Contents

Beginning of Assignment 1

```
%I'm not sure if I was supposed to do this report completely
%in the publishing tool. I have a LaTeX document attached that has
%more information and graphs on it
%(graphs that would show up if part2 or part3 were set to false below)
clear
close all

%Which parts are included in the run
part2 = true;
part3 = true;
specular = true;%boundaries for part 3 boxes are specular, if false boxes are diffusive
```

```
m_e = 9.1093837015e-31;%rest mass in kg
m_n = 0.26*m_e;%Effective mass of electrons

%Nominial size of the region in nm
xlimit = [0 200e-9];
ylimit = [0 100e-9];
```

Part 1: Electron Modelling

Part 1: Question 1: What is the thermal velocity?

```
kb=1.380649e-23;%Boltzmann's constant [m^2*kg/K*s^2]
T = 300;%Temperature in Kelvin
vth = sqrt((3*kb*T)/m_n);
```

Part 1: Question 2: What is the mean free path?

```
tmn = 0.2e-12;%mean time between collisions
MFP = vth*tmn;
```

Part 1: Question 3 Setting initial values of particles

```
%Amount of particles (total or plotted particles[plots first X particles])
particles = 10000;
plottedparticles = 7;

timesteps = 1000;
dt=2e-15;%time step
tr=timesteps*dt;%runtime

%Place particles in random locations
x = zeros([1 particles]);
y = zeros([1 particles]);
for n=1:particles
    x(n) = xlimit(1) + (xlimit(2)-xlimit(1)).*rand;
    y(n) = ylimit(1) + (ylimit(2)-ylimit(1)).*rand;
    if part3 %don't place particles in part 3 boxes
        while (x(n) >= 0.8e-7 && x(n) <= 1.2e-7 && y(n) <= 0.4e-7)...
                || (x(n) >= 0.8e-7 && x(n) <= 1.2e-7 && y(n) >= 0.6e-7)
            x(n) = xlimit(1) + (xlimit(2)-xlimit(1)).*rand;
            y(n) = ylimit(1) + (ylimit(2)-ylimit(1)).*rand;
        end
    end
end
```

Part 2: Question 1: Give each particle a random velocity

```
Rx = (vth).*randn(particles,1) + vth;%Where the average is vth and the variance is vth/4
Ry = (vth).*randn(particles,1) + vth;
```

Part 1: Question 3 Continued: Setting initial values of particles

```
Vx = zeros([1 particles]);
Vy = zeros([1 particles]);
%Get random normalized angles
for n = 1:particles
    angX = cos((2*rand-1)*2*pi);
    angY = sin((2*rand-1)*2*pi);
    normalized = [angX angY]./norm([angX angY]);
    if part2
        Vx(n)= Rx(n)*normalized(1);%Random velocity Rx is used for part 2
        Vy(n)= Ry(n)*normalized(2);
    else
        Vx(n)= vth*normalized(1);%part 1 velocity has magnitude v_thermal
        Vy(n)= vth*normalized(2);
    end

end
```

Part 1: Question 3 Simulation of the random motion of the electrons after time t

```
%Initialize variables used for storage for plotting later
n=1;
Px = zeros([timesteps particles]);
Py = zeros([timesteps particles]);
Temp = zeros([1 timesteps]);
Time = zeros([1 timesteps]);
```

Part 2 Question initialization of variables for MFP and time between collisions

```
numberOfCollisions = zeros([1 particles]);
meanTimeBetweenCollisions = zeros([1 particles]);
meanFreePath = zeros([1 particles]);
noCollisionYet = true([1 particles]);
```

General particle movement for all parts

```
for t = 0:dt:tr
```

Part 1: Question 3 Particle location update

```
    %Store Location
    for p = 1:particles
        Px(n, p)= x(p);
        Py(n, p)= y(p);
    end
```

Part 2: Question 2 Scattering probabilities

```
    if part2
        Pscat = 1-exp(-(dt)/(tmn));%The scattering probability
        for p = 1:particles
            if Pscat > rand
                Vscatter(p) = sqrt(Vx(p)^2+Vy(p)^2);%Velocity of the particle when it scatter
s

                %Give new random velocities and directions
                angX = cos((2*rand-1)*2*pi);
                angY = sin((2*rand-1)*2*pi);
                normalized = [angX angY]./norm([angX angY]);
                Rx = (vth).*randn + vth;
                Ry = (vth).*randn + vth;
                Vx(p) = Rx*normalized(1);
                Vy(p) = Ry*normalized(2);

                %Find the time and distance since the last collision
                if noCollisionYet(p)
                    noCollisionYet(p) = false;
                else
                    numberOfCollisions(p) = numberOfCollisions(p) + 1;
                    meanTimeBetweenCollisions(p) = (meanTimeBetweenCollisions(p) + (t - lastC
ollisionTime(p)))/ numberOfCollisions(p);

                end
                lastCollisionTime(p) = t;
            end
        end
    end
```

Part 1: Question 3 Particle outer boundaries

```
    %Check boundaries Outer Boundaries
    for p = 1:particles
        if (y(p)+ Vy(p)*dt) <= ylimit(1) || (y(p)+Vy(p)*dt) >= ylimit(2)
            Vy(p) = -1*Vy(p);
        end
        if x(p) <= xlimit(1)
            x(p) = x(p) + xlimit(2);
        elseif x(p) >= xlimit(2)
            x(p) = x(p) - xlimit(2);
        end
    end
```

Part 3: Question 1 Boundary boxes

```
    %Check Box Boundaries
    if part3

        %Specular boundaries
        if specular
            for p = 1:particles
                   %if the next position of the particle is in the box
```

```
                %change the particle's direction
                if ((x(p) + Vx(p)*dt) >= 0.8e-7 && (x(p) + Vx(p)*dt) <= 1.2e-7 &&...
                        (y(p) + Vy(p)*dt <= 0.4e-7 || y(p) + Vy(p)*dt >= 0.6e-7))
                    %if the position of the particle is currently between
                    %the two boxes reflect in the y-direction
                    if x(p) >= 0.8e-7 && x(p) <= 1.2e-7
                        Vy(p) = -1*Vy(p);
                    %otherwise reflect in the x-direction
                    else
                        Vx(p) = -1*Vx(p);
                    end
                end
            end
        %Diffusive boundaries
        else
            for p = 1:particles
                %if the next position of the particle is in the box
                %change the particle's direction
                if ((x(p) + Vx(p)*dt) >= 0.8e-7 && (x(p) + Vx(p)*dt) <= 1.2e-7 &&...
                        (y(p) + Vy(p)*dt <= 0.4e-7 || y(p) + Vy(p)*dt >= 0.6e-7))
                    %if the position of the particle is currently between
                    %the two boxes reflect in the y-direction with a
                    %random x velocity
                    if x(p) >= 0.8e-7 && x(p) <= 1.2e-7
                        angX = cos((2*rand-1)*2*pi);
                        angY = sin((2*rand-1)*2*pi);
                        normalized = [angX angY]./norm([angX angY]);
                        Rx = (vth).*randn + vth;
                        Vx(p) = Rx*normalized(1);
                        Vy(p) = -1*Vy(p);
                    %otherwise reflect in the x-direction with a random
                    %y velocity
                    else
                        angX = cos((2*rand-1)*2*pi);
                        angY = sin((2*rand-1)*2*pi);
                        normalized = [angX angY]./norm([angX angY]);
                        Ry = (vth).*randn + vth;
                        Vx(p) = -1*Vx(p);
                        Vy(p) = Ry*normalized(2);
                    end
                end
            end
        end
    end
```

Question 1 Part 3: Advance the particle's location and find the temp

```
    %Advance Location
    distance = Vx*dt;
    x= x + Vx*dt;
    y= y + Vy*dt;
    %Calculate the temperature at each time step and store the values of
    %Temp and Time
    Temp(n) = (2/3)*(1/kb)*(1/2)*m_n*(sum(Vx.^2+Vy.^2))/p;
    Time(n) = t;
```

Part 3 Question 4: Temperature map

Proceed to next column of matrix that plots particle trajectories for data storage

```
    n=n+1;%New column in matrix for each timestep
```

```
end
```

Part 2: Question 4 Mean free path and mean time between collisions

```
measuredMeanTimeBetweenCollisions = sum(meanTimeBetweenCollisions)/particles
measuredMeanFreePath = sum(measuredMeanTimeBetweenCollisions*Vscatter)/particles
```

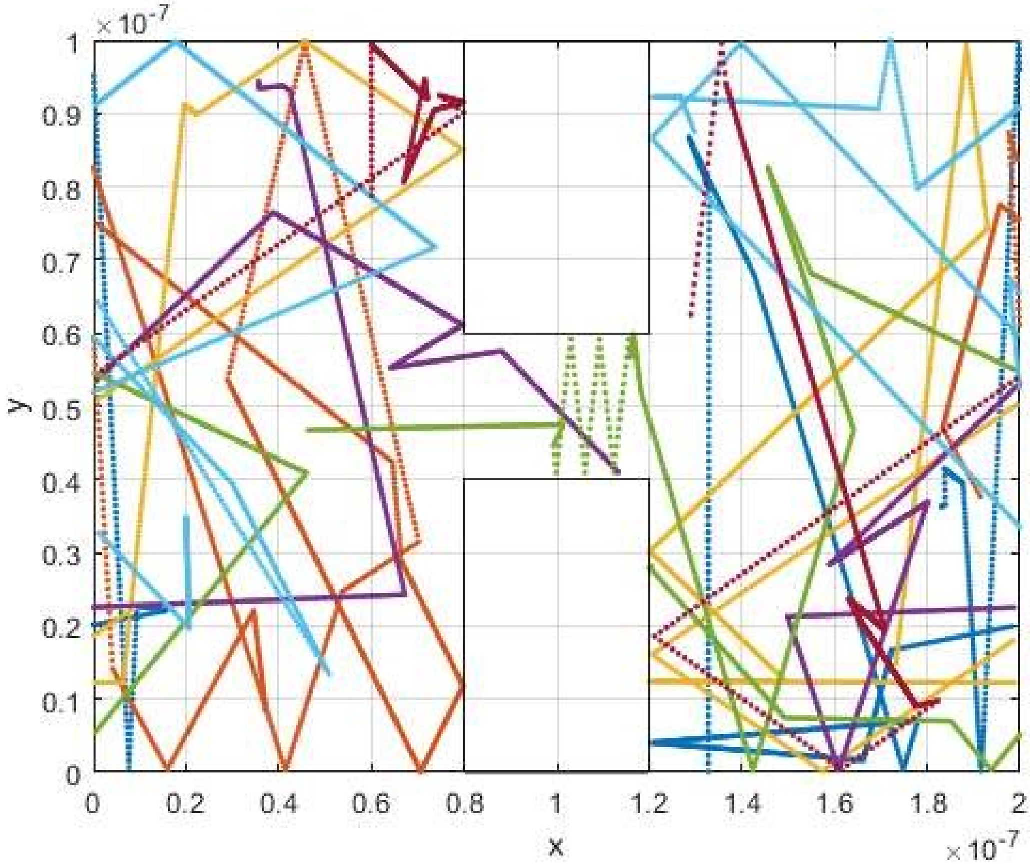
```
measuredMeanTimeBetweenCollisions =

   4.2276e-14


measuredMeanFreePath =

   1.2015e-08
```

Part 1: Question 3 i) Plot Trajectories

```
s = size(Px);
figure;
for N = 1:s(1)
    for pl = 1:plottedparticles
        plot(Px(1:N, pl),Py(1:N, pl),'.');
        hold on
    end
    hold off
    xlim(xlimit)
    ylim(ylimit)
    xlabel('x');
    ylabel('y');
    grid on
    if part3
```

Part 3: Enhancements Question 1 Boundary boxes

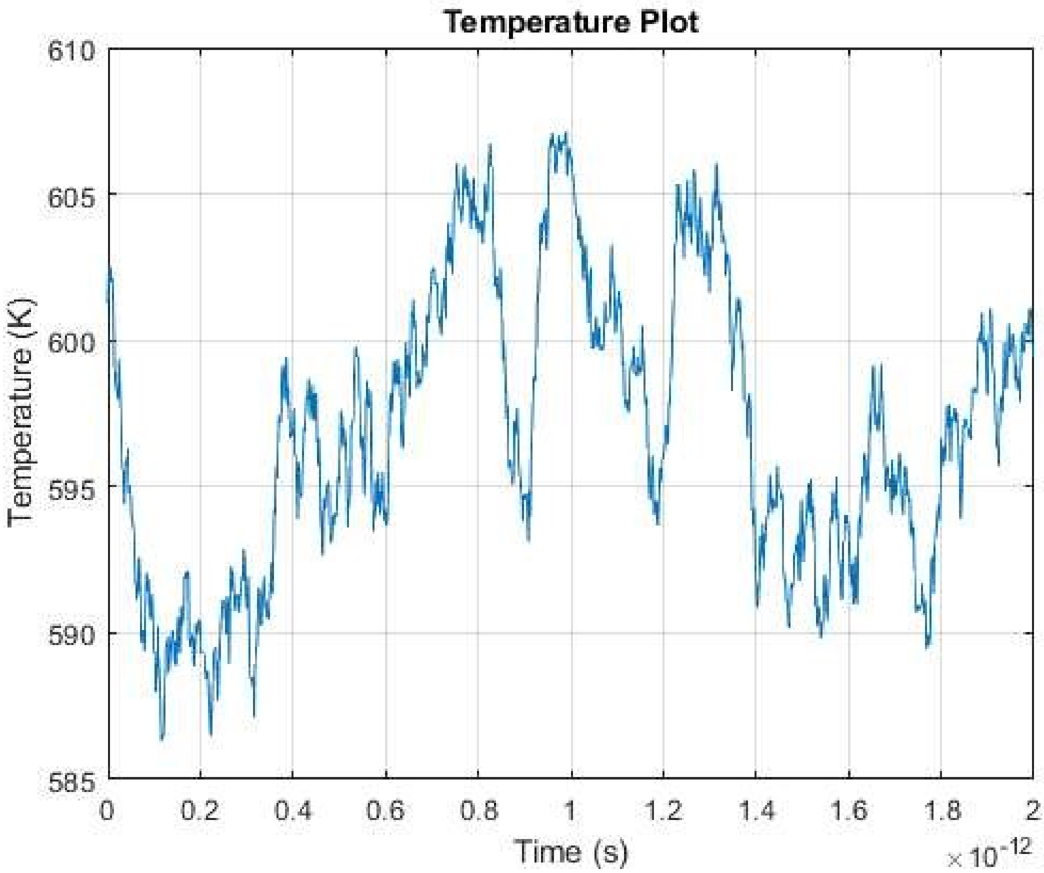
```
        rectangle('Position',[0.8e-7 0 0.4e-7 0.4e-7])
        rectangle('Position',[0.8e-7 0.6e-7 0.4e-7 0.4e-7])
```




```
    end
    pause(0.001)
end
```

P1: Question 3 ii) Temperature plot

```
figure;
plot(Time,Temp,'-')
xlabel('Time (s)');
ylabel('Temperature (K)');
title('Temperature Plot')
grid on
```

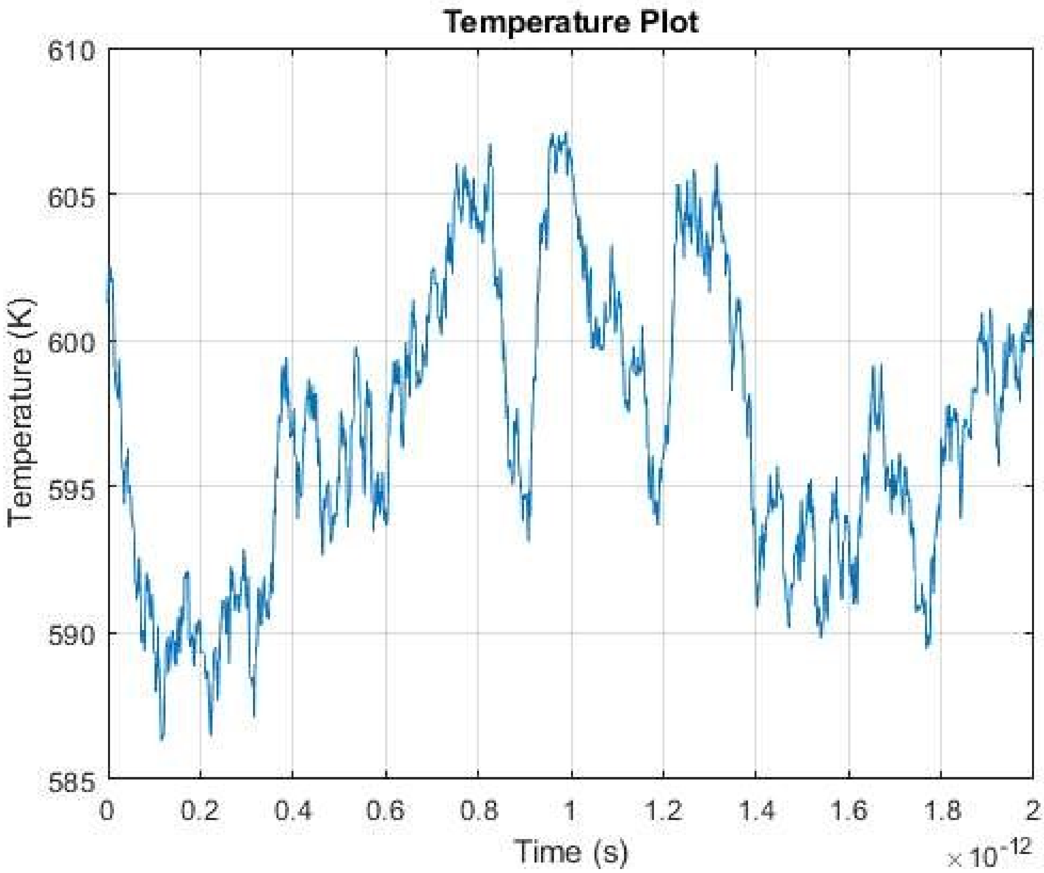


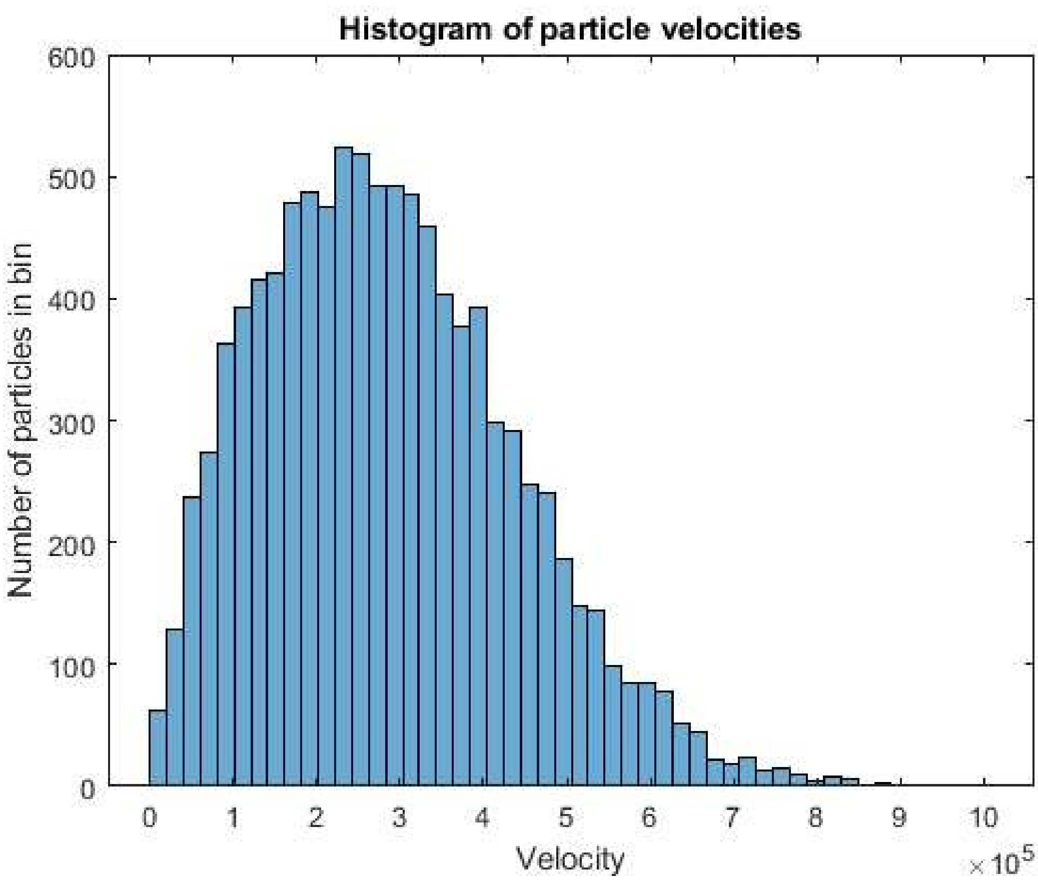
Part 2: Collisions with Mean Free Path

```
%Velocity from part 1 is now assigned random velocities using
%Maxwell-Boltzmann, average speed should be vth
```

Part 2: Question 1 Histogram

```
V = sqrt(Vx.^2+Vy.^2);%get the velocity of each particle
figure;
histogram(V,50)%Plot histogram of Boltzmann distribution of velocities
xlabel('Velocity');
ylabel('Number of particles in bin');
title('Histogram of particle velocities')
```

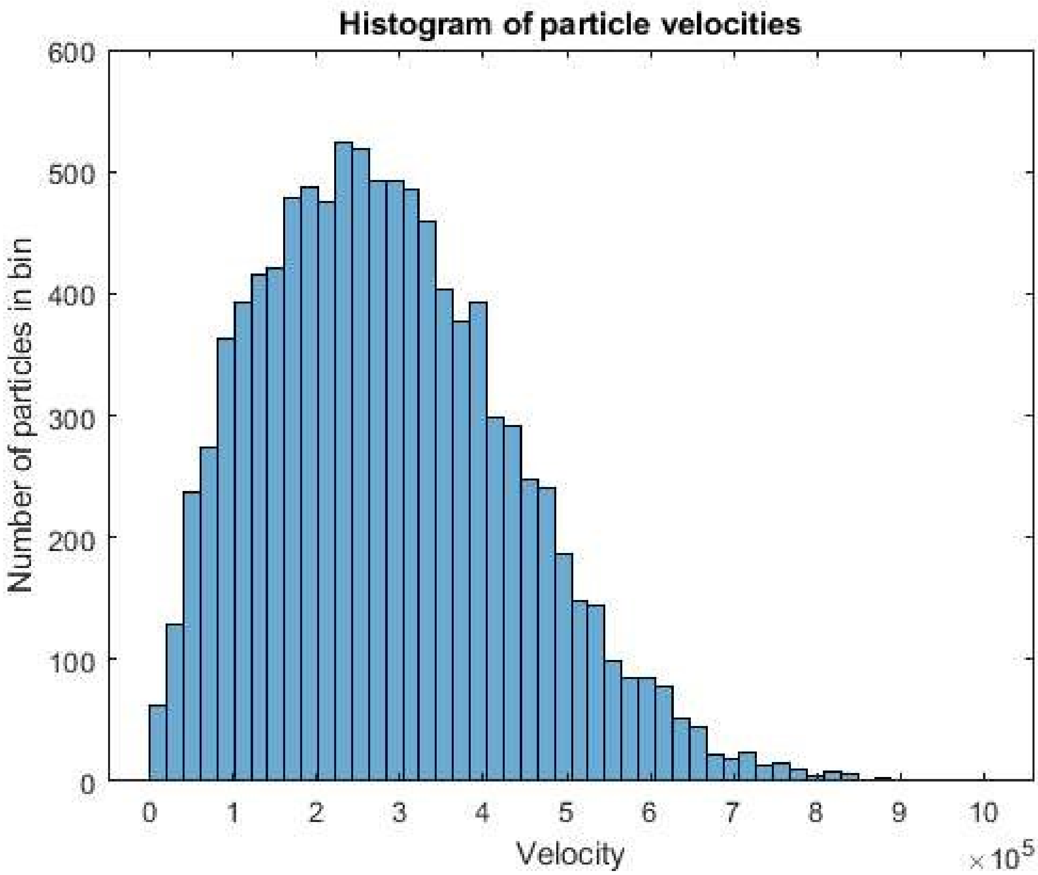


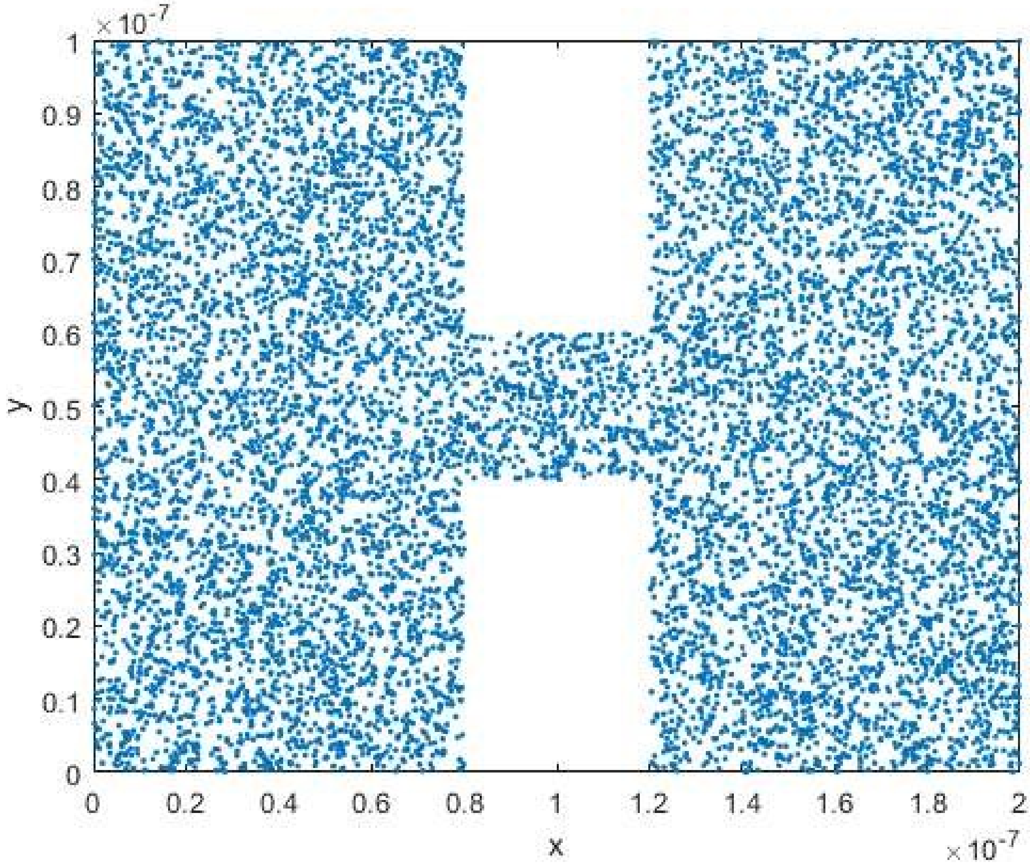


Part 3: Enhancements

Part 3 Question 3: Electron Density map

```
[q,r] = size(Px);%dimension q is the final index of the timesteps for each particle
figure;
plot(Px(q,1:particles),Py(q,1:particles),'.');%position P(for last timestep, showing all part
icles)
xlim(xlimit)
ylim(ylimit)
xlabel('x');
ylabel('y');
```





Part 3 Question 4: Temperature map

End of assignment

Published with MATLAB® R2019b