Table of Contents

Ville Tiukuvaara, 100934692

Part 1: Electron Modelling

To calculate the thermal energy, it is noted that there are two degrees of freedom for the electrons. Using Maxwell's principle of equipartition of energy,

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

```
clear all
close all
m0 = 9.10938356e-31;
m = 0.26*m0;
T = 300;
k = 1.38064852e-23;
vth = sqrt(2*k*T/m)
```

```

vth =

   1.8702e+05

```

Or 187 km/s. The mean free path, l, is simply

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

```

l =

   3.7404e-08

```

Thus the mean free path is about 37.4 nm.

Here are some variables to control the simulations:

```
height = 100e-9;
length = 200e-9;
population_size = 3000;
plot_population = 10;
time_step = height/vth/100;
iterations = 1000;
% Set to 1 to watch the movies,
% or to 0 to just see the final plots
```

```
show_movie = 0;
```

For the simulations, these arrays will hold information about the state of the system, including the positions, velocities, and temperatures.

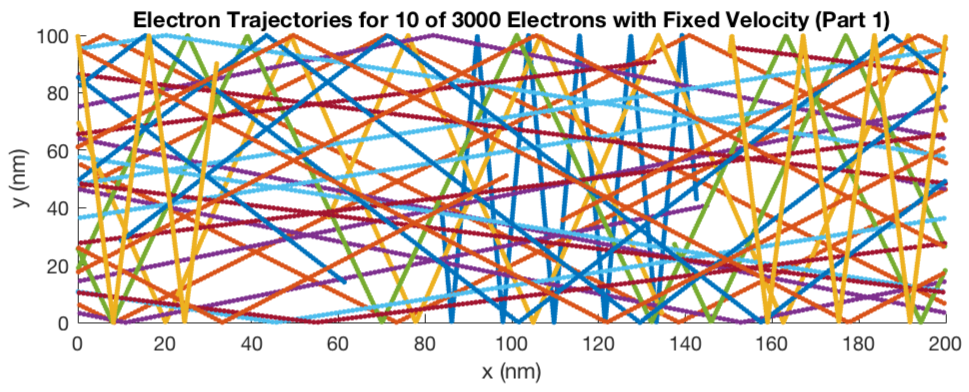
```
% Each row corresponds to an electron with the positions and
 velocities
% [x y vx vy]
state = zeros(population_size, 4);
trajectories = zeros(iterations, plot_population*2);
temperature = zeros(iterations,1);
```

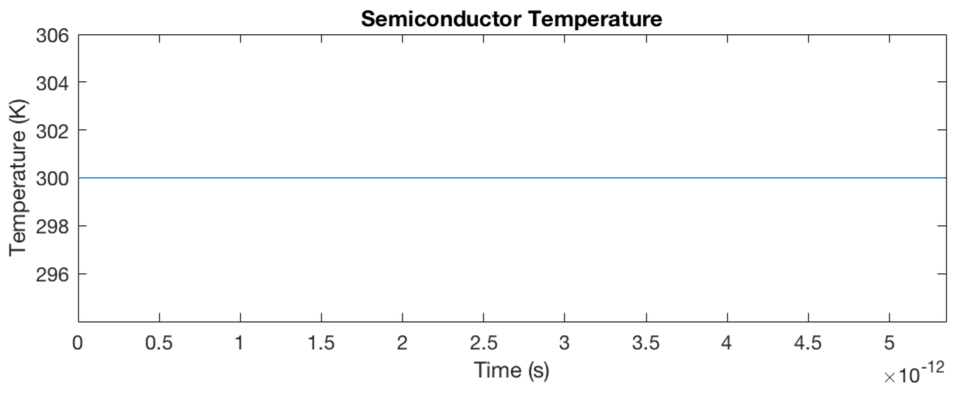
Generate an initial population with constant speeds

```
for i = 1:population_size
    angle = rand*2*pi;
    state(i,:) = [length*rand height*rand vth*cos(angle)
 vth*sin(angle)];
end
```

Iterate over time and update the positions, while plotting the state.

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

    % Look for collisions with the boundaries
    j = state(:,1) > length;
    state(j,1) = state(j,1) - length;

    j = state(:,1) < 0;
    state(j,1) = state(j,1) + length;

    j = state(:,2) > height;
    state(j,2) = 2*height - state(j,2);
    state(j,4) = -state(j,4);

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

    temperature(i) = (sum(state(:,3).^2) + sum(state(:,4).^2))*m/k/2/
population_size;

    % Record the trajectories
    for j=1:plot_population
        trajectories(i, (2*j):(2*j+1)) = state(j, 1:2);
    end

    % Update the movie every 5 iterations
    if show_movie && mod(i,5) == 0
        figure(1);
        subplot(2,1,1);
        hold off;
        plot(state(1:plot_population,1)./1e-9,
 state(1:plot_population,2)./1e-9, 'o');
        axis([0 length/1e-9 0 height/1e-9]);
        title(sprintf('Trajectories for %d of %d Electrons with Fixed
 Velocity (Part 1)',...
        plot_population, population_size));
        xlabel('x (nm)');
        ylabel('y (nm)');
        if i > 1
            subplot(2,1,2);
            hold off;
            plot(time_step*(0:i-1), temperature(1:i));
            axis([0 time_step*iterations min(temperature)*0.98
 max(temperature)*1.02]);
            title('Semiconductor Temperature');
            xlabel('Time (s)');
            ylabel('Temperature (K)');
        end
        pause(0.05);
    end
end

% Show trajectories after the movie is over
figure(1);
subplot(2,1,1);
title(sprintf('Electron Trajectories for %d of %d Electrons with Fixed
 Velocity (Part 1)',...
    plot_population, population_size));
xlabel('x (nm)');
ylabel('y (nm)');
axis([0 length/1e-9 0 height/1e-9]);
hold on;
for i=1:plot_population
    plot(trajectories(:,i*2)./1e-9, trajectories(:,i*2+1)./1e-9, '.');
end

if(~show_movie)
    subplot(2,1,2);
    hold off;
    plot(time_step*(0:iterations-1), temperature);
    axis([0 time_step*iterations min(temperature)*0.98
 max(temperature)*1.02]);
    title('Semiconductor Temperature');
    xlabel('Time (s)');
    ylabel('Temperature (K)');
end
```





Part 2: Collisions with Mean Free Path

For the second simulation, the initial velocities are assigned based on a Maxwell-Boltzmann distribution, and scattering is modelled. First, calculate the scattering probability in one time step:

```
p_scat = 1 - exp(-time_step/0.2e-12)
```

```
p_scat =

    0.0264
```

The distribution of velocities in x and y is Gaussian, with a standard deviation of $\sqrt{kT/m}$. This results in an overall Maxwell-Boltzmann velocity distribution at temperature T

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

The initial population is generated using this distribution:

```
for i = 1:population_size
    angle = rand*2*pi;
    state(i,:) = [length*rand height*rand random(v_pdf)
 random(v_pdf)];
end
```

The average velocity should be calculated to be correct:

```
avg_v = sqrt(sum(state(:,3).^2)/population_size + ...
    sum(state(:,4).^2)/population_size)


avg_v =

   1.8836e+05
```

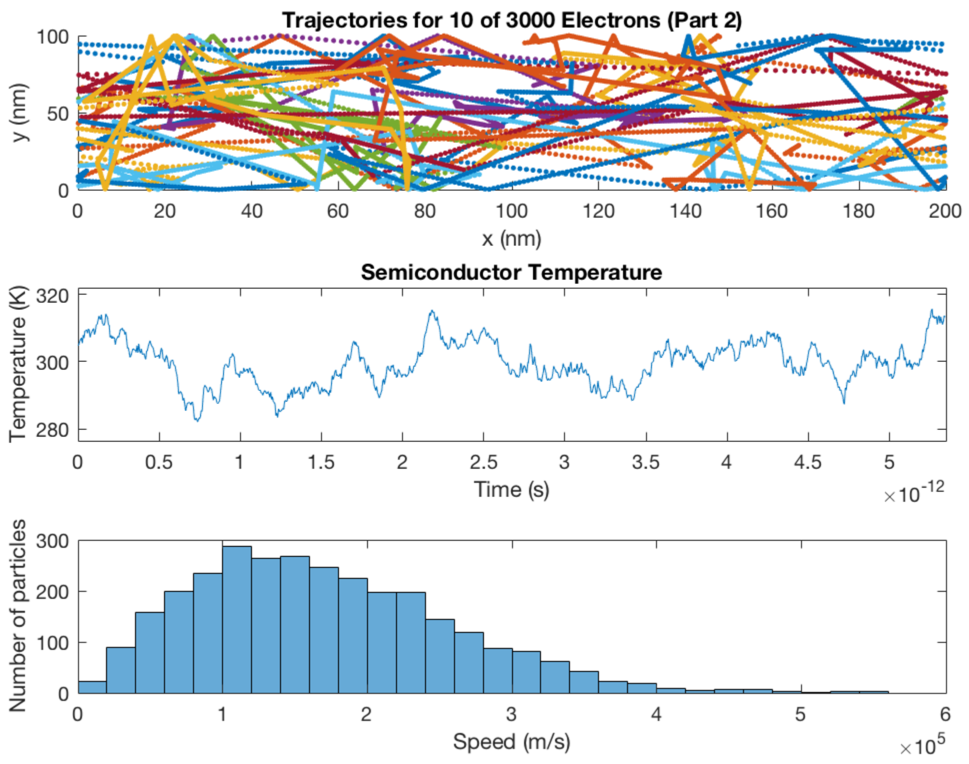
This returns a velocity of about 187 km/s, which is correct. This varies a little bit, since the initial velocities are random with a MB distribution.

The second simulation loop follows:

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

    j = state(:,1) > length;
    state(j,1) = state(j,1) - length;

    j = state(:,1) < 0;
    state(j,1) = state(j,1) + length;

    j = state(:,2) > height;
    state(j,2) = 2*height - state(j,2);
    state(j,4) = -state(j,4);

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

    % Scatter particles
    j = rand(population_size, 1) < p_scat;
    state(j,3:4) = random(v_pdf, [sum(j),2]);

    % Record the temperature
    temperature(i) = (sum(state(:,3).^2) + sum(state(:,4).^2))*m/k/2/
population_size;

    % Record positions for subset of particles that will be graphed
    for j=1:plot_population
        trajectories(i, (2*j):(2*j+1)) = state(j, 1:2);
    end

    % Update the movie every 5 iterations
    if show_movie && mod(i,5) == 0
        figure(2);
        subplot(3,1,1);
        hold off;
        plot(state(1:plot_population,1)./1e-9,
 state(1:plot_population,2)./1e-9, 'o');
        axis([0 length/1e-9 0 height/1e-9]);
        title(sprintf('Trajectories for %d of %d Electrons (Part
 2)',...
```

```
        plot_population, population_size));
        xlabel('x (nm)');
        ylabel('y (nm)');
        if i > 1
            subplot(3,1,2);
            hold off;
            plot(time_step*(0:i-1), temperature(1:i));
            axis([0 time_step*iterations min(temperature)*0.98
 max(temperature)*1.02]);
            title('Semiconductor Temperature');
            xlabel('Time (s)');
            ylabel('Temperature (K)');
        end

        % Show histogram of speeds
        subplot(3,1,3);
        v = sqrt(state(:,3).^2 + state(:,4).^2);
        title('Histogram of Electron Speeds');
        histogram(v);
        xlabel('Speed (m/s)');
        ylabel('Number of particles');

        pause(0.05);
    end
end

% Show trajectories after the movie is over
figure(2);
subplot(3,1,1);
title(sprintf('Trajectories for %d of %d Electrons (Part 2)',...
    plot_population, population_size));
xlabel('x (nm)');
ylabel('y (nm)');
axis([0 length/1e-9 0 height/1e-9]);
hold on;
for i=1:plot_population
    plot(trajectories(:,i*2)./1e-9, trajectories(:,i*2+1)./1e-9, '.');
end

% Show temperature plot over time
if(~show_movie)
    subplot(3,1,2);
    hold off;
    plot(time_step*(0:iterations-1), temperature);
    axis([0 time_step*iterations min(temperature)*0.98
 max(temperature)*1.02]);
    title('Semiconductor Temperature');
    xlabel('Time (s)');
    ylabel('Temperature (K)');
end

% Show speed histogram
subplot(3,1,3);
v = sqrt(state(:,3).^2 + state(:,4).^2);
```

```
title('Histogram of Electron Speeds');
histogram(v);
xlabel('Speed (m/s)');
ylabel('Number of particles');
```



This second simulation shows that the average temperature fluctuates over time due to the scattering, but it has an average of 300K over time.

Part 3: Enchancements

Here, the boundaries can be set to be specular or diffusive. If they are diffusive, the electrons bounce off at a random angle rather than one symmetrical about the normal with the boundary.

The non-periodic top and bottom boundaries can be set to be either specular (1) or diffusive (0) with the following parameters:

```
top_specular = 0;
bottom_specular = 0;
```

This simulation also includes boxes (obstacles) Also, each box can seperately be set to be specular (1) or diffusive (0)

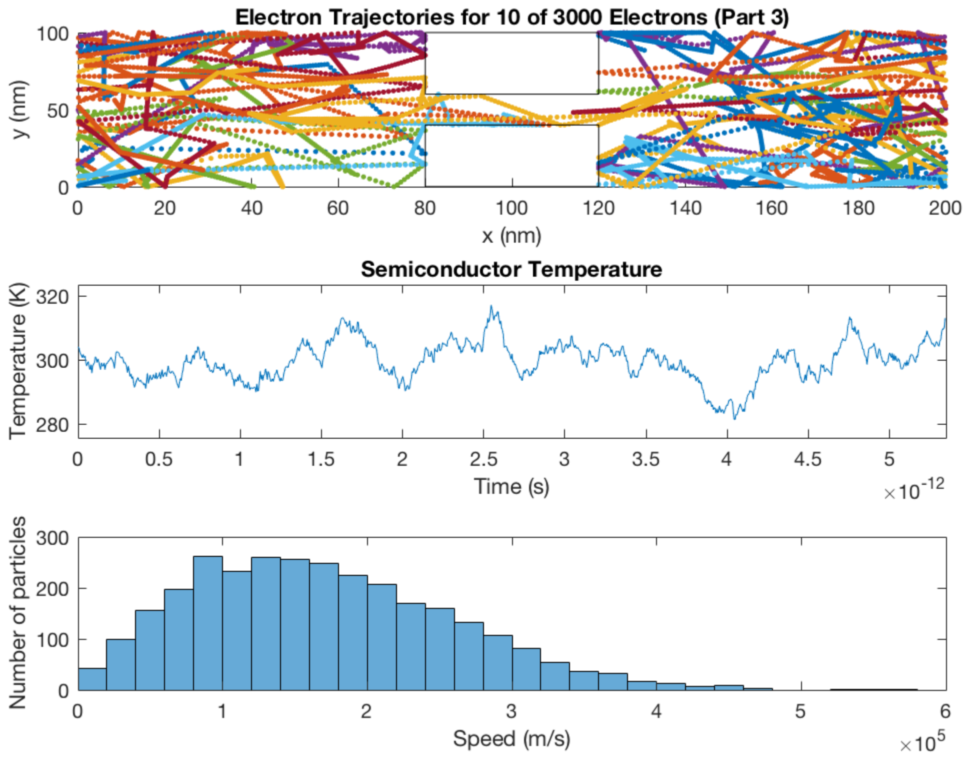
```
% The values are [xmin xmax ymin ymax] for each box
boxes = 1e-9.*[80 120 0 40; 80 120 60 100];
boxes_specular = [0 1];

% Generate an initial population
```

```
for i = 1:population_size
    angle = rand*2*pi;
    state(i,:) = [length*rand height*rand random(v_pdf)
 random(v_pdf)];

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

Run through the third simulation:

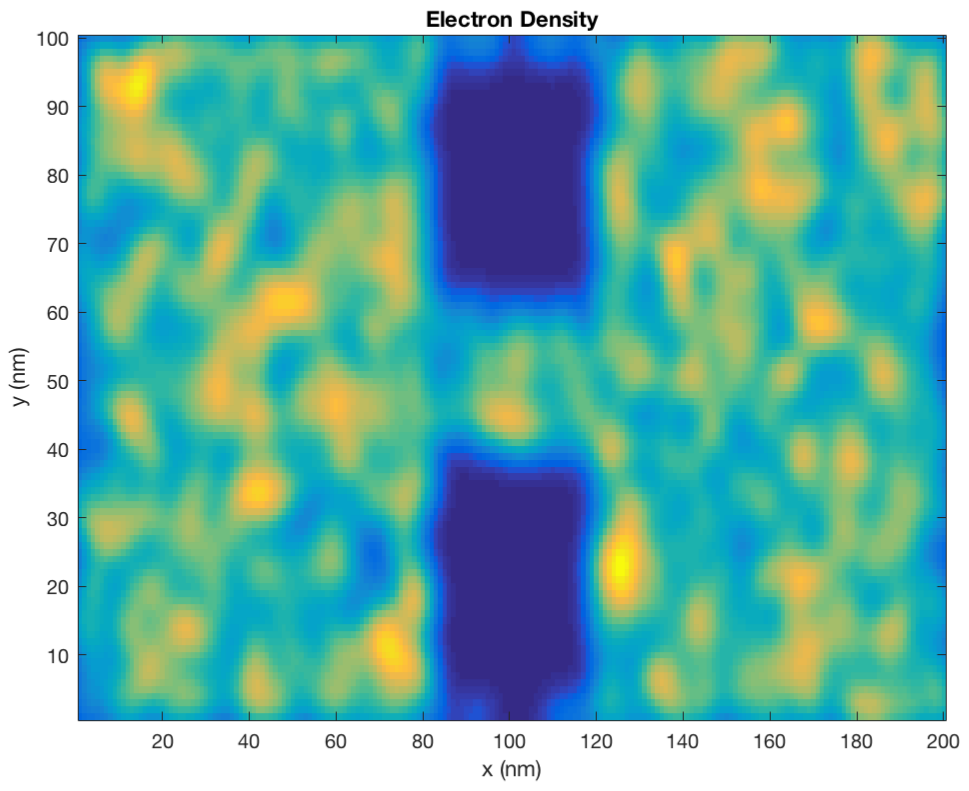
```
for i = 1:iterations
    state(:,1:2) = state(:,1:2) + time_step.*state(:,3:4);

    j = state(:,1) > length;
    state(j,1) = state(j,1) - length;

    j = state(:,1) < 0;
    state(j,1) = state(j,1) + length;

    j = state(:,2) > height;

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

    j = state(:,2) < 0;

    if(bottom_specular)
        state(j,2) = -state(j,2);
        state(j,4) = -state(j,4);
    else % Diffusive
        % The electron bounces off at a random angle
        state(j,2) = 0;
        v = sqrt(state(j,3).^2 + state(j,4).^2);
        angle = rand([sum(j),1])*2*pi;
        state(j,3) = v.*cos(angle);
        state(j,4) = abs(v.*sin(angle));
    end

    % Look for particles that have "entered" a box and move them to
    % where they should be.
    for j=1:population_size
        box_num = in_box(state(j,1:2), boxes);
        while(box_num ~= 0)
            % To see which side the electron collided with,
            % find which one it's closer to
            x_dist = 0;
            new_x = 0;
            if(state(j,3) > 0)
                x_dist = state(j,1) - boxes(box_num,1);
                new_x = boxes(box_num,1);
            else
                x_dist = boxes(box_num,2) - state(j,1);
                new_x = boxes(box_num,2);
            end

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

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

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


    % Scatter particles
    j = rand(population_size, 1) < p_scat;
    state(j,3:4) = random(v_pdf, [sum(j),2]);

    % Record the temperature
    temperature(i) = (sum(state(:,3).^2) + sum(state(:,4).^2))*m/k/2/
population_size;

    % Record positions for subset of particles that will be graphed
    for j=1:plot_population
        trajectories(i, (2*j):(2*j+1)) = state(j, 1:2);
    end

    % Update the movie every 5 iterations
    if show_movie && mod(i,5) == 0
        figure(3);
        subplot(3,1,1);
        hold off;
        plot(state(1:plot_population,1)./1e-9,
 state(1:plot_population,2)./1e-9, 'o');
        hold on;

        % Plot the 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

        axis([0 length/1e-9 0 height/1e-9]);
        title(sprintf('Trajectories for %d of %d Electrons (Part
 3)',...
        plot_population, population_size));
        xlabel('x (nm)');
        ylabel('y (nm)');
        if i > 1
            subplot(3,1,2);
            hold off;
            plot(time_step*(0:i-1), temperature(1:i));
            axis([0 time_step*iterations min(temperature(1:i))*0.98
 max(temperature)*1.02]);
            title('Semiconductor Temperature');
            xlabel('Time (s)');
            ylabel('Temperature (K)');
        end

        subplot(3,1,3);
        v = sqrt(state(:,3).^2 + state(:,4).^2);
        title('Histogram of Electron Speeds');
        histogram(v);
        xlabel('Speed (m/s)');
        ylabel('Number of particles');

        pause(0.05);
    end
```

```
end

% Show trajectories after the movie is over
figure(3);
subplot(3,1,1);
title(sprintf('Electron Trajectories for %d of %d Electrons (Part
 3)',...
    plot_population, population_size));
xlabel('x (nm)');
ylabel('y (nm)');
axis([0 length/1e-9 0 height/1e-9]);
hold on;
for i=1:plot_population
    plot(trajectories(:,i*2)./1e-9, trajectories(:,i*2+1)./1e-9, '.');

end

% Plot the 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

% Plot temperature
if(~show_movie)
    subplot(3,1,2);
    hold off;
    plot(time_step*(0:iterations-1), temperature);
    axis([0 time_step*iterations min(temperature)*0.98
 max(temperature)*1.02]);
    title('Semiconductor Temperature');
    xlabel('Time (s)');
    ylabel('Temperature (K)');
end

subplot(3,1,3);
v = sqrt(state(:,3).^2 + state(:,4).^2);
title('Histogram of Electron Speeds');
histogram(v);
xlabel('Speed (m/s)');
ylabel('Number of particles');
```



For the final simulation, an electron density map is created, by creating a 2D histogram over space:

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

% Smooth out the electron density map
N = 20;
sigma = 3;
[x y]=meshgrid(round(-N/2):round(N/2), round(-N/2):round(N/2));
f=exp(-x.^2/(2*sigma^2)-y.^2/(2*sigma^2));
f=f./sum(f(:));
figure(4);
imagesc(conv2(density,f,'same'));
set(gca,'YDir','normal');
title('Electron Density');
xlabel('x (nm)');
ylabel('y (nm)');
```



The temperature map is created using a similar procudure. The electron velocities are put into bins over space to calculate the temperature at different points:

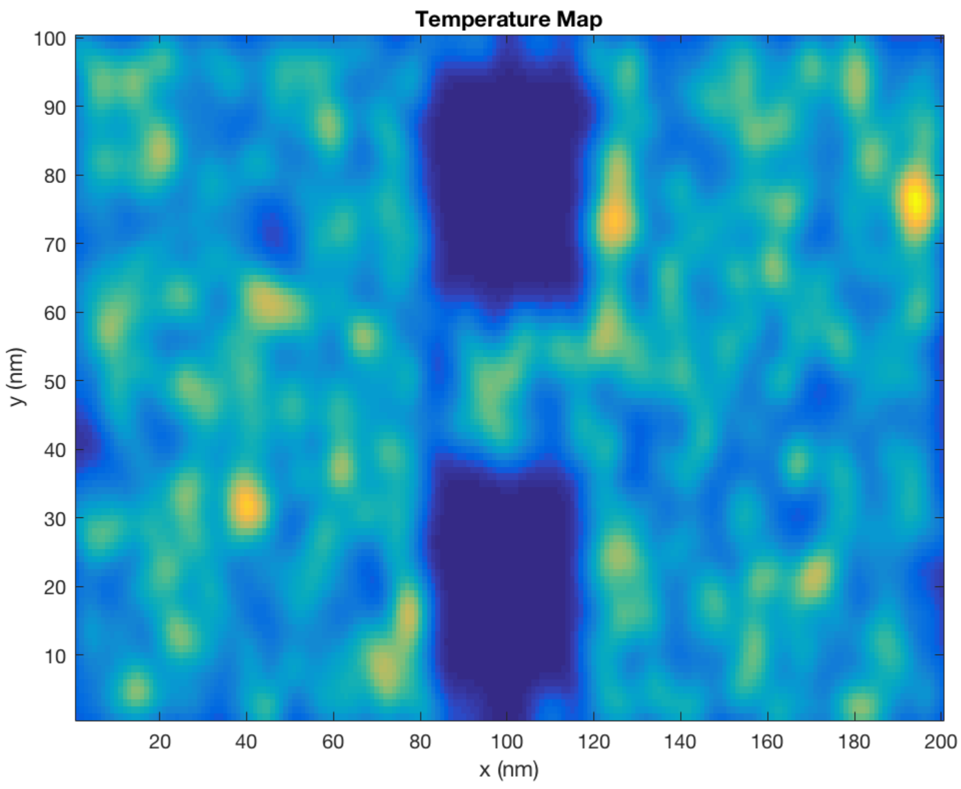
```
temp_sum_x = zeros(ceil(length/1e-9),ceil(height/1e-9));
temp_sum_y = zeros(ceil(length/1e-9),ceil(height/1e-9));
temp_num = zeros(ceil(length/1e-9),ceil(height/1e-9));

% Look at velocities of all the particles
for i=1:population_size
    % Find which "bin" it belongs in:
    x = floor(state(i,1)/1e-9);
    y = floor(state(i,2)/1e-9);
    if(x==0)
        x = 1;
    end
    if(y==0)
        y= 1;
    end

    % Add its velocity components to the cumulative count:
    temp_sum_y(x,y) = temp_sum_y(x,y) + state(i,3)^2;
    temp_sum_x(x,y) = temp_sum_x(x,y) + state(i,4)^2;
    temp_num(x,y) = temp_num(x,y) + 1;
end
```

Now, with the velocities added up, calculate the temperatures:

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

Like with the density map, perform some smoothing:

```
N = 20;
sigma = 3;
[x y]=meshgrid(round(-N/2):round(N/2), round(-N/2):round(N/2));
f=exp(-x.^2/(2*sigma^2)-y.^2/(2*sigma^2));
f=f./sum(f(:));
figure(5);
imagesc(conv2(temp,f,'same'));
set(gca,'YDir','normal');
title('Temperature Map');
xlabel('x (nm)');
ylabel('y (nm)');
```



The relationship between the temperature map and the electron density map is very noticable. However, some the of the electrons have considerably higher speeds, and this can be seen on the temperature map.

Published with MATLAB® R2016b