Assignment 1

Table of Contents

Question 1:

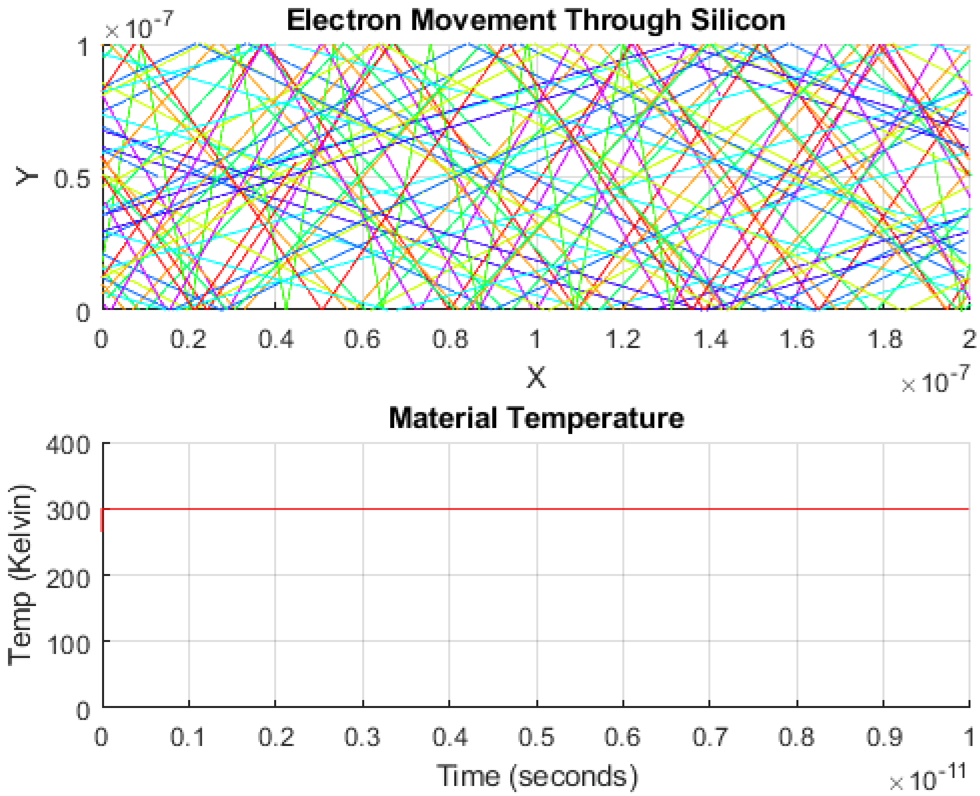
Adam Langevin, 100935879 vth is the velocity due to the tempurature of the material under test. The equation is: $vth = sqrt(2 * kb * T/m)$ for a tempurature of 300K the vth was calculated to be $1.87 * 10^5$. The mean free path of an electron was found to be $2.36 * 10^-17$. Figure 1a shows the 2-D plot of the material, and the average tempurature is shown in figure 1b.

Question 1 Code:

```
clearvars
clearvars -Global
close all
format shorte

global C
global Vx Vy x y xp yp
global numElect MarkerSize
global Mass T SavePics

numElect = 10000;
SavePics = 1;
numVisable = 10;         %This sets the amount of visable electrons
numSteps = 1000;

len = 200e-9;
wid = 100e-9;

C.Mo = 9.10938215e-31;      % electron mass
C.kb = 1.3806504e-23;       % Blotzmann Const

T = 300;
Mass = 0.26*C.Mo;
k = 1.381 * 10 ^-23;
vth = sqrt(2*(C.kb*T)/(Mass));
dt = 10e-15;
TStop = numSteps*dt;
Limits = [0 len 0 wid];
MarkerSize = 1;
```

```
%initialize the position of each electron
%inside the material.
for i = 1:numElect
    x(i) = rand()*len;
    y(i) = rand()*wid;
end

%previous values will be used to track
%the trajectories of the electrons

xp = zeros(numElect);
yp = zeros(numElect);

%initial velocities
Vx(1:numElect) = vth * cos(2*pi*randn(1,numElect));
Vy(1:numElect) = vth * sin(2*pi*randn(1,numElect));

Vt = sqrt(Vx.*Vx + Vy.*Vy);
tempSum = 0;

t = 0;

%initialize the electron position plot
figure(1);
subplot(2,1,1);
axis(Limits);
title('Electron Movement Through Silicon');
xlabel('X');
ylabel('Y');
hold on;
grid on;

%initialize the material temperuature plot
subplot(2,1,2);
axis([0 TStop 0 400]);
title('Material Temperature');
xlabel('Time (seconds)');
ylabel('Temp (Kelvin)');
hold on;
grid on;

for i = 1:numElect          %Find the initial temp of the material
    tempSum  = tempSum + (Mass*Vt(i)^2)/(2*C.kb);
end
avgTemp = tempSum/numElect;
Temp = [300 avgTemp];
Time = [0 t];
plot(t, avgTemp, '-');

colorVec = hsv(numVisable);      %Random color assignments
tempSum = 0;                     %Reseting some values to zero
avgTemp = 0;                     %to ensure proper calculations
Vt = 0;
```

```
prevTemp = 0;

while t < TStop                  %Loop to calcualte pos, and temp
    xp = x;
    yp = y;

    x(1:numElect) = x(1:numElect) + (dt .* Vx(1:numElect));
    y(1:numElect) = y(1:numElect) + (dt .* Vy(1:numElect));

    for i=1:numElect  %Loop to calcuate the boundaries, left and
                      %right are periodic, the top and bottom
                      %are reflections
       if x(i) >= len
           xp(i) = 0;
           x(i) = dt * Vx(i);
       end
       if x(i) <= 0
           xp(i) = xp(i) + len;
           x(i) = xp(i) + dt*Vx(i);
       end
       if y(i) >= wid || y(i) <= 0
           Vy(i) = - Vy(i);
       end

       Vt = sqrt(Vx(i)^2 + Vy(i)^2);       %As we loop to check bounds
       tempSum = tempSum + (Mass*Vt^2)/(2*C.kb);%we might aswell do
                                           %the temp cacluations

       X = [xp(i) x(i)];
       Y = [yp(i) y(i)];   %reduce this to the inside of the plot
       if i < numVisable
           subplot(2,1,1);
           plot(X,Y,'color',colorVec(i,:));
       end
    end

    avgTemp = tempSum/numElect;%evaluate the avg temp of the system
    Temp = [prevTemp avgTemp]; %takes two points to make a line
    Time = [(t-dt) t];         %the previous temp and the previous
    subplot(2,1,2);            %time should line up, so t-dt is the
                               %previous temp
    plot(Time, Temp, '-', 'color', colorVec(1,:));

    prevTemp = avgTemp;
    avgTemp = 0;
    tempSum = 0;
    %pause(0.00001);
    t = t + dt;
end
```



Question 2:

Figure 2 shows the initial velocities, figure 3a shows the 2-D plot of particle trajectory, and 3b shows the average temperature over time. the calculation of the mean free path was made with the equation $lambda = V/sqrt(2)Npid^2$ The actual time between collisions was calculated by summing all the time between collisions and the number of collisions. It is output in the Matlab console.

Question 2 Code:

```
clearvars
clearvars -Global
close all
format shorte

numElect = 10000;
SavePics = 1; %used at the end to save graphs on a 1, and not on a 0
numSteps = 1000;

len = 200e-9;
wid = 100e-9;

wallWidth = 1.2e-7;
wallX = .8e-7;
wallH1 = .4e-7;
```

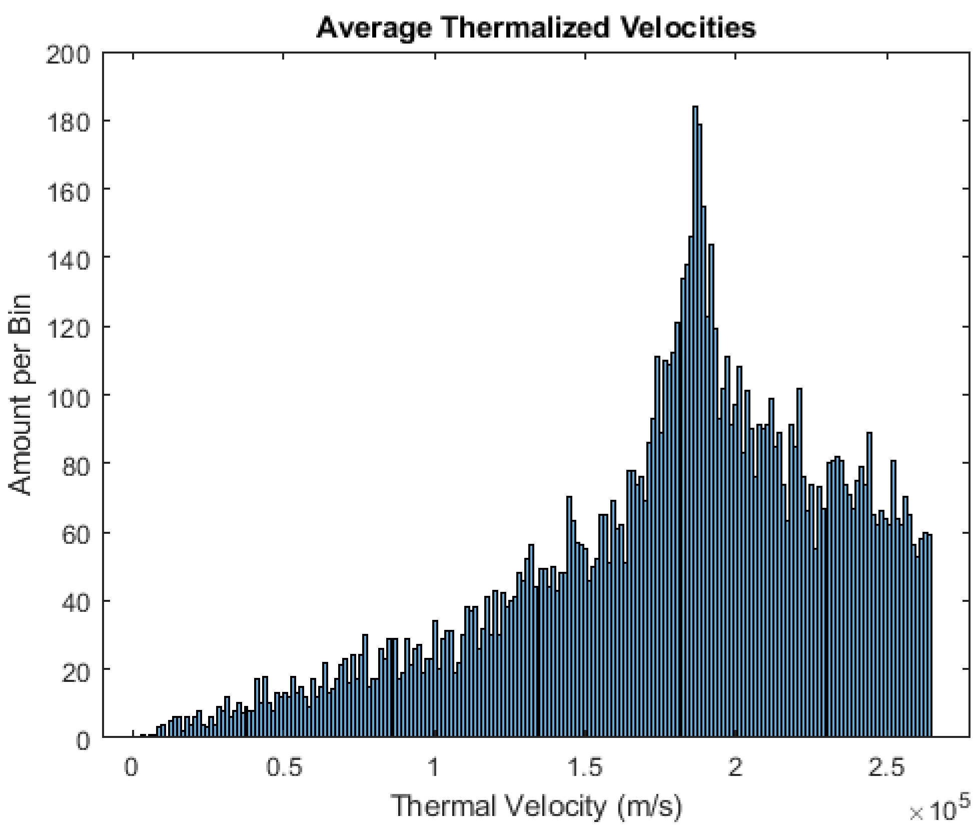
```
wallH2 = .6e-7;

C.Mo = 9.10938215e-31;      % electron mass
C.kb = 1.3806504e-23;       % Blotzmann Const

T = 300;
Mass = 0.26*C.Mo;
k = 1.381 * 10 ^-23;
vth = sqrt(2*(C.kb*T)/(Mass));   %vth = 1.8702e5
dt = 10e-15;                     %10fs
TStop = numSteps*dt;
%probbility to interact with the backgorund
Prob = 1 - exp(-dt/.2e-12);

Limits = [0 len 0 wid];%the drawing limits of the material simulated
MarkerSize = 1;

for i = 1:numElect  %initialize the position of each electron
                         %inside the material.
    x(i) = rand()*len;
    y(i) = rand()*wid;
end

xi = x;
yi = y;

%averaging the distance to each neibouring
%electron to calculate mean free path
avgDistX(1:numElect) = x.*x(1:numElect);
avgDistY(1:numElect) = y.*y(1:numElect);

Collisions = zeros(1,numElect);

xp = zeros(numElect);   %previous values will be used to track
yp = zeros(numElect);   %the trajectories of the electrons

Vx = vth .* cos(2*pi*randn(1,numElect)); %initial velocities
Vy = vth .* sin(2*pi*randn(1,numElect));

Vxi = Vx;
Vyi = Vy;

Vt = sqrt(Vx.*Vx + Vy.*Vy);
avgVel = sum(Vt)/numElect;

%histogram
figure(3);
histogram(Vt,200);
title('Average Thermalized Velocities');
xlabel('Thermal Velocity (m/s)');
ylabel('Amount per Bin');

fprintf('The Avg velocity is: %e; Calculated Thermal Velocity: %e
 \n'...
```

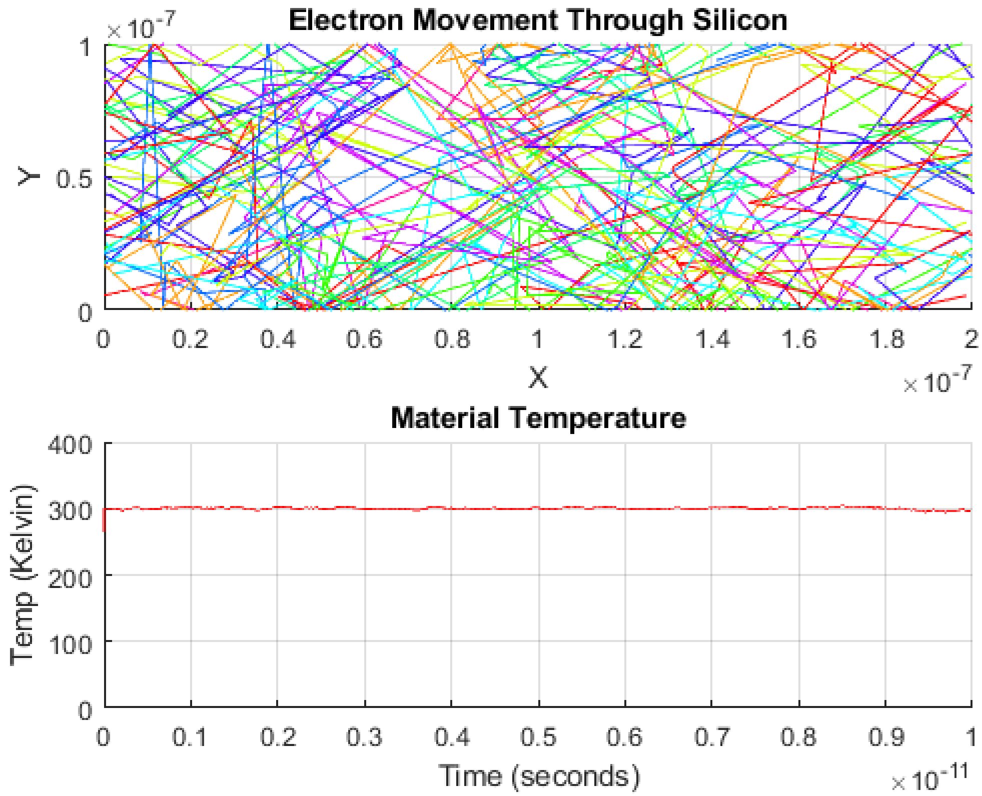
```
    , avgVel, vth);

tempSum = 0;
t = 0;

%initialize the electron position plot
figure(2);
subplot(2,1,1);
axis(Limits);
title('Electron Movement Through Silicon');
xlabel('X');
ylabel('Y');
hold on;
grid on;

%initialize the material temperuature plot
figure(2);
subplot(2,1,2);
axis([0 TStop 0 400]);
title('Material Temperature');
xlabel('Time (seconds)');
ylabel('Temp (Kelvin)');
hold on;
grid on;

%Find the initial temp of the material
for i = 1:numElect
    tempSum  = tempSum + (Mass*Vt(i)^2)/(2*C.kb);
end

avgTemp = tempSum/numElect;
Temp = [300 avgTemp];
Time = [0 t];
plot(t, avgTemp, '-');

numVisable = 10;      %This sets the amount of visable electrons
colorVec = hsv(numVisable);
tempSum = 0;          %Reseting some values to zero to ensure
avgTemp = 0;          %proper calculations
Vt = 0;
prevTemp = 0;

sumCollision = 0;     %initializing some helpers to calculate
sumCollTime = 0;      %the average collision time
numColl = 0;

while t < TStop       %Loop to calcualte pos, and temp
    xp = x;
    yp = y;

    %update position before the bounds check
    %the bounds will rewrite this if an electron
    %is outside the bounds
    x(1:numElect) = x(1:numElect) + (dt .* Vx(1:numElect));
```

```
y(1:numElect) = y(1:numElect) + (dt .* Vy(1:numElect));

for i=1:numElect%Loop to calcuate the boundaries, left and
                %right are periodic, the top and bottom
                %are reflections

   %Boundary conditions, not rethermalized
   if x(i) >= len
       xp(i) = 0;
       x(i) = dt * Vx(i);
   end
   if x(i) <= 0
       xp(i) =  xp(i) + len;
       x(i) =  xp(i) + dt*Vx(i);
   end
   if y(i) >= wid || y(i) <= 0
       Vy(i) = - Vy(i);
   end

   %implement scattering here,
   %the velocity is re-thermalized
   if rand() < Prob
      Vx(i) = vth * cos(2*pi*randn());
      Vy(i) = vth * sin(2*pi*randn());

      %take the time of the
      %last walk, reset the time between
      %collisions, count the number of collisions
      sumCollTime = sumCollTime + Collisions(i);
      Collisions(i) = 0;
      numColl = numColl + 1;
   end
   %sum the time between collions per electron
   Collisions(i) = Collisions(i) + dt;

   Vt = sqrt(Vx(i)^2 + Vy(i)^2);   %As we loop to check bounds
   tempSum = tempSum + (Mass*Vt^2)/(2*C.kb);%we might aswell do
                                            %the temp cacluations

   if i <= numVisable       %plot the difference in position,
       figure(2);           %but only a small number will show
       subplot(2,1,1);
       plot([xp(i) x(i)], [yp(i) y(i)],'color',colorVec(i,:));
   end
end

avgTemp = tempSum/numElect;%evaluate the avg temp of the system
Temp = [prevTemp avgTemp]; %takes two points to make a line
Time = [(t-dt) t];         %the previous temp and the previous
figure(2);                 %time should line up, so t-dt is the
subplot(2,1,2);            %previous temp
plot(Time, Temp, '-', 'color', colorVec(1,:));

prevTemp = avgTemp;       %used to calculate the material temp
```

```
    avgTemp = 0;
    tempSum = 0;
    %pause(0.00001);
    t = t + dt;
    hold on;
end

%mean free path calculation
avgx = sum(Vxi - Vx);
avgy = sum(Vyi - Vy);
AvgDist = sum(sqrt(avgDistX.^2 + avgDistY.^2))/numElect;
avgTot = sqrt(avgx^2 + avgy^2)/sqrt(2)*pi*numElect*(AvgDist)^2;

%mean time between collisions
avgCollTime = sumCollTime / numColl;

fprintf('Mean Free Path Calcuated: %g Avg time between collisions: %g
\n'...
    , avgTot, avgCollTime);
```

The Avg velocity is: 1.789890e+05; Calculated Thermal Velocity: 1.870193e+05
Mean Free Path Calcuated: 1.78835e-16 Avg time between collisions: 2.00656e-13





Question 3:

Figure 4a shows the 2-D trajectory plot with the interactions between the boundaries, and the "boxes" of unlike material, figure 4b is the average temperature over time. The random initial velocities are shown be the histogram in figure 5. The electron density is show as a surface plot in figure 6, and the final tempuratures of the electrons are shown in figure 7.

Question 3 Code:

```
clearvars
clearvars -Global
close all
format shorte

numElect = 10000;
numSteps = 1000;
SavePics = 1; %used at the end to save graphs on a 1, and not on a 0
numVisable = 10;%This sets the amount of visable electrons

len = 200e-9;
wid = 100e-9;

xlim = 0.01 * len;     %For density calculations at end
ylim = 0.01 * wid;     %0.01 is shows almost 1 per division
```

```
wallWidth = 1.2e-7;
wallX = .8e-7;
wallH1 = .4e-7;
wallH2 = .6e-7;

C.Mo = 9.10938215e-31; %electron mass
C.kb = 1.3806504e-23;  %Blotzmann Const

T = 300;
Mass = 0.26*C.Mo;
k = 1.381 * 10 ^-23;
vth = sqrt(2*(C.kb*T)/(Mass));  %vth = 1.8702e5
dt = 10e-15;                    %10fs
TStop = numSteps*dt;

%probbility to interact with the backgorund
Prob = 1 - exp(-dt/.2e-12);
fprintf("The proability to scatter is %g \n", Prob);

Limits = [0 (len)  0 (wid)];
MarkerSize = 1;

%initialize the position of each electron
%inside the material
for i = 1:numElect

    x(i) = rand()*len;
    y(i) = rand()*wid;

    if x(i) >= wallX && x(i) <= wallWidth && y(i) >= wallH2
        x(i) = x(i) + wallWidth + 1e-7;
    end
    if x(i) >= wallX && x(i) <= wallWidth && y(i) <= wallH1
        x(i) = x(i) + wallWidth + 1e-7;
    end
end

xi = x;
yi = y;

%averaging the distance to each neibouring
%electron to calculate mean free path
avgDistX(1:numElect) = x.*x(1:numElect);
avgDistY(1:numElect) = y.*y(1:numElect);

Collisions = zeros(1,numElect);

xp = zeros(numElect);
yp = zeros(numElect);

%initial velocities
Vx = vth .* cos(2*pi*randn(1,numElect));
Vy = vth .* sin(2*pi*randn(1,numElect));
```

```
Vxi = Vx;
Vyi = Vy;

Vt = sqrt(Vx.*Vx + Vy.*Vy);
avgVel = sum(Vt)/numElect;

%histogram
figure(5);
histogram(Vt,200);
title('Average Thermalized Velocities');
xlabel('Thermal Velocity (m/s)');
ylabel('Amount per Bin');

fprintf('The Avg velocity is: %e; Calculated Thermal Velocity: %e
 \n'...
    , avgVel, vth);

tempSum = 0;
t = 0;

%initialize the electron position plot
figure(4);
subplot(2,1,1);
axis(Limits);
title('Electron Movement Through Silicon');
xlabel('X');
ylabel('Y');
hold on;
grid on;

%initialize the material temperuature plot
figure(4);
subplot(2,1,2);
axis([0 TStop 0 400]);
title('Average Material Temperature');
xlabel('Time (seconds)');
ylabel('Temp (Kelvin)');
hold on;
grid on;

for i = 1:numElect %Find the initial temp of the material
    tempSum  = tempSum + (Mass*Vt(i)^2)/(2*C.kb);
end

avgTemp = tempSum/numElect;
Temp = [300 avgTemp];
Time = [0 t];
plot(t, avgTemp, '-');

colorVec = hsv(numVisable);
tempSum = 0;       %Reseting some values to zero to ensure
avgTemp = 0;       %proper calculations
Vt = 0;
prevTemp = 0;
```

```
sumCollision = 0; %initializing some helpers to calculate
sumCollTime = 0;  %the average collision time
numColl = 0;

subplot(2,1,1);
rectangle('Position', [wallX 0 4e-8 4e-8]);
rectangle('Position', [wallX wallH2 4e-8 4e-8]);

while t < TStop   %Loop to calcualte pos, and temp
    xp = x;
    yp = y;

    %update position before the bounds check
    %the bounds will rewrite this if an electron
    %is outside the bounds
    x(1:numElect) = x(1:numElect) + (dt .* Vx(1:numElect));
    y(1:numElect) = y(1:numElect) + (dt .* Vy(1:numElect));

    %Loop to calcuate the boundaries, left and
    %right are periodic, the top and bottom
    %are reflections
    for i=1:numElect
       %Boundary conditions, not rethermalized
       %right side boundry
       if x(i) >= len
           if rand() >= Prob
               xp(i) = 0;
               x(i) = dt * vth * cos(2*pi*randn());
               Vx(i) = vth * cos(2*pi*randn());
               if x(i) <=0 || x(i) >= len || x(i)+dt*Vx(i) >= len
 || ...
                       x(i)+dt*Vx(i) <= 0
                    Vx(i) = -Vx(i);
                    x(i) = xp(i) +dt*Vx(i);
               end
           else
               xp(i) = 0;
               x(i) = dt * Vx(i);
           end
       end

       %left side boundry
       if x(i) <= 0
           if rand() >= Prob
               xp(i) = x(i) + len;
               x(i) = xp(i) + dt * vth * cos(2*pi*randn());
               Vx(i) = vth*cos(2*pi*randn());
               if x(i) <= 0 || x(i) >= len || x(i)+dt*Vx(i) >= len
 || ...
                       x(i)+dt*Vx(i) <= 0
                   Vx(i) = -Vx(i);
                   x(i) = xp(i) + dt*Vx(i);
               end
```

```
            else
                xp(i) =  xp(i) + len;
                x(i) =  xp(i) + dt*Vx(i);
            end
        end

        %Upper and lower boundries
        if y(i) >= wid || y(i) <= 0
            if rand() >= Prob
                Vy(i) = - vth * sin(2*pi*randn());
                y(i) = yp(i);
                yp(i) = y(i) - dt*Vy(i);
                if  y(i) >= wid || y(i) <= 0 || y(i) +dt*Vy(i) >= wid
|| ...
                        y(i) +dt*Vy(i) <= 0
                    Vy(i) = -Vy(i);
                    y(i) = yp(i);
                    yp(i) = y(i) - dt*Vy(i);
                end
            else
                Vy(i) = -Vy(i);
                y(i) = yp(i);
                yp(i) = y(i) - dt*Vy(i);
            end
        end

        %left side of the boxes
        if ((y(i) <= 4e-8 || y(i) >= 6e-8) && x(i)+dt*Vx(i) >= 8e-8
&& ...
                x(i) <= 8e-8)
            if rand() >= Prob
                xt = x(i);
                Vx(i) = - vth * cos(2*pi*randn());
                x(i) = xp(i) + dt*Vx(i);
                xp(i) = xt;
                if x(i) + dt*Vx(i) >= 8e-8
                    xt = x(i);
                    Vx(i) = -Vx(i);
                    x(i) = xp(i) + dt*Vx(i);
                    xp(i) = xt;
                end
            else
                 Vx(i) = -Vx(i);
            end
        end

        %right side of the boxes
        if ((y(i) <= 4e-8 || y(i) >= 6e-8) && x(i)+dt*Vx(i) <= 12e-8
&& ...
                x(i) >= 12e-8)
            if rand() >= Prob
                xt = x(i);
                Vx(i) = - vth * cos(2*pi*randn());
                x(i) = xp(i) + dt*Vx(i);
```

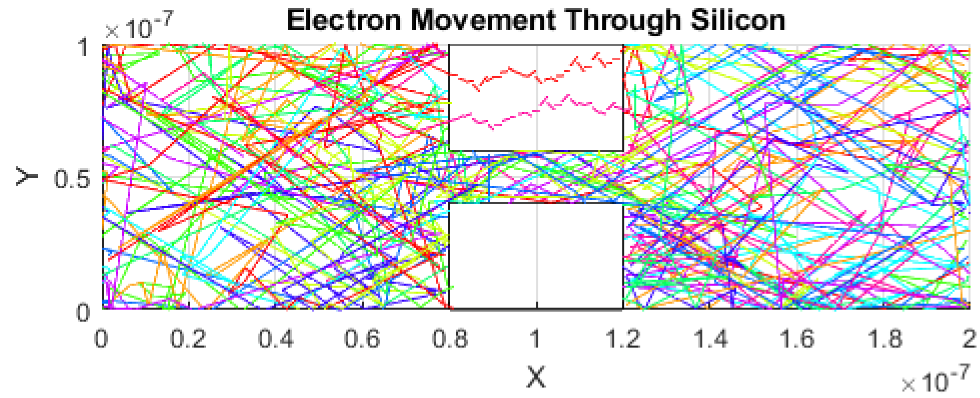
```
                xp(i) = xt;
                if x(i) + dt*Vx(i) <= 12e-8
                    xt = x(i);
                    Vx(i) = -Vx(i);
                    x(i) = xp(i) + dt*Vx(i);
                    xp(i) = xt;
                end
            else
                Vx(i) = -Vx(i);
            end
        end

        %inbetween the two boxes
        if ((y(i)+dt*Vy(i) >= 6e-8 || y(i)+dt*Vy(i) <= 4e-8) && x(i)
<= ...
                12e-8 && x(i) >= 8e-8)
            if rand() >= Prob
                Vy(i) = vth * sin(2*pi*randn());
                if y(i) + dt*Vy(i) >= 6e-8 || y(i) + dt*Vy(i) <= 4e-8
                    yt = y(i);
                    Vy(i) = -Vy(i);
                    y(i) = yp(i) + dt*Vy(i);
                    yp(i) = yt;
                end
            else
                 Vy(i) = -Vy(i);
            end
        end

        %implement scattering here, the velocity is re-thermalized
        if rand() < Prob
           Vx(i) = vth * cos(2*pi*randn());
           Vy(i) = vth * sin(2*pi*randn());

           %take the time of the
           %last walk, reset the time between
           %collisions, and count the number of Collisions
           sumCollTime = sumCollTime + Collisions(i);
           Collisions(i) = 0;
           numColl = numColl + 1;
        end
        %sum the time between collions per electron
        Collisions(i) = Collisions(i) + dt;

        Vt = sqrt(Vx(i)^2 + Vy(i)^2);%As we loop to check bounds
        tempSum = tempSum + (Mass*Vt^2)/(2*C.kb);%we might aswell do
                                             %the temp cacluations

        if i <= numVisable          %plot the difference in position,
            figure(4);              %but only a small number will show
            subplot(2,1,1);
            plot([xp(i) x(i)], [yp(i) y(i)],'color',colorVec(i,:));
        end
    end
```

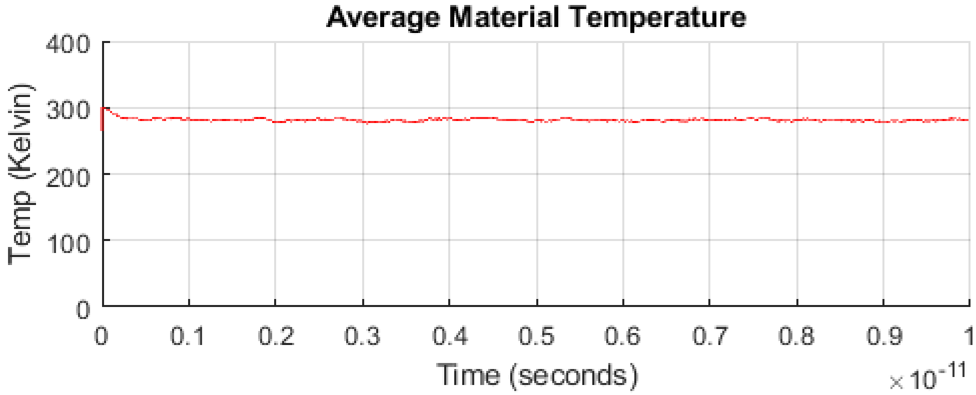
```
    avgTemp = tempSum/numElect;%evaluate the avg temp of the system
    Temp = [prevTemp avgTemp]; %takes two points to make a line
    Time = [(t-dt) t];         %the previous temp and the previous
    figure(4);                 %time should line up, so t-dt is the
    subplot(2,1,2);            %previous temp
    plot(Time, Temp, '-', 'color', colorVec(1,:));


    prevTemp = avgTemp;        %used to calculate the material temp
    avgTemp = 0;
    tempSum = 0;
    %pause(0.0000000000001);
    t = t + dt;
    hold on;
end

%electron density using 10% of total area
xbox = (len/xlim) + 1;
ybox = (wid/ylim) + 1;
c(1:xbox, 1:ybox) = zeros();
Vfx(1:xbox, 1:ybox) = zeros();
Vfy(1:xbox, 1:ybox) = zeros();
Vf(1:xbox, 1:ybox) = zeros();

for i = 1:xbox
   for j = 1:ybox
      for n = 1: numElect
          if x(n) > (i-1)*xlim && x(n) < i*xlim && y(n) > (j-1)*ylim
 && ...
                  y(n) < j*ylim
            c(i,j) = c(i,j) + 1;
            Vfx(i,j) = Vfx(i,j) + Vx(n);
            Vfy(i,j) = Vfy(i,j) + Vy(n);
          end
      end
   end
end

%color maps for the surfs
CL(:,:,1) = zeros(int32(xbox-1));
CL(:,:,2) = ones(int32(xbox-1)).*linspace(0.5,0.6,int32(xbox-1));
CL(:,:,3) = ones(int32(xbox-1)).*linspace(0,1,int32(xbox-1));

CL2 = CL;
CL2(:,:,1) = ones(int32(xbox-1)).*linspace(0.75,0.95,int32(xbox-1));

figure(6);
s1 = surf(1:xbox,1:ybox,c,'FaceAlpha',0.5);
title('Electron Density');
xlabel('X');
ylabel('Y');
zlabel('amount per division');
```

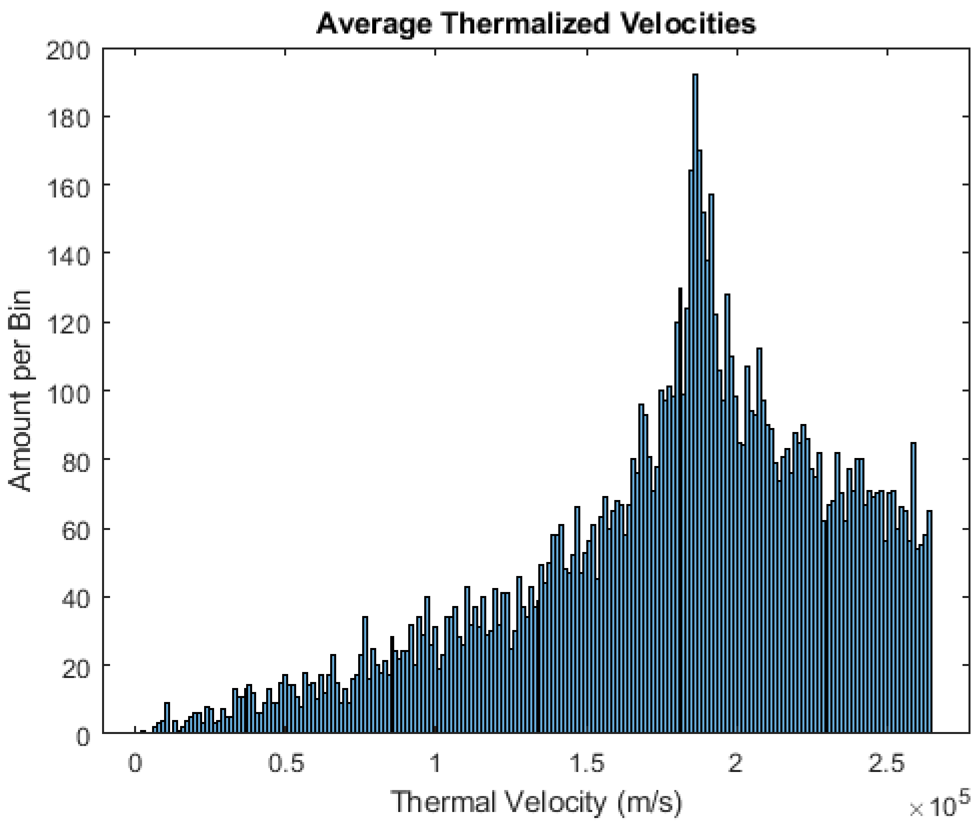
```
for i = 1:xbox
    for j = 1:ybox              %(Mass*Vt^2)/(2*C.kb)
        Vf(i,j) = (Mass/(2*C.kb))*mean(Vfx(i,j).^2 + Vfy(i,j).^2);
    end
end

figure(7);
s2 = surf(1:xbox,1:ybox,Vf,'FaceAlpha',0.5);
title('Temperature Map');
xlabel('X');
ylabel('Y');
zlabel('Tempurature per division');

%mean free path calculation
avgx = sum(Vxi - Vx);
avgy = sum(Vyi - Vy);
AvgDist = sum(sqrt(avgDistX.^2 + avgDistY.^2))/numElect;
avgTot = sqrt(avgx^2 + avgy^2)/sqrt(2)*pi*numElect*(AvgDist)^2;

%mean time between collisions
avgCollTime = sumCollTime / numColl;

fprintf('Mean Free Path Calcuated: %g Avg time between collisions: %g
\n'...
    , avgTot, avgCollTime);

%save the final state of the graphs to add to the report
if SavePics
     figure(1);
    saveas(gcf, 'ElectronsInSiliconQ1.jpg');

    figure(2);
    saveas(gcf, 'ElectronsInSiliconQ2.jpg');

    figure(3);
    saveas(gcf, 'VelocityHistQ2.jpg');

    figure(4);
    saveas(gcf, 'ElectronsInSiliconQ3.jpg');

    figure(5);
    saveas(gcf, 'VelocityHistQ3.jpg');

    figure(6);
    saveas(gcf, 'ElectronDensityQ3.jpg');

    figure(7);
    saveas(gcf, 'TempuratureMapQ4.jpg');
end
```

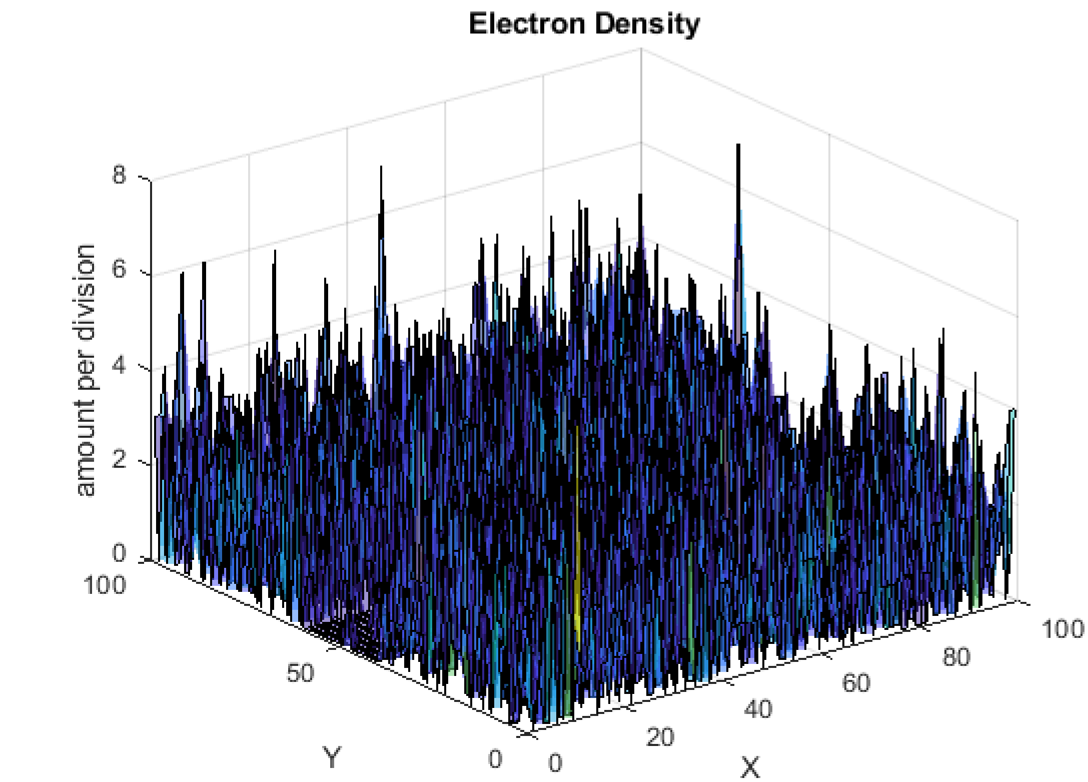
The proability to scatter is 0.0487706
The Avg velocity is: 1.792002e+05; Calculated Thermal Velocity: 1.870193e+05

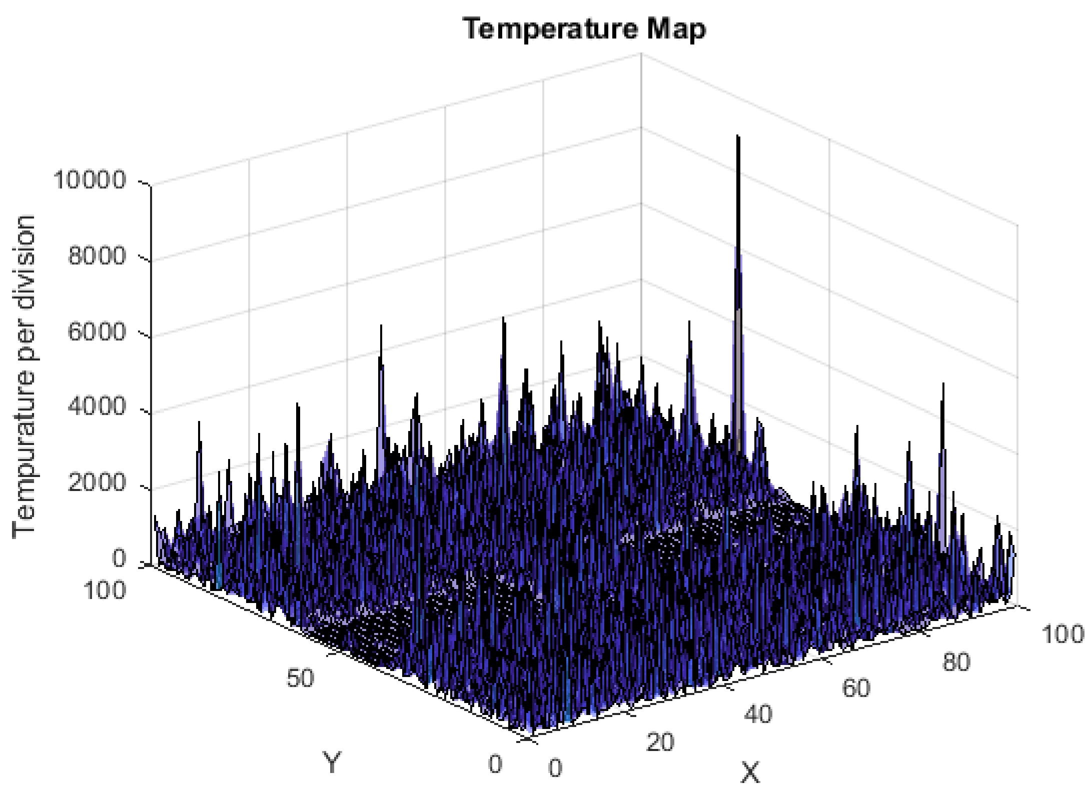
```
Warning: Integer operands are required for colon operator when used as
 index
Warning: Integer operands are required for colon operator when used as
 index
Warning: Integer operands are required for colon operator when used as
 index
Warning: Integer operands are required for colon operator when used as
 index
Warning: Integer operands are required for colon operator when used as
 index
Warning: Integer operands are required for colon operator when used as
 index
Warning: Integer operands are required for colon operator when used as
 index
Warning: Integer operands are required for colon operator when used as
 index
Mean Free Path Calcuated: 4.89445e-16 Avg time between collisions:
 2.00377e-13
```


Electron Movement Through Silicon
×10^-7
1
0.5
0
Y
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
X
×10^-7


Average Material Temperature
400
300
200
100
0
Temp (Kelvin)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Time (seconds)
×10^-11


Average Thermalized Velocities
200
180
160
140
120
100
80
60
40
20
0
Amount per Bin
0
0.5
1
1.5
2
2.5
Thermal Velocity (m/s)
×10^5


Electron Density
amount per division
8
6
4
2
0
100
50
0
Y
0
20
40
60
80
100
X


Temperature Map
Tempurature per division
10000
8000
6000
4000
2000
0
100
50
0
Y
0
20
40
60
80
100
X

Published with MATLAB® R2017b