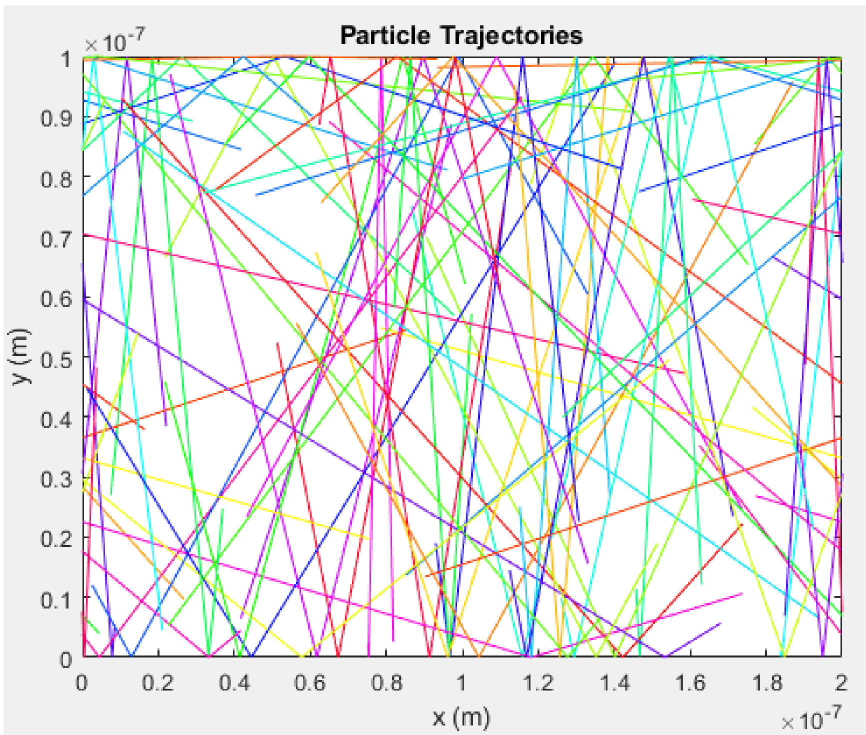
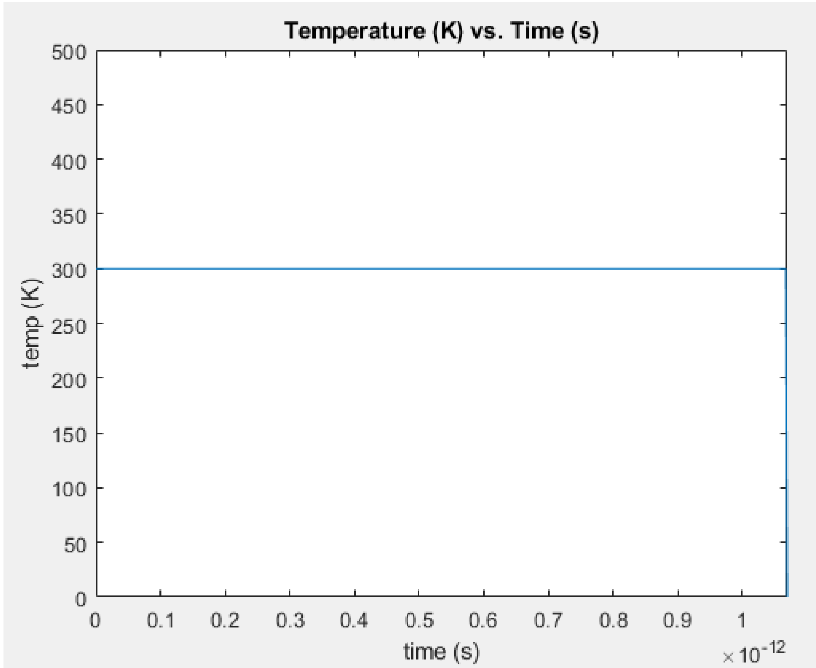
```
%% ELEC 4700 - Assignment 1
%
%% Monte-Carlo Modeling of Electron Transport
%
% Jacob Godin - 100969991
%
%% --------------------------------------------------------------------------
%
% Modeling of the carriers as a population of electrons in an N-type Si
% semiconductor crystal.
%
% Effective mass of electrons mn = 0.26m0
% Nominal size of the region is 200nm X 100 nm
%
% --------------------------------------------------------------------------
clear all; clc;

global mn, global k, global T;
m0 = 9.11e-31;
mn = 0.26*m0;
dim_x = 200e-9;
dim_y = 100e-9;

k = 1.38064852e-23;

%% Part 1: Electron Modelling
%
% Part 1 - Question 1: What is the thermal velocity Vth? Assume T = 300K
%
% Vth = sqrt(vx^2 + vy^2) = sqrt(2kT/mn) = 1.8701e+05 K

T = 300;
Vth = sqrt(2*k*T/mn);

%% Part 1 - Question 2: If the mean time between collisions is Tmn = 0.2ps
what is
% the mean free path?
%
% Mean free path = Tmn * Vth = 3.7403e-08 s
Tmn = 0.2e-12;
Mfp = Tmn * Vth;

%% Part 1 - Question 3: Write a pogram that will model the random motion of
electrons
% TODO: optimize calculations

num_e = 10; % number of electrons

% initialize x and y position of electrons
[x_vec, y_vec] = initPosition(num_e, dim_x, dim_y);

% initialize x and y velocity of electrons
[vx_vec, vy_vec] = initVelocity(num_e, Vth);

% initialize time variables
```

```
t = 0;
steps = 250;
t_step = max(dim_x, dim_y)/(1000*Vth);
t_final = steps*t_step;
t_vec = zeros(1,steps+1);

% initialize time vector with time step
t_vec = zeros(1,length(t_vec));
for i=1:length(t_vec)
    t_vec(i) = (i-1)*t_step;
end

colour = hsv(num_e);
j=0;
Temp = zeros(1,length(t_vec));
while t < t_final
    j=j+1;
    % Calculate temp
    Temp(j) = (mean(vx_vec.^2 + vy_vec.^2)*mn)/(2*k);

    % Save previous positions
    x_vec_prev = x_vec;
    y_vec_prev = y_vec;

    % Calculate new position
    x_vec = x_vec + vx_vec*t_step;
    y_vec = y_vec + vy_vec*t_step;

    % Boundary conditions
    for i=1:num_e
        if x_vec(i) < 0 % left boundary, periodic
            x_vec(i) = x_vec(i)+dim_x;
            x_vec_prev(i) = dim_x;
        end
        if x_vec(i) > dim_x % right boundary, periodic
            x_vec(i) = x_vec(i)-dim_x;
            x_vec_prev(i) = 0;
        end
        if y_vec(i) > dim_y % top boundary, reflect
            vy_vec(i) = -vy_vec(i);
            y_vec(i) = 2*dim_y - y_vec(i);
        end
        if y_vec(i) < 0 % bottom boundary, reflect
            vy_vec(i) = -vy_vec(i);
            y_vec(i) = abs(y_vec(i));
        end
    end

    % Plot trajectories
    figure(1);
    xlabel('x (m)')
    ylabel('y (m)')
    title('Particle Trajectories')
    xlim([0 dim_x])
    ylim([0 dim_y])
    pause(0.1)
```

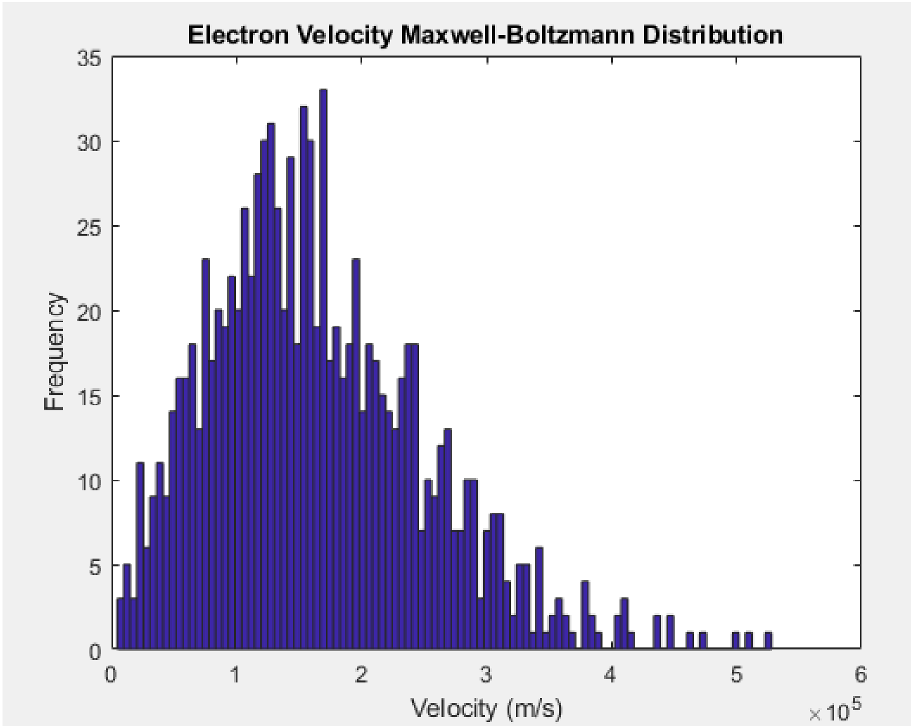
```
    for i=1:num_e
        
plot([x_vec_prev(i);x_vec(i)],[y_vec_prev(i);y_vec(i)],'color',colour(i,:))
        hold on
    end
    
    t=t+t_step;
end

% Trajectory Plot
```




```
% Plot temp
figure(2);
plot(t_vec, Temp)
xlabel('time (s)')
ylabel('temp (K)')
ylim([0 500])
xlim([0 t_final])
title('Temperature (K) vs. Time (s)')
```

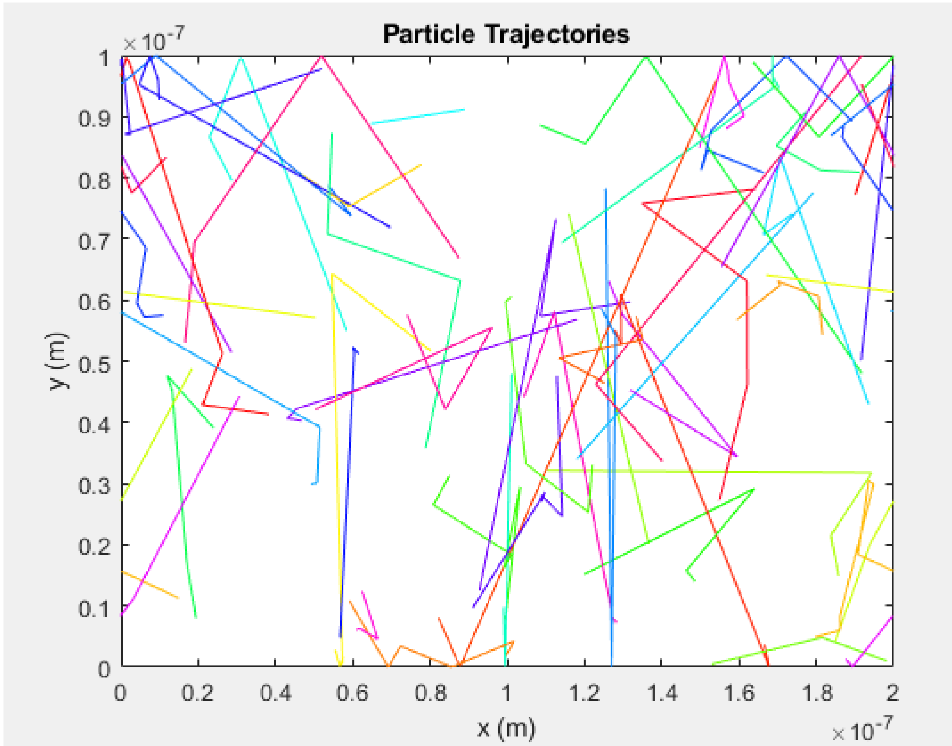



```
% As seen from the plot, the temperature remains constant as time progresses.
% The velocity component of the electrons remain in even distribution such
% that the average is 300K.

% -----------------------------------------------------------------------
%
%% Part 2: Collisions with Mean Free Path (MFP)
%
% Part 2 - Question 1: Assign a random velocity to each of the particles at
the
% start. Use a Maxwell-Boltzmann distribution for each velocity component.
% The average of all speeds will be Vth. Plot distribution in a histogram.
%
% -----------------------------------------------------------------------

num_e = 10;
dim_x = 200e-9;
dim_y = 100e-9;

% Initialize new x and y positions
[x_vec, y_vec] = initPosition(num_e, dim_x, dim_y);

% Initialize new vx and vy velocities in a Maxwell-Boltzmann distribution.
The distribution can be taken as a Gaussian distribution with a standard
deviation of sqrt(k*T/mn)
[vx_vec, vy_vec] = initBoltDist(num_e);

figure(3);
hist(sqrt(vx_vec.^2 + vy_vec.^2), 100);
title('Electron Velocity Maxwell-Boltzmann Distribution');
xlabel('Velocity (m/s)');
ylabel('Frequency');
```

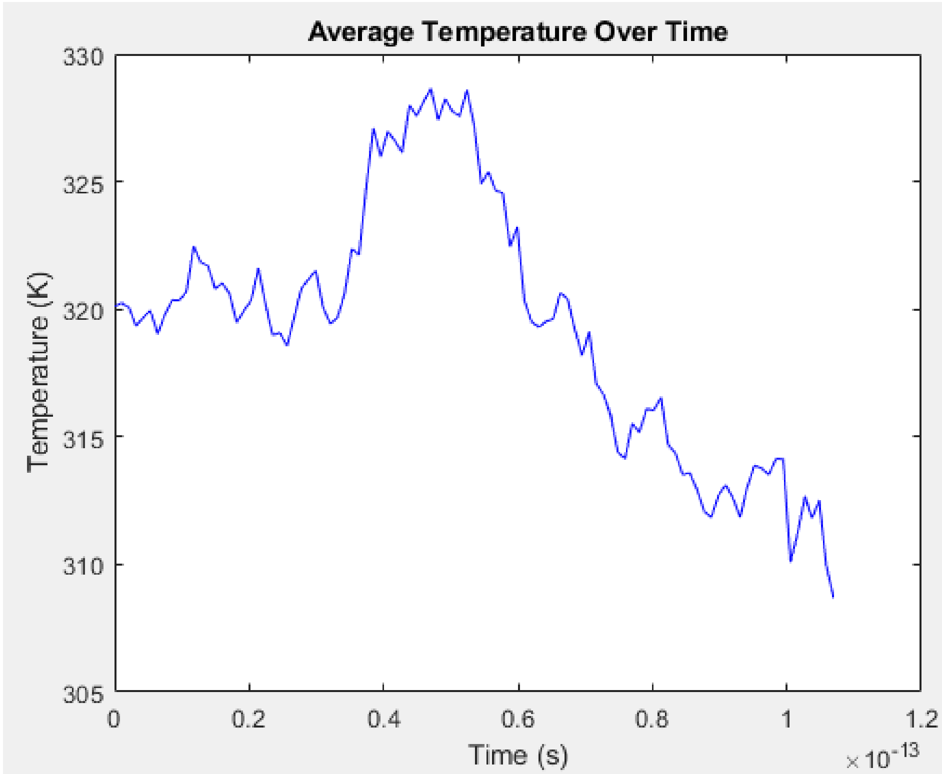



```
V_avg = mean(sqrt(vx_vec.^2 + vy_vec.^2));
Vth = sqrt(2*k*T/mn);

%% Part 2 - Question 2: Model scattering of electrons using exponential
scattering
% probability: P = 1 - exp(-dt/Tmn) where dt is the time since last time
% step. Uf P > rand() then the particle scatters. When the electron
% scatters, re-thermalize its velocities and assign new velocities Vx and
% Vy from the Maxwell-Boltzmann distributions.

P_scatter = 1 - exp(-t_step/Tmn);

t = 0;

colour = hsv(num_e);

j=0;
while t < t_final
    j=j+1;
    % Calculate new velocity if electron scatters
    for i=1:length(x_vec)
        if P_scatter > rand()
            [vx_vec(i), vy_vec(i)] = newBoltDist();
        end
    end

    % Save previous positions
    x_vec_prev = x_vec;
    y_vec_prev = y_vec;
```

```matlab
    % Calculate new position
    x_vec = x_vec + vx_vec*t_step;
    y_vec = y_vec + vy_vec*t_step;

    % Boundary conditions
    for i=1:num_e
        if x_vec(i) < 0 % left boundary, periodic
            x_vec(i) = x_vec(i)+dim_x;
            x_vec_prev(i) = dim_x;
        end
        if x_vec(i) > dim_x % right boundary, periodic
            x_vec(i) = x_vec(i)-dim_x;
            x_vec_prev(i) = 0;
        end
        if y_vec(i) > dim_y % top boundary, reflect
            vy_vec(i) = -vy_vec(i);
            y_vec(i) = 2*dim_y - y_vec(i);
        end
        if y_vec(i) < 0 % bottom boundary, reflect
            vy_vec(i) = -vy_vec(i);
            y_vec(i) = abs(y_vec(i));
        end
    end

    % Calculate temperature
    Vsq = vx_vec.^2 + vy_vec.^2;
    Temp(j) = (mean(Vsq)*mn)/(2*k);

    % Plot trajectories
    figure(4);
    xlabel('x (m)')
    ylabel('y (m)')
    title('Particle Trajectories')
    xlim([0 dim_x])
    ylim([0 dim_y])
    pause(0.1)

    for i=1:num_e

plot([x_vec_prev(i);x_vec(i)],[y_vec_prev(i);y_vec(i)],'color',colour(i,:))
        hold on
    end


    t=t+t_step;
end

% Particle Trajectories with Scatter Probability
```




```
figure(5);
plot(t_vec,Temp, 'b')
xlabel('Time (s)');
ylabel('Temperature (K)');
title('Average Temperature Over Time');
```

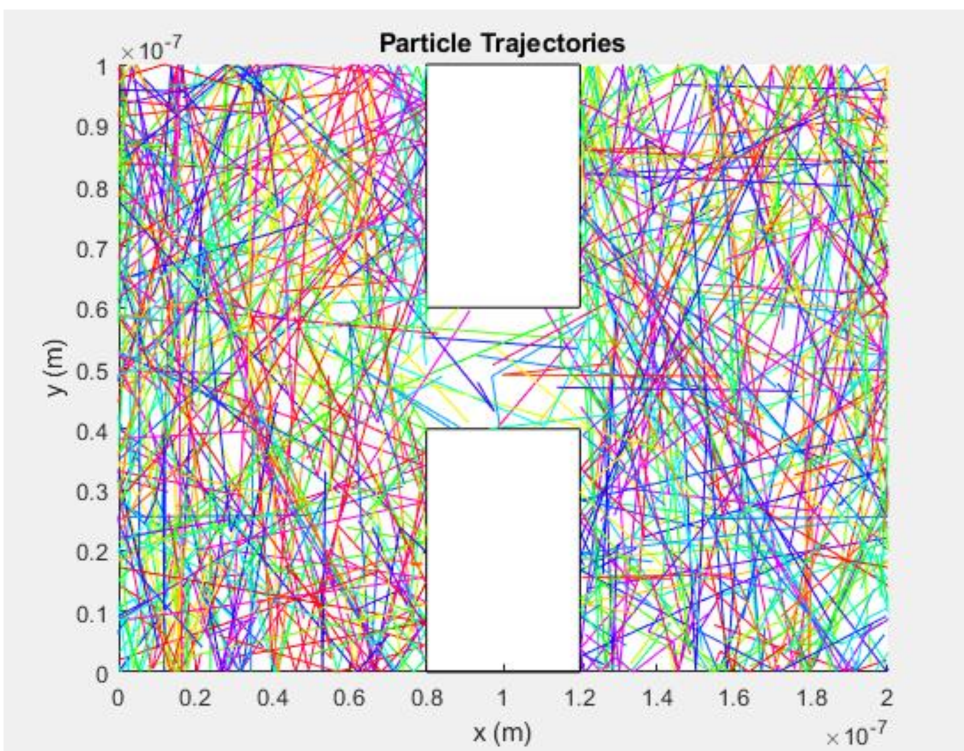



```
% ----------------------------------------------------------------------
%
%% Part 3: Enhancements
%
% Part 3 - Question 1: Add an inner rectangle "bottle neck" boundary.
%
% ----------------------------------------------------------------------

% Define box limits
box_x = 40;
box_y = 40;

% Box limits no greater than 120x40
if box_x > 120
    boy_x = 120;
end

if box_y > 40
    box_y = 40;
end

box_x = box_x*1e-9;
box_y = box_y*1e-9;

lim_x_low = (dim_x/2)-(box_x/2);
lim_x_high = (dim_x/2)+(box_x/2);

% Draw boxes on figure
figure(6);
```

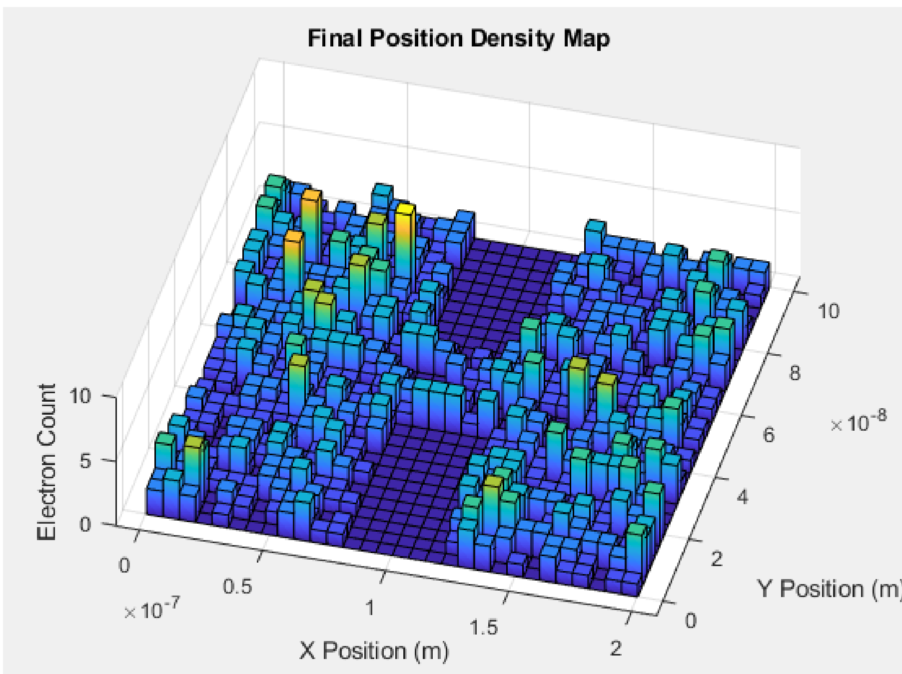
```
hold on;
rectangle('position', [lim_x_low 0 box_x box_y]);
rectangle('position', [lim_x_low dim_y-box_y box_x box_y]);
axis([0 dim_x 0 dim_y])

% Initialize x and y positions of electrons
[x_vec, y_vec] = initPositionOutsideBox(num_e, dim_x, dim_y, box_x, box_y);

% Initialize x and y velocities
[vx_vec, vy_vec] = initBoltDist(num_e);

t = 0;
mode = 1; %1 for specular, 0 for diffusive boundaries

P_scatter = 1 - exp(-t_step/Tmn);

t_step = max(dim_x, dim_y)/(250*Vth);
steps = 100;
t_final = steps*t_step;

while t < t_final

    % Calculate new velocity if electron scatters
    for i=1:length(x_vec)
        if P_scatter > rand()
            [vx_vec(i), vy_vec(i)] = newBoltDist();
        end
    end

    % Save previous positions
    x_vec_prev = x_vec;
    y_vec_prev = y_vec;

    % Calculate new position
    x_vec = x_vec + vx_vec*t_step;
    y_vec = y_vec + vy_vec*t_step;

    for i=1:num_e
        if mode == 1
            % Boundary conditions for semiconductor edges
            if x_vec(i) < 0 % left boundary, periodic
                x_vec(i) = x_vec(i)+dim_x;
                x_vec_prev(i) = dim_x;
            end
            if x_vec(i) > dim_x % right boundary, periodic
                x_vec(i) = x_vec(i)-dim_x;
                x_vec_prev(i) = 0;
            end
            if y_vec(i) > dim_y % top boundary, reflect
                vy_vec(i) = -vy_vec(i);
                y_vec(i) = 2*dim_y - y_vec(i);
            end
            if y_vec(i) < 0 % bottom boundary, reflect
                vy_vec(i) = -vy_vec(i);
                y_vec(i) = abs(y_vec(i));
```

```
            end
            % Boundary conditions for box edges
            % if x_vec(i) hits left or right edge of box between box y
            % boundaries, reflect
            if (x_vec(i) > lim_x_low && x_vec(i) < lim_x_high) && ~(y_vec(i)
> box_y && y_vec(i) < (dim_y-box_y))
                vx_vec(i) = -vx_vec(i);

                % Remove particles from box
                x_vec(i) = x_vec_prev(i);
                y_vec(i) = y_vec_prev(i);
            end
            % if y_vec(i) hits bottom or top edge of box between box x
            % boundaries, reflect
            if (y_vec(i) < box_y && y_vec(i) > dim_y-box_y) && (x_vec(i) >
lim_x_low && x_vec(i) < lim_x_high)
                vy_vec(i) = -vy_vec(i);

                % Remove particles from box
                x_vec(i) = x_vec_prev(i);
                y_vec(i) = y_vec_prev(i);
            end

        elseif mode == 0
            % Boundary conditions for semiconductor edges
            if x_vec(i) < 0 % left boundary, periodic
                x_vec(i) = x_vec(i)+dim_x;
                x_vec_prev(i) = dim_x;
            end
            if x_vec(i) > dim_x % right boundary, periodic
                x_vec(i) = x_vec(i)-dim_x;
                x_vec_prev(i) = 0;
            end
            if y_vec(i) > dim_y % top boundary, reflect
                vy_vec(i) = -vy_vec(i);
                y_vec(i) = 2*dim_y - y_vec(i);
            end
            if y_vec(i) < 0 % bottom boundary, reflect
                vy_vec(i) = -vy_vec(i);
                y_vec(i) = abs(y_vec(i));
            end
            % Boundary conditions for box edges
            % if x_vec(i) hits left or right edge of box between box y
            % boundaries, reflect
            if (x_vec(i) > lim_x_low && x_vec(i) < lim_x_high) && ~(y_vec(i)
> box_y && y_vec(i) < (dim_y-box_y))
                vy_vec(i) = newBoltDist();
                vx_vec(i) = newBoltDist();

                % Remove particles from box
                x_vec(i) = x_vec_prev(i);
                y_vec(i) = y_vec_prev(i);
            end
            % if y_vec(i) hits bottom or top edge of box between box x
            % boundaries, reflect
```

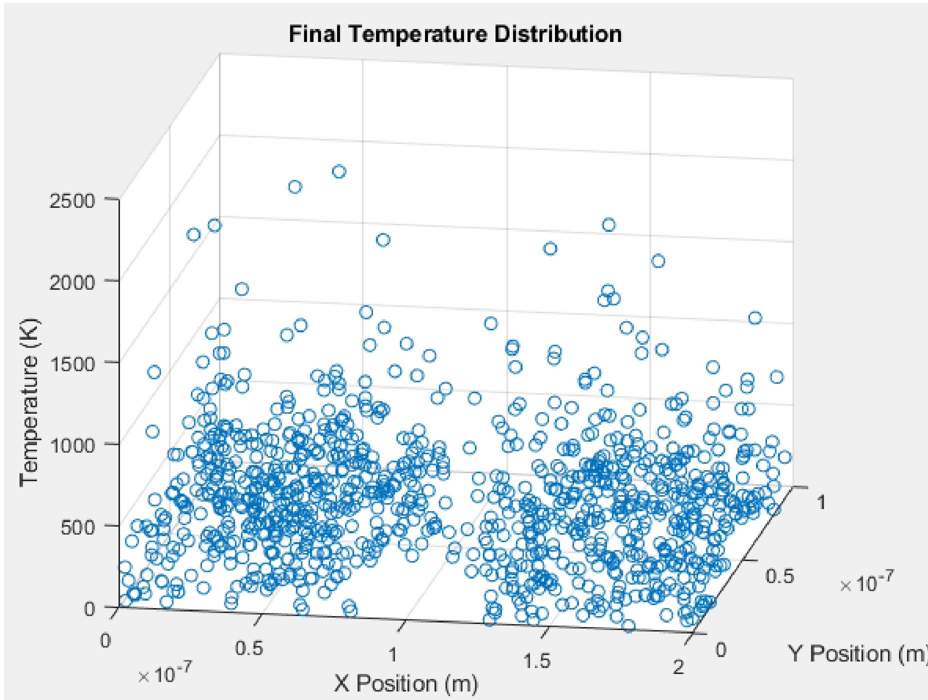
```
            if (y_vec(i) < box_y && y_vec(i) > dim_y-box_y) && (x_vec(i) >
lim_x_low && x_vec(i) < lim_x_high)
                vy_vec(i) = newBoltDist();
                vx_vec(i) = newBoltDist();

                % Remove particles from box
                x_vec(i) = x_vec_prev(i);
                y_vec(i) = y_vec_prev(i);
            end
        end
    end


    % Plot trajectories
    xlabel('x (m)')
    ylabel('y (m)')
    title('Particle Trajectories')
    pause(0.1)

    for i=1:num_e

plot([x_vec_prev(i);x_vec(i)],[y_vec_prev(i);y_vec(i)],'color',colour(i,:))
        hold on
    end

    t=t+t_step;
end

% Particle Trajectories Plot with Scattering Probability and Boxes
```



```
%% Density Map
Z = [transpose(x_vec), transpose(y_vec)];
figure(7);
hist3(Z, [30,30]);
hold on;

surfHandle = get(gca, 'child');
set(surfHandle,'FaceColor','interp', 'CdataMode', 'auto');

view(15,70);
grid on;
title('Final Position Density Map')
xlabel('X Position (m)');
ylabel('Y Position (m)');
zlabel('Electron Count');
hold off;
```




```
%% Temperature Map
figure(8);
Vsq = vx_vec.^2 + vy_vec.^2;
Temp = (Vsq.*mn)./(2*k);
a = scatter3(x_vec, y_vec, Temp);
title('Final Temperature Distribution')
xlabel('X Position (m)');
ylabel('Y Position (m)');
zlabel('Temperature (K)');
view(10, 20);
grid on;
```



Final Temperature Distribution
Temperature (K)
2500
2000
1500
1000
500
0
0
0.5
1
1.5
2
×10-7
X Position (m)
0
0.5
1
×10-7
Y Position (m)