Table of Contents

Part 1: Electron Modelling

To model the carriers as a population of electrons in an N-type Si semiconductor crystal, the thermal velocity is first considered. Maxwell's principle of equipartition of energy can be used. Since this simulation models an electron's movement in the X and Y directions, the system has two degrees of freedom and the equation below can be used to model the system.

$$\overline{KE} = \frac{1}{2}kT = 2(\frac{1}{2}m\overline{v^2}) \Rightarrow \overline{v^2} = \frac{2kT}{m}$$

Rearranged, the below equation is obtained:

$$vth = \sqrt{((2KT/(m))}$$

```
m0 = 9.109383e-31; %electron mass
m = 0.26*m0;       %effective mass of electron
T = 300;           %temperature in Kelvins
kb = 1.3806504e-23;%Boltzmann's constant
top_spec = 0;      % specular (1) or diffusive (0)
bottom_spec = 0;   % for part 3
vth = sqrt(2*kb*T/m)
```

```
vth =

   1.8702e+05
```

Or 187,020 m/s. The mean free path, l, is

```
l = vth*0.2e-12
```

```
l =

   3.7404e-08
```

The mean free path is about 37.4 nm.

Initial variables to control the simulations:

```
region_width = 100e-9;  %nominal size iof region: 100 nm
```

```
region_length = 200e-9; %nominal size iof region: 200 nm
iterations = 1000;      %for nominally 1000 timesteps
population_size = 4000;
plot_population = 10;
time_step = region_width/vth/100;
% The arrays below contain information about positions, velocities,
 and
% temperature.
elec_state = zeros(population_size, 4); %matrix to represent [Px Py Vx
 Vy]
elec_trajectories = zeros(iterations, plot_population*2);
temp = zeros(iterations,1);
```

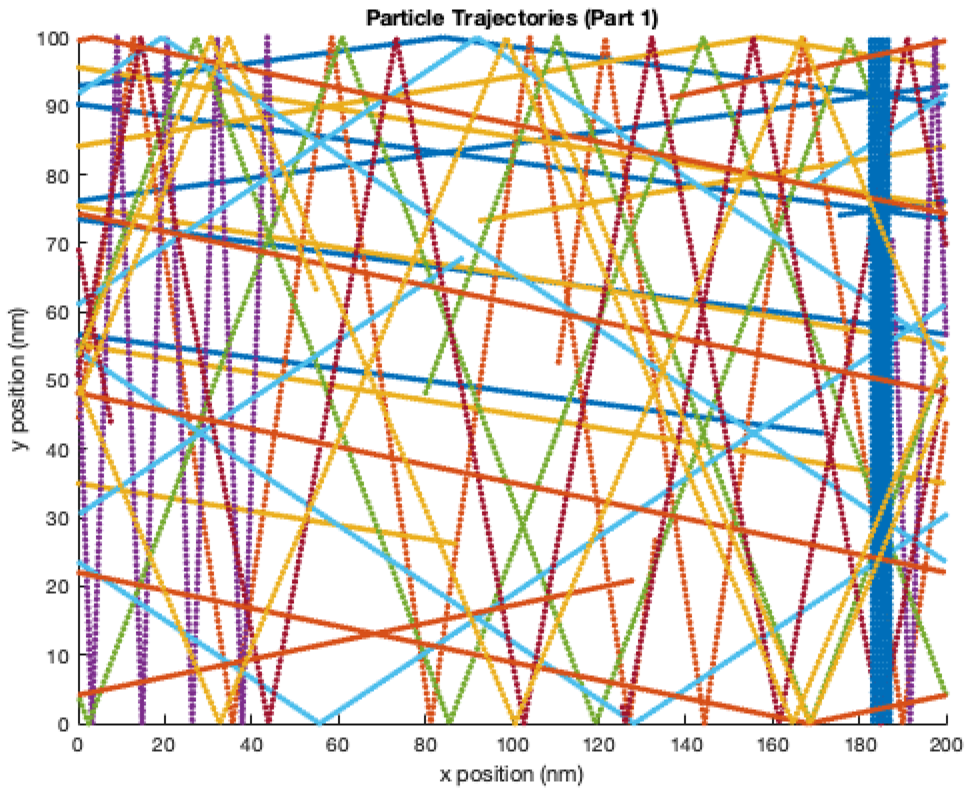
Random initial position of all electrons without change in speed

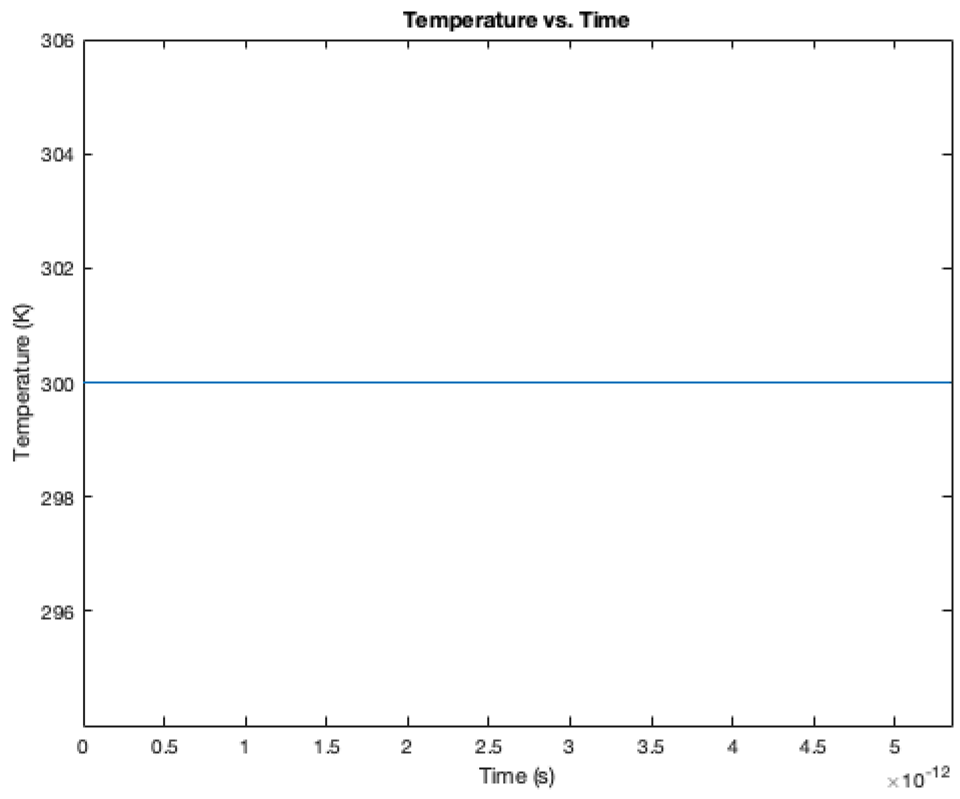
```
for i = 1:population_size
    angle = rand*pi; %generating a random angle
    elec_state(i,:) = [region_length*rand region_width*rand
 vth*cos(angle) vth*sin(angle)]; %each particle with their own
 position and velocity
end
```

Update position and plot

```
for i = 1:iterations
    %update old position with new
    elec_state(:,1:2) = elec_state(:,1:2) +
 time_step.*elec_state(:,3:4);

    % collisions with the boundaries
    j = elec_state(:,1) > region_length;
    elec_state(j,1) = elec_state(j,1) - region_length;

    j = elec_state(:,1) < 0;
    elec_state(j,1) = elec_state(j,1) + region_length;

    j = elec_state(:,2) > region_width;
    elec_state(j,2) = 2*region_width - elec_state(j,2);
    elec_state(j,4) = -elec_state(j,4);

    j = elec_state(:,2) < 0;
    elec_state(j,2) = -elec_state(j,2);
    elec_state(j,4) = -elec_state(j,4);

    temp(i) = (sum(elec_state(:,3).^2) + sum(elec_state(:,4).^2))*m/
kb/2/population_size;

    % plotting the trajectories
    for j=1:plot_population
        elec_trajectories(i, (2*j):(2*j+1)) = elec_state(j, 1:2);
    end
 end

% plot trajectories
```

```
figure(1);
%subplot(2,1,1);
title('Particle Trajectories (Part 1)');
xlabel('x position (nm)');
ylabel('y position (nm)');
axis([0 region_length/1e-9 0 region_width/1e-9]);
hold on;

for i=1:plot_population
    plot(elec_trajectories(:,i*2)./1e-9,
 elec_trajectories(:,i*2+1)./1e-9, '.');
end

figure(2);
%subplot(2,1,2);
hold off;
plot(time_step*(0:iterations-1), temp);
axis([0 time_step*iterations min(temp)*0.98 max(temp)*1.02]);

title('Temperature vs. Time');
xlabel('Time (s)');
ylabel('Temperature (K)');
```





Part 2: Collisions with Mean Free Path (MFP)

Scattering is modelled using the Maxwell-Boltzmann distribution to assign initial velocities. After scattering, the electrons get a new velocity from the Maxwell-Boltzmann distribution.

```
% Probability of scattering:
pscat = 1 - exp(-time_step/0.2e-12)
```

```
pscat =

    0.0264
```

A standard deviation of $\sqrt{kT/m}$ to set up initial particles

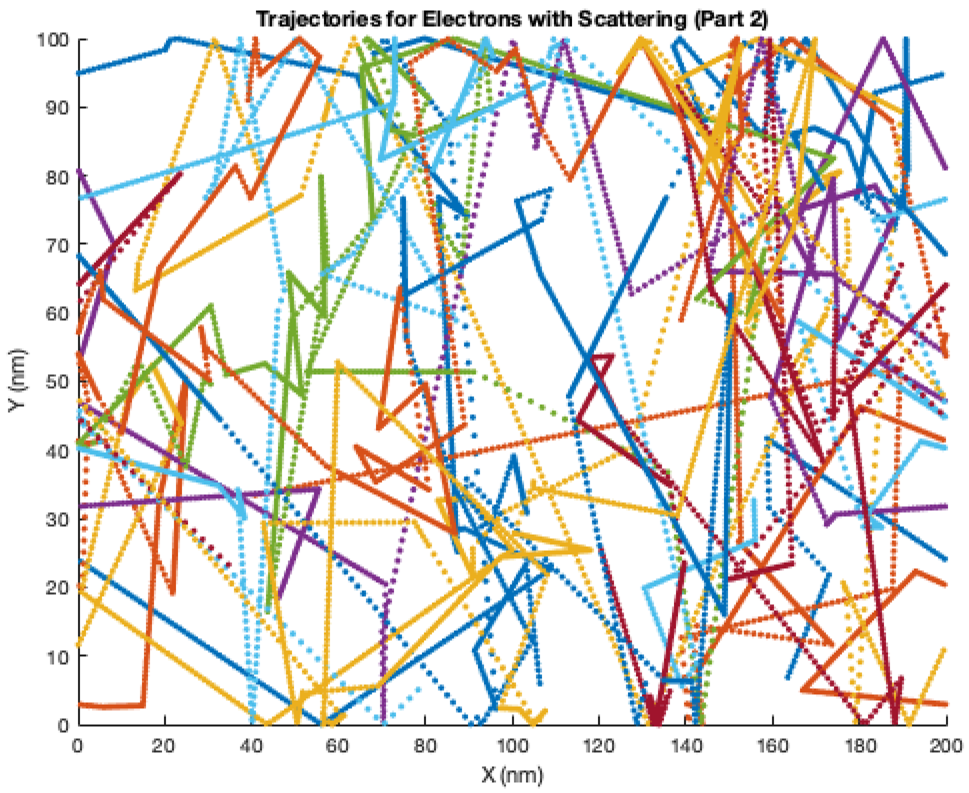
```
vdist = makedist('Normal', 'mu', 0, 'sigma', sqrt(kb*T/m));
```

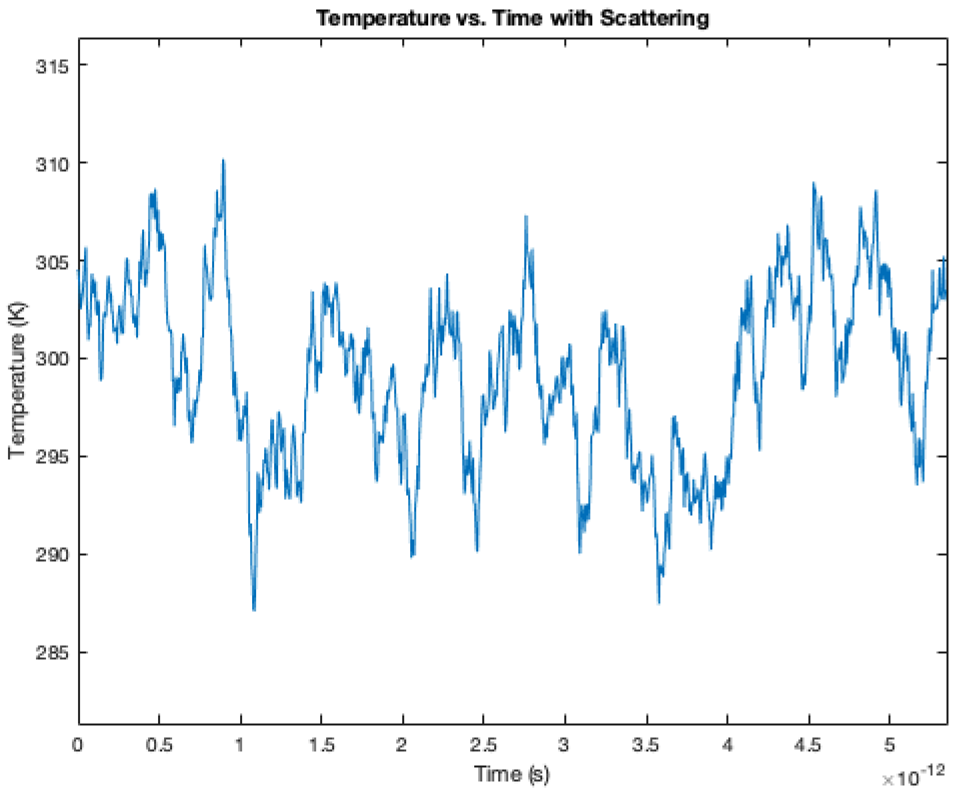
The initial population:

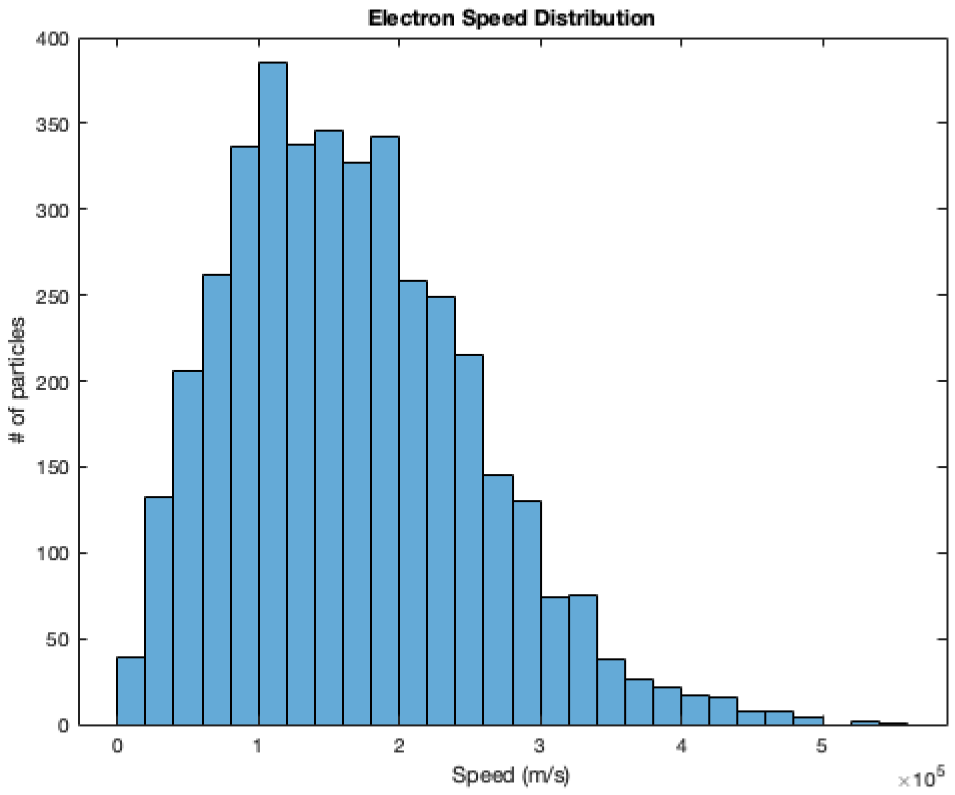
```
for i = 1:population_size
    angle = rand*2*pi;
    elec_state(i,:) = [region_length*rand region_width*rand
 random(vdist) random(vdist)];
end
```

```
for i = 1:iterations
    %Update positions
    elec_state(:,1:2) = elec_state(:,1:2) +
 time_step.*elec_state(:,3:4);

    j = elec_state(:,1) > region_length;
    elec_state(j,1) = elec_state(j,1) - region_length;

    j = elec_state(:,1) < 0;
    elec_state(j,1) = elec_state(j,1) + region_length;

    j = elec_state(:,2) > region_width;
    elec_state(j,2) = 2*region_width - elec_state(j,2);
    elec_state(j,4) = -elec_state(j,4);

    j = elec_state(:,2) < 0;
    elec_state(j,2) = -elec_state(j,2);
    elec_state(j,4) = -elec_state(j,4);

    % Scatter
    j = rand(population_size, 1) < pscat;
    elec_state(j,3:4) = random(vdist, [sum(j),2]);

    % temperature
    temp(i) = (sum(elec_state(:,3).^2) + sum(elec_state(:,4).^2))*m/
kb/2/population_size;

    % Particles to be graphed
    for j=1:plot_population
        elec_trajectories(i, (2*j):(2*j+1)) = elec_state(j, 1:2);
    end
 end

% graph trajectories
figure(3);
%subplot(3,1,1);
title('Trajectories for Electrons with Scattering (Part 2)')
xlabel('X (nm)');
ylabel('Y (nm)');
axis([0 region_length/1e-9 0 region_width/1e-9]);
hold on;

for i=1:plot_population
    plot(elec_trajectories(:,i*2)./1e-9,
 elec_trajectories(:,i*2+1)./1e-9, '.');
end

% temperature plot over time
figure(4);
%subplot(3,1,2);
hold off;
plot(time_step*(0:iterations-1), temp);
axis([0 time_step*iterations min(temp)*0.98 max(temp)*1.02]);
title('Temperature vs. Time with Scattering');
```

```
xlabel('Time (s)');
ylabel('Temperature (K)');

% speed distribution histogram
figure(5);
%subplot(3,1,3);
v = sqrt(elec_state(:,3).^2 + elec_state(:,4).^2);
histogram(v);
title('Electron Speed Distribution');
xlabel('Speed (m/s)');
ylabel('# of particles');
```




Temperature vs. Time with Scattering
Temperature (K)
Time (s)
×10^-12
315
310
305
300
295
290
285
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


Electron Speed Distribution
# of particles
Speed (m/s)
×10^5
400
350
300
250
200
150
100
50
0
1
2
3
4
5

This simulation shows that the temperature changes over time from random scattering.

Part 3: Enchancements

```
% If bounderies are specular, they bounce off in symmetircal angles.
% If bounderies are diffusive, the bounce off in random angles.
```

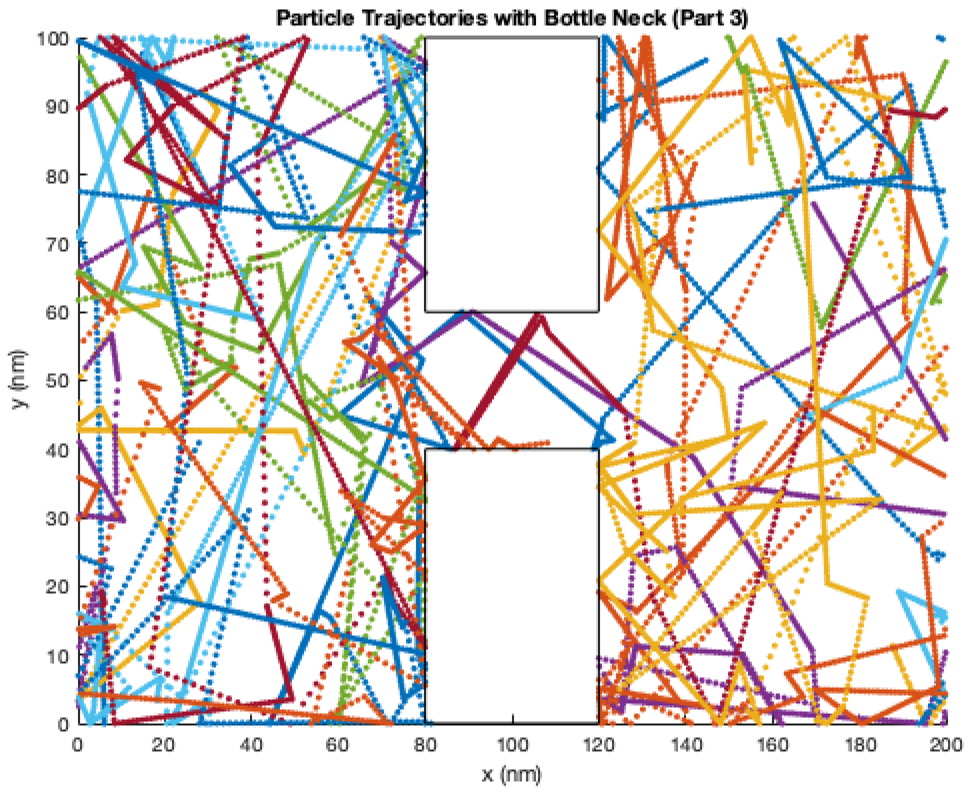
The dimensions for each box properly scaled

```
boxes = 1e-9.*[80 120 0 40; 80 120 60 100];
boxes_specular = [0 1];

% Generate an initial population
for i = 1:population_size
    angle = rand*2*pi;
    elec_state(i,:) = [region_length*rand region_width*rand
 random(vdist) random(vdist)];

    % Make sure no particles start in a box
    while(in_box(elec_state(i,1:2), boxes))
        elec_state(i,1:2) = [region_length*rand region_width*rand];
    end
end
```

Third simulation

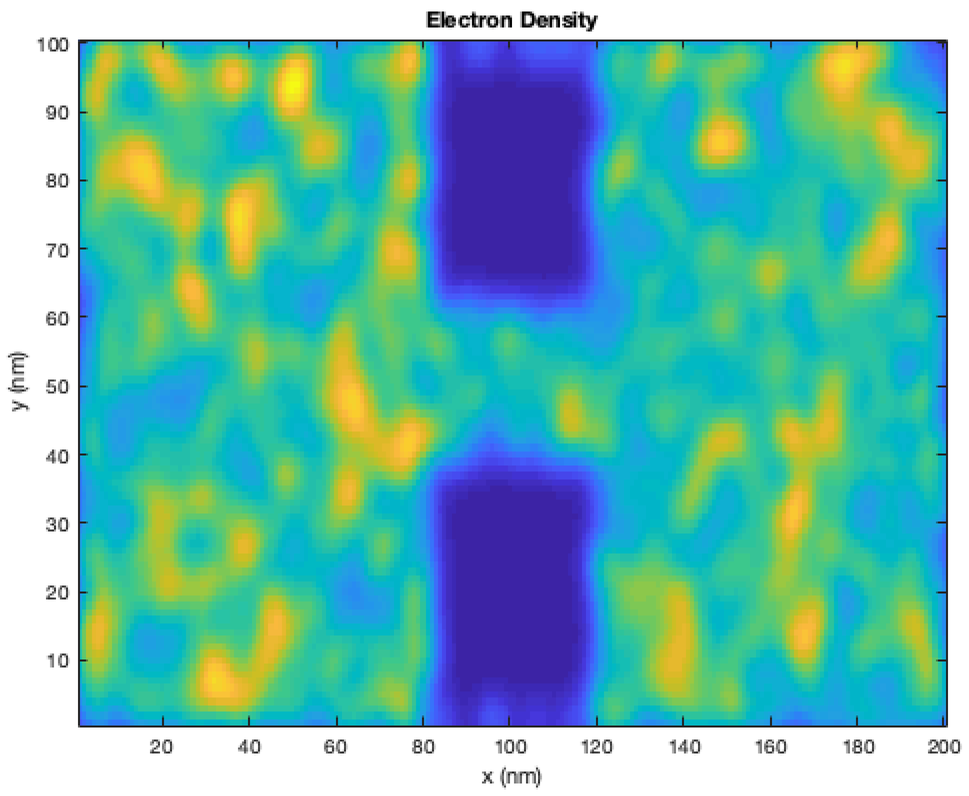
```
for i = 1:iterations
    elec_state(:,1:2) = elec_state(:,1:2) +
 time_step.*elec_state(:,3:4);

    j = elec_state(:,1) > region_length;
    elec_state(j,1) = elec_state(j,1) - region_length;

    j = elec_state(:,1) < 0;
    elec_state(j,1) = elec_state(j,1) + region_length;

    j = elec_state(:,2) > region_width;

    if(top_spec)
        elec_state(j,2) = 2*region_width - elec_state(j,2);
        elec_state(j,4) = -elec_state(j,4);

    % Diffusive: electron to bounces off at a random angle
    else
        elec_state(j,2) = region_width;
        v = sqrt(elec_state(j,3).^2 + elec_state(j,4).^2);
        angle = rand([sum(j),1])*2*pi;
        elec_state(j,3) = v.*cos(angle);
        elec_state(j,4) = -abs(v.*sin(angle));
    end

    j = elec_state(:,2) < 0;

    if(bottom_spec)
```

```
        elec_state(j,2) = -elec_state(j,2);
        elec_state(j,4) = -elec_state(j,4);

    else
        elec_state(j,2) = 0;
        v = sqrt(elec_state(j,3).^2 + elec_state(j,4).^2);
        angle = rand([sum(j),1])*2*pi;
        elec_state(j,3) = v.*cos(angle);
        elec_state(j,4) = abs(v.*sin(angle));
    end

    % Moving particles out of the box
    for j=1:population_size
        box_num = in_box(elec_state(j,1:2), boxes);
        while(box_num ~= 0)
            % Side in which electron collides
            x_dist = 0;
            new_x = 0;
            if(elec_state(j,3) > 0)
                x_dist = elec_state(j,1) - boxes(box_num,1);
                new_x = boxes(box_num,1);
            else
                x_dist = boxes(box_num,2) - elec_state(j,1);
                new_x = boxes(box_num,2);
            end

            y_dist = 0;
            new_y = 0;
            if(elec_state(j,4) > 0)
                y_dist = elec_state(j,2) - boxes(box_num, 3);
                new_y = boxes(box_num, 3);
            else
                y_dist = boxes(box_num, 4) - elec_state(j,2);
                new_y = boxes(box_num, 4);
            end

            if(x_dist < y_dist)
                elec_state(j,1) = new_x;
                if(~boxes_specular(box_num))
                    sgn = -sign(elec_state(j,3));
                    v = sqrt(elec_state(j,3).^2 + elec_state(j,4).^2);
                    angle = rand()*2*pi;
                    elec_state(j,3) = sgn.*abs(v.*cos(angle));
                    elec_state(j,4) = v.*sin(angle);
                else % Specular
                    elec_state(j,3) = -elec_state(j,3);
                end
            else
                elec_state(j,2) = new_y;
                if(~boxes_specular(box_num))
                    sgn = -sign(elec_state(j,4));
                    v = sqrt(elec_state(j,3).^2 + elec_state(j,4).^2);
                    angle = rand()*2*pi;
                    elec_state(j,3) = v.*cos(angle);
                    elec_state(j,4) = sgn.*abs(v.*sin(angle));
                else % Specular
                    elec_state(j,4) = -elec_state(j,4);
                end
            end

            box_num = in_box(elec_state(j,1:2), boxes);
        end
    end


    % Scatter particles
    j = rand(population_size, 1) < pscat;
    elec_state(j,3:4) = random(vdist, [sum(j),2]);

    % temperature
    %temp(i) = (sum(elec_state(:,3).^2) + sum(elec_state(:,4).^2))*m/
kb/2/population_size;

    % Record positions
    for j=1:plot_population
        elec_trajectories(i, (2*j):(2*j+1)) = elec_state(j, 1:2);
    end
end

%plot
figure(6);
%subplot(3,1,1);
title('Particle Trajectories with Bottle Neck (Part 3)');
xlabel('x (nm)');
ylabel('y (nm)');
axis([0 region_length/1e-9 0 region_width/1e-9]);
hold on;

for i=1:plot_population
    plot(elec_trajectories(:,i*2)./1e-9,
 elec_trajectories(:,i*2+1)./1e-9, '.');

end

% Plot boxes
for j=1:size(boxes,1)
   plot([boxes(j, 1) boxes(j, 1) boxes(j, 2) boxes(j, 2) boxes(j,
 1)]./1e-9,...
        [boxes(j, 3) boxes(j, 4) boxes(j, 4) boxes(j, 3) boxes(j,
 3)]./1e-9, 'k-');
end
```



creating electron density map

```
density = hist3(elec_state(:,1:2),[200 100])';

% electron density
figure(7);
imagesc(conv2(density,f,'same'));
set(gca,'YDir','normal');
title('Electron Density');
xlabel('x (nm)');
ylabel('y (nm)');
```



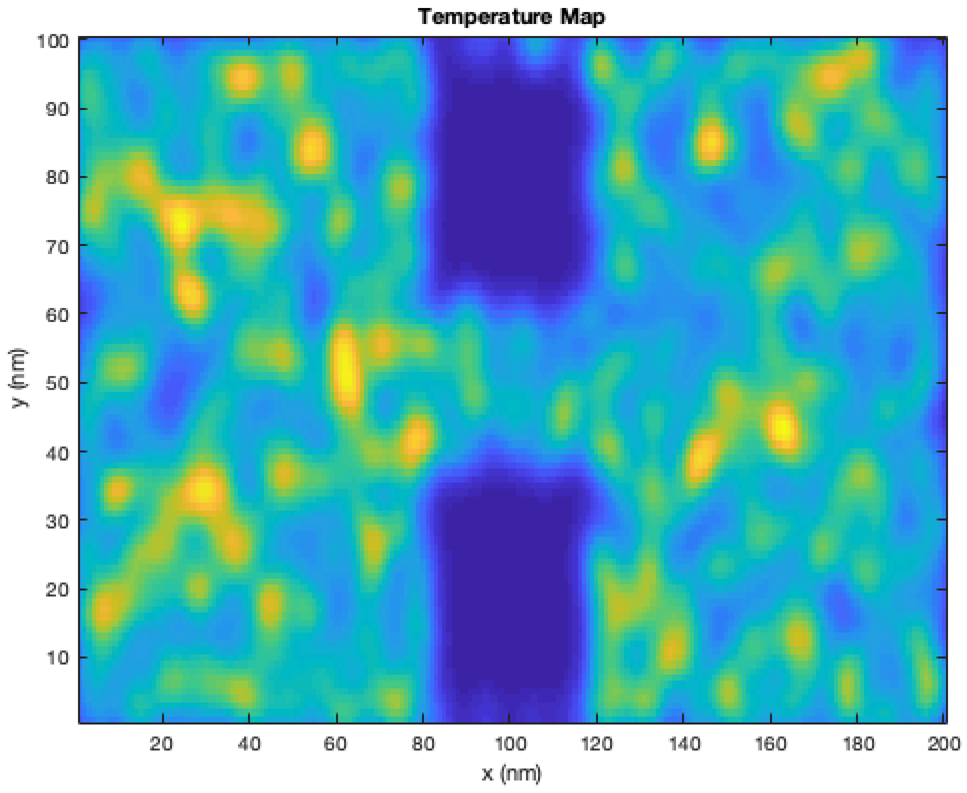
Temperature map

```
temp_sum_x = zeros(ceil(region_length/1e-9),ceil(region_width/1e-9));
temp_sum_y = zeros(ceil(region_length/1e-9),ceil(region_width/1e-9));
temp_num = zeros(ceil(region_length/1e-9),ceil(region_width/1e-9));

% Velocities of all particles
for i=1:population_size
    x = floor(elec_state(i,1)/1e-9);
    y = floor(elec_state(i,2)/1e-9);
    if(x==0)
        x = 1;
    end
    if(y==0)
        y= 1;
    end

    % Add all velocity components
    temp_sum_y(x,y) = temp_sum_y(x,y) + elec_state(i,3)^2;
    temp_sum_x(x,y) = temp_sum_x(x,y) + elec_state(i,4)^2;
    temp_num(x,y) = temp_num(x,y) + 1;
end
```

Calculating the temperatures:

```
temp = (temp_sum_x + temp_sum_y).*m./kb./2./temp_num;
temp(isnan(temp)) = 0;
```

```
temp = temp';

%temperature map
figure(8);
imagesc(conv2(temp,f,'same'));
set(gca,'YDir','normal');
title('Temperature Map');
xlabel('x (nm)');
ylabel('y (nm)');
```



Published with MATLAB® R2018b