Table of Contents

ELEC4700 Assignment 1

Reyad ElMahdy 101064879


```
clc
close all
clear
```

Part 1: Electron Modelling

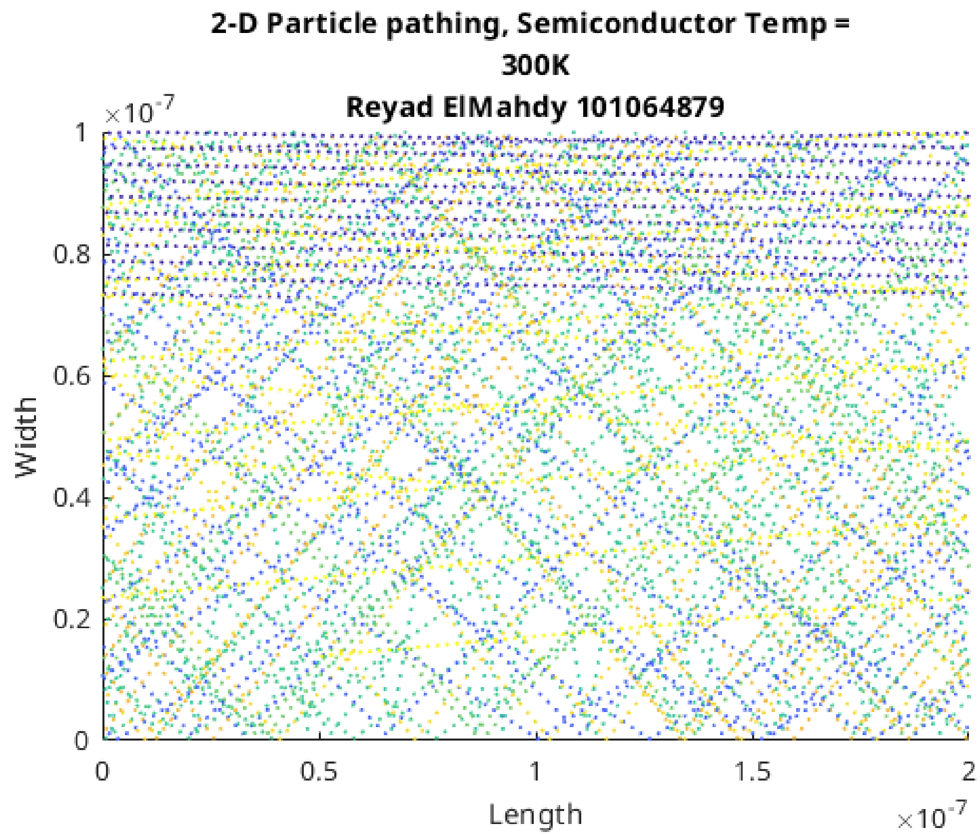
Setting the constants to be used in the calculations

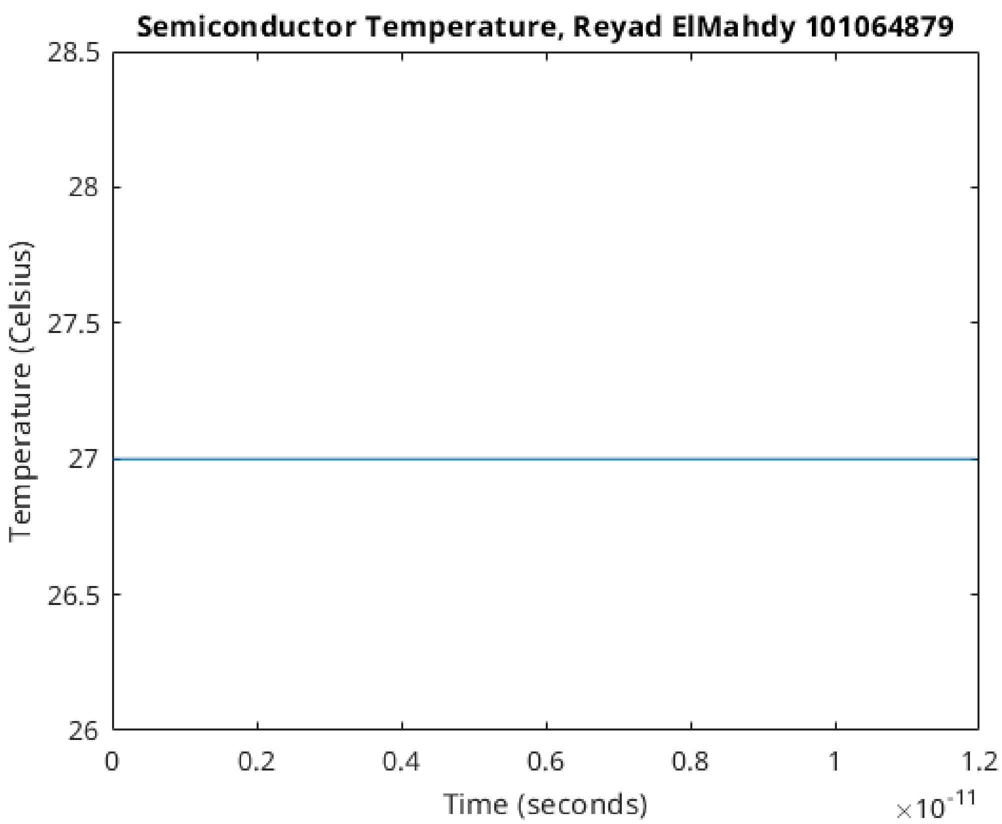
```
mElec = 9.11e-31; % Electron rest mass (kg)
mEff = 0.26*mElec; % Effective mass (kg)
kb = 1.381e-23; % Boltzmann's Constant (J/K)
T = 300; % Room Temperature (K)
L = 200e-9; % Length (m)
W = 100e-9; % Width (m)

% Finding the thermal velocity
Vth = sqrt((2*kb*T)/mEff); %Thermal Velocity in m/s
% Displaying the velocity in km/s on the command window
fprintf('Assuming the temperature T to be 300K the thermal velocity is
 %f km/s. \n', Vth/1000)

% a) The thermal velocity due to the particles' vibrations at room
% temperature was calculated with the equation Vth = sqrt((2*kb*T)/
mEff)
% where T is the absolute temperature, kb is boltzmann's constant, and
% meEff is the effective particle mass. This value is the same for all
% particles in the region assuming they have the same temperature.

% Finding the mean free path
mt = 0.2e-12; % mean time (s)
mfp = Vth*mt; % mean free path (m)
% Displaying the value in nm as the value is too small in meters
fprintf('Assuming the mean time to be 0.2 ps, the mean free path is %f
 nm.\n', mfp*1e9)

% b) Since the mean free path is just the average distance travelled
 due to the thermal velocity during
% the mean time. This is calculated by multiplying the thermal
 velocity by
```

```
% the mean time.

% Modeling the electron paths

numPar = 1000; % Number of particles
numPar2 = 10; % Subset for plots
% Assigning particle positions
posX = L.*rand(numPar,2);
posY = W.*rand(numPar,2);
posX(:,1) = posX(:,2);
posY(:,1) = posY(:,2);

% Assigning particle Directions (and velocity)
angle = (2*pi).*rand(numPar,2);
Vx = Vth*cos(angle);
Vy = Vth*sin(angle);
Vx(:,1) = Vx(:,2);
Vy(:,1) = Vy(:,2);

% Calculating step time
spacialStep = sqrt(L^2+W^2)/100;
stepTime = spacialStep/Vth;

% Calculating Displacement per step
dispX = stepTime*Vx(:,1);
dispY = stepTime*Vy(:,1);

colors = rand(numPar2,3);
timeVec = zeros(1,1000); sTemp = timeVec; avgKE = sTemp;
% Looping through and creating the simulation
for i = 1:1000
    for j = 1:numPar
        if (posX(j,1)+dispX(j) > L)
            posX(j,2) = posX(j,1)+dispX(j)-L;
        elseif (posX(j,1)+dispX(j) < 0)
            posX(j,2) = posX(j,1)+dispX(j)+L;
        else
            posX(j,2) = posX(j,1)+dispX(j);
        end

        if ((posY(j,1)+dispY(j) > W) || (posY(j,1)+dispY(j) < 0))
            dispY(j) = -dispY(j);
            posY(j,2) = posY(j,1)+dispY(j);
        else
            posY(j,2) = posY(j,1)+dispY(j);
        end
    end

    Vmag = sqrt(Vx(:,1).^2 + Vy(:,1).^2); % Magnitude of particle
 velocities
    KEmat = 0.5*(mEff*Vmag.^2);
    avgKE(i) = sum(KEmat(:,1))/numPar;
    sTemp(i) = avgKE(i)/kb;
    timeVec(i) = stepTime*i;
```

```
    if (i-1 == 0)
       figure(1);
       scatter(posX(1:numPar2, 2),posY(1:numPar2,2),1,colors(:,1));
       hold on;
       title([{"2-D Particle pathing, Semiconductor Temp = "
 +sTemp(i)+ "K"},{'Reyad ElMahdy 101064879'}]);
       xlabel('Length');
       ylabel('Width');
       xlim([0 200e-9])
       ylim([0 100e-9])
    elseif (i-1 < 1000)
       title([{"2-D Particle pathing, Semiconductor Temp = "
 +sTemp(i)+ "K"},{'Reyad ElMahdy 101064879'}]);
       scatter(posX(1:numPar2, 2),posY(1:numPar2,2),1,colors(:,1));
    else
       title([{"2-D Particle pathing, Semiconductor Temp = "
 +sTemp(i)+ "K"},{'Reyad ElMahdy 101064879'}]);
       scatter(posX(1:numPar2, 2),posY(1:numPar2,2),1,colors(:,1));
       hold off;
    end
    pause(0.001);
    posX(:,1) = posX(:,2);
    posY(:,1) = posY(:,2);
end

figure(5)
plot(timeVec,sTemp-273)
title('Semiconductor Temperature, Reyad ElMahdy 101064879')
xlabel('Time (seconds)')
ylabel('Temperature (Celsius)')

% As you can see from the plots, the semiconductor temperature is
% constant, which is expected since the magnitude of each particle's
% velocity is constant, only the direction changes. Since we're
 assuming
% the only movement is caused by the heat/vibrations due to the
 resting
% temperature of the semiconductor, the temperature will stay constant
% throughout the simulation unless something causes the particle
 velocities
% to change.
```

Assuming the temperature T to be 300K the thermal velocity is 187.036601 km/s.
Assuming the mean time to be 0.2 ps, the mean free path is 37.407320 nm.


2-D Particle pathing, Semiconductor Temp = 300K
Reyad ElMahdy 101064879
Width
Length
0
0.2
0.4
0.6
0.8
1
×10^-7
0
0.5
1
1.5
2
×10^-7


Semiconductor Temperature, Reyad ElMahdy 101064879
Temperature (Celsius)
Time (seconds)
26
26.5
27
27.5
28
28.5
0
0.2
0.4
0.6
0.8
1
1.2
×10^-11

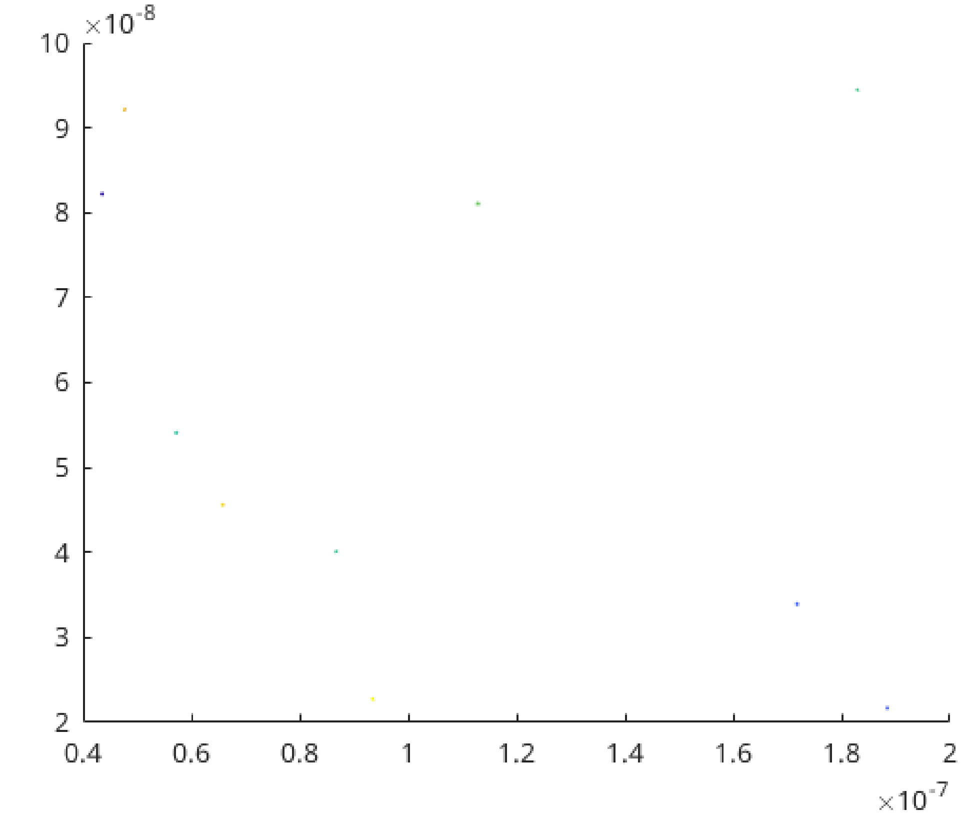
Part 2: Collisions with Mean Free Path

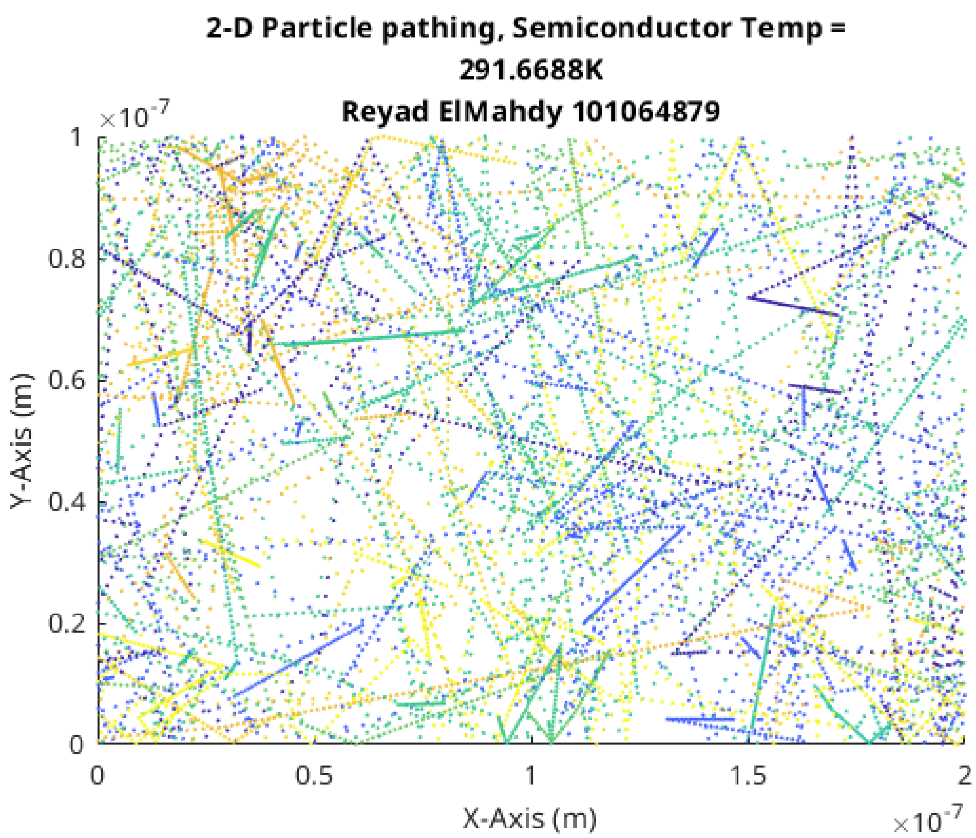
```
% In this section, the particle collisions are simulated, causing
% fluctuations in particle velocities as well as direction. As you'll
 see
% in this section of the report, this causes the temperature and the
% trajectory of every electron to change at random intervals, and the
 mean
% free path will depend on the total number of particle collisions

posX = L.*rand(numPar,2);
posY = W.*rand(numPar,2);
posX(:,1) = posX(:,2);
posY(:,1) = posY(:,2);
% Assigning random velocities to each particle
Vx = randn(numPar,2)*sqrt((kb*T)/mEff);
Vy = randn(numPar,2)*sqrt((kb*T)/mEff);

% Generating the histogram
V = (Vx(:,1).^2 + Vy(:,1).^2).^0.5;
figure(3)
histogram(V,numPar)
title('Histogram of Particle Speeds, Reyad ElMahdy 101064879')
xlabel('Velocity (m/s)')
ylabel('Number of Particles')

dispX = stepTime*Vx(:,1);
dispY = stepTime*Vy(:,1);

Pscatter = 1 - exp(-stepTime/mt); % Probability of an electron
 scattering

% Positions before collision
collX = posX(:,1);
collY = posY(:,1);

collTime = 0;
numColl = 0;

sTemp = zeros(1,1000); timeVec = sTemp; sTempAvg = sTemp;
for i = 1:1000
    for j = 1:numPar
        % Checking if the electron scatters
        if(rand(1) < Pscatter)
            % Incrementing the collision counter and generating new
            % velocity/displacment values
            numColl = numColl+1;
            Vx(j,:) = randn(1)*sqrt((kb*T)/mEff);
            Vy(j,:) = randn(1)*sqrt((kb*T)/mEff);
            dispX(j,:) = stepTime*Vx(j,1);
            dispY(j,:) = stepTime*Vy(j,1);
```

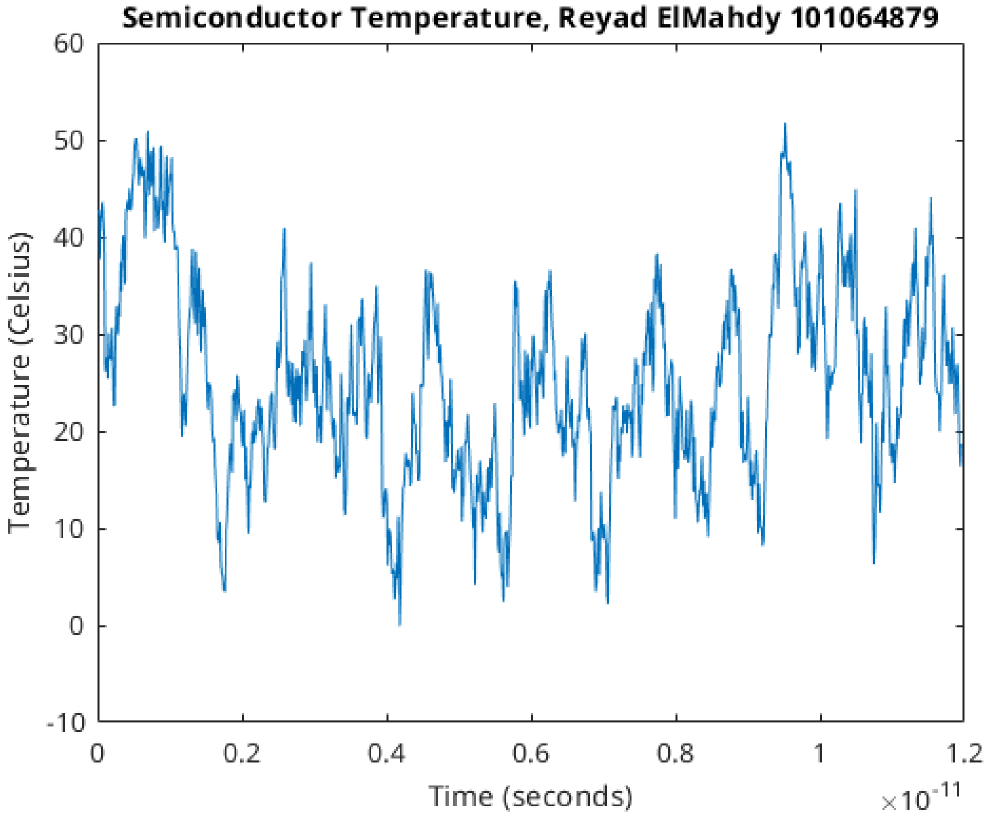
```
            % Finding the distance travelled and the time before a
collision
            dCollX = posX(j,2) - collX(j);
            dCollY = posY(j,2) - collY(j);
            dColl(numColl) = sqrt(dCollX^2+dCollY^2);
            tCollVec(numColl) = abs(stepTime*j - collTime);
            collTime = stepTime*j;

            collX = posX(:,1);
            collY = posY(:,1);
        end

        % Same as Q1
        if (posX(j,1)+dispX(j) > 2e-7)
            posX(j,2) = posX(j,1)+dispX(j)-2e-7;
        elseif (posX(j,1)+dispX(j)<0)
            posX(j,2) = posX(j,1)+dispX(j)+2e-7;
        else
            posX(j,2) = posX(j,1)+dispX(j);
        end
        if (posY(j,1)+dispY(j) > 1e-7)|| (posY(j,1)+dispY(j) < 0)
            dispY(j) = -dispY(j);
            posY(j,2) = posY(j,1)+dispY(j);
        else
            posY(j,2) = posY(j,1)+dispY(j);
        end
    end
    % Calculating kinetic energy
    Vsq = (Vx(:,1).^2 + Vy(:,1).^2);
    KEmat = 0.5*(mEff*Vsq);
    avgKE(i) = sum(KEmat(:,1))/numPar;
    sTemp(i) = avgKE(i)/kb;
    timeVec(i) = stepTime*i;

    if (i-1 == 0)
        figure(4)
        scatter(posX(1:numPar2,2),posY(1:numPar2,2),1,colors(:,1))
        hold on
       title([{"2-D Particle pathing, Semiconductor Temp = "
+sTemp(i)+ "K"},{'Reyad ElMahdy 101064879'}]);
        xlabel('X-Axis (m)')
        ylabel('Y-Axis (m)')
        xlim([0 200e-9])
        ylim([0 100e-9])
    elseif (i-1 < 1000)
        title([{"2-D Particle pathing, Semiconductor Temp = "
+sTemp(i)+ "K"},{'Reyad ElMahdy 101064879'}]);
        scatter(posX(1:numPar2,2),posY(1:numPar2,2),1,colors(:,1))
    else
        title([{"2-D Particle pathing, Semiconductor Temp = "
+sTemp(i)+ "K"},{'Reyad ElMahdy 101064879'}]);
        scatter(posX(1:numPar2,2),posY(1:numPar2,2),1,colors(:,1))
        hold off
    end
```

```
    pause(0.001)

    % Updating position vectors between iterations
    posX(:,1) = posX(:,2);
    posY(:,1) = posY(:,2);

    % Storing step times in a time vector
    timeVec(i) = stepTime*i;
end
% Plotting the temperature
figure(5)
plot(timeVec,sTemp-273)
title('Semiconductor Temperature, Reyad ElMahdy 101064879')
xlabel('Time (seconds)')
ylabel('Temperature (Celsius)')

% Calclating the MFP
dTotal = sum(dColl);
MFP = dTotal/numColl;
fprintf('The MFP of the simulation is %f nm \n', MFP*1e9)
% Calculating the mean time between collisions
timeTotal = sum(timeVec);
meanTime = timeTotal/numColl;
fprintf('The mean time between collisions of the simulation is %f ps
 \n', meanTime*1e12)
```

The MFP of the simulation is 0.046945 nm
The mean time between collisions of the simulation is 0.102935 ps




2-D Particle pathing, Semiconductor Temp =
291.6688K
Reyad ElMahdy 101064879
Y-Axis (m)
X-Axis (m)



Part 3: Enhancements

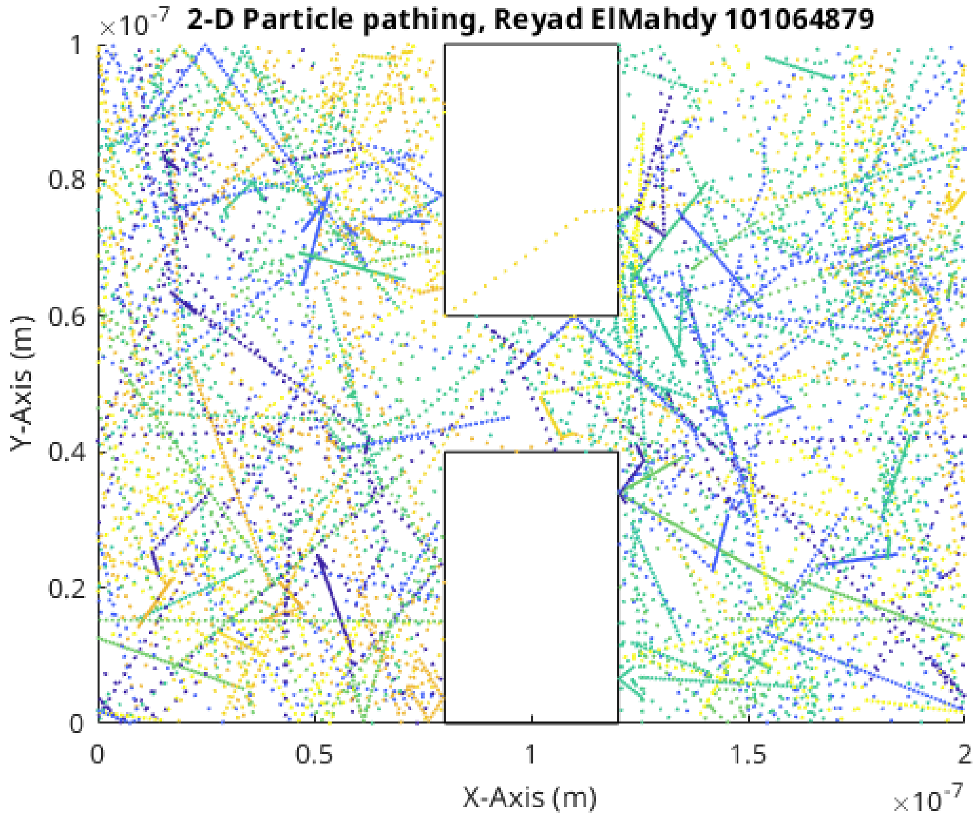
```
posX = L.*rand(numPar,2);
posY = W.*rand(numPar,2);

% Ensure no particles spawn inside the boxes
for i = 1:numPar
    while ((posX(i,2) > 0.8e-7 && posX(i,2) < 1.2e-7) && (posY(i,2) >
 0.6e-7 ||  posY(i,2) < 0.4e-7))
        posX(i,2) = L.*rand(1,1);
        posY(i,2) = W.*rand(1,1);
    end
end

posX(:,1) = posX(:,2);
posY(:,1) = posY(:,2);

Vx = randn(numPar,2)*sqrt((kb*T)/mEff);
Vy = randn(numPar,2)*sqrt((kb*T)/mEff);
V = (Vx(:,1).^2 + Vy(:,1).^2).^0.5;

dispX = stepTime*Vx(:,1);
dispY = stepTime*Vy(:,1);

for i = 1:1000
```

```
    for j = 1:numPar
        % Checking if the electron scatters
        if(rand(1) < Pscatter)
            % Incrementing the collision counter and generating new
            % velocity/displacment values
            numColl=numColl+1;
            angle = 2*pi.*rand(1);
            Vx(j,:) = randn(1)*sqrt((kb*T)/mEff);
            Vy(j,:) = randn(1)*sqrt((kb*T)/mEff);
            dispX(j,:) = stepTime*Vx(j,1);
            dispY(j,:) = stepTime*Vy(j,1);

            % Finding the distance travelled and the time before a
collision
            dCollX = posX(j,2) - collX(j);
            dCollY = posY(j,2) - collY(j);
            dColl(numColl) = sqrt(dCollX^2+dCollY^2);
            tCollVec(numColl) = abs(stepTime*j - collTime);
            collTime = stepTime*j;

            collX = posX(:,1);
            collY = posY(:,1);
        end

        % More boundary conditions need to be added to simulate the
boxes
        % in part 3
        if (posX(j,1)+dispX(j) > 2e-7)
            posX(j,2) = posX(j,1)+dispX(j)-2e-7;
        elseif (posX(j,1)+dispX(j)<0)
            posX(j,2) = posX(j,1)+dispX(j)+2e-7;
        elseif (posX(j,1) >= 0 && posX(j,1) <= 0.8e-7) && (posY(j,1)
>= 0 && posY(j,1) <= 0.4e-7) && (posX(j,1)+dispX(j,1) >= 0.8e-7)
            dispX(j) = -dispX(j);
            posX(j,2) = posX(j,1) + dispX(j);
        elseif (posX(j,1) >= 0 && posX(j,1) <= 0.8e-7) && (posY(j,1)
>= 0.6e-7 && posY(j,1) <= 1e-7) && (posX(j,1)+dispX(j) >= 0.8e-7)
            dispX(j) = -dispX(j);
            posX(j,2) = posX(j,1) + dispX(j);
        elseif (posX(j,1) >= 1.2e-7 && posX(j,1) <= 2e-7) &&
(posY(j,1) >= 0 && posY(j,1) <= 0.4e-7) && (posX(j,1)+dispX(j) <=
1.2e-7)
            dispX(j) = -dispX(j);
            posX(j,2) = posX(j,1) + dispX(j);
        elseif (posX(j,1) >= 1.2e-7 && posX(j,1) <= 2e-7) &&
(posY(j,1) >= 0.6e-7 && posY(j,1) <= 1e-7) && (posX(j,1)+dispX(j) <=
1.2e-7)
            dispX(j) = -dispX(j);
            posX(j,2) = posX(j,1) + dispX(j);
        else
            posX(j,2) = posX(j,1)+dispX(j);
        end
        if (posY(j,1)+dispY(j) > 1e-7) || (posY(j,1)+dispY(j) < 0)
            dispY(j) = -dispY(j);
```

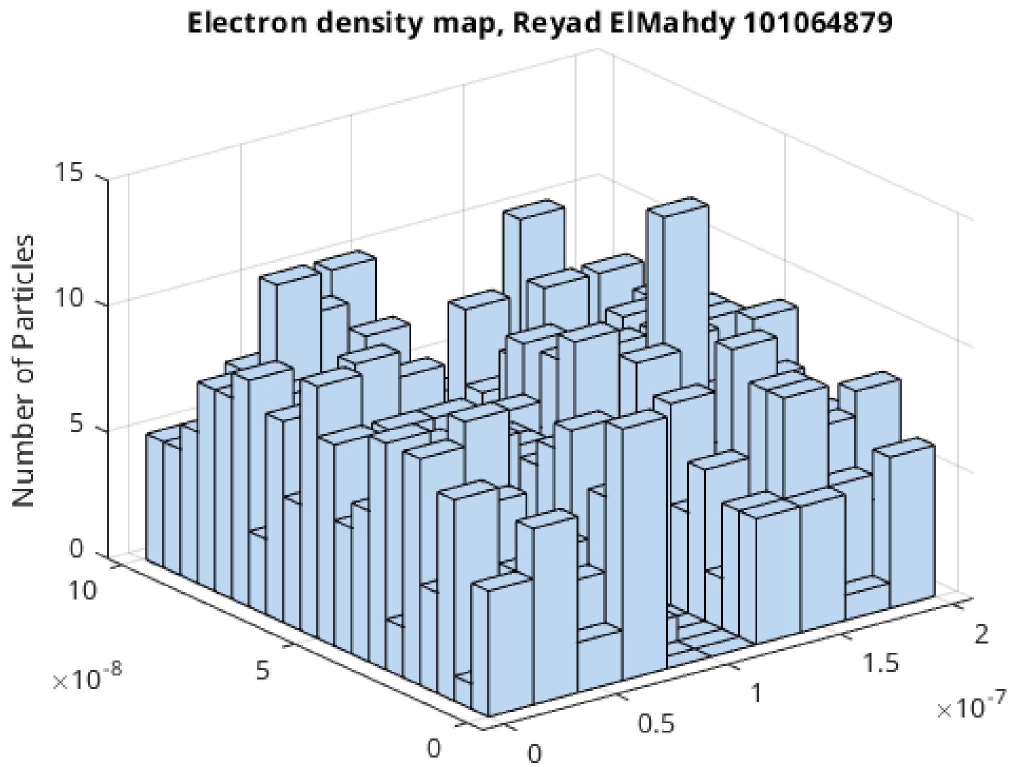
```
            posY(j,2) = posY(j,1)+dispY(j);
        elseif(posX(j,1)>=0.8e-7 && posX(j,1) <= 1.2e-7) && (posY(j,1)
 >= 0.4e-7 && posY(j,1) <= 0.6e-7) && (posY(j,1)+dispY(j) <= 0.4e-7)
            dispY(j) = -dispY(j);
            posY(j,2) = posY(j,1)+dispY(j);
        elseif(posX(j,1)>=0.8e-7 && posX(j,1) <= 1.2e-7) && (posY(j,1)
 >= 0.4e-7 && posY(j,1) <= 0.6e-7) && (posY(j,1)+dispY(j) >= 0.6e-7)
            dispY(j) = -dispY(j);
            posY(j,2) = posY(j,1)+dispY(j);
        else
            posY(j,2) = posY(j,1)+dispY(j);
        end
    end
    % Calculating kinetic energy
    KEmat = 0.5*(mEff*(Vx.^2 + Vy.^2));
    avgKE(i) = sum(KEmat(:,1))/numPar;
    sTemp(i) = sTemp(i)+avgKE(i)/kb;
    sTempAvg(i) = sTemp(i)/i;
    timeVec(i) = stepTime*i;

    if (i-1 == 0)
        figure(6)
        scatter(posX(1:numPar2,2),posY(1:numPar2,2),1,colors(:,1))
        hold on

        % Plotting the boxes
        plot([0.8e-7,0.8e-7],[0,0.4e-7],'k',[0.8e-7,1.2e-7],
[0.4e-7,0.4e-7],'k',[1.2e-7,1.2e-7],[0.4e-7,0],'k',[0.8e-7,1.2e-7],
[0,0],'k')
        plot([0.8e-7,0.8e-7],[1e-7,0.6e-7],'k',[0.8e-7,1.2e-7],
[0.6e-7,0.6e-7],'k',[1.2e-7,1.2e-7],[0.6e-7,1e-7],'k',[0.8e-7,1.2e-7],
[1e-7,1e-7],'k')

        title('2-D Particle pathing, Reyad ElMahdy 101064879')
        xlabel('X-Axis (m)')
        ylabel('Y-Axis (m)')
        xlim([0 200e-9])
        ylim([0 100e-9])
    elseif (i-1 < 1000)
        scatter(posX(1:numPar2,2),posY(1:numPar2,2),1,colors(:,1))
    else
        scatter(posX(1:numPar2,2),posY(1:numPar2,2),1,colors(:,1))
        hold off
    end
    pause(0.001)

    % Updating position vectors between iterations
    posX(:,1) = posX(:,2);
    posY(:,1) = posY(:,2);

    % Storing step times in a time vector
    timeVec(i) = stepTime*i;
end
```

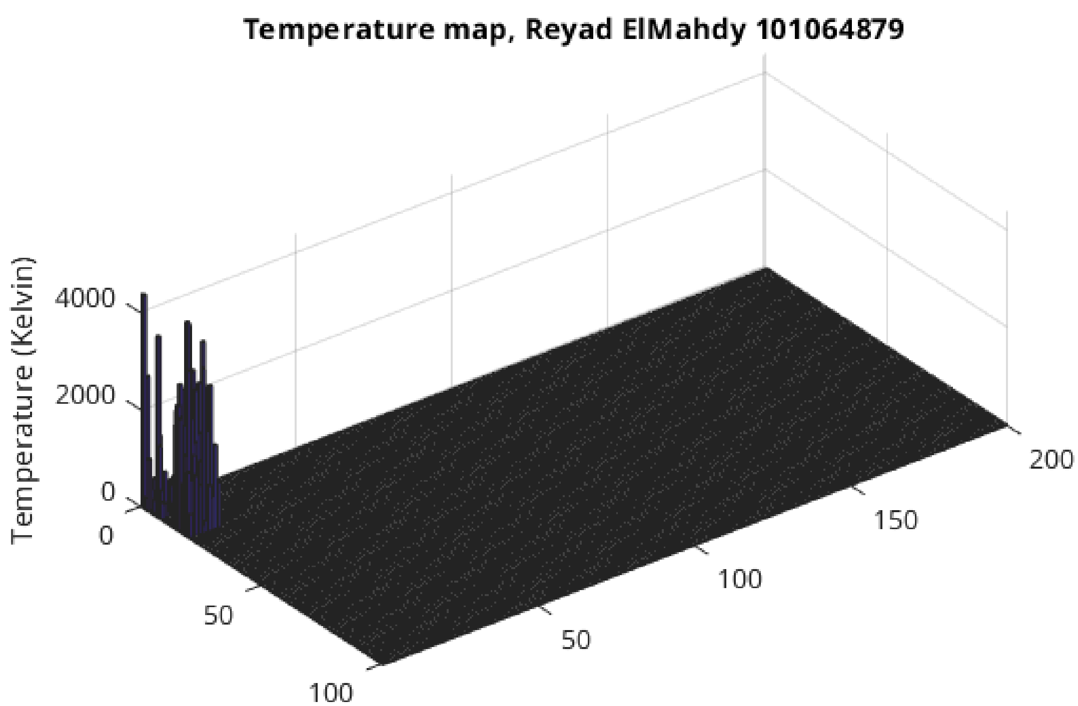
```
% Electron Density Map
figure(7)
hist3([posX(:,1),posY(:,1)],[10,20])
title('Electron density map, Reyad ElMahdy 101064879')
zlabel('Number of Particles')
% The electrons are pretty much evenl disributed outside the boxes
 which is
% expected.

% Temperature Map
kEtemp = (mEff.*V.^2)/2;
temp = kEtemp./kb;

% Scaling and rounding positions to the nearest nm
posXnm = round(posX(:,1).*1e9, -1)./10;
posYnm = round(posY(:,1).*1e9, -1)./10;

tempArr = zeros(round(W/1e-9),round(L/1e-9));

for i = 1:numPar
    posXnm = round(posX(i,1).*1e9,-1)/10;
    posYnm = round(posY(i,1).*1e9,-1)/10;

    if(posXnm <= 0 || posYnm <= 0)
        posXnm = 1;
        posYnm = 1;
    end
    tempArr(posXnm,posYnm) = tempArr(posXnm,posYnm) + temp(i);
end

matrix = hist3([posX(:,1),posY(:,1)],[10,20]);

for i = 1:round(W/1e-9)/10
    for j = 1:round(L/1e-9)/10
        if(matrix(i,j) == 0)
            tempArr(i,j) = 0;
        else
            tempArr(i,j) = tempArr(i,j)/matrix(i,j);
        end
    end
end

figure(8)
bar3(tempArr)
title('Temperature map, Reyad ElMahdy 101064879')
zlabel('Temperature (Kelvin)')
% I was unable to bin the temperatures properly to create a proper
% temperature map, but all other results for this section are pretty
 much
% exactly as expected. The only other bug I can think of is that when
 an
% electron hits a box's corner exactly at a 45 degree angle it might
 pass
% through.. but this is hightl unlikely as the angles are randomly
```

```
% generated.
```



2-D Particle pathing, Reyad ElMahdy 101064879
×10^-7
1
0.8
0.6
0.4
0.2
0
Y-Axis (m)
0
0.5
1
1.5
2
X-Axis (m)
×10^-7


Electron density map, Reyad ElMahdy 101064879
Number of Particles
15
10
5
0
10
5
0
$\times 10^{-8}$
0
0.5
1
1.5
2
$\times 10^{-7}$


Temperature map, Reyad ElMahdy 101064879
Temperature (Kelvin)
4000
2000
0
0
50
100
50
100
150
200

Published with MATLAB® R2020b