ELEC 4700 Assignment 3

Table of Contents

1 - Monte-Carlo Simulation

1.a - Electric Field

The equation for electric field is given by...

$$E = V/d$$

...where E is the electric field, V is the voltage differance, and d is the distance over which the voltage differance occurs. For a 0.1V applied voltage and a distance of x = 200nm, we obtain an electric field of magnitude 500 kN/C.

1.b - Electron Force

The force on each electron is given by...

$$F = E * q$$

...where F is the force on each electron, E is the electric field, and q is the electron charge. For an E-field of magnitude 500 kN/C and taking the electron charge of 0.1602e-18 C, we obtain a force of 8.01e-13 N.

1.c - Electron Acceleration

Acceleration is given by...

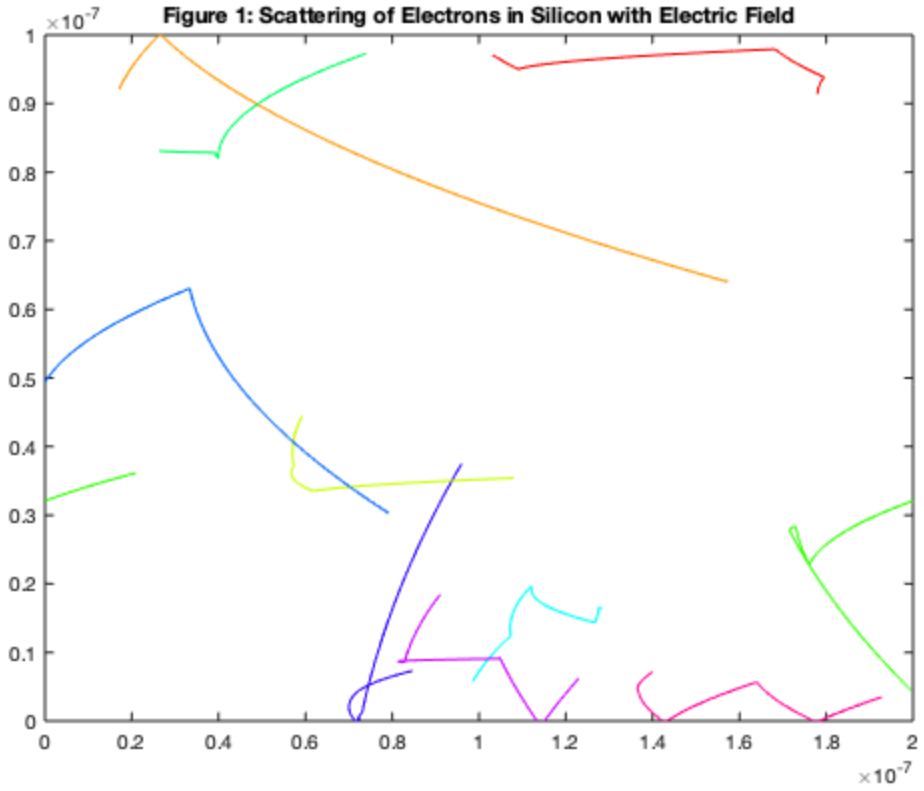
$$a = F/m$$

...where a is the acceleration, f is the force applied, and m is the mass of the object. For an applied force of 8.01e-13 N and an electron mass of 9.109e-31 kg, we obtain an acceleration of 8.7935e17 m/s^2.

This acceleration can then be added to the monte-carlo electon scattering model as show in the code below.

```
% Setup
clear
clc

% Constant values used for modeling
m0 = 9.11e-31; % Electron Rest Mass (kg)
mn = 0.26*m0;  % Effective Mass (kg)
T = 300;       % Temperature (K)
k = 1.381e-23; % Boltzmann Constant (J/K)
q = 0.1602e-17; % Electron Charge (C)

% Thermal Velocity (m/s)
thermalVelocity = sqrt(2*k*T/mn);

% Model Parameters
numElec = 10000; % Number of Electrons
xDim = 200e-9;  % Dimention of Si region in X direction (m)
yDim = 100e-9;  % Dimention of Si region in Y direction (m)
deltaT = 5e-15;  % Time step value (s)
lastPos = zeros(2,numElec); % Array to hold previous particle position
color = hsv(10); % Set particle colours for plot
avgVelocityX = zeros(1,100);
Tmn = 0.2e-12; % Mean time between collisions (s)
acceleration = 8.7935e17; % Acceleration due to E-field (m/s^2)

% Scattering probability
Pscat = 1 - exp(-(deltaT/Tmn));


% Initiate particle positions as array of random numbers within
 boundaries.
position = [xDim;yDim].*rand(2,numElec);

% Create array of X and Y velocity components (m/s)
maxBoltz = sqrt(k*T/mn).*randn(2,numElec);

% Initial variables
temp = 300; % Si starting temperature (K)
time = zeros(1,100);   % Initial time (s)
numCollisions = 0; % Initial number of collisions occured


for T = 1:100
    lastPos = position;
    for c = 1:numElec

        % Scattering condition
         if Pscat > rand
             maxBoltz(1,c) = sqrt(k*T/mn)*randn;
             maxBoltz(2,c) = sqrt(k*T/mn)*randn;
             numCollisions = numCollisions + 1;
        end

        % Particle behaviour if boundary is reached
```

```
        if position(2,c) <= 0 || position(2,c) >= yDim
            maxBoltz(2,c) = (-1)*maxBoltz(2,c);
        end
        if position(1,c) <= 0
            position(1,c) = xDim;
            lastPos(1,c) = xDim;
        elseif position(1,c) >= xDim
            position(1,c) = 0;
            lastPos(1,c) = 0;
        end
    end

    sumVelocityX = sum(maxBoltz(1,:));

    % Update velocities due to electric field acceleration.
        maxBoltz(1,:) = maxBoltz(1,:) + (deltaT.*acceleration);

    % Advance all particles in direction of their velocity
        position(1,:) = position(1,:) + (deltaT.*maxBoltz(1,:));
        position(2,:) = position(2,:) + (deltaT.*maxBoltz(2,:));

    % Update time
    time(T) = T*deltaT;

   % Plot new particle positions.
    figure(1)
    for c = 1:10
        plot([lastPos(1,c) position(1,c)],[lastPos(2,c)
 position(2,c)], 'color', color(c,:))
        xlim([0 2e-7]);
        ylim([0 1e-7]);
        title('Figure 1: Scattering of Electrons in Silicon with
 Electric Field')
        hold on
    end

    % Calculate average velocity (m/s)
    avgVelocityX(T) = sumVelocityX/numElec;

    % Update time
    time(T) = T*deltaT;

end
```



1.d - Current Behaviour

The current density as a function of drift velocity is given in the equation below.

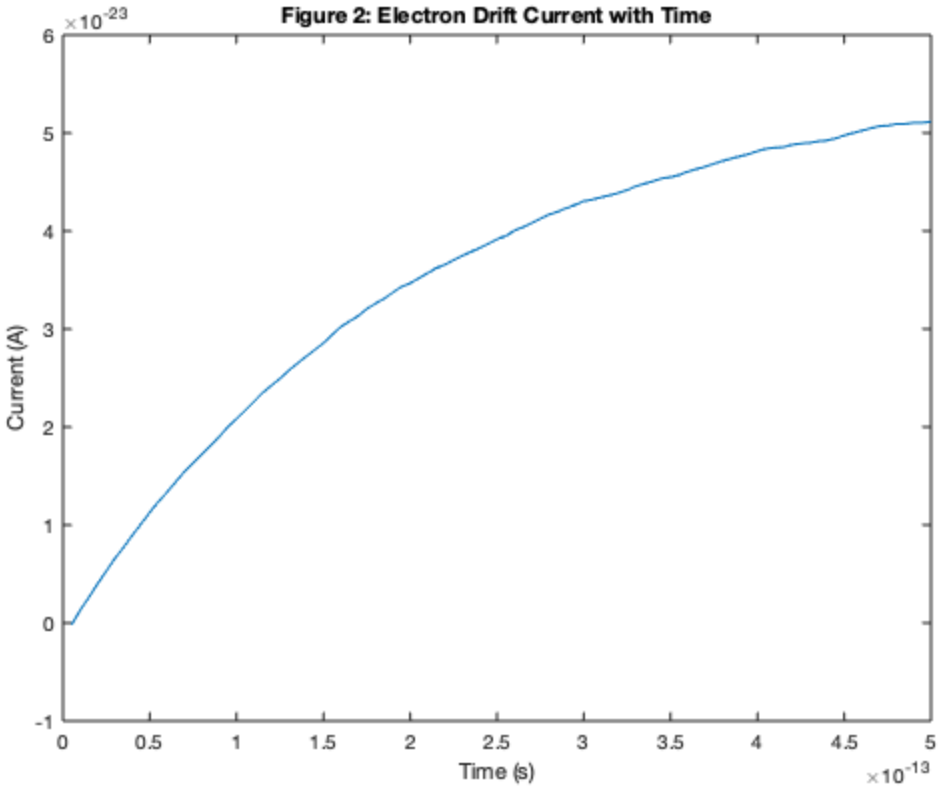
$$J = I/A = v_d * n * e$$

Where I is the drift current, A is the area, v_d is the drift velocity, n is the number of carriers, and e is the electorn charge. This equation can now be rearranged to give current as shown...

$$I = v_d * n * e * A$$

The code below implements this formula to provide a plot of average current flow in the x direction with respect to time.

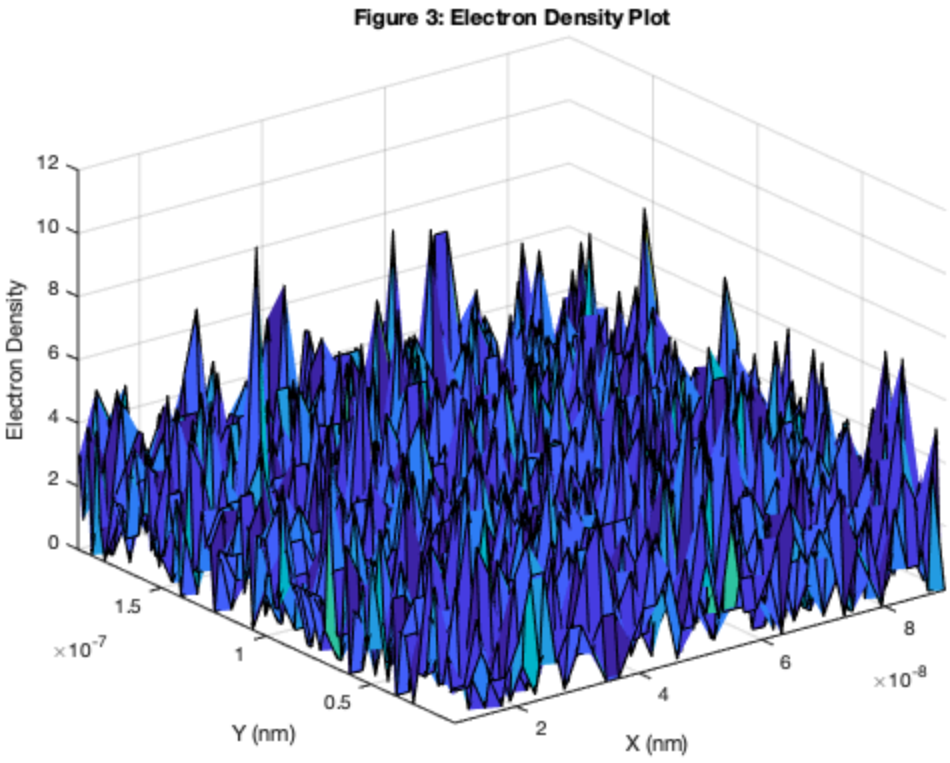
```
Id = avgVelocityX.*(numElec*q*xDim*yDim);

figure(2)
plot(time,Id)
title('Figure 2: Electron Drift Current with Time')
xlabel('Time (s)')
ylabel('Current (A)')
```

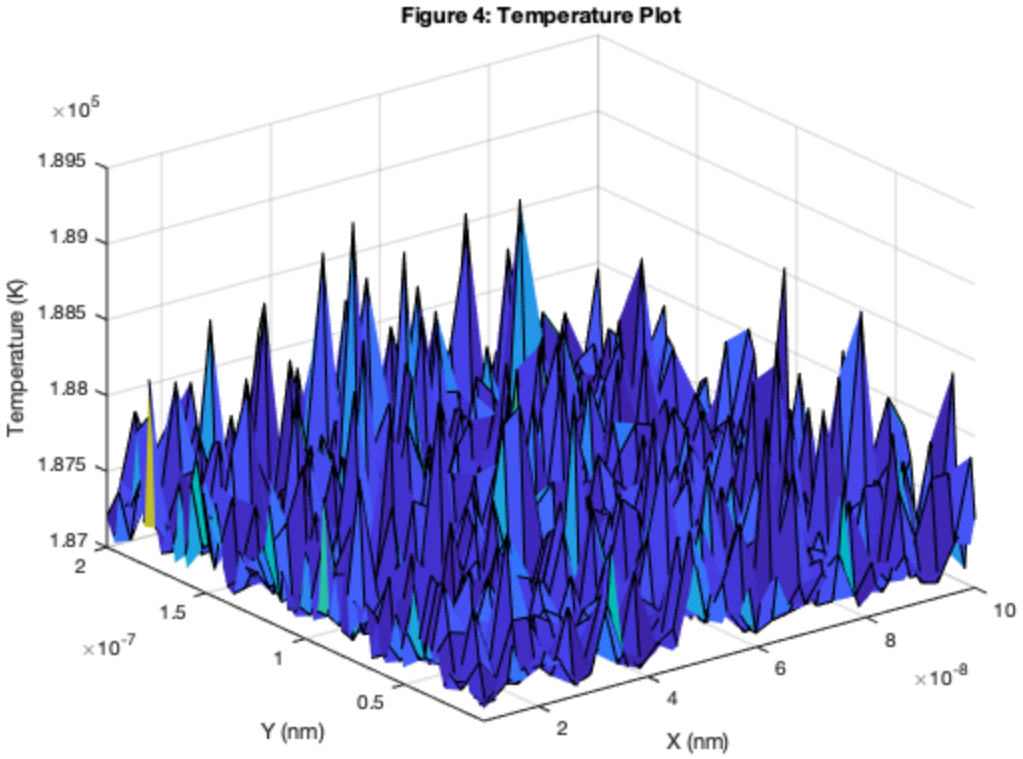


1.e - Density and Temperature Maps

```
A = hist3(position','CdataMode','auto','Nbins',[100 50]);
figure(3)
X = linspace(0,100e-9,50);
Y = linspace(0,200e-9,100);
surf(X,Y,A)
xlim([0.1e-7 0.9e-7])
ylim([0.1e-7 1.9e-7])
title('Figure 3: Electron Density Plot')
xlabel('X (nm)')
ylabel('Y (nm)')
zlabel('Electron Density')


% Temperature Plot

 temperature = zeros(1,numElec);

for n = 1:numElec
    temperature(n) = sqrt((maxBoltz(1,n)^2) + (maxBoltz(2,n)^2));
end

temperature = ((temperature.^2).*mn./(3*k))+thermalVelocity;
```

```
[xq,yq] = meshgrid(0:3e-9:200e-9, 0:3e-9:100e-9);
vq = griddata(position(1,:),position(2,:),temperature,xq,yq);
X = linspace(0,100e-9,34);
Y = linspace(0,200e-9,67);
figure(4)
surf(X,Y,vq')
title('Figure 4: Temperature Plot')
xlim([0.1e-7 1e-7])
ylim([0.1e-7 2e-7])
xlabel('X (nm)')
ylabel('Y (nm)')
zlabel('Temperature (K)')
```





2 - Electric Field with Finite Difference

2.a - Potential with Bottle Neck

The matrix form of the finite differance method can be utilized to solve for the current flow through bottle-neck. Firstly a matrix of conductivity is created which defines the location and resistance of the bottle neck. The potential can then be fond using the finite differance method, and the three equations below can then be implemented to solve for the electric field and the current density.

$$E_x = -dV/dx$$

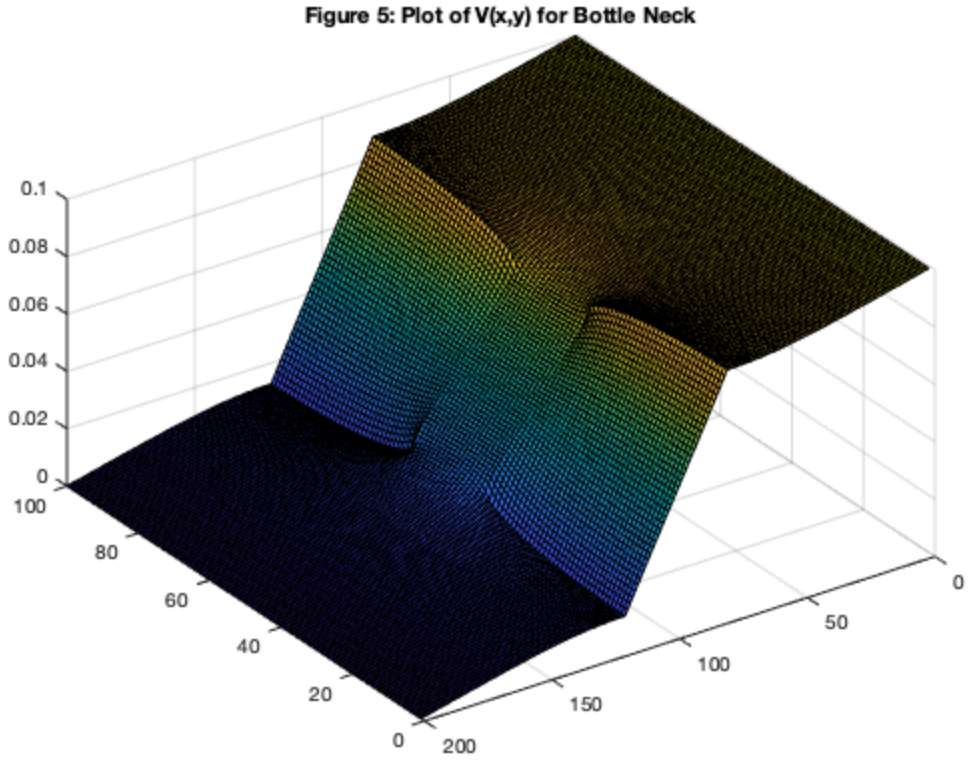
$$E_y = -dV/dy$$

$$J(x,y) = \sigma * V(x,y)$$

```
% Define Variables
V0 = 0.1;

% Define Matrix
nx = 200;
ny = 100;
G = sparse(nx*ny,nx*ny);
B = zeros(nx*ny,1);
S = ones(nx,ny).*(1/(6.2e2)); % Accounts for conductivity of Si
```

```
for i = 1:nx
    for j = 1:ny
        if i >= 80 && i <= 120
            if j <= 40 || j >= 60
                S(i,j) = 1e-10;
            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;
            B(n) = V0;
        elseif i == nx
            G(n,:) = 0;
            G(n,n) = 1;
            B(n) = 0;


        elseif j == 1
            nxm = j + (i-2)*ny;
            nxp = j + (i)*ny;
            nyp = j+1 + (i-1)*ny;

            rxm = (S(i,j) + S(i-1,j))/2;
            rxp = (S(i,j) + S(i+1,j))/2;
            ryp = (S(i,j) + S(i,j+1))/2;

            G(n,n) = -(rxm+rxp+ryp);
            G(n,nxm) = rxm;
            G(n,nxp) = rxp;
            G(n,nyp) = ryp;

        elseif j == ny
            nxm = j + (i-2)*ny;
            nxp = j + (i)*ny;
            nym = j-1 + (i-1)*ny;

            rxm = (S(i,j) + S(i-1,j))/2;
            rxp = (S(i,j) + S(i+1,j))/2;
            rym = (S(i,j) + S(i,j-1))/2;

            G(n,n) = -(rxm+rxp+rym);
            G(n,nxm) = rxm;
            G(n,nxp) = rxp;
            G(n,nym) = rym;

        else
```

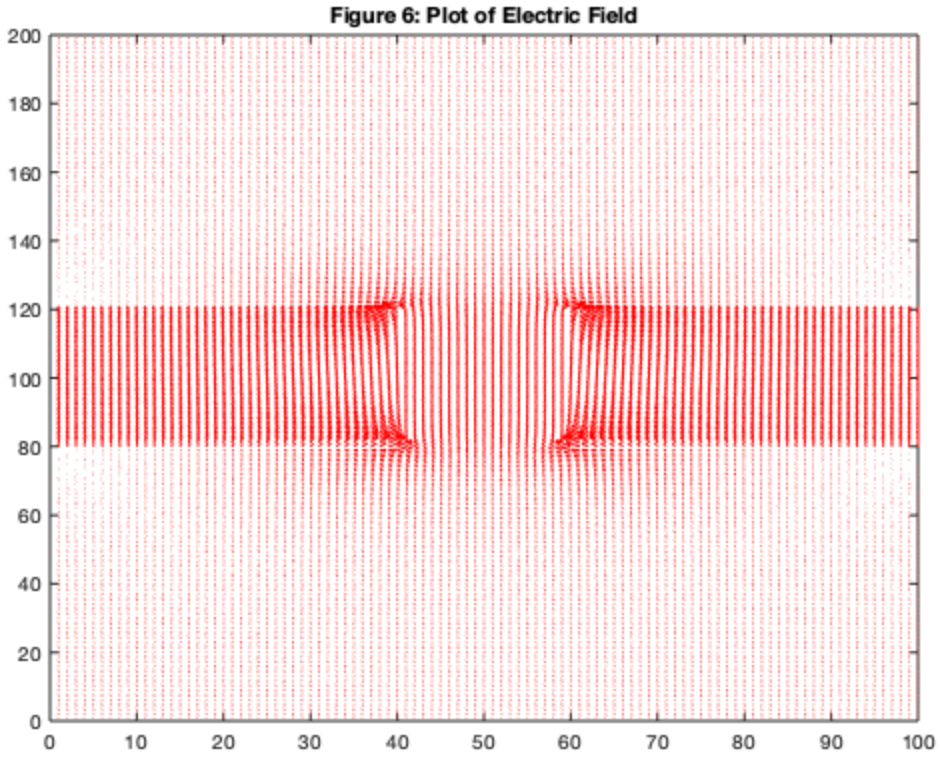
```
            nxm = j + (i-2)*ny;
            nxp = j + (i)*ny;
            nym = j-1 + (i-1)*ny;
            nyp = j+1 + (i-1)*ny;

            rxm = (S(i,j) + S(i-1,j))/2;
            rxp = (S(i,j) + S(i+1,j))/2;
            rym = (S(i,j) + S(i,j-1))/2;
            ryp = (S(i,j) + S(i,j+1))/2;

            G(n,n) = -(rxm+rxp+rym+ryp);
            G(n,nxm) = rxm;
            G(n,nxp) = rxp;
            G(n,nym) = rym;
            G(n,nyp) = ryp;
        end
    end
end

solve = G\B;
data_z = zeros(nx,ny);
data_x = linspace(1,nx,nx);
data_y = linspace(1,ny,ny);

for i = 1:nx
    for j = 1:ny
        position = j + (i-1)*ny;
        data_z(i,j) = solve(position,1);
    end
end

figure(5)
surf(data_y,data_x,data_z)
view(-125,45)
title('Figure 5: Plot of V(x,y) for Bottle Neck')
```



2.b - Electric Field

Since the electric field can is related to the change in voltage over a given distance, we can find the E field in both the x and the y directions by taking the derivative (gradient) of the obtained voltage matrix in these directions.

```
[Ex,Ey] = gradient(-1.*data_z);
Ex = Ex.*(100/200e-9);
Ey = Ey.*(100/200e-9);

figure(6)
quiver(Ex, Ey, 2, 'r')
xlim([0 100])
ylim([0 200])
title('Figure 6: Plot of Electric Field')
```



2.c - Monte carlo using Finite Difference Electric Field

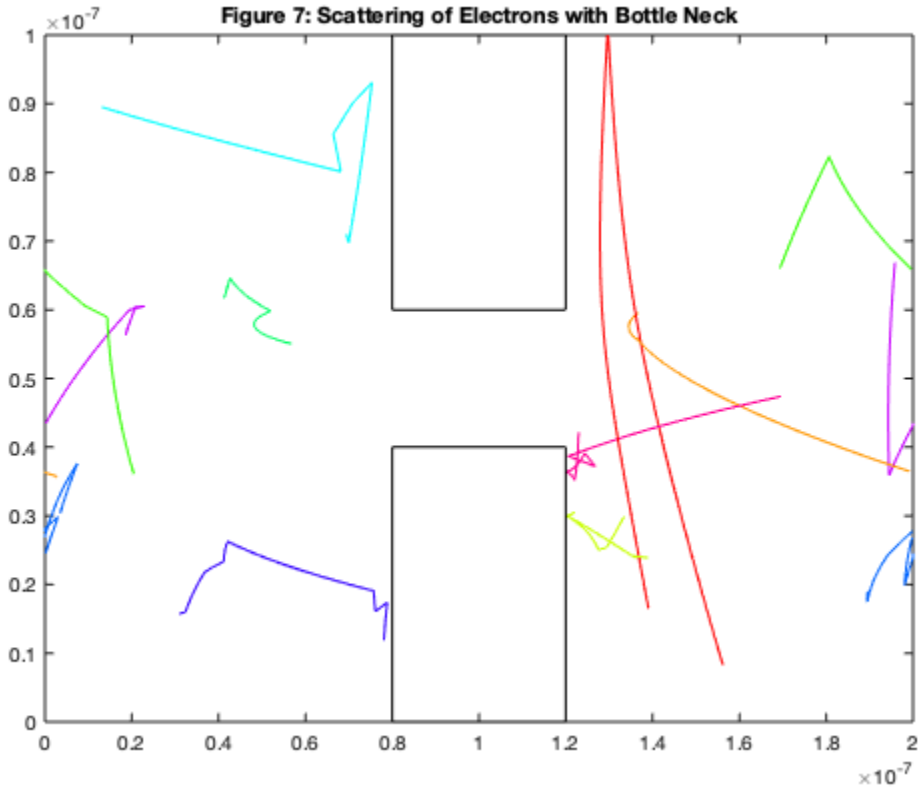
We can now apply the electric field derived using the finite differance method to the monte-carlo electron motion simulation given previously. New conditions had to be added such that a particles acceleration was dependant on its locaction, the code below shows this.

```
% Create array of X and Y velocity components (m/s)
maxBoltz = sqrt(k*T/mn).*randn(2,numElec);

% Initiate particle positions as array of random numbers within
 boundaries.
position = [xDim;yDim].*rand(2,numElec);

particleFound = true;

while particleFound

    count = 0;

    for i = 1:numElec
        if position(1,i)>=0.8e-7 && position(1,i)<=1.2e-7
            if position(2,i)<=0.4e-7 || position(2,i)>=0.6e-7
                position(1,i) = xDim*rand;
                position(2,i) = yDim*rand;
```

```
                count = count + 1;
            end
        end
    end

    if count == 0
        particleFound = false;
    end
end

for T = 0:150
    lastPos = position;

    for c = 1:numElec

        % Scattering condition
        if Pscat > rand
            maxBoltz(1,c) = sqrt(k*T/mn)*randn;
            maxBoltz(2,c) = sqrt(k*T/mn)*randn;
            numCollisions = numCollisions + 1;
        end

        % Advance all particles in direction of their velocity
        position(1,c) = position(1,c) + (deltaT*maxBoltz(1,c));
        position(2,c) = position(2,c) + (deltaT*maxBoltz(2,c));


        % Particle behaviour if boundary is reached
        if position(2,c) <= 0 || position(2,c) >= yDim
            maxBoltz(2,c) = (-1)*maxBoltz(2,c);
        end
        if position(1,c) <= 0
            position(1,c) = xDim;
            lastPos(1,c) = xDim;
        elseif position(1,c) >= xDim
            position(1,c) = 0;
            lastPos(1,c) = 0;
        end

        % Particle behaviour at bottle neck
        if position(2,c)<=0.4e-7
            if position(1,c)>=0.8e-7 && lastPos(1,c)<=0.8e-7
                maxBoltz(1,c) = (-1)*maxBoltz(1,c);
                position(1,c) = 0.79e-7;
            end
            if position(1,c)<=1.2e-7 && lastPos(1,c)>=1.2e-7
                maxBoltz(1,c) = (-1)*maxBoltz(1,c);
                position(1,c) = 1.21e-7;
            end
        end
        if position(2,c)>=0.6e-7
            if position(1,c)>=0.8e-7 && lastPos(1,c)<=0.8e-7
                maxBoltz(1,c) = (-1)*maxBoltz(1,c);
                position(1,c) = 0.79e-7;
```

```
            end
            if position(1,c)<=1.2e-7 && lastPos(1,c)>=1.2e-7
                maxBoltz(1,c) = (-1)*maxBoltz(1,c);
                position(1,c) = 1.21e-7;
            end
        end
        if position(1,c)>0.8e-7 && position(1,c)<1.2e-7
            if position(2,c)>=0.6e-7 && lastPos(2,c)<=0.6e-7
                maxBoltz(2,c) = (-1)*maxBoltz(2,c);
                position(2,c) = 0.59e-7;
            end
            if position(2,c)<=0.4e-7 && lastPos(2,c)>=0.4e-7
                maxBoltz(2,c) = (-1)*maxBoltz(2,c);
                position(2,c) = 0.41e-7;
            end
        end


    end

    sumVelocity = sum(sqrt(sum(maxBoltz.^2)));

    for n = 1:numElec

        % Get acceleration for particle position
        Pval = position.*1e9;

        Xval = ceil(Pval(1,n));
        Yval = ceil(Pval(2,n));
        if Xval < 1
            Xval = 1;
        end
        if Xval > 200
            Xval = 200;
        end
        if Yval < 1
            Yval = 1;
        end
        if Yval > 100
            Yval = 100;
        end
        accelerationX = Ex(Xval,Yval)*q/m0;
        accelerationY = Ey(Xval,Yval)*q/m0;

        % Update velocities due to electric field acceleration.
            maxBoltz(1,n) = maxBoltz(1,n) + (deltaT.*accelerationX);
            maxBoltz(1,n) = maxBoltz(1,n) + (deltaT.*accelerationY);
    end

  % Plot new particle positions.
  figure(7)
   for c = 1:10
       plot([lastPos(1,c) position(1,c)],[lastPos(2,c)
position(2,c)],'color',color(c,:))
```

```
            xlim([0 2e-7]);
            ylim([0 1e-7]);
            title('Figure 7: Scattering of Electrons with Bottle Neck')
            hold on
            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])
        end

        % Calculate average velocity (m/s)
        avgVelocity = sumVelocity/numElec;

    end
```



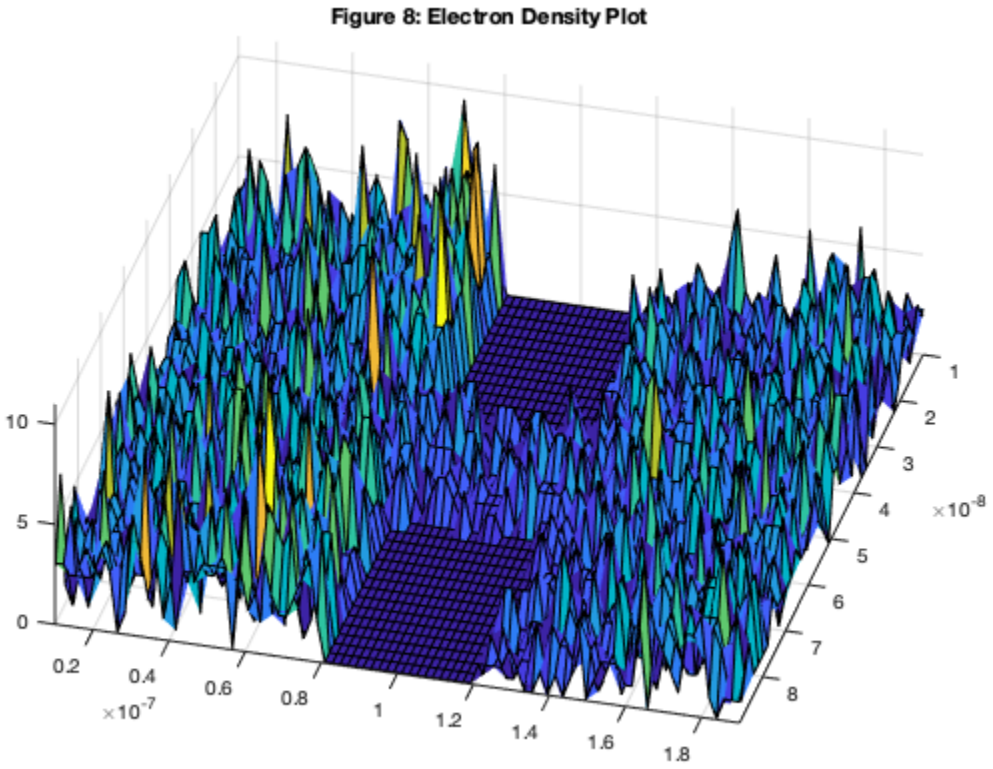
3 - Device Parameters

3.a - Density Map

When plotting a density map of the electron's positions at the end of the simulation (shown below), we see that many particles are caught along the left hand wall of the bottle neck (around x = 80 nm), particles which do make it through the bottle neck opening are most likely to be found nearest to the far right hand boundary of the box (at x = 200 nm). Particles are least likely to be found inside of the bottle neck.

```
A = hist3(position','CdataMode','auto','Nbins',[100 50]);
figure(8)
X = linspace(0,100e-9,50);
```

```
Y = linspace(0,200e-9,100);
surf(X,Y,A)
view(105,60)
xlim([0.1e-7 0.9e-7])
ylim([0.1e-7 1.9e-7])
title('Figure 8: Electron Density Plot')
```



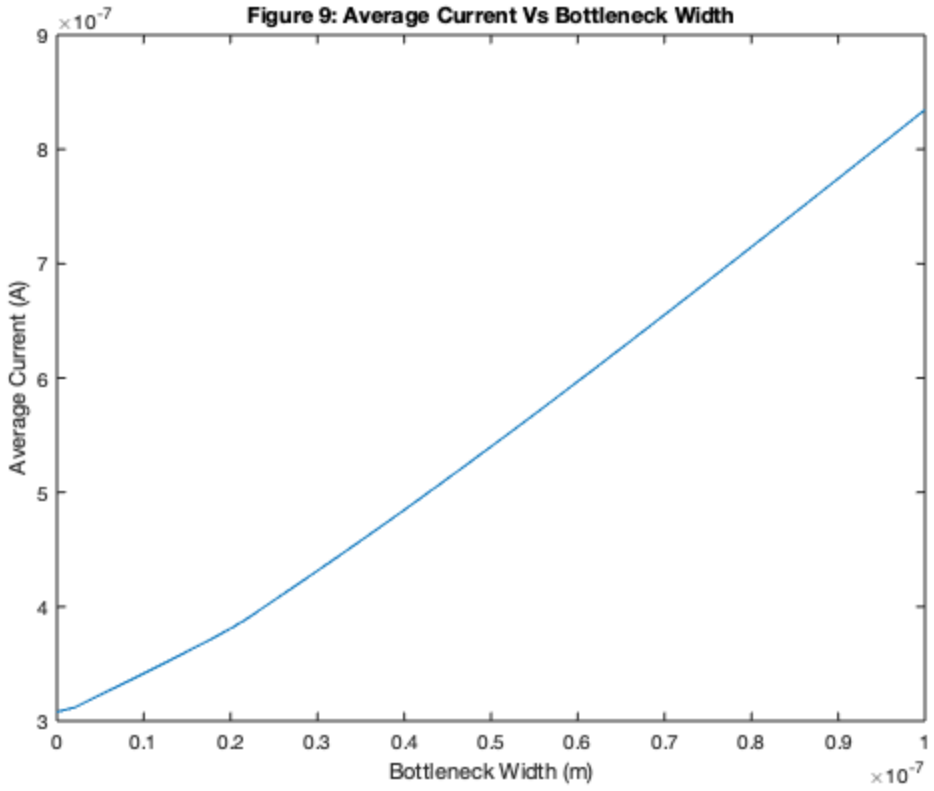
3.b - Current Density with Bottleneck Width

As we increase the width of the opening in the bottle neck, we observe that the drift current of the system increases. The bottle neck can be though of as the resistance of the system, as the bottle neck widens (and the resistance decreases) carriers are able to more easily flow from an area of higher potential to an area of lower potential, thus the current increases. The code below demonstraits this point.

```
A = zeros(1,51);

 for count = 1:51

    S = ones(nx,ny).*(1/(6.2e2)); % Accounts for conductivity of Si

    for i = 1:nx
        for j = 1:ny
            if i >= 80 && i <= 120
                if j <= (51-count) || j >= (49+count)
                    S(i,j) = 1e-10;
                end
```

```
        end
    end
end

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

if i == 1
    G(n,:) = 0;
    G(n,n) = 1;
    B(n) = V0;
elseif i == nx
    G(n,:) = 0;
    G(n,n) = 1;
    B(n) = 0;


elseif j == 1
    nxm = j + (i-2)*ny;
    nxp = j + (i)*ny;
    nyp = j+1 + (i-1)*ny;

    rxm = (S(i,j) + S(i-1,j))/2;
    rxp = (S(i,j) + S(i+1,j))/2;
    ryp = (S(i,j) + S(i,j+1))/2;

    G(n,n) = -(rxm+rxp+ryp);
    G(n,nxm) = rxm;
    G(n,nxp) = rxp;
    G(n,nyp) = ryp;

elseif j == ny
    nxm = j + (i-2)*ny;
    nxp = j + (i)*ny;
    nym = j-1 + (i-1)*ny;

    rxm = (S(i,j) + S(i-1,j))/2;
    rxp = (S(i,j) + S(i+1,j))/2;
    rym = (S(i,j) + S(i,j-1))/2;

    G(n,n) = -(rxm+rxp+rym);
    G(n,nxm) = rxm;
    G(n,nxp) = rxp;
    G(n,nym) = rym;

else
    nxm = j + (i-2)*ny;
    nxp = j + (i)*ny;
    nym = j-1 + (i-1)*ny;
    nyp = j+1 + (i-1)*ny;

    rxm = (S(i,j) + S(i-1,j))/2;
    rxp = (S(i,j) + S(i+1,j))/2;
    rym = (S(i,j) + S(i,j-1))/2;
    ryp = (S(i,j) + S(i,j+1))/2;
```

```
            G(n,n) = -(rxm+rxp+rym+ryp);
            G(n,nxm) = rxm;
            G(n,nxp) = rxp;
            G(n,nym) = rym;
            G(n,nyp) = ryp;
        end

        solve = G\B;
        data_z = zeros(nx,ny);

        for i = 1:nx
            for j = 1:ny
                position = j + (i-1)*ny;
                data_z(i,j) = solve(position,1);
            end
        end

        [Ex,Ey] = gradient(-1.*data_z);
        Jx = S.*Ex;
        Jy = S.*Ey;

        % For current we will take the average of the current density
     across the
        % area of the material.

        J = sqrt(Jx.^2 + Jy.^2);
        A(1,count) = mean(mean(J));
     end

    figure(9)
    plot(linspace(0,100e-9,51),A)
    xlabel('Bottleneck Width (m)')
    ylabel('Average Current (A)')
    title('Figure 9: Average Current Vs Bottleneck Width')
```



3.c - Next Step

In order to make this simulation a more accurate reperesentation of what would be observed on real life, one could introduce the effect of electrons interracting with one-another. When two particle of similar charge (such as two negitivly charged electrons) come close to one another, they will exert a repelling force between them. With many electrons in a relitivly small confined box, the impact of this interraction would not be insignificant, and so is important in the production of an accurate simulation.

Published with MATLAB® R2019b