ELEC 4700 Assignment 1

Table of Contents

1 - Electron Modeling 1
1.1 - Thermal Velocity 1
1.2 - Mean Free Path 1
1.3 Model of Electron Random Motion 2
2 - Collisions with Mean Free Path' 5
2.1 - Maxwell-Boltzmann Distribution 5
2.2 - Electron Scattering 6
2.3 - Temperature over Time 8
2.4 - Comparison of Values 9
3 - Enhancements 10

1 - Electron Modeling

1.1 - Thermal Velocity

It is possible to calculate the thermal velocity of the electrons using the equation given below.

$$v_{th} = \sqrt{2kT/m}$$

By applying the formula to the known values of k, T, and m, we are able to obtain a value for thermal velocity as shown in the code below.

```
% Setup
clear
clc

% Constant values used for modeling
m0 = 9.11e-31; % Electron Rest Mass (kg)
mn = 0.26*m0;  % Effective Mass (kg)
T = 300;       % Temperature (K)
k = 1.381e-23; % Boltzmann Constant (J/K)

% Thermal Velocity (m/s)
thermalVelocity = sqrt(2*k*T/mn)
```

```
thermalVelocity =

   1.8704e+05
```

1.2 - Mean Free Path

Assuming that the electron velocity within the silicon is constant and equal to the calculated thermal velocity, one is able to calculate the mean free path of the electrons using the code shown below.

```
Tmn = 0.2e-12; % Mean time between collisions (s)

% Mean Free Path (m)
meanFreePath = Tmn*thermalVelocity


meanFreePath =

   3.7407e-08
```

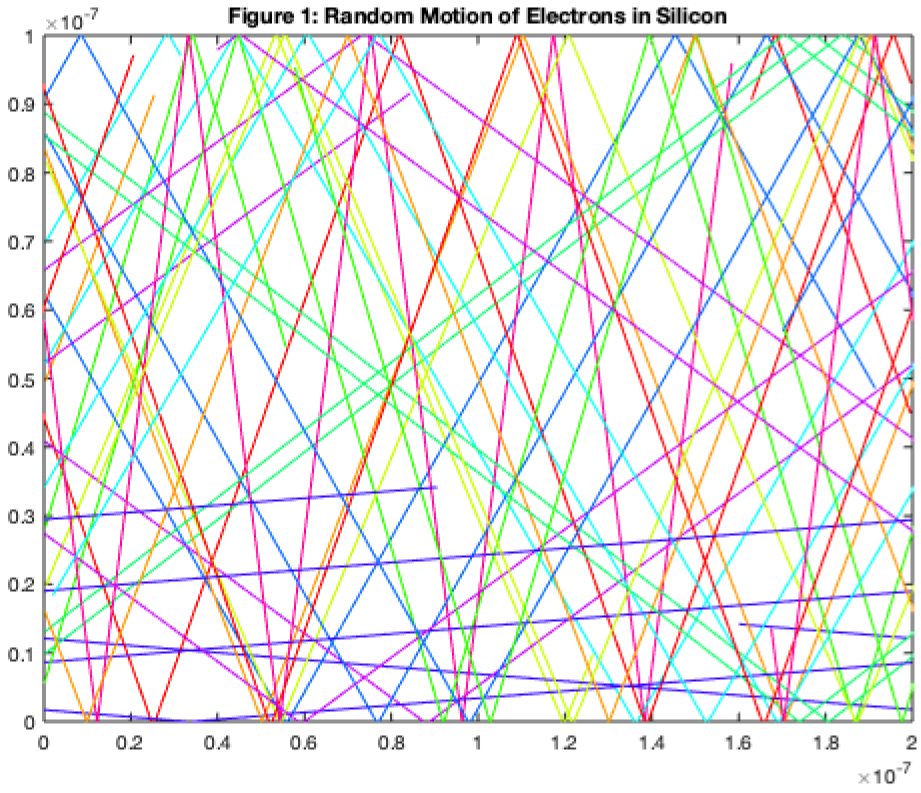
1.3 Model of Electron Random Motion

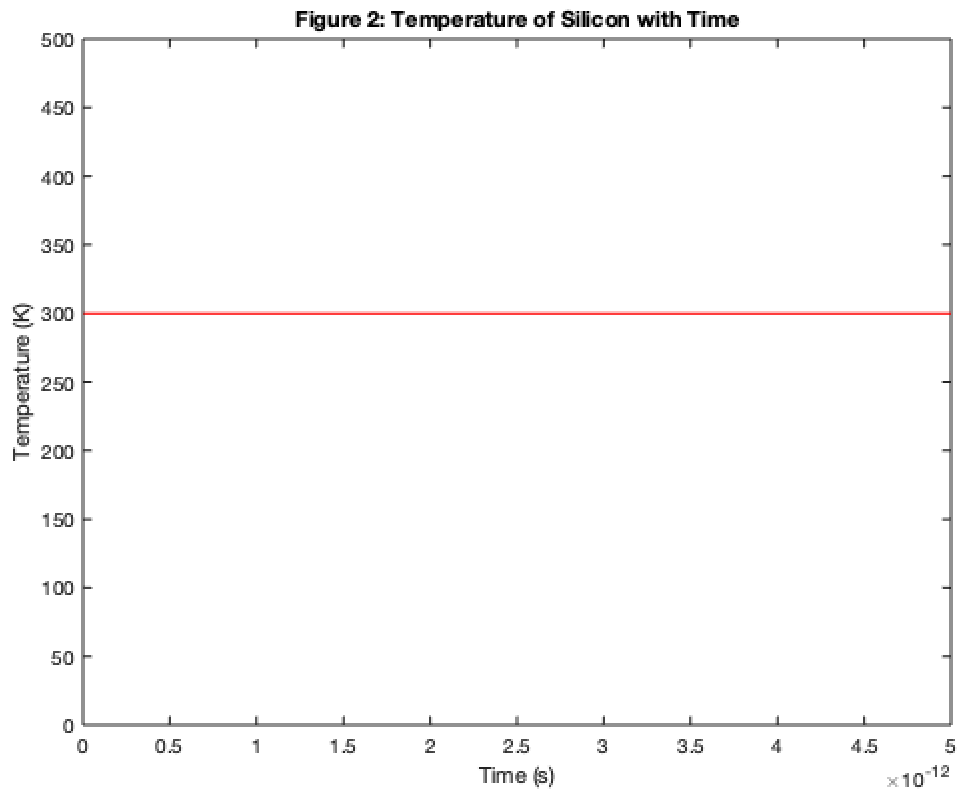
In order to model the random motion of the elctrons within silicon, we must first initialize a given number of particles with evenly distributed random positions within the desired 100nm x 200nm region. Each particle can then be assigned random X and Y directional data for their velocities, which must then be scaled to ensure all particles have a velocity equal to that of the thermal velocity. The system is modeled using 1000 electrons, and a plot showing the trajectories of a reduced number of these particles can be seen in Figure 1 below. A graph showing the temperature of the semiconductor with respect to time can be obtained by findng the average thermal velocity of all electrons during each time step and working backward to obtain temperature, this is shown in Figure 2 below.

```
% Model Parameters
numElec = 1000; % Number of Electrons
xDim = 200e-9;  % Dimention of Si region in X direction (m)
yDim = 100e-9;  % Dimention of Si region in Y direction (m)
deltaT = 5e-15  % Time step value (s)
lastPos = zeros(2,numElec); % Array to hold previous particle position
color = hsv(10); % Set particle colours for plot
avgVelocity = zeros(1,1000);


% Initiate particle positions as array of random numbers within
 boundaries.
position = [xDim;yDim].*rand(2,numElec);

%Initiate velocities as random numbers between 0 and 1.
velocity = rand(2,numElec);

% Shift X velocity to random value between -1 and 1, and set Y
 velocity
% accordingly.
for c = 1:numElec
    velocity(1,c) = (velocity(1,c) - 0.5)*2; % Ensures X velocity can
 also be negative.
    velocity(2,c) = (velocity(2,c) - 0.5)*2; % Ensures Y velocity can
 also be negative.
    velocity(2,c) = (velocity(2,c)/abs(velocity(2,c)))*sqrt(1-
(velocity(1,c))^2);
end

% Scale velocities to obtain desired values for this model.
velocity = thermalVelocity.*velocity;
```

```
% Initial variables
temp = 300; % Si starting temperature (K)
time = zeros(1,1000);   % Initial time (s)

for T = 1:1000
    lastPos = position;

    % Advance all particles in direction of their velocity
    position(1,:) = position(1,:) + (deltaT.*velocity(1,:));
    position(2,:) = position(2,:) + (deltaT.*velocity(2,:));

    for c = 1:numElec

        % Particle behaviour if boundary is reached
        if position(2,c) <= 0 || position(2,c) >= yDim
            velocity(2,c) = (-1)*velocity(2,c);
        end
        if position(1,c) <= 0
            position(1,c) = xDim;
            lastPos(1,c) = xDim;
        elseif position(1,c) >= xDim
            position(1,c) = 0;
            lastPos(1,c) = 0;
        end
    end

    sumVelocity = sum(sqrt(sum(velocity.^2)));

    % Plot new particle positions.
    figure(1)
    for c = 1:10
        plot([lastPos(1,c) position(1,c)],[lastPos(2,c)
 position(2,c)],'color', color(c,:))
        xlim([0 2e-7]);
        ylim([0 1e-7]);
        title('Figure 1: Random Motion of Electrons in Silicon')
        hold on
    end

    % Calculate average velocity (m/s)
    avgVelocity(T) = sumVelocity/numElec;

    % Update time
    time(T) = T*deltaT;

end

% Calculate and plot semiconductor temperature (K)
    temp = (avgVelocity.^2).*mn./(2*k);

    figure(2)
    plot(time, temp, 'r')
    xlim([0 5e-12]);
    ylim([0 500]);
    title('Figure 2: Temperature of Silicon with Time')
    xlabel('Time (s)');
    ylabel('Temperature (K)');
```

```
deltaT =

   5.0000e-15
```





2 - Collisions with Mean Free Path'

2.1 - Maxwell-Boltzmann Distribution

Particles must have their initial velocities set randomly using the Maxwell-Boltzmann distribution for velocity components. The equation to acheive this is given by

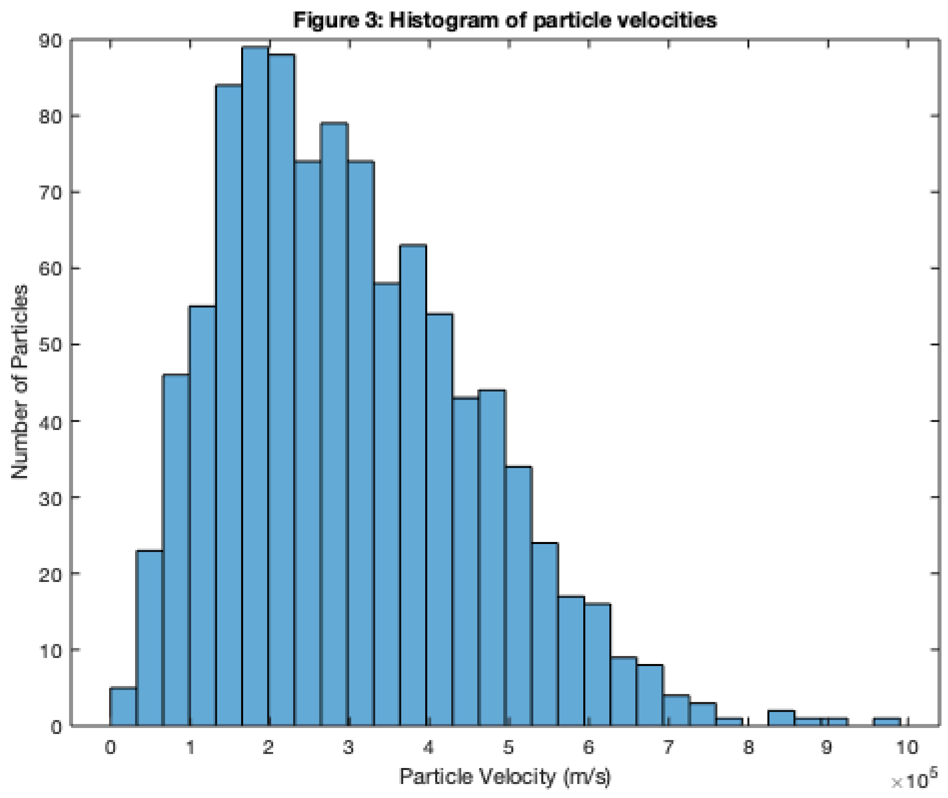
$$v_{i,\alpha} = \sqrt{kT/m_i} \times N(1,0)$$

where i is an integer reperesenting the specific particle, alpha is the direction of the velocity component (X or Y), and N(1,0) reperesents a normally distributed value with a mean of 0 and a variance of 1. The code below initializes an array of velocities, and Figure 3 shows a histogram of velocity distribution.

```
% Create array of X and Y velocity components (m/s)
maxBoltz = sqrt(k*T/mn).*randn(2,numElec);

% Determine magnitude of each velocity
velMag = zeros(1,numElec);
sumVelocity = 0;
for c = 1:numElec
    velMag(c) = sqrt((maxBoltz(1,c))^2 + (maxBoltz(2,c))^2);
    sumVelocity = sumVelocity + velMag(c);
end
```

```
% Plot histogram of velocity distribution
figure(3)
histogram(velMag,30)
title('Figure 3: Histogram of particle velocities')
xlabel('Particle Velocity (m/s)')
ylabel('Number of Particles')

% Validate average velocity is equal to the thermal velocity
averageVelocity = sumVelocity/numElec
thermalVelocity
```

```
averageVelocity =

   3.0063e+05


thermalVelocity =

   1.8704e+05
```



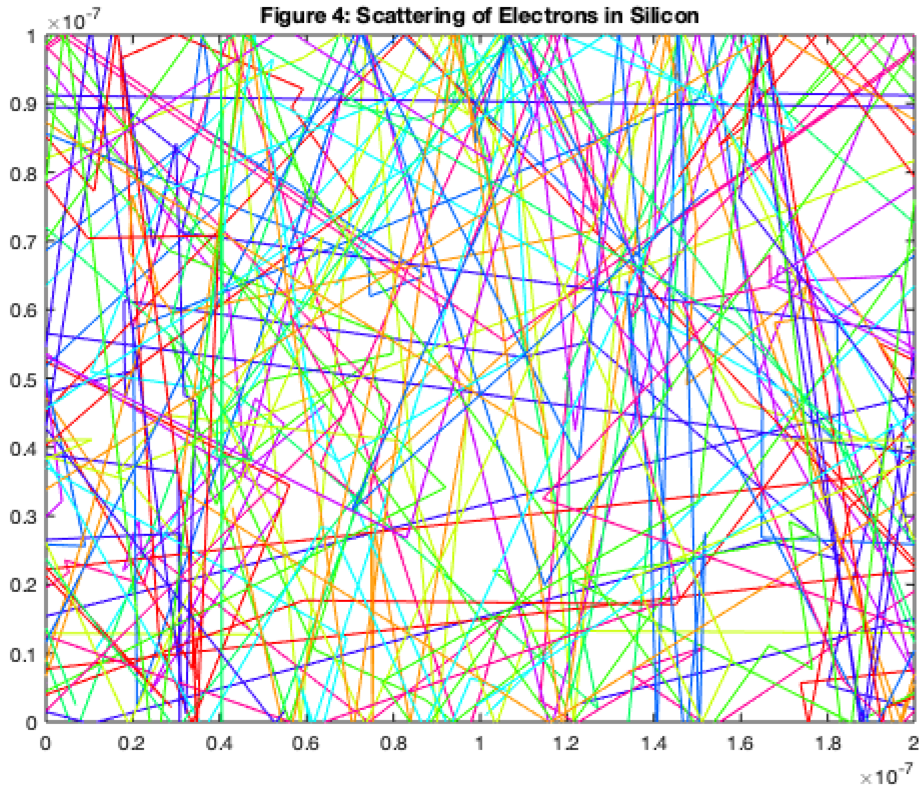
2.2 - Electron Scattering

The scattering of electrons within the silicon can be modeled by first determining the scattering probability using the equation shown below.

$$P_{scat} = 1 - e^{-dt/T_{mn}}$$

It is then possible to introduce a condition to the model whereby particles experience scattering with a probability equal to that of the scattering probability. Scattered particles are set to a new random velocity according to the maxwell-Boltzmann distribution. The code below shows this.

```
% Scattering probability
Pscat = 1 - exp(-(deltaT/Tmn));


% Initiate particle positions as array of random numbers within
 boundaries.
position = [xDim;yDim].*rand(2,numElec);

% Initial variables
temp = 300; % Si starting temperature (K)
time = zeros(1,1000);   % Initial time (s)
numCollisions = 0; % Initial number of collisions occured


for T = 1:1000
    lastPos = position;

    for c = 1:numElec

        % Scattering condition
         if Pscat > rand
             maxBoltz(1,c) = sqrt(k*T/mn)*randn;
             maxBoltz(2,c) = sqrt(k*T/mn)*randn;
             numCollisions = numCollisions + 1;
        end

        % Particle behaviour if boundary is reached
        if position(2,c) <= 0 || position(2,c) >= yDim
            maxBoltz(2,c) = (-1)*maxBoltz(2,c);
        end
        if position(1,c) <= 0
            position(1,c) = xDim;
            lastPos(1,c) = xDim;
        elseif position(1,c) >= xDim
            position(1,c) = 0;
            lastPos(1,c) = 0;
        end
    end

    sumVelocity = sum(sqrt(sum(maxBoltz.^2)));


    % Advance all particles in direction of their velocity
        position = position + (deltaT*maxBoltz);
        position = position + (deltaT*maxBoltz);

    % Update time
    time(T) = T*deltaT;
    % Plot new particle positions.
    figure(4)
    for c = 1:10
        plot([lastPos(1,c) position(1,c)],[lastPos(2,c)
 position(2,c)], 'color', color(c,:))
        xlim([0 2e-7]);
        ylim([0 1e-7]);
        title('Figure 4: Scattering of Electrons in Silicon')
        hold on
    end

    % Calculate average velocity (m/s)
    avgVelocity(T) = sumVelocity/numElec;

    % Update time
    time(T) = T*deltaT;

end
```

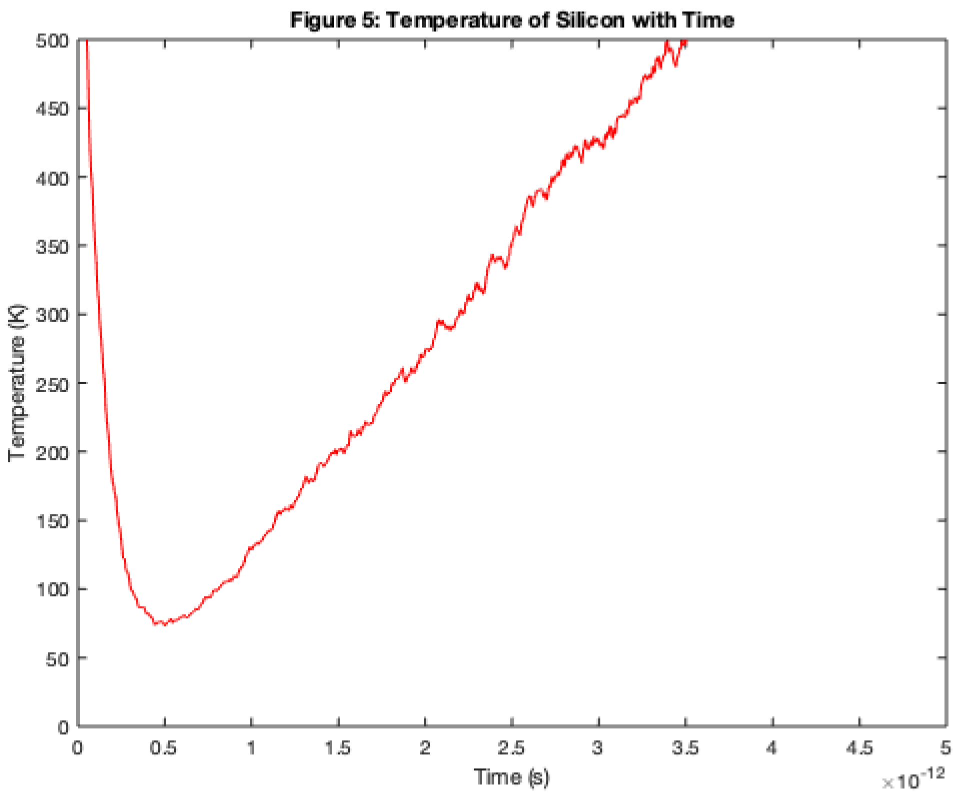


2.3 - Temperature over Time

Code showing how the temperature within the semiconductor changes with respect to time can be seen below, Figure 5 reperesents a plot of this data.

```
% Calculate and plot semiconductor temperature (K)
```

```
temp = (avgVelocity.^2).*mn./(2*k);

figure(5)
plot(time, temp, 'r')
xlim([0 5e-12]);
ylim([0 500]);
title('Figure 5: Temperature of Silicon with Time')
xlabel('Time (s)');
ylabel('Temperature (K)')
```



2.4 - Comparison of Values

To validate our modle we can compair the expected and actual values for the Mean Free Path and mean collision time parameters.

```
% Values given by model
meanCollisionTime = time(T)/(numCollisions/numElec)
meanFreePath = avgVelocity(T)*meanCollisionTime

% Expected values
meanCollisionTime_expected = Tmn
meanFreePath_expected = thermalVelocity*Tmn


meanCollisionTime =
   2.0371e-13


meanFreePath =

   6.1320e-08


meanCollisionTime_expected =

   2.0000e-13


meanFreePath_expected =

   3.7407e-08
```

3 - Enhancements

A bottle neck restriction can be added to the model given in section 2 above, inplementing a condition whereby particles are reflected from the surface of the blocks forming the bottle neck in the same way they reflect from the top and bottom surfaces of the previous model. The code below shows this, and Figure 6 depicts a trace of the particle movement for the new model.

A density map of the location of electrons after the model has run for 1000 time steps is given in Figure 7, and a temperature map of is given by Figure 8.

```
% Create array of X and Y velocity components (m/s)
maxBoltz = sqrt(k*T/mn).*randn(2,numElec);

% Initiate particle positions as array of random numbers within
 boundaries.
position = [xDim;yDim].*rand(2,numElec);

particleFound = true;

while particleFound

    count = 0;

    for i = 1:numElec
        if position(1,i)>=0.8e-7 && position(1,i)<=1.2e-7
            if position(2,i)<=0.4e-7 || position(2,i)>=0.6e-7
                position(1,i) = xDim*rand;
                position(2,i) = yDim*rand;
                count = count + 1;
            end
        end
    end

    if count == 0
        particleFound = false;
    end
```

```
end

for T = 0:1000

    lastPos = position;

    for c = 1:numElec

        % Scattering condition
        if Pscat > rand
            maxBoltz(1,c) = sqrt(k*T/mn)*randn;
            maxBoltz(2,c) = sqrt(k*T/mn)*randn;
            numCollisions = numCollisions + 1;
        end

        % Advance all particles in direction of their velocity
        position(1,c) = position(1,c) + (deltaT*maxBoltz(1,c));
        position(2,c) = position(2,c) + (deltaT*maxBoltz(2,c));


        % Particle behaviour if boundary is reached
        if position(2,c) <= 0 || position(2,c) >= yDim
            maxBoltz(2,c) = (-1)*maxBoltz(2,c);
        end
        if position(1,c) <= 0
            position(1,c) = xDim;
            lastPos(1,c) = xDim;
        elseif position(1,c) >= xDim
            position(1,c) = 0;
            lastPos(1,c) = 0;
        end

        % Particle behaviour at bottle neck
        if position(2,c)<=0.4e-7
            if position(1,c)>=0.8e-7 && lastPos(1,c)<=0.8e-7
                maxBoltz(1,c) = (-1)*maxBoltz(1,c);
                position(1,c) = 0.79e-7;
            end
            if position(1,c)<=1.2e-7 && lastPos(1,c)>=1.2e-7
                maxBoltz(1,c) = (-1)*maxBoltz(1,c);
                position(1,c) = 1.21e-7;
            end
        end
        if position(2,c)>=0.6e-7
            if position(1,c)>=0.8e-7 && lastPos(1,c)<=0.8e-7
                maxBoltz(1,c) = (-1)*maxBoltz(1,c);
                position(1,c) = 0.79e-7;
            end
            if position(1,c)<=1.2e-7 && lastPos(1,c)>=1.2e-7
                maxBoltz(1,c) = (-1)*maxBoltz(1,c);
                position(1,c) = 1.21e-7;
            end
        end
        if position(1,c)>0.8e-7 && position(1,c)<1.2e-7
```

```
                if position(2,c)>=0.6e-7 && lastPos(2,c)<=0.6e-7
                    maxBoltz(2,c) = (-1)*maxBoltz(2,c);
                    position(2,c) = 0.59e-7;
                end
                if position(2,c)<=0.4e-7 && lastPos(2,c)>=0.4e-7
                    maxBoltz(2,c) = (-1)*maxBoltz(2,c);
                    position(2,c) = 0.41e-7;
                end
            end


        end

        sumVelocity = sum(sqrt(sum(maxBoltz.^2)));

        % Plot new particle positions.
        figure(6)
        for c = 1:10
            plot([lastPos(1,c) position(1,c)],[lastPos(2,c)
     position(2,c)],'color',color(c,:))
            xlim([0 2e-7]);
            ylim([0 1e-7]);
            title('Figure 6: Scattering of Electrons with Bottle Neck')
            hold on
            rectangle('Position',[0.8e-7 0 0.4e-7 0.4e-7])
            rectangle('Position',[0.8e-7 0.6e-7 0.4e-7 0.4e-7])
        end

        % Calculate average velocity (m/s)
        avgVelocity = sumVelocity/numElec;

    end

    % Plot electron density map from final positions
    figure(7)
    hist3(position','CdataMode','auto','Nbins',[50 25])
    colorbar
    view(2)
    title('Figure 7: Electron Density Plot')

    density = hist3(position', [50 25]);
    temperatures = density.*(averageVelocity.^2).*mn./(2*k);
    figure(8)
    [x,y] = meshgrid(1:25,1:50);
    surf(x,y,temperatures)
    title('Figure 8: Temperature Map')
```

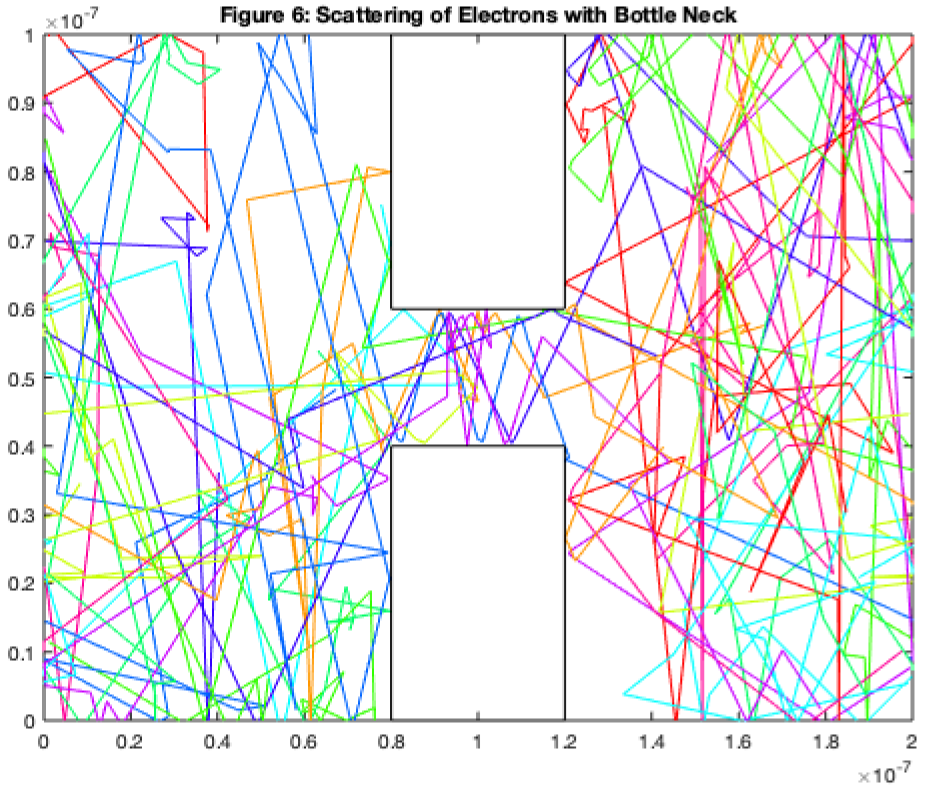


Figure 6: Scattering of Electrons with Bottle Neck
×10^-7
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
×10^-7

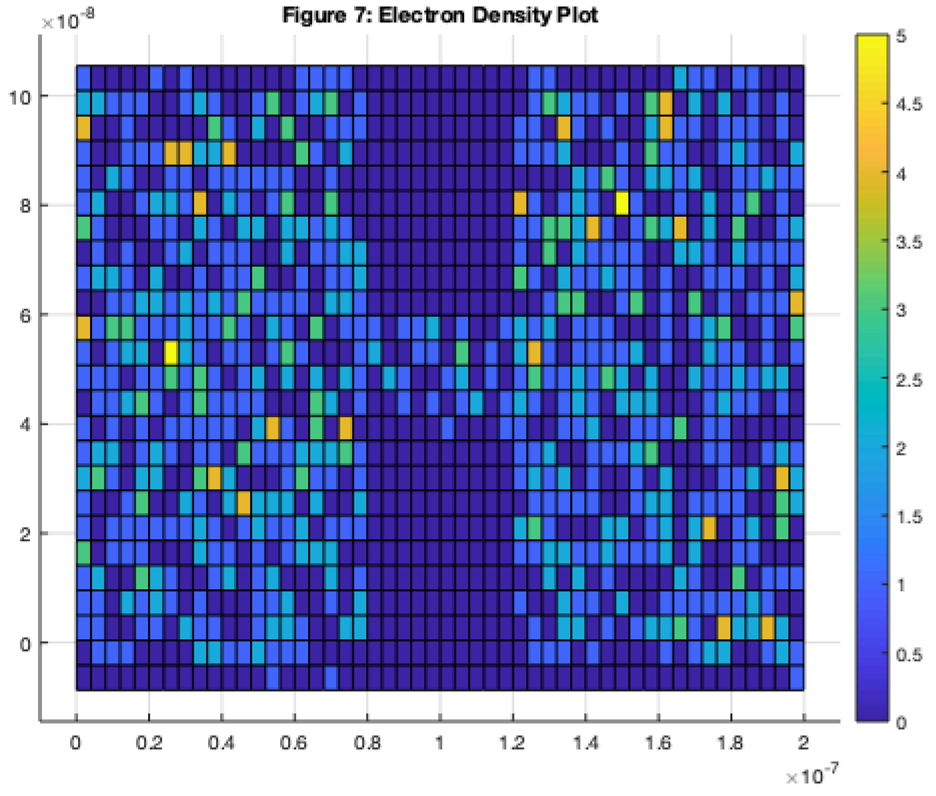
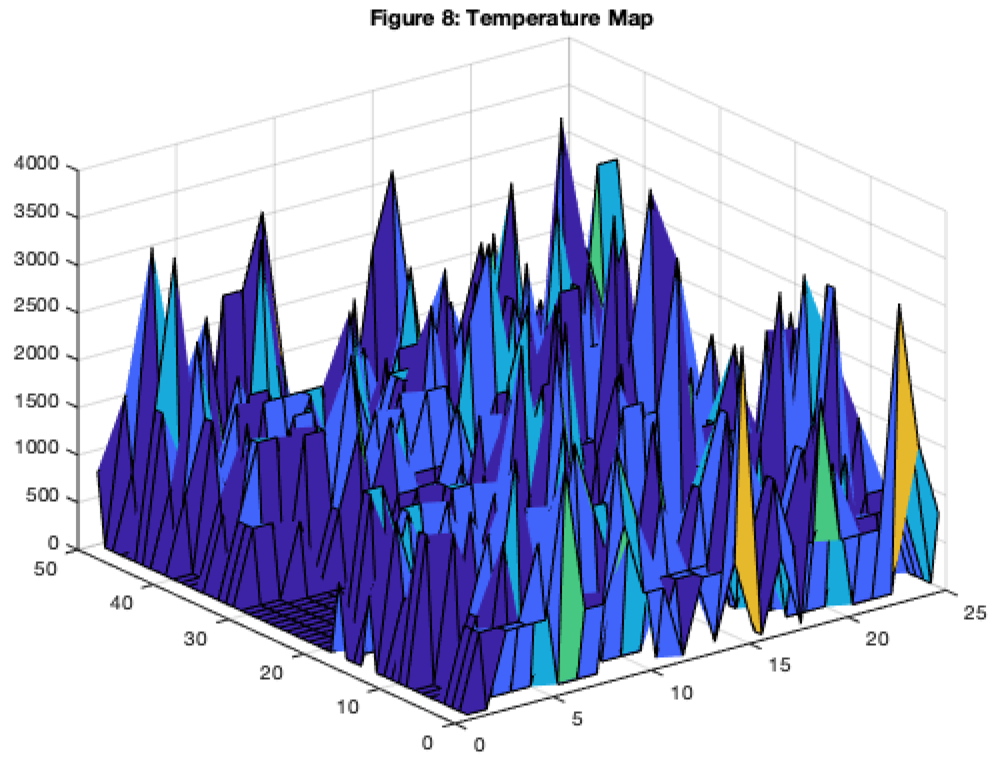


Figure 7: Electron Density Plot
×10^-8
10
8
6
4
2
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
×10^-7
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0



Published with MATLAB® R2019b