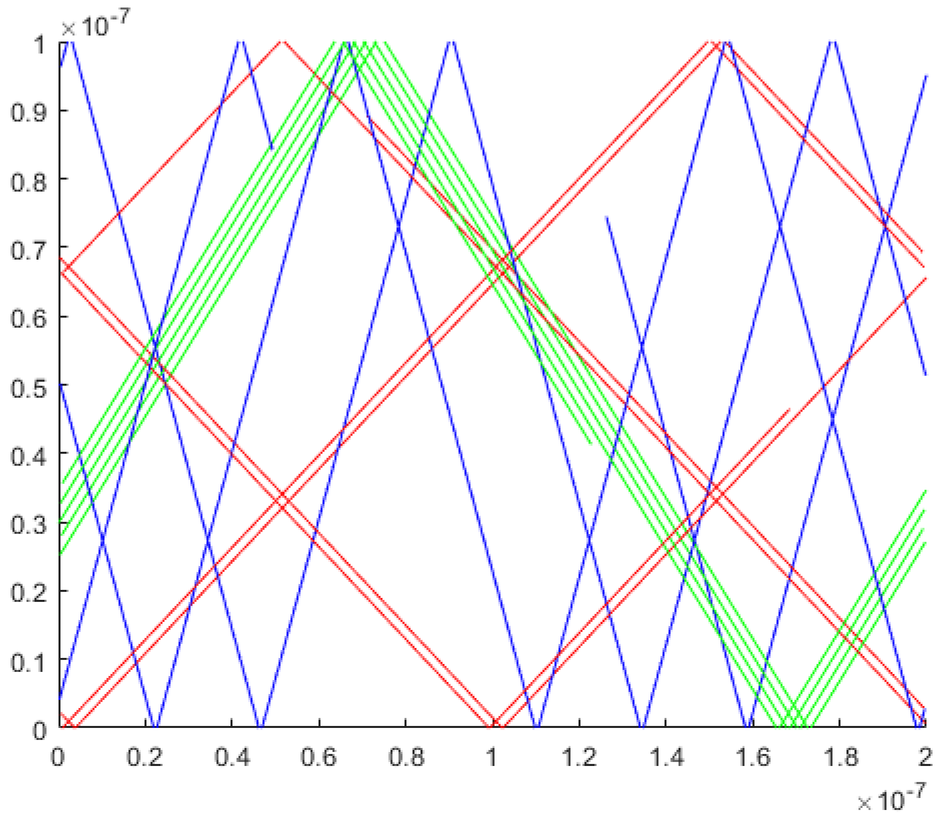
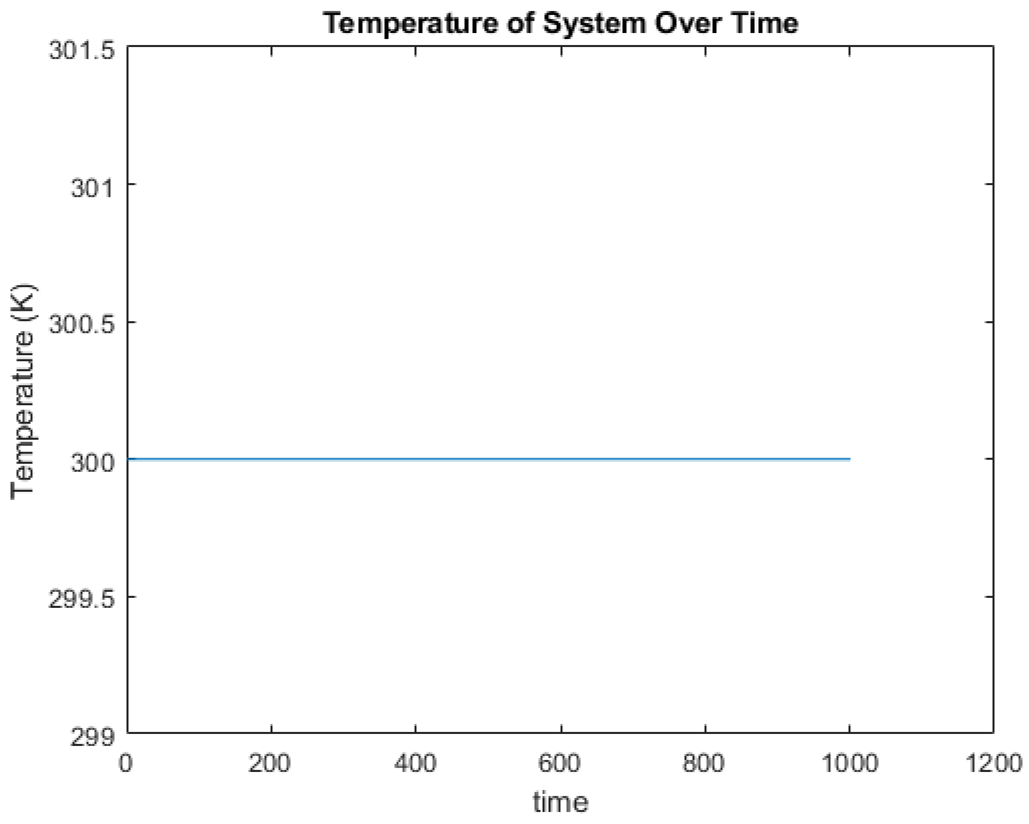
Section 1 - Electron Modelling

modelling the motion of these electrons was both interesting and difficult. these electrons start with a constant equal velocity, and it never changes throughout the simulation. For this reason, the temperature of the system never changes. the only way a particle can change direction in this simulation is to bounce off the bottom or top wall. The reflection from the top only reverses the angle, so the particle exhibits periodic motion in a general direction based on its initial angle. I spent a great deal of time debugging the plotting. I found that the particles would leave an ugly, unintended horizontal trail behind them when they passed through the right and left walls. Once that was fixed, the rest of the program came together quickly. This program could definitely make use of speed improvements, especially when the simulation lasts for lots of time steps. The thermal velocity is 1.3224e+5 m/s. the mean free path is 4.85e-8 m.

```
%constants
clear
C.q_0 = 1.60217653e-19;
C.m_0 = 9.10938215e-31;
C.kb = 1.3806504e-23;
C.T = 300;
frameWidth = 200e-9;
frameHeight = 100e-9;
Vth = sqrt(C.kb * C.T / (0.26*C.m_0));
iteration = 1;
Tstop = 1e-11;
t = 0;
dt = 1e-14;

%initialization of vectors
nAtoms = 1000;
Xnext = zeros(1,nAtoms);
Ynext = zeros(1,nAtoms);
direction = 2*pi*rand(1, nAtoms);
VX = Vth * cos(direction);
VY = Vth * sin(direction);
X = frameWidth * rand(1, nAtoms);
Y = frameHeight * rand(1, nAtoms);
Temperature = zeros(1, 100);
%electrons placed randomly, with uniform velocity and random
 directions

while t < Tstop
    %for each cycle in time, move electrons in accordance with their
    %velocities
    Xnext = X + VX*dt;
    Ynext = Y + VY*dt;
    %X boundary conditions set
    right = Xnext>frameWidth;
    left = Xnext<0;
    Xnext(right) = Xnext(right)-frameWidth;
    Xnext(left) = Xnext(left) + frameWidth;
    %Y boundary conditions set
    top = Ynext > frameHeight;
    bottom = Ynext < 0;
    VY(top | bottom) = VY(top | bottom) * -1;
    %calculation for temperature
    V = sqrt(VX.*VX+VY.*VY);
    Temperature(iteration) = 0.26*C.m_0*mean(V.^2)/C.kb;
    figure(1)
    xlim([0 frameWidth])
    ylim([0 frameHeight])
    hold on
    %plotting, but avoid plotting the full horizontal jump
    if abs(Xnext(1) - X(1)) < 2*abs(VX(1))*dt
        figure(1)
        plot([Xnext(1) X(1)], [Ynext(1) Y(1)], 'blue')
    end
    if abs(Xnext(2) - X(2)) < 2*abs(VX(2))*dt
        figure(1)
        plot([Xnext(2) X(2)], [Ynext(2) Y(2)], 'red')
    end
    if abs(Xnext(3) - X(3)) < 2*abs(VX(3))*dt
        figure(1)
        plot([Xnext(3) X(3)], [Ynext(3) Y(3)], 'green')
    end

    %updating positions, and advancing time a step forward so the
 while
    %loop works
    X = Xnext;
    Y = Ynext;
    t = t+dt;
    iteration = iteration + 1;
    pause(0.0001);
end
%plotting temperature, should be constant
figure(2)
dummy = linspace(0,iteration, length(Temperature));
plot(dummy, Temperature)
title('Temperature of System Over Time')
xlabel('time')
ylabel('Temperature (K)')
```


×10⁻⁷
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
×10⁻⁷


Temperature of System Over Time
Temperature (K)
301.5
301
300.5
300
299.5
299
0
200
400
600
800
1000
1200
time

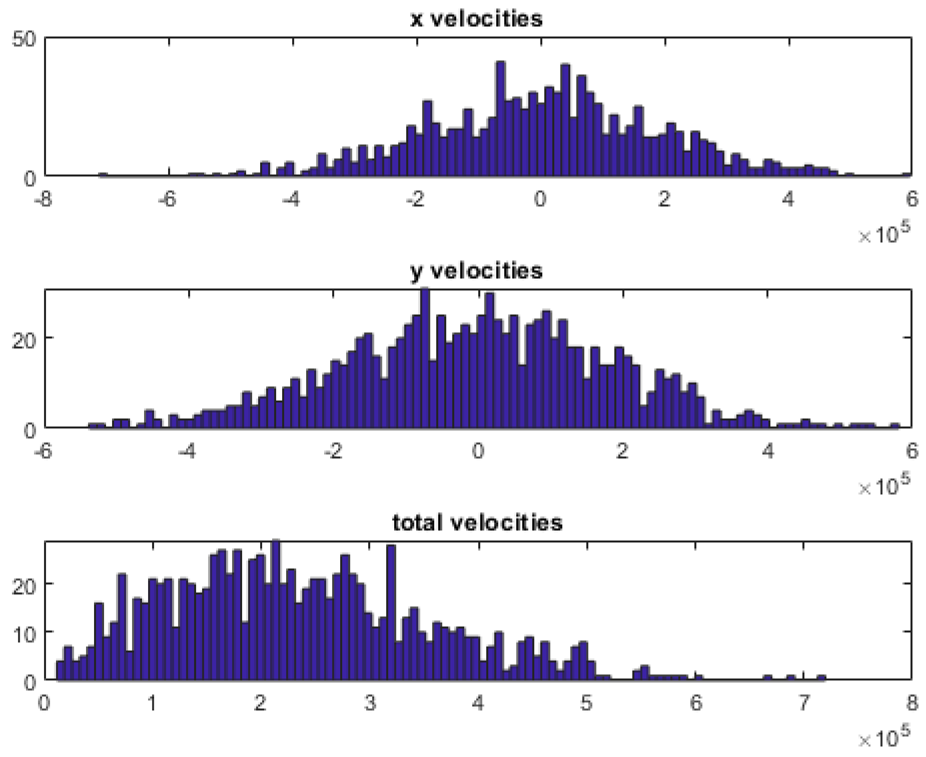
Section 2 - Collisions with Mean Free Path (MFP)

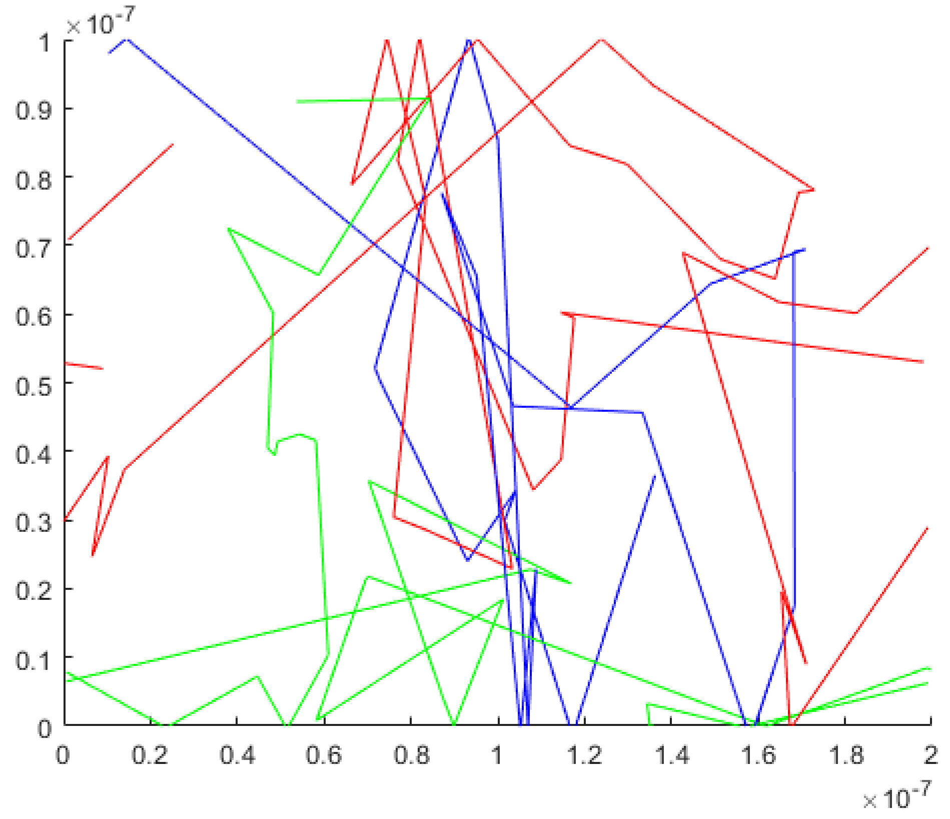
I found this section to be considerably easier than the previous, since most of the hard work was already done from the previous section. Adding the scattering was fairly easy, and I already had the temperature and trajectory plots working. Measuring the mean free path and mean time between collisions was also fairly straightforward. Adding the histograms for the velocity distributions wasn't hard, and I got the results I expected. The average speed was indeed the thermal voltage, and the x and y velocities were properly normally distributed.

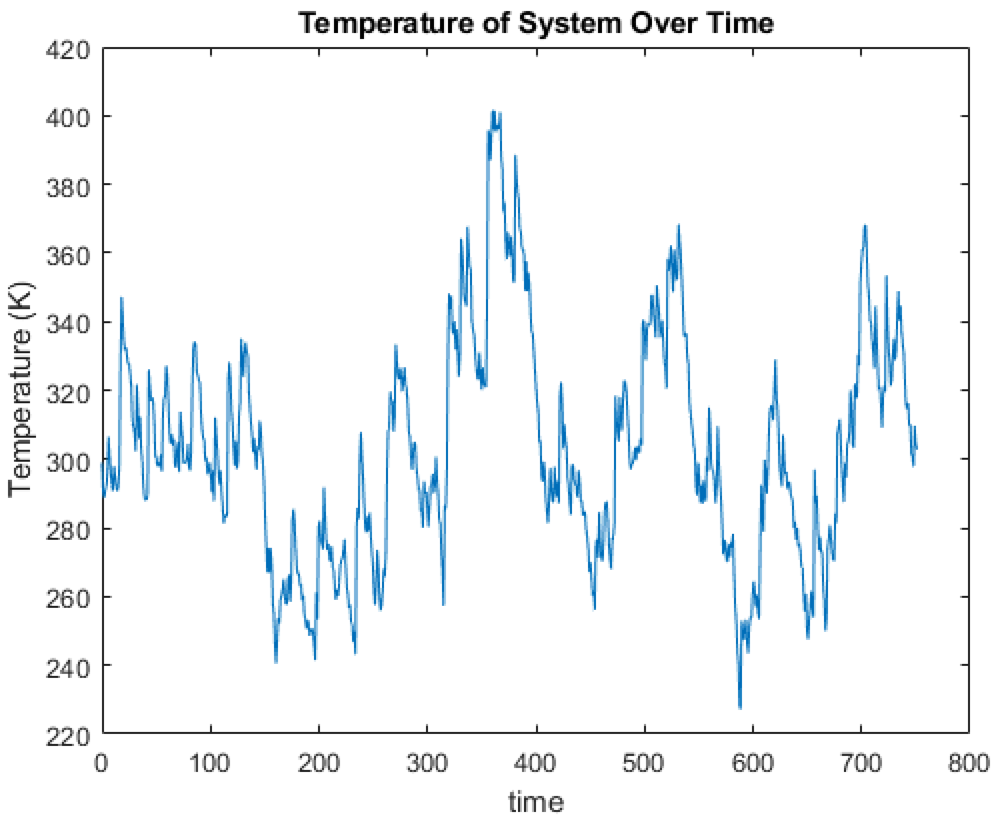
```
%constants
clear
C.q_0 = 1.60217653e-19;
C.m_0 = 9.10938215e-31;
C.kb = 1.3806504e-23;
C.T = 300;
frameWidth = 200e-9;
frameHeight = 100e-9;
nAtoms = 1000;
bins = nAtoms / 10;
Vth = sqrt(2*C.kb*C.T /(0.26*C.m_0));
dt = frameHeight/Vth/100;
Tstop = 750*dt;
t = 0;
freepath = 0.2e-12;
Pscatter = 1 - exp(-dt/freepath);

%initializing vectors
Xnext = zeros(1,nAtoms);
Ynext = zeros(1,nAtoms);
VX = Vth * randn(1,nAtoms);
VY = Vth * randn(1,nAtoms);
V = sqrt(VY.*VY+VX.*VX);
X = frameWidth * rand(1, nAtoms);
Y = frameHeight * rand(1, nAtoms);
R = zeros(1, nAtoms);
Temperature = zeros(1, 100);
meanpaths = zeros(1,nAtoms);
iteration = 1;
%histograms of X,Y, and overall velocities
figure(3)
subplot(3,1,1);
hist(VX,bins)
title('x velocities')
subplot(3,1,2);
hist(VY,bins)
title('y velocities')
subplot(3,1,3);
hist(V,bins);
title('total velocities')
```

```
while t < Tstop
    %determines which particles scatter and performs calculations on
 them
    %to determine mean free path and time between collisions
    R = rand(1,nAtoms);
    VX(R<Pscatter) = Vth*randn(1);
    VY(R<Pscatter) = Vth*randn(1);
    V = sqrt(VY.*VY+VX.*VX);
    meanpaths(R<Pscatter) = 0;
    unscattered = ismissing(R<Pscatter,0);
    meanpaths(unscattered) = meanpaths(unscattered) +
 V(unscattered)*dt;
    MFP = sum(meanpaths)/nAtoms;
    MTBC = sum(meanpaths)/sum(V);


    Xnext = X + VX*dt;
    Ynext = Y + VY*dt;
    %X boundary conditions set
    right = Xnext>frameWidth;
    left = Xnext<0;
    Xnext(right) = Xnext(right)-frameWidth;
    Xnext(left) = Xnext(left) + frameWidth;
    %Y boundary conditions set
    top = Ynext > frameHeight;
    bottom = Ynext < 0;
    VY(top | bottom) = VY(top | bottom) * -1;
    %calculations for temperature
    Temperature(iteration) = 0.26*C.m_0*mean(V.^2)/4/C.kb;
    figure(4)
    xlim([0 frameWidth])
    ylim([0 frameHeight])
    hold on
    %plotting, but avoid plotting the full horizontal jump
    if abs(Xnext(1) - X(1)) < 2*abs(VX(1))*dt
        figure(4)
        plot([Xnext(1) X(1)], [Ynext(1) Y(1)], 'blue')
    end
    if abs(Xnext(2) - X(2)) < 2*abs(VX(2))*dt
        figure(4)
        plot([Xnext(2) X(2)], [Ynext(2) Y(2)], 'red')
    end
    if abs(Xnext(3) - X(3)) < 2*abs(VX(3))*dt
        figure(4)
        plot([Xnext(3) X(3)], [Ynext(3) Y(3)], 'green')
    end

    %updating positions, and advancing time a step forward so the
 while
    %loop works
    X = Xnext;
    Y = Ynext;
    t = t+dt;
    iteration = iteration + 1;
    pause(0.0001);
end
%Outputs, temperature, mean free path, and mean time between
 collisions
figure(5)
dummy = linspace(0,iteration, length(Temperature));
plot(dummy, Temperature)
title('Temperature of System Over Time')
xlabel('time')
ylabel('Temperature (K)')

fprintf('The mean free path is %d m.\n',MFP);
fprintf('The Mean Time Between Collisions is %d s.\n',MTBC);
```

```
The mean free path is 4.691454e-08 m.
The Mean Time Between Collisions is 1.965724e-13 s.
```




×10⁻⁷
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
×10⁻⁷

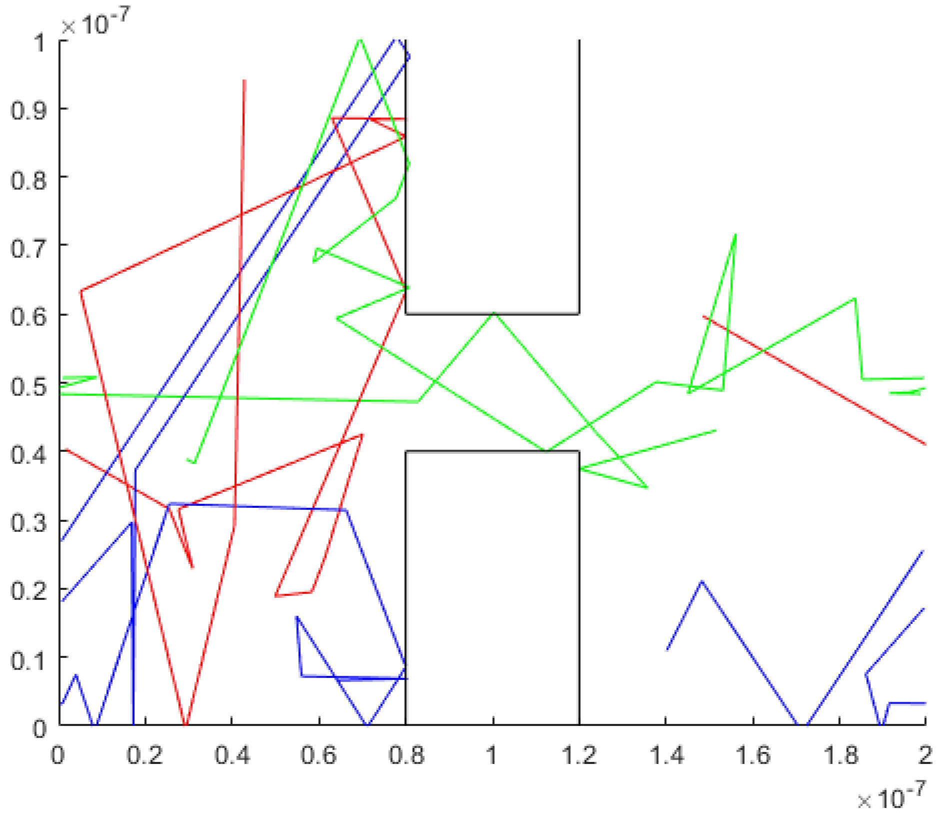

Temperature of System Over Time
Temperature (K)
420
400
380
360
340
320
300
280
260
240
220
0
100
200
300
400
500
600
700
800
time

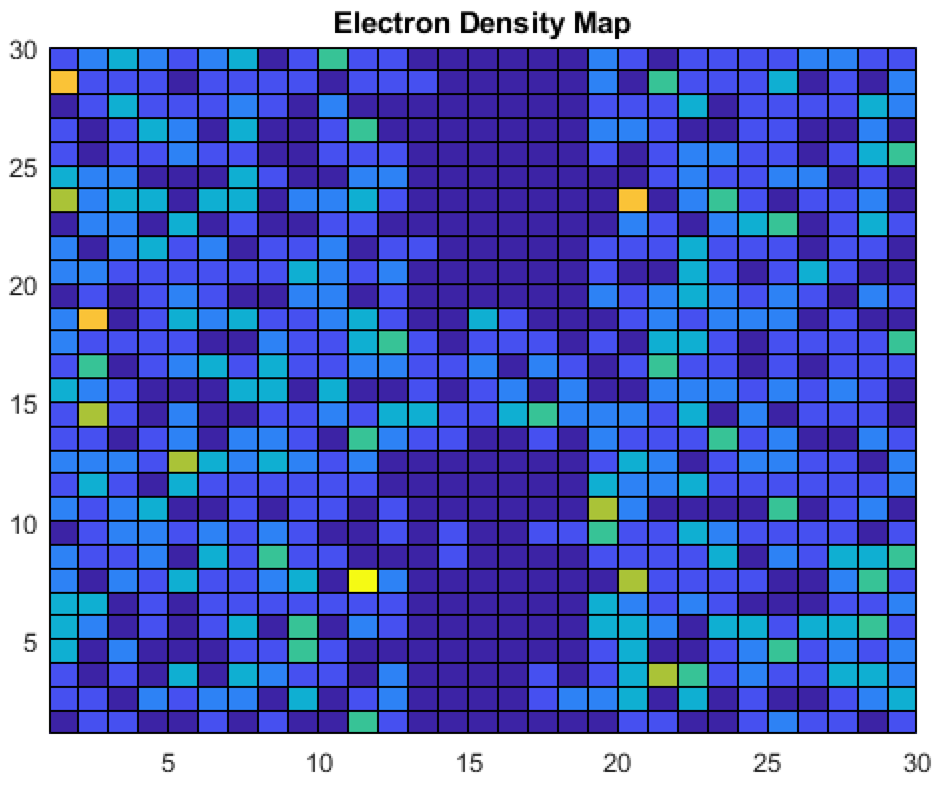
Section 3 - Enhancements

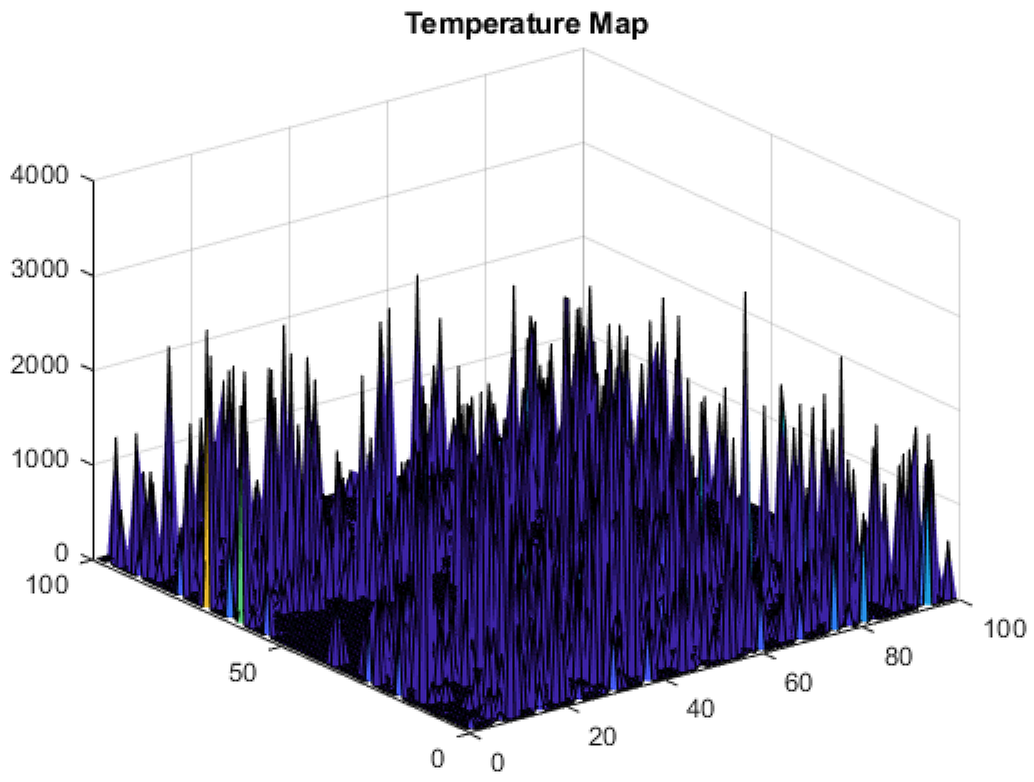
This section was fairly challenging, and certainly made sure I understood how to use logical indexing for the boundary conditions. The trajectory plot seems to work as intended, but the electron density map reveals that my boundary conditions are not perfect. Before the particle reflects off the boundary, it penetrates into the boxes just slightly, and it shows on the map. Other than this issue, the electron density map and temperature map are both looking proper.

```
%constants
clear
C.q_0 = 1.60217653e-19;
C.m_0 = 9.10938215e-31;
C.kb = 1.3806504e-23;
C.T = 300;
frameWidth = 200e-9;
frameHeight = 100e-9;
nAtoms = 1000;
bins = nAtoms / 10;
Vth = sqrt(2*C.kb*C.T /(0.26*C.m_0));
dt = frameHeight/Vth/100;
Tstop = 500*dt;
t = 0;
freepath = 0.2e-12;
Pscatter = 1 - exp(-dt/freepath);

%initializing vectors
Xnext = zeros(1,nAtoms);
Ynext = zeros(1,nAtoms);
VX = Vth * randn(1,nAtoms);
VY = Vth * randn(1,nAtoms);
V = sqrt(VY.*VY+VX.*VX);
X = frameWidth * rand(1, nAtoms);
Y = frameHeight * rand(1, nAtoms);
R = zeros(1, nAtoms);
Temperature = zeros(1, 100);
iteration = 1;

%defining box boundaries
boxleft = X>0.8e-7;
boxright = X<1.2e-7;
box1bottom = Y>0.6e-7;
box2top = Y<0.4e-7;
boxedin = (boxleft&boxright&box1bottom)|(boxleft&boxright&box2top);
%removing particles from the boxes
while sum(boxedin)>0
X(boxedin) = rand*frameWidth;
Y(boxedin) = rand*frameHeight;
boxleft = X>0.8e-7;
boxright = X<1.2e-7;
box1bottom = Y>0.6e-7;
box2top = Y<0.4e-7;
boxedin = (boxleft&boxright&box1bottom)|(boxleft&boxright&box2top);
```

```
end

while t < Tstop
    %determines which particles scatter and performs calculations on
 them
    %to determine mean free path and time between collisions
    R = rand(1,nAtoms);
    VX(R<Pscatter) = Vth*randn(1);
    VY(R<Pscatter) = Vth*randn(1);
    V = sqrt(VY.*VY+VX.*VX);
    Xnext = X + VX*dt;
    Ynext = Y + VY*dt;
    %set X boundary conditions
    right = Xnext>frameWidth;
    left = Xnext<0;
    Xnext(right) = Xnext(right)-frameWidth;
    Xnext(left) = Xnext(left) + frameWidth;
    %set Y boundary conditions
    top = Ynext > frameHeight;
    bottom = Ynext < 0;
    VY(top | bottom) = VY(top | bottom) * -1;
    %set boundary for sides of boxes
    box1sides = (Ynext>0.6e-7)&(Xnext>0.8e-7)&(Xnext<1.2e-7);
    box2sides = (Ynext<0.4e-7)&(Xnext>0.8e-7)&(Xnext<1.2e-7);
    VX(box1sides|box2sides) = VX(box1sides|box2sides) * -1;
    topbox = (Y<0.6e-7)&(Ynext>0.6e-7)&(Xnext>0.8e-7)&(Xnext<1.2e-7);
    bottombox =
 (Y>0.4e-7)&(Ynext<0.4e-7)&(Xnext>0.8e-7)&(Xnext<1.2e-7);
    VY(topbox|bottombox) = VY(topbox|bottombox) * -1;
    VX(topbox|bottombox) = VX(topbox|bottombox) * -1;

    %calculations for temperature
    Temperature(iteration) = 0.26*C.m_0*mean(V.^2)/4/C.kb;
    %setting up figure, drawing both boxes
    figure(6)
    xlim([0 frameWidth])
    ylim([0 frameHeight])
    hold on
    plot([0.8e-7 0.8e-7],[0 0.4e-7], 'black')
    plot([1.2e-7 1.2e-7],[0 0.4e-7], 'black')
    plot([0.8e-7 1.2e-7],[0.4e-7 0.4e-7], 'black')
    plot([0.8e-7 0.8e-7],[1e-7 0.6e-7], 'black')
    plot([1.2e-7 1.2e-7],[1e-7 0.6e-7], 'black')
    plot([0.8e-7 1.2e-7],[0.6e-7 0.6e-7], 'black')

    %plotting, but avoid plotting the full horizontal jump from right
 and
    %left boundaries
    if abs(Xnext(1) - X(1)) < 2*abs(VX(1))*dt
        figure(6)
        plot([Xnext(1) X(1)], [Ynext(1) Y(1)], 'blue')
    end
    if abs(Xnext(2) - X(2)) < 2*abs(VX(2))*dt
        figure(6)
        plot([Xnext(2) X(2)], [Ynext(2) Y(2)], 'red')
    end
    if abs(Xnext(3) - X(3)) < 2*abs(VX(3))*dt
        figure(6)
        plot([Xnext(3) X(3)], [Ynext(3) Y(3)], 'green')
    end
    %optional plotting for 2 more particles
    %if abs(Xnext(4) - X(4)) < 2*abs(VX(4))*dt
    %    figure(2)
    %    plot([Xnext(4) X(4)], [Ynext(4) Y(4)], 'cyan')
    %end
    %if abs(Xnext(5) - X(5)) < 2*abs(VX(5))*dt
    %    figure(2)
    %    plot([Xnext(5) X(5)], [Ynext(5) Y(5)], 'magenta')
   % end

    %updating positions, and advancing time a step forward so the
 while
    %loop works
    X = Xnext;
    Y = Ynext;
    t = t+dt;
    iteration = iteration + 1;
   % pause(0.0001);
end

%electron density map
figure(7)
EDM = hist3([X',Y'],[30,30]);
pcolor(EDM')
title('Electron Density Map')

%temperature map
xLim = linspace(0,frameWidth,100);
yLim = linspace(0,frameHeight,100);
xTempReg = discretize(X,xLim);
yTempReg = discretize(Y,yLim);
for q=1:1:100
    for w=1:1:100
        %Temperature contained in defined region
        tempReg = (q == xTempReg) & (w == yTempReg);

        %Total velocities in region
        vxTot=sum(VX(tempReg));
        vyTot=sum(VY(tempReg));
        vTot = sqrt((vxTot)^2+(vyTot)^2);

        %Calculate Temperature
        tempMap(q,w) = C.m_0*0.26*(vTot)^2/(2*C.kb);
    end
end
figure(8)
surf(tempMap)
title('Temperature Map')
```


×10⁻⁷
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
×10⁻⁷


Electron Density Map
30
25
20
15
10
5
5
10
15
20
25
30



Published with MATLAB® R2018a