Assignment 1: Part 1

This assignment serves as an introduction to Monte-Carlo modeling. A Monte Carlo simulation was created modeling the motion electrons in a silicon semi-conductor. For this assignment, it is assumed that the electric field is constant and its effects are neglectic. See assignment 3 for introduction of the electric field.

The Thermal Velocity, vth, was first found using the parameters provided in the assignemnt. The calculation can be found below. The resulting thermal velocity was found to be $1.869 * 10^5 m/s$.

```
close all;

% constants
T = 300; % temperature in Kelvin
m0 = 9.11E-31;
mn = 0.26*m0;
kb = 1.38E-23;
vth = sqrt((2*kb*T)/mn);
```

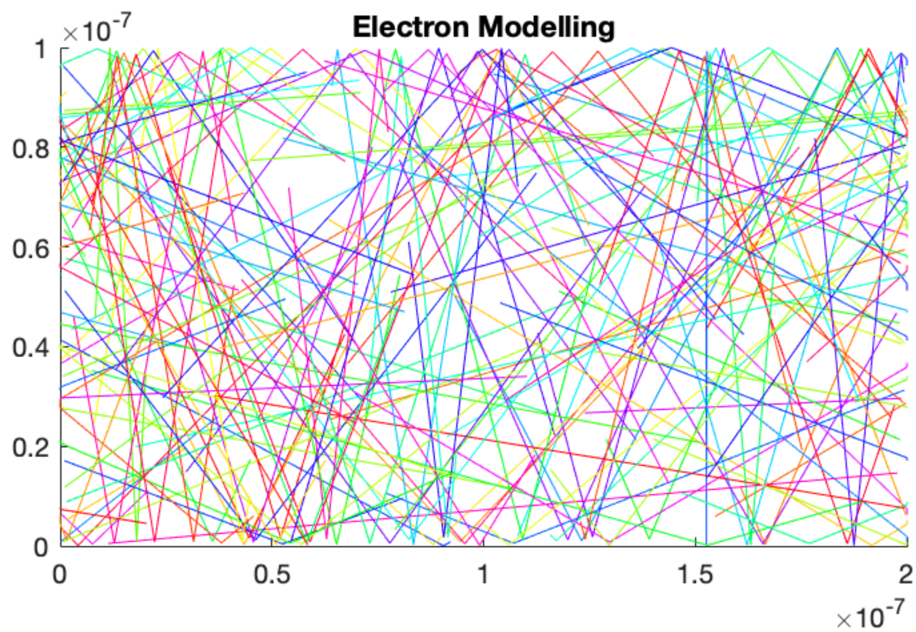
Given the mean time between collisions was 0.2 ps, the mean free path was then calculated. The mean free path (MFP) was found to be $MFP = \tau_m n * v_t h = 3.716 * 10^-5$.

The random motion of the particles was then plot in a boundary defined in the assignment. The particles were assigned a uniform velocity defined by the thermal velocity found above. The 2-D plot of the particle trajectories can be found in the figure below.

```
%region limits
xlim = 200E-9;
ylim = 100E-9;

num_particles = 100;

%initialize all of the particles
% 1 - x
% 2 - y
% 3 - direction (angle)
% 4 - vth
particle_vector = zeros(num_particles, 4);

%past position matrix
past_position = zeros(num_particles, 2);

%Setting up the timing for the plot
%time step should be smaller than 1/100 of region size
time_step = 1E-14;
total_time = time_step * 100;
number_steps = total_time/time_step;
colour = hsv(num_particles);
time = 0;
l = 0;
%Temperature
temp = zeros(number_steps, 1);
time_array = zeros(number_steps, 1);
```

```
%loop to assign an initial position and fixed velocity
for i=1:num_particles
    %loop assigning the matrix
    for j=1:4
        if (j==1)
            %assign random x position
            particle_vector(i, j) = xlim*(rand());
        elseif (j==2)
            %assign random y position
            particle_vector(i, j) = ylim*(rand());
        elseif (j==3)
            %assign some random direction
            particle_vector(i, j) = 2*(pi)*(rand());
        else
            %assign fixed velocity
            particle_vector(i, j) = vth;
        end
    end
end


%loop that updates particles position with respect to each time step
for m=0:time_step:total_time
   temp_avg = 0;
   l = l+1;
    %for loop to update each particle
    for n=1:1:num_particles

        %handle all of the boundary conditions
        %x boudary conditions - particle jumps to other side
        if (particle_vector(n, 1)>=xlim)
            particle_vector(n, 1) = 0;
            past_position(n, 1) = 0;
        elseif (particle_vector(n, 1) <= 0)
            particle_vector(n, 1) = xlim;
            past_position(n, 1) = xlim;
        end

        %y boundary conditions - particle reflects at the same angle
        if
 (((particle_vector(n,2)+particle_vector(n,4)*sin(particle_vector(n,3))*time_step)
 ylim) || ((particle_vector(n,
 2)+particle_vector(n,4)*sin(particle_vector(n,3))*time_step)<= 0))
            particle_vector(n, 3) =  pi - particle_vector(n, 3);
            particle_vector(n, 4) = - particle_vector(n, 4);
        end

        %create the plot
        if (m~=0)
            figure(1)
            %plot lines in matlab
            plot([past_position(n, 1),particle_vector(n, 1)],
[past_position(n, 2), particle_vector(n, 2)], 'color', colour(n, :));
```

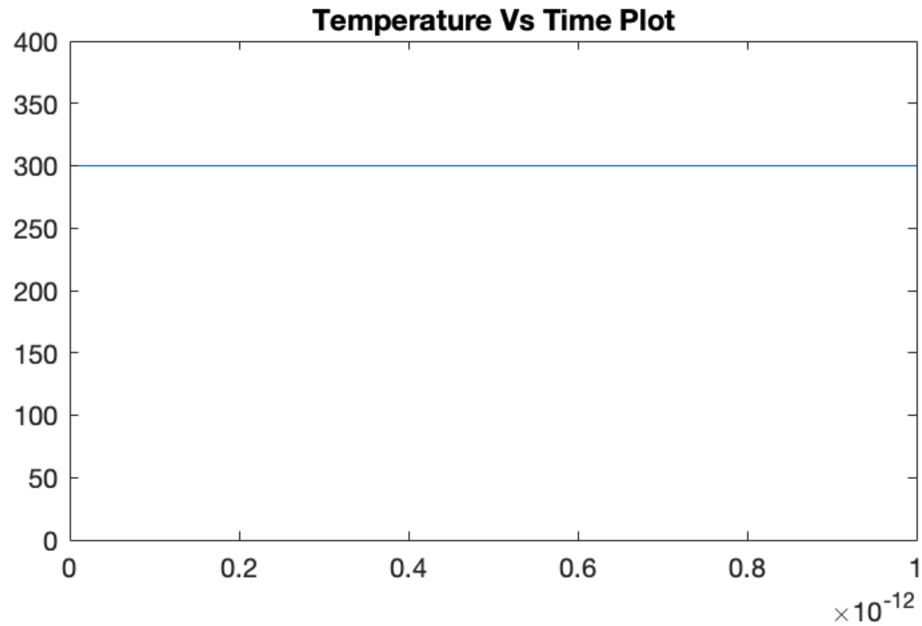
```
            axis([0 xlim 0 ylim]);
        end
        temp_avg = temp_avg + (((particle_vector(n,4))^2)*mn)/(2*kb);
    end


    %set past position equal to current position
    past_position (:, 1) = particle_vector(:, 1);
    past_position (:, 2) = particle_vector(:, 2);

    %update the current x position taking into account the velocity
    particle_vector(:,1) = particle_vector(:,1) +
 particle_vector(:,4).*cos(particle_vector(:, 3)).*time_step;
    %update the current y position taking into account the velocity
    particle_vector(:,2) = particle_vector(:,2) +
 particle_vector(:,4).*sin(particle_vector(:, 3)).*time_step;

    title 'Electron Modelling';
    hold on;
    pause(0.001);

    time = time + time_step;
    %Calculate SemiConductor Temperatures
    %temp(l, 1) = (((particle_vector(1,4))^2)*mn)/(2*kb);
    temp(l,1) = (temp_avg/num_particles);
    time_array(l, 1) = time;

end
```



Finally, a temperature plot was determined from the particles. The temperature plot can be found in the figure below.

```
figure(2)
```

```
plot(time_array, temp);
axis([0 total_time 0 400])
title 'Temperature Vs Time Plot';
```



Published with MATLAB® R2017a