Part 3

In the last part, we are now doing the full implementation of applying voltage across the region with bottle-neck inserted into the electron simulation with proper scaling.

```
% Reset Everything
close all
clear

% Constant
q_0 = 1.60217653e-19;                    % electron charge
m_0 = 9.10938215e-31;                    % electron mass
meff = 0.26*m_0;                         % electron effective mass
kb = 1.3806504e-23;                      % Boltzmann constant
tmn = 0.2e-12;                           % mean time between collisions

% Region Defining
L = 200e-9;
W = 100e-9;

% Setting variables for G matrix
nx = 200;               % Length of the region
ny = 100;               % Width of the region
G = sparse(nx*ny);      % Initialize a G matrix
D = zeros(1, nx*ny);    % Initialize a matrix for G matrix operation
S = zeros(ny, nx);      % Initialize a matrix for sigma
sigma1 = 0.01;          % Setting up parameter of sigma in different
 region
sigma2 = 1;
box = [nx*2/5 nx*3/5 ny*2/5 ny*3/5]; % Setting up the bottle neck

%Sigma matrix setup
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

% Implement the G matrix with the bottle neck condition in the region
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;
```

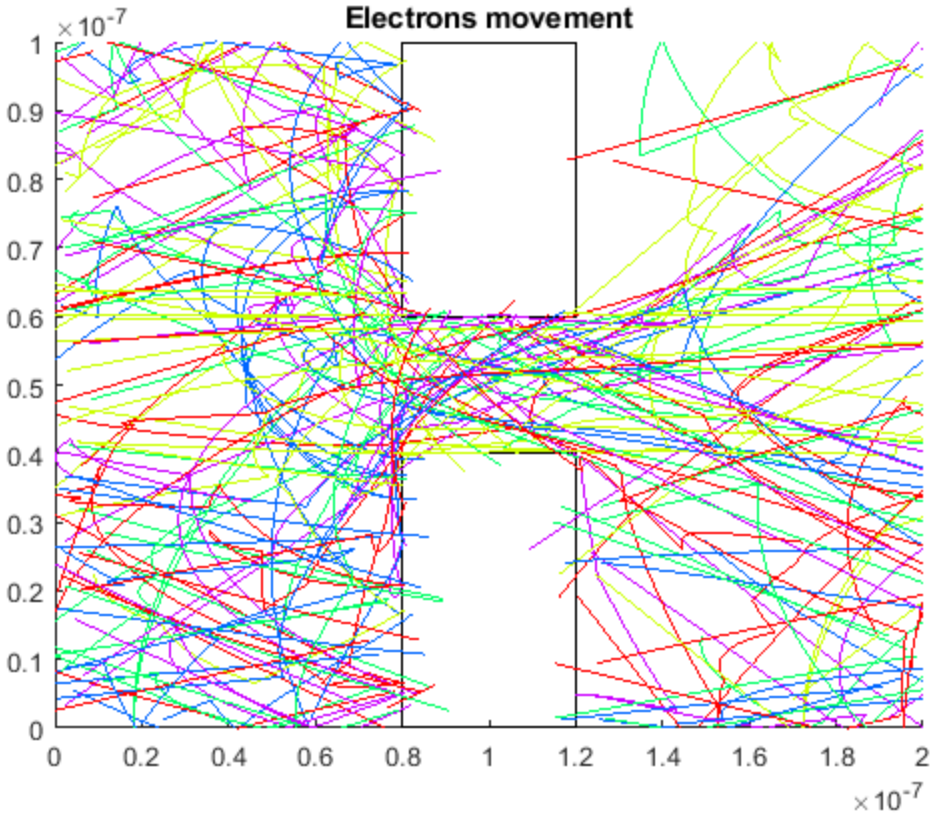
```
        if i == 1
            G(n, :) = 0;
            G(n, n) = 1;
            D(n) = 1;
        elseif i == nx
            G(n, :) = 0;
            G(n, n) = 1;
            D(n) = 0;
        elseif j == 1
            G(n, nip) = (sigma(i+1, j) + sigma(i,j))/2;
            G(n, nim) = (sigma(i-1,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

% Calculating the voltage
V = G\D';

% Inverting the G matrix
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

% Calculating the electric field from voltage with proper scaling
[Ex, Ey] = gradient(X*10^9);

% Current Condition and variables
num = 1e4;                                  % Number of electrons
T = 300;                                    % Temperature (Kelvin)
vth_e = sqrt((kb*T)/(meff));                % Thermal velocity of an
 electron
vth_ex = (vth_e)*randn(num, 1);             % X-component of thermal
 velocity
vth_ey = (vth_e)*randn(num, 1);             % Y-component of thermal
 velocity
```

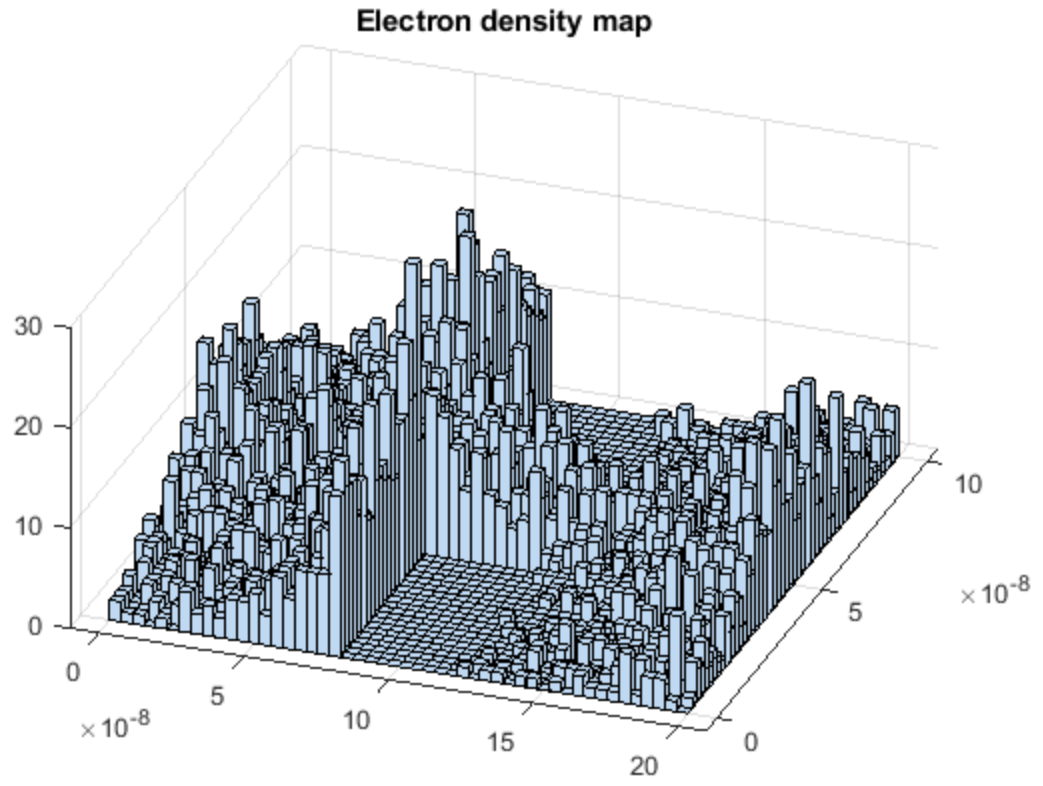
```
vthdis = sqrt(vth_ex.^2+vth_ey.^2);     % Distribution of electrons
 thermal velocity
vthav = mean(sqrt(vth_ex.^2+vth_ey.^2));% Average of thermal velocity
MFP = vthav*tmn;                        % Mean free path of electrons
Fx = -Ex*q_0;                           % X-component of force applied
 to electrons
Fy = -Ey*q_0;                           % Y-component of force applied
 to electrons
accelx = Fx/meff;                       % X-component of acceleration
 of electrons
accely = Fy/meff;                       % Y-component of acceleration
 of electrons

% Electrons Defining
Elec = zeros(num, 4);
Elec(:, 1) = L*rand(num, 1);
Elec(:, 2) = W*rand(num, 1);
Elec(:, 3) = vth_ex;
Elec(:, 4) = vth_ey;
previous = Elec;

% Setting up electrons and spawning limitation
for m = 1:1:num
    nrep = 1;
    while nrep
        xr = L*rand();
        yr = W*rand();
        if xr > 80e-9 && xr < 120e-9 && (yr > 60e-9 ||yr < 40e-9)
            nrep = 1;
        else
            Elec(m, 1) = xr;
            Elec(m, 2) = yr;
            nrep = 0;
        end
    end
end

% Electron simulation variables
t = 1e-11;                          % Total Time
dt = 1e-14;                         % Time Step
Psat = 1 - exp(-dt/tmn);            % Exponential Scattering
 Probability
numplot = 5;                        % Number of electron plotted
color = hsv(numplot);               % Colour Setup

% Setting figure for later assigning
f1 = figure;
f2 = figure;
f3 = figure;

% Setting up rectangles boundaries
set(0, 'CurrentFigure', f1)
rectangle('Position', [80e-9 0 40e-9 40e-9])
rectangle('Position', [80e-9 60e-9 40e-9 40e-9])
```

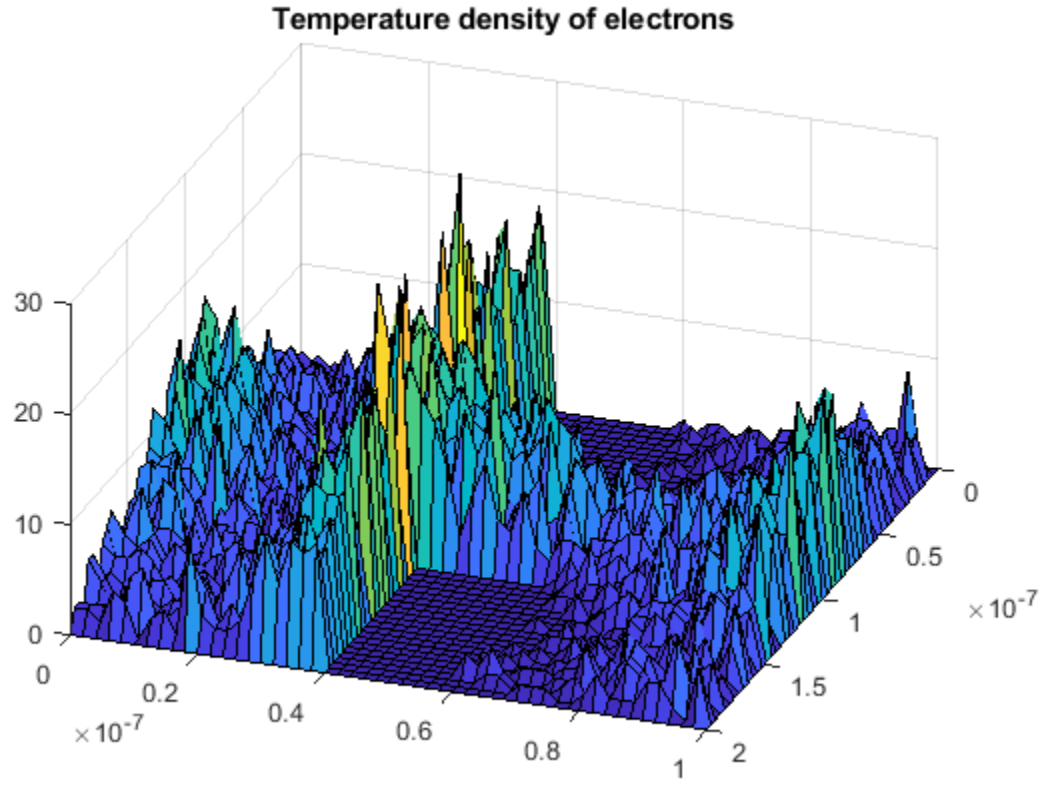
```
hold on

% Electrons simulation
for n = 0:dt:t

    % Adding appropriate acceleration into the electron velocity
    for i = 1:1:num
        % Rounding position of electrons
        Dimx(i) = ceil(Elec(i, 1)*10^9);
        Dimy(i) = ceil(Elec(i, 2)*10^9);

        % Adding acceleration of the electric field into the electrons
        Elec(i, 3) = Elec(i, 3) + accelx(Dimy(i), Dimx(i))*dt;
        Elec(i, 4) = Elec(i, 4) + accely(Dimy(i), Dimx(i))*dt;
    end

    % Electrons scattering
    if Psat > rand()
        vth_ex = (vth_e/sqrt(2))*randn(num, 1);
        vth_ey = (vth_e/sqrt(2))*randn(num, 1);
        Elec(:, 3) = vth_ex;
        Elec(:, 4) = vth_ey;
    end

    % Moving the electrons
    for p = 1:1:num
        previous(p, 1) = Elec(p, 1);
        previous(p, 2) = Elec(p, 2);
        Elec(p, 1) = Elec(p, 1)+ Elec(p, 3)*dt;
        Elec(p, 2) = Elec(p, 2)+ Elec(p, 4)*dt;
    end

    % Plotting limited amount of electrons
    set(0, 'CurrentFigure', f1)
    for q = 1:1:numplot
        title('Electrons movement');
        plot([previous(q, 1), Elec(q, 1)], [previous(q, 2),
 Elec(q,2)],...
            'color', color(q, :))
        xlim([0 L])
        ylim([0 W])
        hold on
    end

    % Setting up the boundaries
    for o = 1:1:num
        % Looping on x-axis
        if Elec(o, 1) > L
            Elec(o, 1) = Elec(o, 1) - L;
        end
        if Elec(o, 1) < 0
            Elec(o, 1) = Elec(o, 1) + L;
        end
        % Reflecting on y-axis
```

```
            if Elec(o, 2) > W || Elec(o, 2) < 0
                Elec(o, 4) = -1*Elec(o, 4);
                Elec(o,2) = previous(o,2);
            end
            % Part 3 Simulation, boundaries for electrons movements
            if Elec(o, 1) > 80e-9 && Elec(o, 1) < 120e-9 &&...
                    (Elec(o, 2) < 40e-9 || Elec(o, 2) > 60e-9)
                % in the box
                if Elec(o, 2) < 40e-9
                    % lower box
                    if previous(o, 2) > 40e-9
                        % in hole
                        Elec(o, 4) = -1*Elec(o, 4);
                        Elec(o,2) = previous(o,2);
                    else
                        Elec(o, 3) = -1*Elec(o, 3);
                        Elec(o,1) = previous(o,1);
                    end
                end
                if Elec(o, 2) > 60e-9
                    % upper box
                    if previous(o, 2) < 60e-9
                        % in hole
                        Elec(o, 4) = -1*Elec(o, 4);
                        Elec(o,2) = previous(o,2);
                    else
                        Elec(o, 3) = -1*Elec(o, 3);
                        Elec(o,1) = previous(o,1);
                    end
                end
            end
        end

    %     pause(0.01)

    end

    % Electron Density map
    set(0, 'CurrentFigure', f2)
    hist3(Elec(:, 1:2), [50 50]);
    view([20 45]);
    title("Electron density map")

    % Temperature map
    set(0, 'CurrentFigure', f3)
    [binx, biny] = meshgrid(0:L/50:L, 0:W/50:W);
    zcheck = zeros(51, 51);
    tempcheck = zeros(51, 51);
    counter = 0;
    vtotal = 0;
    for i = 1:50
        txmn = binx(1,i);
        txmx = binx(1, i+1);
        for r = 1:50
```

```
        tymn = biny(r, 1);
        tymx = biny(r+1, 1);
        for mm = 1:num
            if(Elec(mm,1)>txmn & Elec(mm,1)<txmx & Elec(mm,2)<tymx
 & ...
                    Elec(mm,2)>tymn)
                counter = counter + 1;
                zcheck(i, r) = zcheck(i, r)+1;
                vtotal = vtotal + sqrt(Elec(mm, 3)^2+Elec(mm, 4)^2);
                if(counter ~= 0)
                    tempcheck(i,r) = meff*(vtotal^2)/(counter*kb);
                end
            end
        end
        vtotal = 0;
        counter = 0;
    end
end
surf(binx, biny,zcheck)
view([110 40]);
title("Temperature density of electrons")
```




Electron density map
30
20
10
0
0
5
10
15
20
×10^-8
0
5
10
×10^-8


Temperature density of electrons
30
20
10
0
0
0.2
0.4
0.6
0.8
1
×10^-7
0
0.5
1
1.5
2
×10^-7

Both of the density plot shows that most electrons populate on the left edge of the boxes of the bottle-neck because the voltage applied across the field forces the electrons to constantly try to move toward the right side of the region but they cannot pass through the bottle-neck most of time hence creating a higher temperature on the left side.

Discussion

The next step to improve the simulation is to increase the mesh size of the G-matrix to get more accurate electric field strength and acceleration on each electrons on its respective location without rounding its location within the region.

Published with MATLAB® R2017b