Assignment 1: Monte Carlo Modelling of Electron Transport

```
% Tyler Armstrong
% 101009324

% This program simulates N electrons in a 200nm by 100nm area. The
% electrons have thermally-distributed initial velocities, a
 probability to
% scatter, and can reflect from the y-boundaries (or optional barriers
% around the center). The thermal velocity assigned to the electrons
 is
% 1.32e5m/s.

% The mean free path of the electrons is about 26.4nm at 300K. The
 average
% temperature of all the electrons is around 300K, but varies
 significantly
% locally.

xlength = 200E-9; % Length of region in the x-direction, m
ylength = 100E-9; % Length of region in the y-direction, m
N = 200000; % Total number of electrons
px = 0; % x position of electrons, m
py = 0; % y position of electrons, m
vx = 0; % x-velocity of electrons, m/s
vy = 0; % y-velocity of electrons, m/s
m = 0.26*(9.11E-31); % Electron mass, kg
T = 300; % Temperature, K
dt = 1E-15; % Time step, s
reflecty = zeros(N,1); % For reflecting boundary
reflectx = zeros(N,1);
scatter = zeros(N,1); % For scattering
Pscat = 1 - exp(-dt/0.2E-12); % Scattering probability
Time = 1000; % Total number of time steps
maxTraj = 20; % Maximum number of trajectories to plot

boxes = 1; % Turn boxes on/off

px = -(xlength/2) + xlength*rand(N,1);
py = -(ylength/2) + ylength*rand(N,1);
vx = sqrt(1.380E-23*T/m)*randn(N,1);
vy = sqrt(1.380E-23*T/m)*randn(N,1);

if (boxes == 1) % Ensure that no electrons begin inside the barriers
    while (sum(px(abs(px) < xlength/20 & abs(py) > ylength/5) +
 py(abs(py) > ylength/5 & abs(px) < xlength/20)) ~= 0)
        px(abs(px) < xlength/20 & abs(py) > ylength/5) = -(xlength/2)
 + xlength*rand(1,1);
        py(abs(py) > ylength/5 & abs(px) < xlength/20) = -(xlength/2)
 + ylength*rand(1,1);
```

```
    end
end
colours = rand(min(maxTraj, N), 3);
xold = px(1:min(maxTraj,N));
yold = py(1:min(maxTraj,N));
wrapped = zeros(min(maxTraj,N),1);
clf;
figure(1);
for t = 1:Time

   xold = px(1:min(maxTraj,N));
   yold = py(1:min(maxTraj,N));
   % Update positions
   px = px + vx.*dt;
   py = py + vy.*dt;

   % Wrapping boundary
   wrapped = (abs(px(1:min(maxTraj,N))) > xlength/2);
   px = px + xlength*(px < -xlength/2);
   px = px - xlength*(px > xlength/2);

   % Reflecting boundary and boxes
   if (boxes == 0)
       reflecty = -1*(abs(py) > ylength/2);
       vy = vy.*(2*reflecty+1);
   end
   if (boxes == 1)
       reflecty = -1*((abs(py) > ylength/2) | (abs(py) > ylength/5 &
 abs(px) < xlength/20));
       reflectx = -1*(abs(px) < xlength/20 & abs(py) > ylength/5);
       vy = vy.*(2*reflecty+1);
       vx = vx.*(2*reflectx+1);
   end
   py = py - vy.*reflecty.*dt;
   px = px - vx.*reflectx.*dt;

   % Scattering
   scatter = rand(N,1) < Pscat;
   vx = vx + (sqrt(1.380E-23*T/m)*randn(N,1) - vx).*scatter;
   vy = vy + (sqrt(1.380E-23*T/m)*randn(N,1) - vy).*scatter;

   if (boxes == 1) % Draw the boxes
       plot([-xlength/20 xlength/20], [-ylength/5 -ylength/5], 'k');
       hold on;
       plot([-xlength/20 xlength/20], [ylength/5 ylength/5], 'k');
       hold on;
       plot([-xlength/20 -xlength/20], [ylength ylength/5], 'k');
       hold on;
       plot([-xlength/20 -xlength/20], [-ylength -ylength/5], 'k');
       hold on;
       plot([xlength/20 xlength/20], [ylength ylength/5], 'k');
       hold on;
       plot([xlength/20 xlength/20], [-ylength -ylength/5], 'k');
       hold on;
```

```
    end

    for i = 1:min(maxTraj,N)
     if (wrapped(i) == 0)
        plot([xold(i) px(i)], [yold(i) py(i)], 'color', colours(i,:));
     end
    end
    xlim([-xlength/2 xlength/2]);
    ylim([-ylength/2 ylength/2]);
    hold on;
    %pause(0.0001);
end
Tav = 0.5*m/(1.380E-23)*sum(vx.^2 + vy.^2)/N;
title('Electron trajectories')

figure(2);
xlim([-xlength/2 xlength/2]);
ylim([-ylength/2 ylength/2]);
phist = histogram2(px, py, 'binwidth', [1E-9
 1E-9], 'displaystyle', 'tile');
title('Electron Density Map');

figure(3);
Tmap = Tav*N./phist.Values;
for i = 1:xlength*1E9
    for j = 1:ylength*1E9
        Tmap(i,j) = 0.5*m/(1.380E-23)*sum(vx(px > (i-1)*1E-9-
xlength/2 & px < i*1E-9-xlength/2 & py > (j-1)*1E-9-ylength/2 & py <
 j*1E-9-ylength/2).^2 + vy(px > (i-1)*1E-9 -xlength/2 & px < i*1E-9 -
xlength/2 & py > (j-1)*1E-9 -ylength/2 & py < j*1E-9 -ylength/2).^2)./
phist.Values(i,j);
    end
end
X = linspace(-xlength/2, xlength/2, xlength*1E9);
Y = linspace(-ylength/2, ylength/2, ylength*1E9);
surfc(Y, X, Tmap);
title('Temperature map');
zlabel('Temperature (K)');
```

Published with MATLAB® R2018b