Assignment 3

Table of Contents

Question 1:

a) The electric feild accross the x direction would be equvialent to E = V/d = 0.1V/200e-9m = 5.0000e +05V/m.

b) The Force due to the electric feild is F = q*E, where q is the change on each electron, the force being F = (1.602e-19C * 5e5V/m) = 8.01e-14N.

c) The acceleration of each electron is defined by Newton's 2nd law, F = ma then solving for a = F/m. Then using our values, a = 8.01e-14N / 2.3684e-31kg = 3.3820e+17 m/s^2.

d) The relation of current to drift velocity is by combining the average drift velocity of the electrons v=u*E, where u is the density of electrons and the electric feild, and the current densiy J=q*n*u*E, then we get the current density as J=q*n*v. Figure 3 shwos the change in current over time. The current increases over time because the feild that is placed on either ends of the x direction apply an acceleration to each electron. The proportion is not 1:1 because of scttering, when the electrons are re-thermalized the build up of velocity since the last collision is reset.

Question 1 Code:

```
global C
global Vx Vy x y xp yp xi yi Vxi Vyi Collisions
global Mass T SavePics

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

Volx = 0.1;
Voly = 0;

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
kb = 1.3806504e-23;       % Blotzmann Const
q  = 1.60217662e-19;      % charge of an electron

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

Prob = 1 - exp(-dt/.2e-12);   %probbility to interact with the
 backgorund

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

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));

Vx = -sqrt(2*kb*T/Mass + log(numElect)...
    + (1/2)*log(2*pi*kb*T/Mass)).*randn(1,numElect);
Vy = -sqrt(2*kb*T/Mass + log(numElect)...
    + (1/2)*log(2*pi*kb*T/Mass)).*randn(1,numElect);

Vxi = Vx;
Vyi = Vy;

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

%histogram
figure(2);
```

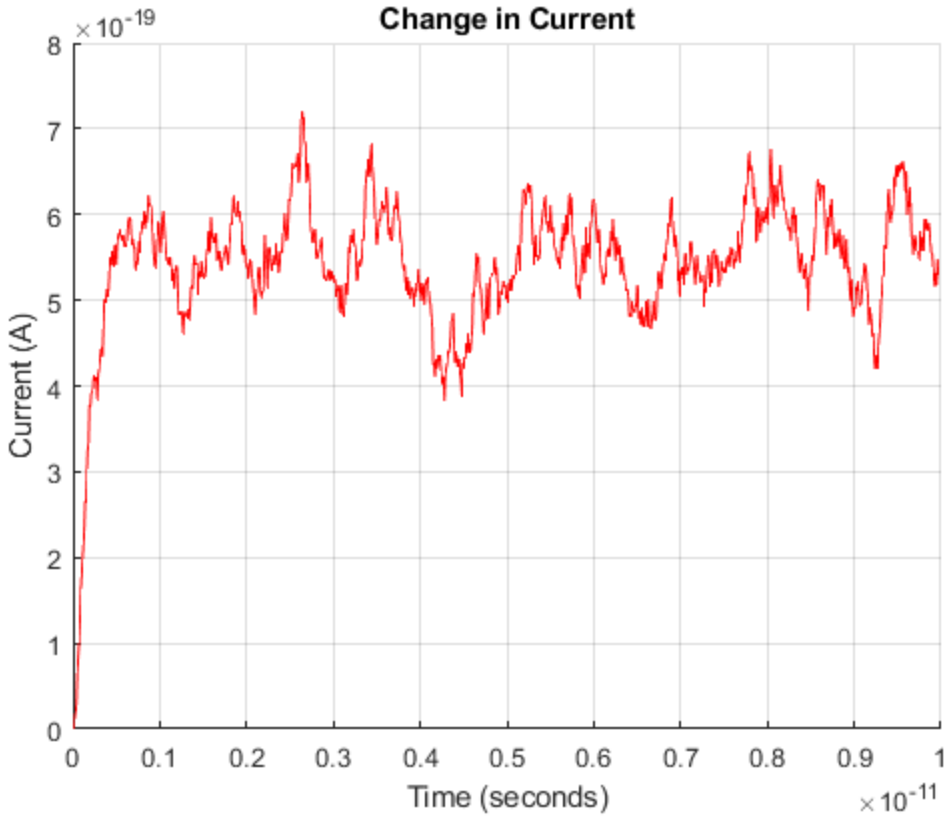
```
histogram(Vt,200);
title('Average Thermalized Velocities');
xlabel('Thermal Velocity (m/s)');
ylabel('Amount per Bin');

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

tempSum = 0;
t = 0;

%initialize the electron position plot
figure(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
figure(1);
subplot(2,1,2);
axis([0 TStop 0 500]);
title('Material Temperature');
xlabel('Time (seconds)');
ylabel('Temp (Kelvin)');
hold on;
grid on;

%current
figure(3);
title('Change in Current');
axis([0 TStop 0 8e-19]);
xlabel('Time (seconds)');
ylabel('Current (A)');
hold on;
grid on;

%Find the initial temp of the material
for i = 1:numElect
    tempSum  = tempSum + (Mass*Vt(i)^2)/(4*kb);
end

%voltageFeild2(100,100,0.1,60,60,1,1);
Ax = (q*Volx)/(Mass*len);
Ay = (q*Voly)/(Mass*wid);

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

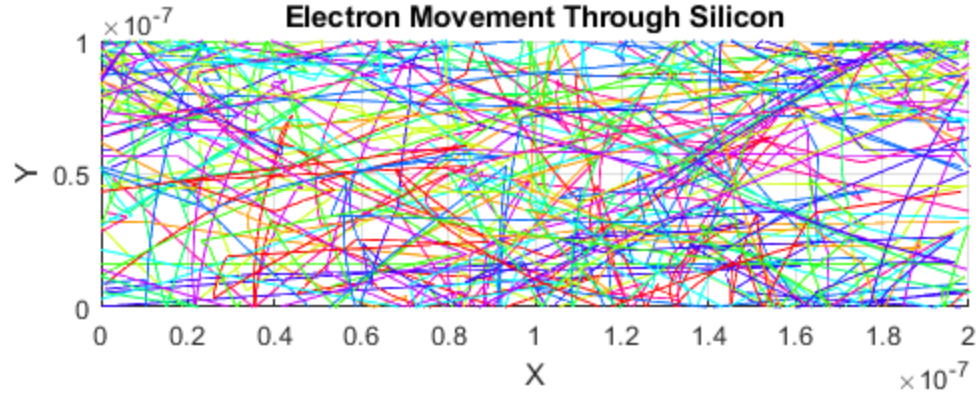
```
numVisable = 10;                %This sets the amount of visable
 electrons
colorVec = hsv(numVisable);
tempSum = 0;                    %Reseting some values to zero to
 ensure
avgTemp = 0;                    %proper calculations
Vt = 0;
prevTemp = 0;
pcurr = 0;

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(1:numElect) = x(1:numElect);
    yp(1:numElect) = y(1:numElect);

    %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


      %implement scattering here, the velocity is re-thermalized
      if rand() < Prob
         Vx(i) = -sqrt(2*kb*T/Mass + log(numElect)...
             + (1/2)*log(2*pi*kb*T/Mass)).*randn(1,1);
         Vy(i) = -sqrt(2*kb*T/Mass + log(numElect)...
             + (1/2)*log(2*pi*kb*T/Mass)).*randn(1,1);

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

      Vx(i) = Vx(i) + dt*Ax;
      Vy(i) = Vy(i) + dt*Ay;

      %Boundary conditions, not re-thermalized
      if x(i) >= len
          xp(i) = 0;
```

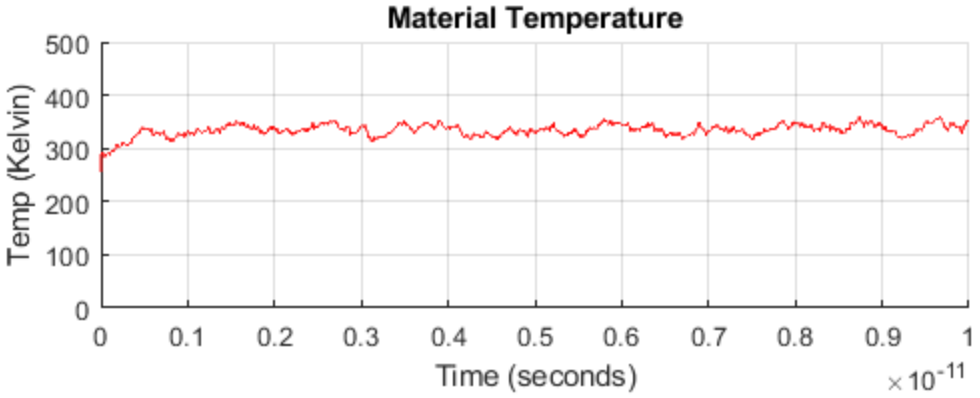
```
            x(i) = dt * Vx(i);
        end
        if x(i) <= 0
            xp(i) =  xp(i) + len;
            x(i) =  xp(i) + dt*Vx(i);
        end
        if y(i) >= wid || y(i) <= 0
            Vy(i) = - Vy(i);
        end

        %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)/(4*kb);  %we might aswell do
the
                                                 %temp cacluations

        if i <= numVisable              %plot the difference in
position,
            figure(1);                  %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(1);                          %time should line up, so t-dt is
the
    subplot(2,1,2);                     %previous temp
    plot(Time, Temp, '-', 'color', colorVec(1,:));

    cur = q*(10e-15)*sum(Vx)/len;
    figure(3);
    subplot(1,1,1);
    Curr = [pcurr cur];
    plot(Time,Curr, '-', 'color', colorVec(1,:));

    pcurr = cur;
    prevTemp = avgTemp;                 %used to calculate the material
temp
    avgTemp = 0;
    tempSum = 0;
    pause(0.00001);
    t = t + dt;
    hold on;
end
```

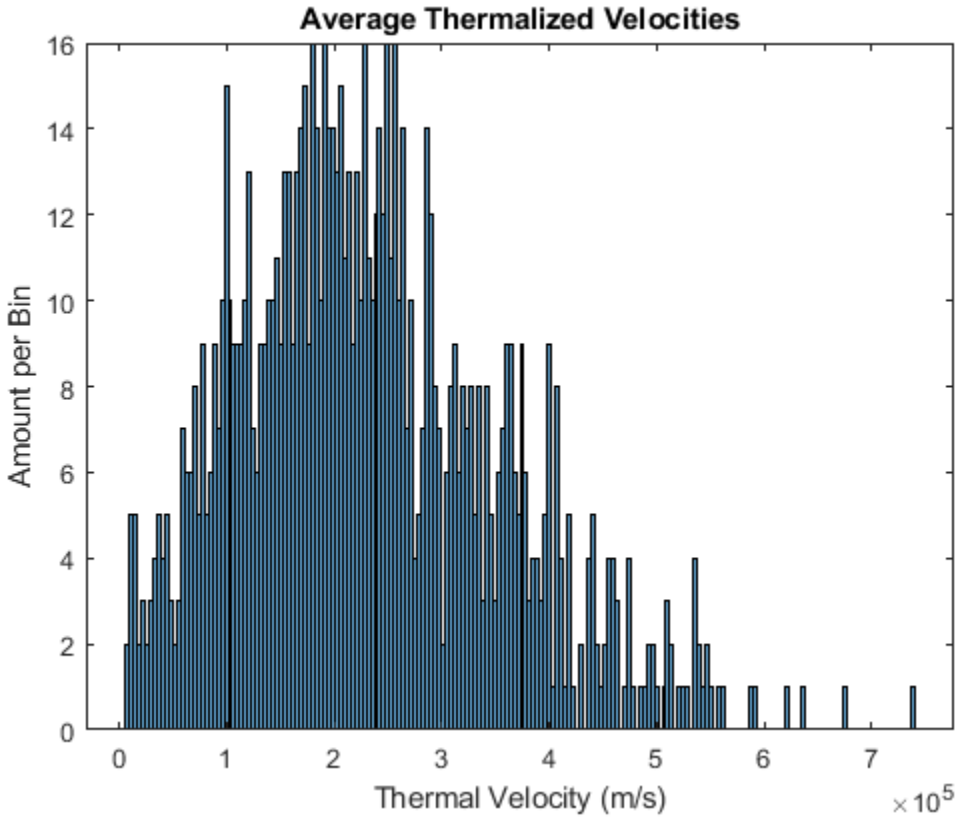
```
%mean free path calculation
avgx = sum(Vxi - Vx);
avgy = sum(Vyi - Vy);
AvgDist = sum(sqrt(avgDistX.^2 + avgDistY.^2))/numElect;
avgTot = sqrt(avgx^2 + avgy^2)/sqrt(2)*pi*numElect*(AvgDist)^2;

%mean time between collisions
avgCollTime = sumCollTime / numColl;

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

%save the final state of the graphs to add to the report
if SavePics
    figure(1);
    saveas(gcf, 'ElectronsInSiliconQ2.jpg');

    figure(2);
    saveas(gcf, 'VelocityHistQ2.jpg');
end
```

```
The Avg velocity is: 2.312901e+05; Calculated Thermal Velocity:
 1.870193e+05
Mean Free Path Calcuated: 3.76434e-17 Avg time between collisions:
 2.00869e-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


Material Temperature
500
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
16
14
12
10
8
6
4
2
0
Amount per Bin
0
1
2
3
4
5
6
7
Thermal Velocity (m/s)
×10^5

Question 2:

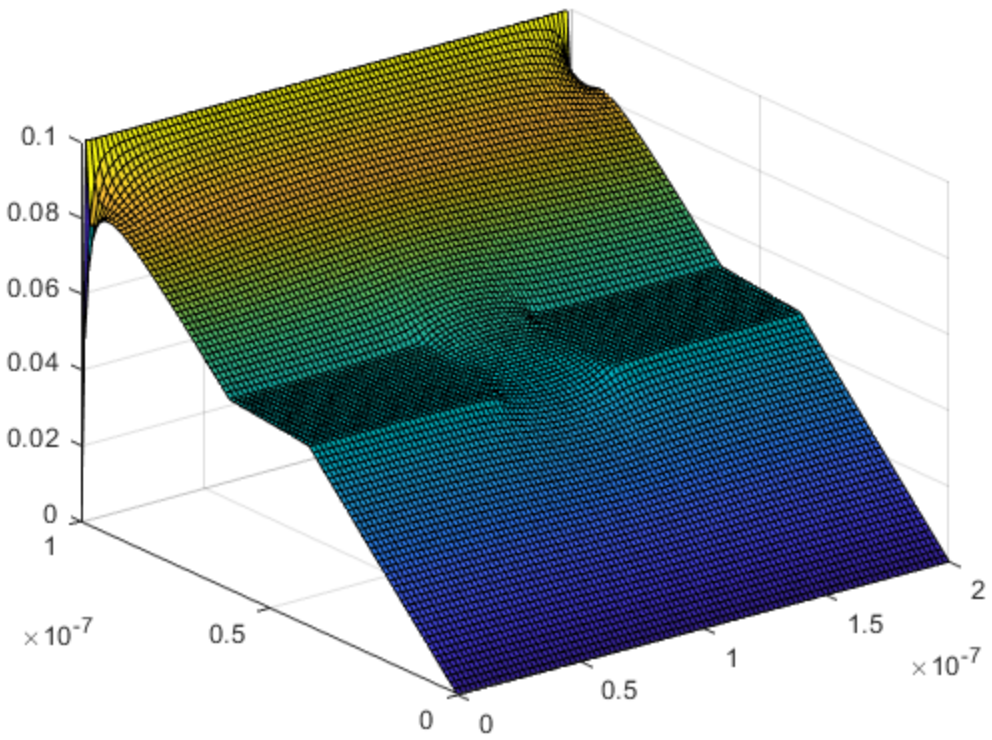
```
len = 200e-9;
wid = 100e-9;

global V2;
global xn;
global yn;
global Ex Ey Jx Jy Cm B curr;

Lx = len;
Ly = wid;
nx = 100;
ny = 100;
xn = linspace(0,Lx,nx);
yn = linspace(Ly,0,ny);

%diff = (1/width)^2;

V0 = 0.1
V2 = zeros(nx,ny);

G = sparse(nx*ny,nx*ny);
B = zeros(1,nx*ny);

apatureWidth = 20;
apatureLength = 20;

%set up the conductivity map
C1 = 0.1;
C2 = 1;
Cm = ones(nx,ny);
for i=1:nx
    for j=1:ny
        if j < (nx/2 - apatureWidth/2) && i > (ny/2 - apatureLength/2)
 && i < (ny/2 + apatureLength/2 + 1)
            Cm(i,j) = 1/C1; %inside the top box
        elseif j > (nx/2 + apatureWidth/2 + 1) && i<nx+1 && i >(ny/2 -
 apatureLength/2) && i < (ny/2 + apatureLength/2 + 1)
           Cm(i,j) = 1/C1; %inside the bottom box
        else
           Cm(i,j) = 1/C2;
        end
    end
end

%set up the BCs
for i=1:nx
    for j=1:ny
        n = i +(j-1)*nx;
        if i == 1
            B(n) = 0;
        elseif i == nx
```

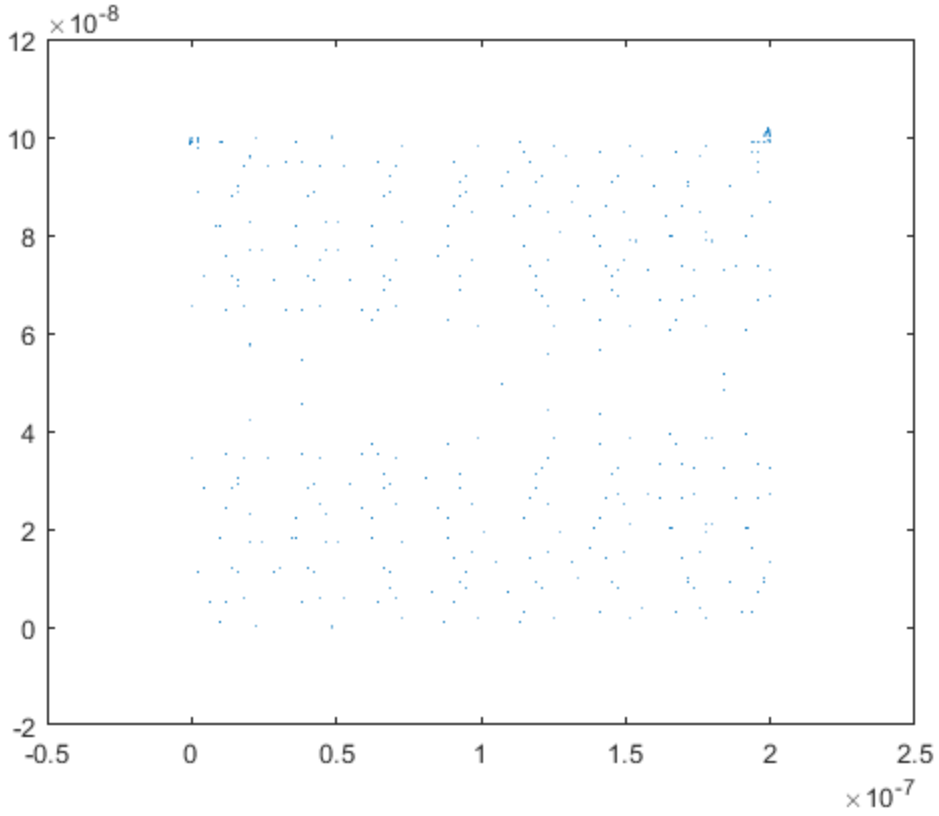
```
            B(n) = 0;
        elseif j == 1
            B(n) = V0;
        elseif j == ny
            B(n) = 0;
        end
    end
end

%G matrix
for i = 1:nx
    for j = 1:ny
       n = j + (i-1)*ny;

        if i == 1                   %left
           G(n,n) = 1;

        elseif i ==nx               %right
           G(n,n) = 1;

        elseif j == 1               %bottom
            nxm = j + (i-2)*ny;
            nxp = j + (i)*ny;
            nyp = j+1 + (i-1)*ny;

            rxm = (Cm(i,j) + Cm(i-1,j))*0.5;
            rxp = (Cm(i,j) + Cm(i+1,j))*0.5;
            ryp = (Cm(i,j) + Cm(i,j+1))*0.5;

            G(n, n) = -(rxm+rxp+ryp);
            G(n, nxm) = rxm;
            G(n, nxp) = rxp;
            G(n, nyp) = ryp;

        elseif j == ny              %top side
            nxm = j + (i-2)*ny;
            nxp = j + (i)*ny;
            nym = j-1 + (i-1)*ny;

            rxm = (Cm(i,j) + Cm(i-1,j))*0.5;
            rxp = (Cm(i,j) + Cm(i+1,j))*0.5;
            rym = (Cm(i,j) + Cm(i,j-1))*0.5;

            G(n, n) = -(rxm + rxp + rym);
            G(n, nxm) = rxm;
            G(n, nxp) = rxp;
            G(n, nym) = rym;

        else                        %interrior
            nxm = j + (i-2)*ny;
            nxp = j + (i)*ny;
            nym = j-1 + (i-1)*ny;
            nyp = j+1 + (i-1)*ny;
```

```
            rxm = (Cm(i,j) + Cm(i-1,j))*0.5;
            rxp = (Cm(i,j) + Cm(i+1,j))*0.5;
            rym = (Cm(i,j) + Cm(i,j-1))*0.5;
            ryp = (Cm(i,j) + Cm(i,j+1))*0.5;

            G(n,n) = -(rxm+rxp+rym+ryp);
            G(n,nxm) = rxm;
            G(n,nxp) = rxp;
            G(n,nym) = rym;
            G(n,nyp) = ryp;
        end
    end
end

V = G\B';

%reassign to an x,y coordinate system
for i = 1:nx
    for j = 1:ny
        n = j + (i-1)*nx;
        V2(i,j) = V(n);
    end
end

Ex = zeros(nx,ny);
Ey = zeros(nx,ny);

%calcualte the electric feilds
for i = 1:nx
 for j = 1:ny
     if i == 1
         Ex(i,j) = (V2(i+1,j) - V2(i,j));
     elseif i == nx
         Ex(i,j) = (V2(i,j) - V2(i-1,j));
     else
         Ex(i,j) = (V2(i+1,j) - V2(i-1, j))/2.0;
     end
     if j == 1
         Ey(i,j) = (V2(i,j+1) - V2(i,j));
     elseif j == ny
         Ey(i,j) = (V2(i,j) - V2(i, j-1));
     else
         Ey(i,j) = (V2(i,j+1) - V2(i,j-1))/2.0;
     end
 end
end

Ex = -Ex;
Ey = -Ey;

%Current density
Jx = Cm.*Ex;
Jy = Cm.*Ey;
```

```
SC0 = sum(Jx(1,:));
SC1 = sum(Jx(nx,:));
curr = (SC0 + SC1)*0.5;

figure(4);
title('Potential inside the material');
xlabel('x');
ylabel('y');
zlabel('Potential');
surf(xn,yn,V2);
hold on;

figure(5);
title('Electric Feild as Vectors');
xlabel('x');
ylabel('y');
quiver(xn,yn,Ex,Ey);
hold on;
```

```
V0 =

   1.0000e-01
```





Question 3:

a) Figure 6 shows teh 2-D trajectories through the material.

b) The electron density is shown in Figure 8. The electron density should show a favoring to the side of lower potential, as the electrons should move opposit to the current flow.

c) The next step

Question 3 Code:

```
numElect = 1000;
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;    %0.05 is 4 or less per division

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

```
wallH1 = .4e-7;
wallH2 = .6e-7;

C.Mo = 9.10938215e-31; %electron mass
kb = 1.3806504e-23;       % Blotzmann Const
q  = 1.60217662e-19;      % charge of an electron


T = 300;
Mass = 0.26*C.Mo;
k = 1.381 * 10 ^-23;
vth = sqrt(2*(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);

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

Vx = -sqrt(2*kb*T/Mass + log(numElect) + (1/2)*log(2*pi*kb*T/Mass))...
```

```
    .*randn(1,numElect);
Vy = -sqrt(2*kb*T/Mass + log(numElect) + (1/2)*log(2*pi*kb*T/Mass))...
    .*randn(1,numElect);

Vxi = Vx;
Vyi = Vy;

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

%histogram
figure(7);
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(6);
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(6);
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)/(4*kb);
end

voltageFeild2(100,100,wid,len,0.8,40,40,0.1,1);
Ax(1:100,1:100) = (q*Ex(1:100,1:100))/(Mass);
Ay(1:100,1:100) = (q*Ey(1:100,1:100))/(Mass);

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;

figure(6);
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(1:numElect) = x(1:numElect);
    yp(1:numElect) = y(1:numElect);

    %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
       %right side boundry
       if x(i) >= len
           if rand() >= Prob
               xp(i) = 0;
               x(i) = dt * -sqrt(2*kb*T/Mass + log(numElect)...
                           + (1/2)*log(2*pi*kb*T/Mass)).*randn(1,1);

               Vx(i) = -sqrt(2*kb*T/Mass + log(numElect)...
                           + (1/2)*log(2*pi*kb*T/Mass)).*randn(1,1);

               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 * -sqrt(2*kb*T/Mass +
log(numElect)...
                        + (1/2)*log(2*pi*kb*T/Mass)).*randn(1,1);

            Vx(i) = -sqrt(2*kb*T/Mass + log(numElect)...
                        + (1/2)*log(2*pi*kb*T/Mass)).*randn(1,1);

            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) = - -sqrt(2*kb*T/Mass + log(numElect)...
                        + (1/2)*log(2*pi*kb*T/Mass)).*randn(1,1);

            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
```

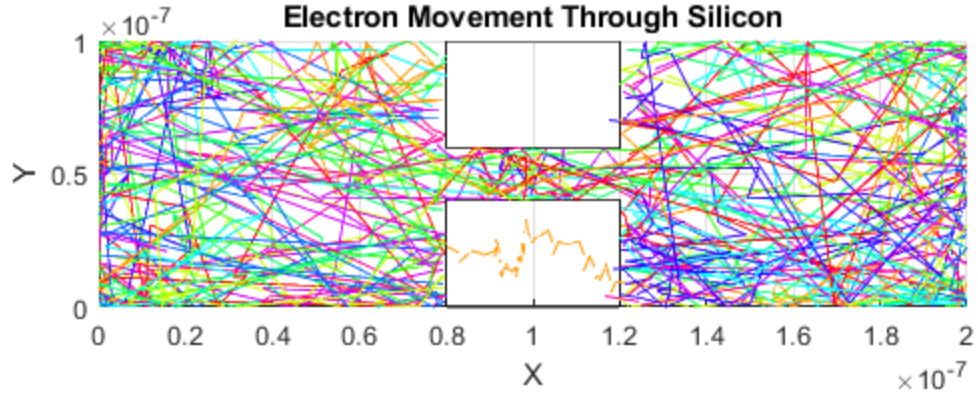
```
        %left side of the boxes
        if ((y(i) <= 4e-8 || y(i) >= 6e-8) && x(i)+dt*Vx(i) >= 8e-8
&&...
                x(i) <= 8e-8)

            if rand() >= Prob
                xt = x(i);
                Vx(i) = - -sqrt(2*kb*T/Mass + log(numElect)...
                            + (1/2)*log(2*pi*kb*T/Mass)).*randn(1,1);

                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) = - -sqrt(2*kb*T/Mass + log(numElect)...
                            + (1/2)*log(2*pi*kb*T/Mass)).*randn(1,1);

                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) = -sqrt(2*kb*T/Mass + log(numElect)...
                            + (1/2)*log(2*pi*kb*T/Mass)).*randn(1,1);
```

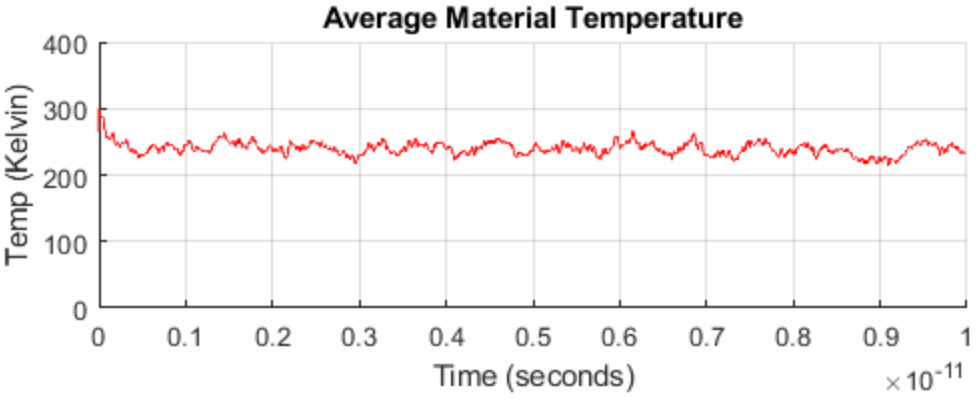
```
            if y(i) + dt*Vy(i) >= 6e-8 || y(i) + dt*Vy(i) <= 4e-8
                yt = y(i);
                Vy(i) = -Vy(i);
                y(i) = yp(i) + dt*Vy(i);
                yp(i) = yt;
            end
        else
             Vy(i) = -Vy(i);
        end
    end

    %implement scattering here, the velocity is re-thermalized
    if rand() < Prob
       Vx(i)  = -sqrt(2*kb*T/Mass + log(numElect)...
           + (1/2)*log(2*pi*kb*T/Mass)).*randn(1,1);

       Vy(i) = -sqrt(2*kb*T/Mass + log(numElect)...
           + (1/2)*log(2*pi*kb*T/Mass)).*randn(1,1);


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

    for ix = 1:99
         for iy = 1:99
            if((x(i) <= xn(ix+1) || x(i) > xn(ix))&&(y(i) <=...
                    yn(iy+1) || y(i) > yn(iy)))

                Vx(i) = Vx(i) + Ax(ix, iy).*dt;
                Vy(i) = Vy(i) + Ay(ix, iy).*dt;
            end
         end
     end

    Vt = sqrt(Vx(i)^2 + Vy(i)^2);           %As we loop to check
bounds
    tempSum = tempSum + (Mass*Vt^2)/(4*kb);%we might aswell do the
                                           %temp cacluations

    if i <= numVisable             %plot the difference in
position,
        figure(6);                 %but only a small number will
show
        subplot(2,1,1);
        plot([xp(i) x(i)], [yp(i) y(i)],'color',colorVec(i,:));
    end
```

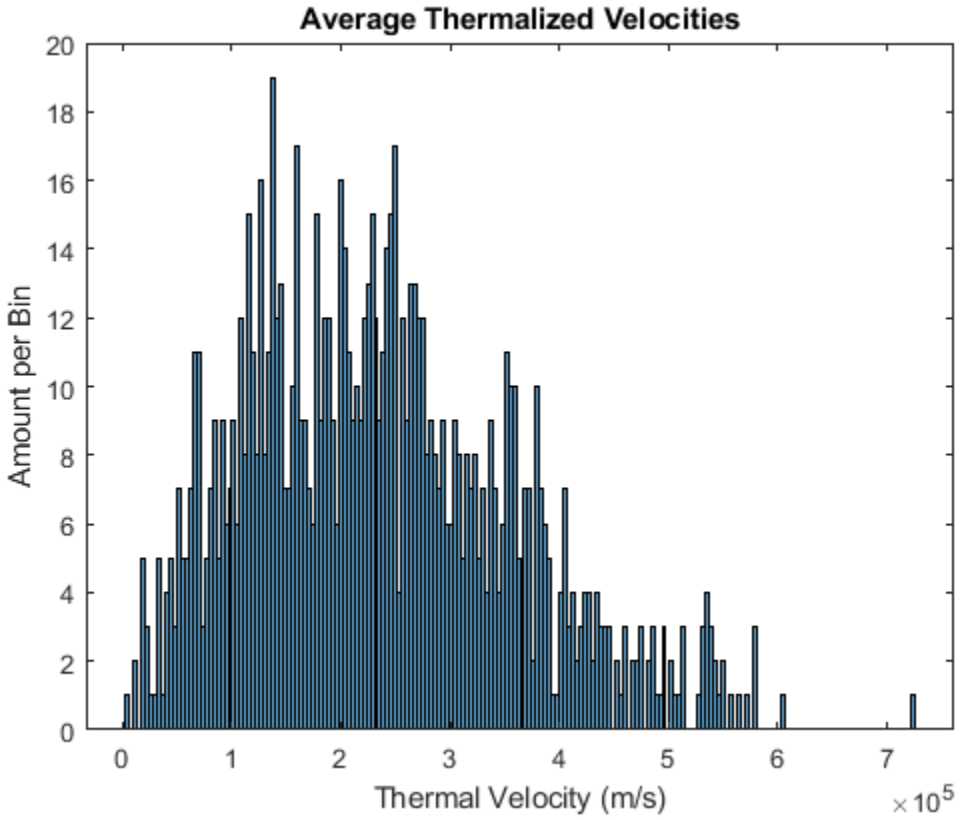
```
    end

    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(6);                         %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(8);
```

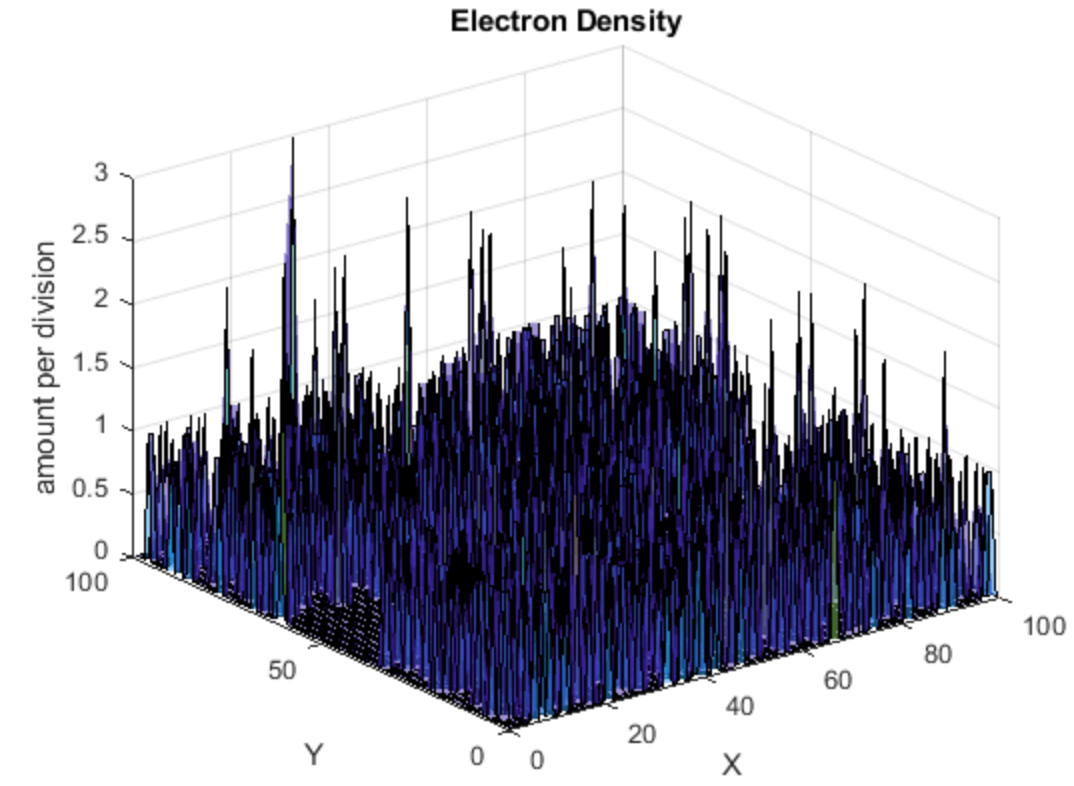
```
s1 = surf(1:xbox,1:ybox,c,'FaceAlpha',0.5);
title('Electron Density');
xlabel('X');
ylabel('Y');
zlabel('amount per division');


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

figure(9);
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(6);
    saveas(gcf, 'ElectronsInSiliconQ3.jpg');

    figure(7);
    saveas(gcf, 'VelocityHistQ3.jpg');

    figure(8);
    saveas(gcf, 'ElectronDensityQ3.jpg');

    figure(9);
    saveas(gcf, 'TempuratureMapQ4.jpg');
end
```

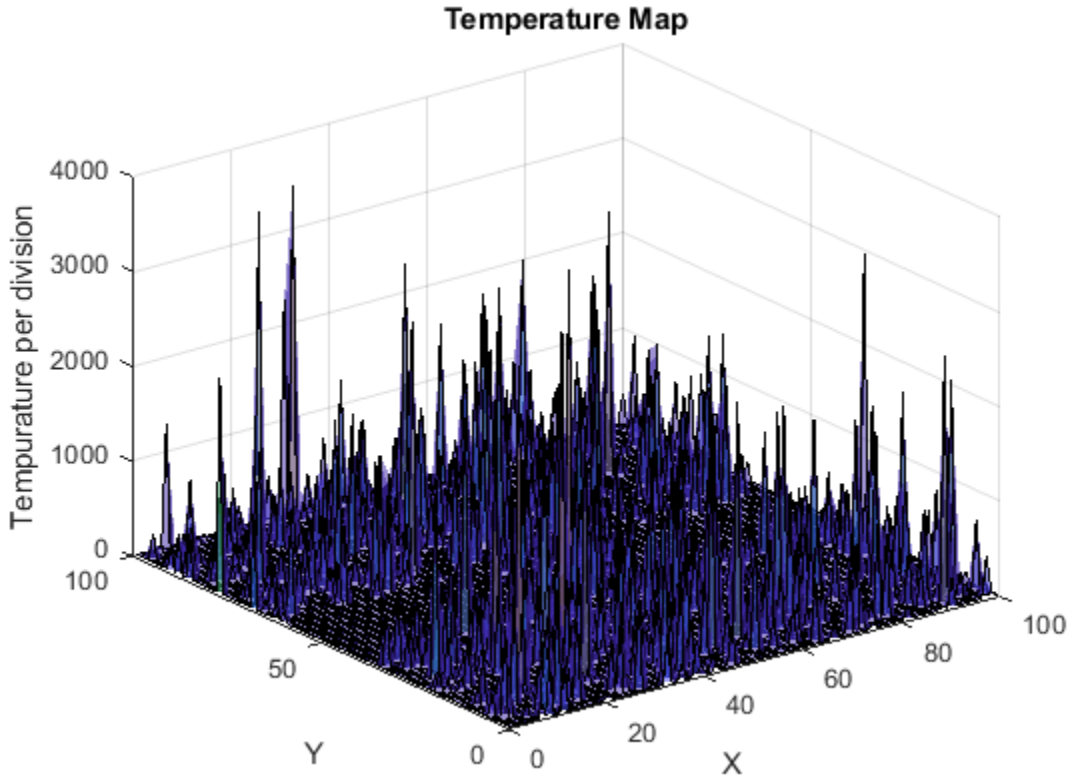
```
The proability to scatter is 0.0487706
The Avg velocity is: 2.354178e+05; Calculated Thermal Velocity:
 1.870193e+05
Mean Free Path Calcuated: 1.78317e-17 Avg time between collisions:
 2.00128e-13
```


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
20
18
16
14
12
10
8
6
4
2
0
Amount per Bin
0
1
2
3
4
5
6
7
Thermal Velocity (m/s)
×10^5


Electron Density
amount per division
3
2.5
2
1.5
1
0.5
0
100
50
0
Y
0
20
40
60
80
100
X


Temperature Map
Tempurature per division
4000
3000
2000
1000
0
100
50
0
Y
0
20
40
60
80
100
X

Published with MATLAB® R2017b