Assignment 3

Table of Contents

Joel Demetre (100943543)

Question 1

```
close all;
clear all;
Q1 = true;
Q2 = true;
Q3 = true;



xlimits = [0, 2e-7];
ylimits = [0, 1e-7];
PlotHowMany = 10;
Timestep = 5e-15;
NumParticles = floor((xlimits(2) - xlimits(1))*(ylimits(2) -
 ylimits(1))*10^19);
T = 300;
Kbolt = 1.38064852E-23;
tmn = 0.2e-12;
Colomb = -1.602e-19;
mo = 9.11e-31;
mn = mo * 0.26;
mycolors = hsv(PlotHowMany);
vx = (Kbolt*T/mn)^.5;
vy = (Kbolt*T/mn)^.5;
vth = (vx^2 + vy^2)^.5;
MeanFreePath = tmn*vth;
Pscat = 1-exp(-Timestep/tmn);

if Q1 == true
```

A - Electric Field on Electrons

```
voltDiffx = 0.1;
voltDiffy = 0;
xLeng = 50;
yLeng = 50;



%Part 1 A)
% The Electric Field in the X Direction is
Efieldx = voltDiffx/(xlimits(2) - xlimits(1))
% The Electric Field in the Y Direction is
Efieldy = voltDiffy/(ylimits(2) - ylimits(1))

% In units of V/m.
```

```

Efieldx =

   5.0000e+05


Efieldy =

     0
```

B - Force on the Electrons

```
%Part 1 B)
% The Force in the X Direction is
Forcex = Efieldx*Colomb
% The Force in the Y Direction is
Forcey = Efieldy*Colomb

% In units of VC/m.
```

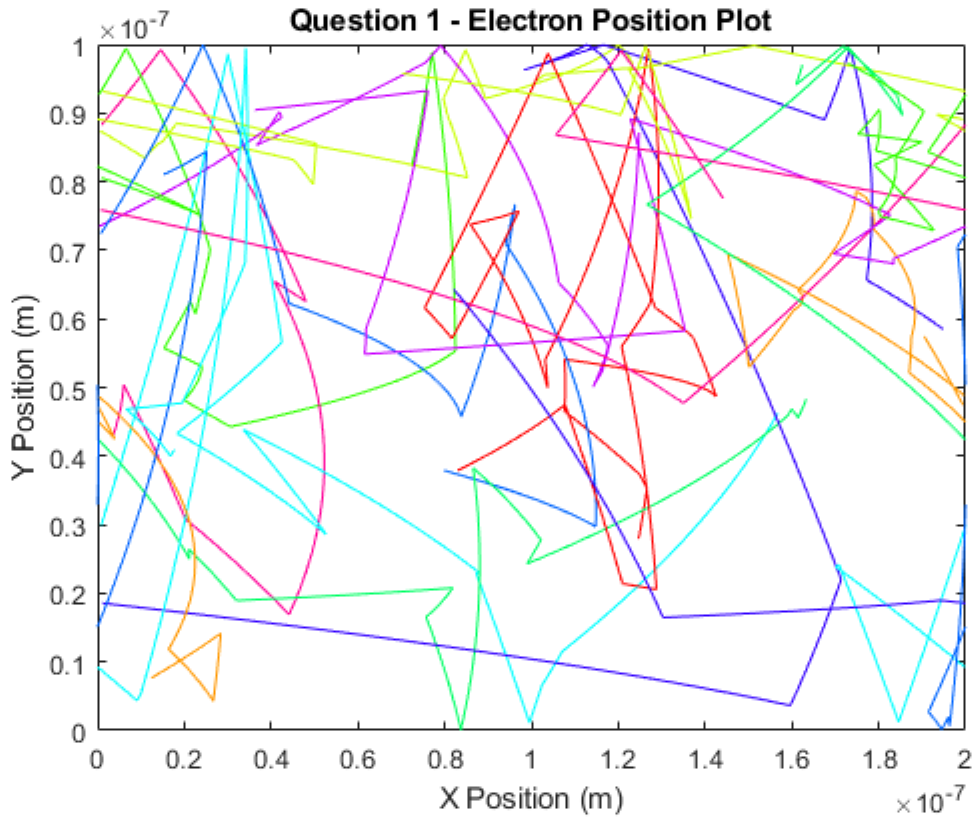
```

Forcex =

  -8.0100e-14


Forcey =

     0
```

C - Acceleration on the Electrons

```
%Part 1 C)
```

```
% The Acceleration in the X Direction is
Accelx = Forcex/mn
% The Acceleration in the X Direction is
Accely = Forcey/mn

% In units of m/s.


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*500;
graphfor = Timestep*500;

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

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
%Update the Position
ScatterTime = zeros(floor(endtime/Timestep)+ 1,1);
Current = 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
    count = 0;
    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);
           count = count - 1;
       elseif x(1,kt) +  x(2,kt)*Timestep > xlimits(2)
           x(1,kt) = xlimits(1);
           xprev(1,kt) = xlimits(1);
           count = count + 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
    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;
    Temperature(round(i/Timestep) + 1,1) = CalcTemp;
    x(2,:) = x(2,:) + Accelx*Timestep;
    y(2,:) = y(2,:) + Accely*Timestep;
    Current(round(i/Timestep) + 1,1) = Colomb*count/Timestep;
end
title('Question 1 - Electron Position Plot');
xlabel('X Position (m)');
ylabel('Y Position (m)');
hold off;
```

```
Accelx =

  -3.3817e+17


Accely =

     0
```



D - Relationship between Electron Drift and Average Carrier Velocity

The electron drift current density is given by:

$J = q\rho\mu E$

where q is the electron charge, ρ is the resistivity, μ is the electron mobility and E is the electric field applied. This equation is the represents the drift current and then the average velocity of the particle is given by:

$v_d = \mu E$

where v_d is the average velocity, and μ is electron mobility and E is electric field again. By combing these two we can create an equation of current density in terms of average velocity:

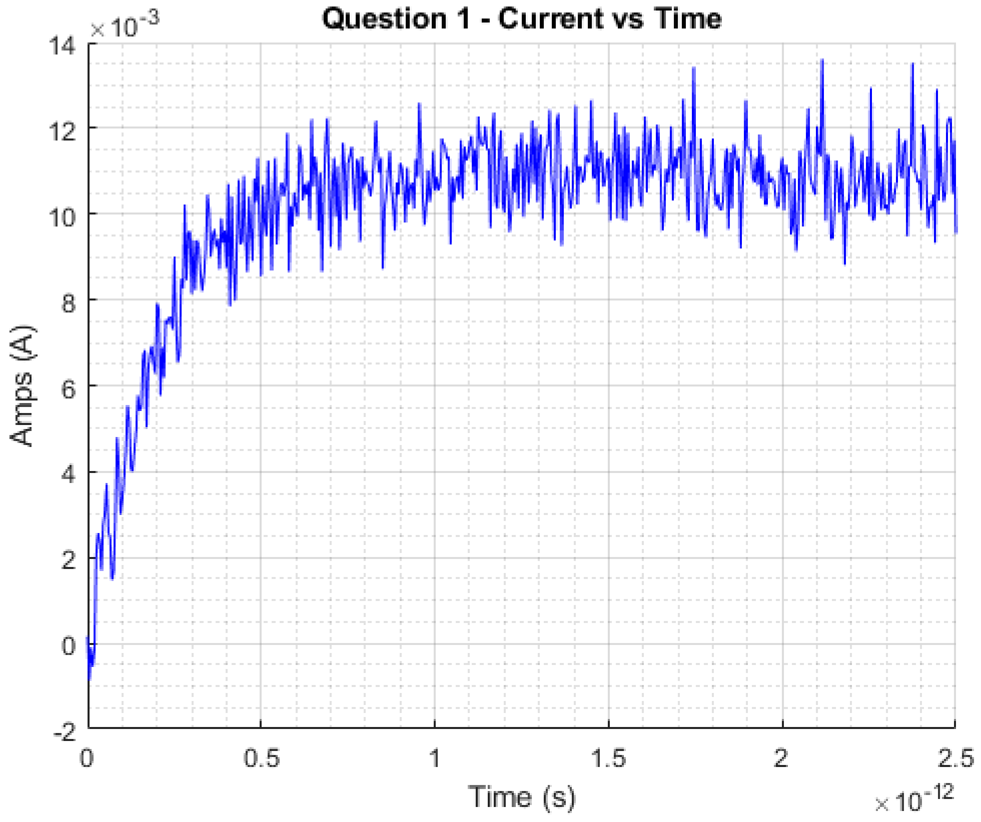
$J = q\rho v_d$

The calculation of current can be done by multiplying the current density by the cross sectional area.

$I = JA = Aq\rho v_d$

```
figure;
```

```
title('Question 1 - Current vs Time');
xlabel('Time (s)');
ylabel('Amps (A)');
hold on;
plot(0:Timestep:endtime, Current(:, 1), 'b');
grid on;
grid minor;
hold off;
```

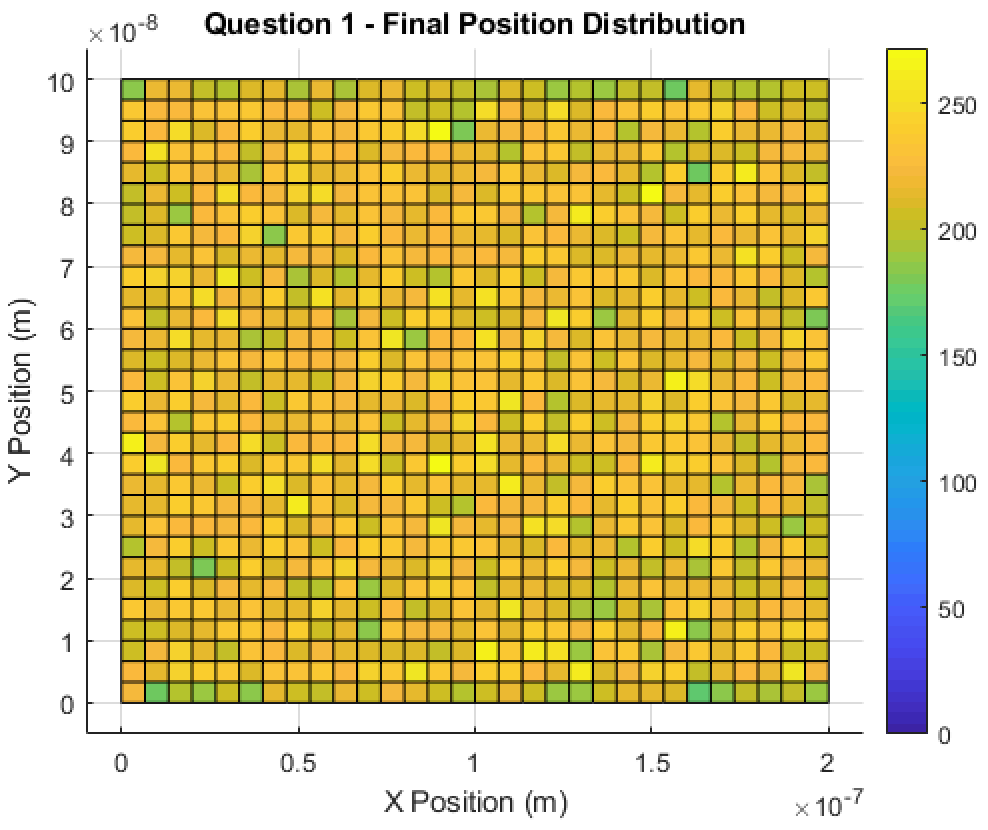


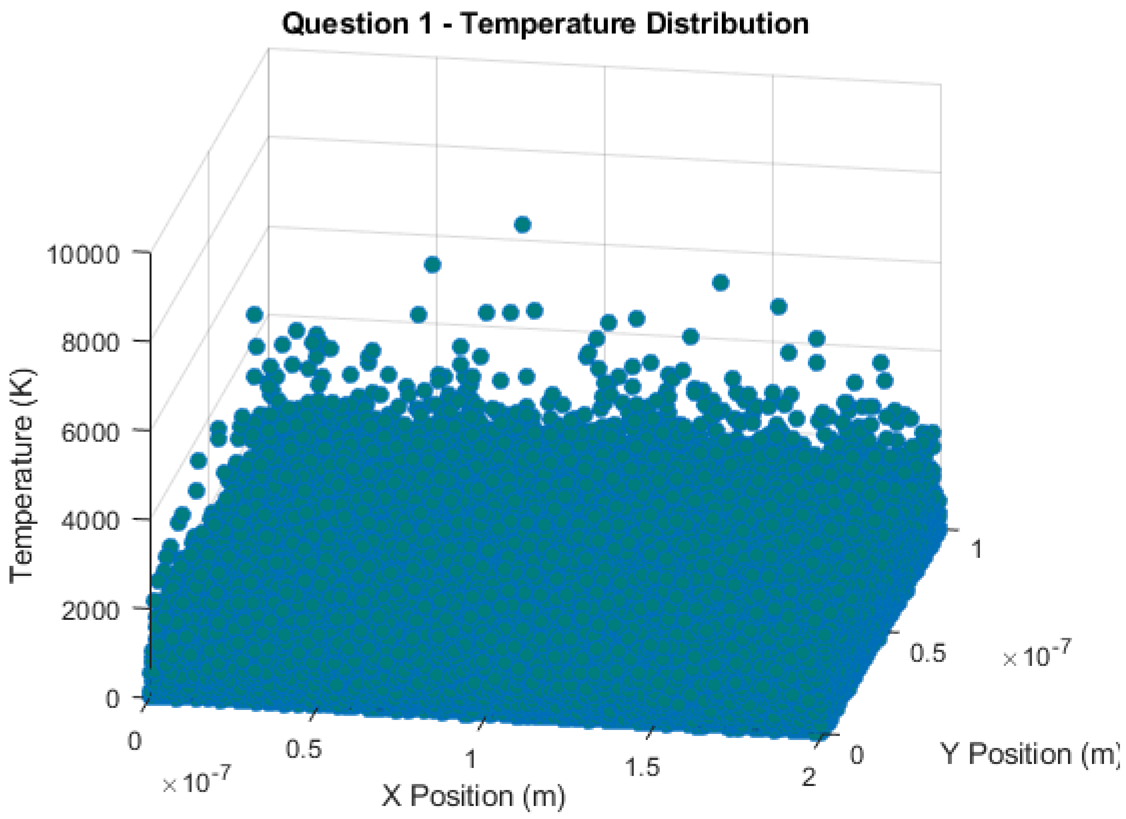
The current starts of at zero or very small, this is due to the fact that only the electrons that are moving randomly out of the box through the x axis limits are counting towards current flow. Then as more acceleration is added to the electrons with more time steps; more begin to accelerate over the boundary in the direction by the electric field. This approaches a limit and is somewhat stablizies with some flucuation due to the inherent randomness with diffusion and scattering.

E - Electron Density Map and Temperature Map

```
figure;
Z = [transpose(x(1,:)), transpose(y(1,:))];
hist3(Z, 'Nbins',[30 30], 'CdataMode', 'auto');
hold on;
colorbar;
view(2);
grid on;
title('Question 1 - Final Position Distribution');
xlabel('X Position (m)');
```

```
ylabel('Y Position (m)');
hold off;


figure
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([xlimits(1), xlimits(2)]);
ylim([ylimits(1), ylimits(2)]);
zlabel('Temperature (K)');
title('Question 1 - Temperature Distribution');
view(10,25);
hold off;
```





```
end
```

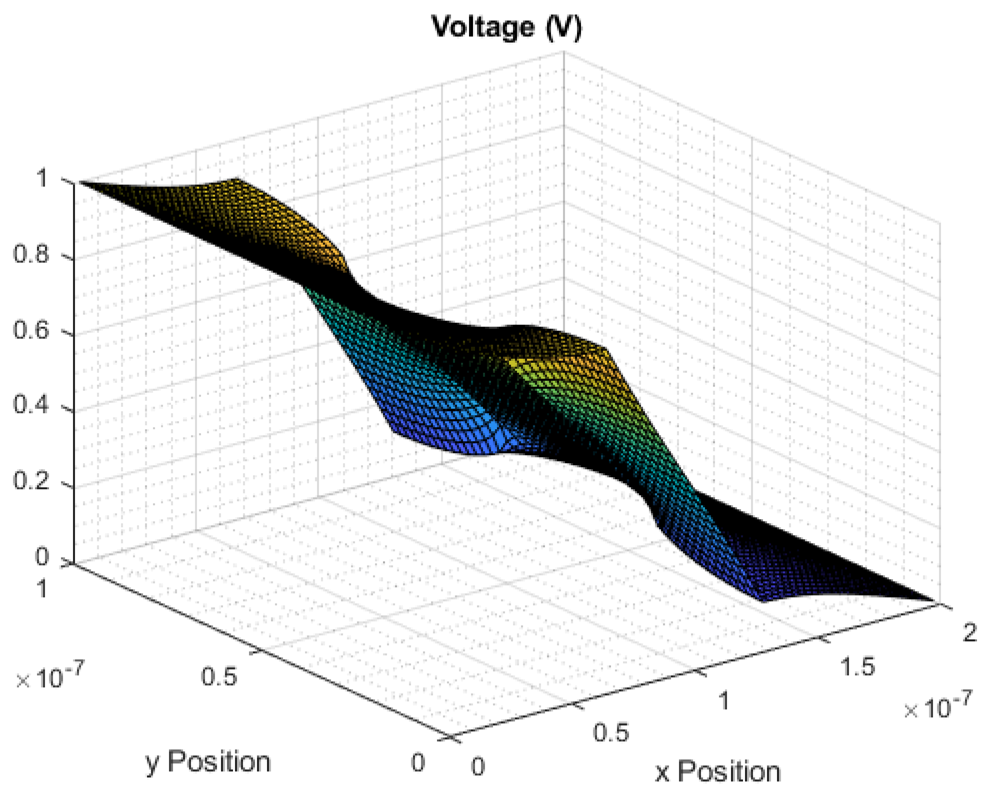
Question 2

```
if Q2 == true

v0 = 1;
L = xlimits(2)- xlimits(1);
W = ylimits(2)- ylimits(1);
nx =75;
ny= 50;
wb = W/3;
lb = L/3;
map = @(j,i) j + (i-1)*ny;
sigma = ones(ny,nx);
for i = 1:nx
if i > (nx/L)*(L-lb)/2 && i < (nx/L)*(L+lb)/2
    for j = 1:ny
        if j < (ny/W)*wb || j> ceil((ny/W)*(W-wb))
           sigma(j,i) = 0.01;
        end
    end
end
end
```

```
for i = 1:nx
for j = 1:ny
    n = map(j,i);
    if i==1
        B(n) = v0;
        G(n,:) = 0;
        G(n,n) = 1;
    elseif i == nx
        B(n) = 0;
        G(n,:) = 0;
        G(n,n) = 1;
    elseif j ==1
        up = (sigma(j,i) + sigma(j+1,i))/2;
        left = (sigma(j,i) + sigma(j,i-1))/2;
        right = (sigma(j,i) + sigma(j,i+1))/2;

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

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

        G(n,n) = -(up+left+right+down);
        G(n,map(j,i-1)) = left;
        G(n,map(j,i+1)) = right;
        G(n,map(j+1,i)) = up;
        G(n,map(j-1,i)) = down;
    end
end
end
E = G\B';
d = zeros(ny,nx);
for i = 1:nx
    for j = 1:ny
        n = map(j,i);
        d(j,i) = E(n);
    end
end
```

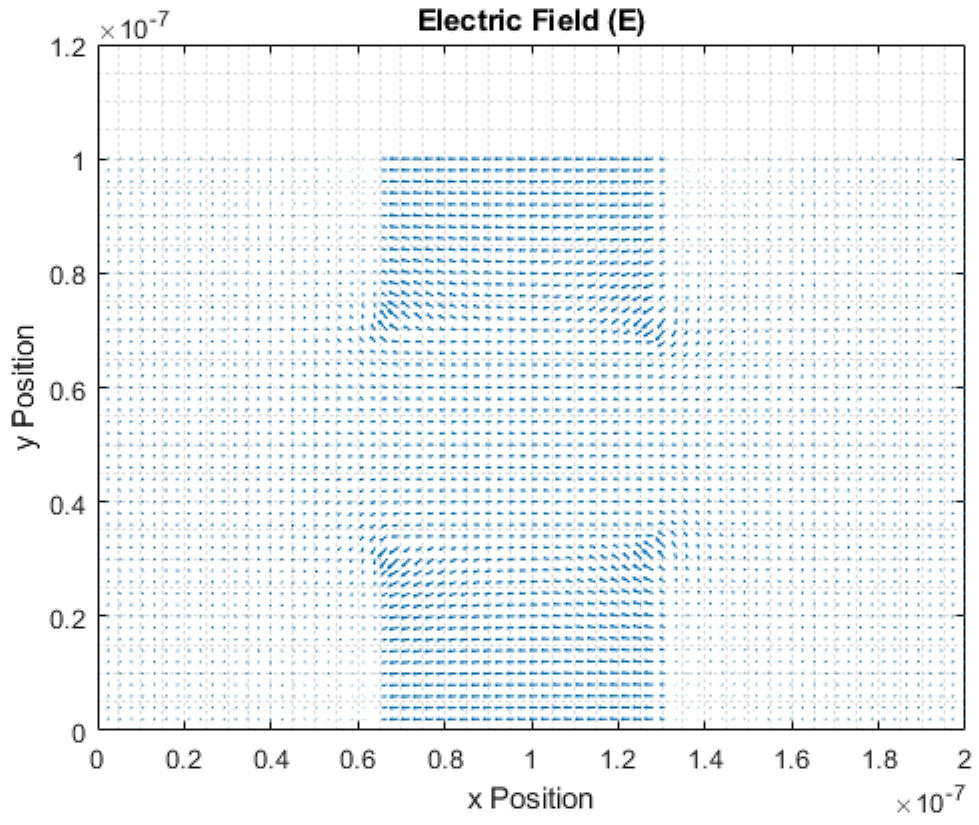
A - Surface Plot of the bottleneck Voltage

```
figure
[X, Y] = meshgrid(L/nx:L/nx:L, W/ny:W/ny:W);
surf(X, Y, d);
grid minor;
title('Voltage (V)');
xlabel('x Position');
ylabel('y Position');
```



B - Electric Field Vector Plot

```
[Ex, Ey] = gradient(d, L/nx);
figure
quiver(X,Y,Ex, Ey);
title('Electric Field (E)');
grid minor;
xlabel('x Position');
ylabel('y Position');
```



```
end
```

Question 3

```
if Q3 == true

NumParticles = 100000;

xboxLim = [.4*(xlimits(2)-xlimits(1)), .6*(xlimits(2)-xlimits(1))];
yboxLim1 = [ylimits(1), .4*(ylimits(2)-ylimits(1))];
yboxLim2 = [.6*(ylimits(2)-ylimits(1)), ylimits(2)];
xbox = xboxLim([1 1 2 2 1]);
ybox1 = yboxLim1([1 2 2 1 1]);
ybox2 = yboxLim2([1 2 2 1 1]);
endtime = Timestep*1000;
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,:) =  xlimits(1) + x(1,:).*(xlimits(2) - xlimits(1));
```

```
y(1,:) =  ylimits(1) + y(1,:).*(ylimits(2) - ylimits(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)) = xlimits(1) + rand*(xlimits(2) - xlimits(1));
    y(1,ind(counter)) = ylimits(1) + rand*(ylimits(2) - ylimits(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,:) = ((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,:));

DiffusionBarrierProbability = 0.2;
```

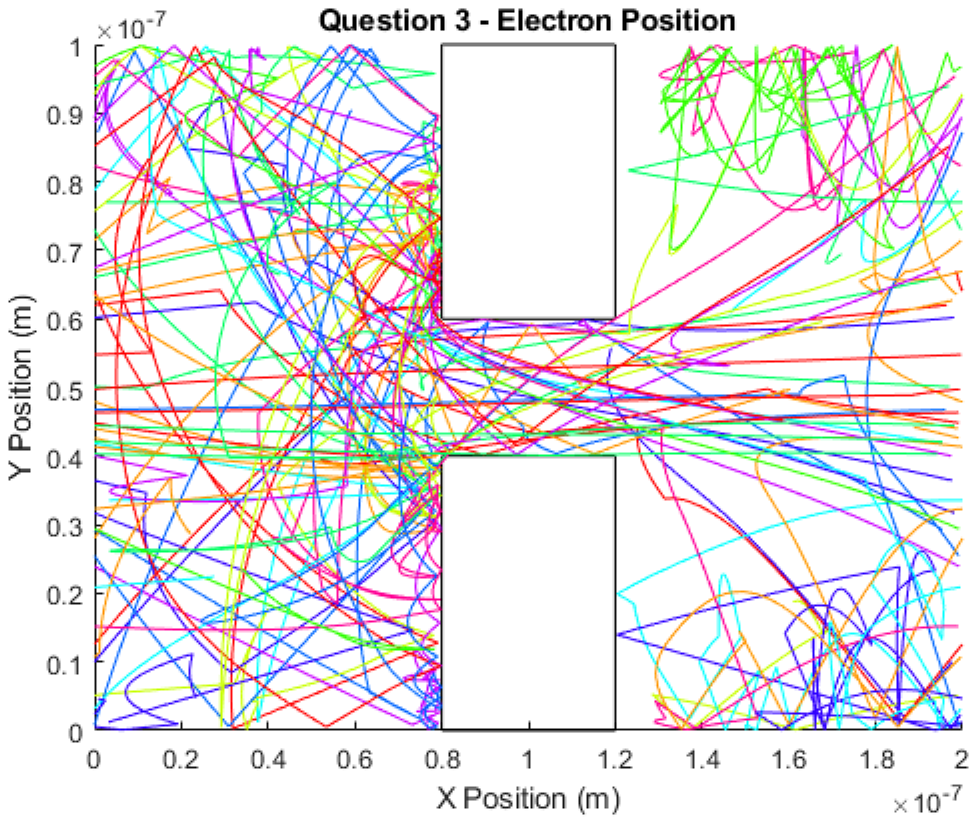
A - Plot of Particle Trajectories

```
figure;
title('Question 3 - Electron Position');
xlabel('X Position (m)');
ylabel('Y Position (m)');
xlim([xlimits(1), xlimits(2)]);
ylim([ylimits(1), ylimits(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 < 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)
            if rand > DiffusionBarrierProbability
            y(2,kt) = -y(2,kt);
            else
                if y(1,kt) +  y(2,kt)*Timestep < ylimits(1)
                    x(3,kt) = rand*pi;
                elseif y(1,kt) +  y(2,kt)*Timestep > ylimits(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)
```

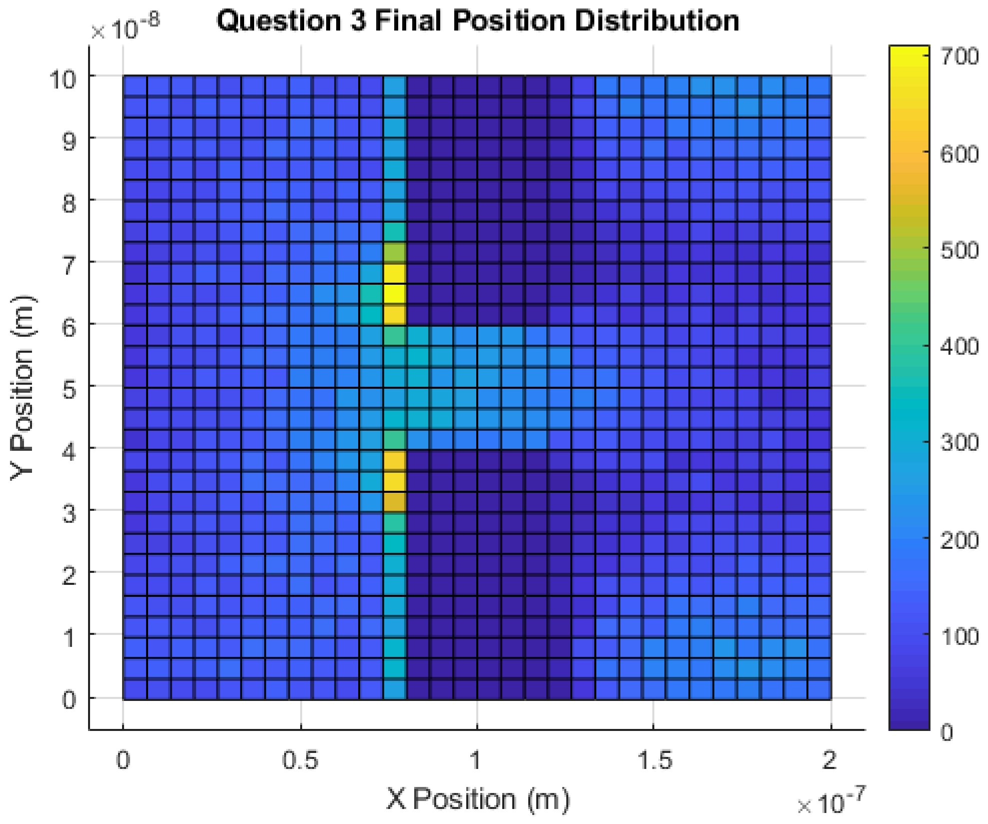
```
          [xinter1, xinter2, yinter1, yinter2]  =
BoxIntercept( x(1,kt), y(1,kt), x(1,kt) +  x(2,kt)*Timestep,
y(1,kt) +  y(2,kt)*Timestep, xboxLim(1), xboxLim(2), ylimits(1),
yboxLim1(2));
          if rand > DiffusionBarrierProbability
          if xinter1 || xinter2
               x(2,kt) = - 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
   xpo = round(nx*x(1,kt)/(xlimits(2)-xlimits(1)));
   ypo = round(ny*y(1,kt)/(ylimits(2)-ylimits(1)));
   if xpo <= 0
       xpo = 1;
   end
   if ypo <= 0
       ypo = 1;
   end
   if xpo >= 51
```

```
            xpo = 50;
        end
        if ypo >= 51
            ypo = 50;
        end
        x(2,kt) = x(2,kt) + Ex(ypo,xpo)*Colomb/mn*Timestep;
        y(2,kt) = y(2,kt) + Ey(ypo,xpo)*Colomb/mn*Timestep;
        end
        hold on;

end
```



B - Density Plot

```
figure;
Z = [transpose(x(1,:)), transpose(y(1,:))];
hist3(Z, 'Nbins',[30 30], 'CdataMode', 'auto');
hold on;
colorbar;
view(2);
grid on;
title('Question 3 Final Position Distribution');
xlabel('X Position (m)');
ylabel('Y Position (m)');
hold off;
```



The electrons have a dominate flow in one direction. This causes the electrons to be accelerated, the electrons will continue to acceeelerate in one direction and be stuck at the bottleneck until they pass through the small hole this causes a build up on one side of the bottleneck and also causes the passage way of the bottleneck to be more densly populated by electrons. On the other side of the bottleneck there is a lower density as the electrons are allowed to spread out and not be restricted.

```
end
```

C - More Accurate Simulation

Allowing electrons to flow through the bottleneck would be more accurate as it has higher resistivity but doesn't stop all electrons from passing through it

The probability of diffusion is set to be a given value by the user, having a more accurate diffusion rate would also produce more accurate simluations

In materials there are always impurities, allowing there to be random impurities that cause greater scattering at that specific point, or diffusion or a higher restivity would provide a more accurate simulation.

Currently this is a two dimensional example, using a three dimensional example would be more accurate.

Published with MATLAB® R2018a