Assignment 3 Part 3 - Andrew Paul

The last section of this assignment involved applying the electric field from part 2 to the electron trajectory region of part 1. Some modifications were made to the dimensions of the boxes and the region bounds as they were diffrernt for each section. The map of velocities created in part 2 using the finite diffrence method was also transposed so that it was oritented on the same place as the boxes implimented in this section. The ptoetntial across the region was increased to 0.8 V and the gradient was divided by a factor of 5 x 10^{-7} in order to achieve proper units of acceleration on the electrons with respect to the spacing units.

It should be noted that the number of electrons (num_electrons) is currently set to 1000 and if the movie is run the plotting sections must me uncommented and the number of electron must be set to 10 or less or the program will run for too long.

```
clear

nx = 50;
ny = 50;

% Create sparse G matrix
G = sparse(nx*ny,nx*ny);

% Conductivity outside box
sigma1 = 1;
% Conductivity inside box
sigma2 = 10^-2;

% Generate F matrix to set boundary conditions
F = zeros(nx*ny,1);

% Change for difference in bottle neck width
Lb = 0.4;
Wb = 0.6;

% Create matrix for mapping the conductivity and loop through to
 assign
% conductivity values for the given conditions
condMap = zeros(nx,ny);

for i = 1:nx
    for j = 1:ny
        if (i>=Lb*nx && i<=Wb*nx && j<=Lb*ny) || (i>=Lb*nx && i<=Wb*nx
 && j>=Wb*ny)
            condMap(i,j) = sigma2;
        else
            condMap(i,j) = sigma1;
        end
    end
end

% Loop through to set boundary conditions

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) = 1;

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

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

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

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

```
            G(n,nyp) = ryp;

        end

    end

end

% Find voltage values using matrix operations
V = G\F;

% Create matrix to map voltage and loop through matrix to assign
 values
% from calculated voltage matrix
VMap = zeros(nx,ny);

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

        VMap(i,j) = V(n);
    end
end

VMap = VMap';

% list of constants
m0 = 9.11e-31;
mn = 0.26*m0;
kB = 1.38e-23;
T = 300;
q = 1.602e-19;

%assignment 3 calcualtions
voltage = 0.8;
[Ex,Ey] = gradient(-VMap);
Fx = Ex*q/5e-8;
Fy = Ey*q/5e-8;
accx = Fx/mn;
accy = Fy/mn;

%region limits
xlim = 50e-9;
ylim = 50e-9;

%
vth = sqrt(2*kB*T/mn);

%initialize the number of electrons
% CHANGE TO LESS THAN 10 BEFORE RUNNING
num_electrons = 1000;

% defining array for electrons (x postion, y position, angle,
 velocity)
```

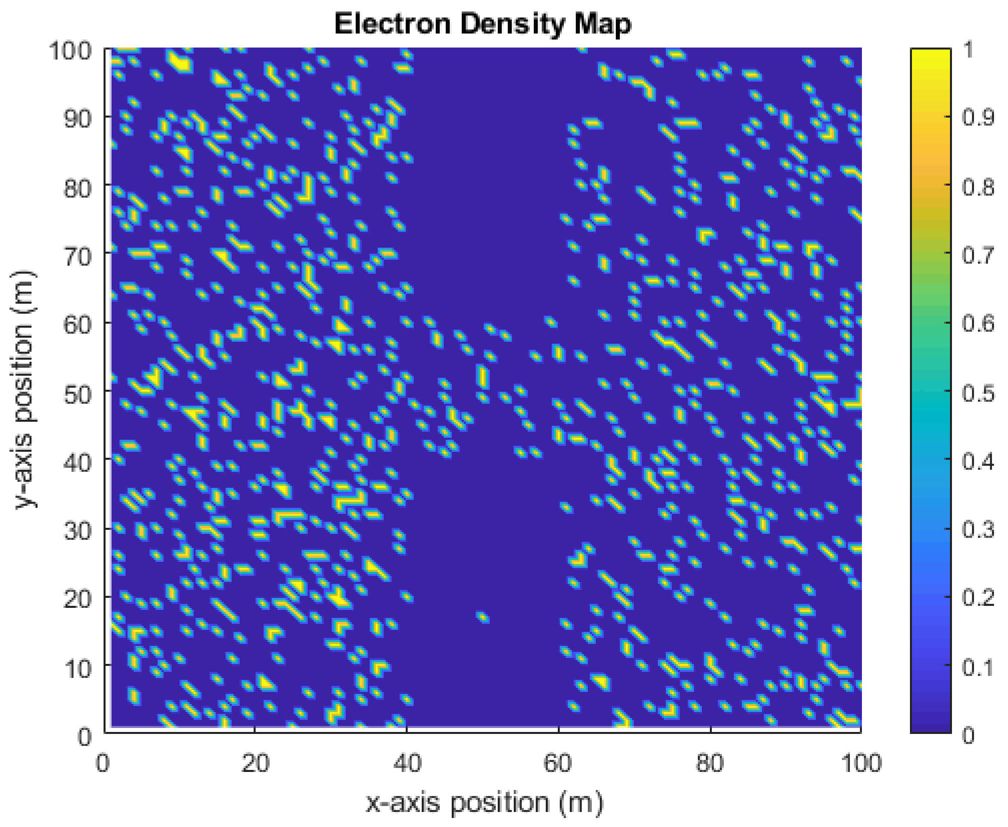
```
electron = zeros(num_electrons, 6);

% the previous position of the electron (previous x position, previous
 y
% position)
electron_prev = zeros(num_electrons, 2);

%spacial step should be smaller than 1/100 of region size
time_step = xlim/vth/100;
time_total = time_step*250;
%num_step = time_total/time_step;

% used to make each electron a different colour
electron_colour = hsv(num_electrons);

% counter used to check temperature is constant
count = 0;

% scattering probability
Pscat = 1-exp(-time_step/0.2e-12);

% box dimensions
x_lower = 0.2e-7;
x_upper = 0.3e-7;
y_lower = 0.2e-7;
y_upper = 0.3e-7;

%set an initial random postion and a fixed velocity for each electron
for i=1:num_electrons
    for j=1:6
        if(j==1)
            electron(i,j) = xlim*rand();
        elseif(j==2)
            electron(i,j) = ylim*rand();
        while((electron(i,1) >= x_lower && electron(i,1) <= x_upper
 && electron(i,2) <= y_lower) || (electron(i,1) >= x_lower &&
 electron(i,1) <= x_upper && electron(i,2) >= y_upper))
            electron(i,j) = xlim*rand();
             electron(i,j) = ylim*rand();
        end
        elseif(j==3)
            electron(i,j) = 2*pi*rand();
        elseif(j==4)
            electron(i,j) = randn()*vth;
        elseif(j==5)
            electron(i,j) = cos(electron(i,3))*electron(i,4);
        else
            electron(i,j) = sin(electron(i,3))*electron(i,4);
        end
    end
end

%Histogram commented out
%{
```

```
figure(3)
hist(electron(:,4))
title('Velocity Distribution')
%}

% define a temperature and time array for plotting
temperature= zeros(time_total/time_step,1);
time = zeros(time_total/time_step,1);

% counter for mean collision time
collision_count = 0;

running_time = 0;

% velocity array used to calculated mean free path
velocity = zeros(time_total/time_step,1);

% update each electrons positon for each time step
for k=0:time_step:time_total
    avg_temp = 0;
    avg_velocity = 0;
    for m=1:num_electrons
        % allows electrons to pass through to the other side of the
 region
        %in the x-direction
        if (electron(m,1) >= xlim)
            electron(m,1) = 0;
            electron_prev(m,1) = 0;
        elseif (electron(m,1) <= 0)
            electron(m,1) = xlim;
            electron_prev(m,1) = xlim;
        end
        % electrons are reflected at the same angle if they strike the
 limits
        % of the region in the y-driection
        if ((electron(m,2) >= ylim) || (electron(m,2) <= 0))
%             electron(m,3) = pi - electron(m,3);
%             electron(m,4) = -electron(m,4);
              electron(m,6) = -electron(m,6);
        end

        % boundary conditions when interacting with the boxes
        if(electron(m,1) <= x_lower && (electron(m,2) <= y_lower ||
 electron(m,2) >= y_upper))
            if((electron(m,1)+electron(m,5)*time_step) >= x_lower)
%                 electron(m,3) = - electron(m,3);
%                 electron(m,4) = - electron(m,4);
                  electron(m,5) = -electron(m,5);
            end
        end

        if(electron(m,1) >= x_upper && (electron(m,2) <= y_lower ||
 electron(m,2) >= y_upper))
             if((electron(m,1)+electron(m,5)*time_step) <= x_upper)
```

```
%                   electron(m,3) = - electron(m,3);
%                   electron(m,4) = - electron(m,4);
                    electron(m,5) = -electron(m,5);
              end
          end

          if((electron(m,1) >= x_lower && electron(m,1) <= x_upper) &&
 (electron(m,2) >= y_lower && electron(m,2) <= y_upper))
              if(((electron(m,2)+electron(m,6)*time_step) >= y_upper) ||
 ((electron(m,2)+electron(m,6)*time_step) <= y_lower))
%                   electron(m,3) = pi - electron(m,3);
%                   electron(m,4) = - electron(m,4);
                    electron(m,6) = -electron(m,6);
              end
          end

          % see if the particle scatters or not
%           if(Pscat > rand())
%               % scatters at a random angle
%               electron(m,3) = 2*pi*rand();
%               % new velocity for scattering - gaussian with some
%               % MAXWELL-BOLTZMAN standard deviation
%               vx_new = randn()*vth;
%               vy_new = randn()*vth;
%               v_new = sqrt(vx_new^2+vy_new^2);
%               electron(m,4) = v_new;
%               electron(m,5) = cos(electron(m,3))*v_new;
%               electron(m,5) = sin(electron(m,3))*v_new;
%               collision_count =+ 1;
%           end

          avg_temp = avg_temp + (electron(m,4)^2)*mn/(2*kB);
          avg_velocity = avg_velocity + electron(m,4);

          %plot the movement of each electron
          if(k~=0)
%               UNCOMMENT TO SEE MOVIE
%               figure(1)
%               plot([electron_prev(m,1),electron(m,1)],
[electron_prev(m,2),electron(m,2)],'color',electron_colour(m,:))
%               axis([0 xlim 0 ylim]);
%               rectangle('Position',[x_lower 0 (x_upper-x_lower)
 y_lower])
%               rectangle('Position',[x_lower y_upper (x_upper-x_lower)
 ylim])
          end
      end

      %             UNCOMMENT TO SEE MOVIE
%     title('Electron movement')
%     xlabel('x-axis position (m)')
%     ylabel('y-axis position (m)')
%     hold on
%     pause(0.001)
```

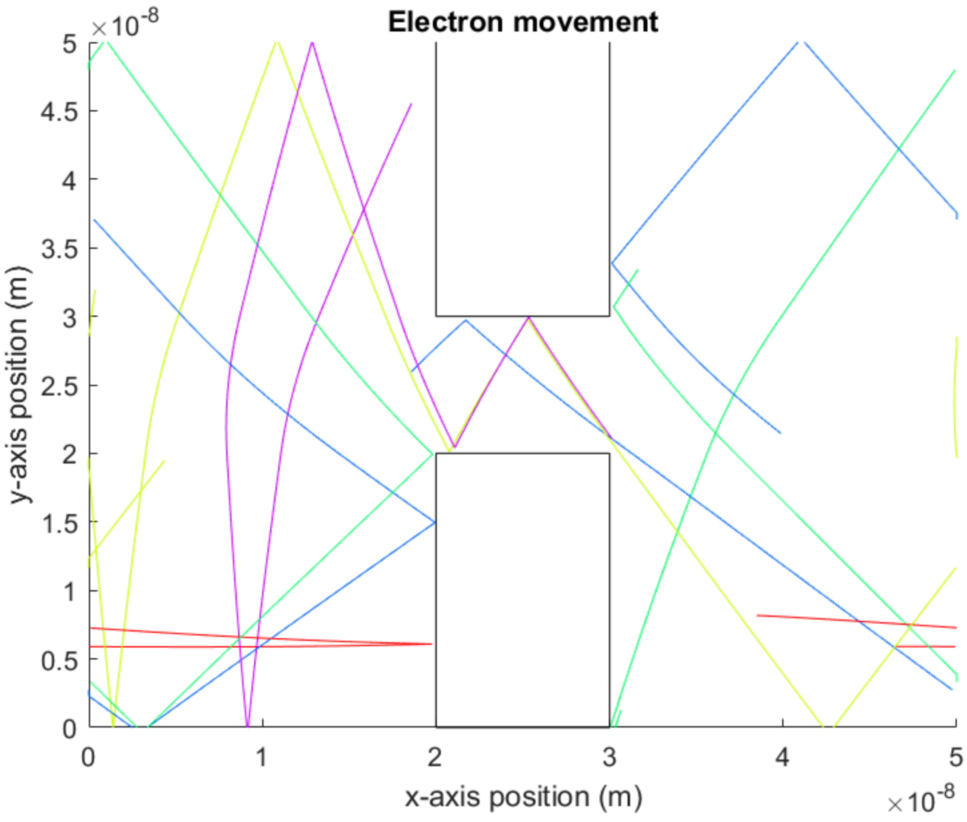
```
        % set the previous postion of the electron to the current electron
        %postion for the next itteration
        electron_prev(:,1) = electron(:,1);
        electron_prev(:,2) = electron(:,2);


        % set the electron postion to an updated position
        electron(:,1) = electron(:,1) + electron(:,5).*time_step;
        electron(:,2) = electron(:,2) + electron(:,6).*time_step;

        count = count +1;
        temperature(count,1) = avg_temp/num_electrons;
        time(count,1) = k + time_step;
        velocity(count,1) = avg_velocity;

        x_index = 1;
        y_index = 1;

        for i = 1:num_electrons
            x_index = round(electron(i,1)*1e9);
            y_index = round(electron(i,2)*1e9);

            if(x_index == 0)
                x_index = 1;
            end
            if(y_index == 0)
                y_index = 1;
            end
            if(x_index < 0)
                x_index = -1*x_index;
            end
            if(y_index < 0)
                y_index = -1*y_index;
            end
            if(x_index > 50)
                x_index = 50;
            end
            if(y_index > 50)
                y_index = 50;
            end


%           if(x_index > xlim*1e9)
%               x_index = xlim;
%           elseif(y_index > ylim*1e9)
%               y_index = ylim;
%           end

            electron(i,5) = electron(i,5) +
 accx(x_index,y_index)*time_step;
            electron(i,6) = electron(i,6) +
 accy(x_index,y_index)*time_step;
        end
```

```
end

electron_grid = zeros(100,100);

% create density regions with grid vectors of final temperature and
%electron position
for x_pos=1:100
    for y_pos=1:100
        for q = 1:num_electrons
            if((electron(q,1) <= (xlim*(x_pos/100))) && (electron(q,1)
 > (xlim*((x_pos-1)/100))) && (electron(q,2) <= (ylim*(y_pos/100))) &&
 (electron(q,2) > (ylim*((y_pos-1)/100))))
                electron_grid(x_pos,y_pos) =+ 1;
            end
        end
    end
end

figure(4)
surf(electron_grid')
colorbar
title('Electron Density Map')
xlabel('x-axis position (m)')
ylabel('y-axis position (m)')
shading interp
view(0,90)
```



The imgae above displays an electron position density map of the region for 1000 electrons.

The image below shows the electron trajectories of 5 electrons with a finte differnce electric field applied to the region. It is clear that the electrons are curving in regions where the electric field inestiy is greater and that their path is not bending as much in regions where the electric field (as displayed in part 2) is not as intense.



Another step to make the simulation better and more accurate would be to have a finer mesh when mapping the voltage or to create tunneling properties for the electrons if they reach relativitic speeds. Another improvement could be to include the interation between neighbouring electrons and how their charge would reppel other electrons which were close in proximity.

Published with MATLAB® R2018b