Mary Oji 101036761

Part 1: Electron Modelling(2000 electrons)

```
%Using Maxwell's principle of equipartition of energy and this is
 given by;
%
%
```

$$\overline{KE} = \frac{1}{2}kT = 2(\frac{1}{2}m\overline{v^2}) \Rightarrow \overline{v^2} = \frac{2kT}{m}$$

```
L = 200e-9;
W = 100e-9;
q = -1.60217662e-19;
Vx = 0.1;
Vy = 0;
conc = 1e15*100^2; % Concentration of electrons in 1/m^2
mo = 9.1e-31; %kg
mn = 0.26 * mo; %effective mass
T = 300; %K
k = 1.28e-23; %J/K??
vth = sqrt((2*k*T)/mn); %thermal velocity
tmn = 0.2e-12; %seconds(mean time between collisions)

%Mean of free path
meanFP = vth * tmn

%electrons
eplot = 30000;
loop = 1000;
movie = 0;

%spacial step
t_step0 = 0.01 * 2e-14; %2e-14 ia the area of the region
%ideal spacial step
t_step = t_step0 - 0.1e-16; % smaller than 1/100 of region

pos = zeros(eplot,4);
traj = zeros(loop,eplot*2);
temp = zeros(loop,1);

% The non-periodic top and bottom boundaries can be set to be either
% specular (1) or diffusive (0) with the following parameters:
top_specular = 0;
bottom_specular = 0;


meanFP =

   3.6033e-08
```

When the given voltages are applied, the electric field components in the solid are (assuming that the fields are uniform):

```
Ex = Vx/L
Ey = Vy/W
```

```
Ex =

   5.0000e+05


Ey =

     0
```

The force on each electron is

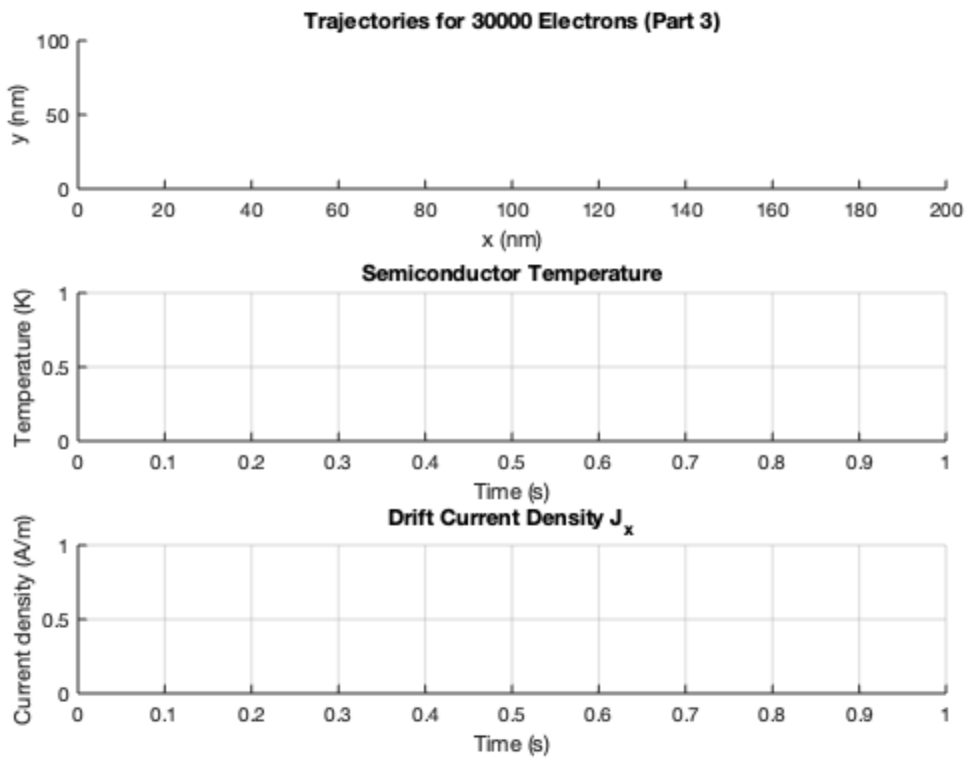
```
Fx = q*Ex
Fy = q*Ey
```

```
Fx =

  -8.0109e-14


Fy =

     0
```

For one time step, this increases the speed in each direction by

```
dvx = Fx*t_step/mn;
dvy = Fy*t_step/mn;
dvx = dvx.*ones(eplot,1);
dvy = dvy.*ones(eplot,1);


p_Scat = 1 - exp(-t_step/tmn);
v_o = makedist('Normal', 'mu', 0, 'sigma', sqrt(k*T/mn));

%initial polpulation
 for i = 1:eplot
    ang = rand*2*pi;
    pos(i,:) = [L*rand W*rand random(v_o) random(v_o)];
 end

figure(1);
subplot(3,1,1);
plot([],[]);
axis([0 L/1e-9 0 W/1e-9]);
```

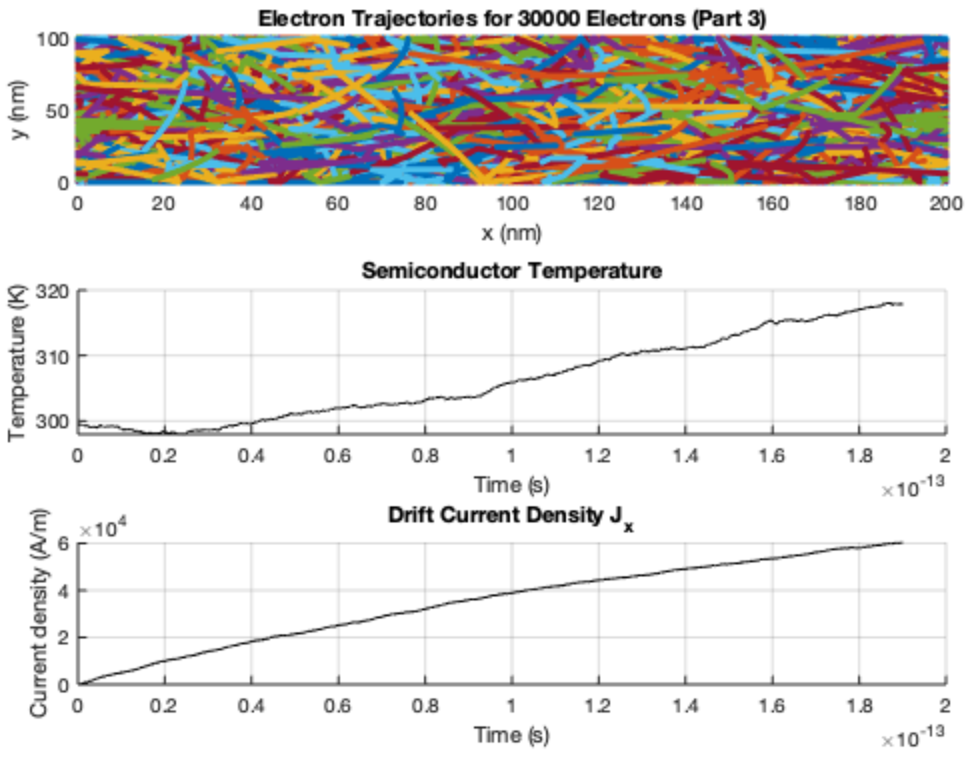
```
title(sprintf('Trajectories for %d Electrons (Part 3)',...
    eplot));
xlabel('x (nm)');
ylabel('y (nm)');

figure(1);
subplot(3,1,2);
temperature_plot = animatedline;
title('Semiconductor Temperature');
xlabel('Time (s)');
ylabel('Temperature (K)');
grid on;

figure(1);
subplot(3,1,3);
current_plot = animatedline;
title('Drift Current Density J_x');
xlabel('Time (s)');
ylabel('Current density (A/m)');
grid on;
```

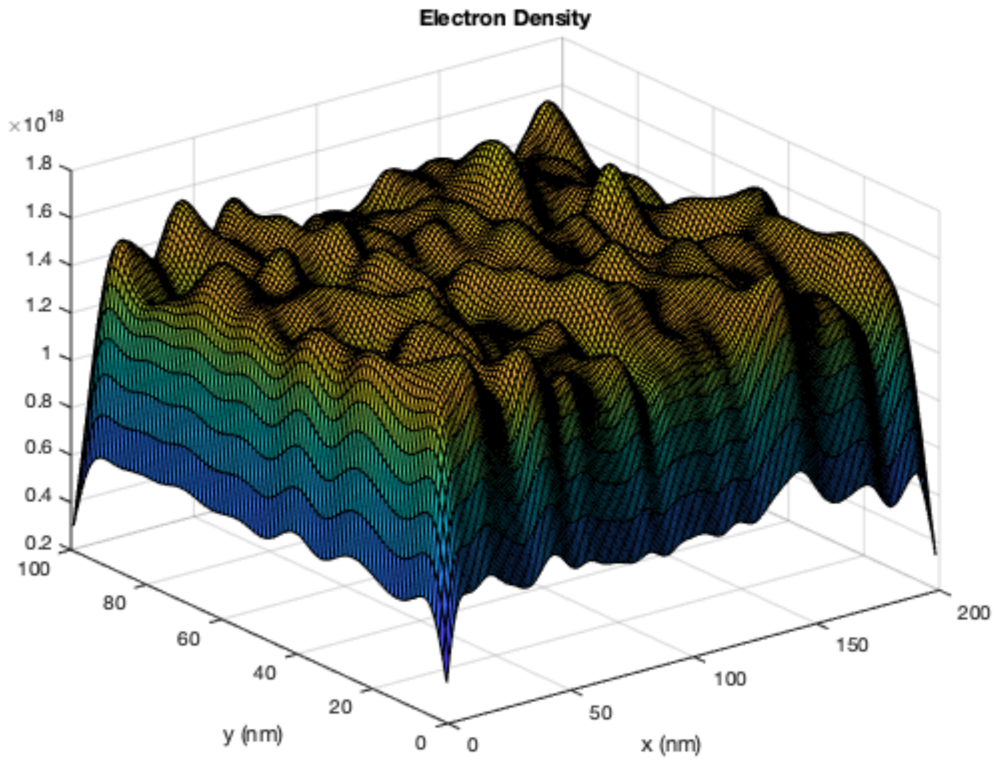


Run through the simulation:

```
for i = 1:loop
    % Update the velocities

    pos(:,3) = pos(:,3) + dvx;
    pos(:,4) = pos(:,4) + dvy;

    %Update the positions
    pos(:,1:2) = pos(:,1:2) + t_step.*pos(:,3:4);

    j = pos(:,1) > L;
    pos(j,1) = pos(j,1) - L;

    j = pos(:,1) < 0;
    pos(j,1) = pos(j,1) + L;

    j = pos(:,2) > W;

    if(top_specular)
        pos(j,2) = 2*W - pos(j,2);
        pos(j,4) = -pos(j,4);
    else % Diffusive
        % The electron bounces off at a random angle
        pos(j,2) = W;
        v = sqrt(pos(j,3).^2 + pos(j,4).^2);
        ang = rand([sum(j),1])*2*pi;
        pos(j,3) = v.*cos(ang);
        pos(j,4) = -abs(v.*sin(ang));
    end
    j = pos(:,2) < 0;

    if(bottom_specular)
        pos(j,2) = -pos(j,2);
        pos(j,4) = -pos(j,4);
    else % Diffusive
        % The electron bounces off at a random angle
        pos(j,2) = 0;
        v = sqrt(pos(j,3).^2 + pos(j,4).^2);
        ang = rand([sum(j),1])*2*pi;
        pos(j,3) = v.*cos(ang);
        pos(j,4) = abs(v.*sin(ang));
    end

    %scatter
    j = rand(eplot, 1) < p_Scat;
    pos(j,3:4) = random(v_o, [sum(j),2]);

    temp(i) = (sum(pos(:,3).^2) + sum(pos(:,4).^2))*mn/k/2/eplot;

    %Trajectory
    for j=1:eplot
        traj(i, (2*j):(2*j+1)) = pos(j, 1:2);
    end

    %temperature
    temperature(i) = (sum(pos(:,3).^2) + sum(pos(:,4).^2))*mn/k/2/
eplot;

    % Calculate and record the current density
    J(i, 1) = q.*conc.*mean(pos(:,3));
    J(i, 2) = q.*conc.*mean(pos(:,4));

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

     if(movie && mod(i,10) == 0)
        figure(1);
        subplot(3,1,1);
        hold off;
        plot(pos(1:eplot,1)./1e-9, pos(1:eplot,2)./1e-9, 'o');
        axis([0 L/1e-9 0 W/1e-9]);
        hold on;
        title(sprintf('Trajectories for %d Electrons (Part 3)',...
        eplot));
        xlabel('x (nm)');
        ylabel('y (nm)');
        %pause(0.05);
    end
end

% Show trajectories after the movie
figure(1);
subplot(3,1,1);
title(sprintf('Electron Trajectories for %d Electrons (Part 3)',...
    eplot));
xlabel('x (nm)');
ylabel('y (nm)');
axis([0 L/1e-9 0 W/1e-9]);
grid on;
hold on;
for i=1:eplot
    plot(traj(:,i*2)./1e-9, traj(:,i*2+1)./1e-9, '.');
end

%Electron density map using a histogram
density = hist3(pos(:,1:2),[200 100])';

% Smooth out the electron density map
N = 20;
sigma = 3;
[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(2);
density = conv2(density,f,'same');
density = density/(W./size(density,1)*L./size(density,2));
surf(conv2(density,f,'same'));
title('Electron Density');
xlabel('x (nm)');
ylabel('y (nm)');

temp_sum_x = zeros(ceil(L/1e-9),ceil(W/1e-9));
```

```
temp_sum_y = zeros(ceil(L/1e-9),ceil(W/1e-9));
temp_num = zeros(ceil(L/1e-9),ceil(W/1e-9));

% Look at velocities of all the particles
for i=1:eplot
    % Find which "bin" it belongs in:
    x = floor(pos(i,1)/1e-9);
    y = floor(pos(i,2)/1e-9);
    if(x==0)
        x = 1;
    end
    if(y==0)
        y= 1;
    end

    % Add its velocity components to the cumulative count:
    temp_sum_y(x,y) = temp_sum_y(x,y) + pos(i,3)^2;
    temp_sum_x(x,y) = temp_sum_x(x,y) + pos(i,4)^2;
    temp_num(x,y) = temp_num(x,y) + 1;
end
```



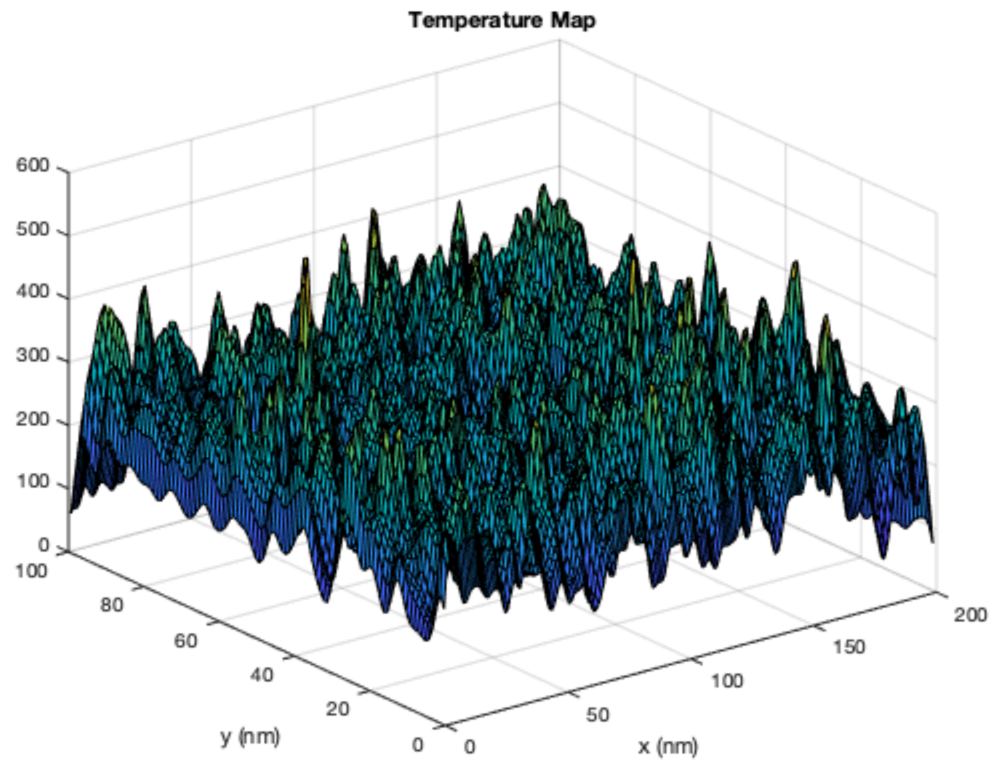


Now, with the velocities added up, calculate the temperatures:

```
temp = (temp_sum_x + temp_sum_y).*mn./k./2./temp_num;
temp(isnan(temp)) = 0;
temp = temp';
```

Like with the density map, perform some smoothing:

```
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);
surf(conv2(temp,f,'same'));
title('Temperature Map');
xlabel('x (nm)');
ylabel('y (nm)');
```



Published with MATLAB® R2019b

QUESTION 2

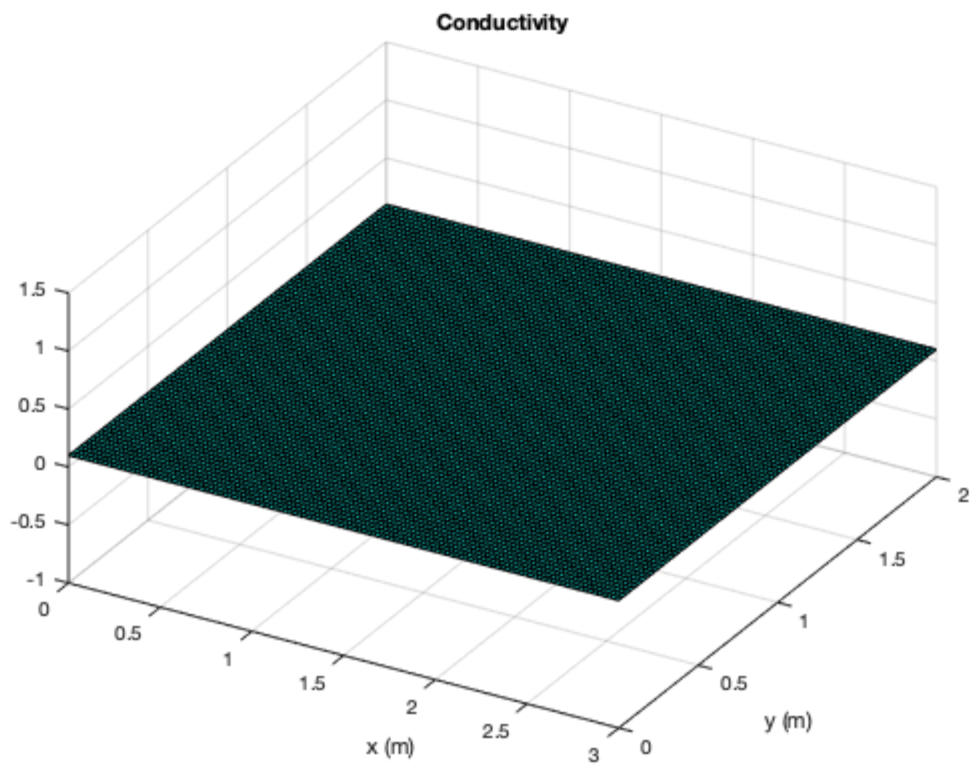
Using the Finite Difference Method in Assignment-2 to calculate the electric field and providing a field for the Monte-Carlo bottle-neck simulation.

```
W = 2;
L = 3;
V0 = 1;

dx = 0.025; % x mesh spacing
dy = 0.025; % y mesh spacing
nx = L/dx; % Number of points along x
ny = W/dy; % Number of points along y

Lb = 20;
Wb = 10;
% Generating the map of conductivity of the area
sigma_conduct = 1;
sigma_insulate = 10e-2;
% Construct the C matrix:
C = sigma_conduct.*ones(ny,nx);
Csubtract = zeros(ny,nx);

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

        % The resistivity is made high in the rectangular regions:
        if(xx <= (L+Lb)/2 && xx >= (L-Lb)/2 && (yy >= W-Wb || yy <=
 Wb))
            Csubtract(y,x) = sigma_conduct-sigma_insulate;
        end
    end
end

% Filter the condicivity to avoid numerical issues that can occur if
 the derivatives are large.
Csubtract = imgaussfilt(Csubtract, 1);
C = C - Csubtract;
```

Below, the conducitivity is plotted. I performed some filtering/smoothing so that the derivative is not very large (approaching infinity) at the junction of the two regions.

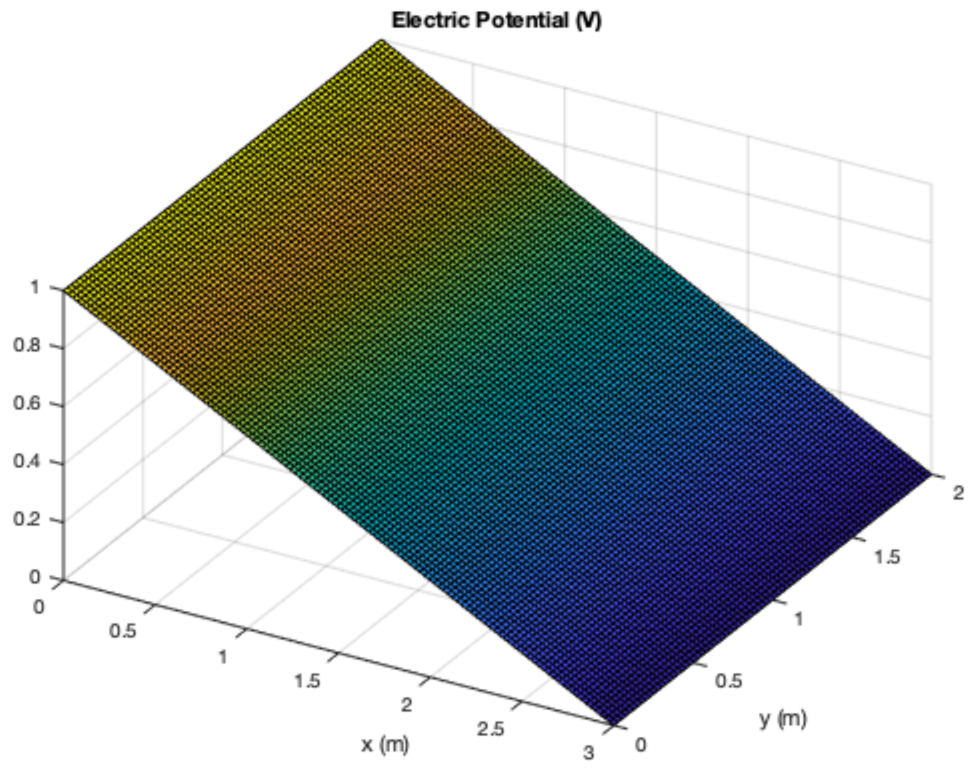
```
figure(1);
surf(linspace(0,L,nx),linspace(0,W,ny),C);
title('Conductivity');
view(30,45);
xlabel('x (m)');
ylabel('y (m)');
grid on;
```

```
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 = coordinate(x,y,nx);

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

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



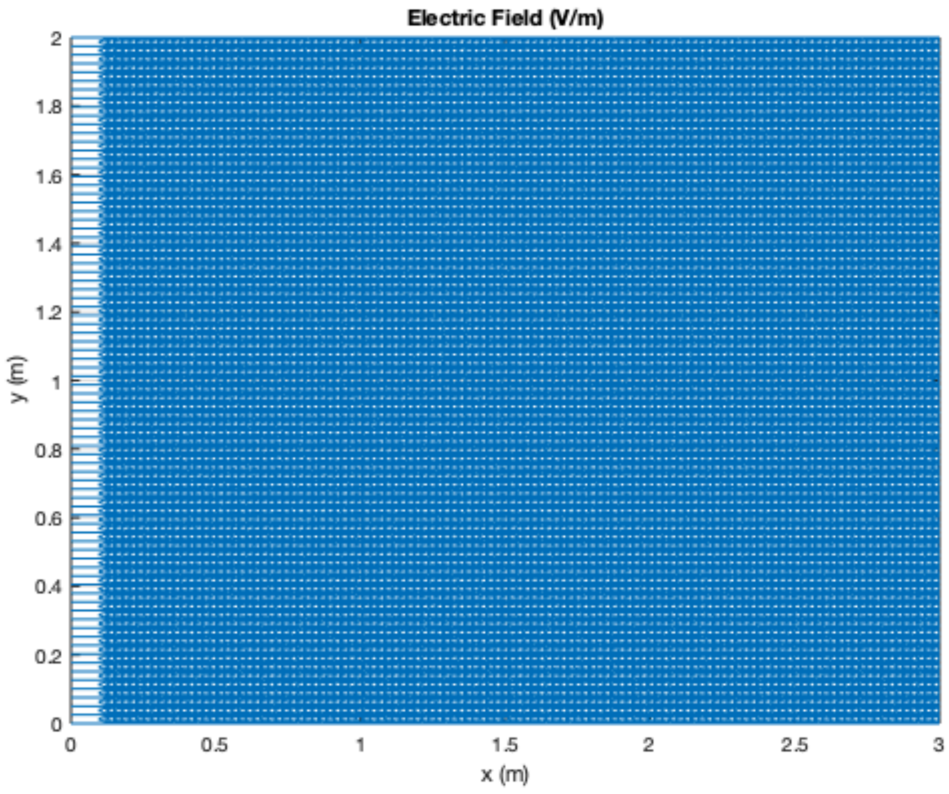
Next, the F matrix is generated.

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

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

% The vertical boundaries
for y=1:ny
    index = coordinate(1,y,nx);
    G(index,index) = 1;
    F(index) = V0;

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

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

figure(2);
surf(linspace(0,L,nx),linspace(0,W,ny),V);
view(30,45);
xlabel('x (m)');
ylabel('y (m)');
title('Electric Potential (V)');
grid on;
```



The electric field is $E = -\nabla V$. Here it is plotted along with the voltage.

```
figure(3);
[Ex,Ey] = gradient(V,dx,dy);
Ex = -1.*Ex;
Ey = -1.*Ey;
quiver(linspace(0,L,nx),linspace(0,W,ny),Ex,Ey,4);
xlabel('x (m)');
ylabel('y (m)');
title('Electric Field (V/m)');
axis([0 L 0 W]);
grid on;
```



Published with MATLAB® R2019b

FINITE DIFFERENCE METHOD

Contents

QUESTION 2

Using the Finite Difference Method in Assignment-2 to calculate the electric field and providing a field for the Monte-Carlo bottle-neck simulation.

```
W = 2;
L = 3;
V0 = 1;

dx = 0.025; % x mesh spacing
dy = 0.025; % y mesh spacing
nx = L/dx; % Number of points along x
ny = W/dy; % Number of points along y

Lb = 20;
Wb = 10;
% Generating the map of conductivity of the area
sigma_conduct = 1;
sigma_insulate = 10e-2;
% Construct the C matrix:
C = sigma_conduct.*ones(ny,nx);
Csubtract = zeros(ny,nx);

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

        % The resistivity is made high in the rectangular regions:
        if(xx <= (L+Lb)/2 && xx >= (L-Lb)/2 && (yy >= W-Wb || yy <= Wb))
            Csubtract(y,x) = sigma_conduct-sigma_insulate;
        end
    end
end

% Filter the condicivity to avoid numerical issues that can occur if the derivatives are large.
Csubtract = imgaussfilt(Csubtract, 1);
C = C - Csubtract;
```

Below, the conducitivity is plotted. I performed some filtering/smoothing so that the derivative is not very large (approaching infinity) at the junction of the two regions.

```
% figure(1);
% surf(linspace(0,L,nx),linspace(0,W,ny),C);
% title('Conductivity');
% view(30,45);
% xlabel('x (m)');
% ylabel('y (m)');
% grid on;

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 = coordinate(x,y,nx);

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

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

Next, the F matrix is generated.

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

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

% The vertical boundaries
for y=1:ny
    index = coordinate(1,y,nx);
    G(index,index) = 1;
    F(index) = V0;

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

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

% figure(2);
% surf(linspace(0,L,nx),linspace(0,W,ny),V);
% view(30,45);
% xlabel('x (m)');
% ylabel('y (m)');
% title('Electric Potential (V)');
% grid on;
```

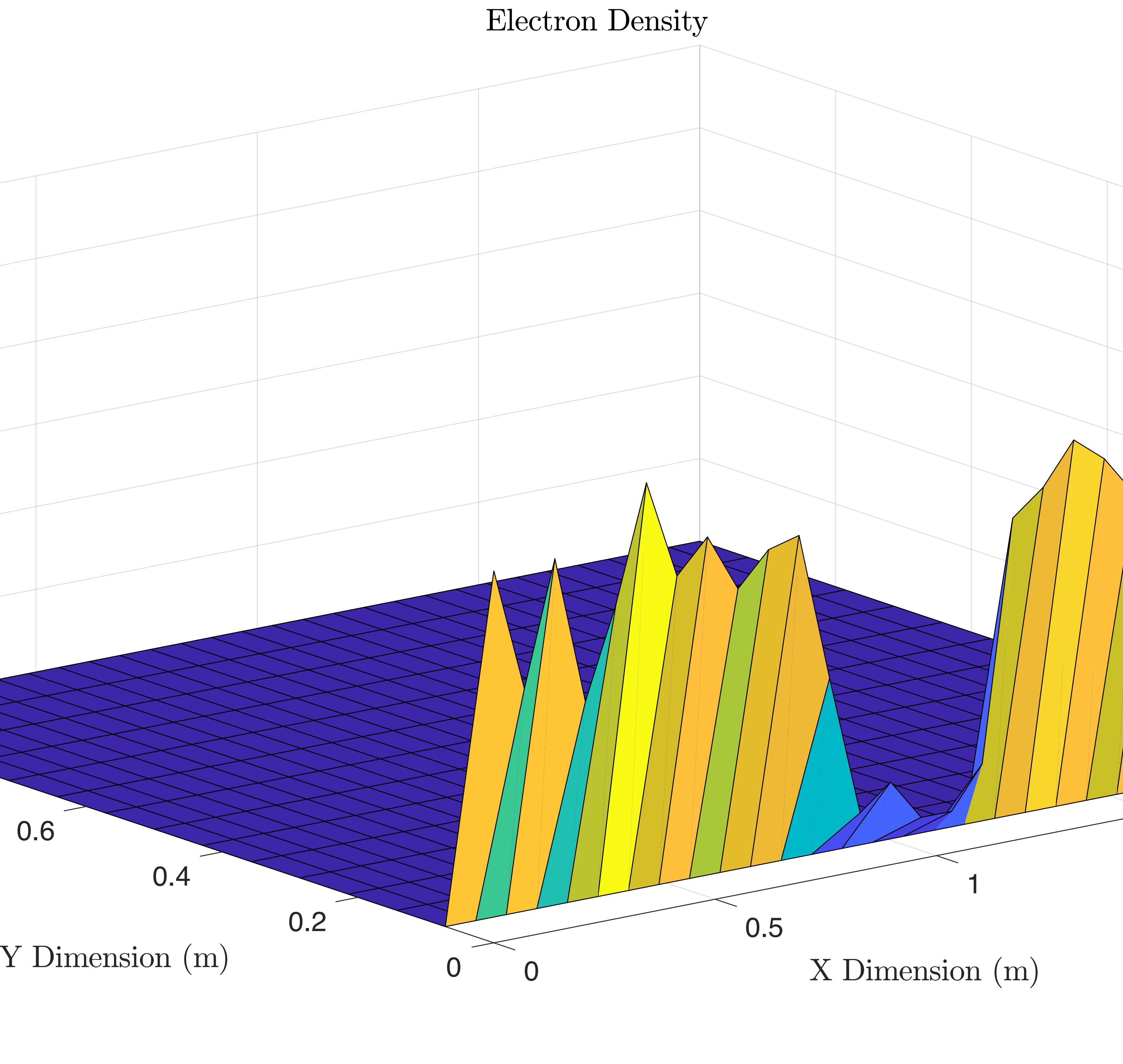
The electric field is $E = -\nabla V$. Here it is plotted along with the voltage.

```
%figure(3);
[Ex,Ey] = gradient(V,dx,dy);
Ex = -1.*Ex;
Ey = -1.*Ey;
% quiver(linspace(0,L,nx),linspace(0,W,ny),Ex,Ey,4);
% xlabel('x (m)');
% ylabel('y (m)');
% title('Electric Field (V/m)');
% axis([0 L 0 W]);
% grid on;
```

DEVICE INVESTIGATION

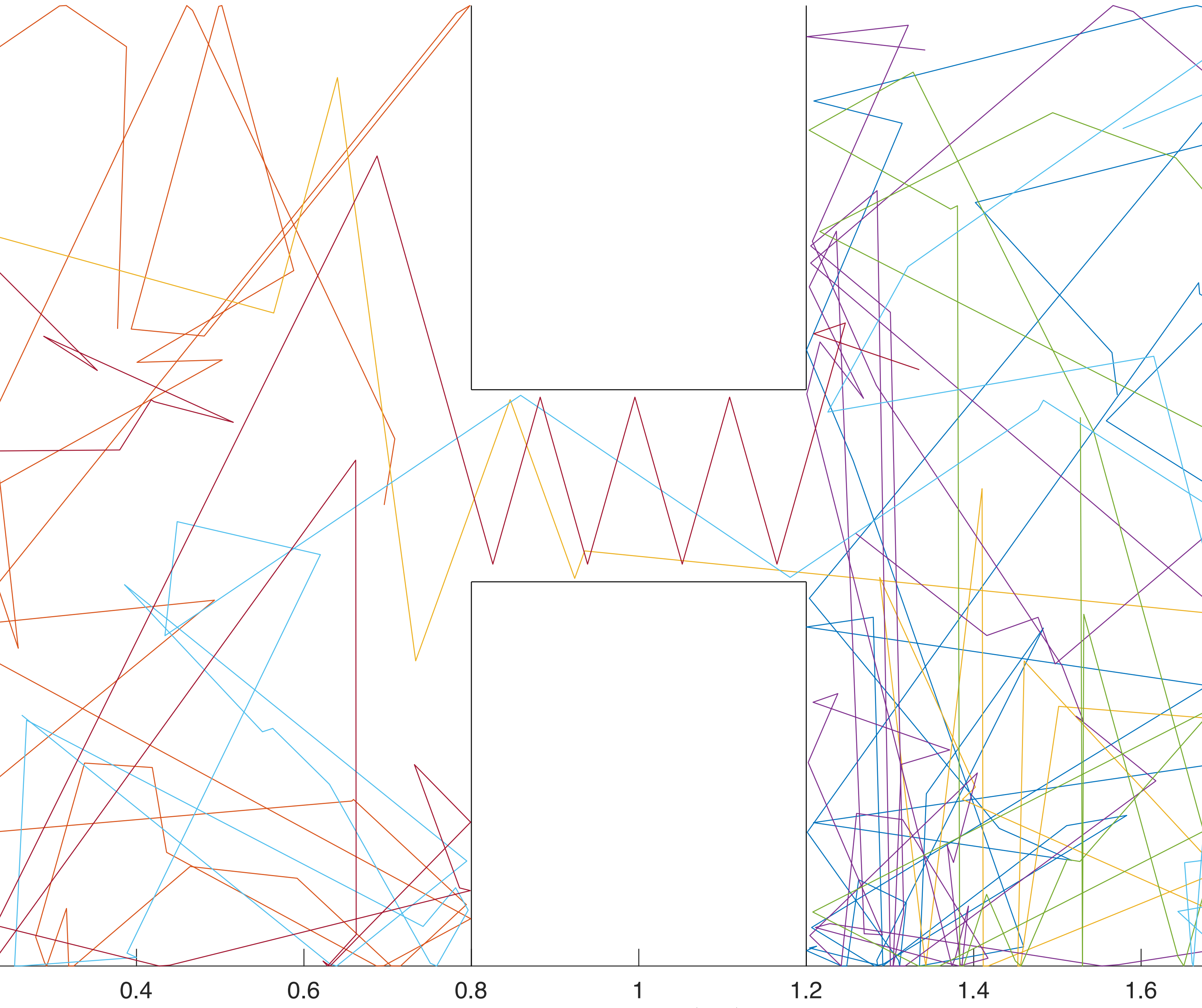
QUESTION 3

```
%Universal constants
c.eRestMass = 9.109E-31; %kg
c.boltzmann = 1.381E-23; %J/K

sys.x = 200E-9; %m
sys.y = 100E-9; %m
sys.Temp = 300; %K
sys.Tmn = 0.2E-12; %s

vX = 0.8; % V
vY = 0;

sys.EfieldX = vX./sys.x; %V/m
sys.EfieldY = vY./sys.y;


density = 10.^19; %1/m^2
sys.EDensity = density.*sys.x.*sys.y;

numOfParticles = 1000;
numLoops = 1000;

electron.effM = 0.26.*c.eRestMass;
electron.num = numOfParticles;
electron.x = zeros(1,numOfParticles);
electron.y = zeros(1,numOfParticles);
electron.vx = zeros(1, numOfParticles);
electron.vy = zeros(1, numOfParticles);

%Calculation for thermal velocity and mean free path
sys.thermalV = sqrt(2.*c.boltzmann.*sys.Temp./(electron.effM));
sys.meanFreePath = sys.thermalV.*sys.Tmn;

[electron.x, electron.y] = Position(sys.x, sys.y, electron.num);

t_Step = min([sys.x sys.y])./(100.*sys.thermalV);

part3(sys, electron, numLoops, t_Step, 'specular', Ex, Ey);
```



Electron Density
0.6
0.4
0.2
0
Y Dimension (m)
0
0.5
1
X Dimension (m)

Electron Trajectories With Calculated Electric Field



X Dimension (nm)