Assignment1: 100963181 Sonya Stuhec-Leonard

Electron Modeling Effective mass of electron Melectron = 0.26*m0, m0=rest mass Nominal size of region is 200nm X 100nm, with rectangular inserts cerntered in the x direction, with a width of 50nm

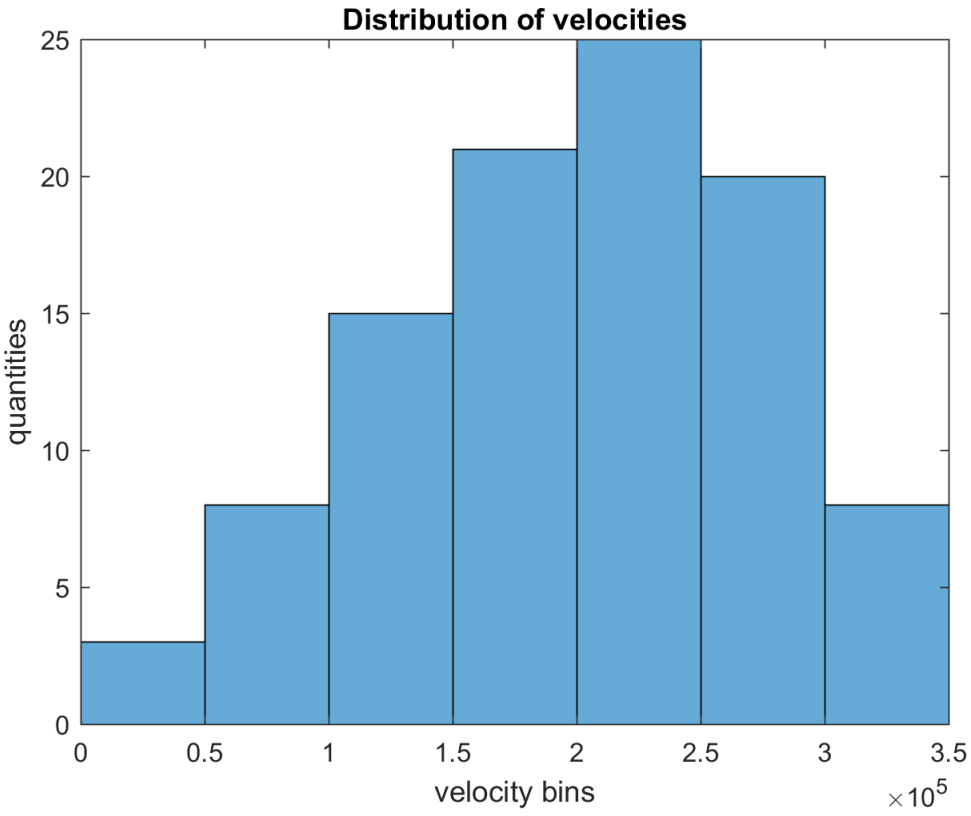
```
clear
close all

%constants
m0 = 9.109e-31; %in kg from source: https://en.wikipedia.org/wiki/
Electron
Melectron = 0.26*m0;
k = physconst('Boltzmann'); %Use of constants in matlab
T = 300; % temperature in Kalvin

%define thermal velocity (source:
%https://en.wikipedia.org/wiki/Thermal_velocity)
v_th = sqrt(k*T/Melectron);

a1 = 3; %acceleration
F1= m0*a1;
numP = 100; %number of particles
iterations = 50; %number of iterations
TauMN = 0.2e-12; %mean time between collisions

%box definitions
xmax = 200e-9;
xmin = 0;
ymax = 100e-9;
ymin = 0;

%bottleneck defineitions
TopboxYmax = ymax;
TopboxYmin = 75e-9;

BoxXmax = 125e-9;
BoxXmin = 75e-9;
BboxYmax = 25e-9;
BboxYmin = ymin;

%initalize randome positions for particles
%randmone number between 0 and 100nm or 200nm to be within box
xlocations = rand(numP, 1).*xmax;
ylocations = rand(numP, 1).*ymax;
positions = [xlocations, ylocations];

% randome velocity angle with magnitude v_th
%generate a randome inital angle for each particle in radians
% angle = rand(1, numP).*2.*pi;
% velocityX = v_th.* cos(angle);
```

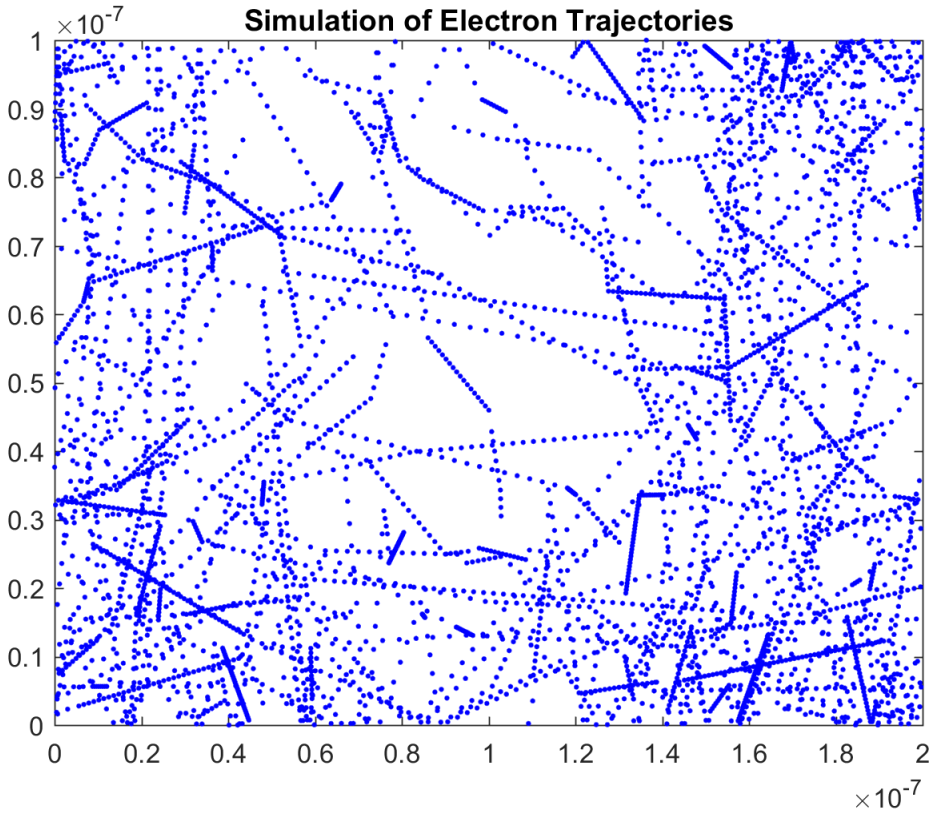
```
% velocityY = v_th.* sin(angle);
% velocity = [velocityX, velocityY];

% Randome velocity and magnitude based on Maxell boltzman
% disibution -the distribution does not work properly, it yeilds a
% parabolic profile and I was unable to determine why. (source:
% https://chem.libretexts.org/Core/Physical_and_Theoretical_Chemistry/
% Kinetics/Rate_Laws/Gas_Phase_Kinetics/Maxwell-
Boltzmann_Distributions)

MBfunc = @(v) (Melectron/(2*pi*k*T))^(1/2)*exp((Melectron*v^2)/
(2*k*T));% @(c) (4*pi.*c^2)*(Melectron/(2*pi*k*T))^(3/2)*exp(-
Melectron.*c^2/(2*k*T))
%vels is a vecotor of more than numP randome velocities to be selected
 from
%to have a range of velocities to choose from for the Maxwell-Boltzman
vels = (1:10:v_th*2);

for index = 1:length(vels)
    weight(index) =  MBfunc(index);
end

%generates randome velocity based of MBfunc distribution
RandVelX = randsample(vels,numP,true,weight);
RandVelY = randsample(vels,numP,true,weight);

%generate a randome inital angle for each particle in radians
angle = rand(1, numP).*2.*pi;
%save velocities as vectors and then as a velocity matrix
velocityX= RandVelX.* cos(angle);
velocityY = RandVelY.* sin(angle);
velocity = [velocityX; velocityY]';

velAverage = sqrt(RandVelX.^2+ RandVelY.^2);

%showing distribution of velocities
figure(1)
histogram(velAverage)
title('Distribution of velocities')
xlabel('velocity bins')
ylabel('quantities')

%use 100 steps to get across the region 200nm long
t = (200e-9/v_th)/100;

%inner boxes should not have any particles initalized inside them
for inity = 1:numP
    while (1)
        if positions(inity, 1)<=BoxXmax && positions(inity,
 1)>=BoxXmin && (positions(inity, 2)<= TopboxYmin || positions(inity,
 2)>= BboxYmax) %for inner box
            %inity=inity-1;

            %choose new y particale inital  location
```

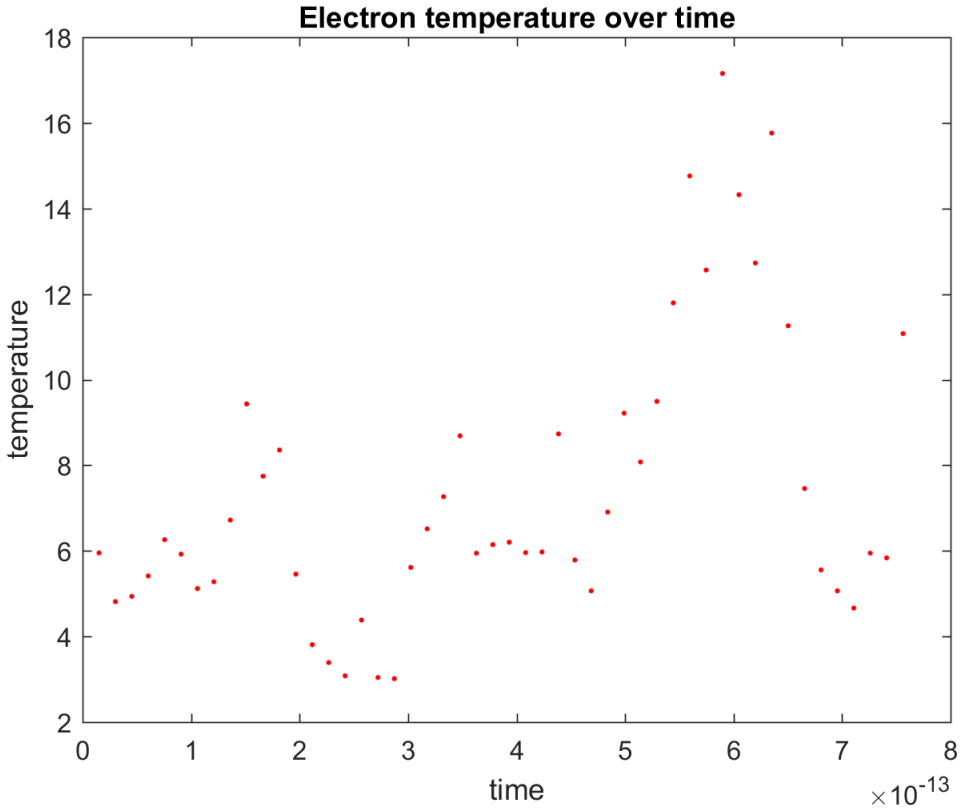
```
            positions(inity, 1) = rand(1).*xmax;
            %choose new y particale inital  location
            positions(inity, 2) = rand(1).*ymax;
        else
            break
        end
    end
end

%Probability of scattering
ProbScat = 1- exp(-t/TauMN);
scatterTime = zeros(numP, 1);


%Main loops for producing the "movie" of particles
for iter =1:iterations
    scatterTime= scatterTime+t*iter;
    %Keep position and velocity form previouse iteration
    oldP = positions;
    oldV = velocity;

    for n=1:numP
        if ProbScat > rand()
            %rethermalize the particle's velocity by assigning new Vx
            %and Vy from the MB distribution
            RandVelX = randsample(vels,1,true,weight);
            RandVelY = randsample(vels,1,true,weight);
            velocityX= RandVelX.* cos(angle(n));
            velocityY = RandVelY.* sin(angle(n));
            velocity(n, :) = [velocityX, velocityY];
            scatterTime(n,1) = 0;
        end
    end

    %Boundary conditions
    for j=1:2
        for n=1:numP

            positions(n, j) = positions(n, j) + velocity(n, j)*t;
            %restrications of x-cordinate of each particle
            if j == 1 %(for all x cordinates)
                if positions(n, 1) <= xmin

                    positions(n, 1) = xmax + velocity(n, 1)*t;

                elseif positions(n, 1)>= xmax

                    positions(n, 1)= xmin + velocity(n, 1)*t;

                end
            end

            %y parmaters of region 100X200nm
            if j == 2
```

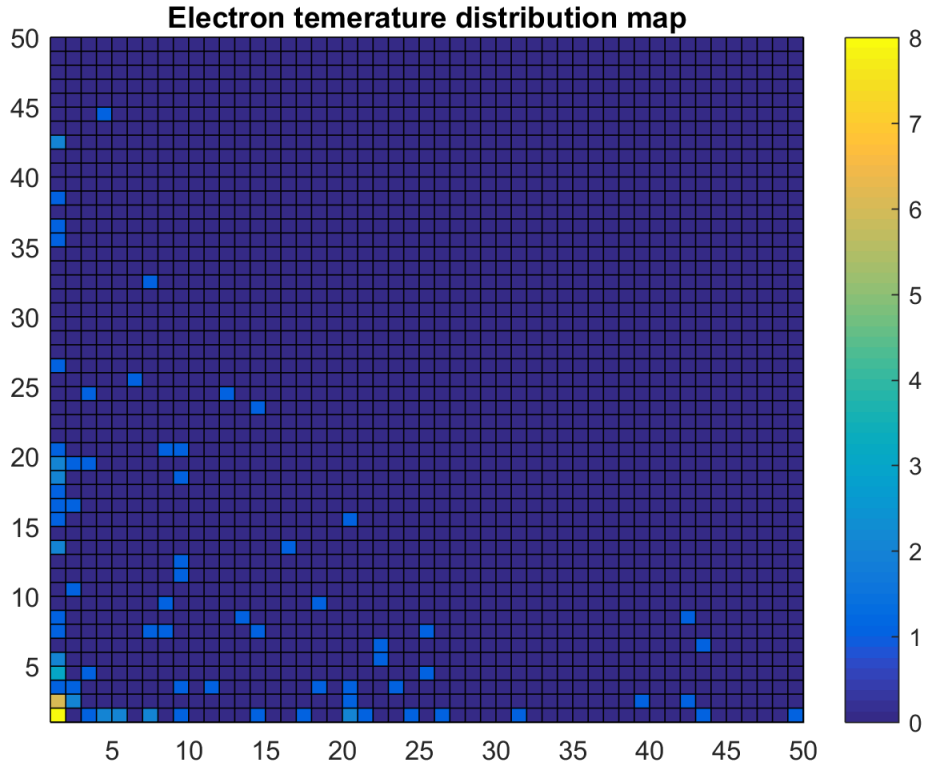
```
                if positions(n, 2) <= ymin || positions(n, 2) >= ymax
                    velocity(n, 2) = -1*velocity(n, 2);%just negate y
component

                end
            end

            %for hitting the sides of the bottleneck boxes
            if (positions(n, 2)>=TopboxYmin || positions(n,
2)<=BboxYmax) && oldP(n, 1)>= BoxXmax && oldP(n, 1)<= BoxXmin
                if positions(n, 1)<= BoxXmax && positions(n, 1)>=
BoxXmin
                    velocity(n, 1) = -0.1*velocity(n, 1); %changing
velocity magnitude removes some of the engery (difussive)
                end
            end
            %for hitting the bottleneck part of top or bottom boxes
            if positions(n, 1)<=BoxXmax && positions(n, 1)>=BoxXmin &&
oldP(n, 2)<= TopboxYmin && oldP(n, 2)>= BboxYmax
                if (positions(n, 2) <= BboxYmax || positions(n,
2)>=TopboxYmin)
                    velocity(n, 2) = -0.1*velocity(n, 2); %changing
velocity magnitude removes some of the engery (difussive)
                end

            end

            % Creating a temperature map of the electrons.
            Teperature(n, j) = Melectron.*velocity(n, j).^2./k;

        end
    end

    % Temperature formula from: https://en.wikipedia.org/wiki/
Thermal_velocity
    temp = (mean(velocity(:, 1))^2 + mean(velocity(:,
2))^2)*Melectron/k;

    TempText = strcat('Temperature:', num2str(temp));


    figure (2)
    plot(positions(:, 1), positions(:, 2), '.b')
    hold on
    h=text(0, 0,char(TempText));
    axis([xmin, xmax, ymin, ymax])
    pause(0.2)
    title ('Simulation of Electron Trajectories')

    delete(h)


    %Below would display boxes on the plot, however it is based on
fgure
```

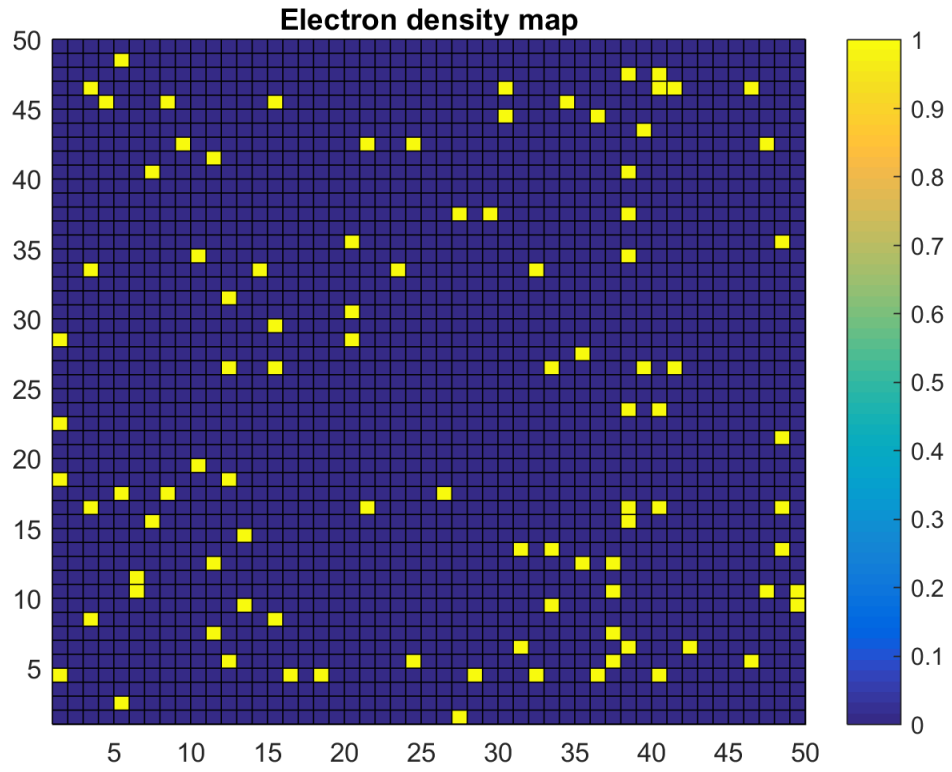
```
    %dimentions and this does not apear in the right spots
    %     dim1 = [BoxXmin/xmax (BboxYmin+11e-9)/ymax (BoxXmax-
BoxXmin)/xmax (BboxYmax-BboxYmin)/ymax];
    %     annotation('rectangle',dim1,'Color','k')
    %     dim2 = [BoxXmin/xmax (TopboxYmin-7.5e-9)/ymax (BoxXmax-
BoxXmin)/xmax (TopboxYmax-TopboxYmin)/ymax];
    %     annotation('rectangle',dim2,'Color','k')

    %Plot temperature over time
    figure(3)
    plot(iter*t, temp, '.r')
    hold on
    title ('Electron temperature over time')
    xlabel('time')
    ylabel('temperature')


    %temperature map
    %convert velocieites into temperatures, then use hist3 to bin and
 plot
    tempDistVals = (velocity.^2).*(Melectron/k);

    figure(4)
    tempDist = hist3(tempDistVals, [50, 50]);
    pcolor(tempDist')
    title ('Electron temerature distribution map')
    colorbar



%calculate the mean free path of the electrons. The time between
%collisions  is incimetned each iteratin at the top of the iter loop.
MFP = mean(scatterTime(:, 1));

end

% electron denisty map
figure(5)
density = hist3(positions, [50, 50]);
pcolor(density')
title ('Electron density map')
colorbar
```


Distribution of velocities
quantities
velocity bins
×10^5
0
5
10
15
20
25
0
0.5
1
1.5
2
2.5
3
3.5


Simulation of Electron Trajectories
×10^-7
×10^-7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


Electron temperature over time
temperature
time
×10^-13


Electron temerature distribution map



Published with MATLAB® R2016b