Table of Contents

Part 1: Electron Modelling

Assignment 1 is modified here without a bottle neck with 0.1V as the applied voltage across the x dimension of the semiconductor.

```
clear all;
close all;

% Program parameters
nx = 200e-9;
ny = 100e-9;
Vx = 0.1; % Voltage across x dimension
Vy = 0; % Voltage applied across y

qe = -1.60217662e-19; % Charge of an electron
Edensity = 1e15*100^2; % Concentration of the electrons (per 1/m^2)
m0 = 9.10938356e-31; % rest mass of electron
m = 0.26*m0; % effective mass of electrons
T = 300; % temperature in K
Bconst = 1.38064852e-23;
vth = sqrt(2*Bconst*T/m);
l = vth*0.2e-12; % Mean free path

pop_size = 40000;
plot_pop = 10;
time_step = ny/vth/100;
iterations = 300;
p_scat = 1 - exp(-time_step/0.2e-12);
v_pdf = makedist('Normal', 'mu', 0, 'sigma', sqrt(Bconst*T/m));
show_movie = 0;
```

The boundaries can be specular or diffusive. Electrons bounce off at random angles if it is diffusive. At specular, the electrons bounce off symmetric angles.

specular (1) or diffusive (0)

```
top_specular = 0;
bottom_specular = 0;
```

Assuming the fields are uniform, the electric field components summed up together gives the total electric field in the region.

```
Ex = Vx/nx;
Ey = Vy/ny;
```

```
Etotal = Ex + Ey;
fprintf('The electric field is %d V/m.\n',Etotal);
```

The electric field is 5.000000e+05 V/m.

The force on each electron is the sum of its individual components.

```
Fx = qe*Ex;
Fy = qe*Ey;
Ftotal = abs(Fx + Fy);
fprintf('The force on each electron is %d N.\n',Ftotal);
```

The force on each electron is 8.010883e-14 N.

Given that F = ma, acceleration can be found using a = F/a.

```
accel = Ftotal/m;
fprintf('The acceleration of each electron is %0.5e m/s^2.\n',accel);
```

The acceleration of each electron is 3.38235e+17 m/s^2.

The current formula is J = vnqNy. v represents the velocity in the x direction, n is the electron concentration, q is the charge of an electron, Ny is the length of y. The change in speed in each direction for one time step.

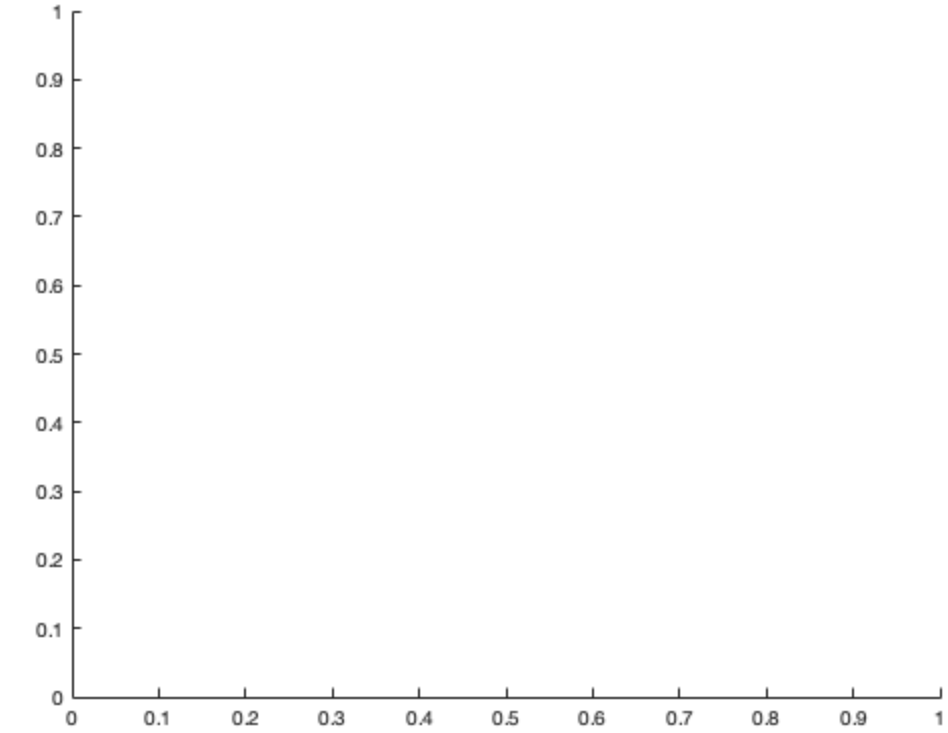
```
dirvx = Fx*time_step/m;
dirvy = Fy*time_step/m;
dirvx = dirvx.*ones(pop_size,1);
dirvy = dirvy.*ones(pop_size,1);
```

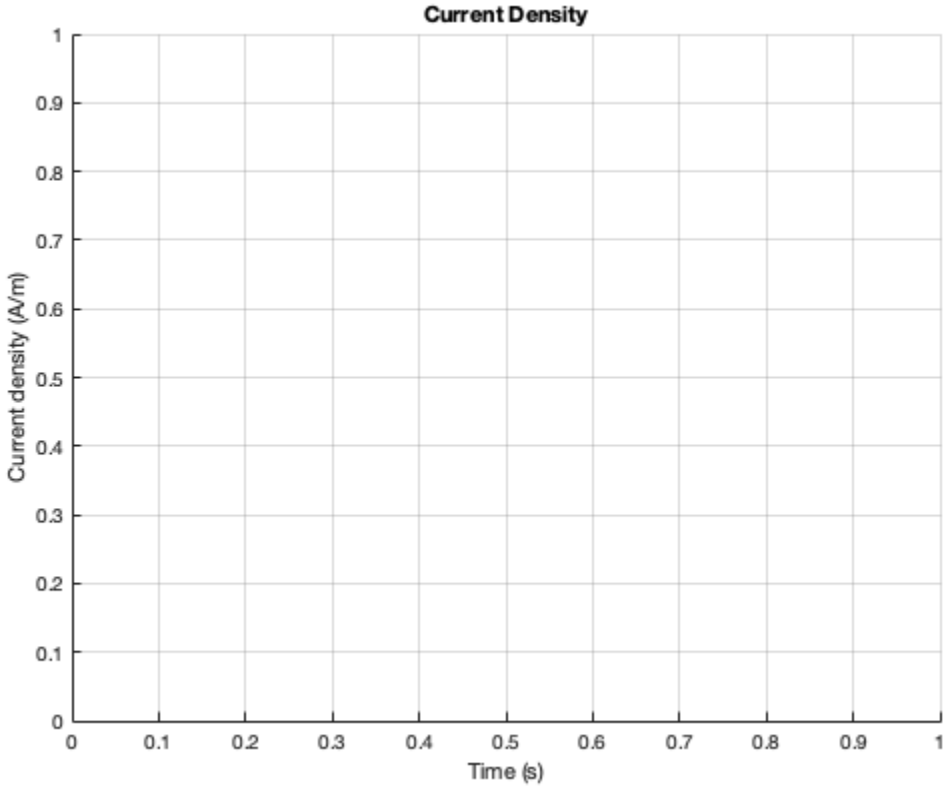
Each row corresponds to electrons with the information [x y vx vy].

```
state = zeros(pop_size, 4);
traj = zeros(iterations, plot_pop*2);
temp = zeros(iterations,1);
J = zeros(iterations,2); % Current density rows represented as [Jx Jy]
```

Initial population

```
for i = 1:pop_size
    angle = rand*2*pi;
    state(i,:) = [nx*rand ny*rand random(v_pdf) random(v_pdf)];
end

temperature_plot = animatedline;

figure(2);
current_plot = animatedline;
title('Current Density');
xlabel('Time (s)');
ylabel('Current density (A/m)');
grid on;
```

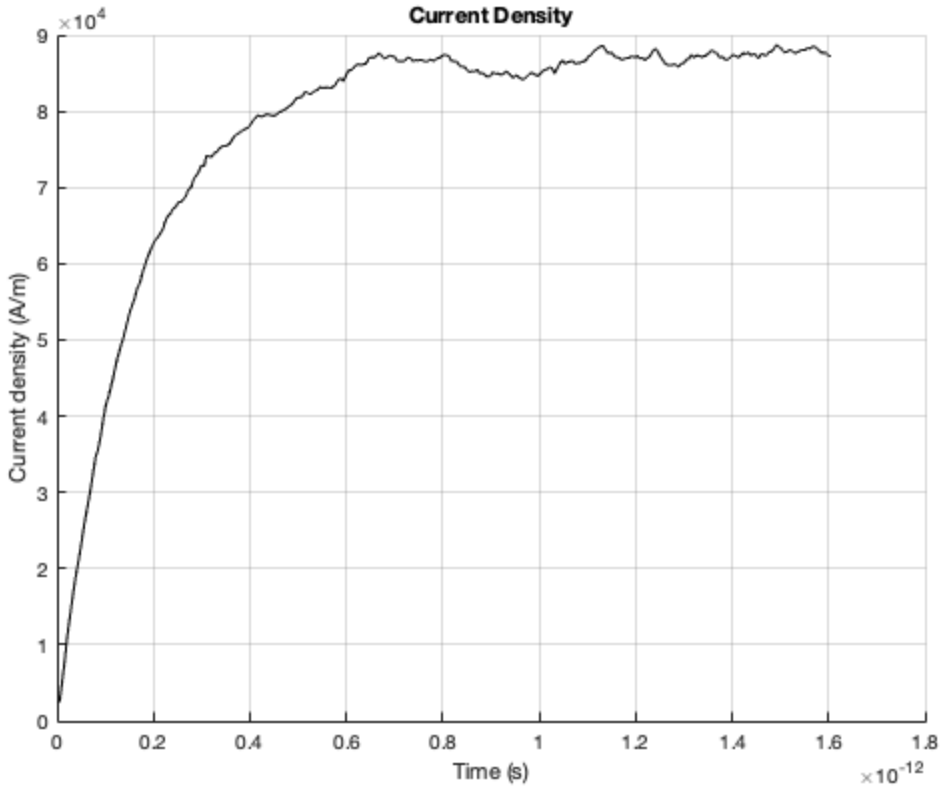

1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

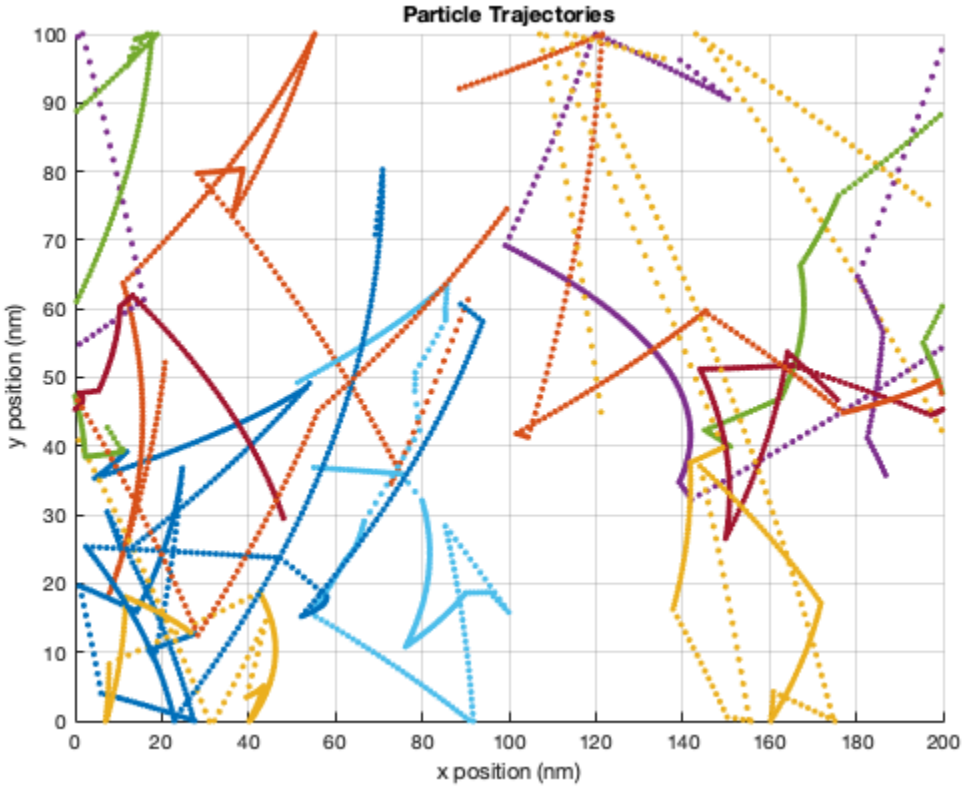

Current Density
Current density (A/m)
Time (s)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Run through the simulation:

```
for i = 1:iterations
    % run though velocities

    state(:,3) = state(:,3) + dirvx;
    state(:,4) = state(:,4) + dirvy;

    % positions
    state(:,1:2) = state(:,1:2) + time_step.*state(:,3:4);

    j = state(:,1) > nx;
    state(j,1) = state(j,1) - nx;

    j = state(:,1) < 0;
    state(j,1) = state(j,1) + nx;

    j = state(:,2) > ny;

    if(top_specular)
        state(j,2) = 2*ny - state(j,2);
        state(j,4) = -state(j,4);
    else % Diffusive
        state(j,2) = ny;
        v = sqrt(state(j,3).^2 + state(j,4).^2);
        angle = rand([sum(j),1])*2*pi;
        state(j,3) = v.*cos(angle);
        state(j,4) = -abs(v.*sin(angle));
    end

    j = state(:,2) < 0;

    if(bottom_specular)
        state(j,2) = -state(j,2);
        state(j,4) = -state(j,4);
    else % Diffusive
        state(j,2) = 0;
        v = sqrt(state(j,3).^2 + state(j,4).^2);
        angle = rand([sum(j),1])*2*pi;
        state(j,3) = v.*cos(angle);
        state(j,4) = abs(v.*sin(angle));
    end

    % Scatter particles
    j = rand(pop_size, 1) < p_scat;
    state(j,3:4) = random(v_pdf, [sum(j),2]);

    % record temp
    temp(i) = (sum(state(:,3).^2) + sum(state(:,4).^2))*m/Bconst/2/
pop_size;

    for j=1:plot_pop
        traj(i, (2*j):(2*j+1)) = state(j, 1:2);
    end
```

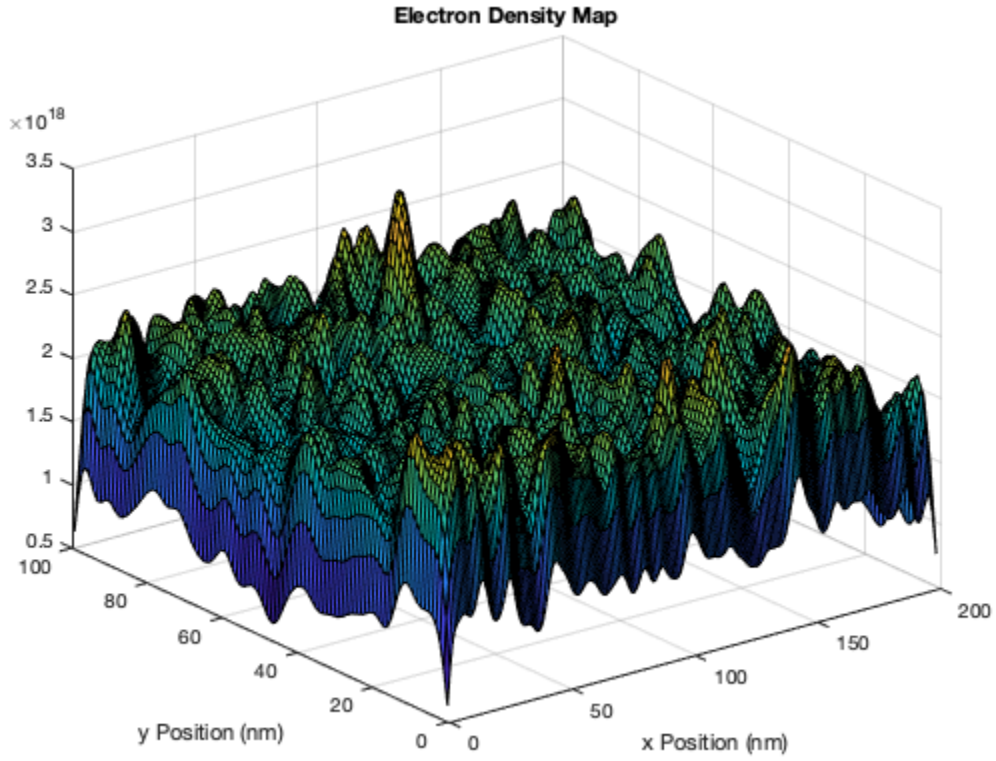
```
    % Calculate and record the current density
    J(i, 1) = qe.*Edensity.*mean(state(:,3));
    J(i, 2) = qe.*Edensity.*mean(state(:,4));

    % Plot temperature and current
    addpoints(temperature_plot, time_step.*i, temp(i));
    addpoints(current_plot, time_step.*i, J(i,1));

    if(show_movie && mod(i,5) == 0)
        figure(1);
        hold off;
        plot(state(1:plot_pop,1)./1e-9,
 state(1:plot_pop,2)./1e-9, 'o');
        axis([0 nx/1e-9 0 ny/1e-9]);
        hold on;
        title('Particle Trajectories');
        xlabel('x position (nm)');
        ylabel('y positiom (nm)');
        pause(0.05);
    end
end

% Show trajectories after the movie is over
figure(1);
title('Particle Trajectories');
xlabel('x position (nm)');
ylabel('y position (nm)');
axis([0 nx/1e-9 0 ny/1e-9]);
grid on;
hold on;

for i=1:plot_pop
    plot(traj(:,i*2)./1e-9, traj(:,i*2+1)./1e-9, '.');
end
```


Current Density
×10^4
9
8
7
6
5
4
3
2
1
0
Current density (A/m)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
Time (s)
×10^-12


Particle Trajectories
100
90
80
70
60
50
40
30
20
10
0
y position (nm)
0
20
40
60
80
100
120
140
160
180
200
x position (nm)

The plot of current over time shows the current starts low and speeds up as the electron accelerates.

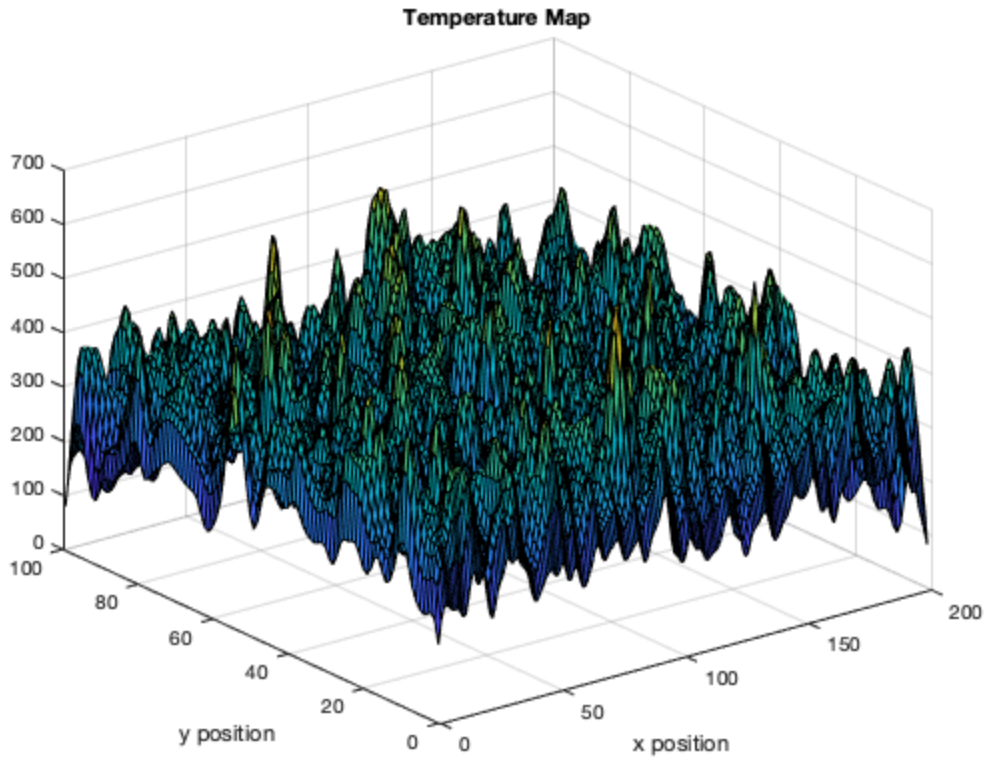
```
Edensity = hist3(state(:,1:2),[200 100])';
N = 20;
sigma = 1.5;
[x y] = meshgrid(round(-N/2):round(N/2), round(-N/2):round(N/2));
f=exp(-x.^2/(2*sigma^2)-y.^2/(2*sigma^2));
f=f./sum(f(:));
figure(3);
Edensity = conv2(Edensity,f,'same');
Edensity = Edensity/(ny./size(Edensity,1)*nx./size(Edensity,2));
surf(conv2(Edensity,f,'same'));
title('Electron Density Map');
xlabel('x Position (nm)');
ylabel('y Position (nm)');
```



The temperature map

```
sum_x = zeros(ceil(nx/1e-9),ceil(ny/1e-9));
sum_y = zeros(ceil(nx/1e-9),ceil(ny/1e-9));
temp_num = zeros(ceil(nx/1e-9),ceil(ny/1e-9));

% velocities of all particles
for i=1:pop_size
    % Find where it belongs:
    x = floor(state(i,1)/1e-9);
    y = floor(state(i,2)/1e-9);
```

```
    if(x==0)
        x = 1;
    end
    if(y==0)
        y= 1;
    end

    % Add its velocity :
    sum_y(x,y) = sum_y(x,y) + state(i,3)^2;
    sum_x(x,y) = sum_x(x,y) + state(i,4)^2;
    temp_num(x,y) = temp_num(x,y) + 1;
end

% calculate the temperatures:
temp = (sum_x + sum_y).*m./Bconst./2./temp_num;
temp(isnan(temp)) = 0;
temp = temp';

N = 20;
sigma = 1.5;
[x y] = meshgrid(round(-N/2):round(N/2), round(-N/2):round(N/2));
f=exp(-x.^2/(2*sigma^2)-y.^2/(2*sigma^2));
f=f./sum(f(:));
figure(4);
surf(conv2(temp,f,'same'));
title('Temperature Map');
xlabel('x position');
ylabel('y position');
```



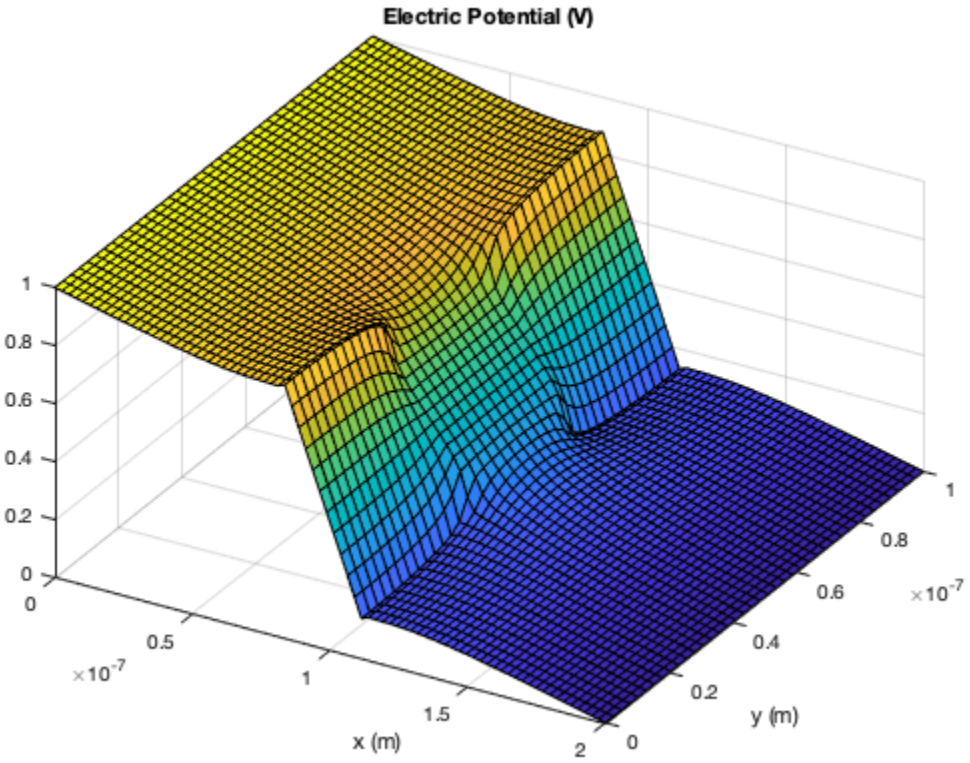
Part 2

Assignment 2 was modified for the bottle neck case. The applied voltage is set to 0.8 V.

```
scale = 100e-9;
V0 = 1;
W = 1;
L = 2;
dx = 0.025; % spacing along x
dy = 0.025; % spacing along y
nx = ceil(L/dx); % points along x
ny = ceil(W/dy); % points along y
dx = L/nx;
dy = W/ny;
Lb = 0.4; % Length of regions
Wb = 0.4; % Width of regions
sig1 = 1;
sig2 = 1e-2;

% Construct the C matrix:
C = sig1.*ones(ny,nx);
Csubtract = zeros(ny,nx);

for x=1:nx
    for y=1:ny
        xx = x*dx;
```

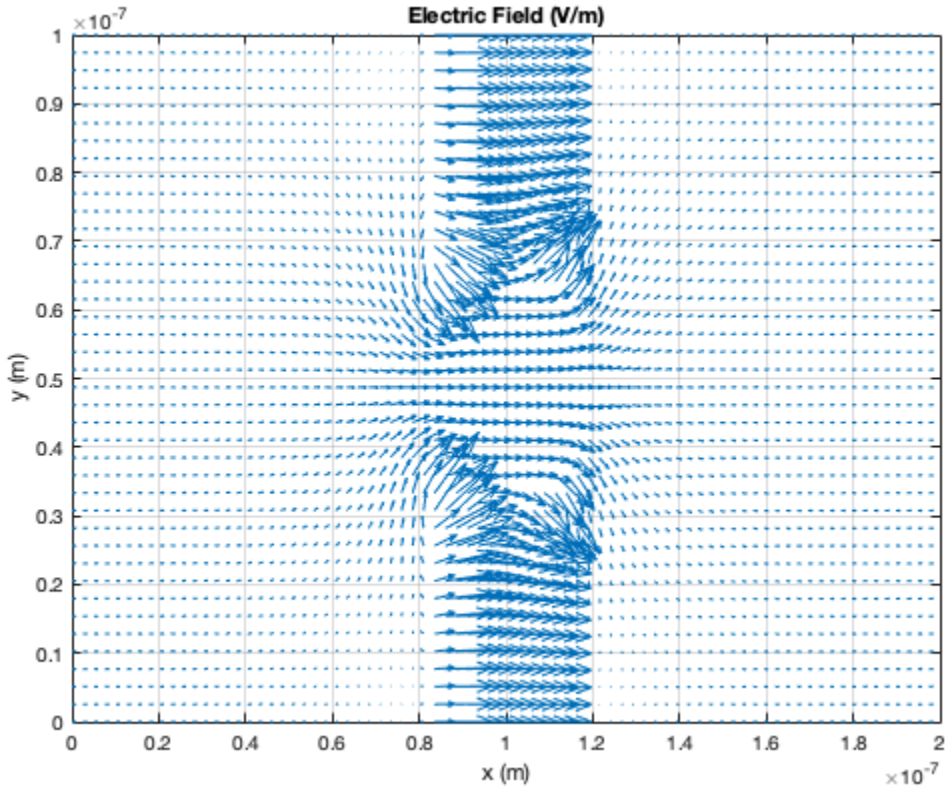
```
        yy = y*dy;

        % high resistivity in regions
        if(xx <= (L+Lb)/2 && xx >= (L-Lb)/2 && (yy >= W-Wb || yy <=
 Wb))
            Csubtract(y,x) = sig1-sig2;
        end
    end
end

Csubtract = imgaussfilt(Csubtract, 1);
C = C - Csubtract;

G = zeros(nx*ny,nx*ny);
F = zeros(nx*ny,1);

dx2 = 1./(dx.^2);
dy2 = 1./(dy.^2);

for x=2:(nx-1)
    for y=2:(ny-1)
        index = mapCoordinate(x,y,nx);

        % Apply the equation derived earlier:
        G(index,index) = -2.*C(y,x).*(dx2 + dy2);
        G(index, mapCoordinate(x+1,y,nx)) = dx2.*(0.25.*(C(y,x+1) -
 C(y,x-1)) + C(y,x));
        G(index, mapCoordinate(x-1,y,nx)) = dx2.*(-0.25.*(C(y,x+1) -
 C(y,x-1)) + C(y,x));

        G(index, mapCoordinate(x,y+1,nx)) = dy2.*(0.25.*(C(y+1,x) -
 C(y-1,x)) + C(y,x));
        G(index, mapCoordinate(x,y-1,nx)) = dy2.*(-0.25.*(C(y+1,x) -
 C(y-1,x)) + C(y,x));
    end
end

% The top and bottom boundaries
for x=2:(nx-1)
    index = mapCoordinate(x,1,nx);
    G(index,index) = 1;
    G(index,mapCoordinate(x,2,nx)) = -1;
    F(index) = 0;

    index = mapCoordinate(x,ny,nx);
    G(index,index) = 1;
    G(index,mapCoordinate(x,ny-1,nx)) = -1;
    F(index) = 0;
end

% The vertical boundaries
for y=1:ny
    index = mapCoordinate(1,y,nx);
    G(index,index) = 1;
```

```
    F(index) = V0;

    index = mapCoordinate(nx,y,nx);
    G(index,index) = 1;
    F(index) = 0;
end

V = G\F;
V = reshape(V,[],ny)';

figure(5);
surf(linspace(0,L.*scale,nx),linspace(0,W.*scale,ny),V);
view(30,45);
xlabel('x (m)');
ylabel('y (m)');
title('Electric Potential (V)');
grid on;
```




```
figure(6);
[Ex,Ey] = gradient(V,dx.*scale,dy.*scale);
Ex = -1.*Ex;
Ey = -1.*Ey;

quiver(linspace(0,L.*scale,nx),linspace(0,W.*scale,ny),Ex,Ey,4);
xlabel('x (m)');
ylabel('y (m)');
title('Electric Field (V/m)');
```

```
axis([0 L.*scale 0 W.*scale]);
grid on;
```



Part 3

```
clear all

numelec = 1000;
numit = 00;
time = zeros(1, numit);
m0 = 9.10938215e-31;
m = 0.26*m0;
Bconst = 1.38064852e-23;
T = 300;
vth = sqrt(2*Bconst*T/m);
dt = (100e-9)/vth/100;

temperature = zeros(1, numit);

x = linspace(0, 200, 400)*10^(-9); %x axis
y = linspace(0, 100, 400)*10^(-9); %y axis
tau = 0.2e-12;
l = vth*tau; %mean free path
Pscat = 1 - exp(-dt/tau);
nx = 200;
ny = 100;
```

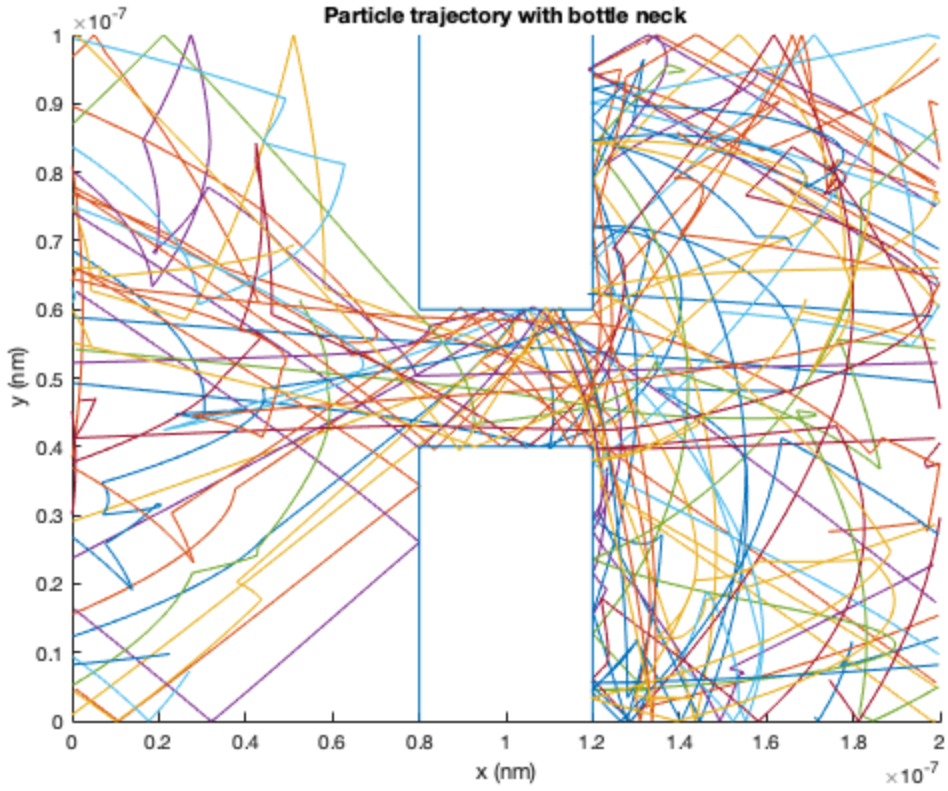
```
dx = 1;
dy = 1;
nx = nx / dx;
ny = ny / dy;

G = sparse(nx*ny, nx*ny);
F = zeros(1, nx*ny);

sig = ones(nx, ny);
for i = 1:nx
    for j = 1:ny
        if j <= (40) || j >= (60)
            if i >= (80) && i <= (120)
                sig(i, j) = 10^(-2);
            end
        end
    end
end

for i = 1:nx
    for j = 1:ny

        n = j + (i - 1) * ny;

        if i == 1
            G(n, :) = 0;
            G(n, n) = 1;
            F(n) = 0.8;
        elseif i == nx
            G(n, n) = 1;
        elseif j == 1
            sigup = (sig(i, j) + sig(i, j+1)) / 2.0;
            sigR = (sig(i, j) + sig(i+1, j)) / 2.0;
            sigL = (sig(i, j) + sig(i-1, j)) / 2.0;
            G(n, n) = -sigup - sigR - sigL;
            G(n, n + 1) = sigup;
            G(n, n + ny) = sigR;
            G(n, n - ny) = sigL;
        elseif j == ny
            sigR = (sig(i, j) + sig(i+1, j)) / 2.0;
            sigL = (sig(i, j) + sig(i-1, j)) / 2.0;
            sigmaLOWER = (sig(i, j) + sig(i, j-1)) / 2.0;

            G(n, n) = -sigup - sigR - sigL;
            G(n, n - 1) = sigmaLOWER;
            G(n, n + ny)= sigR;
            G(n, n - ny)= sigL;
        else
            sigup = (sig(i, j) + sig(i, j+1)) / 2.0;
            sigR = (sig(i, j) + sig(i+1, j)) / 2.0;
            sigL = (sig(i, j) + sig(i-1, j)) / 2.0;
            sigmaLOWER = (sig(i, j) + sig(i, j-1)) / 2.0;

            G(n, n)= -sigup - sigmaLOWER - sigR - sigL;
```

```
            G(n, n + 1) = sigup;
            G(n, n - 1) = sigmaLOWER;
            G(n, n + ny) = sigR;
            G(n, n - ny) = sigL;
        end
    end
end

V2 = zeros(nx, ny);
V = G\F';

for i = 1:nx
    for j = 1:ny
        n = j + (i - 1)*ny;
        V2(i, j) = V(n);
    end
end

[Ey, Ex] = gradient(V2*1e9);
Ex = -Ex;
Ey = -Ey;

numelec = 1000;
numit = 3500;
time = zeros(1, numit);
m0 = 9.10938215e-31; %rest mass of an electron (kg)
m = 0.26*m0; %effective mass of an electron (kg)
tau = 0.2e-12; %mean time between collisions (s)
Bconst = 1.38064852e-23; %boltzmann constant (J/K)
T = 300; %temperature (K)
vth = sqrt(2*Bconst*T/m); %thermal velocity
dt = (100e-9)/vth/500;
l = vth*tau; %mean free path
temperature = zeros(1, numit);
Pscat = 1 - exp(-dt/tau);
x = linspace(0, 200, 400)*10^(-9); %x axis
y = linspace(0, 100, 400)*10^(-9); %y axis

Px = zeros(numelec, numit);
Py = zeros(numelec, numit);

for n = 1 : numelec
    Px(n, 1) = x(randi(400));
    Py(n, 1) = y(randi(400));
    while (Px(n, 1) >= 80e-9 && Px(n, 1) <= 120e-9 && Py(n, 1) >=
 60e-9) || (Px(n, 1) >= 80e-9 && Px(n, 1) <= 120e-9 && Py(n, 1) <=
 40e-9)
        Px(n, 1) = x(randi(400));
        Py(n, 1) = y(randi(400));
    end
end

Vx = zeros(numelec, numit);
Vy = zeros(numelec, numit);
```

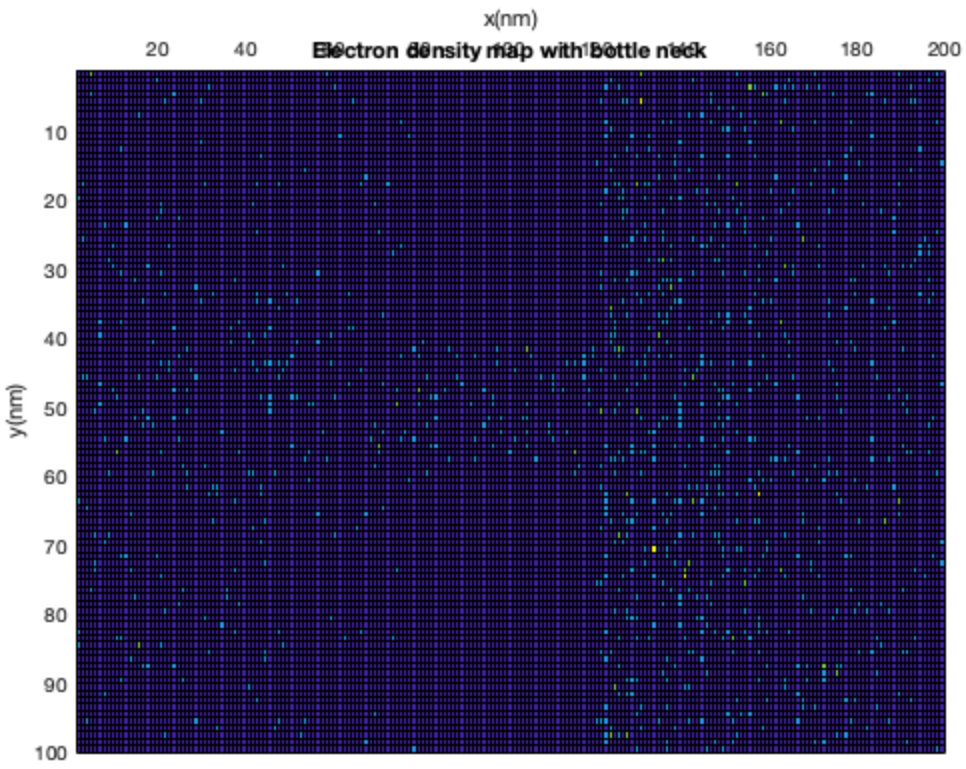
```
xaccel = zeros(numelec, numit);
yaccel = zeros(numelec, numit);

dist = makedist('Normal', 'mu', 0, 'sigma', sqrt(Bconst*T/m));

for k = 1 : numelec
    Vx(k, :) = random(dist);
    Vy(k, :) = random(dist);
    if round(Px(k, 1)*(10^9)) == 0 && round(Py(k, 1)*(10^9)) == 0
        xaccel(k, 1) = Ex(1, 1) * (-1.60217653e-19/m);
        yaccel(k, 1) = Ey(1, 1) * (-1.60217653e-19/m);
    elseif round(Px(k, 1)*(10^9)) == 0
        xaccel(k, 1) = Ex(1, round(Py(k, 1)*(10^9))) *
 (-1.60217653e-19/m);
        yaccel(k, 1) = Ey(1, round(Py(k, 1)*(10^9))) *
 (-1.60217653e-19/m);
    elseif round(Py(k, 1)*(10^9)) == 0
        xaccel(k, 1) = Ex(round(Px(k, 1)*(10^9)), 1) *
 (-1.60217653e-19/m);
        yaccel(k, 1) = Ey(round(Px(k, 1)*(10^9)), 1) *
 (-1.60217653e-19/m);
    else
        xaccel(k, 1) = Ex(round(Px(k, 1)*(10^9)), round(Py(k,
 1)*(10^9))) * (-1.60217653e-19/m);
        yaccel(k, 1) = Ey(round(Px(k, 1)*(10^9)), round(Py(k,
 1)*(10^9))) * (-1.60217653e-19/m);
    end
end

avgV = sqrt(sum(Vx(:, 1).^2)/numelec + sum(Vy(:, 1).^2/numelec));

for j = 1 : numelec
    for w = 2 : numit
        Vx(j, w) = Vx(j, w-1) + xaccel(j, w-1) * dt;
        Vy(j, w) = Vy(j, w-1) + yaccel(j, w-1) * dt;
        if isnan(Px(j, w-1))
            if left == 1
                if Vx(j, w) < 0
                    Vx(j, w:end) = -Vx(j, w);
                end
                Px(j, w) = 0 + Vx(j, w)*dt;
            end
            if right == 1
                if Vx(j, w) > 0
                    Vx(j, w:end) = -Vx(j, w);
                end
                Px(j, w) = 200e-9 + Vx(j, w)*dt;
            end
        else
            Px(j, w) = Px(j, w-1) + Vx(j, w)*dt;
        end

        if Px(j, w) > 200e-9
            left = 1;
```

```
            right = 0;
            Px(j, w) = NaN;
        end
        if Px(j, w) < 0
            left = 0;
            right = 1;
            Px(j, w) = NaN;
        end

        Py(j, w) = Py(j, w-1) + Vy(j, w)*dt;
        if Py(j, w) > 100e-9
            Py(j, w) = 100e-9;
            Vy(j, w:end) = -Vy(j, w);
        end
        if Py(j, w) < 0
            Py(j, w) = 0;
            Vy(j, w:end) = -Vy(j, w);
        end

        if (Px(j, w) >= 80e-9 && Px(j, w) <= 120e-9 && Py(j, w) >=
60e-9) || (Px(j, w) >= 80e-9 && Px(j, w) <= 120e-9 && Py(j, w) <=
40e-9)
            if (Px(j, w-1) <= 80e-9 && Py(j, w-1) <= 40e-9) || (Px(j,
w-1) <= 80e-9 && Py(j, w-1) >= 60e-9) || (Px(j, w-1) >= 120e-9 &&
Py(j, w-1) <= 40e-9) || (Px(j, w-1) >= 120e-9 && Py(j, w-1) >= 40e-9)
                Vx(j, w:end) = -Vx(j, w-1);
            end
            if Px(j, w-1) >= 80e-9 && Px(j, w-1) <= 120e-9 && Py(j,
w-1) <= 60e-9 && Py(j, w-1) >= 40e-9
                Vy(j, w:end) = -Vy(j, w-1);
            end
        end

        if Pscat > rand()
            Vx(j, w:end) = random(dist);
            Vy(j, w:end) = random(dist);
        end

        if isnan(Px(j, w))
            if round(Px(j, w-1)*(10^9)) == 0 && round(Py(j,
w-1)*(10^9)) == 0
                xaccel(j, w) = Ex(1, 1) * (-1.60217653e-19/m);
                yaccel(j, w) = Ey(1, 1) * (-1.60217653e-19/m);
            elseif round(Px(j, w-1)*(10^9)) == 0
                xaccel(j, w) = Ex(1, round(Py(j, w-1)*(10^9))) *
(-1.60217653e-19/m);
                yaccel(j, w) = Ey(1, round(Py(j, w-1)*(10^9))) *
(-1.60217653e-19/m);
            elseif round(Py(j, w-1)*(10^9)) == 0
                xaccel(j, w) = Ex(round(Px(j, w-1)*(10^9)), 1) *
(-1.60217653e-19/m);
                yaccel(j, w) = Ey(round(Px(j, w-1)*(10^9)), 1) *
(-1.60217653e-19/m);
            else
```

```
                xaccel(j, w) = Ex(round(Px(j, w-1)*(10^9)),
 round(Py(j, w-1)*(10^9))) * (-1.60217653e-19/m);
                yaccel(j, w) = Ey(round(Px(j, w-1)*(10^9)),
 round(Py(j, w-1)*(10^9))) * (-1.60217653e-19/m);
            end
        else
            if round(Px(j, w)*(10^9)) == 0 && round(Py(j, w)*(10^9))
 == 0
                xaccel(j, w) = Ex(1, 1) * (-1.60217653e-19/m);
                yaccel(j, w) = Ey(1, 1) * (-1.60217653e-19/m);
            elseif round(Px(j, w)*(10^9)) == 0
                xaccel(j, w) = Ex(1, round(Py(j, w)*(10^9))) *
 (-1.60217653e-19/m);
                yaccel(j, w) = Ey(1, round(Py(j, w)*(10^9))) *
 (-1.60217653e-19/m);
            elseif round(Py(j, w)*(10^9)) == 0
                xaccel(j, w) = Ex(round(Px(j, w)*(10^9)), 1) *
 (-1.60217653e-19/m);
                yaccel(j, w) = Ey(round(Px(j, w)*(10^9)), 1) *
 (-1.60217653e-19/m);
            else
                xaccel(j, w) = Ex(round(Px(j, w)*(10^9)), round(Py(j,
 w)*(10^9))) * (-1.60217653e-19/m);
                yaccel(j, w) = Ey(round(Px(j, w)*(10^9)), round(Py(j,
 w)*(10^9))) * (-1.60217653e-19/m);
            end
        end
    end
end


for g = 1:10
    figure(7)
    xLim = [80e-9 80e-9 120e-9 120e-9];
    yLim1 = [0 40e-9 40e-9 0];
    yLim2 = [100e-9 60e-9 60e-9 100e-9];
    line(xLim, yLim1)
    hold on
    line(xLim, yLim2)
    plot(Px(g, :), Py(g, :))
    xlabel('x (nm)')
    ylabel('y (nm)')
    title('Particle trajectory with bottle neck')
    hold off
end
```




```
denM = [Px(:, 1000), Py(:, 1000)];

figure(8)
pcolor(hist3(denM, [200 100]))
camroll(-90)
title('Electron density map with bottle neck')
ylabel('x(nm)')
xlabel('y(nm)')
```




```
%The next steps can be increasing the sample number of electrons for
 the
%monte-carlos method.
```

Published with MATLAB® R2018b