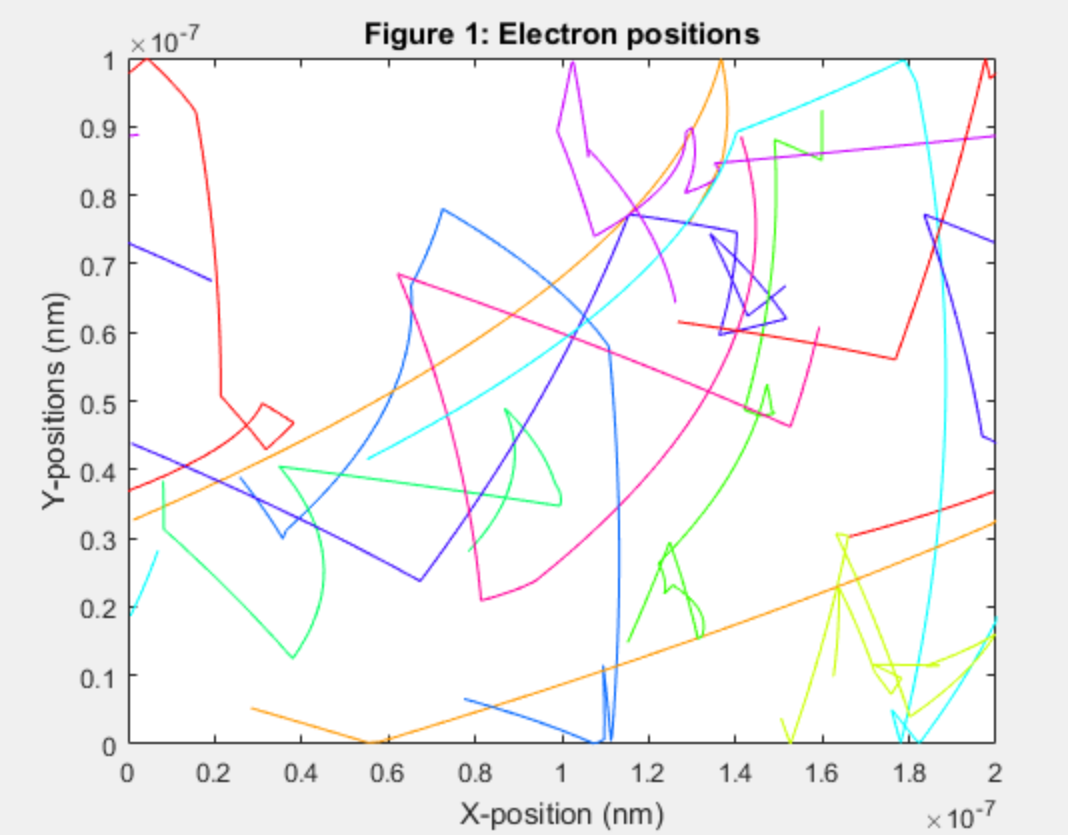
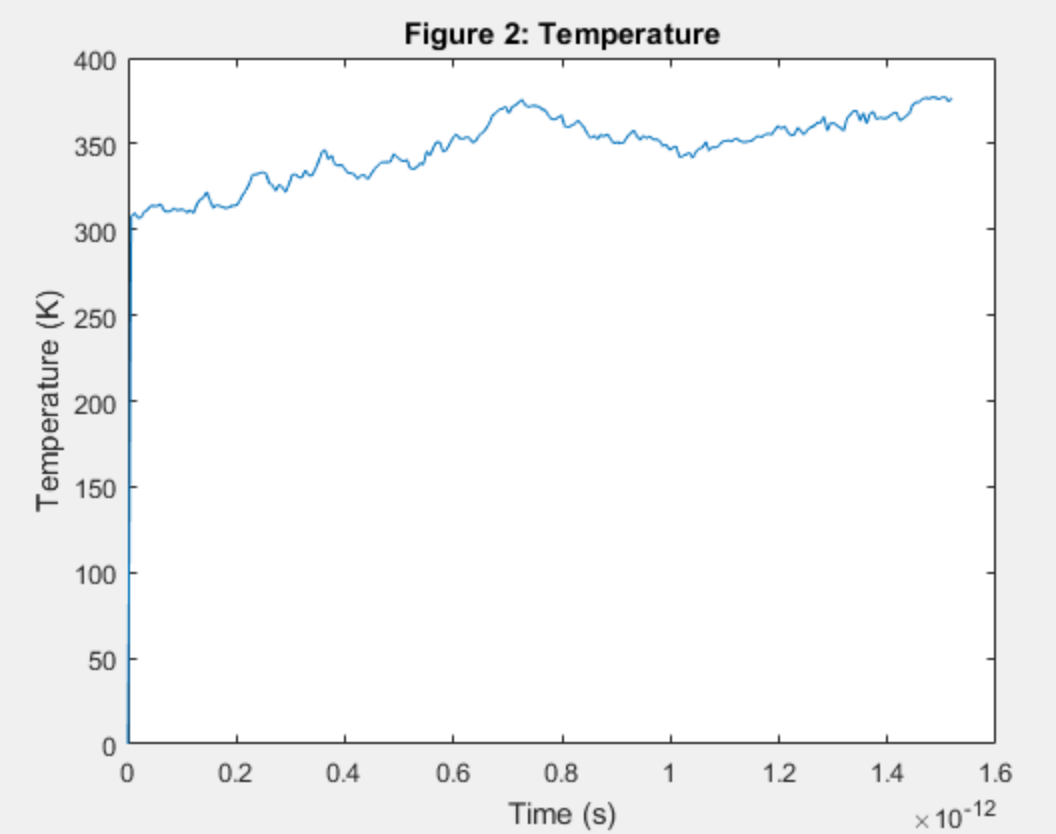
```
%% Assignment 3 - Kyle Poulin 100939284
%% Q1.)
% In assignment 3, we start with the same setup as in assignment 1, then
% add an electric field by applying 0.1V across the box. The force due to
% the electric field accelerates the electrons negatively, so the electrons
% will be more likely found on the positive side of the box.

% Taking the region to be the same area as in assignment 1:

xDim = 200*10^-9; %nanometers
yDim = 100*10^-9; %nanometers

% Number of Electrons:
numElectrons = 1000;



% Voltage:
voltage = 0.1;

kB = 1.3806*10^-23; %boltzmann constant
me = 9.11*10^-31; %mass of an electron
ce = 1.602*10^-19; %charge of an electron
electricField = voltage / xDim;
force = electricField*ce;
acceleration = force / (0.26*me);
vth = 132.2*10^3;
taus = zeros(1,numElectrons);
Tau=0;
mfp = 0;




barrier1 = 2/5;
barrier2 = 3/5;

% Next we need the initial positions of the electrons:
% Random numbers between 0->1 for the number of electrons
x_positions = rand(1,numElectrons);
y_positions = rand(1,numElectrons);

% Multiply by the dimmensions to get the location:
x_positions = x_positions*xDim;
y_positions = y_positions*yDim;

%velocities:
x_velocities = randn(1,numElectrons);
y_velocities = randn(1,numElectrons);

x_velocities = x_velocities*(vth);
y_velocities = y_velocities*(vth);

% set up the movie to run 1000 steps
```

```
time=0;
steps = 250;
timeStep = xDim/(steps*vth);
stopTime = steps*timeStep;
scatterProb = 1-exp(-timeStep/(0.2*10^-12));
timeVector = zeros(1,10000);
for i = 0:9999
    timeVector(i+1) = (i)*timeStep;
end
temperatures = 0;
chargeDensity = 1*10^-19;
currentDensity = 0;

% Check for forbidden positions:
i=1;
while i<=(xDim)
    if((x_positions(i)<barrier2*xDim && x_positions(i)>barrier1 &&
(y_positions(i)<barrier1*yDim || y_positions(i)>barrier2*yDim)))
        x_positions(i) = rand(1)*xDim;
        y_positions(i) = rand(1)*yDim;
        i = i-1;
    end
    i = i+1;
end
velocities = sqrt(x_velocities.*x_velocities + y_velocities.*y_velocities);

% Create a color map:
colours = hsv(10);
index = 0;
currentDensity = 0;
while time< stopTime
    % Scatter if necessary, then update mfp
    for i = 1:length(x_positions)
        taus(i) = taus(i) + timeStep;
        if rand()<scatterProb
            Tau = [Tau,taus(i)];
            mfp = [mfp, taus(i)*sqrt(x_velocities(i)^2+y_velocities(i)^2)];
            taus(i) = 0;
            x_velocities(i) = 132.2*10^3*randn();
            y_velocities(i) = 132.2*10^3*randn();
            velocities = sqrt(x_velocities.*x_velocities +
y_velocities.*y_velocities);
        end
    end

    % Do the temperature calculation:
    temperatures = [temperatures,(1/2)/kB*me*0.26*(mean(x_velocities.^2)
+mean(y_velocities.^2))];

    % Update electron positions then check for the forbidden ones
    x_positions2 = x_positions;
    y_positions2 = y_positions;

    x_positions = x_positions+x_velocities*timeStep;
    y_positions = y_positions+y_velocities*timeStep;
```

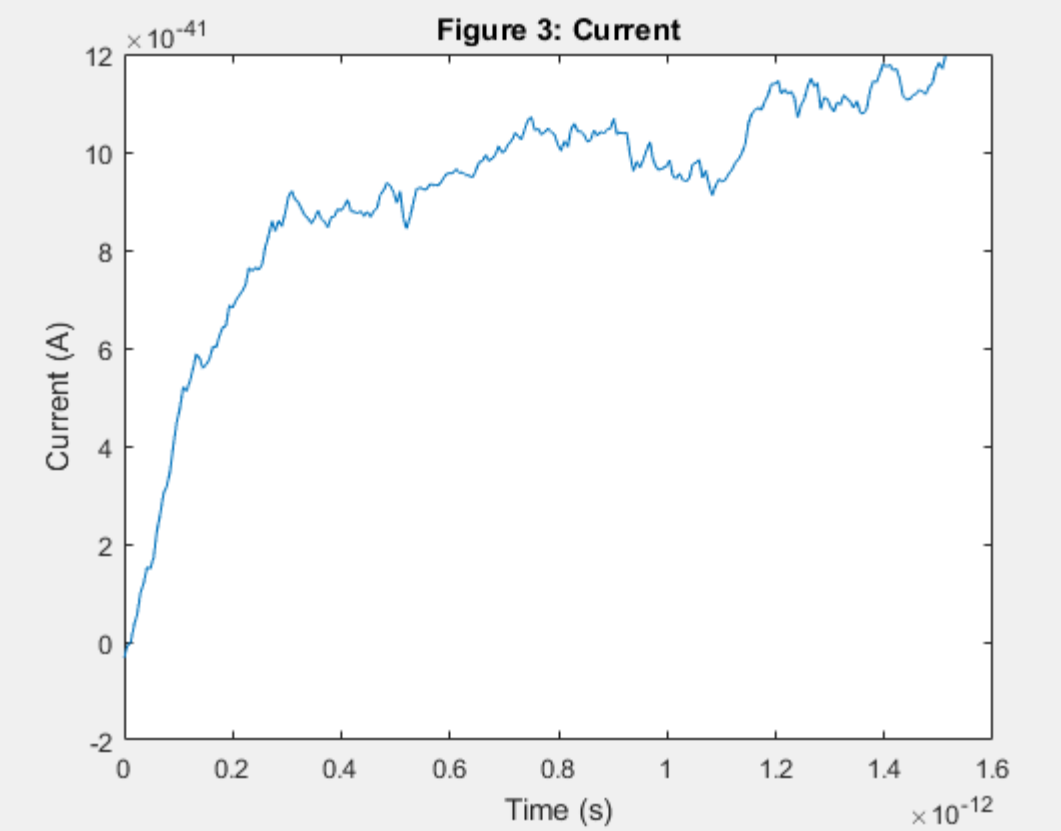
```
    x_velocities = x_velocities-acceleration*timeStep;

    for i=1:numElectrons
        if x_positions(i) > xDim
            x_positions(i) = x_positions(i)-xDim;
            x_positions2(i) = 0;
        end
        if x_positions(i) < 0
            x_positions(i) = x_positions(i) + xDim;
            x_positions2(i) = xDim;
        end
        if y_positions(i) > yDim
           y_velocities(i) = -y_velocities(i);
           y_positions(i) = 2*yDim-y_positions(i);
        end
        if y_positions(i) < 0
            y_positions(i) = abs(y_positions(i));
            y_velocities(i) = -y_velocities(i);
        end
    end

    %% Plot positions:
    figure(1);
    title('Figure 1: Electron positions');
    xlabel('X-position (nm)');
    ylabel('Y-positions (nm)');
    xlim([0 xDim]);
    ylim([0 yDim]);
    pause(0.05);
    for i=1:10
        plot([x_positions2(i);x_positions(i)],
[y_positions2(i);y_positions(i)], 'color', colours(i,:));
        hold on
    end
```




```
%% Plot temperature:
figure(2);
plot(timeVector(1:length(temperatures)),temperatures);
title('Figure 2: Temperature');
xlabel('Time (s)');
ylabel('Temperature (K)');
```

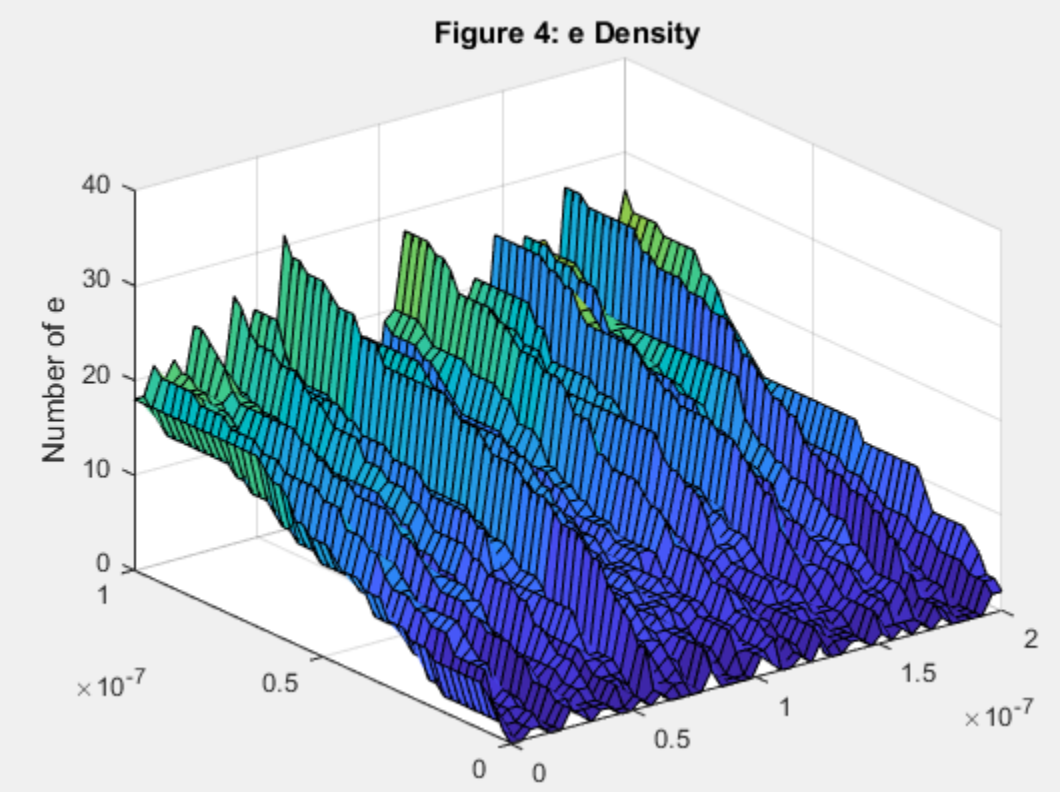



```
% Update the index and time:
index = index+1;
time = time+timeStep;

dVelocity = mean(x_velocities);
currentDensity(index) = dVelocity*chargeDensity*ce;

%% Plot Current Density
% The average drift velocity of the electrons is used in conjunction
% with the charge of the electron and the dimmensions of the box to
% find the current.
figure(3);
plot(timeVector(1:length(currentDensity)),-currentDensity*yDim);
title('Figure 3: Current');
xlabel('Time (s)');
ylabel('Current (A)');
```

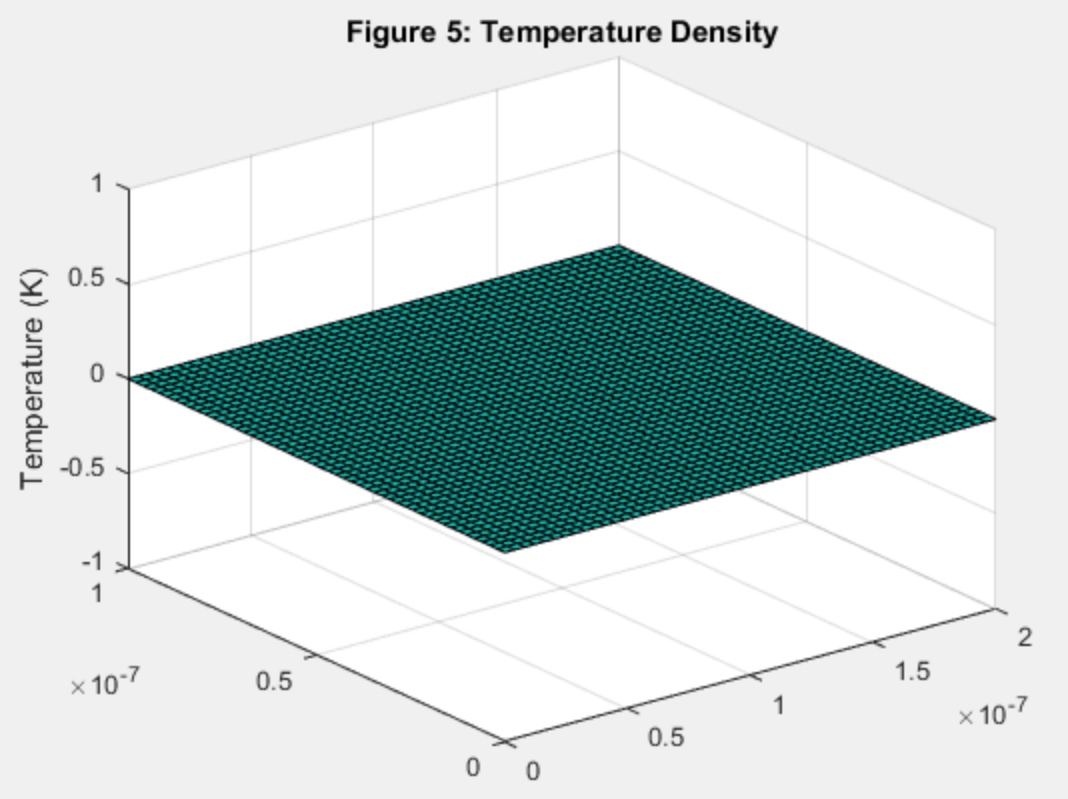



```
end

% Create the position, velocity and temperature density maps

positions = zeros(50);
velX = zeros(50);
velY = zeros(50);
temp = zeros(50);

for i=1:50
    for j=1:50
        for nE = 1:numElectrons
            if (x_positions(nE)>=((i-
1)*xDim/50))&&(x_positions(nE)<(i*xDim/50))&&(y_positions(nE)>=(j-
1)*y_positions(nE)/50)&&(y_positions(nE)<(j*yDim/50))
                positions(j,i)= positions(j,i)+1;

                velY(j,i) = velY(j,i)+velY(nE)^2
                velX(j,i) = velX(j,i)+velX(nE)^2
            end
            if positions(j,i)==0
                temp(j,i) = 0;
            else
                temp(j,i) = 0.26*me*(velX(j,i)+velY(j,i))/positions(j,i)/kB;
            end
        end
    end
end
```

```
figure(4);
surf(linspace(0,xDim,50), linspace(0,yDim,50),positions);
title('Figure 4: e Density');
zlabel('Number of e');
```



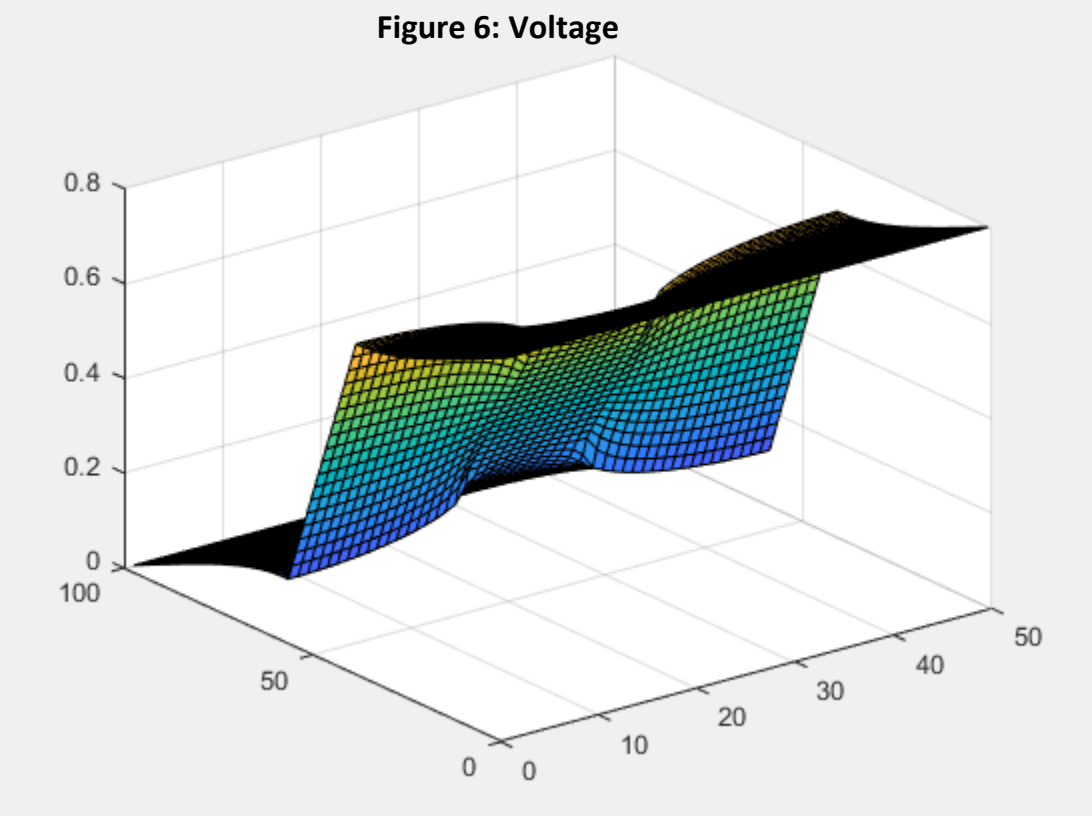

```
figure(5);
surf(linspace(0,xDim,50),linspace(0,yDim,50),temp);
title('Figure 5: Temperature Density');
zlabel('Temperature (K)');
```




```
%% Q 2.)
% With the addition of the bottle-neck, the electric field is no longer
% uniform due to the voltage drop over the barriers. In this question,
% finite difference is used to show the potential and field.


nx = 100;
ny = 50;

sigma1= 1;
sigma2 = 0.001;

xlowbound = round((2/5)*nx);
xhighbound = round((3/5)*nx);
ylowbound = round((2/5)*ny);
yhighbound = round((3/5)*ny);

g = sparse(nx*ny);
b = zeros(1,nx*ny);

box = [xlowbound xhighbound ylowbound yhighbound];
sigma = zeros(nx,ny);

for i=1:nx
    for j = 1:ny
        if (i>box(1)) && (i<box(2)) && ((j<box(3)) || (j>box(4)))
            sigma(i,j) = sigma2;
        else
            sigma(i,j)=sigma1;
```

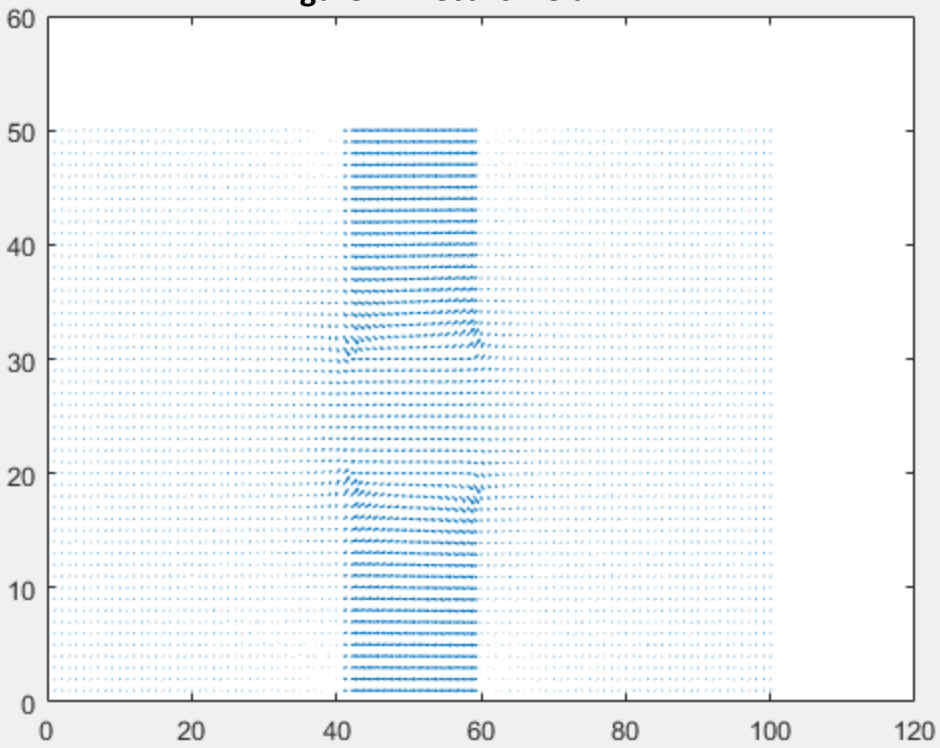
```
        end
    end
end


for i = 1:nx
    for j=1:ny
        n=j+(i-1)*ny;
        %set boundary conditions
        if i==1
            g(n,:) = 0;
            g(n,n) = 1;
            b(n) = 0.8;
        elseif i==nx
            g(n,:) = 0;
            g(n,n) = 1;

        elseif j==1
            up = (sigma(i,j) + sigma(i,j+1))/2;
            left = (sigma(i,j) + sigma(i-1,j))/2;
            right = (sigma(i,j) + sigma(i+1,j))/2;

            g(n,n) = -(up + left + right);
            g(n,n+1) = up;
            g(n,n-ny) = left;
            g(n,n+ny) = right;

        elseif j==ny
            down = (sigma(i,j) + sigma(i,j-1))/2;
            left = (sigma(i,j) +sigma(i-1,j))/2;
            right= (sigma(i,j) + sigma(i+1,j))/2;

            g(n,n) =  -(up + left + right);
            g(n,n+ny) = right;
            g(n,n-1) = down;
            g(n,n-ny) = left;

        else
            up = (sigma(i,j) + sigma(i,j+1))/2;
            down=(sigma(i,j)+sigma(i,j-1))/2;
            left=(sigma(i,j)+sigma(i-1,j))/2;
            right=(sigma(i,j)+sigma(i+1,j))/2;

            g(n,n) = -(up + down + right+ left);
            g(n,n+1) = up;
            g(n,n-1) = down;
            g(n,n+ny) = right;
            g(n,n-ny) = left;
        end
    end



end
```

```
E = g\b';

w = zeros(nx,ny);

for i=1:nx
    for j = 1:ny
        n = j+ (i-1)*ny;
        w(i,j) = E(n);
    end

end

figure(6);
title('Figure 6: Voltage');
xlabel('Width');
ylabel('length');
view(-256,36);
surf(w);
```




```
[ex,ey] = gradient(w);
ex = -ex;
ey=-ey;
figure(7)
title('Figure 7: Electric Field');
quiver(ey',ex');
```

Figure 7: Electric Field



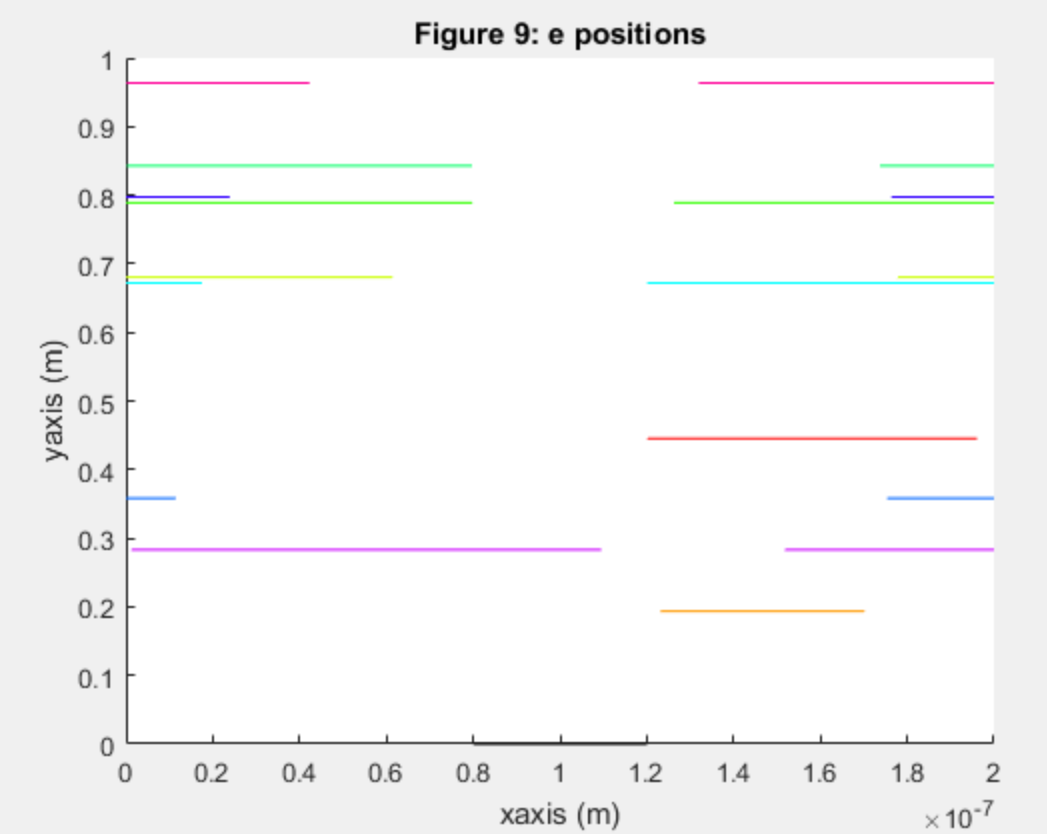

```
%% Q 3.)
% Using the bottle-neck barrier, the electron positions are traced. The
% bottle-neck should cause a build-up of particles on one side after being
% blocked by the barrier.

clear
close all

xDim = 200*10^-9; %nanometers
yDim = 100*10^-9; %nanometers

% Number of Electrons:
numElectrons = 1000;



% Voltage:
voltage = 0.1;

kB = 1.3806*10^-23; %boltzmann constant
me = 9.11*10^-31; %mass of an electron
ce = 1.602*10^-19; %charge of an electron
electricField = voltage / xDim;
force = electricField*ce;
acceleration = force / (0.26*me);
vth = 132.2*10^3;
taus = zeros(1,numElectrons);
Tau=0;
mfp = 0;
```

```
barrier1 = 2/5;
barrier2 = 3/5;


mfp = 0;
rethermBool = 0;
temperatures = zeros(1,300);

% Random numbers between 0->1 for the number of electrons
x_positions = rand(1,numElectrons);
y_positions = rand(1,numElectrons);

% Multiply by the dimmensions to get the location:
x_positions = x_positions*xDim;
y_positinos = y_positions*yDim;

%velocities:
x_velocities = randn(1,numElectrons);
y_velocities = randn(1,numElectrons);

x_velocities = x_velocities*(vth);
y_velocities = y_velocities*(vth);

% set up the movie to run 1000 steps
time=0;
steps = 250;
timeStep = xDim/(steps*vth);
stopTime = steps*timeStep;
scatterProb = 1-exp(-timeStep/(0.2*10^-12));
timeVector = zeros(1,10000);
for i = 0:9999
    timeVector(i+1) = (i)*timeStep;
end
temperatures = 0;
chargeDensity = 1*10^-19;
currentDensity = 0;

% Check for forbidden values
i=1;
while i<=(xDim)
    if((x_positions(i)<barrier2*xDim && x_positions(i)>barrier1 &&
(y_positions(i)<barrier1*yDim || y_positions(i)>barrier2*yDim)))
        x_positions(i) = rand(1)*xDim;
        y_positions(i) = rand(1)*yDim;
        i = i-1;
    end
    i = i+1;
end

% Create a color map:
```

```
colours = hsv(10);
index = 0;

figure(1)
hold on
rectangle('position',[barrier1*xDim,barrier2*yDim,0.2*xDim,barrier1*yDim]);
rectangle('position',[barrier1*xDim,0,0.2*xDim,barrier1*yDim]);

while time < stopTime
    %Scattering
    for i = 1:length(x_positions)
        taus(i) = taus(i) + timeStep;
        if rand()<scatterProb
            Tau = [Tau,taus(i)];
            mfp = [mfp, taus(i)*sqrt(x_velocities(i)^2+y_velocities(i)^2)];
            taus(i) = 0;
            x_velocities(i) = 132.2*10^3*randn();
            y_velocities(i) = 132.2*10^3*randn();
            velocities = sqrt(x_velocities.*x_velocities +
y_velocities.*y_velocities);
        end
    end
    % Get acceleration
    for i=1:100
        for j=1:50
            for nE = 1:numElectrons
                if x_positions(nE) >= ((i-
1)*xDim/100)&&(x_positions(nE)<(i*xDim/100))&&(y_positions(nE)>=(j-
1)*y_positions(nE)/50)&&(y_positions(nE)<(j*yDim/50))
                    x_velocities = x_velocities(n) - (ey(i,j)/10^-
9/(0.26*me)*ce*timeStep);
                    y_velocities = y_velocities(n) - (ex(i,j)/10^-
9/(0.26*me)*ce*timeStep);
                end
            end
        end
    end

    % Temperature
    temperatures = [temperatures,(1/2)/kB*me*0.26*(mean(x_velocities.^2)
+mean(y_velocities.^2))];

    % Update electron positions then check for the forbidden ones
    x_positions2 = x_positions;
    y_positions2 = y_positions;

    x_positions3 = x_positions;
    y_positions3 = y_positions;

    x_positions = x_positions+x_velocities*timeStep;
    y_positions = y_positions+y_velocities*timeStep;

    % Check for out of bounds
    for i=1:numElectrons
        if x_positions(i) > xDim
```

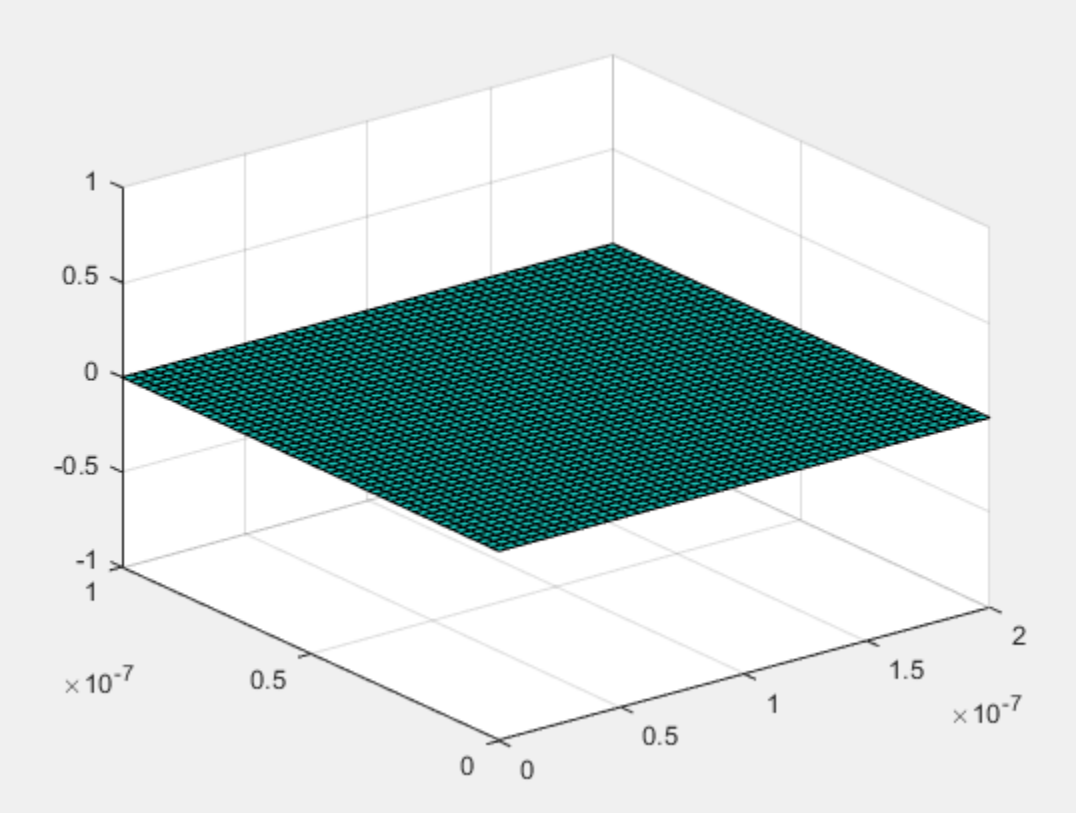
```
            x_positions(i) = x_positions(i)-xDim;
            x_positions2(i) = 0;
        end
        if x_positions(i) < 0
            x_positions(i) = x_positions(i) + xDim;
            x_positions2(i) = xDim;
        end
        if y_positions(i) > yDim
           y_positions(i) = 2*yDim-y_positions(i);
           y_velocities(i) = -y_velocities(i);
        end
        if y_positions(i) < 0
            y_velocities(i) = -y_velocities(i);
            y_positions(i) = abs(y_positions(i));
        end
        %check box boundaries:
        if y_positions(i)<barrier1*yDim && x_positions(i)>barrier1*xDim &&
x_positions(i)<barrier2*xDim
            if y_positions(i)<barrier1*yDim && y_positions3(i)>barrier1*yDim
                y_positions(i) = abs(y_positions(i)-
barrier1*yDim)+barrier1*yDim;
                if rethermBool==1
                    x_velocities(i) = vth*randn(1);
                    y_velocities(i) = vth*abs(randn(1));
                else
                    y_velocities(i) = -y_velocities(i);
                end
            end
            if x_positions3(i)<barrier1*xDim && x_positions(i)>barrier1*xDim
                x_positions(i) = barrier1*xDim - abs(x_positions(i)-
barrier1*xDim);
                if rethermBool==1
                    x_velocities(i) = -vth*abs(randn(1));
                    y_velocities(i) = vth*randn(1);
                else
                    x_velocities(i) = -x_velocities(i);
                end
            end
            if x_positions3(i)>barrier2*xDim && x_positions(i)<barrier2*xDim
                x_positions(i) = abs(x_positions(i)-barrier2*xDim) +
barrier2*xDim;
                if rethermBool==1
                    x_velocities(i) = vth*abs(randn(1));
                    y_velocities(i) = vth*randn(1);
                else
                    x_velocities(i) = -x_velocities(i);
                end
            end
        end


        if y_positions(i)>barrier2*yDim && x_positions(i)>barrier1*xDim &&
x_positions(i)<barrier2*xDim
            if y_positions(i)>barrier2*yDim && y_positions3(i)<barrier2*yDim
                y_positions(i) = 1.2*yDim-y_positions(i);
                if rethermBool==1
```

```
                    x_velocities(i) = vth*randn(1);
                    y_velocities(i) = -vth*abs(randn(1));
                else
                    y_velocities(i) = -y_velocities(i);
                end
            end
            if x_positions3(i)<barrier1*xDim && x_positions(i)>barrier1*xDim
                x_positions(i) = barrier1*xDim - abs(x_positions(i)-
barrier1*xDim);
                if rethermBool==1
                    x_velocities(i) = -vth*abs(randn(1));
                    y_velocities(i) = vth*randn(1);
                else
                    x_velocities(i) = -x_velocities(i);
                end
            end
            if x_positions3(i)>barrier2*xDim && x_positions(i)<barrier2*xDim
                x_positions(i) = abs(x_positions(i)-barrier2*xDim) +
barrier2*xDim;
                if rethermBool==1
                    x_velocities(i) = vth*abs(randn(1));
                    y_velocities(i) = vth*randn(1);
                else
                    x_velocities(i) = -x_velocities(i);
                end
            end
        end

    end
    % Plots!

    title('Figure 9: e positions');
    xlabel('xaxis (m)');
    ylabel('yaxis (m)');
    pause(0.01);
    for i=1:10

plot([x_positions2(i);x_positions(i)],[y_positions2(i);y_positions(i)],
'color', colours(i,:));
        hold on
    end
```

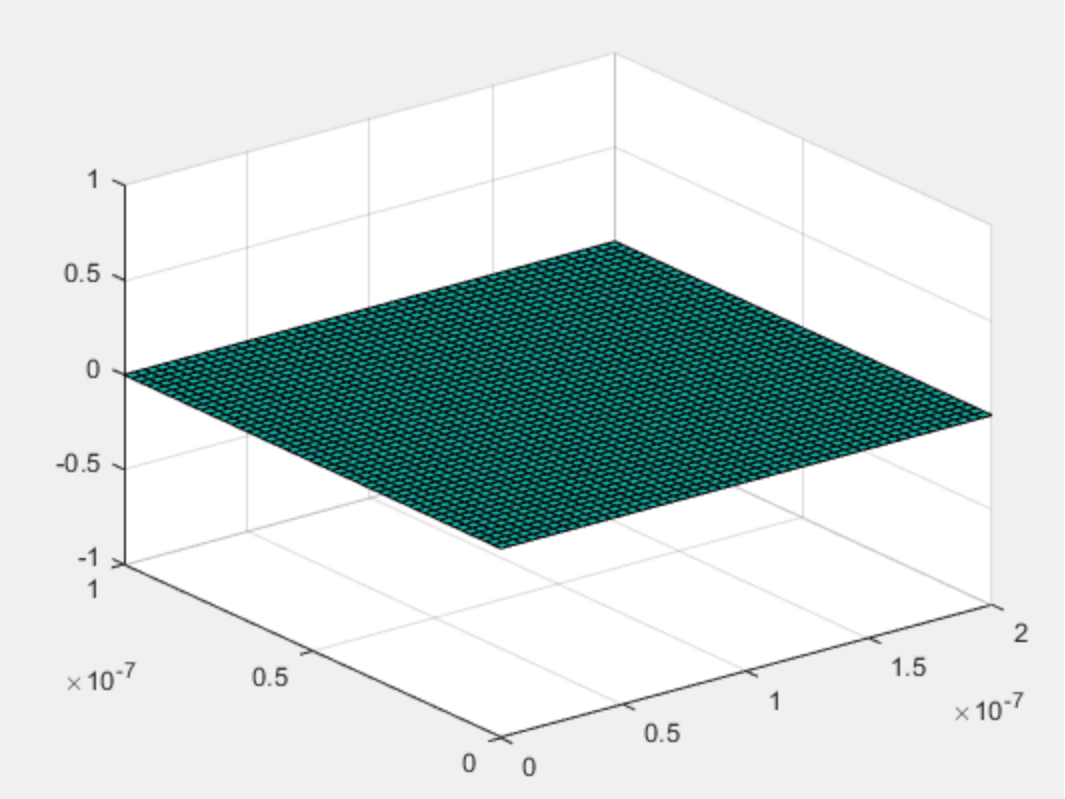



```
    time = time+timeStep;
    index = index +1;


    positions = zeros(50);
    vel = zeros(50);
    temp = zeros(50);

    for i=1:50
        for j=1:50
            for nE = 1:numElectrons
                if (x_positions(nE) >= ((i-
1)*xDim/50))&&(x_positions(nE)<(i*xDim/50))&&(y_positions(nE)>=(j-
1)*y_positions(nE)/50)&&(y_positions(nE)<(j*yDim/50))
                    positions(j,i)= positions(j,i)+1;


                    vel(j,i) = vel(j,i)+sqrt(x_velocities(nE)^2
+y_velocities(nE)^2);
                end
                if positions(j,i)==0
                    temp(j,i) = 0;
                else
                    temp(j,i) = 0.26*me*(vel(j,i))/positions(j,i)/kB;
                end
            end
        end
```

```
    end
end

figure(2)
title('Figure 10: electron density');
zlabel('num electrons');
surf(linspace(0,xDim,50),linspace(0,yDim,50),positions);
```




```
figure(3)
title('Figure 10: temperature density');
zlabel('temperature (K)');
surf(linspace(0,xDim,50),linspace(0,yDim,50),temp);
```




```
% To improve this simulation, more time steps could be taken, and more
% electrons could be used.
```