Assignment 3

Table of Contents

Part 3 - FDM and Electron Simulation

Using the finite difference method, an electric field is provided. Using this field, electrons are placed and each has an acceleration associated.

```
close all
clear

global C


C.q_0 = 1.60217653e-19;             % electron charge
C.hb = 1.054571596e-34;             % Dirac constant
C.h = C.hb * 2 * pi;                % Planck constant
C.m_0 = 9.10938215e-31;             % electron mass
C.kb = 1.3806504e-23;               % Boltzmann constant
C.eps_0 = 8.854187817e-12;          % vacuum permittivity
C.mu_0 = 1.2566370614e-6;           % vacuum permeability
C.c = 299792458;                    % speed of light
C.g = 9.80665; %metres (32.1740 ft) per sÂ²


nx = 200;                %
ny = 100;            % rectangular region so 3/2
G = sparse(nx*ny);     %
D = zeros(1, nx*ny);
S = zeros(ny, nx);
sigma1 = .01;
sigma2 = 1;
box = [nx*2/5 nx*3/5 ny*2/5 ny*3/5]; % Setting up the bottle neck
 conditions
```

```
sigma = zeros(nx, ny);
for i = 1:nx
    for j = 1:ny
```

```
        if i > box(1) && i < box(2) && (j < box(3)||j > box(4))
            sigma(i, j) = sigma1;
        else
            sigma(i, j) = sigma2;
        end
    end
end


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



        n = j + (i-1)*ny;
        nip = j + (i+1-1)*ny;
        nim = j + (i-1-1)*ny;
        njp = j + 1 + (i-1)*ny;
        njm = j - 1 + (i-1)*ny;

        if i == 1
            G(n, :) = 0;
            G(n, n) = 1;
            D(n) = 1;
        elseif i == nx
            G(n, :) = 0;
            G(n, n) = 1;
            D(n) = 0;
        elseif j == 1
            G(n, nip) = (sigma(i+1, j) + sigma(i,j))/2;
            G(n, nim) = (sigma(i-1, j) + sigma(i,j))/2;
            G(n, njp) = (sigma(i, j+1) + sigma(i,j))/2;
            G(n, n) = -(G(n,nip)+G(n,nim)+G(n,njp));
        elseif j == ny
            G(n, nip) = (sigma(i+1, j) + sigma(i,j))/2;
            G(n, nim) = (sigma(i-1, j) + sigma(i,j))/2;
            G(n, njm) = (sigma(i, j-1) + sigma(i,j))/2;
            G(n, n) = -(G(n,nip)+G(n,nim)+G(n,njm));
        else
            G(n, nip) = (sigma(i+1, j) + sigma(i,j))/2;
            G(n, njp) = (sigma(i, j+1) + sigma(i,j))/2;
            G(n, njm) = (sigma(i, j-1) + sigma(i,j))/2;
            G(n, nim) = (sigma(i-1, j) + sigma(i,j))/2;

            G(n, n) = -(G(n,nip)+G(n,nim)+G(n,njp)+G(n,njm));
        end
    end
end

% Voltage
V = G\D';


X = zeros(ny, nx, 1);
```

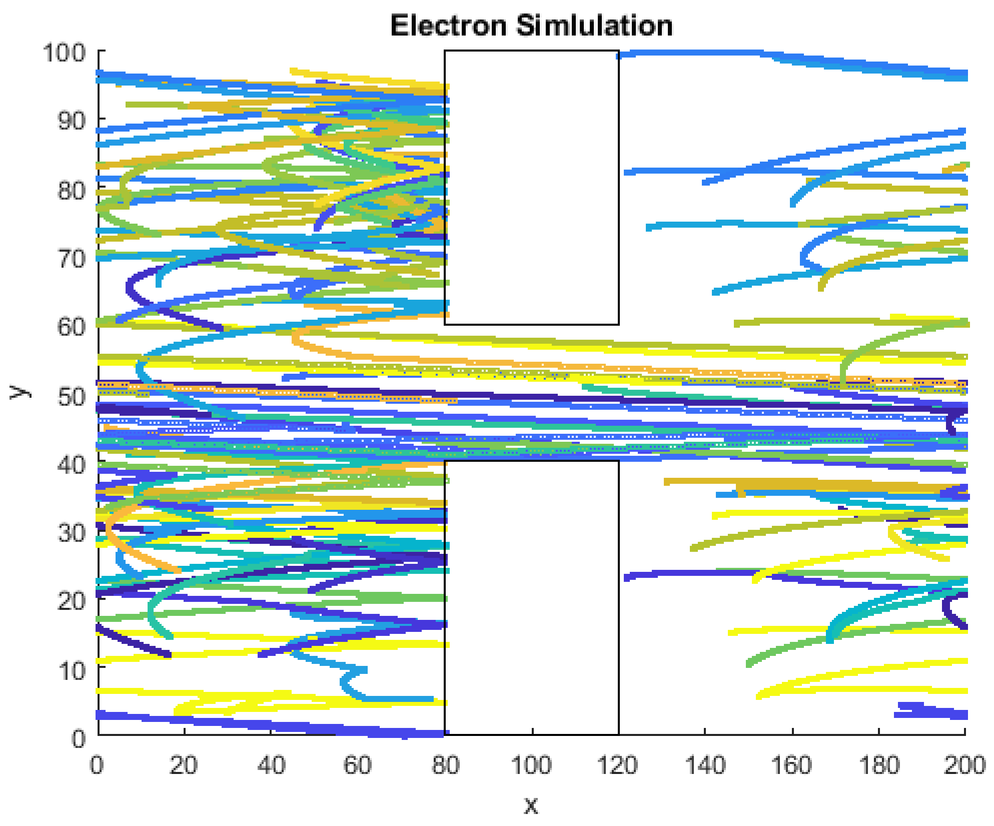
```
for i = 1:nx
    for j = 1:ny
        n = j + (i-1)*ny;
        X(j,i) = V(n);
    end
end


[Ex, Ey] = gradient(X);



Fx = -Ex*C.q_0;
Fy = -Ey*C.q_0;
ax = Fx/(0.26 * C.m_0);
ay = Fy/(0.26 * C.m_0);


nSim = 1000;
noe = 1000;
r2 = randi(360,noe,1);


colourArray = rand(50,1);


xbound = 200;
ybound = 100;
x =  randi(199,noe,1);
y =  randi(99,noe,1);
vth = sqrt((C.kb * 300)/(C.m_0 * 0.26));
vx = vth * cos(r2) ;
vy = vth * sin(r2);


for pos = 1: noe
    xpos = x(pos);
    if (xpos < 120 && xpos > 80)
        if (y(pos) < 40)
            xpos = xpos + 50;
            x(pos) = xpos;


        elseif(y(pos) > 60)
            xpos = xpos - 50;
            x(pos) = xpos;

        else

        end


    end
end
```

```
MFP = vth * 0.2 * 10^-12;


pScat = 1 - exp((-3 * 10^-16)/(0.2 * 10^-12));


tMatrix = zeros(noe);


for t = 1:nSim

    for i = 1:1:noe
        % Rounding position of electrons
        if(x(i) > 199)
            Dimx(i) = 199;
        else

            Dimx(i)= ceil(x(i));
            if Dimx(i) < 1.0
                Dimx(i) = 1;

            end

        end



        if(y(i) > 99)
            Dimy(i) = 99;
        else


            Dimy(i) = ceil(y(i));
            if Dimy(i) < 1.0
                Dimy(i) = 1;

            end


        end
        % Update velocity with acceleration
        vx = vx + ax(Dimy(i), Dimx(i))*(1e-8);
        vy = vy + ay(Dimy(i), Dimx(i))*(1e-8);

    end




    vxc = vx; % create copy of vx
    vyc = vy; % create copy of vy
```

```
[n,m] = size(vx);
[n1,m1] = size(vy);

%%randomly permutation of positions in vx and vy%%%%%
idx = randperm(n);
randomvx = vx;
randomvx(idx,1)= vx (:,1) ;

idy = randperm(n1);
randomvy = vy;
randomvy(idy,1) = vy(:,1);


%Modelling scattering%%%%%%
rScatter= rand(noe,1);

%this gives 1s and 0s. 1 means it scatters
tempScatter = rScatter < pScat;
randomvx = tempScatter .* randomvx; % not scattered are 0s
randomvy = tempScatter .* randomvy;  % not scattered are 0s

%not scattered
notScatter = rScatter >= pScat;
%%%%%%%%%%%%%%%%%%%%%%%%%

vx = vx .* notScatter; % the scattered vx are now 0
vy = vy .* notScatter; % scattered vy = 0

vx = vx + randomvx;
vy = vy + randomvy;



%%%%%%%%%%%%%%
xc = x; % x copy
yc = y ;% y copy


%Reflecting for y bounds%
temp = y >= ybound;
temp1 = y < ybound;


temp = temp * -1;

tempHigher = temp + temp1;


temp2 = y <= 0;
temp3 = y > 0;

temp2 = temp2 * -1;
tempLower = temp2 + temp3;
```

```
vy = vy .* tempHigher;
vy = vy .* tempLower;

%%%%%%%%%%%%%%%%%%%

% when x > 200%%%%%
tempx1 = x <= 200;

x = x .* tempx1;
%%%%%%%%%%%%%%%%%%

%%When x goes less than zero , come from 200 %%%%

tempx2 = x < -0.01;


tempx2 = tempx2 * 200;
tempxFinal = x + tempx2;

x = tempxFinal;

%%%%Dealing with the lower rectangle%%%%%
tLR1s = ( x > 80 & x < 120) & y < 40;
tLR0s = tLR1s == 0;
tLR1s = -1 * tLR1s;

f = tLR1s + tLR0s;

vx = vx .* f;

tLR1s = ( x > 80 & x < 120) & (y < 41 & y >= 40);
tLR0s = tLR1s == 0;
tLR1s = -1 * tLR1s;

f = tLR1s + tLR0s;

vy = vy .* f;

%tempFinalLower = x .* y
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%Dealing with the upper rectangle%%%%%%
tUR1s = ( x > 80 & x < 120) & y > 60;
tUR0s = tUR1s == 0;
tUR1s = -1 * tUR1s;

f = tUR1s + tUR0s;

vx = vx .* f;


tUR1s = ( x > 80 & x < 120) & (y >59 & y < 60);
tUR0s = tUR1s == 0;
tUR1s = -1 * tUR1s;
```

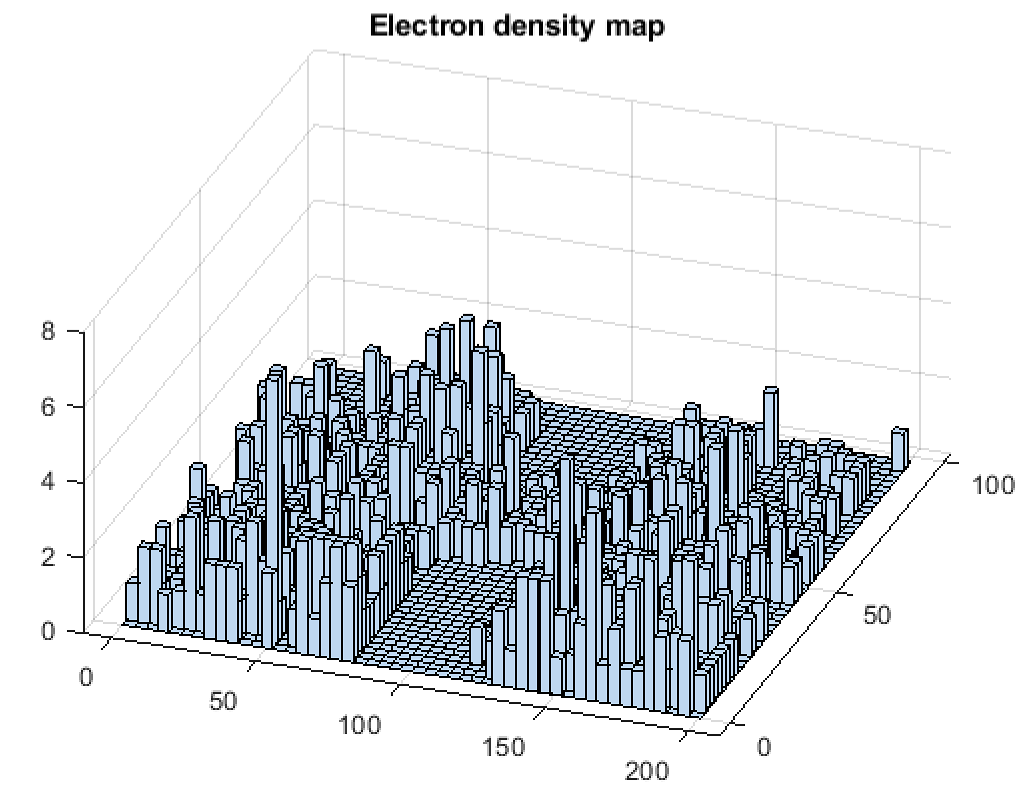
```
        f = tUR1s + tUR0s;

        vy = vy .* f;

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


        %%%%%%%%%%%%%%%%%%%
        dx = vx * (1/20000000);
        dy = vy * (1/20000000);

        x = x + dx;
        y = y + dy;
        vsq = (vy).^2 + (vx).^2 ;
        average = mean(vsq);

        tMatrix = ((vsq * 0.26 *  C.m_0)/C.kb);

        semiCTemperature = (average *(0.26)* C.m_0)/(C.kb);



        for q = 1:1:50
            plotx(q) = x(q);
            ploty(q) = y(q);
        end

        figure(4);

        scatter (plotx, ploty , 3 ,colourArray);
        axis([0 200 0 100]);
        rectangle('Position',[80 0 40 40]);
        rectangle('Position',[80 60 40 40]);
        xlabel("x");
        ylabel("y");

        hold on;

        title ("Electron Simlulation");

        hold on;




    end
```

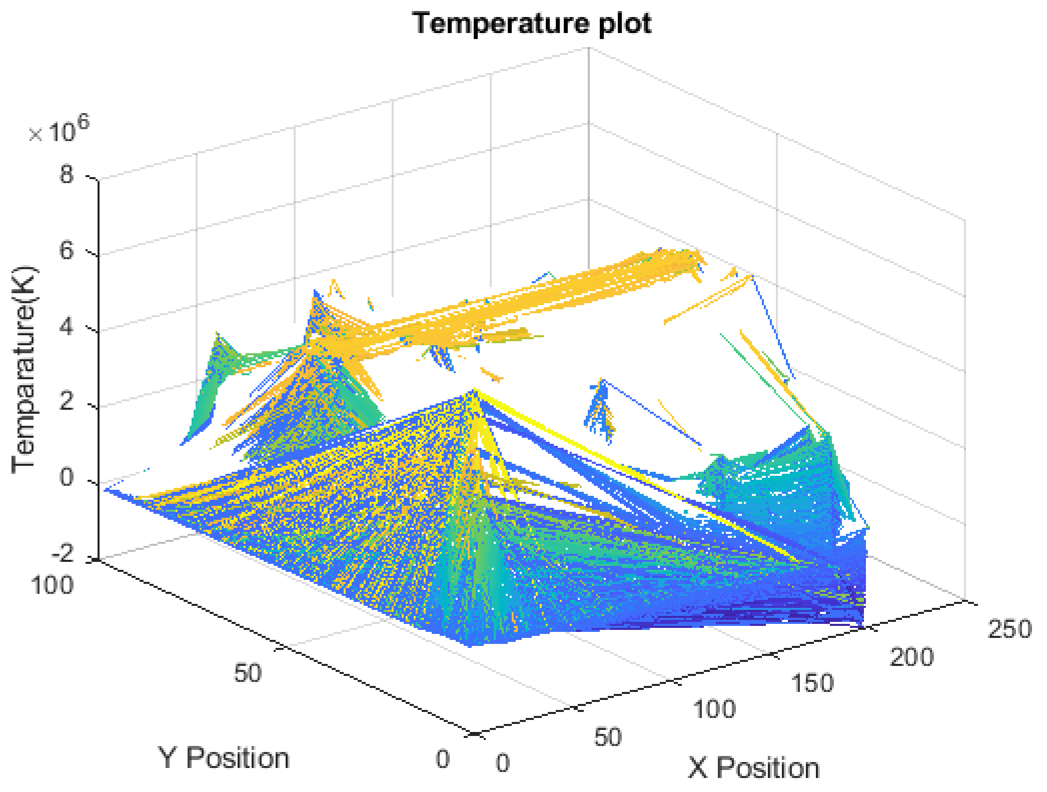


Electron Density map

Most of the electrons are on the left side of the bottleneck since the right allows you to pass but coming from the left side , the chance of passing through is not 100%

```
set(0, 'CurrentFigure', figure)
hist3([x y], [50 50]);
view([20 45]);
title("Electron density map")

[X,Y] = meshgrid (x , y);
f1 = scatteredInterpolant(x,y,tMatrix);
Z = f1(X,Y);
figure (10);
mesh(X,Y,Z);
title('Temperature plot');
xlabel('X Position');
ylabel('Y Position');
zlabel('Temparature(K)');
%axis tight;hold on
```


Electron density map
8
6
4
2
0
0
50
100
150
200
0
50
100


Temperature plot
×10^6
8
6
4
2
0
-2
Temparature(K)
100
50
0
Y Position
0
50
100
150
200
250
X Position

Next Steps

Having a smaller step size would make it more accurate so that the velocity is calculted continously without delay.

Published with MATLAB® R2018a