshgrid(0:meshspace:length_y,0:meshspace:width_x);
figure(6)
surf(X',Y',Voltage_map)
hold on
imagesc([0 length_y],[0 width_x],Voltage_map')
xlabel('x')
ylabel('y')
zlabel('electric potential')
title('V(x,y)')
hold off

[electricfield_y, electricfield_x] = gradient(Voltage_map,meshspace);
electricfield_x = -electricfield_x;
electricfield_y = -electricfield_y;

figure(7)
```

```
quiver(X',Y',electricfield_x,electricfield_y, 'm')
xlim([0 length_y])
ylim([0 width_x])
xlabel('x')
ylabel('y')
title('2-D Electric Field Vector Plot')
```

PART 3

```
%Spencer Tigere 101001717

%This section of codes finds the particle tractories for parts 2 c and 3 c
%This code also fid the Density plot for Part 3 a and the Average Current
%vs Bottleneck width in part 2 b). Execute the code twice to observe figures for 3 b
%and 3 c
clc
clear
set(0,'DefaultFigureWindowStyle','docked')

global C
global Ecount
global Vx Vy Vtotal x y
Ecount =1000;
C.mo = 9.10938215e-31;
C.k = 1.3806504e-23;
electron_charge = -1.60217662e-19;
Temperature =300;
effective_m = 0.26*C.mo;
Length = 200e-9;
Width = 100e-9;
Thermal_v = sqrt((2*C.k*Temperature)/effective_m);
time_step = 10e-15;
frame = 100*time_step;
x = zeros(Ecount, 2);
y = zeros(Ecount, 2);
Temperature = zeros(1,2);
Time = 0;
VisibleEcount = 50;
tmn = 0.2e-12;
PScat = 1 - exp(-time_step/tmn);
V_Histogram = zeros(Ecount, 1);
```

```
bottleneck_X = [80e-9 80e-9 120e-9 120e-9 80e-9];

bottleneck_Y1 = [100e-9 60e-9 60e-9 100e-9 100e-9];
bottleneck_Y2 = [40e-9 0 0 40e-9 40e-9];
Specular = true;
Inside_Box = true;
Mapping_S = 10e-9;
Density_Mapping = zeros(Width/Mapping_S, Length/Mapping_S);
Temperature_Mapping = zeros(Width/Mapping_S, Length/Mapping_S);

wid_x = 30;
len_y = 20;
change_x = Length/wid_x;
change_y = Width/len_y;
conduction_outside = 1;
conduction_inside = 01e-2;
conductivity = zeros(wid_x,len_y);
G_matrix = sparse (wid_x*len_y, wid_x*len_y);
V_matrix = zeros(1, wid_x*len_y);
Voltage_x = 0.1;

for i = 1:wid_x
        for j = 1:len_y
        n = j + (i - 1)*len_y;
        nxm = j + ((i-1) - 1)*len_y;
        nxp = j + ((i+1) - 1)*len_y;
        nym = (j-1) + (i - 1)*len_y;
        nyp = (j+1) + (i - 1)*len_y;
        if (i > (0.3*wid_x) || i < (0.6*wid_x)) && (j > (0.6*len_y) || j < (0.3*len_y))
        conductivity(i,j) = conduction_inside;
        else
        conductivity(i,j) = conduction_outside;
        end

        end
end
for i = 1:wid_x
        for j = 1:len_y
        n = j + (i - 1)*len_y;
        nxm = j + ((i-1) - 1)*len_y;
        nxp = j + ((i+1) - 1)*len_y;
        nym = (j-1) + (i - 1)*len_y;
        nyp = (j+1) + (i - 1)*len_y;
        if (i == 1)
```

```
n = j + (i - 1)*len_y;
nxm = j + ((i-1) - 1)*len_y;
nxp = j + ((i+1) - 1)*len_y;
nym = (j-1) + (i - 1)*len_y;
nyp = (j+1) + (i - 1)*len_y;
V_matrix(n) = Voltage_x;
G_matrix(n,n) = 1;
elseif (i == wid_x)
n = j + (i - 1)*len_y;
nxm = j + ((i-1) - 1)*len_y;
nxp = j + ((i+1) - 1)*len_y;
nym = (j-1) + (i - 1)*len_y;
nyp = (j+1) + (i - 1)*len_y;
V_matrix(n) = 0;
G_matrix(n,n) =1;
elseif (j == 1)
n = j + (i - 1)*len_y;
nxm = j + ((i-1) - 1)*len_y;
nxp = j + ((i+1) - 1)*len_y;
nym = (j-1) + (i - 1)*len_y;
nyp = (j+1) + (i - 1)*len_y;
G_matrix(n,n) = -((conductivity(i,j) + conductivity(i-1,j))/2) - ((conductivity(i,j) +
conductivity(i+1,j))/2) - ((conductivity(i,j) + conductivity(i,j+1))/2);
G_matrix(n, nxm) = ((conductivity(i,j) + conductivity(i-1,j))/2);
G_matrix(n,nxp) = ((conductivity(i,j) + conductivity(i+1,j))/2);
G_matrix(n, nyp) = ((conductivity(i,j) + conductivity(i,j+1))/2);
elseif (j == len_y)
n = j + (i - 1)*len_y;
nxm = j + ((i-1) - 1)*len_y;
nxp = j + ((i+1) - 1)*len_y;
nym = (j-1) + (i - 1)*len_y;
nyp = (j+1) + (i - 1)*len_y;
G_matrix(n,n) = -((conductivity(i,j) + conductivity(i-1,j))/2) - ((conductivity(i,j) +
conductivity(i+1,j))/2) - ((conductivity(i,j) + conductivity(i,j-1))/2);
G_matrix(n,nxm) = ((conductivity(i,j) + conductivity(i-1,j))/2);
G_matrix(n,nxp) = ((conductivity(i,j) + conductivity(i+1,j))/2);
G_matrix(n,nym) = ((conductivity(i,j) + conductivity(i,j-1))/2);
else
n = j + (i - 1)*len_y;
nxm = j + ((i-1) - 1)*len_y;
nxp = j + ((i+1) - 1)*len_y;
nym = (j-1) + (i - 1)*len_y;
nyp = (j+1) + (i - 1)*len_y;
```

```
        G_matrix(n,n) = -((conductivity(i,j) + conductivity(i-1,j))/2) - ((conductivity(i,j) +
conductivity(i+1,j))/2) - ((conductivity(i,j) + conductivity(i,j-1))/2) - ((conductivity(i,j) +
conductivity(i,j+1))/2);
        G_matrix(n,nxm) = ((conductivity(i,j) + conductivity(i-1,j))/2);
        G_matrix(n,nxp) = ((conductivity(i,j) + conductivity(i+1,j))/2);
        G_matrix(n,nym) = ((conductivity(i,j) + conductivity(i,j-1))/2);
        G_matrix(n,nyp) = ((conductivity(i,j) + conductivity(i,j+1))/2);
        end
        end
end

Solution = G_matrix\V_matrix';
surface = zeros(wid_x,len_y);
for i = 1:wid_x
        for j = 1:len_y
        n = j + (i - 1)*len_y;
        nxm = j + ((i-1) - 1) * len_y;
        nxp = j + ((i+1) - 1) * len_y;
        nym = (j-1) + (i - 1) * len_y;
        nyp = (j+1) + (i - 1) * len_y;
        surface(i,j) = Solution(n);
        end
end
[Electricfield_x, Electricfield_y] = gradient(-surface);
Force_x = electron_charge*Electricfield_x;
Force_y = electron_charge*Electricfield_y;
Acceleration_x = Force_x /effective_m;
Acceleration_y = Force_y /effective_m;

for i = 1:Ecount
        x(i,1) = rand()*200e-9;
        y(i,1) = rand()*100e-9;
        Inside_Box = true;
        while Inside_Box == true
        if (x(i) >= 40e-9 && x(i) <= 120e-9) && (y(i) >= 60e-9 ||...
        y(i) <= 40e-9)
        x(i,1) = rand * 200e-9;
        y(i,1) = rand * 100e-9;
        else
        Inside_Box = false;
        end
        end

end
```

```
for i = 1:Ecount

Vx(1:Ecount) = Thermal_v * randn;
Vy(1:Ecount) = Thermal_v * randn;
end

figure(8)
subplot(2,1,1);
plot(bottleneck_X, bottleneck_Y1, bottleneck_X, bottleneck_Y2)
axis([0 Length 0 Width]);
title('Particle Trajecories');
xlabel('x');
ylabel('y');
hold on;

while Time < frame
        subplot(2,1,1)
        for j = 1:Ecount
        leaking = true;
        if PScat> rand
        Vx(j) = Thermal_v * randn;
        Vy(j) = Thermal_v * randn;
        end
        x_index = round((x(j,2)/Length) * 30);
        y_index = round((y(j,2)/Width)*20);
        if x_index < 1
        x_index = 1;
        elseif x_index > 30
        x_index = 30;
        end
        if y_index < 1
        y_index = 1;
        elseif y_index > 20
        y_index = 20;
        end

        Vx(j) =  Vx(j) + Acceleration_x(x_index,y_index)*time_step;
        Vy(j) =  Vy(j) + Acceleration_y(x_index,y_index)*time_step;
        x(j,2) = x(j,1);
        y(j,2) = y(j,1);
        x(j,1) = x(j,1) + (time_step * Vx(j));
        y(j,1) = y(j,1) + (time_step * Vy(j));
```

```
if (x(j,1) >= 80e-9 && x(j,1) <= 120e-9) && y(j,1) >= 60e-9

if y(j,2) < 60e-9
        Vy(j) = -Vy(j);
        y(j,1) = 60e-9;
        y(j,2) = 60e-9;

elseif x(j,2) < 80e-9
        Vx(j) = -Vx(j);
        x(j,1) = 80e-9;
        x(j,2) = 80e-9;

elseif x(j,2) > 120e-9
        Vx(j) = -Vx(j);
        x(j,1) = 120e-9;
        x(j,2) = 120e-9;
end

if Specular == true

x(j,1) = x(j,2) + Vx(j)*time_step;
y(j,1) = y(j,2) + Vy(j)*time_step;
else

Vx(j) = Thermal_v * randn;
Vy(j) = Thermal_v * randn;

while leaking == true
if(x(j,2) < 80e-9 && Vx(j) >= 0) || ...
        (x(j,2) > 120e-9 && Vx(j) <= 0) || ...
        (y(j,2) < 60e-9 && Vy(j) >= 0)
        Vx(j) = Thermal_v * randn;
        Vy(j) = Thermal_v * randn;
else
        leaking = false;
end
end
x(j,1) = x(j,2) + Vx(j)*time_step;
y(j,1) = y(j,2) + Vy(j)*time_step;
end
end
if (x(j,1) >= 80e-9 && x(j,1) <= 120e-9) && y(j,1) <= 40e-9
if y(j,2) > 40e-9
        Vy(j) = -Vy(j);
```

```
            y(j,1) = 40e-9;
            y(j,2) = 40e-9;
elseif x(j,2) < 80e-9
            Vx(j) = -Vx(j);
            x(j,1) = 80e-9;
            x(j,2) = 80e-9;
elseif x(j,2) > 120e-9
            Vx(j) = -Vx(j);
            x(j,1) = 120e-9;
            x(j,2) = 120e-9;
end
if Specular == true
x(j,1) = x(j,2) + Vx(j)*time_step;
y(j,1) = y(j,2) + Vy(j)*time_step;
else
Vx(j) = Thermal_v * randn;
Vy(j) = Thermal_v * randn;
while leaking == true
if(x(j,2) < 80e-9 && Vx(j) >= 0) || ...
            (x(j,2) > 120e-9 && Vx(j) <= 0) || ...
            (y(j,2) > 40e-9 && Vy(j) <= 0)
            Vx(j) = Thermal_v * randn;
            Vy(j) = Thermal_v * randn;
else
            leaking = false;
end
end
x(j,1) = x(j,2) + Vx(j)*time_step;
y(j,1) = y(j,2) + Vy(j)*time_step;
end
end
if x(j,1) > Length
x(j,2) = 0;
x(j,1) = time_step * Vx(j);
end
if x(j,1) < 0
x(j,2) = Length;
x(j,1) = x(j,2) + (time_step * Vx(j));
end
if y(j,1) > Width || y(j,1) < 0
Vy(j) = -Vy(j);
end
XPlot = [x(j,2) x(j,1)];
YPlot = [y(j,2) y(j,1)];
```

```
        if j < VisibleEcount
        plot(XPlot,YPlot);
        end

        VTotal = sqrt(Vx(j)^2 + Vy(j)^2);
        end

        AvgTemperature = Temperature(1,2)/Ecount;
        TemperaturePlot = [Temperature(1,1) AvgTemperature];
        TimePlot = [(Time - time_step) Time];
        subplot(2,1,2);
        plot(TimePlot, TemperaturePlot);
        Temperature(1,1) = AvgTemperature;
        AvgTemperature = 0;
        Temperature(1,2) = 0;
        pause(1e-19)
        Time = Time + time_step;
end
for i = 1:(Length/Mapping_S)
        for j = 1:(Width/Mapping_S)
        for m = 1:Ecount
        if(x(m,1) > Mapping_S*(i -1)) && ...
                (x(m,1) < Mapping_S*(i)) && ...
                (y(m,1) > Mapping_S*(j - 1)) && ...
                (y(m,1) < Mapping_S*(j))

        Vtotal(m) = sqrt(Vx(m)^2 + Vy(m)^2);

        Density_Mapping(j, i) = Density_Mapping(j, i) + 1;
        Temperature_Mapping(j, i) = Temperature_Mapping(j,i) + ...
                (effective_m*Vtotal(m)^2)/(2*C.k);
        end
        Temperature_Mapping(j,i) = Temperature_Mapping(j,i)/Density_Mapping(j,i);
        end
        end
end
figure(9)
imagesc(Density_Mapping)
xlabel('x');
ylabel('y');
set(gca, 'Ydir', 'Normal')
title('Denisty Plot')
```