```matlab
%%Adam Heffernan 100977570 Assignment 1 ELEC 4700. Completed on
2/1/2020.

clc
close all

m_0 = 9.10938e-31; %Mass of Electron
m = 0.26*m_0;%Effective mass in silicon
T = 300;%temperature in Kelvin
k = 1.38064e-23;%Boltzmans Constant
v_th = sqrt((2*k*T)/m); %Thermal velocity of particles
mean = 0.2e-12;%Mean value
width = 100e-9;%Width of Wafer
length = 200e-9;%Length of the channel
par = 10000;%Particles
pop = 10; %Mapping population
iter = 200;%Iterations
pos_velo = zeros(par,4);%Initializing position velocity Matrix
temp = zeros(iter,1);%Initializing position velocity Matrix
traj = zeros(iter, 2*pop);%Initializing trajectory Matrix
step = width/v_th/100;%Calculating step size for the movement of
particles
boundaries_len = false(1); %Either diffusive or specular boundaries
true for specular false for diffusive
boundaries_width = false(1);%Either diffusive or specular boundaries
true for specular false for diffusive
total_scatters = 0;
%Calculating mean free path for part 1
l=v_th*mean;
%Outputting mean free path and thermal velocity for part 1
fprintf('Thermal Velocity for part 1 is %f km/s\n',v_th/10^3)
fprintf('Mean free path for part 1 is %f nm\n',l/10^-9);
%%Question 1
for j = 1:par
    %Creating a position, velocity vector in the x and y directions
for
    %the number of particles in the simulation.
    ang = rand*2*pi;
    pos_velo(j,:) = [length*rand width*rand v_th*cos(ang)
v_th*sin(ang)];
end

for j = 1:iter
    %Updating the position velocity vector at each iteration. Using
step
    %size * position velocity vector of each particle at each
iteration in
    %time plus the previous X and Y locations of the particle
respectively.
    %
    pos_velo(:,1:2) = pos_velo(:,1:2) + step*pos_velo(:,3:4);
    %
```

```matlab
    %Left Bound Check
    i = pos_velo(:,1) > length;
    pos_velo(i,1) = pos_velo(i,1) - length;
    %
    %Right Bound Check
    %
    i = pos_velo(:,1) < 0;
    pos_velo(i,1) = pos_velo(i,1) + length;
    %
    %Top Bound Check
    i = pos_velo(:,2) > width;
    pos_velo(i,2) = 2*width - pos_velo(i,2);
    pos_velo(i,4) = -pos_velo(i,4);
    %
    %Bottom Bound Check
    i = pos_velo(:,2) < 0;
    pos_velo(i,2) = -pos_velo(i,2);
    pos_velo(i,4) = -pos_velo(i,4);
    %
    %Storing the Temperature value of the semiconductor at each
  iteration
    temp(j) = (sum(pos_velo(:,3).^2) + sum(pos_velo(:,4).^2))*m/k/2/
par;
    %Storing the Trajectory value of the semiconductor at each
  iteration
    traj(1:iter,2*j:2*j+1) = [pos_velo(1:iter,1) pos_velo(1:iter,2)];
    figure(1);
    subplot(2,1,1)
    %Plotting each particle in the population at each time step
    plot(pos_velo(1:pop,1)./1e-9, pos_velo(1:pop,2)./1e-9, 'o');
    axis([0 length/1e-9 0 width/1e-9]);
    title('Simulation of Electrons in Silicon Crystal')
    xlabel('(nm)')
    ylabel('(nm)')
    subplot(2,1,2)
    %Plotting the temperature of the Silicon crystal over time
    plot(step*(0:j-1), temp(1:j),'Color',[0.1 0.1 0.1]);
    axis([0 step*iter 200 400]);
    title('Temperature of Electrons in Silicon Crystal')
    xlabel('Time(s)')
    ylabel('Temperature(K)')
    hold on;
end


for i = 1:pop

    %Setting a Random colour each particle
    color = [rand rand rand];
    figure(2)
    %Plotting the trajectory vector of each particle in the shown
    %population
    plot(traj(i,2:2:end)./1e-9,
 traj(i,1:2:end-1)./1e-9, '.', 'Color',color);
```

```matlab
    axis([0 length/1e-9 0 width/1e-9]);
    title('Trajectories of Electrons in Silicon Crystal')
    xlabel('(nm)')
    ylabel('(nm)')
    hold on;
    pause(0.5);

end
%%Question 2
%Calculating the probability of a scattering particle
scat = 1 - exp(-step/mean);
%Calculating a normal maxwell-boltzmann distribution
v_boltz = makedist('Normal','mu',0,'sigma',sqrt(k*T/m));

for j = 1:par
    %Creating a position, velocity vector in the x and y directions
 for
    %the number of particles in the simulation.
    ang = rand*2*pi;
    pos_velo(j,:) = [length*rand width*rand random(v_boltz)
 random(v_boltz)];
end


for j = 1:iter
    %Updating the position velocity vector at each iteration. Using
 step
    %size * position velocity vector of each particle at each
 iteration in
    %time plus the previous X and Y locations of the particle
 respectively.
    %
    pos_velo(:,1:2) = pos_velo(:,1:2) + step*pos_velo(:,3:4);
    %
    %Left Bound Check
    %
    i = pos_velo(:,1) > length;
    pos_velo(i,1) = pos_velo(i,1) - length;
    %
    %Right Bound Check
    %
    i = pos_velo(:,1) < 0;
    pos_velo(i,1) = pos_velo(i,1) + length;
    %
    %Top Bound check
    %
    i = pos_velo(:,2) > width;
    pos_velo(i,2) = 2*width - pos_velo(i,2);
    pos_velo(i,4) = -pos_velo(i,4);
    %
    %Bottom Bound check
    %
    i = pos_velo(:,2) < 0;
    pos_velo(i,2) = -pos_velo(i,2);
```

```matlab
        pos_velo(i,4) = -pos_velo(i,4);
        %Creating a random number t for each particle in the population
     that is
        %the probability of this particle scattering at this time
     iteration
        t=rand(pop,1);
        %Setting i equal to a logical index array with the amount of
     colisions.
        i= t < scat;
        pos_velo(i,3:4) = random(v_boltz, [sum(i),2]);
        %Incrementing the total number of scattered particles
        total_scatters = total_scatters + count_scatters(i);
        %Storing the Temperature value of the semiconductor at each
     iteration
        temp(j) = (sum(pos_velo(:,3).^2) + sum(pos_velo(:,4).^2))*m/k/2/
par;
        %Storing the Trajectory value of the semiconductor at each
     iteration
        traj(1:iter,2*j:2*j+1) = [pos_velo(1:iter,1) pos_velo(1:iter,2)];
        figure(3)
        subplot(3,1,1)
        %Plotting each particle in the population at each time step
        plot(pos_velo(1:pop,1)./1e-9, pos_velo(1:pop,2)./1e-9, 'o')
        axis([0 length/1e-9 0 width/1e-9])
        title('Simulation of Electrons in Silicon Crystal')
        xlabel('(nm)')
        ylabel('(nm)')
        subplot(3,1,2)
        %Plotting the temperature of the Silicon crystal over time
        plot(step*(0:j-1), temp(1:j),'Color',[0.1 0.1 0.1])
        axis([0 step*iter 200 400]);
        title('Temperature of Electrons in Silicon Crystal')
        xlabel('Time(s)')
        ylabel('Temperature(K)')
        hold on;
    end

%Calculate the average time
t_mn = ((step*iter*pop)/total_scatters);
%Calculate the average velocity
velocity_average= sqrt(sum((pos_velo(:,3).^2))/par +
 sum(pos_velo(:,4).^2)/par);
%Calculate the mean free path
mfp = (t_mn * velocity_average)*10^9;
%Calculate the average temperature
av_temp = sum(temp)/iter;
%Output the average temperature
fprintf('Average Temperature in the Crystal for part 2 is %f K
 \n',av_temp)
%Output the average velocity
fprintf('Average Electron Velocities in Silicon Crystal for part 2 are
 %f km/s \n',velocity_average/10^3)
%Output the average time between colisions
fprintf('Average time for part 2 is %f ps\n',t_mn/10^-12);
```

```matlab
%Output the mean free path
fprintf('Mean free Path for part 2 is %f nm\n',mfp);
for i = 1:pop

    %Plotting the trajectory vector of each particle in the shown
    %population
    %Setting a Random colour each particle
    color = [rand rand rand];
    figure(4);
    plot(traj(i,2:2:end)./10^-9,
 traj(i,1:2:end-1)./10^-9, '.', 'Color',color)
    axis([0 length/1e-9 0 width/1e-9])
    title('Trajectories of Electrons in Silicon Crystal')
    xlabel('(nm)')
    ylabel('(nm)')
    hold on;
    pause(0.5)

end
figure(3);
subplot(3,1,3);
%Plotting Histogram of Velocities in the Silicon Crystal
velocity = sqrt(pos_velo(:,3).^2 + pos_velo(:,4).^2);
histogram(velocity)
title('Histogram of Electron Velocities in Silicon Crystal')
xlabel('Velocity (km/s)')
ylabel('Frequency')
%%Question 3
%Creating Box Parameters for boxes shown in plots
x1_box1=80;
x2_box1=120;
y1_box1=0;
y2_box1=40;
x1_box2=80;
x2_box2=120;
y1_box2=60;
y2_box2=100;
%Creating Box Parameters
boxes = 1e-9.*[x1_box1 x2_box1 y1_box1 y2_box1; x1_box2 x2_box2
 y1_box2 y2_box2];
%Boxes are specular or diffusive
boxes_specular = [0 1];
for j = 1:par
    %Creating a position, velocity vector in the x and y directions
 for
    %the number of particles in the simulation.
    ang = rand*2*pi;
    pos_velo(j,:) = [length*rand width*rand random(v_boltz)
 random(v_boltz)];
    while(in_box(pos_velo(j,1:2), boxes))
        %Checking for Particles in box, when particles are in
        %box we will reset there x and y position to a random
        %value
        pos_velo(j,1:2) = [length*rand width*rand];
```

```matlab
        end
    end


    for j = 1:iter
        %
        %Updating the position velocity vector at each iteration. Using
     step
        %size * position velocity vector of each particle at each
     iteration in
        %time plus the previous X and Y locations of the particle
     respectively.
        %
        pos_velo(:,1:2) = pos_velo(:,1:2) + step*pos_velo(:,3:4);


        if boundaries_len && boundaries_width
            %Left Bound Check
            i = pos_velo(:,1) > length;
            pos_velo(i,1) = 2*length - pos_velo(i,1) ;
            pos_velo(i,3) = -pos_velo(i,3);
            %
            %Right Bound Check
            %
            i = pos_velo(:,1) < 0;
            pos_velo(i,1) = -pos_velo(i,1);
            pos_velo(i,3) = -pos_velo(i,3);
            %
            %Top Bound Check
            i = pos_velo(:,2) > width;
            pos_velo(i,2) = 2*width - pos_velo(i,2);
            pos_velo(i,4) = -pos_velo(i,4);
            %
            %Bottom Bound Check
            i = pos_velo(:,2) < 0;
            pos_velo(i,2) = -pos_velo(i,2);
            pos_velo(i,4) = -pos_velo(i,4);
            %
        elseif boundaries_len
            %Left Bound Check
            i = pos_velo(:,1) > length;
            pos_velo(i,1) = 2*length - pos_velo(i,1) ;
            pos_velo(i,3) = -pos_velo(i,3);
            %
            %Right Bound Check
            %
            i = pos_velo(:,1) < 0;
            pos_velo(i,1) = -pos_velo(i,1);
            pos_velo(i,3) = -pos_velo(i,3);
            %
            %Top Bound Check
            i = pos_velo(:,2) > width;
            pos_velo(i,2) =  pos_velo(i,2) - width;
```

```matlab
            %
            %Bottom Bound Check
            i = pos_velo(:,2) < 0;
            pos_velo(i,2) = -pos_velo(i,2) + width;
            %
        elseif boundaries_width
            %Left Bound Check
            i = pos_velo(:,1) > length;
            pos_velo(i,1) = pos_velo(i,1) - length;
            %
            %Right Bound Check
            %
            i = pos_velo(:,1) < 0;
            pos_velo(i,1) = pos_velo(i,1) + length;
            %
            %Top Bound Check
            i = pos_velo(:,2) > width;
            pos_velo(i,2) = 2*width - pos_velo(i,2);
            pos_velo(i,4) = -pos_velo(i,4);
            %
            %Bottom Bound Check
            i = pos_velo(:,2) < 0;
            pos_velo(i,2) = -pos_velo(i,2);
            pos_velo(i,4) = -pos_velo(i,4);
            %
        else
            %Left Bound Check
            i = pos_velo(:,1) > length;
            pos_velo(i,1) = pos_velo(i,1) - length;
            %
            %Right Bound Check
            %
            i = pos_velo(:,1) < 0;
            pos_velo(i,1) = pos_velo(i,1) + length;
            %
            %Top Bound Check
            i = pos_velo(:,2) > width;
            pos_velo(i,2) =  pos_velo(i,2) - width;
            %
            %Bottom Bound Check
            i = pos_velo(:,2) < 0;
            pos_velo(i,2) = -pos_velo(i,2) + width;
            %
        end

    %Creating a random number i for each particle in the population
that is
    %the probability of this particle scattering at this time
iteration
    %Setting i equal to a logical index array with the amount of
colisions.
    i= rand(pop,1) < scat;
    pos_velo(i,3:4) = random(v_boltz, [sum(i),2]);
    total_scatters = total_scatters + count_scatters(i) ;
```

```matlab
    for i=1:pop
        %Checking for each particle in the population to see if it has hit
        %a boundary of the box we created above.
        box_num = in_box(pos_velo(i,1:2), boxes);
        x = 0;
        updated_x = 0;
        y = 0;
        updated_y = 0;
        while(box_num ~= 0)
            %If the electron come back that it has hit a boundary then we
            %need to see what boundary it hit in order to predict its next
            %move.

             if(pos_velo(i,3) > 0)
                x = pos_velo(i,1) - boxes(box_num,1);
                updated_x = boxes(box_num,1);
             else
                x = boxes(box_num,2) - pos_velo(i,1);
                updated_x = boxes(box_num,2);
             end

             if(pos_velo(i,4) > 0)
                y = pos_velo(i,2) - boxes(box_num, 3);
                updated_y = boxes(box_num, 3);
             else
                y = boxes(box_num, 4) - pos_velo(i,2);
                updated_y = boxes(box_num, 4);
             end

             if(x < y)
                pos_velo(i,1) = updated_x;
                if(~boxes_specular(box_num))
                    %Diffusive Boundaries in the x direction
                    change = -sign(pos_velo(i,3));
                    v = sqrt(pos_velo(i,3).^2 + pos_velo(i,4).^2);
                    angle = rand()*2*pi;
                    pos_velo(i,3) = change.*abs(v.*cos(angle));
                    pos_velo(i,4) = v.*sin(angle);
                else
                    % Specular Boundaries in the x direction
                    pos_velo(i,3) = -pos_velo(i,3);
                end
             else
                pos_velo(i,2) = updated_y;
                if(~boxes_specular(box_num))
                    %Diffusive Boundaries in the y direction
                    change = -sign(pos_velo(i,4));
                    v = sqrt(pos_velo(i,3).^2 + pos_velo(i,4).^2);
                    angle = rand()*2*pi;
                    pos_velo(i,3) = v.*cos(angle);
```

```matlab
                pos_velo(i,4) = change.*abs(v.*sin(angle));
            else
                % Specular Boundaries in the y direction
                pos_velo(i,4) = -pos_velo(i,4);
            end
        end
        %Getting the box that the particle is hitting
        box_num = in_box(pos_velo(i,1:2), boxes);
    end
end
%Storing the Temperature value of the semiconductor at each
iteration
temp(j) = (sum(pos_velo(:,3).^2) + sum(pos_velo(:,4).^2))*m/k/2/
par;
%Storing the Trajectory value of the semiconductor at each
iteration
traj(1:iter,2*j:2*j+1) = [pos_velo(1:iter,1) pos_velo(1:iter,2)];

figure(5);
subplot(2,1,1)
%Creating the box in the plot
x_plot1 = [x1_box1, x2_box1, x2_box1, x1_box1, x1_box1];
y_plot1 = [y1_box1, y1_box1, y2_box1, y2_box1, y1_box1];
x_plot2 = [x1_box2, x2_box2, x2_box2, x1_box2, x1_box2];
y_plot2 = [y1_box2, y1_box2, y2_box2, y2_box2, y1_box2];
%Plotting the Boxes
plot(x_plot1,y_plot1,'-','Color',[0 0 0])
hold on
plot(x_plot2,y_plot2,'-','Color',[0 0 0])
%Plotting each particle in the population at each time step
plot(pos_velo(1:pop,1)./10^-9, pos_velo(1:pop,2)./10^-9, 'o')
hold off
axis([0 length/1e-9 0 width/1e-9])
title('Simulation of Electrons in Silicon Crystal')
xlabel('(nm)')
ylabel('(nm)')
subplot(2,1,2)
%Plotting the temperature of the Silicon crystal over time
plot(step*(0:j-1), temp(1:j),'Color',[0.1 0.1 0.1])
axis([0 step*iter 200 400]);
title('Temperature of Electrons in Silicon Crystal')
xlabel('Time(s)')
ylabel('Temperature(K)')
hold on
end
%Creating the box in the plot
x_plot1 = [x1_box1, x2_box1, x2_box1, x1_box1, x1_box1];
y_plot1 = [y1_box1, y1_box1, y2_box1, y2_box1, y1_box1];
x_plot2 = [x1_box2, x2_box2, x2_box2, x1_box2, x1_box2];
y_plot2 = [y1_box2, y1_box2, y2_box2, y2_box2, y1_box2];
figure(6)
%Plotting Box
plot(x_plot1,y_plot1,'-','Color',[0 0 0])
hold on
```

```matlab
plot(x_plot2,y_plot2,'-','Color',[0 0 0])
for i = 1:pop
    %Plotting the trajectory vector of each particle in the shown
    %population
    %Setting a Random colour each particle
    color = [rand rand rand];
    figure(6)
    plot(traj(i,2:2:end)./1e-9,
 traj(i,1:2:end-1)./1e-9, '.', 'Color',color);
    axis([0 length/1e-9 0 width/1e-9]);
    title('Trajectories of Electrons in Silicon Crystal')
    xlabel('(nm)')
    ylabel('(nm)')
    hold on;
    pause(0.5);

end
%Calculate the average time
t_mn = ((step*iter*pop)/total_scatters);
%Calculate the average velocity
velocity_average= sqrt(sum((pos_velo(:,3).^2))/par +
 sum(pos_velo(:,4).^2)/par);
%Calculate the mean free path
mfp = (t_mn * velocity_average)*10^9;
%Calculate the average temperature
av_temp = sum(temp)/iter;
%Output the average temperature
fprintf('Average Temperature in the Crystal for part 3 is %f K
 \n',av_temp)
%Output the average velocity
fprintf('Average Electron Velocities in Silicon Crystal for part 3 are
 %f km/s \n',velocity_average/10^3)
%Output the average time between colisions
fprintf('Average time for part 3 is %f ps\n',t_mn/10^-12);
%Output the mean free path
fprintf('Mean free Path for part 3 is %f nm\n',mfp);
figure(7)
%Creating a 3D Histogram with x and y positions
electron_density = hist3([pos_velo(:,1),pos_velo(:,2)],[200 100]);
n = 20;
sigma = 4;
%Performing smoothing on histogram through convolution with meshgrid
[u, v] = meshgrid(round(-n./2):round(n./2),round(-n./2):round(n./2));
smooth_func = exp(-u.^2/(2*sigma^2)-v.^2/(2*sigma^2));
smooth_func = smooth_func./sum(smooth_func(:));
%Performing convolution of the two matrices and plotting the
 convolution
imagesc(conv2(electron_density,smooth_func));
title('Electron Density Map in Silicon Crystal');
xlabel('x (nm)');
ylabel('y (nm)');


%
```

```matlab
% Temperature map calculations
%
%Creating a temperature sum variable for the length and width of the
%silicon crystal
temp_sum = zeros(ceil((length)/10/10^-9), ceil((width)/10/10^-9));
%Counting the temperature variables for the length and width of the
%silicon crystal
temp_num = zeros(ceil((length)/10/10^-9), ceil((width)/10/10^-9));

for i = 1:par
    %
    x = floor(pos_velo(i, 1)/10/10^-9);
    y = floor(pos_velo(i, 2)/10/10^-9);
    %
    if (x==0)
        x = 1;
    end
    if (y==0)
        y = 1;
    end
    %Calculating the temperature sum for e
    temp_sum(x, y) = temp_sum(x, y) + ( pos_velo(i,3)^2 + pos_velo(i,
 4)^2 );
    temp_num(x, y) = temp_num(x, y) + 1;
end

%Calculating temperature matrix
temperature_matrix = temp_sum.*(m./k./2./temp_num);
%Performing convolution of the two matrices and plotting the
 convolution
imagesc(conv2(temperature_matrix,smooth_func));
title('Temperature Map of Silicon Crystal');
xlabel('x (nm)');
ylabel('y (nm)');


function count=count_scatters(matrix)
%This function creates a counter which counts the number of colisions
 of
%particles.
count = 0;
for i=1:size(matrix)
    if matrix(i) == 1
        count=count+1;
    end
end
end
function box_num = in_box(pos, boxes)
%Checks for Particles inside the box, when they are in the box it
 returns
%true so that the particles trajectory can be recalculated.
box_num = 0;
for i=1:size(boxes,1)
```

```matlab
    if(pos(1) > boxes(i,1) && pos(1) < boxes(i,2) && pos(2) >
 boxes(i,3) && pos(2) < boxes(i,4))
        box_num = i;
        return;
    end
end
end
```

Thermal Velocity for part 1 is 187.018585 km/s
Mean free path for part 1 is 37.403717 nm
Average Temperature in the Crystal for part 2 is 298.205368 K
Average Electron Velocities in Silicon Crystal for part 2 are
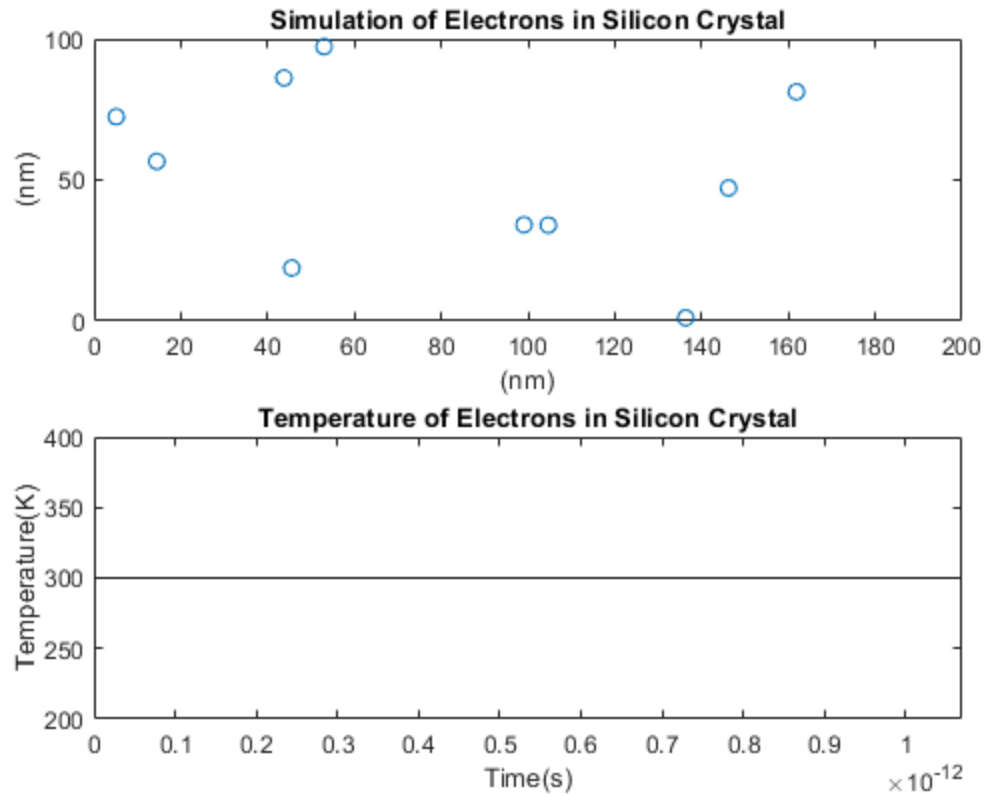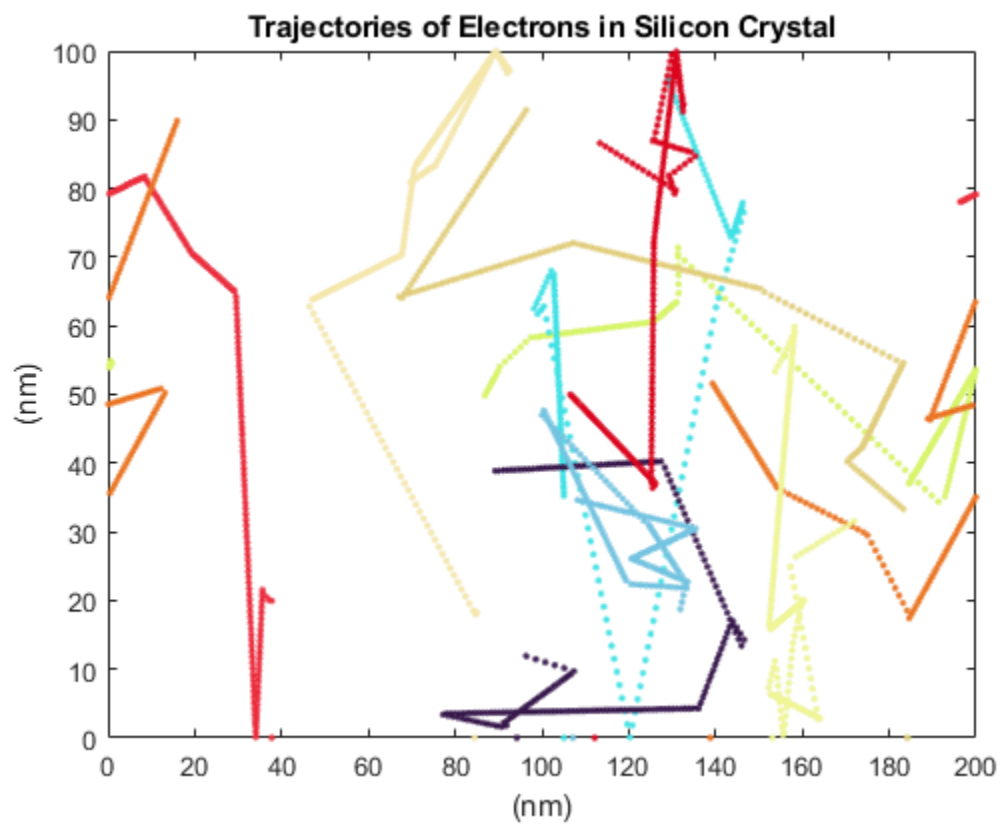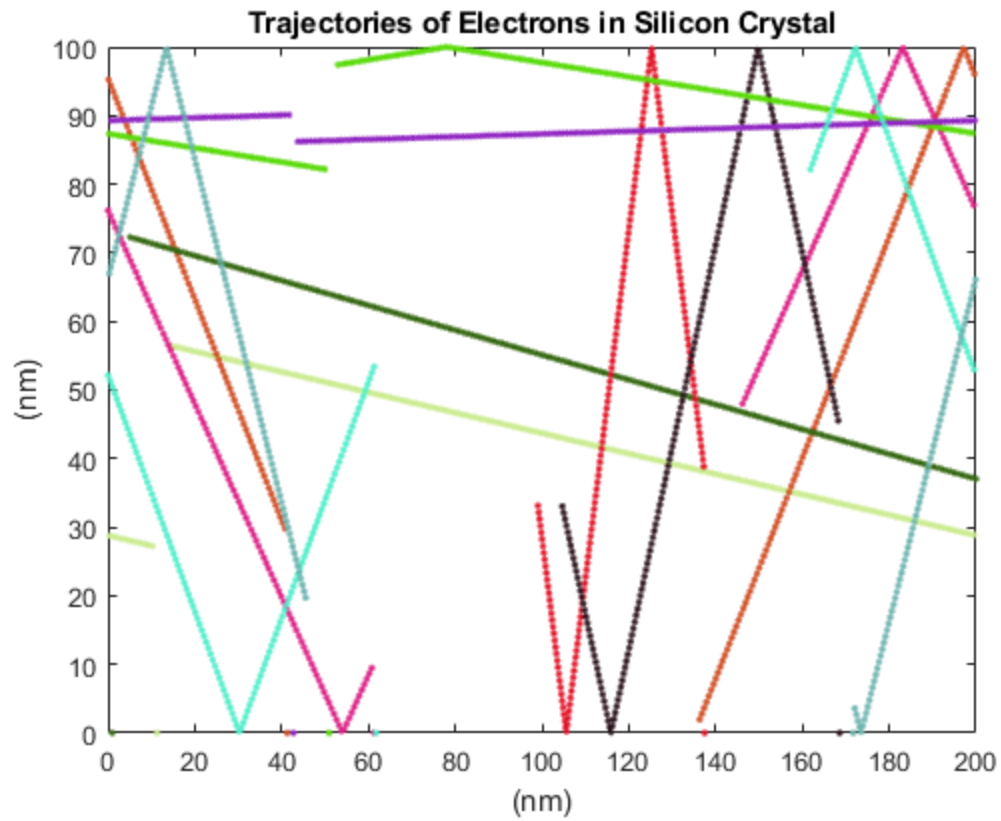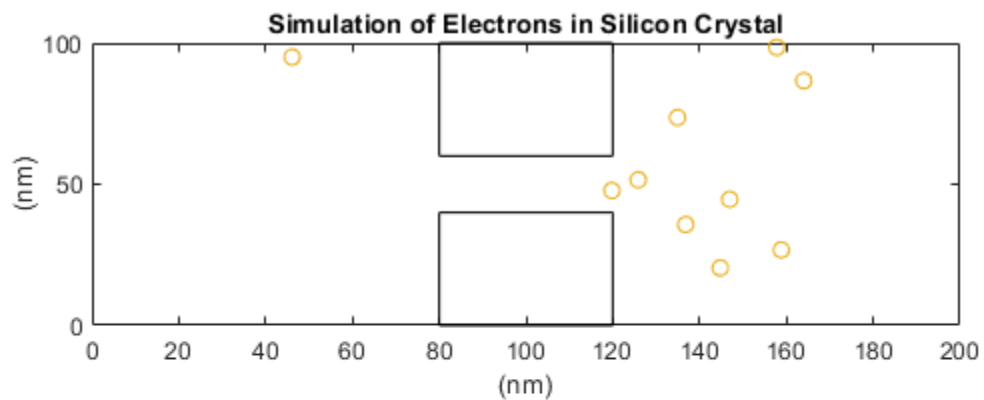 *186.442565 km/s*
Average time for part 2 is 0.157267 ps
Mean free Path for part 2 is 29.321176 nm
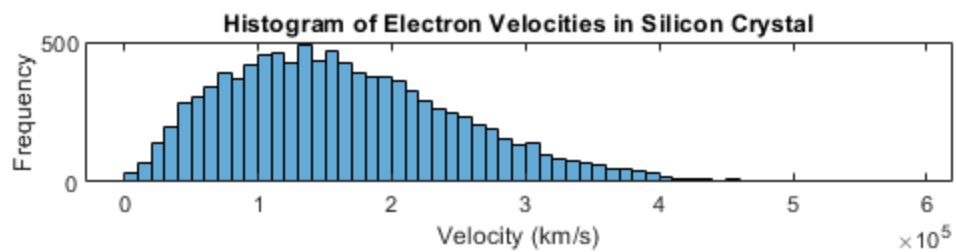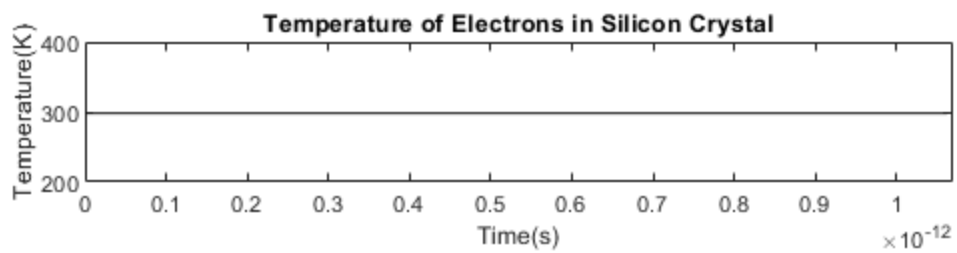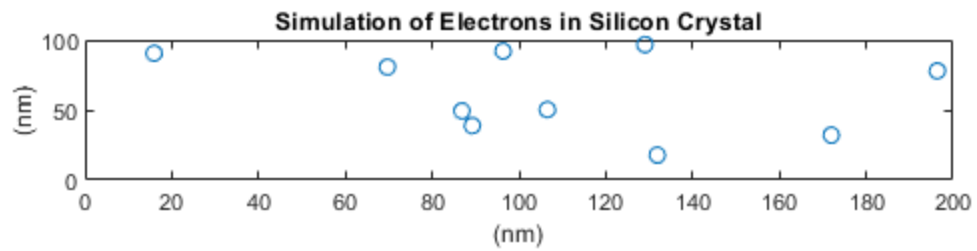*Average Temperature in the Crystal for part 3 is 305.598463 K*
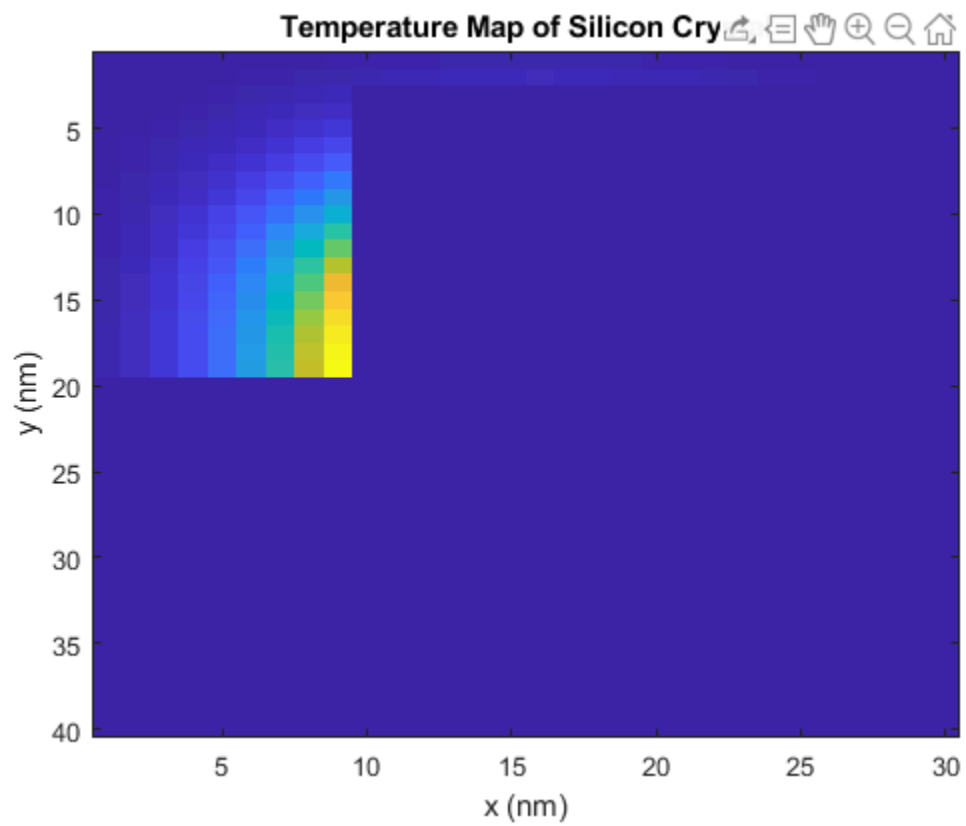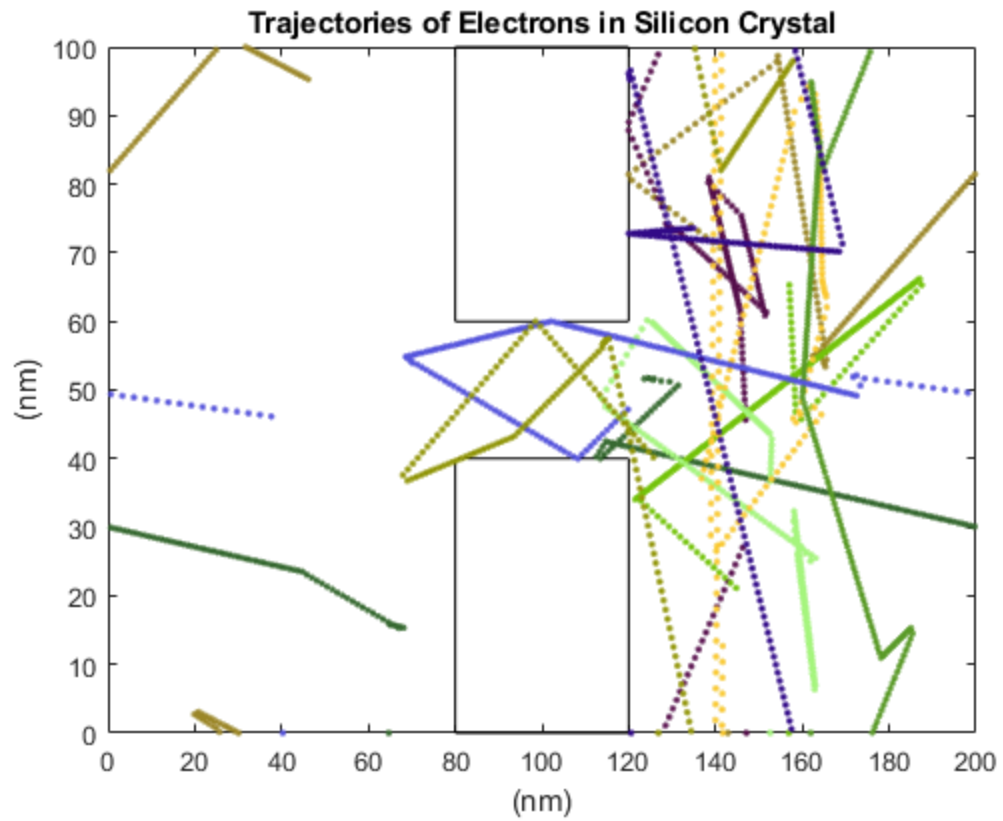*Average Electron Velocities in Silicon Crystal for part 3 are*
 188.725785 km/s
Average time for part 3 is 0.090628 ps
Mean free Path for part 3 is 17.103873 nm

Trajectories of Electrons in Silicon Crystal

Trajectories of Electrons in Silicon Crystal

Simulation of Electrons in Silicon Crystal

Temperature of Electrons in Silicon Crystal

Histogram of Electron Velocities in Silicon Crystal

Simulation of Electrons in Silicon Crystal

Temperature of Electrons in Silicon Crystal

Trajectories of Electrons in Silicon Crystal

Temperature Map of Silicon Crystal