**Agberien Stephen 101047160**

## Part 1: Electron Modelling

Using Maxwell's principle of equipartition of energy and this is given by:

$$\overline{KE} = \frac{1}{2}kT = 2(\frac{1}{2}m\overline{v^2}) \Rightarrow \overline{v^2} = \frac{2kT}{m}$$

$$\overline{v^2} = \frac{2kT}{m}$$

```
mo = 9.11e-31; % rest mass of an electron (kg)
mn = 0.26 * mo; % effective mass of an electron (kg)
T = 300; % Temperature (K)
K = 1.38e-23; % boltzmann constant (J/K)
tmn = 0.2e-12; % mean time between collisions (seconds)
```

Thermal velocity calculation

```
vth = sqrt((2*K*T)/mn) %thermal velocity
```

```
vth =

   1.8697e+05
```

Mean of free path calculation

```
meanFP = vth * tmn
% The mean free path is 37 nm
```

```
meanFP =

   3.7394e-08
```

normal size of region

```
width  = 200e-9; % metres
length = 100e-9; % metres

nE = 10000; % Number of electrons
nT = 1000; % Steps
time = zeros(1, nT);
dt = (length)/vth/100; % Change in electron distance
temp = zeros(1, nT);
rBound = 0; % right boundary
lBound = 0; % left boundary
```

```
x = linspace(0, 200, 400)*10^(-9);
y = linspace(0, 100, 400)*10^(-9);
```

Using an array for the Positions and Velocities. The angles for the velocities are randomized, while keeping the thermal velocity at a constant level. The positions are updated randomly using the linspace in x and y above.

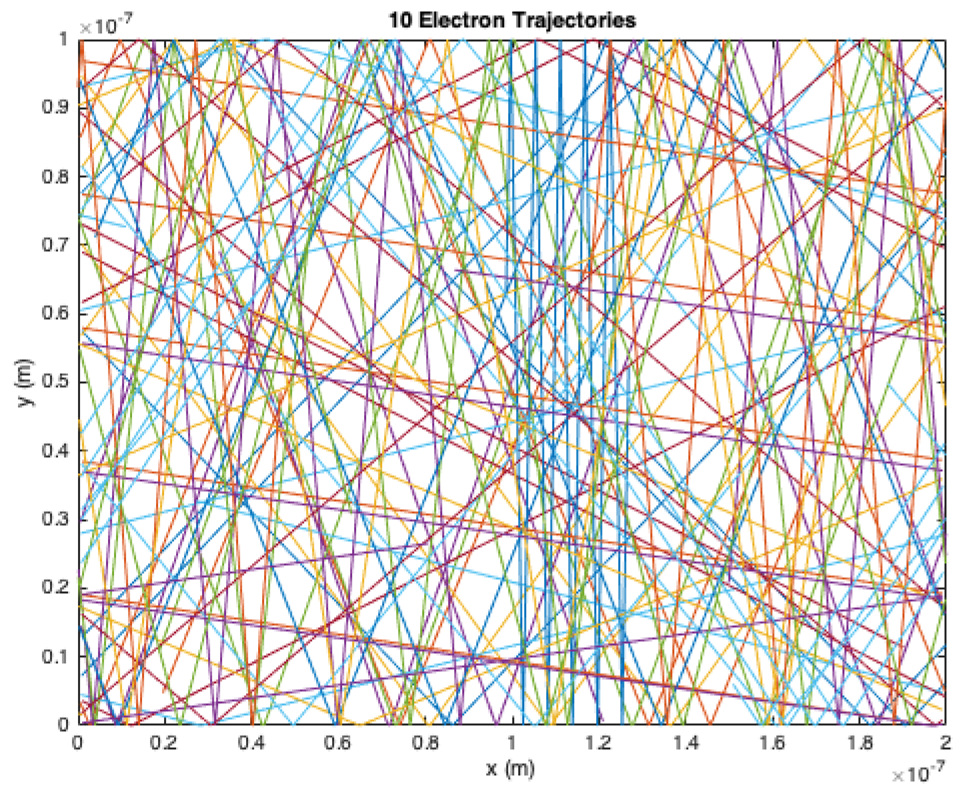
```
Px = zeros(nE, nT);
Py = zeros(nE, nT);
Vx = zeros(nE, nT);
Vy = zeros(nE, nT);


for i = 1 : nE
    angle = rand*pi*2;
    Px(i, 1) = x(randi(400));
    Py(i, 1) = y(randi(400));
    Vx(i, :) = vth * cos(angle);
    Vy(i, :) = vth * sin(angle);
end
```

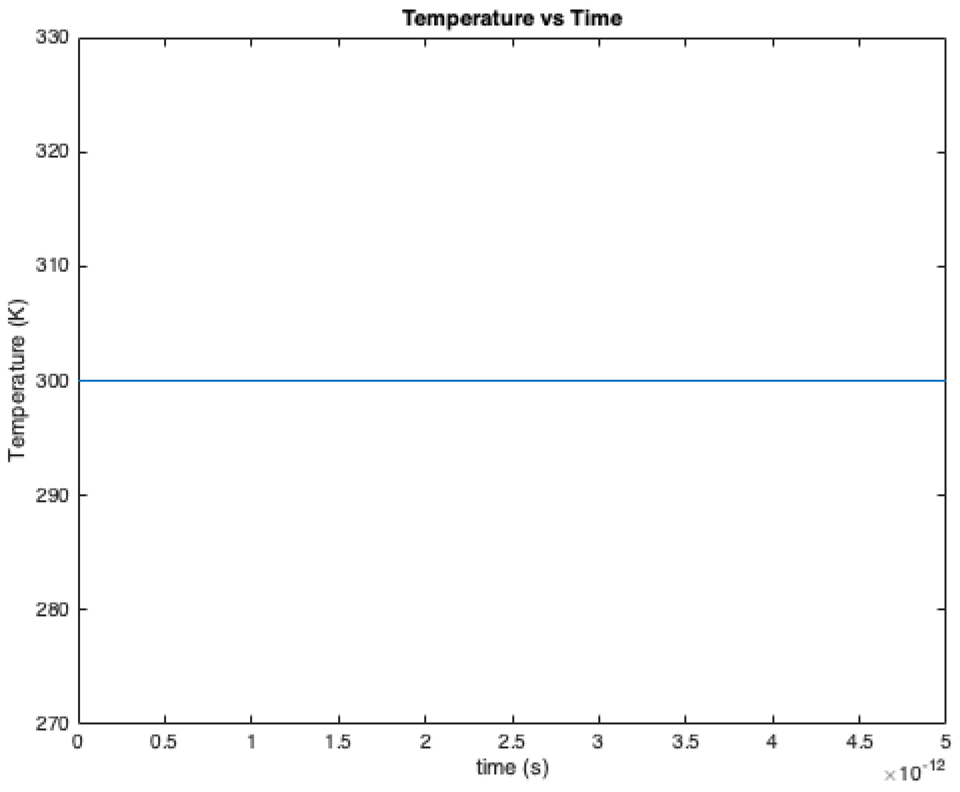
The following block of code updates the electrons' positions and velocities after each time step. In the block of code, if the position of the electron reaches the left boundary or right boundary, the position will wrap around and appear on the other side of the boundary, but if it reaches the top and bottom boundary it will bounce back. The steps for the for loop was gotten online. Tried using a true/false scenario but did not work as expected so I extracted this online.

```
for i = 1 : nE
    for j = 2 : nT
        if isnan(Px(i, j-1))
            if lBound == 1
                Px(i, j) = 0 + Vx(i, j-1)*dt;
            end
            if rBound == 1
                Px(i, j) = width + Vx(i, j-1)*dt;
            end
        else
            Px(i, j) = Px(i, j-1) + Vx(i, j-1)*dt;
        end

        if Px(i, j) > width
            lBound = 1;
            rBound = 0;
            Px(i, j) = NaN;
        end
        if Px(i, j) < 0
            lBound = 0;
            rBound = 1;
            Px(i, j) = NaN;
        end

        Py(i, j) = Py(i, j-1) + Vy(i, j-1)*dt;
        if Py(i, j) > length
            Py(i, j) = length;
            Vy(i, j:end) = -Vy(i, j-1);
        end
        if Py(i, j) < 0
            Py(i, j) = 0;
            Vy(i, j:end) = -Vy(i, j-1);
        end
    end
end

% Temperature analysis (should be constant, around 300K)

for i = 1:nT
        temp(i) = (sum(Vx(:, i).^2) + sum(Vy(:, i).^2))*mn/K/2/nE;
        if i > 1
            time(i) = time(i-1) + dt;
        end
end
```

Plots of the trajectory change in position and temperature over a certain time

```
figure(1)
for i = 1 : 10
    plot(Px(i, :), Py(i, :))
    xlabel('x (m)')
    ylabel('y (m)')
    title('10 Electron Trajectories')
    pause(0.0001)
    hold on
end

figure(2)
plot(time, temp)
title('Temperature vs Time')
xlabel('time (s)')
ylabel('Temperature (K)')
ylim([0.9*T 1.1*T])
xlim([0 5e-12])
hold off
```


10 Electron Trajectories
×10^-7
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
y (m)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
x (m)
×10^-7


Temperature vs Time
330
320
310
300
290
280
270
Temperature (K)
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
time (s)
×10^-12

**Part 2: Collisions with Mean Free Path**

Here, initial velocity is based on Maxwell-Boltzmann distribution. probability of scattering in a time step

```
pScat = 1 - exp(-dt/tmn);
```

Velecity in x and y is gaussian therefore the overall is a Maxwell-Boltzman distribution

```
v_o = makedist('Normal', 'mu', 0, 'sigma', sqrt(K*T/mn));

Px = zeros(nE, nT);
Py = zeros(nE, nT);
Vx = zeros(nE, nT);
Vy = zeros(nE, nT);

for i = 1 : nE
    Px(i,1) = x(randi(400));
    Py(i,1) = y(randi(400));
    Vx(i,:) = random(v_o);
    Vy(1,:) = random(v_o);
end
```

Average velocity calculation should be the average of all vth

```
vAvg = sqrt(sum(Vx(:, 1).^2)/nE + sum(Vy(:, 1).^2/nE));
```

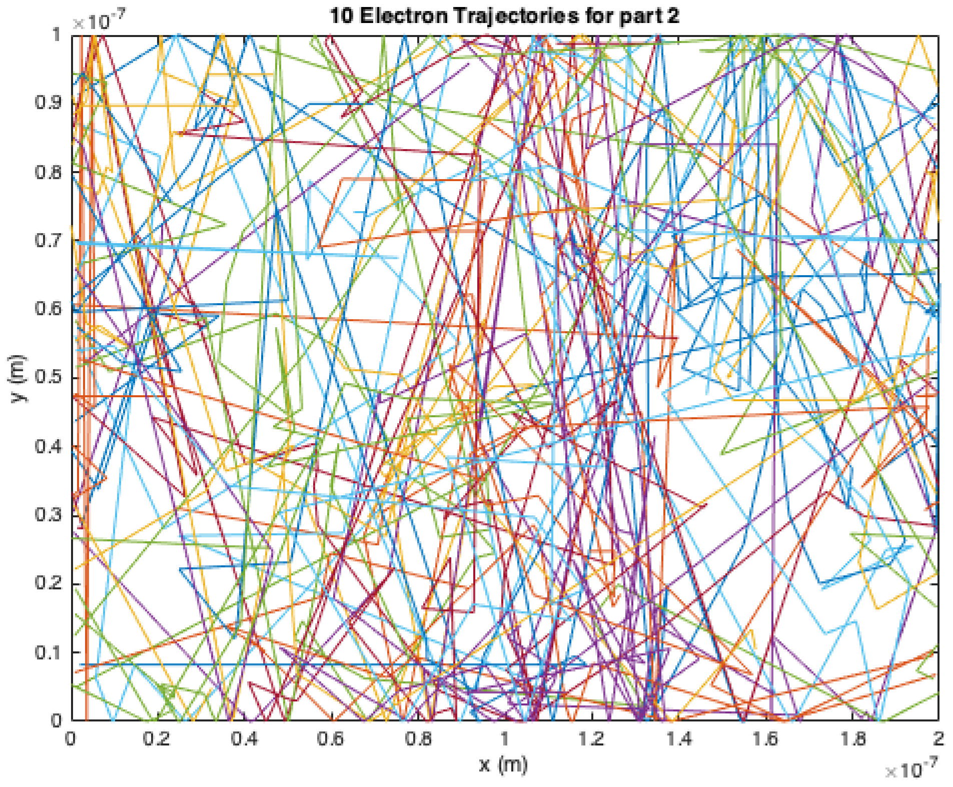
As done in part 1, this block will update the positions and velocities of the electrons. This block of code randomly checks each electron after each steo to determine if it scatters or not. If it scatters, a new velocity will be assigned to the electron using the Maxwell-Boltzmann distribution.

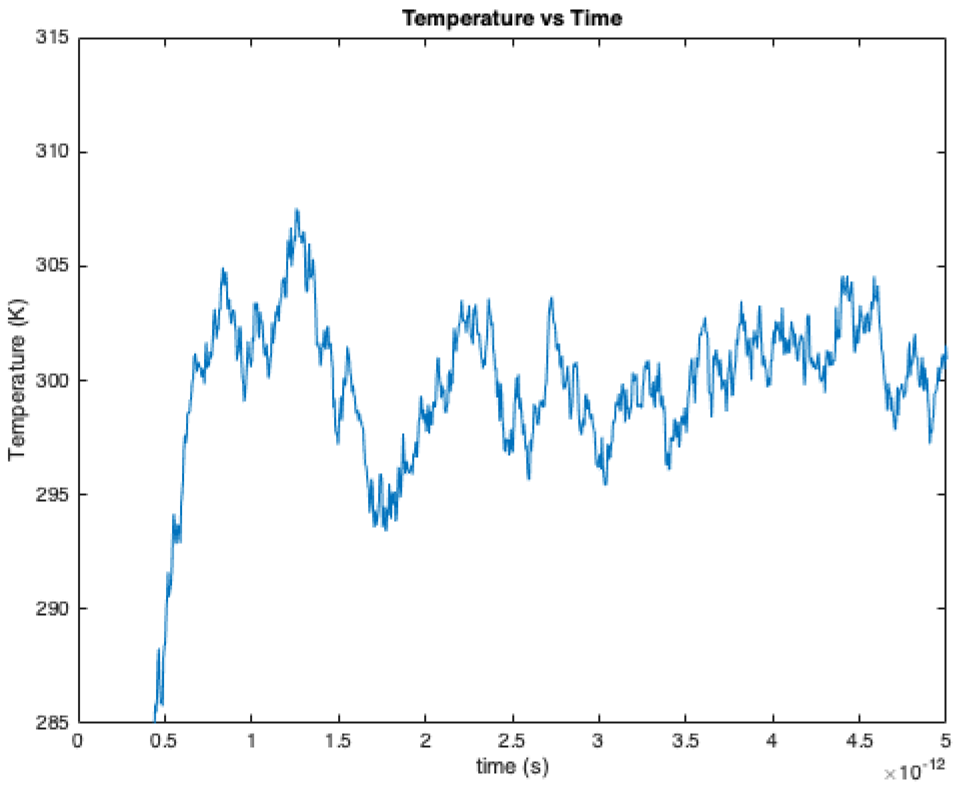
```
for j = 1 : nE
    for k = 2 : nT
        if isnan(Px(j, k-1))
            if left == 1
                Px(j, k) = 0 + Vx(j, k-1)*dt;
            end
            if right == 1
                Px(j, k) = width + Vx(j, k-1)*dt;
            end
        else
            Px(j, k) = Px(j, k-1) + Vx(j, k-1)*dt;
        end

        if Px(j, k) > width
            left = 1;
            right = 0;
            Px(j, k) = NaN;
        end
        if Px(j, k) < 0
            left = 0;
            right = 1;
            Px(j, k) = NaN;
        end

        Py(j, k) = Py(j, k-1) + Vy(j, k-1)*dt;
        if Py(j, k) > length
            Py(j, k) = length;
            Vy(j, k:end) = -Vy(j, k-1);
        end
        if Py(j, k) < 0
            Py(j, k) = 0;
            Vy(j, k:end) = -Vy(j, k-1);
        end
        if pScat > rand()
            Vx(j, k:end) = random(v_o);
            Vy(j, k:end) = random(v_o);
        end
    end
end

% Temperature analysis (should be constant, 300K)
for i = 1:nT
        temp(i) = (sum(Vx(:, i).^2) + sum(Vy(:, i).^2))*mn/K/2/nE;
        if i > 1
            time(i) = time(i-1) + dt;
        end
end
```

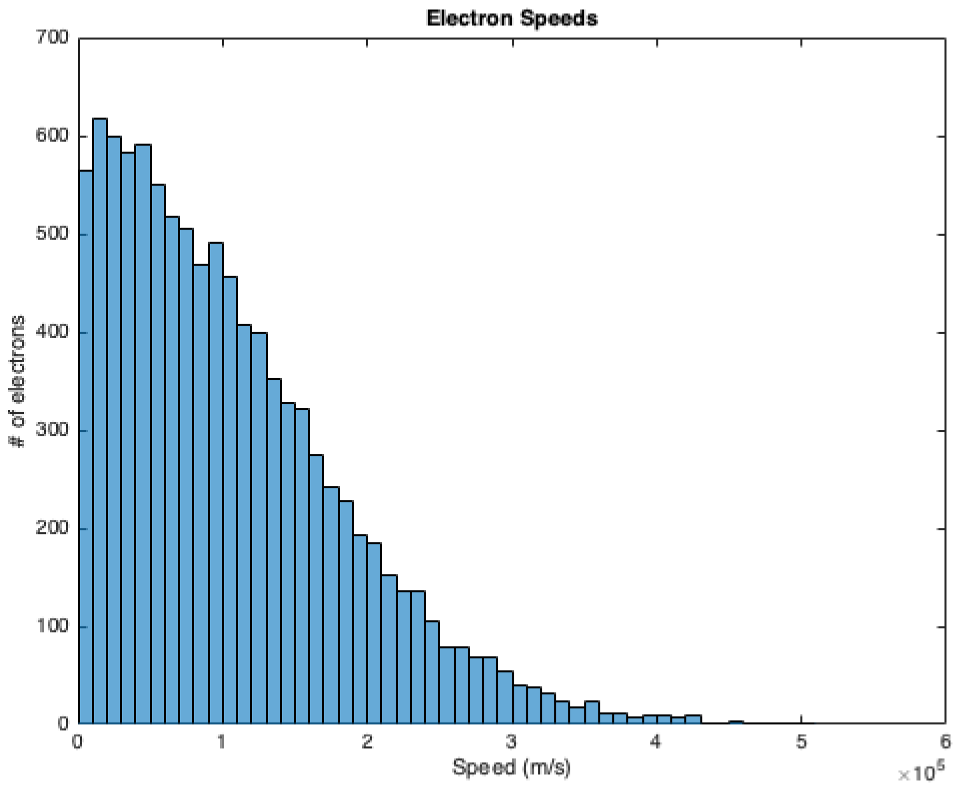
Velocities is plotted as a histogram, displaying the Maxwell-Boltzmann

```
figure(3)
for i = 1:10
    plot(Px(i, :), Py(i, :))
    xlabel('x (m)')
    ylabel('y (m)')
    title('10 Electron Trajectories for part 2')
    pause(0.0001)
    hold on
end

figure(4)
plot(time, temp)
title('Temperature vs Time')
xlabel('time (s)')
ylabel('Temperature (K)')
xlim([0 5e-12])
ylim([0.95*T 1.05*T])

figure(5)
vNet = sqrt(Vx(:, 1).^2 + Vy(:, 1).^2); % velocity of electron
histogram(vNet)
title('Electron Speeds')
xlabel('Speed (m/s)')
ylabel('# of electrons')
hold off

% %
% The temperature varies for the scattering of particles, but the
 average
```

```
% of the temperature flunctuates around 300K. The average speed tends
 to
% move towards the Vth and and distribution is similar to a
% Maxwell-Boltzmann distribution
```




Temperature vs Time
Temperature (K)
time (s)
×10^-12
315
310
305
300
295
290
285
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

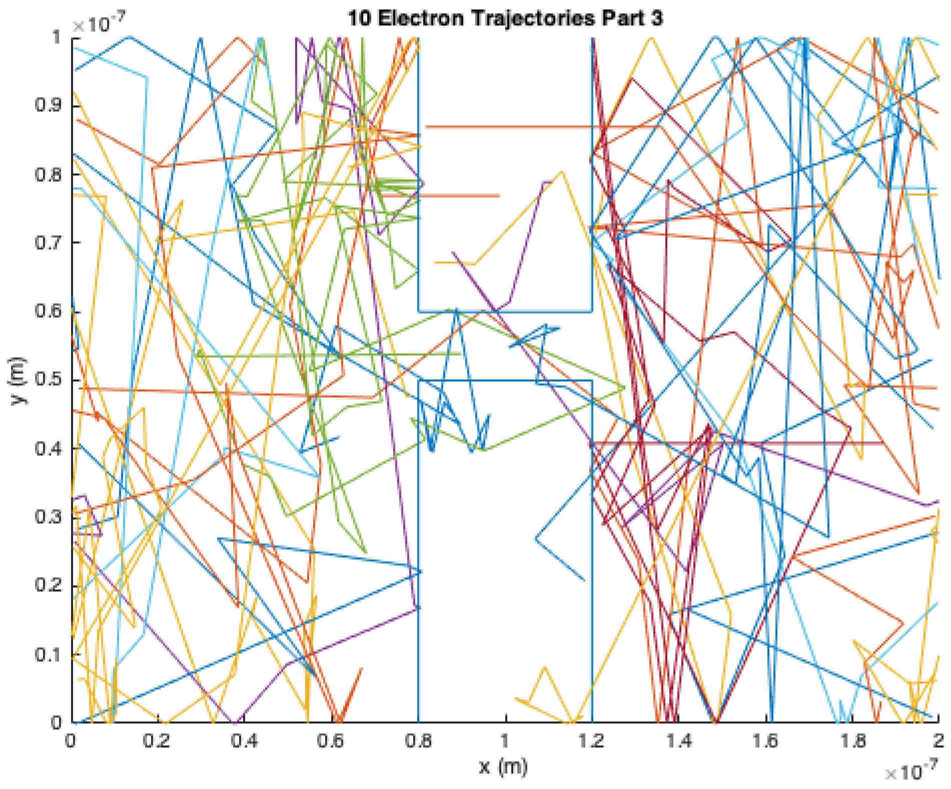

Electron Speeds
# of electrons
Speed (m/s)
×10^5
700
600
500
400
300
200
100
0
0
1
2
3
4
5
6

**Part3: Enhancements** Here, boundaries are either specular or diffusive and this simulation includes obstacles(boxes)

```
Px = zeros(nE, nT);
Py = zeros(nE, nT);
Vx = zeros(nE, nT);
Vy = zeros(nE, nT);

for i = 1 : nE
    Px(i,1) = x(randi(400));
    Py(i,1) = y(randi(400));
    Vx(i,:) = random(v_o);
    Vy(1,:) = random(v_o);
end
```

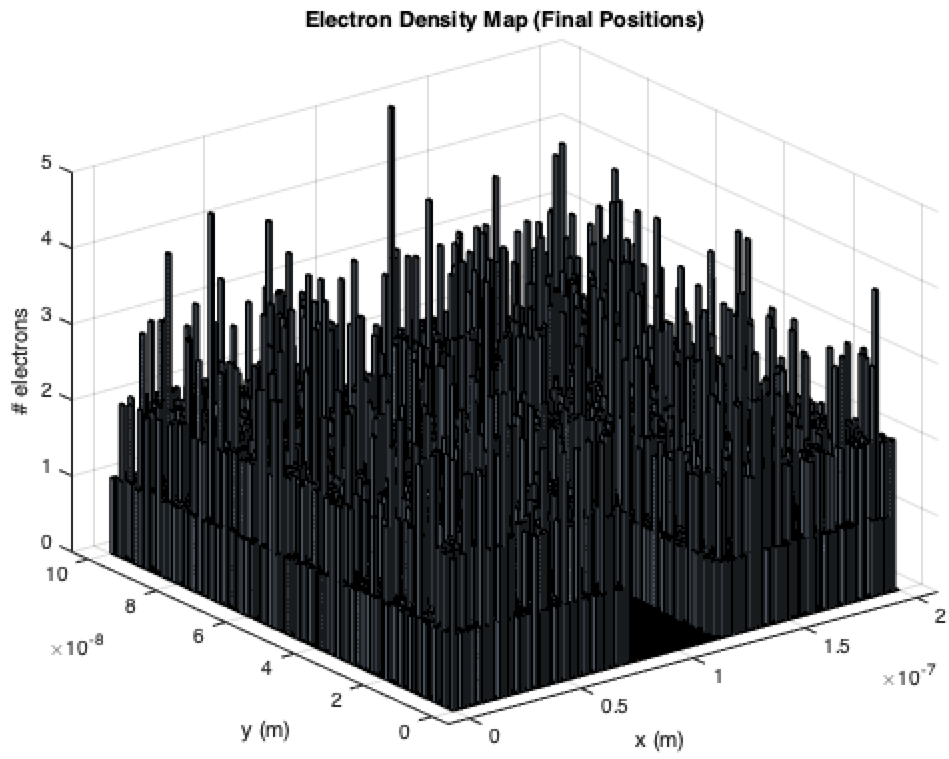
As done in part 1 and part 2, this block will update the positions and velocities of the electrons This block of code randomly checks each electron after each steo to determine if it scatters or not. If it scatters, a new velocity will be assigned to the electron using the Maxwell-Boltzmann distribution.

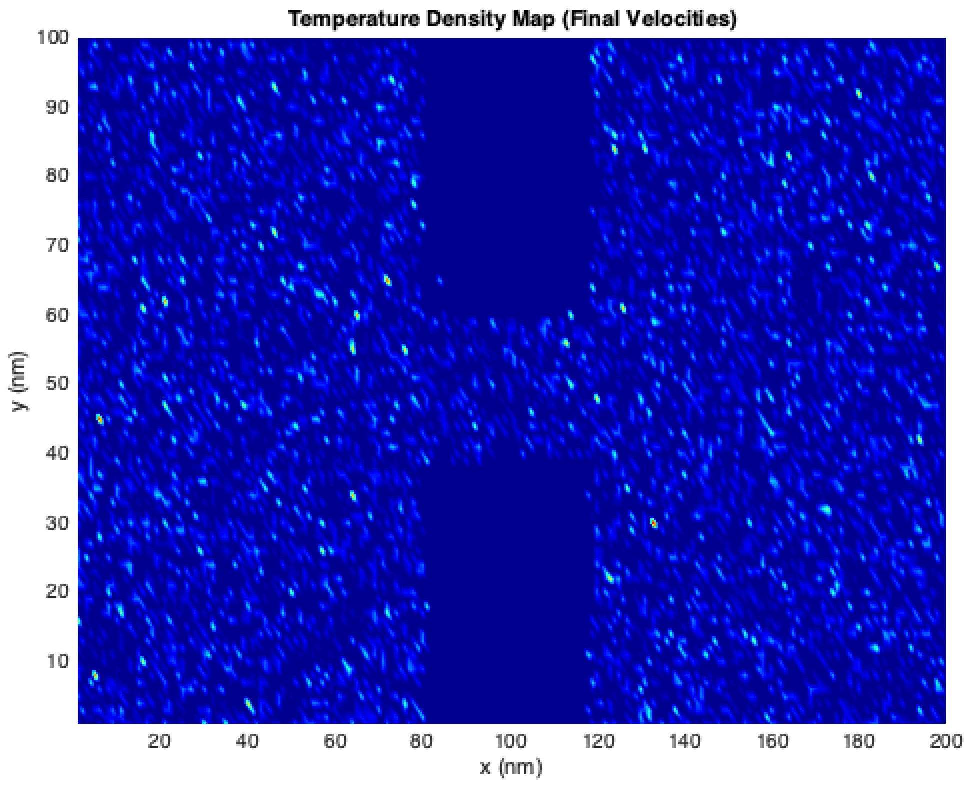
```
for j = 1 : nE
    for k = 2 : nT
        if isnan(Px(j, k-1))
            if left == 1
                Px(j, k) = 0 + Vx(j, k-1)*dt;
            end
            if right == 1
                Px(j, k) = width + Vx(j, k-1)*dt;
            end
        else
            Px(j, k) = Px(j, k-1) + Vx(j, k-1)*dt;
        end

        if Px(j, k) > width
            left = 1;
            right = 0;
            Px(j, k) = NaN;
        end
        if Px(j, k) < 0
            left = 0;
            right = 1;
            Px(j, k) = NaN;
        end

        Py(j, k) = Py(j, k-1) + Vy(j, k-1)*dt;
        if Py(j, k) > length
            Py(j, k) = length;
            Vy(j, k:end) = -Vy(j, k-1);
        end
        if Py(j, k) < 0
            Py(j, k) = 0;
            Vy(j, k:end) = -Vy(j, k-1);
        end
        if pScat > rand()
            Vx(j, k:end) = random(v_o);
            Vy(j, k:end) = random(v_o);
            end
            if (Px(j, k) >= 80e-9 && Px(j, k) <= 120e-9 && Py(j, k) >=
 60e-9) || (Px(j, k) >= 80e-9 && Px(j, k) <= 120e-9 && Py(j, k) <=
 40e-9)
                if (Px(j, k-1) <= 80e-9 && Py(j, k-1) <= 40e-9) || (Px(j,
 k-1) <= 80e-9 && Py(j, k-1) >= 60e-9) || (Px(j, k-1) >= 120e-9 &&
 Py(j, k-1) <= 40e-9) || (Px(j, k-1) >= 120e-9 && Py(j, k-1) >= 40e-9)
                    Vx(j, k:end) = -Vx(j, k-1);
                end
                if Px(j, k-1) >= 80e-9 && Px(j, k-1) <= 120e-9 && Py(j,
 k-1) <= 60e-9 && Py(j, k-1) >= 40e-9
                    Vy(j, k:end) = -Vy(j, k-1);
                end
            end
        end
    end

    figure(6)
    for i = 1:10
        xLim = [80e-9 80e-9 120e-9 120e-9];
        yLim1 = [0 50e-9 50e-9 0];
        yLim2 = [100e-9 60e-9 60e-9 100e-9];
        line(xLim, yLim1)
        hold on
        line(xLim, yLim2)
        plot(Px(i, :), Py(i, :))
        xlabel('x (m)')
        ylabel('y (m)')
        title('10 Electron Trajectories Part 3')
    end
```



The following block of code was gotten from an online source to help plot out the Electron Density Map and Temperature density Map

```
densityM = [Px(:, 1000) Py(:, 1000)];
figure(7)
hist3(densityM, [200 100])
title('Electron Density Map (Final Positions)')
xlabel('x (m)')
ylabel('y (m)')
zlabel('# electrons')

tempx = zeros(ceil(200), ceil(100));
tempy = zeros(ceil(200), ceil(100));
tempn = zeros(ceil(200), ceil(100));

for z = 1:nE
    x = floor(Px(z, 1000)/1e-9);
    y = floor(Py(z, 1000)/1e-9);
    if (x == 0 || isnan(x))
        x = 1;
    end
    if (y == 0 || isnan(y))
        y = 1;
    end
    tempy(x, y) = tempy(x, y) + Vy(z, 1000)^2;
    tempx(x, y) = tempx(x, y) + Vx(z, 1000)^2;
    tempn(x, y) = tempn(x, y) + 1;
end

temp2 = (tempx + tempy) .* mn ./ K ./ 2 ./ tempn;
temp2(isnan(temp2)) = 0;
temp2 = temp2';
figure(8)
xtemp = linspace(1, 200, 200);
ytemp = linspace(1, 100, 100);
pcolor(xtemp, ytemp, temp2)
shading interp
colormap(jet)
title('Temperature Density Map (Final Velocities)')
xlabel('x (nm)')
ylabel('y (nm)')
hold off
```





*Published with MATLAB® R2015b*