Part 3

```
% In this part of the assignment, the Finite-Difference method was use
 to
% provide a field for the Monte-Carlo bottle-neck simulation.

q_0 = 1.60217653e-19;              % electron charge
m_0 = 9.10938215e-31;              % electron mass
kB = 1.3806504e-23;                % Boltzmann constant
deltat = 0.2e-12;                  % mean time between collisions
mn = 0.26*m_0;                     % effective mass of electrons
%setting up dimensions and matrices
nx = 100;
ny = 200;
G = sparse(nx*ny);
Op = zeros(1, nx*ny);


Sigmatrix = zeros(nx, ny);    % a sigma matrix is required for this
 part
Sig1 = 1;                     % sigma value given outside the box
Sig2 = 10^-2;                 % sigma value given inside the box

%The box will be difined using a 1x4 matrix containing it's dimensions
box = [nx*2/5 nx*3/5 ny*2/5 ny*3/5];

for i = 1:nx

    for j = 1:ny


        if i > box(1) && i < box(2) && (j < box(3)||j > box(4))
            Sigmatrix(i, j) = Sig2;

        else
            Sigmatrix(i, j) = Sig1;


        end
    end
end

% Filling in G matrix with corresponding bottleneck conditions
for x = 1:nx

    for y = 1:ny

        n = y + (x-1)*ny;
        nposx = y + (x+1-1)*ny;
        nnegx = y + (x-1-1)*ny;
        nposy = y + 1 + (x-1)*ny;
        nnegy = y - 1 + (x-1)*ny;
```

```
        if x == 1

            G(n, :) = 0;
            G(n, n) = 1;
            Op(n) = 1;

        elseif x == nx

            G(n, :) = 0;
            G(n, n) = 1;
            Op(n) = 0;

        elseif y == 1

            G(n, nposx) = (Sigmatrix(x+1, y) + Sigmatrix(x,y))/2;
            G(n, nnegx) = (Sigmatrix(x-1, y) + Sigmatrix(x,y))/2;
            G(n, nposy) = (Sigmatrix(x, y+1) + Sigmatrix(x,y))/2;
            G(n, n) = -(G(n,nposx)+G(n,nnegx)+G(n,nposy));

        elseif y == ny

            G(n, nposx) = (Sigmatrix(x+1, y) + Sigmatrix(x,y))/2;
            G(n, nnegx) = (Sigmatrix(x-1, y) + Sigmatrix(x,y))/2;
            G(n, nnegy) = (Sigmatrix(x, y-1) + Sigmatrix(x,y))/2;
            G(n, n) = -(G(n,nposx)+G(n,nnegx)+G(n,nnegy));

        else

            G(n, nposx) = (Sigmatrix(x+1, y) + Sigmatrix(x,y))/2;
            G(n, nnegx) = (Sigmatrix(x-1, y) + Sigmatrix(x,y))/2;
            G(n, nposy) = (Sigmatrix(x, y+1) + Sigmatrix(x,y))/2;
            G(n, nnegy) = (Sigmatrix(x, y-1) + Sigmatrix(x,y))/2;
            G(n, n) = -(G(n,nposx)+G(n,nnegx)+G(n,nposy)+G(n,nnegy));

        end
    end
end


%Voltage matrix calculation
Voltage = G\Op';


sol = zeros(ny, nx, 1);
for i = 1:nx
    for j = 1:ny
        n = j + (i-1)*ny;
        sol(j,i) = Voltage(n);
    end
end


[elecx, elecy] = gradient(sol);
```

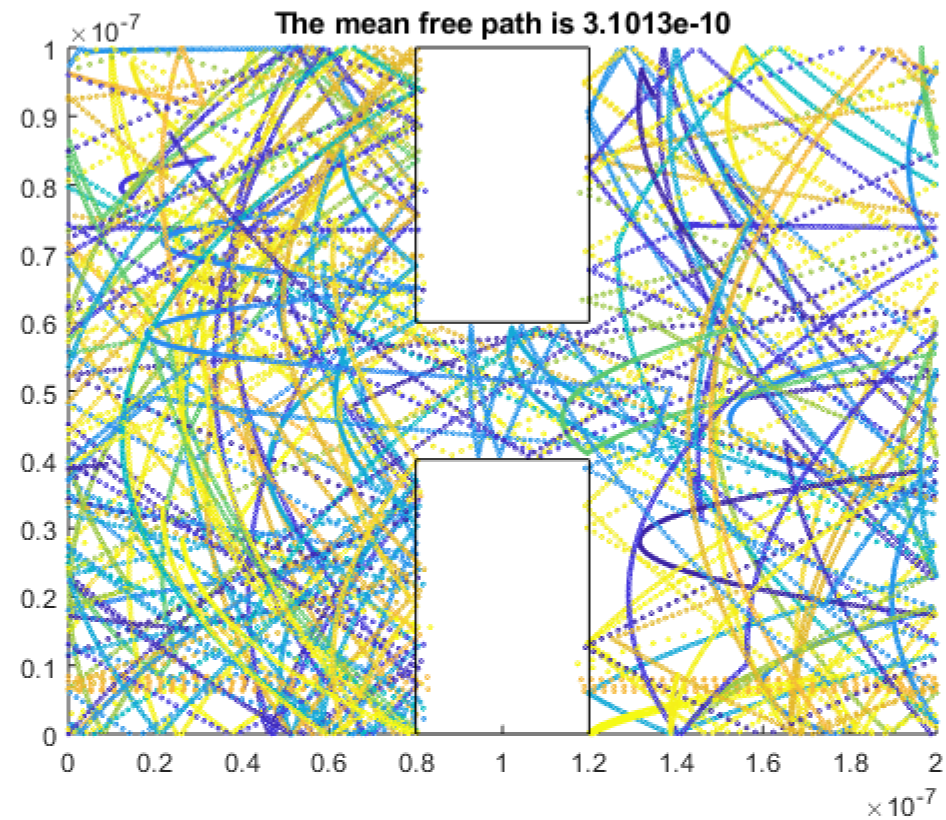
```
numofelec = 1000;              %current numbers of electrons t be
 simulated
T = 300;                       %temperature in kelvin

dt = 1;
%Assign each particle with the fixed velocity given by vth but give
 each one a
%random direction.

vth = sqrt((kB*T)/mn);
%Spatial Boundaries

Length = 200;
Width = 100;


    %I am going to represent the location of each electron using
 vectors

% x = randi([1 Length], 1, numofelec)*1e-9;       %initializing x
% y = randi([1 Width], 1, numofelec)*1e-9;        %initializing y

x = randi([1 Length], 1, numofelec)*1e-9;       %initializing x
y = randi([1 Width], 1, numofelec)*1e-9;        %initializing y
    %top side of lower rectangle
    for it=1:1:numofelec

        %moving spawned electrons outside of rectangles

        if x(1,it) >=(80e-9) && x(1,it) <= (120e-9) && y(1,it)<=
 (40e-9)
               x(1,it) = x(1,it) + randi([45 80], 1,1)*1e-9;
        end

        if x(1,it) >=(80e-9) && x(1,it) <= (120e-9) && y(1,it)>=
 (60e-9)
               x(1,it) = x(1,it) - randi([45 80], 1,1)*1e-9;
        end

    end

    %now we have position vectors for the x and y positions of each
    %electron. Need to create vectors for vy and vx. Remember that
 each
    %electron has a rand angle to start with, but same velocity vth.

angles = randi([0 360], 1, numofelec);
v_x = zeros(1, numofelec);
v_y = zeros(1, numofelec);

v_x = vth*cos(angles);
v_y = vth*sin(angles);

%%%%%%%%%%%%%%% NEW ADDITIONS FOR ASSIGNMENT 3%%%%%%%%%%%%%%%%
```

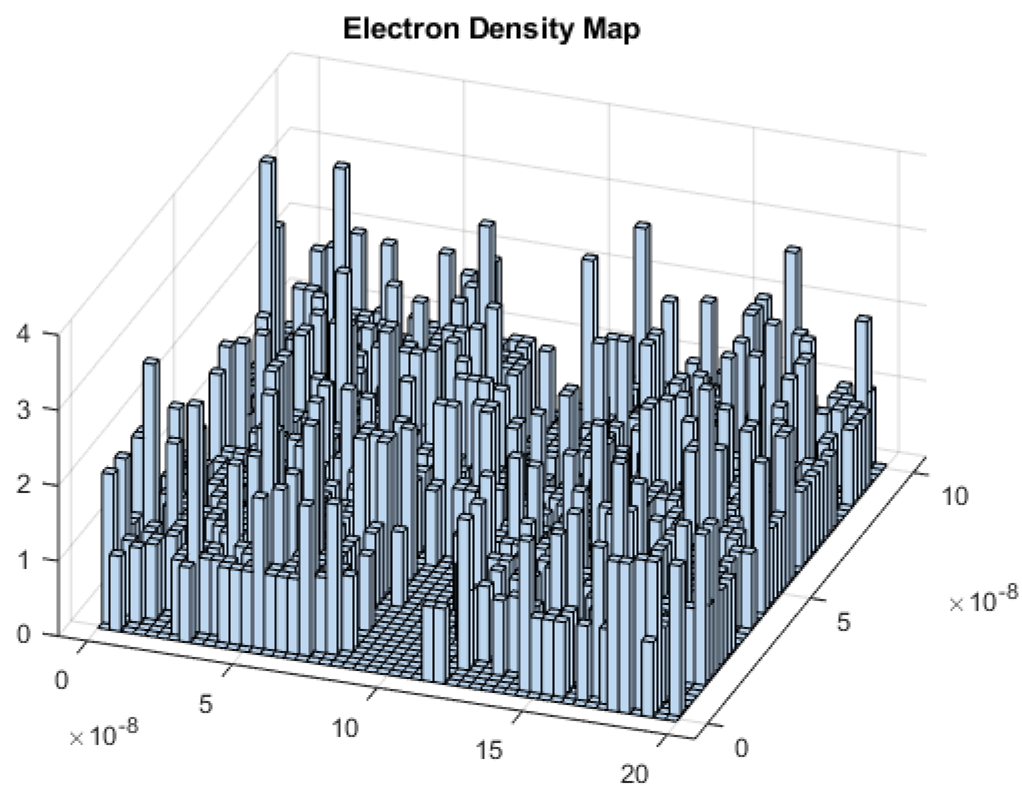
```
Force_x = -elecx*q_0;         %creates a vector containing forces of
 all electrons in x direction
Force_y = -elecy*q_0;         %creates a vector containing forces of
 all electrons in y direction

a_x = Force_x/mn;        %creates a vector containing all acceleration
 of electrons in x direction
a_y = Force_y/mn;         %creates a vector containing all
 acccleration of electrons in y direction


    %scatter
    pscat = 1 - exp(-1e-14/(1e-11*0.2));
    pscatvector = ones(1,numofelec)*pscat;

    %simulating 1000 electrons, but plotting only 20
    colorarray= rand(1,20);
for time= 1:dt:1000

    for i = 1:1:numofelec
        if(x(i) > 199e-9)
            Dim_x(i) = 199;
            else

            Dim_x(i)= ceil(x(i)*10^9);
        end
         if (x(i) < 1e-9)
                Dim_x(i) = 1;

         end



        if(y(i) > 99e-9)
            Dim_y(i) = 99;
        else


            Dim_y(i) = ceil(y(i)*10^9);
        end

        if (y(i) < 1e-9)
                Dim_y(i) = 1;

        end



        % Adding acceleration of the electric field into the electrons
        v_x = v_x + a_x(Dim_x(i), Dim_y(i))*(3e-10);
        v_y = v_y + a_y(Dim_x(i), Dim_y(i))*(3e-10);

    end
```

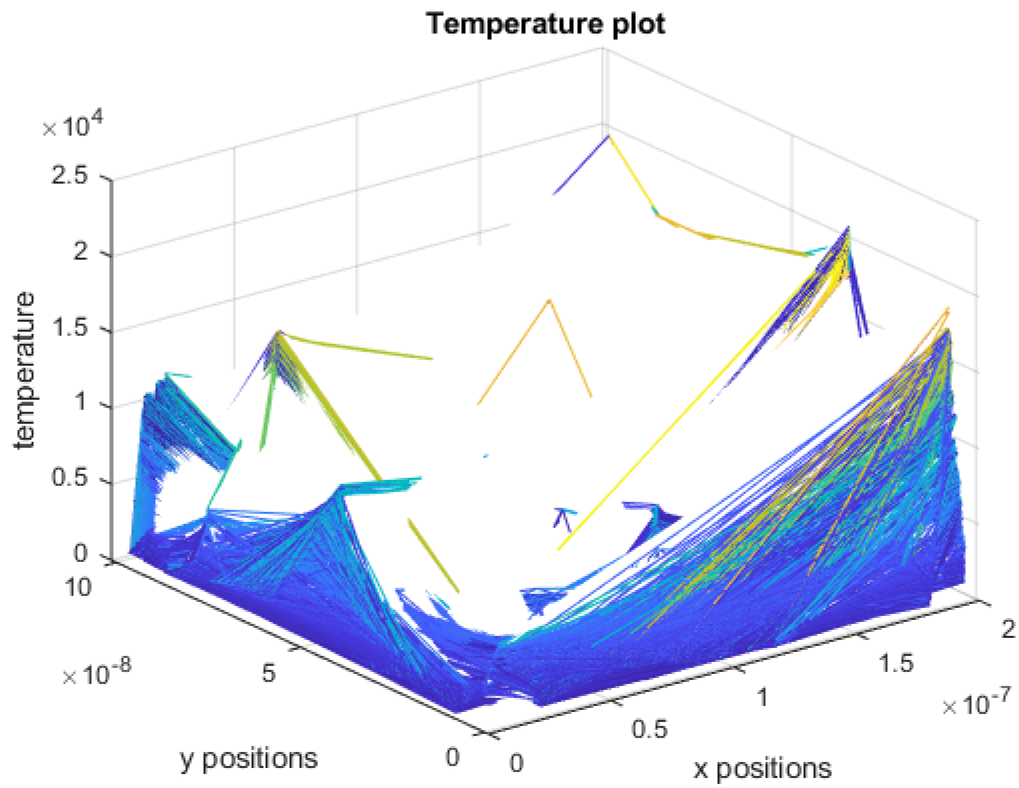
```
%         % Adding acceleration of the electric field into the
 electrons
%         v_x = v_x + a_x(Dim_x(i), Dim_y(i))*dt*1e-15;
%         v_y = v_y + a_y(Dim_x(i), Dim_y(i))*dt*1e-15;
%     end

    random = rand(1,numofelec);

    %all electrons with higher probabilities
    new = random < pscat;

    %all electrons with lower probabilities
    new2 = random >= pscat;


    rand_v_x = zeros(1,numofelec);
    rand_v_y = zeros(1,numofelec);

   for i = 1:1:numofelec
     r1 = randi([1 numofelec], 1,1);
     r2 = randi([1 numofelec], 1,1);
        rand_v_x(1,i) = v_x(1,r1);
        rand_v_y(1,i) = v_y(1,r2);
   end
        %all electrons with lower probabilities will stay the same
   v_x = v_x.*new2;
   v_y = v_y.*new2;

   rand_v_x=rand_v_x.*new;
   rand_v_y=rand_v_y.*new;

   v_x = v_x+rand_v_x;
   v_y = v_y+rand_v_y;

       rb1 = ( x > 80e-9 & x < 120e-9) & y < 40e-9;
    rb0 = rb1 == 0;
    rb1 = -1 * rb1;

    f = rb1 + rb0;

    v_x = v_x .* f;


    rb1 = ( x > 80e-9 & x < 120e-9) & (y < 41e-9 & y >= 40e-9);
    rb0 = rb1 == 0;
    rb1 = -1 * rb1;

    f = rb1 + rb0;

    v_y = v_y .* f;

    %tempFinalLower = x .* y
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%Dealing with the upper rectangle%%%%%%%
rb1 = ( x > 80e-9 & x < 120e-9) & y > 60e-9;
rb0 = rb1 == 0;
rb1 = -1 * rb1;

f = rb1 + rb0;

v_x = v_x .* f;


rb1 = ( x > 80e-9 & x < 120e-9) & (y >59e-9 & y < 60e-9);
rb0 = rb1 == 0;
rb1 = -1 * rb1;

f = rb1 + rb0;

v_y = v_y .* f;
%
dx = v_x*dt*1e-15*5;
dy = v_y*dt*1e-15*5;

x = x + dx;
y = y + dy;

%if y is greater than 200
temp = y>=Width*1e-9;
temp1 = y<Width*1e-9;

temp = temp*(-1);

temphigher = temp + temp1;

v_y = temphigher.*v_y;

 %if y is less than 100
temp2 = y>=0;
temp3 = y<0;

temp3 = temp3*(-1);
templower = temp3 + temp2;
v_y = templower.*v_y;



%if x greater than 200
  temp5 = x<200*1e-9;

  x = x .* temp5;

  %if x is less than 0
  temp4 = x< 0;
  temp4 = temp4*200*1e-9;
```

```matlab
    %temp4 = temp4*200*1e-9;
    x = x + temp4;

     %average thermal velocity
     v_avg = mean(sqrt((v_x.^2)+(v_y.^2)));
     v_matrix = sqrt((v_x.^2)+(v_y.^2));
     T_avg = (mn*(v_avg^2))/kB;

     T_matrix = (mn*(v_matrix.*v_matrix))/kB;

     %mean free path

     mfp = (10^-15)*(v_avg);

     %setting up plot for 20 electrons
     for q =1:1:20
         plotx(q) = x(q);
         ploty(q) = y(q);
     end


     figure(1)
     scatter(plotx,ploty,3,colorarray);
     axis([0 200*10^-9 0 100*10^-9])
     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]);
     title(['The mean free path is ', num2str(mfp)]);
    % pause(0.000000000000000000000000000000000000001)
     hold on
end
%PART E electron density map and temperature map
dens_mat = [x(:) y(:)];
figure(2)
hist3(dens_mat(:,1:2) ,[50 50]);
view([20 45]);
title("Electron Density Map")

%temperature plot
    [X,Y] = meshgrid (x' , y');
    f1 = scatteredInterpolant(x',y',T_matrix');
    Z = f1(X,Y);
    figure (3);
    mesh(X,Y,Z);
    title('Temperature plot')
    xlabel('x positions')
    ylabel('y positions')
    zlabel('temperature')
```


The mean free path is 3.1013e-10
×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
4
3
2
1
0
0
5
10
15
20
×10^-8
0
5
10
×10^-8



Discussion

```
% Part B) from the density and temperature plots, we can observe the
% effects of the 0.8 field. this field cause most electrons to be on
 the
% left most side of the plots, because this is the direction the field
% is pushing them. Notice there are no electrons in the bottleneck
% Part C) To make this simulation more accurate, we could increase the
% mesh of the G matrix. This would yield a more accurate electric
 field
% strength and acceleration each electron experiences.
```

Published with MATLAB® R2018b