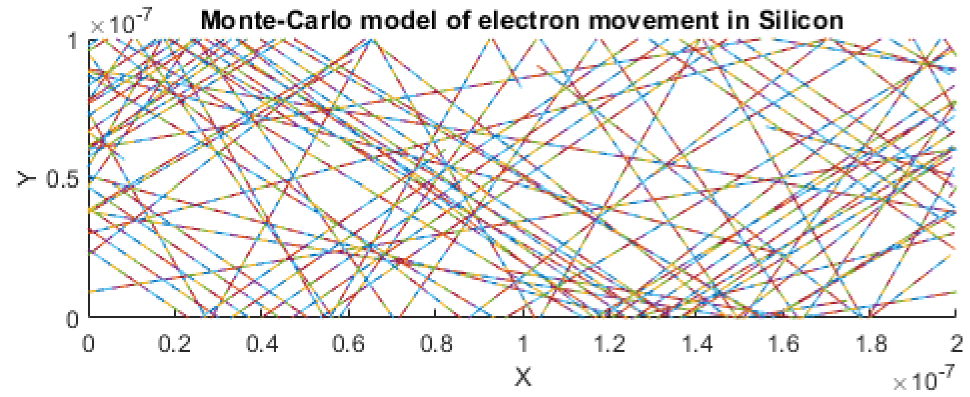
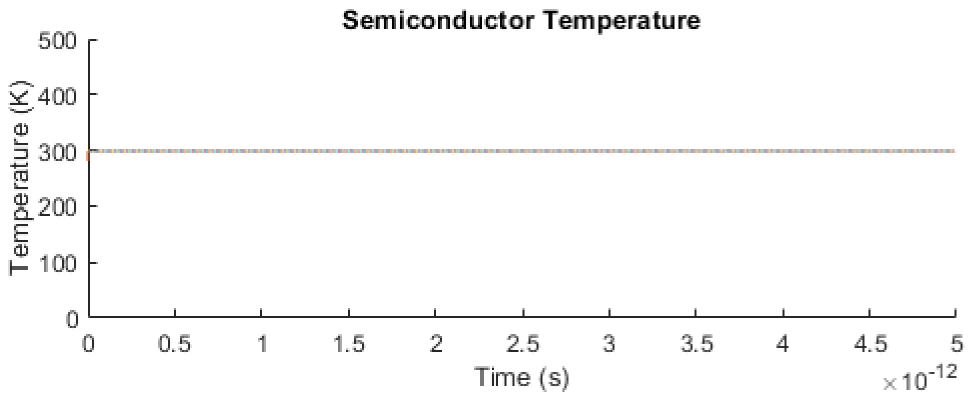
```
% % ELEC 4700 Assignment 1: Monte-Carlo Modelling of Electrons
% % Mike Crawford
% % 100952432
% % Question1
%%Would like the one on one evaluation please
% This code is designed to simulate a specified number of electrons movement
% in silicon using the Monte-Carlo method. The effective mass is known and
% each electron is given a random direction. Boundaries were put on the box
% to allow electrons to pass through the other side of the y plane and
% reflect off the x plane. The thermal velocity is calculated by the equation
%
% Vth = sqrt((C.k*T)/(mn)
%
% The parameters are labeled below. The mean free path is can be found with
% the thermal velocity and the mean time between collisions, which is given
% as 0.2 ps. The simulation plots these electrons during the specified
% timeframe as well as the average semiconductor temperature over time.

%reset
clearvars
clearvars -GLOBAL
close all
format shortE
global C
global Ecount
global Vx Vy Vtotal x y
Ecount =5000;   %setting amount of electrons used
C.mo = 9.10938215e-31;  %Electron mass
C.k = 1.3806504e-23;    %Boltzmann constant

T =300;     %Temperature (K)
mn = 0.26*C.mo;
L = 200e-9; %Length of Border
W = 100e-9; %Width of Border
Vth = sqrt(2*(C.k*T)/mn); %Calculation of Thermal Velocity
dt = 10e-15; %timestep
Stop =500*dt; %timeframe
x = zeros(Ecount, 2);   %array of x positions
y = zeros(Ecount, 2);   %array of y positions

Temperature = zeros(1,2);   %array of temperatures
Time = 0;
VisibleEcount = 10; %number of visible electrons

for i = 1:Ecount %setting random locations for eact electron
    x(i,1) = rand()*200e-9;
    y(i,1) = rand()*100e-9;
end

Vx(1:Ecount) = Vth * cos(2*pi*randn(1, Ecount)); %setting velocities
```

```
Vy(1:Ecount) = Vth * sin(2*pi*randn(1, Ecount));
for i = 1:Ecount
    Vtotal(i) = sqrt(Vx(i)^2 + Vy(i)^2); %summing x and y velocities
end


figure(1)   %plotting electrons
subplot(2,1,1);
axis([0 L 0 W]);
title('Monte-Carlo model of electron movement in Silicon');
xlabel('X');
ylabel('Y');
hold on;

subplot(2,1,2); %plotting avg temperature
axis([0 Stop 0 500]);
title('Semiconductor Temperature');
xlabel('Time (s)');
ylabel('Temperature (K)');
hold on;

for i = 1:Ecount %Sum of all initial temperatures
   Temperature(1,2) = Temperature(1,2) + (mn*Vtotal(i)^2)/(2*C.k);
end
%intitial average temperature
AvgTemperature = Temperature(1,2)/Ecount;
TemperaturePlot = [300 AvgTemperature]; %setting temperature line to plot
TimePlot = [0 Time];    %setting time line to plot
plot(Time, AvgTemperature);

%reseting values before loop
VTotal = 0;
Temperature(1,2) = 0;
AvgTemperature = 0;

%looping through timesteps
while Time < Stop
    subplot(2,1,1)
    for j = 1:Ecount %looping through electrons
        x(j,2) = x(j,1); %updating previous positions
        y(j,2) = y(j,1);
        x(j,1) = x(j,1) + (dt * Vx(j));%updating new positions
        y(j,1) = y(j,1) + (dt * Vy(j));
        if x(j,1) > L %setting right side boundary condition
            x(j,2) = 0;
            x(j,1) = dt * Vx(j);
        end
        if x(j,1) < 0 %setting left side boundary condition
            x(j,2) = L;
            x(j,1) = x(j,2) + (dt * Vx(j));
        end
        %setting roof/floor boundary condition
        if y(j,1) > W || y(j,1) < 0
            Vy(j) = -Vy(j);
        end
        %setting up position line vectors to plot
        XPlot = [x(j,2) x(j,1)];
        YPlot = [y(j,2) y(j,1)];
        if j < VisibleEcount
            %plot visible position line vectors
        plot(XPlot,YPlot);
        end

        %sum of x and y  velocities
       VTotal = sqrt(Vx(j)^2 + Vy(j)^2);

        %sum of all temperatures
       Temperature(1,2) = Temperature(1,2) + (mn*Vtotal(j)^2)/(2*C.k);

    end
    %averaging temperature sum and plotting against each timestep
    AvgTemperature = Temperature(1,2)/Ecount;
    TemperaturePlot = [Temperature(1,1) AvgTemperature];
    TimePlot = [(Time - dt) Time];
    subplot(2,1,2);
    plot(TimePlot, TemperaturePlot);
    Temperature(1,1) = AvgTemperature;
    AvgTemperature = 0;
    Temperature(1,2) = 0;
    pause(1e-19)
    Time = Time + dt;
end
```





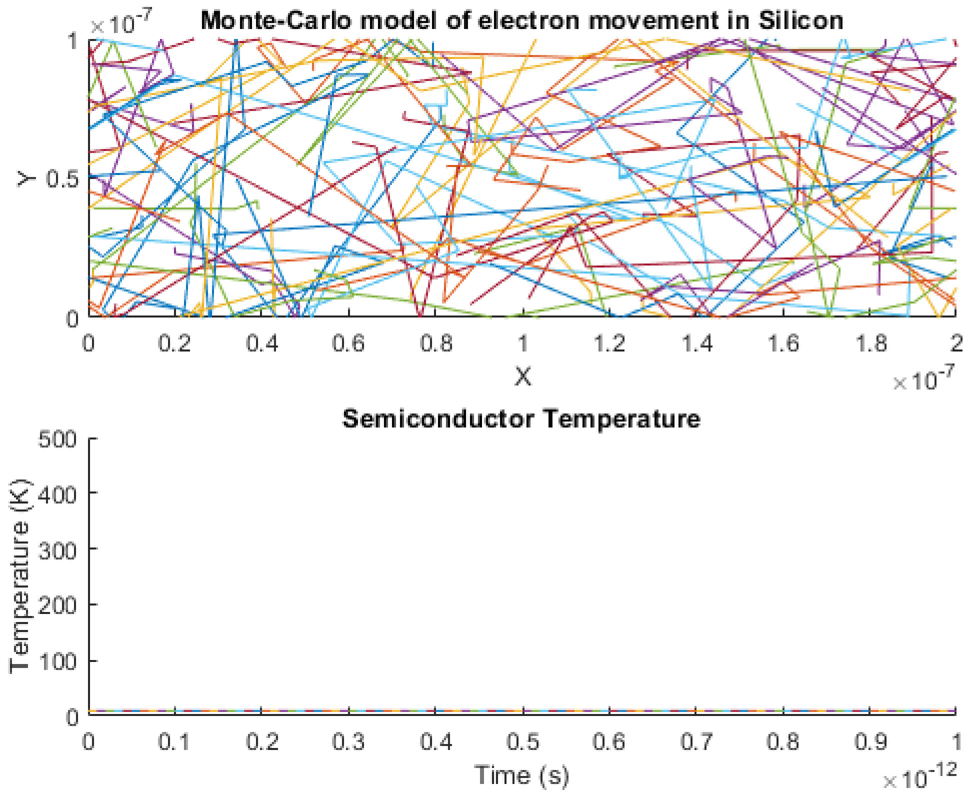
Published with MATLAB® R2018a

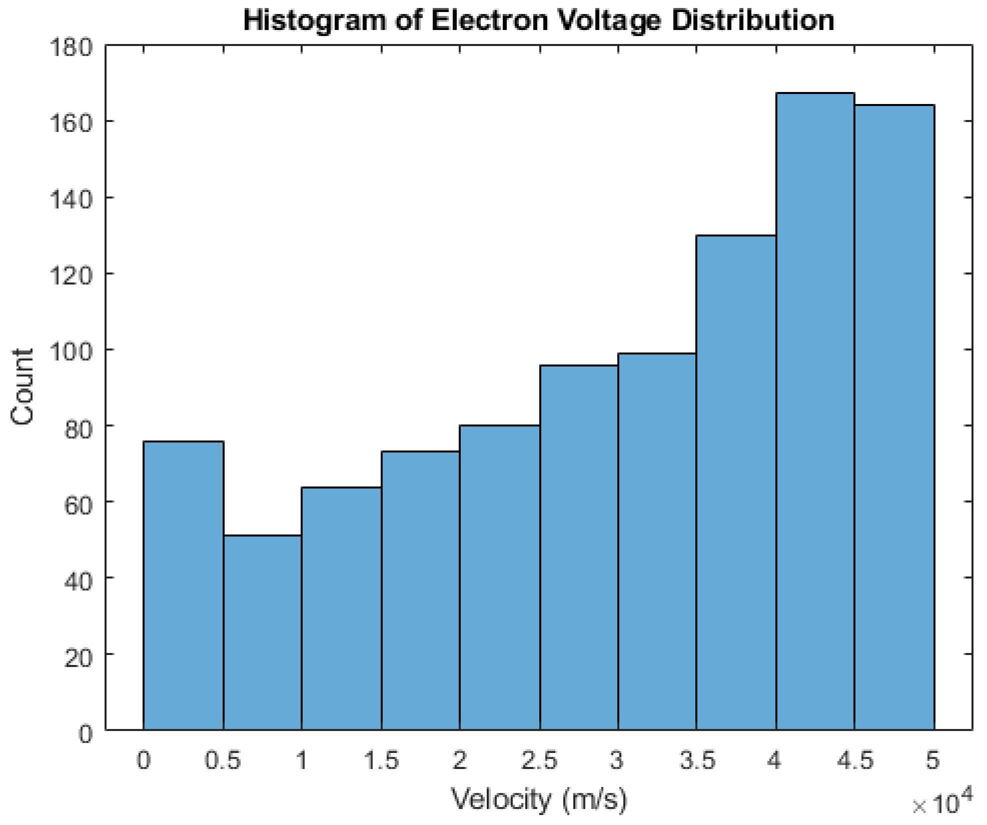
```
% % ELEC 4700 Assignment 1: Monte-Carlo Modelling of Electrons
% % Mike Crawford
% % 100952432
% % Question2
% This next part is similar to the previous, with the same model of electrons
% and average tmeperatures. The difference is it adds a collission factor
% with a scattering probability as welll as a random velocity with each
% electron. A histogram showing the distribution of diferent velocities is
% also given. This part is using the same code skelaton as the first part
%so much of the code will look the same

%reset
clearvars
clearvars -GLOBAL
close all
format shortE
global C
global Ecount
global Vx Vy Vtotal x y
```

```
Ecount =1000;   %setting amount of electrons used
C.mo = 9.10938215e-31;%Electron mass
C.k = 1.3806504e-23; %Boltzmann constant

T =300; %Temperature (K)
mn = 0.26*C.mo;
L = 200e-9;  %Length of Border
W = 100e-9; %Width of Border
Vth = sqrt((2*C.k*T)/mn); %Calculation of Thermal Velocity
dt = 10e-15; %timestep
Stop = 100*dt; %timeframe
x = zeros(Ecount, 2); %array of x positions
y = zeros(Ecount, 2); %array of y positions

Temperature = zeros(1,2); %array of temperatures
Time = 0;
VisibleEcount = 50; %visibble electrons
tmn = 0.2e-12; %mean time between collisions
PScat = 1 - exp(-dt/tmn); %scattering probability equation
VTotalHistogram = zeros(Ecount, 1);%array of thermal velocities
for i = 1:Ecount
    x(i,1) = rand()*200e-9;%setting random electron positions
    y(i,1) = rand()*100e-9;
end
%setting random velocities
Vx(1:Ecount) = Vth*rand * cos(2*pi*randn(1, Ecount));
Vy(1:Ecount) = Vth*rand * sin(2*pi*randn(1, Ecount));
%sum of x and y velocities
for i = 1:Ecount
    Vtotal(i) = sqrt(Vx(i)^2 + Vy(i)^2);
    VTotalHistogram(i) = sqrt(Vx(i)^2 + Vy(i)^2);

end

%plot of electrons
figure(1)
subplot(2,1,1);
axis([0 L 0 W]);
title('Monte-Carlo model of electron movement in Silicon');
xlabel('X');
ylabel('Y');
hold on;
%plot of average temperatures
subplot(2,1,2);
axis([0 Stop 0 500]);
title('Semiconductor Temperature');
xlabel('Time (s)');
ylabel('Temperature (K)');
hold on;
%initial temperature sum
```

```
for i = 1:Ecount
   Temperature(1,2) = Temperature(1,2) + (mn*Vtotal(i)^2)/(2*C.k);
end
%initial average temperatue
AvgTemperature = Temperature(1,2)/Ecount;
TemperaturePlot = [300 AvgTemperature];
TimePlot = [0 Time];
plot(Time, AvgTemperature);
%resetting before loop
VTotal = 0;
Temperature(1,2) = 0;
AvgTemperature = 0;
%looping timestep
while Time < Stop
    subplot(2,1,1)
    %looping electrons
    for j = 1:Ecount
        %Scattering probability conditioned with random value to simulate
        %random scattering at  set rate
        if PScat> rand
                Vx(j) = Vth * randn;
                Vy(j) = Vth * randn;
        end
        %update previous/new x and y positions
        x(j,2) = x(j,1);
        y(j,2) = y(j,1);
        x(j,1) = x(j,1) + (dt * Vx(j));
        y(j,1) = y(j,1) + (dt * Vy(j));
        %check right wall border
        if x(j,1) > L
            x(j,2) = 0;
            x(j,1) = dt * Vx(j);
        end
        %check left wall border
        if x(j,1) < 0
            x(j,2) = L;
            x(j,1) = x(j,2) + (dt * Vx(j));
        end
        %check roof/floor border
        if y(j,1) > W || y(j,1) < 0
            Vy(j) = -Vy(j);
        end
        %set line vectores for x and y positions
        XPlot = [x(j,2) x(j,1)];
        YPlot = [y(j,2) y(j,1)];
        %plot visible x and y line vectors
        if j < VisibleEcount
        plot(XPlot,YPlot);
        end
```

```
         %sum of themal velocities and temperatures
        VTotal = sqrt(Vx(j)^2 + Vy(j)^2);
        Temperature(1,2) = Temperature(1,2) + (mn*Vtotal(j)^2)/(2*C.k);

    end
    %plotting average temperature and ressetting values for next timestep
    AvgTemperature = Temperature(1,2)/Ecount;
    TemperaturePlot = [Temperature(1,1) AvgTemperature];
    TimePlot = [(Time - dt) Time];
    subplot(2,1,2);
    plot(TimePlot, TemperaturePlot);
    Temperature(1,1) = AvgTemperature;
    AvgTemperature = 0;
    Temperature(1,2) = 0;
    pause(1e-19)
    Time = Time + dt;
end
%histogram of thermal velocities
figure(2)
histogram(VTotalHistogram)
title('Histogram of Electron Voltage Distribution')
xlabel('Velocity (m/s)')
ylabel('Count')
```


Monte-Carlo model of electron movement in Silicon
Y
X
×10^-7
Semiconductor Temperature
Temperature (K)
Time (s)
×10^-12


Histogram of Electron Voltage Distribution
Count
Velocity (m/s)
×10^4

Published with MATLAB® R2018a

```
% % ELEC 4700 Assignment 1: Monte-Carlo Modelling of Electrons
% % Mike Crawford
% % 100952432
% % Question3
% This final part add onto the previous by adding two boxes that from a
% bottleneck inside the borders of the simulation. These boxes should reflect
% any electrons coliding into it and stop elctrons from beign positioned
% inside of it. The boxes should be able to ither be specular or diffusive.
% Meaning after an electron bounces, its either given the reflected velocity
% a random velocity respectively. After the simulation is completed, a map
% of both the electron density and the temperature will be displayed. \

%reset
clearvars
clearvars -GLOBAL
close all
format shortE
global C
global Ecount
global Vx Vy Vtotal x y
Ecount =1000;   %setting amount of electrons used
C.mo = 9.10938215e-31; %Electron mass
C.k = 1.3806504e-23; %Boltzmann constant

T =300; %Temperature (K)
mn = 0.26*C.mo;
L = 200e-9;  %Length of Border
W = 100e-9; %Width of Border
Vth = sqrt((2*C.k*T)/mn); %Calculation of Thermal Velocity
dt = 10e-15;  %timestep
Stop = 100*dt; %timeframe
x = zeros(Ecount, 2);  %array of x positions
y = zeros(Ecount, 2);  %array of y positions
Temperature = zeros(1,2); %array of temperatures
Time = 0;
VisibleEcount = 50; %visibble electrons
tmn = 0.2e-12; %mean time between collisions
PScat = 1 - exp(-dt/tmn);  %scattering probability equation
VTotalHistogram = zeros(Ecount, 1); %array of thermal velocities
DrawBoxX = [80e-9 80e-9 120e-9 120e-9 80e-9]; %X border limits of boxes
%(Same X values for both)
DrawBoxY1 = [100e-9 60e-9 60e-9 100e-9 100e-9];%Y border limits of top box
DrawBoxY2 = [40e-9 0 0 40e-9 40e-9]; %Y border limits of bottom box
Specular = true; %boolean indicating whether specular or diffusive
InsideBox = true; %boolean showing if electron spawned inside boxes
MappingStep = 10e-9; %size of each suare used on mapping grid
```

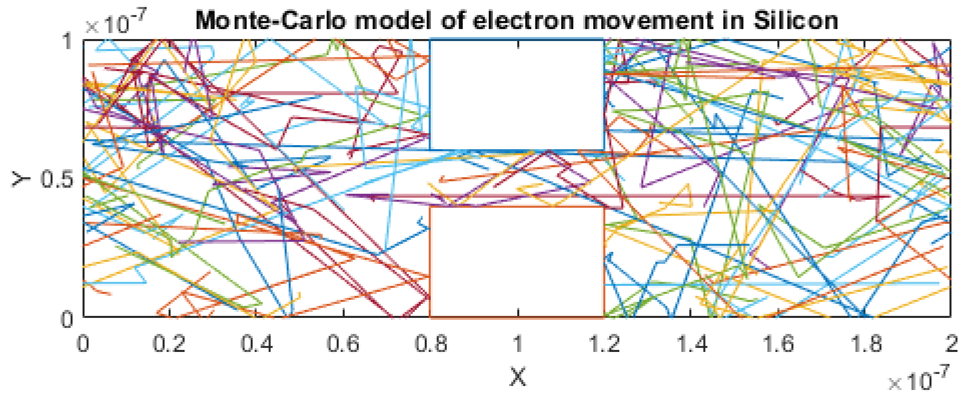
```
%setting arrays of density and temperature based off border demesions and
%mapping step
DensityMap = zeros(W/MappingStep, L/MappingStep);
TemperatureMap = zeros(W/MappingStep, L/MappingStep);
for i = 1:Ecount
    x(i,1) = rand()*200e-9; %setting random electron positions
    y(i,1) = rand()*100e-9;
    InsideBox = true;
    %checks if set electron is inside box, if so reinitialize its position
    while InsideBox == true
        if (x(i) >= 40e-9 && x(i) <= 120e-9) && (y(i) >= 60e-9 ||...
                y(i) <= 40e-9)
            x(i,1) = rand * 200e-9;
            y(i,1) = rand * 100e-9;
        else
            InsideBox = false;
        end
    end

end
%set random velocities
Vx(1:Ecount) = Vth*rand * cos(2*pi*randn(1, Ecount));
Vy(1:Ecount) = Vth*rand * sin(2*pi*randn(1, Ecount));
%sum of x and y velocities
for i = 1:Ecount
    Vtotal(i) = sqrt(Vx(i)^2 + Vy(i)^2);
    VTotalHistogram(i) = sqrt(Vx(i)^2 + Vy(i)^2);

end

%plot of electrons with boxes
figure(1)
subplot(2,1,1);
plot(DrawBoxX, DrawBoxY1, DrawBoxX, DrawBoxY2)
axis([0 L 0 W]);
title('Monte-Carlo model of electron movement in Silicon');
xlabel('X');
ylabel('Y');
hold on;
%set up plot for temperatures
subplot(2,1,2);
axis([0 Stop 0 500]);
title('Semiconductor Temperature');
xlabel('Time (s)');
ylabel('Temperature (K)');
hold on;
%summing temperatures
for i = 1:Ecount
   Temperature(1,2) = Temperature(1,2) + (mn*Vtotal(i)^2)/(2*C.k);
end
```

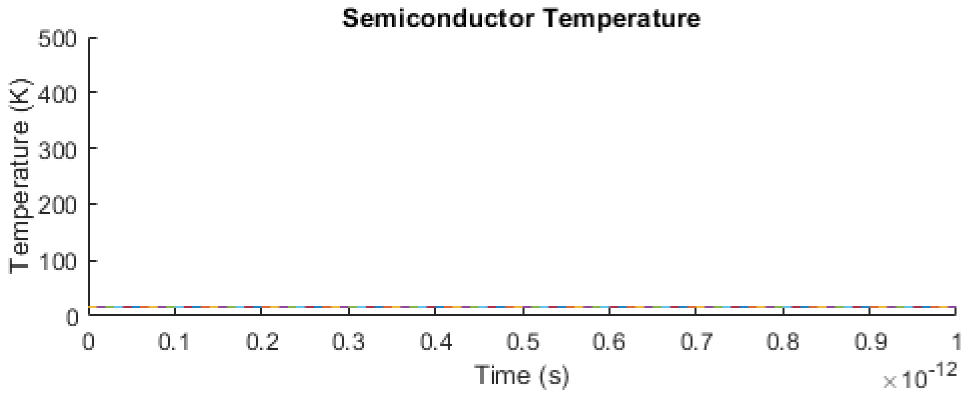
```
%averaging temperatures and plotting over timestep
AvgTemperature = Temperature(1,2)/Ecount;
TemperaturePlot = [300 AvgTemperature];
TimePlot = [0 Time];
plot(Time, AvgTemperature);
%reset variables before loop
VTotal = 0;
Temperature(1,2) = 0;
AvgTemperature = 0;
%looping timestep
while Time < Stop
    subplot(2,1,1)
    %looping electrons
    for j = 1:Ecount
        %set to assume leaking to start condition for diffusive bounces
        leaking = true;
        %scattering probability condition
        if PScat> rand
                Vx(j) = Vth * randn;
                Vy(j) = Vth * randn;
        end
        %updating old/new x and y positions
        x(j,2) = x(j,1);
        y(j,2) = y(j,1);
        x(j,1) = x(j,1) + (dt * Vx(j));
        y(j,1) = y(j,1) + (dt * Vy(j));
        %checking if collision with top box
        if (x(j,1) >= 80e-9 && x(j,1) <= 120e-9) && y(j,1) >= 60e-9
            %checking which side of box collision came from
            %if left side, reflect X velocity and reset positions outside
            %of box
                if y(j,2) < 60e-9
                    Vy(j) = -Vy(j);
                    y(j,1) = 60e-9;
                    y(j,2) = 60e-9;
                    %if right side, reflect X velocity and reset position
                    %outside of box
                elseif x(j,2) < 80e-9
                    Vx(j) = -Vx(j);
                    x(j,1) = 80e-9;
                    x(j,2) = 80e-9;
                    %if bottom side, reflect Y velocity and reset position
                    %outside of box
                elseif x(j,2) > 120e-9
                    Vx(j) = -Vx(j);
                    x(j,1) = 120e-9;
                    x(j,2) = 120e-9;
                end
                %check if specular or diffusive collisions
            if Specular == true
```

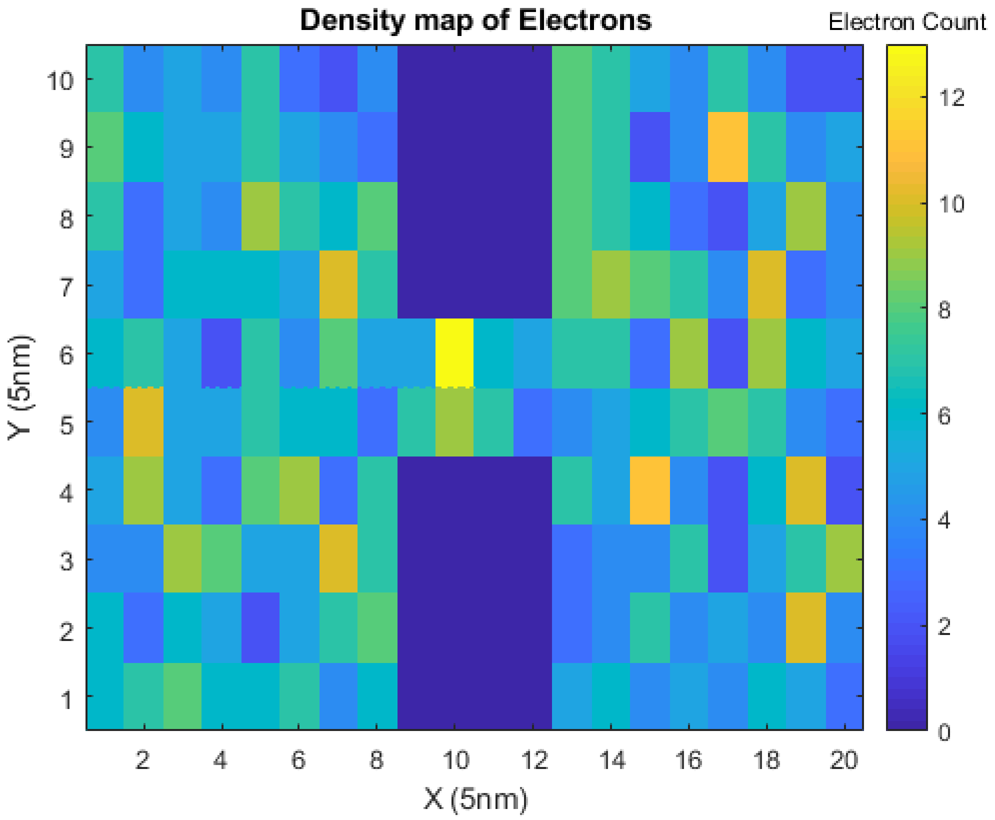
```
                %if specular, simply update positions
                x(j,1) = x(j,2) + Vx(j)*dt;
                y(j,1) = y(j,2) + Vy(j)*dt;
            else
                %if diffusive, update velocities with random value
             Vx(j) = Vth * randn;
             Vy(j) = Vth * randn;
             %assume random velocity will direct it inside the box
             while leaking == true
                 %check if new velocity is directed towards the box by
                 %examining position point and new velocity
                 if(x(j,2) < 80e-9 && Vx(j) >= 0) || ...
                         (x(j,2) > 120e-9 && Vx(j) <= 0) || ...
                         (y(j,2) < 60e-9 && Vy(j) >= 0)
                     %if new velocity conflicts with boxes, reinitialize
                     %the velocity
                     Vx(j) = Vth * randn;
                     Vy(j) = Vth * randn;
                 else
                     leaking = false;
                 end
             end
             %update positions with random velocity
             x(j,1) = x(j,2) + Vx(j)*dt;
             y(j,1) = y(j,2) + Vy(j)*dt;
            end
        end
         %checking if collision with bottom box
        if (x(j,1) >= 80e-9 && x(j,1) <= 120e-9) && y(j,1) <= 40e-9
             %checking which side of box collision came from
            %if top side, reflect Y velocity and reset positions outside
            %of box
                if y(j,2) > 40e-9
                    Vy(j) = -Vy(j);
                    y(j,1) = 40e-9;
                    y(j,2) = 40e-9;
                   %if left side, reflect X velocity and reset positions outside
            %of box
                elseif x(j,2) < 80e-9
                    Vx(j) = -Vx(j);
                    x(j,1) = 80e-9;
                    x(j,2) = 80e-9;
                     %if right side, reflect X velocity and reset position
                    %outside of box
                elseif x(j,2) > 120e-9
                    Vx(j) = -Vx(j);
                    x(j,1) = 120e-9;
                    x(j,2) = 120e-9;
                end
                 %check if specular or diffusive collisions
            if Specular == true
                %if specular, simply update positions
                x(j,1) = x(j,2) + Vx(j)*dt;
                y(j,1) = y(j,2) + Vy(j)*dt;
            else
                 %if diffusive, update velocities with random value
             Vx(j) = Vth * randn;
             Vy(j) = Vth * randn;
              %assume random velocity will direct it inside the box
             while leaking == true
                 %check if new velocity is directed towards the box by
                 %examining position point and new velocity
                 if(x(j,2) < 80e-9 && Vx(j) >= 0) || ...
                         (x(j,2) > 120e-9 && Vx(j) <= 0) || ...
                         (y(j,2) > 40e-9 && Vy(j) <= 0)
                      %if new velocity conflicts with boxes, reinitialize
                     %the velocity
                     Vx(j) = Vth * randn;
                     Vy(j) = Vth * randn;
                 else
                     leaking = false;
                 end
             end
              %update positions with random velocity
             x(j,1) = x(j,2) + Vx(j)*dt;
             y(j,1) = y(j,2) + Vy(j)*dt;
            end
        end
          %check right wall border
        if x(j,1) > L
            x(j,2) = 0;
            x(j,1) = dt * Vx(j);
        end
        %check left wall border
        if x(j,1) < 0
            x(j,2) = L;
            x(j,1) = x(j,2) + (dt * Vx(j));
        end
         %check roof/floor border
        if y(j,1) > W || y(j,1) < 0
            Vy(j) = -Vy(j);
        end
         %set line vectores for x and y positions
        XPlot = [x(j,2) x(j,1)];
        YPlot = [y(j,2) y(j,1)];
         %plot visible x and y line vectors
        if j < VisibleEcount
        plot(XPlot,YPlot);
        end
```

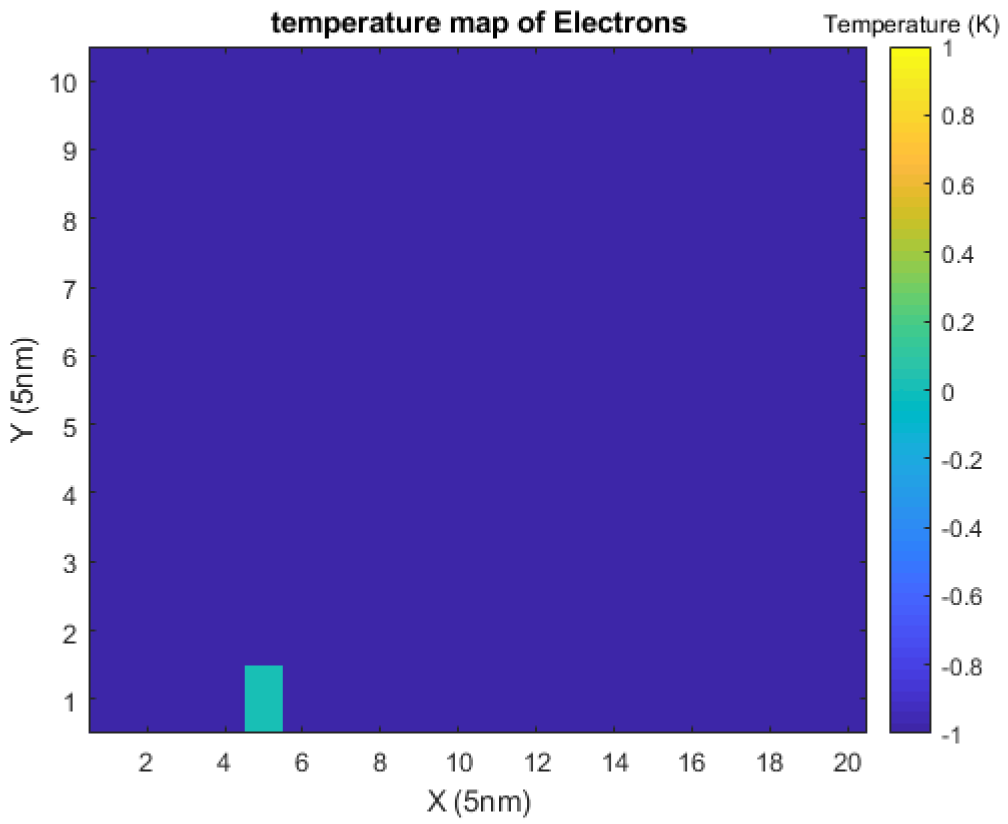
```
         %sum of themal velocities and temperatures
        VTotal = sqrt(Vx(j)^2 + Vy(j)^2);
        Temperature(1,2) = Temperature(1,2) + (mn*Vtotal(j)^2)/(2*C.k);

    end
     %plotting average temperature and ressetting values for next timestep
    AvgTemperature = Temperature(1,2)/Ecount;
    TemperaturePlot = [Temperature(1,1) AvgTemperature];
    TimePlot = [(Time - dt) Time];
    subplot(2,1,2);
    plot(TimePlot, TemperaturePlot);
    Temperature(1,1) = AvgTemperature;
    AvgTemperature = 0;
    Temperature(1,2) = 0;
    pause(1e-19)
    Time = Time + dt;
end
%looping through X indexes of the Mappingstep grid
for i = 1:(L/MappingStep)
    %looping through Y indexes of the Mapping grid
    for j = 1:(W/MappingStep)
        %looping through electrons
        for m = 1:Ecount
            %Checks to see if the electons position is with the map grip
            %being indexed. does this by checking all 4 sides of the
            %square grid index
            if(x(m,1) > MappingStep*(i -1)) && ...
                    (x(m,1) < MappingStep*(i)) && ...
                    (y(m,1) > MappingStep*(j - 1)) && ...
                    (y(m,1) < MappingStep*(j))
                %sum of x and y velocities
                Vtotal(m) = sqrt(Vx(m)^2 + Vy(m)^2);
                %counts increased on the indexed location of both Maps
                DensityMap(j, i) = DensityMap(j, i) + 1;
                TemperatureMap(j, i) = TemperatureMap(j,i) + ...
                    (mn*Vtotal(m)^2)/(2*C.k);
            end
            %Averages the temperature values given in each grid index
            TemperatureMap(j,i) = TemperatureMap(j,i)/DensityMap(j,i);
        end
    end
end


figure(2)
imagesc(DensityMap)
title('Density map of Electrons')
xlabel('X (5nm)');
ylabel('Y (5nm)');
set(gca, 'Ydir', 'Normal')
```

```
c = colorbar;
title(c, 'Electron Count')

figure(3)
imagesc(TemperatureMap)
title('temperature map of Electrons');
xlabel('X (5nm)')
ylabel('Y (5nm)')
set(gca, 'Ydir', 'Normal')
c = colorbar;
title(c, 'Temperature (K)')
```


Monte-Carlo model of electron movement in Silicon
×10^-7
1
Y 0.5
0
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2
X
×10^-7


Semiconductor Temperature
500
400
300
200
100
0
Temperature (K)
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Time (s)
×10^-12


Density map of Electrons
Electron Count
Y (5nm)
10
9
8
7
6
5
4
3
2
1
2 4 6 8 10 12 14 16 18 20
X (5nm)
12
10
8
6
4
2
0


temperature map of Electrons
Temperature (K)
Y (5nm)
X (5nm)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1