

---

# ELEC 4700 - Assignment 1

## Table of Contents

|  |    |
|--|----|
| Question 1 .....   | 1  |
| A) Velocity .....  | 1  |
| B) Mean Free Path .....  | 2  |
| C) Program .....   | 2  |
| Question 2 .....   | 5  |
| A) Maxwell Boltzmann Distributed Velocity .....  | 5  |
| B) Probability of Scattering .....   | 6  |
| C) Average Temperature Over Time .....   | 8  |
| The graph shows now fluctuation in the temperature which is expected due to the scattering. .... | 8  |
| D) Mean Free Path and Mean Time Between Collisions .....   | 9  |
| Question 3 .....   | 11 |
| A) Adding A Bottle Neck .....  | 11 |
| B) Boundaries Specular or Diffusive .....  | 12 |
| C) Electron Density Map .....  | 15 |
| D) Temperature Map .....   | 16 |

Completed By: Joel Demetre (100943543)

Due: February 4th 2018

## Question 1

### A) Velocity

Calculation of Velocity

$$v_i = \sqrt{\frac{KT}{m_n}}$$

$$v_{th} = \sqrt{v_x^2 + v_y^2} = \sqrt{\frac{2KT}{m_n}}$$

where  $m_n$  is the effective mass of electrons  $m_n = 0.26m_o$  T is the temperature in Kelvin (300K), and K is the Boltzmann Constant

```
T = 300;  
Kbolt = 1.38064852E-23;  
mo = 9.11e-31;  
mn = mo * 0.26;  
vx = (Kbolt*T/mn)^.5;  
vy = (Kbolt*T/mn)^.5;  
vth = (vx^2 + vy^2)^.5;
```

$v_{th}$  is then equal to 166 000 m/s

## B) Mean Free Path

The mean free path can be determined from the velocity multiplied by the time between collisions

$$l = v_{th}\tau_{mn}$$

where  $l$  is the mean free path,  $\tau_{mn}$  is the time between collisions and  $v_{th}$  is the velocity.

```
tmn = 0.2e-12; % in seconds
MeanFreePath = tmn*vth; % in meters
```

The Mean Free Path ( $l$ ) is then calculated to be 37.4 nm.

## C) Program

```
%Boundary Conditions
xlimits = [0, 2e-9];
ylimits = [0, 1e-9];
PlotHowMany = 10;
Timestep = 5e-17;
endtime = Timestep*1000;
NumParticles = 1000;

%%SETUP INITIAL PARAMETERS
mycolors = hsv(PlotHowMany);
xprev = zeros(1, NumParticles);
yprev = zeros(1, NumParticles);
x = zeros(3, NumParticles);
y = zeros(2, NumParticles);
temp = zeros(2, NumParticles);

%Start the random distribution in x position
x(1,:) = rand(1, NumParticles);
y(1,:) = rand(1, NumParticles);
x(1,:) = xlimits(1) + x(1,:).*(xlimits(2) - xlimits(1));
y(1,:) = ylimits(1) + y(1,:).*(ylimits(2) - ylimits(1));

%Assign the velocity in x and y direction
x(3,:) = rand(1, NumParticles)*2*pi;
x(2,:) = vth.*cos(x(3,:));
y(2,:) = vth.*sin(x(3,:));

%Create the Time loop
figure(1);
xlim([xlimits(1), xlimits(2)]);
ylim([ylimits(1), ylimits(2)]);
Temperature = zeros(floor(endtime/Timestep)+ 1,1);
for i = 0:Timestep:endtime
    xprev(1,:) = x(1,:);
    yprev(1,:) = y(1,:);

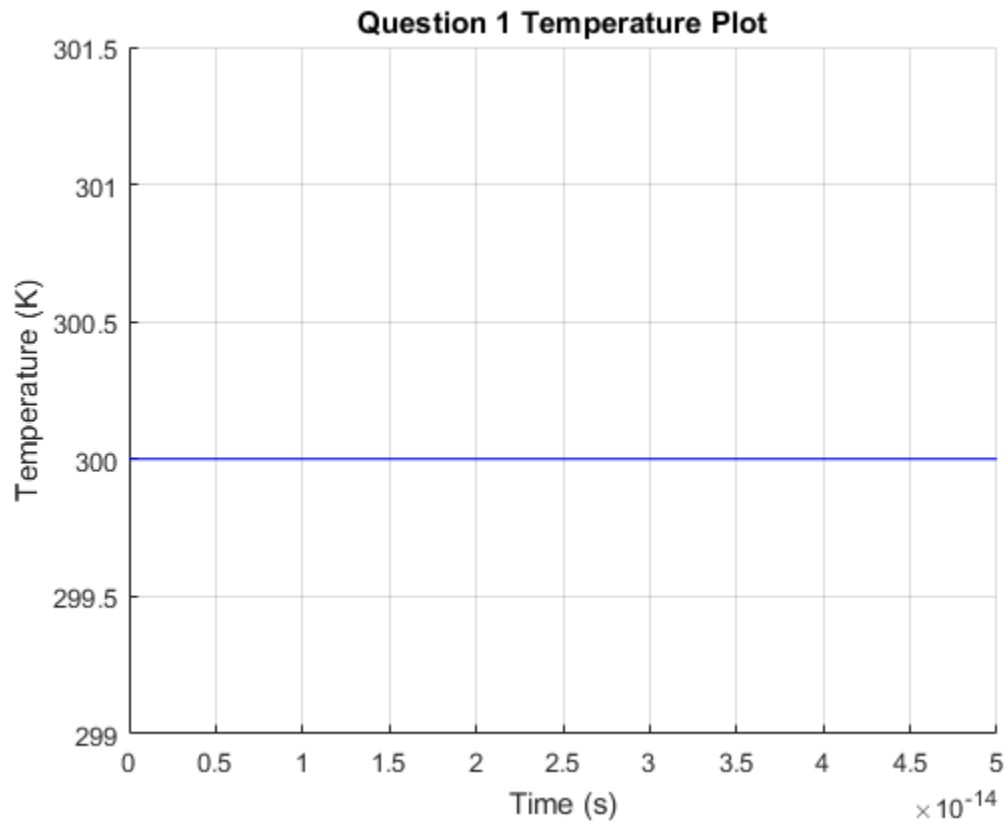
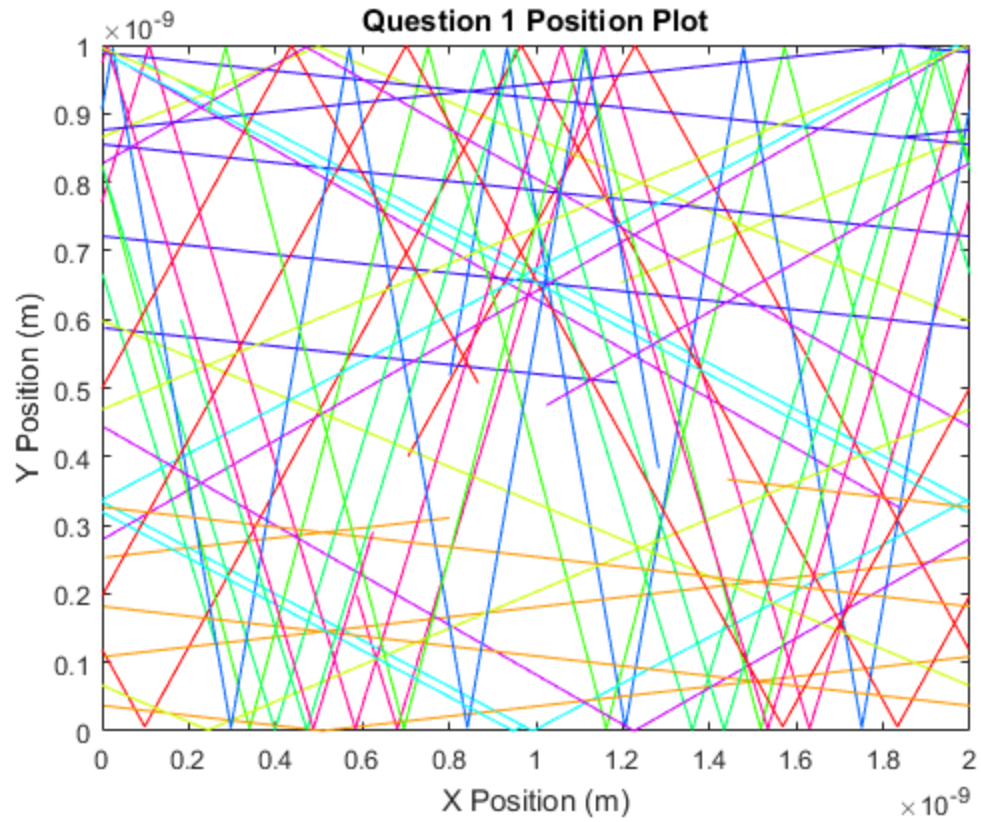
    %Loop Through Each Particle
    for kt = 1:NumParticles
```

```
%Check for the Limits
if x(1,kt) + x(2,kt)*Timestep < xlimits(1)
    x(1,kt) = xlimits(2);
    xprev(1,kt) = xlimits(2);
elseif x(1,kt) + x(2,kt)*Timestep > xlimits(2)
    x(1,kt) = xlimits(1);
    xprev(1,kt) = xlimits(1);
end
if y(1,kt) + y(2,kt)*Timestep < ylimits(1) || y(1,kt) +
y(2,kt)*Timestep > ylimits(2)
    y(2,kt) = -y(2,kt);
end
%Update Positions
x(1,kt) = x(1,kt) + x(2,kt).*Timestep;
y(1,kt) = y(1,kt) + y(2,kt).*Timestep;
%Plot the new positions
if kt <= PlotHowMany
    plot([xprev(1,kt), x(1,kt)], [yprev(1,kt), y(1,kt)], 'color',
mycolors(kt,:) );
    hold on;
end

end

VelSquared = mean((x(2,:).^2 + y(2,:).^2));
CalcTemp = VelSquared*mn/2/Kbolt;
title(['Average Temperature: ' num2str(CalcTemp)]);
Temperature(round(i/Timestep) + 1,1) = CalcTemp;
end

figure(1);
title('Question 1 Position Plot');
xlabel('X Position (m)');
ylabel('Y Position (m)');
hold off;
figure(3);
hold on;
title('Question 1 Temperature Plot');
plot(0:Timestep:endtime, Temperature(:, 1), 'b');
xlabel('Time (s)');
ylabel('Temperature (K)');
grid on;
hold off;
```



We can see that the temperature remains constant for the electrons as it should and that the movement of the electrons is elastic collisions on the y and flows freely on the x-axis.

## Question 2

```
%Initlialize some parameters
xprev = zeros(1, NumParticles);
yprev = zeros(1, NumParticles);
x = zeros(3,NumParticles);
y = zeros(2,NumParticles);
temp = zeros(2, NumParticles);
scatTime = zeros(1,NumParticles);
endtime = Timestep*30000;
graphfor = Timestep*1000;

%Start the random distribution in x position
x(1,:) = rand(1,NumParticles);
y(1,:) = rand(1,NumParticles);
x(1,:) = xlimits(1) + x(1,:).*(xlimits(2) - xlimits(1));
y(1,:) = ylimits(1) + y(1,:).*(ylimits(2) - ylimits(1));
```

## A) Maxwell Boltzmann Distributed Velocity

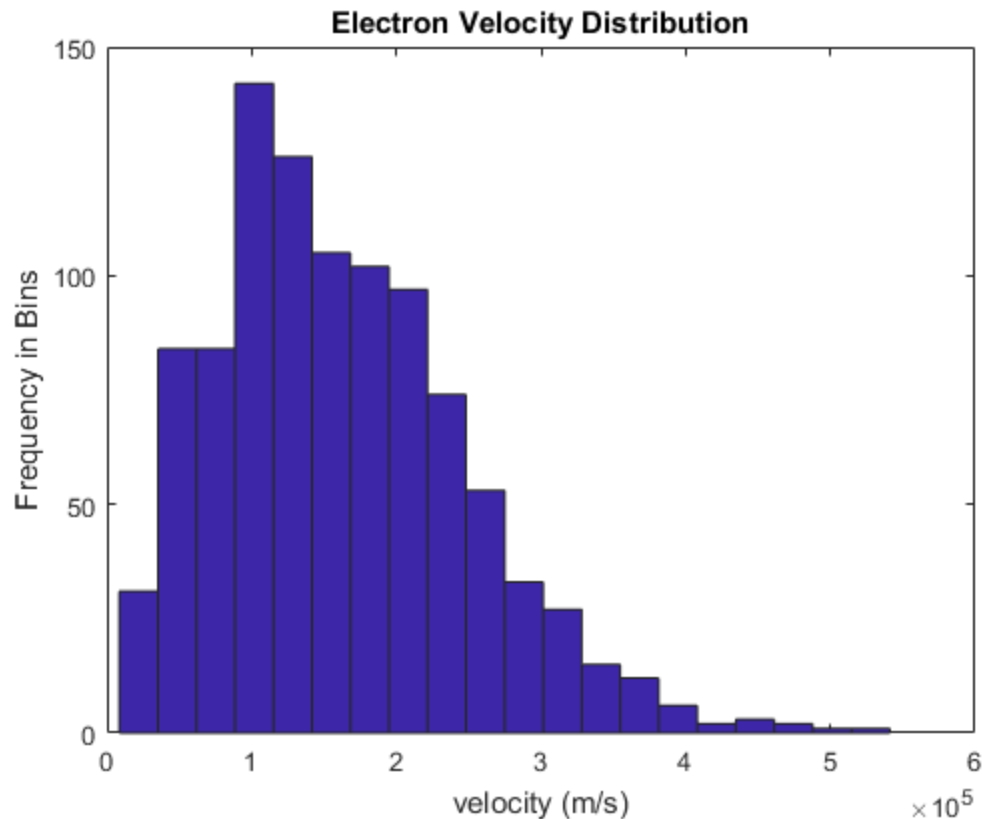
The Maxwell Boltzmann Distribution can be taken as a Gaussian Distribution with a standard deviation

$$\text{of } \sqrt{\frac{KT}{m_n}}$$

```
%Assign the random velocity and random angle

temp(1,:) = ((sqrt(Kbolt*T/mn)*randn(1, NumParticles)).^2 +
  (sqrt(Kbolt*T/mn)*randn(1, NumParticles)).^2).^5;
x(3,:) = rand(1, NumParticles)*2*pi;
x(2,:) = temp(1,:).*cos(x(3,:));
y(2,:) = temp(1,:).*sin(x(3,:));
figure(2);
hist(sqrt(x(2,:).^2 + y(2,:).^2),20);

title('Electron Velocity Distribution');
xlabel('velocity (m/s)');
ylabel('Frequency in Bins');
```



## B) Probability of Scattering

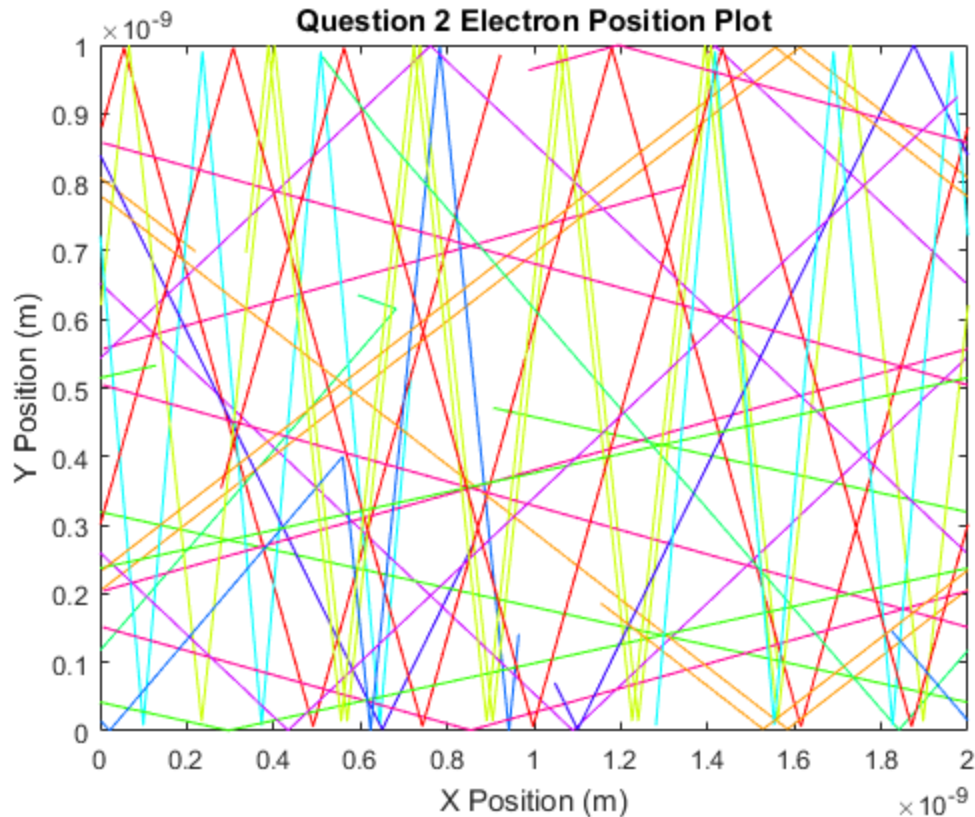
The probability of having a scattering event is  $1 - e^{-\frac{\Delta t}{\tau_{mn}}}$

`Pscat = 1-exp(-Timestep/tmn);`

In this specific case with a time step of  $9 \times 10^{-17} \text{ s}$  and a mean time between collisions  $\tau_{mn}$  of 0.2ps the Probability of Scattering is  $4.4990 \times 10^{-4}$

```
figure(4);
%Update the Position
ScatterTime = zeros(floor(endtime/Timestep)+ 1,1);
Temperature = zeros(floor(endtime/Timestep)+ 1,1);
MFP = zeros(floor(endtime/Timestep)+ 1,1);
for i = 0:Timestep:endtime
    % pause(.1);
    xprev(1,:) = x(1,:);
    yprev(1,:) = y(1,:);
    for kt = 1:NumParticles
        if x(1,kt) + x(2,kt)*Timestep < xlimits(1)
            x(1,kt) = xlimits(2);
            xprev(1,kt) = xlimits(2);
        elseif x(1,kt) + x(2,kt)*Timestep > xlimits(2)
            x(1,kt) = xlimits(1);
            xprev(1,kt) = xlimits(1);
        end
    end
end
```

```
end
    if y(1,kt) + y(2,kt)*Timestep < ylimits(1) || y(1,kt) +
y(2,kt)*Timestep > ylimits(2)
        y(2,kt) = -y(2,kt);
    end
    x(1,kt) = x(1,kt) + x(2,kt).*Timestep;
    y(1,kt) = y(1,kt) + y(2,kt).*Timestep;
    if kt <= PlotHowMany && i < graphfor
        plot([xprev(1,kt), x(1,kt)], [yprev(1,kt), y(1,kt)], 'color',
mycolors(kt,:) );
        hold on;
    end
    %Scattering Check
    if Pscat>rand()
        temp = (normrnd(0, sqrt(Kbolt*T/mn)).^2 + normrnd(0,
sqrt(Kbolt*T/mn)).^2).^5;
        x(3,kt) = rand*2*pi;
        x(2,kt) = temp*cos(x(3,kt));
        y(2,kt) = temp*sin(x(3,kt));
        scatTime(1,kt) = 0;
    end
end
hold on;
AvgScat = mean(scatTime(1,:));
scatTime(1,:) = scatTime(1,:) + Timestep;
VelSquared = mean((x(2,:).^2 + y(2,:).^2));
CalcTemp = VelSquared*mn/2/Kbolt;
%title(['Average Temperature: ' num2str(CalcTemp) ' Average
Scatter Time: ' num2str(AvgScat)]);
xlim([xlimits(1), xlimits(2)]);
ylim([ylimits(1), ylimits(2)]);
ScatterTime(round(i/Timestep) + 1,1) = AvgScat;
MFP(round(i/Timestep) + 1,1) = VelSquared^.5*AvgScat;
figure(4);
Temperature(round(i/Timestep) + 1,1) = CalcTemp;
end
title('Question 2 Electron Position Plot');
xlabel('X Position (m)');
ylabel('Y Position (m)');
hold off;
```



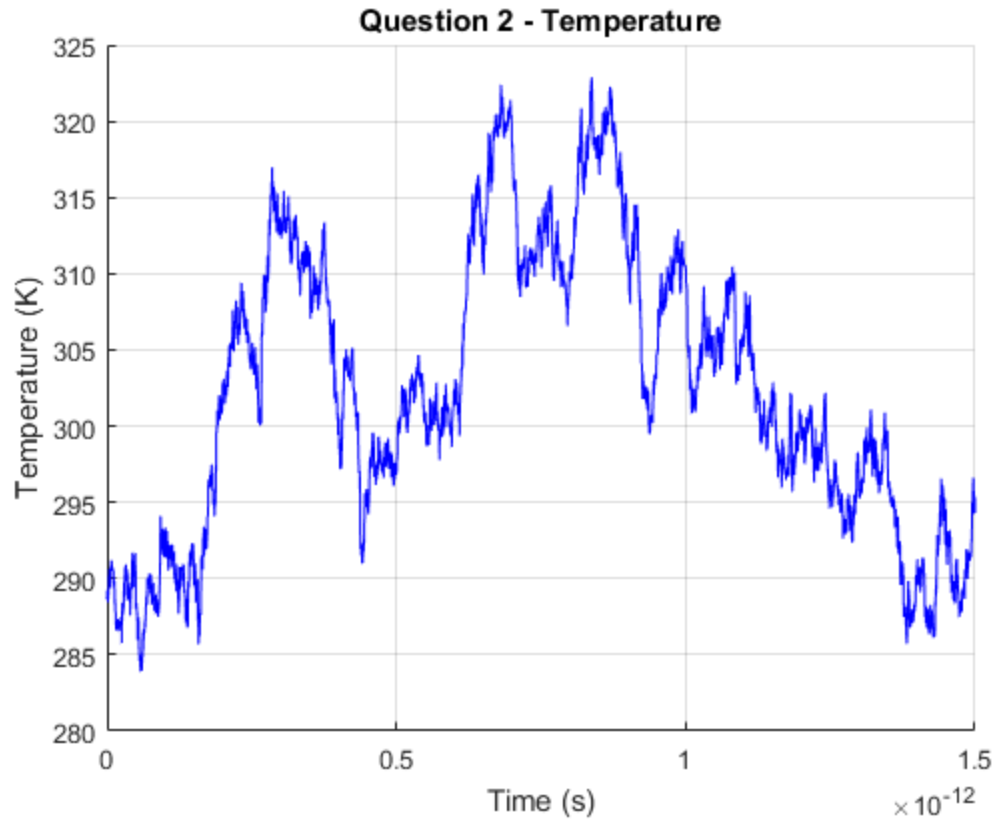
We can see the scattering occurring with the Electron Position Graph above.

## C) Average Temperature Over Time

The graph shows now fluctuation in the temperature which is expected due to the scattering.

```
figure(5);
hold on;
plot(0:Timestep:endtime, Temperature(:, 1), 'b');
title('Question 2 - Temperature');
xlabel('Time (s)');
ylabel('Temperature (K)');
grid on;
hold off;
```



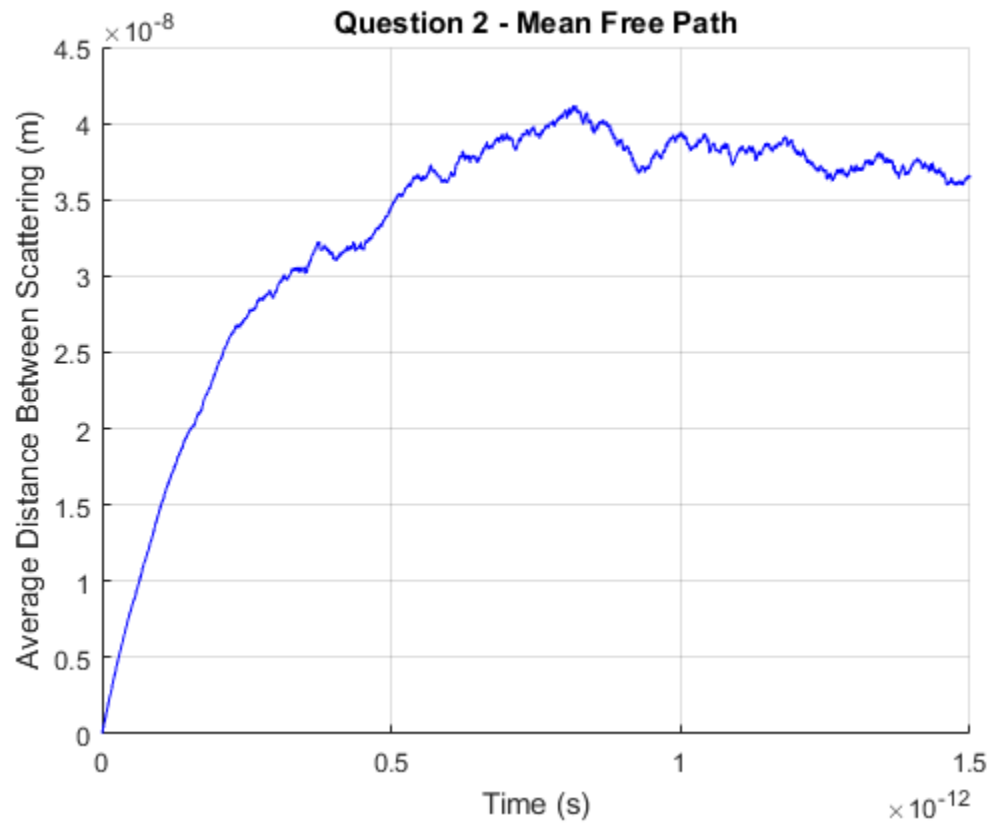
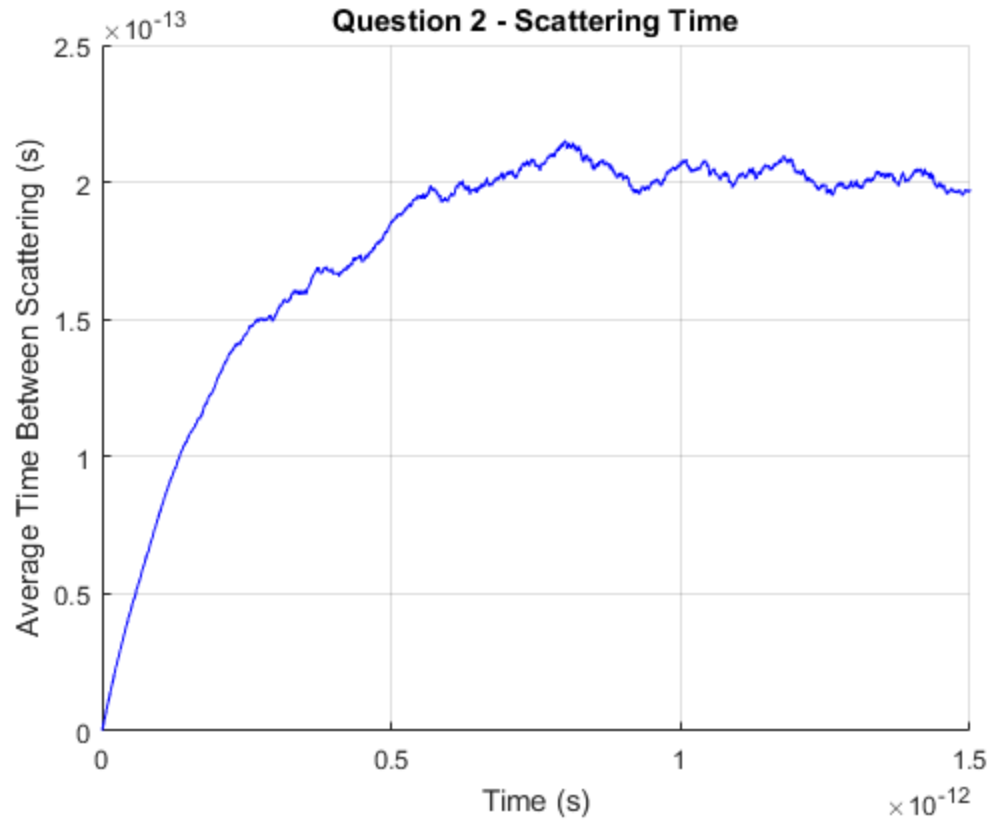


The temperature has some uncertainty but remains on average around 300K based upon the Gaussian Distribution in which we continue to reassign values from this distribution.

## D) Mean Free Path and Mean Time Between Collisions

```
figure(6);
hold on;
plot(0:Timestep:endtime, ScatterTime(:, 1), 'b');
title('Question 2 - Scattering Time');
xlabel('Time (s)');
ylabel('Average Time Between Scattering (s)');
grid on;
hold off;

figure(7);
hold on;
plot(0:Timestep:endtime, MFP(:, 1), 'b');
title('Question 2 - Mean Free Path');
xlabel('Time (s)');
ylabel('Average Distance Between Scattering (m)');
grid on;
hold off;
```



The graph shows the mean free path stabilizing around 36 to 38 nm and with a calculated mean free path of 37.4 nm the simulation and calculated agree. The scattering time approaches 0.2 ps and stabilizes around the given time between collisions of 0.2 ps. Both these plots become asymptotic at the given and calculated values for mean free path and time between collisions.

## Question 3

### A) Adding A Bottle Neck

The Graph Below shows the Electron Position with time and the rectangular bottle neck that occurs with it.

```
%Assign the Box Height and Width as Percentages of the Limits
xboxLim = [.4*(xlims(2)-xlims(1)), .6*(xlims(2)-xlims(1))];
yboxLim1 = [ylims(1), .4*(ylims(2)-ylims(1))];
yboxLim2 = [.6*(ylims(2)-ylims(1)), ylims(2)];
xbox = xboxLim([1 1 2 2 1]);
ybox1 = yboxLim1([1 2 2 1 1]);
ybox2 = yboxLim2([1 2 2 1 1]);
endtime = Timestep*5000;
xprev = zeros(1, NumParticles);
yprev = zeros(1, NumParticles);
x = zeros(3, NumParticles);
y = zeros(2, NumParticles);
temp = zeros(2, NumParticles);
scatTime = zeros(1, NumParticles);

%Start the random distribution in x position
x(1,:) = rand(1, NumParticles);
y(1,:) = rand(1, NumParticles);
x(1,:) = xlims(1) + x(1,:).*(xlims(2) - xlims(1));
y(1,:) = ylims(1) + y(1,:).*(ylims(2) - ylims(1));

%Get the Particle Indexes that are out of bounds
IDX = uint32(1:NumParticles);
ind = IDX((y(1,:) >= yboxLim2(1) | y(1,:) <= yboxLim1(2)) & x(1,:) >=
    xboxLim(1) & x(1,:) <= xboxLim(2));

%Reassign Positions
counter = 1;
%size(ind, 2)
while counter <= size(ind, 2)
    x(1, ind(counter)) = xlims(1) + rand*(xlims(2) - xlims(1));
    y(1, ind(counter)) = ylims(1) + rand*(ylims(2) - ylims(1));
    if ~(y(1, ind(counter)) >= yboxLim2(1) | y(1, ind(counter)) <=
        yboxLim1(2)) && x(1, ind(counter)) >= xboxLim(1) && x(1, ind(counter))
        <= xboxLim(2))
        counter = counter + 1;
    end
end

%Assign the random velocity and random angle
temp(1,:) = (normrnd(0, sqrt(mn/(Kbolt*T)), 1, NumParticles).^2 +
    normrnd(0, sqrt(mn/(Kbolt*T)), 1, NumParticles).^2).^5;
```

```
x(3,:) = rand(1, NumParticles)*2*pi;
x(2,:) = temp(1,:).*cos(x(3,:));
y(2,:) = temp(1,:).*sin(x(3,:));
```

## B) Boundaries Specular or Diffusive

```
DiffusionBarrierProbability = 0.2;

figure(9);
title('Question 3 - Electron Position');
xlabel('X Position (m)');
ylabel('Y Position (m)');
xlim([xlims(1), xlims(2)]);
ylim([ylims(1), ylims(2)]);
hold on;
rectangle('Position', [xboxLim(1), yboxLim1(1), xboxLim(2)-xboxLim(1),
    yboxLim1(2)]);
rectangle('Position', [xboxLim(1), yboxLim2(1), xboxLim(2)-xboxLim(1),
    yboxLim2(2) - yboxLim2(1)]);

counter = 1;
for i = 0:Timestep:endtime
    % pause(.1);
    xprev(1,:) = x(1,:);
    yprev(1,:) = y(1,:);
    for kt = 1:NumParticles

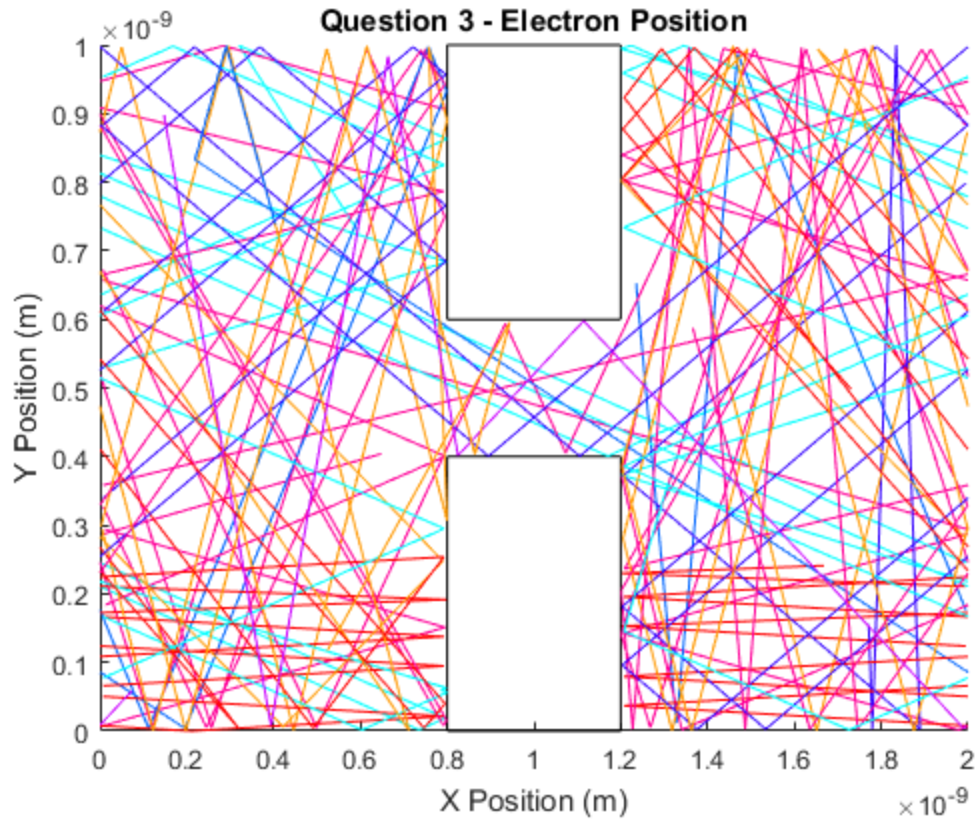
        %PARTICLE'S HITTING TOP and
        if x(1,kt) + x(2,kt)*Timestep < xlims(1)
            x(1,kt) = xlims(2);
            xprev(1,kt) = xlims(2);
        elseif x(1,kt) + x(2,kt)*Timestep > xlims(2)
            x(1,kt) = xlims(1);
            xprev(1,kt) = xlims(1);
        end
        if y(1,kt) + y(2,kt)*Timestep < ylims(1) || y(1,kt) +
y(2,kt)*Timestep > ylims(2)
            if rand > DiffusionBarrierProbability
                y(2,kt) = -y(2,kt);
            else
                if y(1,kt) + y(2,kt)*Timestep < ylims(1)
                    x(3,kt) = rand*pi;
                elseif y(1,kt) + y(2,kt)*Timestep > ylims(2)
                    x(3,kt) = -rand*pi;
                end
                temp = (normrnd(0, sqrt(Kbolt*T/mn))^2 + normrnd(0,
sqrt(Kbolt*T/mn))^2)^.5;
                x(2,kt) = temp*cos(x(3,kt));
                y(2,kt) = temp*sin(x(3,kt));
            end
        end

        %PARTICLE'S HITTING BOX
```

```
        if (y(1,kt) + y(2,kt)*Timestep >= yboxLim2(1) && x(1,kt)
+ x(2,kt)*Timestep >= xboxLim(1) && x(1,kt) + x(2,kt)*Timestep <=
xboxLim(2))
            [xinter1, xinter2, yinter1, yinter2] =
BoxIntercept( x(1,kt), y(1,kt), x(1,kt) + x(2,kt)*Timestep, y(1,kt)
+ y(2,kt)*Timestep, xboxLim(1), xboxLim(2), yboxLim2(1), ylimits(2));
            if rand > DiffusionBarrierProbability
                if xinter2 || xinter1
                    x(2,kt) = - x(2,kt);
                end
                if yinter2 || yinter1
                    y(2,kt) = - y(2,kt);
                end
            else
                if xinter1
                    x(3,kt) = pi/2 + rand*pi;
                elseif xinter2
                    x(3,kt) = pi/2 - rand*pi;
                elseif yinter2
                    x(3,kt) = rand*pi;
                elseif yinter1
                    x(3,kt) = -rand*pi;
                end
                temp = (normrnd(0, sqrt(Kbolt*T/mn))^2 +
normrnd(0, sqrt(Kbolt*T/mn))^2)^.5;
                x(2,kt) = temp*cos(x(3,kt));
                y(2,kt) = temp*sin(x(3,kt));
            end
            elseif y(1,kt) + y(2,kt)*Timestep <= yboxLim1(2) && x(1,kt)
+ x(2,kt)*Timestep >= xboxLim(1) && x(1,kt) + x(2,kt)*Timestep <=
xboxLim(2)
                [xinter1, xinter2, yinter1, yinter2] =
BoxIntercept( x(1,kt), y(1,kt), x(1,kt) + x(2,kt)*Timestep,
y(1,kt) + y(2,kt)*Timestep, xboxLim(1), xboxLim(2), ylimits(1),
yboxLim1(2));
                if rand > DiffusionBarrierProbability
                    if xinter1 || xinter2
                        x(2,kt) = - x(2,kt);
                    end
                    if yinter1 || yinter2
                        y(2,kt) = - y(2,kt);
                    end
                else
                    if xinter1
                        x(3,kt) = pi/2 + rand*pi;
                    elseif xinter2
                        x(3,kt) = pi/2 - rand*pi;
                    elseif yinter2
                        x(3,kt) = rand*pi;
                    elseif yinter1
                        x(3,kt) = -rand*pi;
                    end
                    temp = (normrnd(0, sqrt(Kbolt*T/mn))^2 + normrnd(0,
sqrt(Kbolt*T/mn))^2)^.5;
```

```
x(2,kt) = temp*cos(x(3,kt));
y(2,kt) = temp*sin(x(3,kt));
    end
end

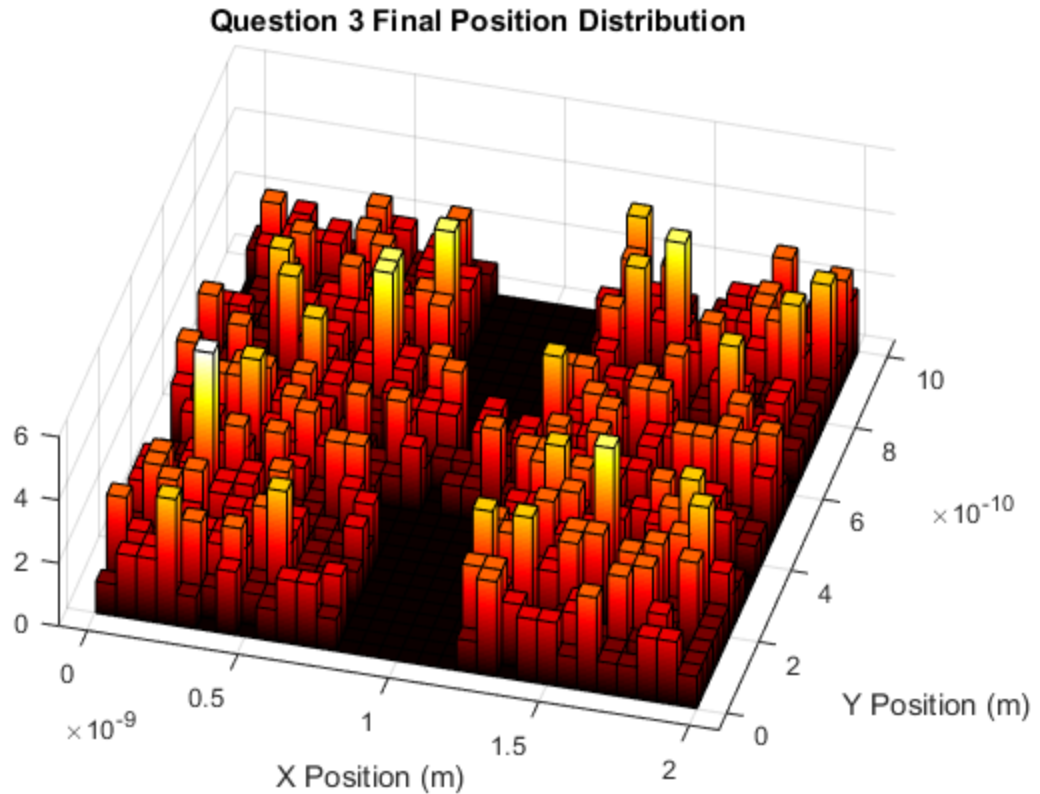
%UPDATE POSITION
x(1,kt) = x(1,kt) + x(2,kt).*Timestep;
y(1,kt) = y(1,kt) + y(2,kt).*Timestep;
%PLOT
if kt <= PlotHowMany
    plot([xprev(1,kt), x(1,kt)], [yprev(1,kt), y(1,kt)], 'color',
mycolors(kt,:) );
    hold on;
end
%Scattering Check
if Pscat>rand()
    x(3,kt) = rand*2*pi;
    temp = (normrnd(0, sqrt(Kbolt*T/mn))^2 + normrnd(0,
sqrt(Kbolt*T/mn))^2)^.5;
    x(2,kt) = temp*cos(x(3,kt));
    y(2,kt) = temp*sin(x(3,kt));
    scatTime(1,kt) = 0;
end
end
hold on;
AvgScat = mean(scatTime(1,:));
scatTime(1,:) = scatTime(1,:) + Timestep;
averagevel = mean((x(2,:).^2 + y(2,:).^2).^5);
    %title(['Average Temperature: ' num2str(averagevel) ' Average
Scatter Time: ' num2str(AvgScat)]);
    figure(9);
end
```



## C) Electron Density Map

```
figure(10);

Z = [transpose(x(1,:)), transpose(y(1,:))];
hist3(Z, [30,30]);
hold on;
set(get(gca, 'child'), 'FaceColor', 'interp', 'CDataMode', 'auto');
colormap(hot);
view(15, 65); % heat map
grid on;
title('Question 3 Final Position Distribution');
xlabel('X Position (m)');
ylabel('Y Position (m)');
hold off;
```

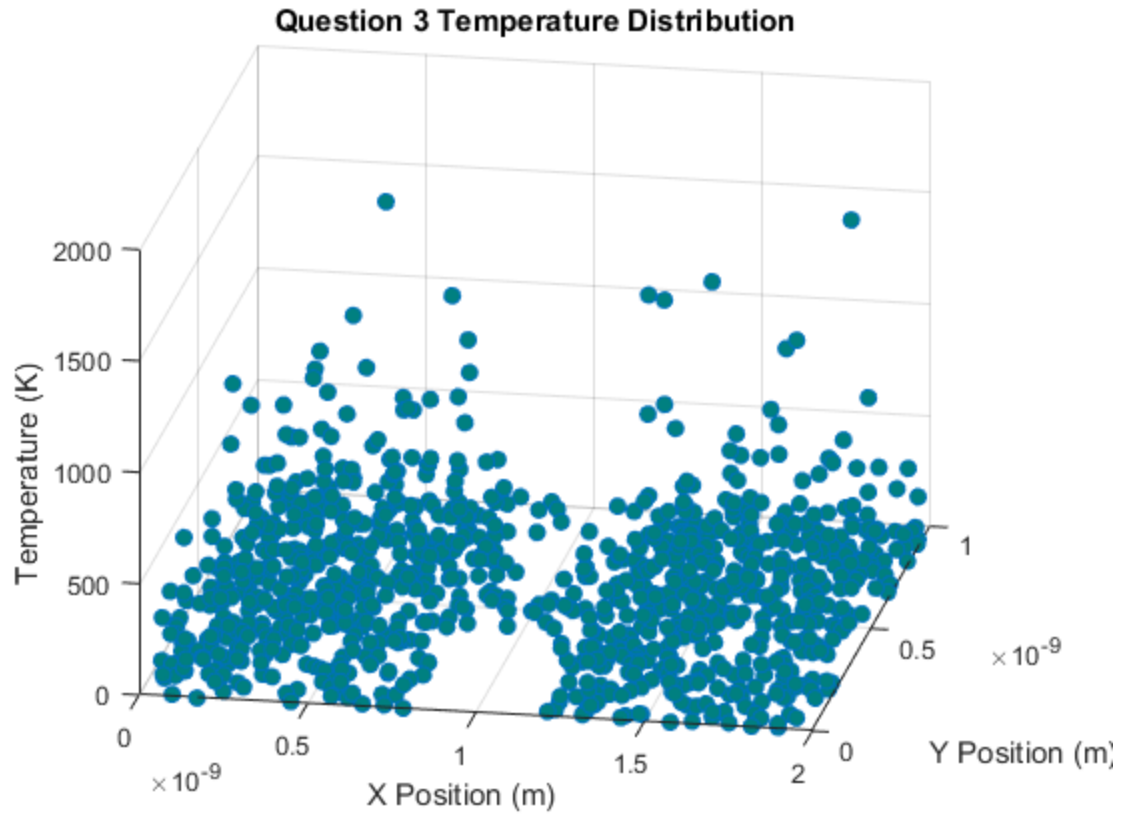


From the figure above we can see that no electrons are present in the rectangular boxes.

## D) Temperature Map

```
figure(11);
VelSquared = (x(2,:).^2 + y(2,:).^2);
CalcTemp = VelSquared.*mn./2./Kbolt;
h = scatter3(x(1,:), y(1,:), CalcTemp(1,:));
h.MarkerFaceColor = [0 0.5 0.5];
grid on;
xlabel('X Position (m)');
ylabel('Y Position (m)');
xlim([xlims(1), xlims(2)]);
ylim([ylims(1), ylims(2)]);
zlabel('Temperature (K)');
title('Question 3 Temperature Distribution');
view(10,25);
hold off;
```





Again from the figure above we can see that no electrons are present in the rectangular boxes.

*Published with MATLAB® R2017a*