Assignment 3

Table of Contents

Matthew Lazarus 100962142

Question 1: Extension of Assignment 1

Here we extend work done in assignment 1 by adding acceleration due to an electric field.

```
% Clear all previous variables, figures, etc, to ensure that the
 workspace
% is clean.
clear all
clearvars
clearvars -GLOBAL
close all

%Define constants that may need to be used later in the code.
global C
C.q_0 = 1.60217653e-19;             % electron charge
C.hb = 1.054571596e-34;             % Dirac constant
C.h = C.hb * 2 * pi;                    % Planck constant
C.m_0 = 9.10938215e-31;             % electron mass
C.kb = 1.3806504e-23;               % Boltzmann constant
C.eps_0 = 8.854187817e-12;          % vacuum permittivity
C.mu_0 = 1.2566370614e-6;           % vacuum permeability
C.c = 299792458;                    % speed of light
C.g = 9.80665; %metres (32.1740 ft) per sÂ²
```

(A)

If 0.1 volts is applied over the x direction of the semiconductor, which is 200nm un length, the electric field is: $E = \Delta * V/d = 500kV/m$

(B)

Using the result from (A) and knowing that 1V/m=1N/C, the force applied on each electron can be found: $F = E * q = 8.0109 * 10^{-14}N$

(C)

From (B), we can calculte the acceleration of each electron. The full code for calculating acceleration from the applied voltage can be seen below. $a = F/m = 3.3825 * 10^{17} m/s^2$

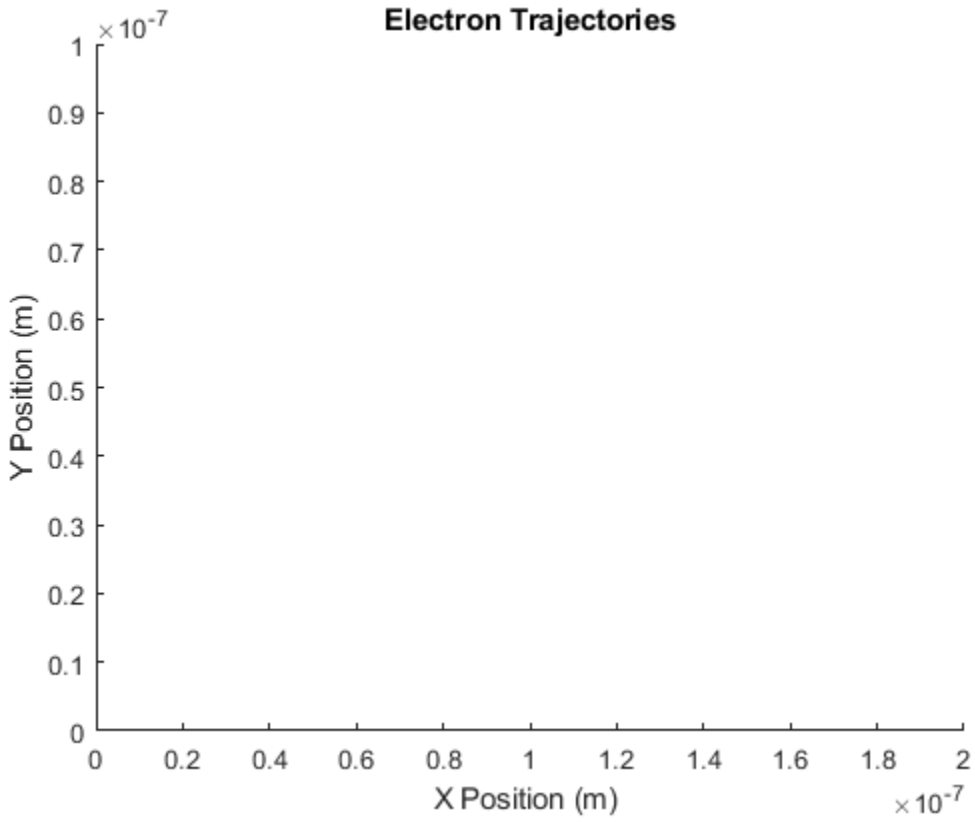
```
vAppX = 1; %V
vAppY = 0; %V
accX  = vAppX*C.q_0/(0.26*C.m_0*200e-9); %m/s^2
accY  = vAppY*C.q_0/(0.26*C.m_0*100e-9); %m/s^2

%Thermal Velocity at 300K:
vth = sqrt(C.kb*300/(0.26*C.m_0));

%Find Mean Free Path
tmn = 0.2*10^-12;
freePath = tmn * vth;

% Set the number of electrons, time step and total time. Initialize
% matrices for the x and y positions, the x and y components of the
% velocity, and the temperature of the system. Column 1 of each
 matrice is
% the previous value, while column 2 is the current value.
numElectrons=20000;
dt = 1.5e-15; %seconds
nTime = 1.5e-12; %Simulation length
x = zeros(numElectrons,2); %Position (x)
y = zeros(numElectrons, 2); %Position (y)
vx = zeros(numElectrons, 2); %Velocity (x)
vy = zeros(numElectrons, 2); %Velocity (y)
vTotal = zeros(numElectrons, 2); %Velocity
temperature = zeros(numElectrons,2);
Ix = zeros(numElectrons,2);

% Define two vectors to keep track of the time and distance since the
 last
% scatter, for each electron. Additionally initalize a counter for
 them.
distanceSinceLastScatter = zeros(numElectrons,1);
timeSinceLastScatter = zeros(numElectrons,1);
scatterCount = 0;
```

Now, randomly assign initial positions & velocities. Choose each velocity component based off of the Gaussian distribution. Divide by the square root of 2 as there are 2 degrees of freedom. This is derived from equating the acceleration of the electron to its kinetic energy, as seen by the equation below.

$$(1/2)mv^2 = (2/2)kT$$

```
for electronCount = 1:numElectrons
    x(electronCount,2)=rand()*200e-9;
    y(electronCount,2)=rand()*100e-9;

    vx(electronCount,2) = vth * randn()/sqrt(2);
    vy(electronCount,2) = vth * randn()/sqrt(2);
```

```
    vTotal(electronCount,2)=
 sqrt(vx(electronCount,2)^2+vy(electronCount,2)^2);
end

% Create a figure for the electron trajectories and the temperature of
 the
% system.
figure(1)
title('Electron Trajectories')
xlabel('X Position (m) ')
ylabel('Y Position (m)')
axis([0 200e-9 0 100e-9]);

% figure(2)
% title('Temperature')
% xlabel('Time (s)')
% ylabel('Temperature (K)')
%axis([0 (nTime) 250 350]);
```

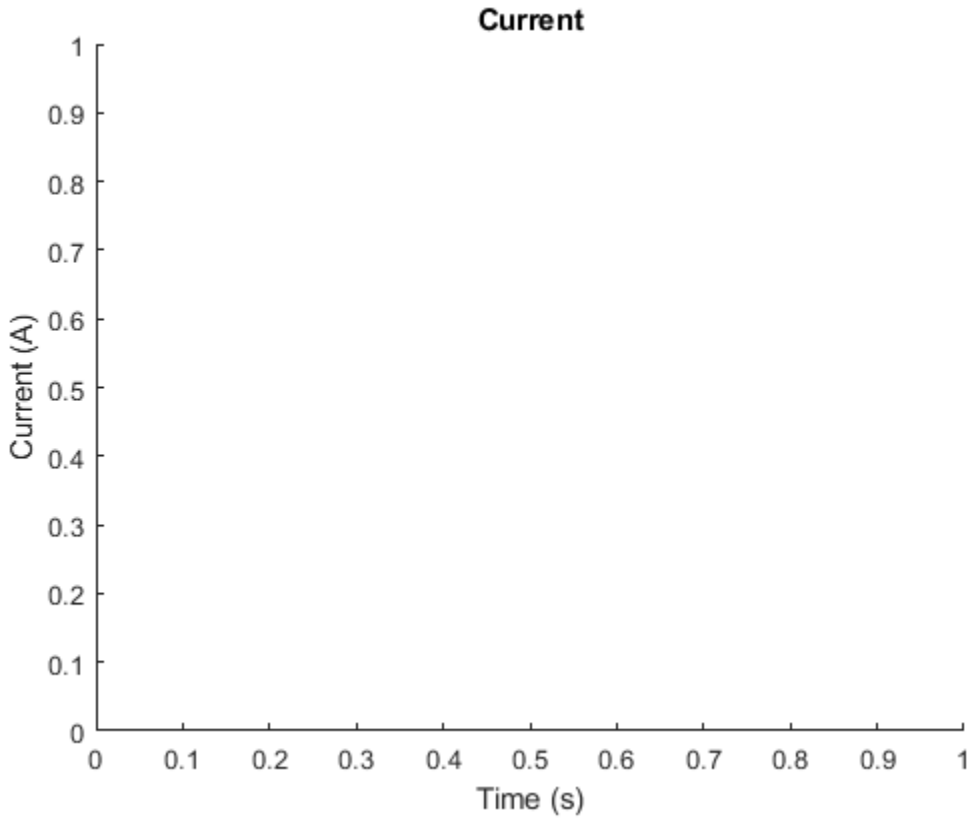


(D)

The relationship between the electron drift current density and the average carrier density is: $J_d = n * q * v_{ave}/A$ Where n is the number of electrons.

Therefore, the current is: $I = n * q * v_{ave}$

```
figure(3)
title('Current')
xlabel('Time (s)')
ylabel('Current (A)')
%axis([0 (nTime) 250 350]);

% Define a vector that will indicate whether an electron crosses a
 horizontal
% boundary. As only 5 electrons will be plotted, if the electron cross
 the
% boundary, a 1 will be set in the position of the vector that
 corresponds
% to the number of the electron (1-5).
xBreakpoint = zeros(5);
```



Electron Trajectories and Temperature

Using the probability of scattering equation, seen below,the probability of scattering was determined and compared to a random number between 0 and 1. If the number is less than the probability of scattering, the electron will scatter. This means that it will be given new velocity component values, calculated using the Gaussian distribution as was done above. Lastly, the every time the electron scatters, the time and distance since its last scatter is recorded.

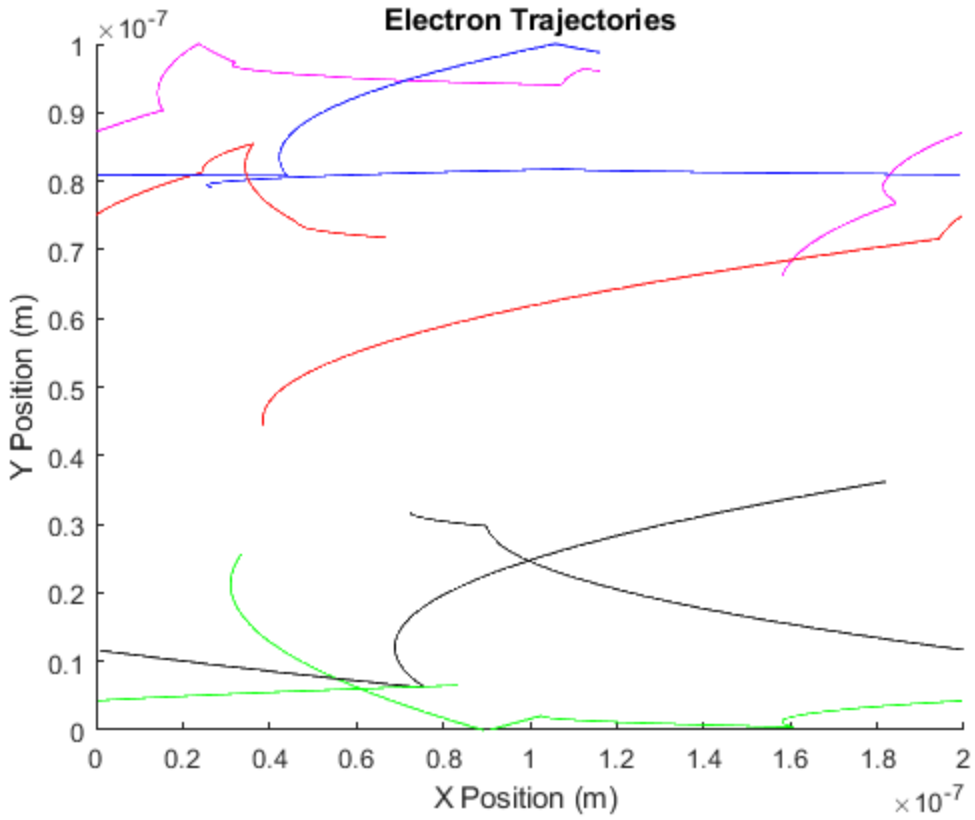
$$P_{scat} = 1 - e^{-dt/\tau_{mn}}$$

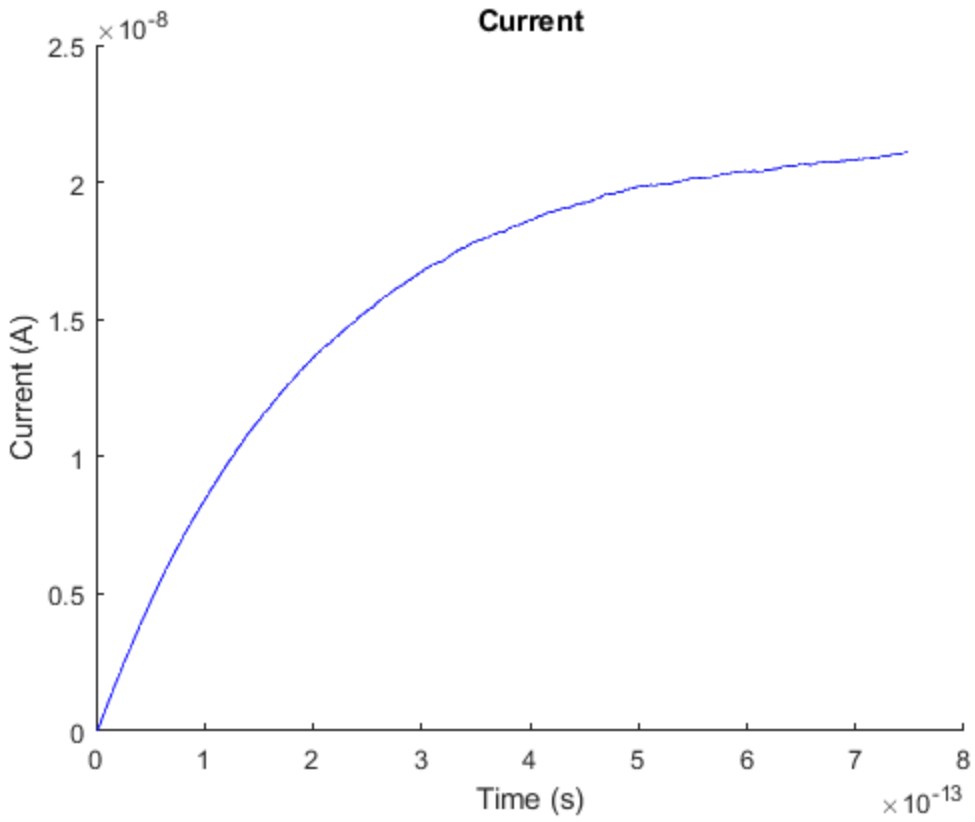
```
% Run simulation over time.
```

```
%ceil((nTime)/dt)
for count =  1:500
 % Run through each electron.
    for c = 1:numElectrons
        if(count~=1)
            % Calculate probability of scattering.
            Pscat = 1-exp(-dt/tmn);

            %  Update the previous positions and velocities.
            vx(c,1)=vx(c,2);
            vy(c,1)=vy(c,2);
            x(c,1)=x(c,2);
            y(c,1)=y(c,2);

            % C: Update the velocities due to the applied voltage
            vx(c,2) = vx(c,2) + accX*dt;
            vy(c,2) = vy(c,2) + accY*dt;

            % Update the current position of the electron.
            x(c,2) = x(c,1) + vx(c, 2)*dt;
            y(c,2) = y(c,1) + vy(c, 2)*dt;

            %Update time and distance since the last scatter

 distanceSinceLastScatter(c)=distanceSinceLastScatter(c)+sqrt((x(c,2)-
x(c,1))^2+(y(c,2)-y(c,1))^2);
            timeSinceLastScatter(c)=timeSinceLastScatter(c)+dt;

            % Check to see if an electron hit a boundary. If it hit a
            % horizontal boundary, move it to the other side of the
 grid
            % (with the same velocity). If it hit a vertical boundary,
 it
            % should bounce off.
            if(x(c,2)>200e-9)
                x(c,2) = x(c,2)-200e-9;
                if(c<6 && c>0)
                    xBreakpoint(c)=1;
                end
            elseif(x(c,2)<0)
                x(c,2)=x(c,2)+200e-9;
                if(c<6 && c>0)
                   xBreakpoint(c)=1;
                end
            end
            if(y(c,2)>=100e-9)
                vy(c,2) = -vy(c,2);
            elseif(y(c,2)<=0)
                vy(c,2)=-vy(c,2);
            end
            r=rand();
             %Check if the electron scatters
            if(Pscat>r)
```

```
                %Scattering occurs. Update velocity components, then
 save
                % and clear scattering time and distance.
                  vx(c,2) = vth * randn()/sqrt(2);
                  vy(c,2) = vth * randn()/sqrt(2);

                  scatterCount = scatterCount +1;
                  scatterDistances(scatterCount) =
 distanceSinceLastScatter(c);
                  scatterTimes(scatterCount)=timeSinceLastScatter(c);
                  distanceSinceLastScatter(c)=0;
                  timeSinceLastScatter(c)=0;
              end
          end
      end

      if(count>1)
          % Plot the displacement of the electrons in different colours.
          figure(1)
          hold on
          if(xBreakpoint(1)~=1)
              plot(x(1,1:2),y(1,1:2),'b')
          end
          if(xBreakpoint(2)~=1)
              plot(x(2,1:2),y(2,1:2),'r')
          end
          if(xBreakpoint(3)~=1)
              plot(x(3,1:2),y(3,1:2),'g')
          end
          if(xBreakpoint(4)~=1)
              plot(x(4,1:2),y(4,1:2),'k')
          end
          if(xBreakpoint(5)~=1)
              plot(x(5,1:2),y(5,1:2),'m')
          end
          hold off
      end

      % Reset xBreakpoint.
      xBreakpoint(:)=0;

      % Update the previous and current temperature values. Plot the
  change
      % in temeperature over the step in time.
      temperature(:,1)=temperature(:,2);
      temperature(:,2) = (vx(:,2).^2 + vy(:,2).^2).*((0.26*C.m_0))./
C.kb;
%     if(count>1)
%         figure(2)
%         hold on
%         plot([(count-1)*dt,count*dt],
[mean(temperature(:,1)),mean(temperature(:,2))],'r');
%         hold off
%     end
```

```
    % Update the current values. Plot the change
    % in current over the step in time.
    Ix(:,1)=Ix(:,2);
    Ix(:,2)= C.q_0*(vx(:,2))*(10^15/(1e-4))*(200e-9*100e-9);
    if(count>1)
        figure(3)
        hold on
        plot([(count-1)*dt,count*dt],
[mean(Ix(:,1)),mean(Ix(:,2))],'b');
        hold off
    end
    pause(0.000001)
end
```



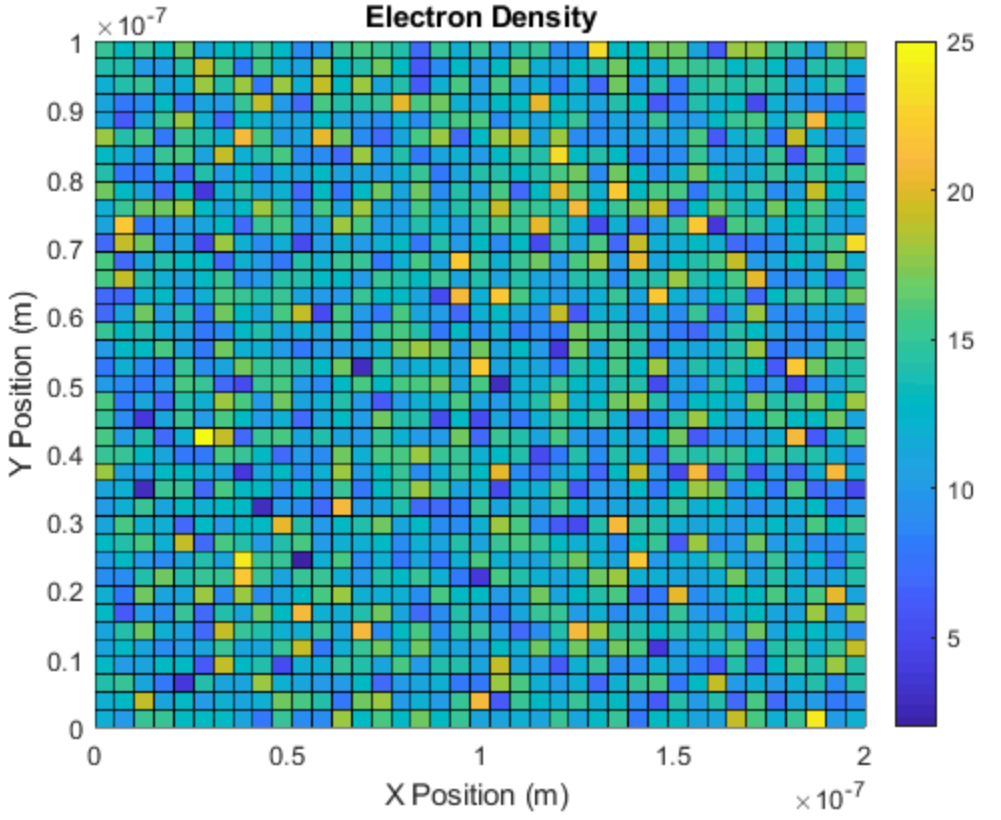


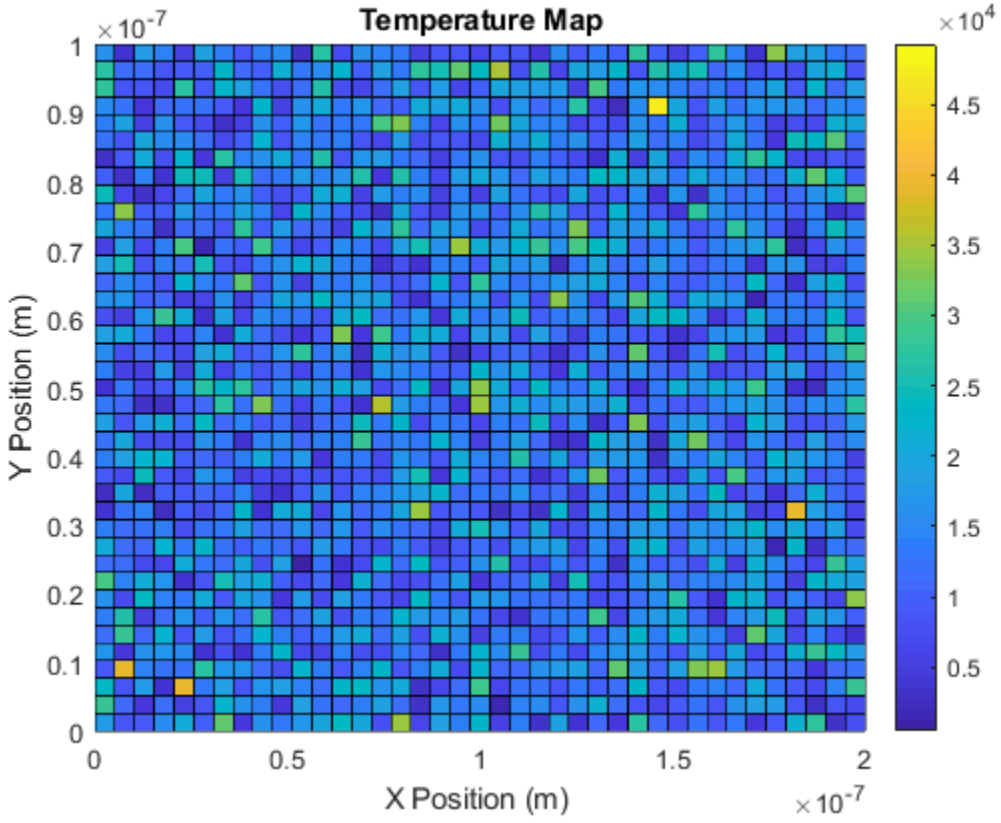
Electron Density and Temperature Map

Divide the total grid into 40x40 grid for electron density and velocity (this will give 1600 "boxes"). For each electron, find its corresponding box and sum the number of electrons and total temperature for each box. Ensure no electrons are in the barrier regions.

```
electronCount=zeros(40,40);
electronTemperature=zeros(40,40);
for  eCount = 1:numElectrons
    % Find the box associated with the electron.
    xVal = ceil(40*x(eCount,2)/200e-9);
    yVal = ceil(40*y(eCount,2)/100e-9);
    if(xVal<1)
        xVal=1;
    end
    if(yVal<1)
        yVal =1;
    end
    if(xVal>40)
        xVal = 40;
    end
    if(yVal>40)
        yVal=40;
    end

    % Add to the total electron count and total temperature.
```

```
    electronCount(yVal,xVal) =electronCount(yVal,xVal)+1;
    electronTemperature(yVal, xVal) = electronTemperature(yVal, xVal)
 + temperature(eCount,2);
end

% Find the average temperature of each box.
electronTemperature = electronTemperature ./ electronCount;

% Plot the Electron Density and the Temperature Map
figure(4)
xAxis = linspace(0,200e-9,40);
yAxis = linspace(0,100e-9,40);
surf(xAxis,yAxis, electronCount);
title('Electron Density')
xlabel('X Position (m) ')
ylabel('Y Position (m)')
zlabel('Number of Electrons')
view(2)
colorbar

figure(5)
surf(xAxis,yAxis, electronTemperature);
title('Temperature Map')
xlabel('X Position (m) ')
ylabel('Y Position (m)')
zlabel('Temperture')
view(2)
colorbar
```


Electron Density
×10^-7
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Y Position (m)
0
0.5
1
1.5
2
X Position (m)
×10^-7
25
20
15
10
5


Temperature Map
×10^-7
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Y Position (m)
0
0.5
1
1.5
2
X Position (m)
×10^-7
×10^4
4.5
4
3.5
3
2.5
2
1.5
1
0.5

Published with MATLAB® R2018a