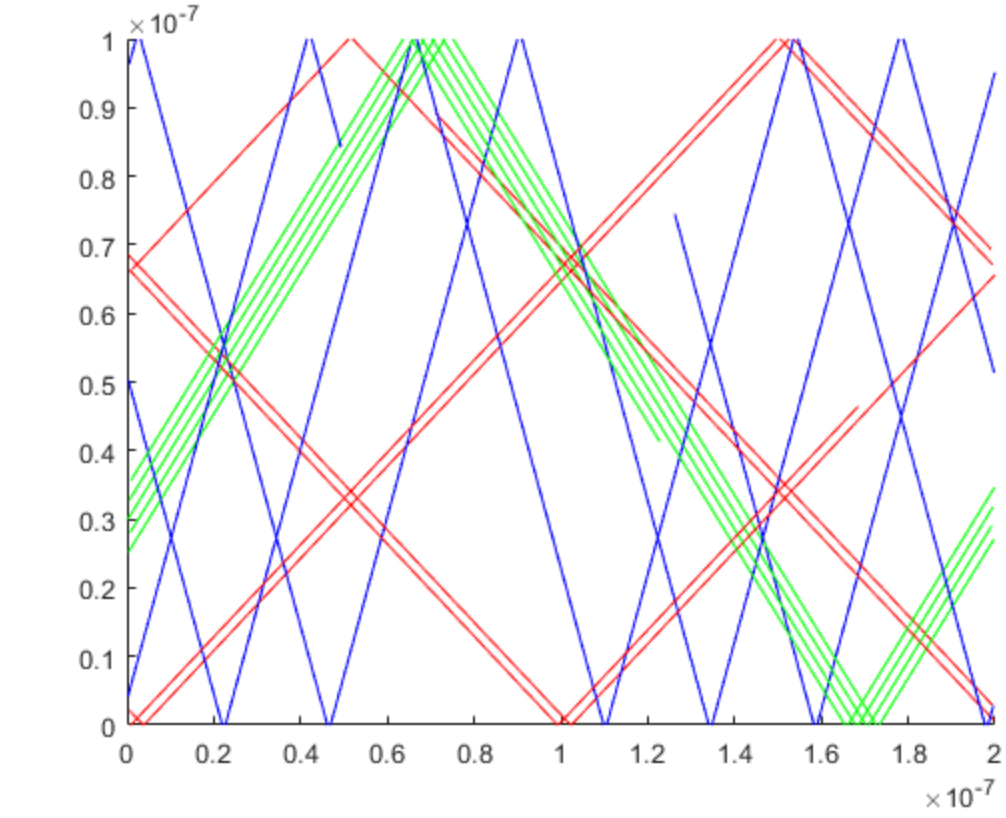
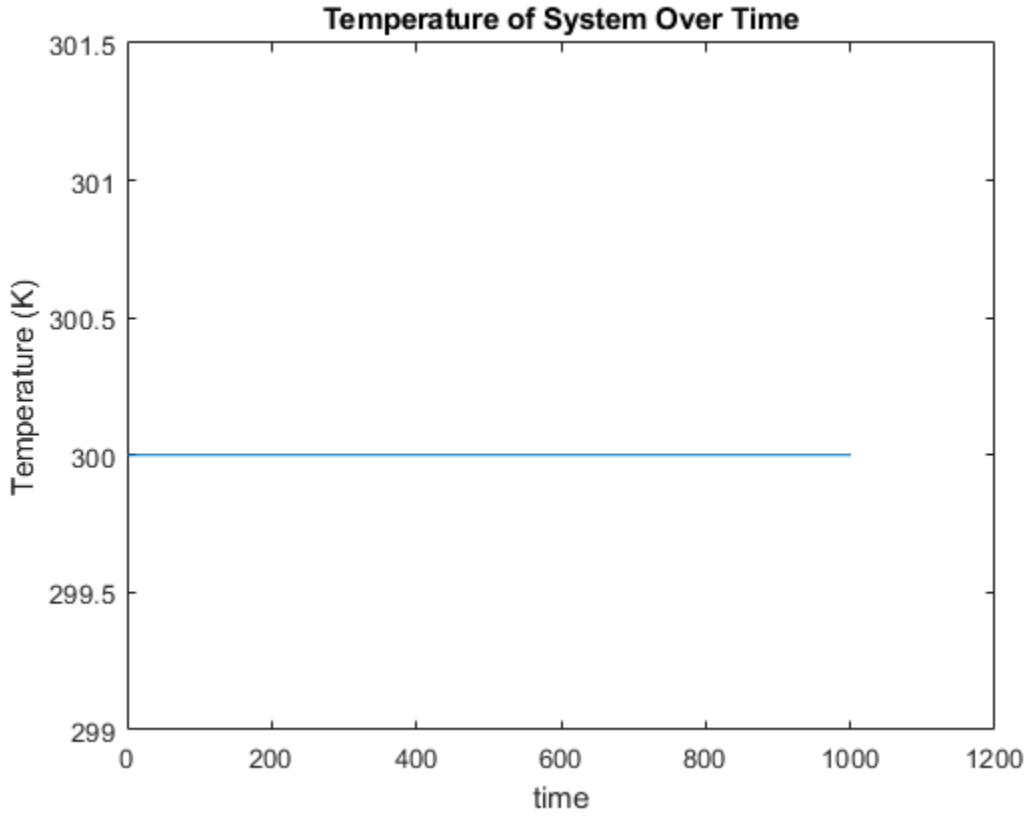
Section 1 - Electron Modelling

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

Published with MATLAB® R2018a