Table of Contents

Part 2 1
Discussion part A 9
part C 9

Part 2

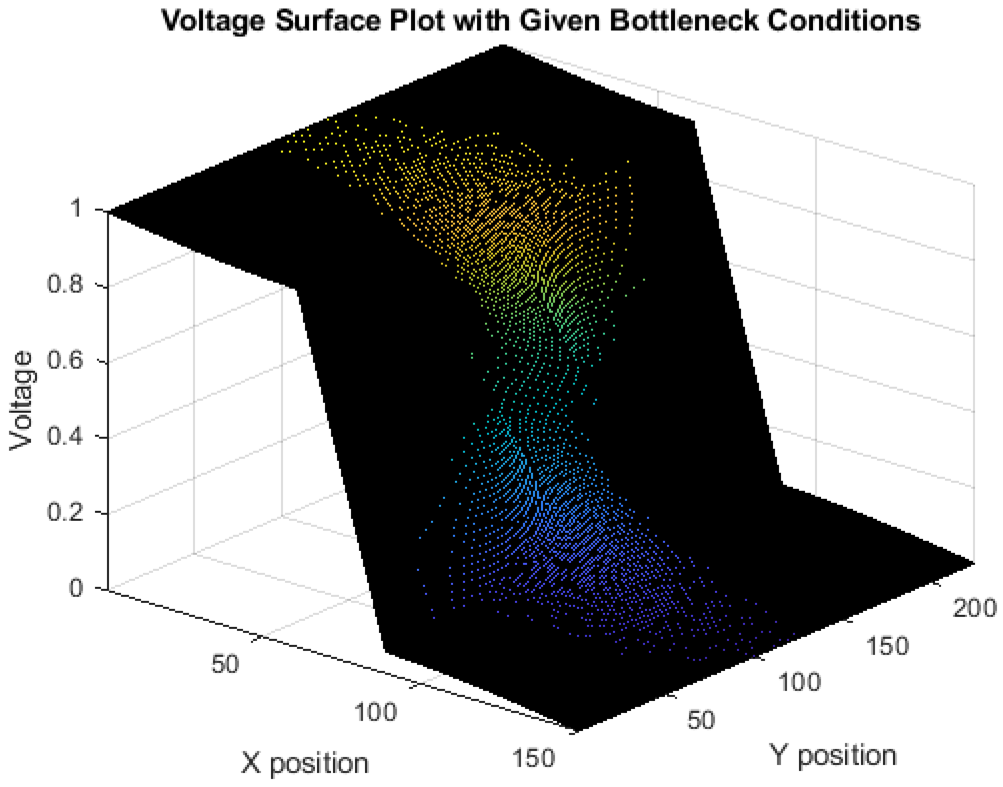
Use of Assignment 2

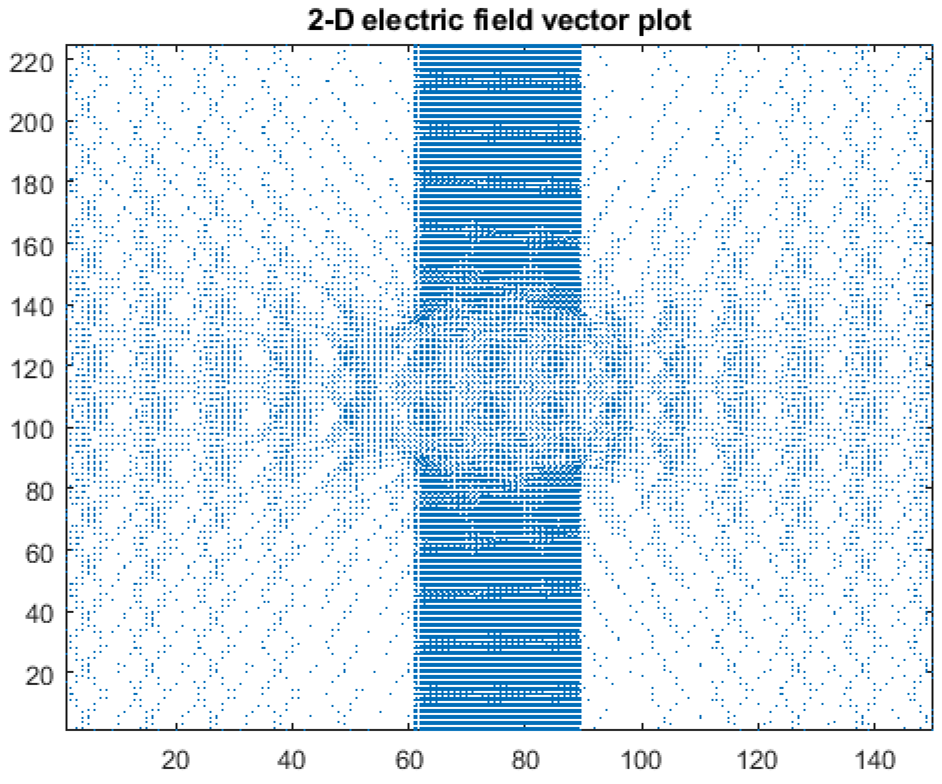
```
clear
close all

%Parameters defined
Length = 150;
Width = (3/2)*Length;
G = sparse(Length*Width);
F = zeros(1, Length*Width);


SigMap = zeros(Length, Width);    % a sigma matrix is required for
 this part
SigOut = 1;                      % sigma value given outside the box
SigIn = 10^-2;                  % sigma value given inside the box

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

for x = 1:Length

    for y = 1:Width


        if x > box(1) && x < box(2) && (y < box(3)||y > box(4))
            SigMap(x, y) = SigIn;

        else
            SigMap(x, y) = SigOut;


        end
    end
end

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

    for y = 1:Width

        n = y + (x-1)*Width; % current posistion
        nxp = y + (x)*Width;
        nxm = y + (x-2)*Width;
        nyp = (y+1) + (x-1)*Width;
        nym = (y-1) + (x-1)*Width;

        if x == 1

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

        elseif x == Length

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

        elseif y == 1

            G(n, nxp) = (SigMap(x+1, y) + SigMap(x,y))/2;
            G(n, nxm) = (SigMap(x-1, y) + SigMap(x,y))/2;
            G(n, nyp) = (SigMap(x, y+1) + SigMap(x,y))/2;
            G(n, n) = -(G(n,nxp)+G(n,nxm)+G(n,nyp));

        elseif y == Width

            G(n, nxp) = (SigMap(x+1, y) + SigMap(x,y))/2;
            G(n, nxm) = (SigMap(x-1, y) + SigMap(x,y))/2;
            G(n, nym) = (SigMap(x, y-1) + SigMap(x,y))/2;
            G(n, n) = -(G(n,nxp)+G(n,nxm)+G(n,nym));

        else

            G(n, nxp) = (SigMap(x+1, y) + SigMap(x,y))/2;
            G(n, nxm) = (SigMap(x-1, y) + SigMap(x,y))/2;
            G(n, nyp) = (SigMap(x, y+1) + SigMap(x,y))/2;
            G(n, nym) = (SigMap(x, y-1) + SigMap(x,y))/2;
            G(n, n) = -(G(n,nxp)+G(n,nxm)+G(n,nyp)+G(n,nym));

        end
    end
end


%Voltage matrix calculation
Voltage = G\F';


solVmatrix = zeros(Width, Length, 1);
for x = 1:Length
    for y = 1:Width
        n = y + (x-1)*Width;
        solVmatrix(y,x) = Voltage(n);
    end
end
```

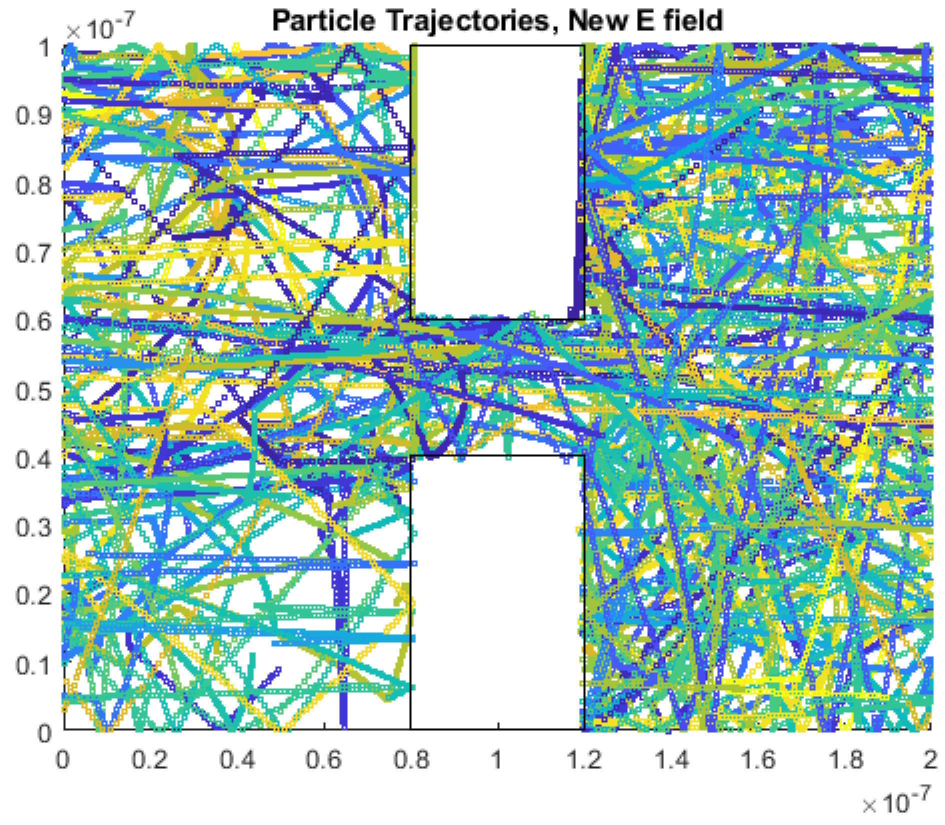
```
%V(x,y) Surface Plot
figure(1)
surf(solVmatrix)
axis tight
xlabel("X position")
ylabel("Y position")
zlabel("Voltage")
view([40 30]);
title("Voltage Surface Plot with Given Bottleneck Conditions")

%The electric field can be derived from the surface voltage using a
%gradient

[E_x, E_y] = gradient(solVmatrix);

%plotting the electric field from the potential using quiver

figure(2)
quiver(-E_x, -E_y);
axis tight
title("2-D electric field vector plot")

% Not sure how to do Part C - this is my attempt at using the field
 fromm
% Q2
%Constants given
width = 200e-9;
height = 100e-9;
iterations = 10;
Noelectron = 50;
K = 1.38064e-23;
T = 300;
mo = 9.1093856e-31;
m = 0.26*mo;
vth = sqrt(2*K*T/m);
Time_step = .002e-12;
q_0 = 1.60217653e-19;             % electron charge


Voltage = 0.1;

Vel_x = zeros(Noelectron,1);
Vel_y = zeros(Noelectron,1);

Pos_x = zeros(Noelectron,1);
Pos_y = zeros(Noelectron,1);

Pos_x(:,1) = width*rand(Noelectron,1);
Pos_y(:,1) = height*rand(Noelectron,1);

%Initialize boundaries for Boxes
xBox = Pos_x >80e-9 & Pos_x<120e-9; %note same box boundary will exist
 for x region
UpperBox = xBox & Pos_y > 60e-9;
```

```
LowerBox = xBox & Pos_y< 40e-9;
Inside = UpperBox | LowerBox;

WidthBox = 40e-9;

%if the particle starts inside, move outside of the box
%movement is not random and should be
for j=1:(length(Inside))

    if(Inside(j))
        Pos_x(j) = Pos_x(j) - WidthBox;
    end
end



force =mean(E_x, 'all')*q_0;         %creates a vector containing
 forces of all electrons


a_elec = force/m;



for n=1:Noelectron
    Vel_x(n,1)= randn(1,1)*vth;
    Vel_y(n,1) = randn(1,1)*vth;
end
AvgV = sqrt(Vel_x.^2 + Vel_y.^2);


%initialize temperature vector
timesteps = 1000;
T_avg_V = zeros(timesteps,1);


%PScattering
colorarray = rand(Noelectron,1);
p_scatter = 1 - exp(-Time_step/0.2e-12);
time = 1:timesteps;
for n= 1:timesteps

    % had to change the time step to get the same trajectory plot as
 in Q1
    %Here adding acceleration of electric field into the electrons
    Vel_x = Vel_x + a_elec*(timesteps*3e-10);

    Pos_x_old = Pos_x;
    Pos_y_old = Pos_y;

    random = rand(Noelectron,1);

    %all electrons with higher probabilities
    new = random < p_scatter;
    %all electrons with lower probabilities
    new2 = random >= p_scatter;


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

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

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

   Vel_x = Vel_x+rand_v_x;
   Vel_y = Vel_y+rand_v_y;

   Pos_x = Pos_x + Vel_x*Time_step;
   Pos_y = Pos_y +Vel_y*Time_step;

   %checking for boundary positions
    idLong = Pos_x>=width;
    Pos_x(idLong) = Pos_x(idLong) - width;
    Pos_x_old(idLong) = 0;

    idShort = Pos_x<=0;
    Pos_x(idShort) = Pos_x(idShort) + width;
    Pos_x_old(idShort) = width;

    %Check for y boundary and correct

    Vel_y(Pos_y>=height) = -1*Vel_y(Pos_y>=height);
    Vel_y(Pos_y<=0) = -1*Vel_y(Pos_y<=0);

    Pos_y(Pos_y>height) = height - (Pos_y(Pos_y>height)-height);
    %Rectangle Boundary Conditions (bottleneck)
    idx = (Pos_x < 1.2e-7) & (Pos_x> 0.8e-7);
    idy = Pos_y >0.6e-7 | Pos_y < 0.4e-7;

    inBox = idx & idy;

    OutX = Pos_x_old < 0.8e-7 | Pos_x_old >1.2e-7; %not used
    BetweenY = Pos_y_old > 0.4e-7 & Pos_y_old < 0.6e-7;

    Vel_y(inBox & BetweenY) = -1*Vel_y(inBox & BetweenY);
    Vel_x(inBox & ~BetweenY) = -1*Vel_x(inBox & ~BetweenY);
```

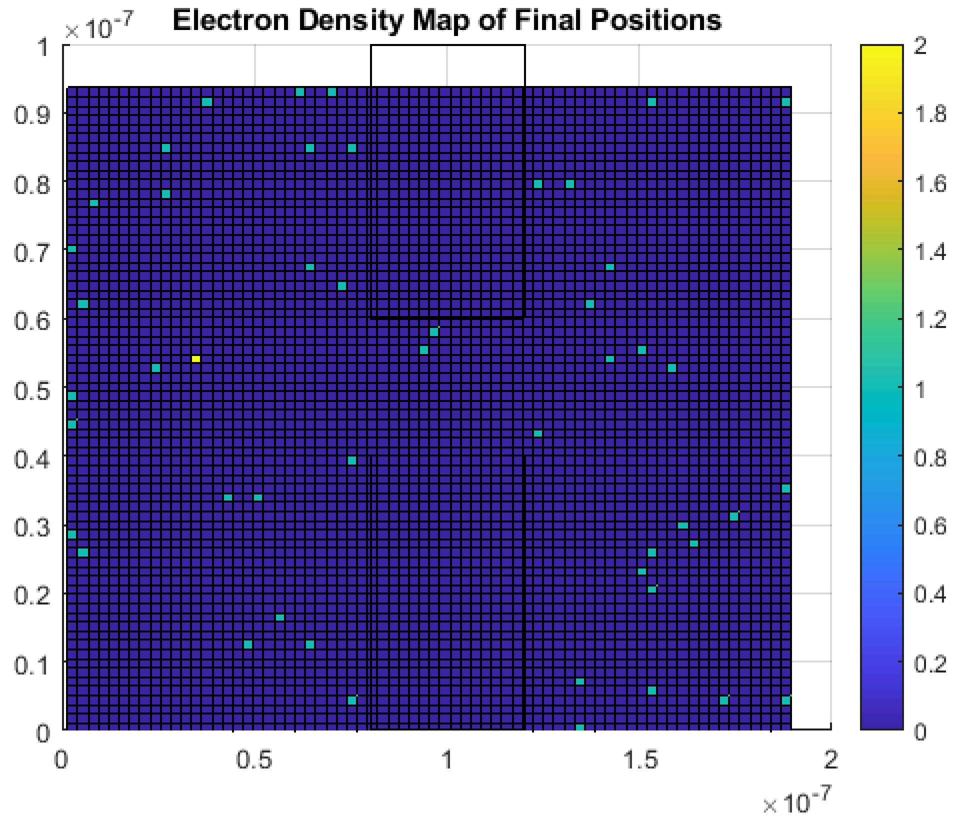
```
%
%
%     T_matrix = (m*(v_matrix.*v_matrix))/K;



    % 2D Trajectory as in Figure 1 in Manual
    figure(3)
    scatter(Pos_x,Pos_y,3,colorarray);
    axis([0 200e-9 0 100e-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('Particle Trajectories, New E field');
    hold on




end
%electron density map - some leaking occurred
figure(4)
hist3([Pos_x Pos_y],'CdataMode','auto', 'Nbins', [70 70]);
axis([0 200e-9 0 100e-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('Electron Density Map of Final Positions')
colorbar
view(2)
```


Voltage Surface Plot with Given Bottleneck Conditions
Voltage
0
0.2
0.4
0.6
0.8
1
X position
50
100
150
Y position
50
100
150
200


2-D electric field vector plot
20
40
60
80
100
120
140
160
180
200
220
20
40
60
80
100
120
140



Particle Trajectories, New E field
×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 of Final Positions
×10^-7
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.5
1
1.5
2
×10^-7
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0

Discussion part A

From the density map, we can see that the electrons are mostly on the right side and that is because the 0.8 field is pushing them in that direction. it was expected that the density map appears different than in Q1

part C

```
%%Part C
%To make the simulation more accurate the effect of forces between the
%elctrons should be considered. THe mesh size should be increased to
 get a
%better E field.
```

Published with MATLAB® R2019b