Assignment 1

Table of Contents

February 4, 2018

Question 1

In this simulation, electrons are modeled as they may exist in an N-type semiconductor. A Monte-Carlo model is used to determine the behaviour of the electrons. The effective mass of the electrons is $m_n = 0.26m_0$, where m_0 is the rest mass of an electron. Assuming the ambient temperature is 300K, the thermal velocity is $132.24km/s$. Given a mean free time of $0.2ps$, the mean free path could be expected to be around $26.449\mu m$, where

$$meanfreepath = \tau_{mn} * v_{th}$$

This code, as found below, models 10,000 electrons (as particles) in a region of $200nm$ by $100nm$. In this section, each particle has a fixed velocity based on v_th ad assigned a random direction. The top an bottom boundaries act as perfect reflectors, or specular boundaries, while the left and right boundaries are periodic, maintaining the same number of electrons throughout the simulation. Figure 1 shows the result of the electrons in place within the region, though the motion of an individual electron is difficult to track due to the sheer number of particles in the region. Figure 2 follows the trajectories of several electrons over the course of the simulation, with each electron of the chosen few being assigned a different color.

In this portion of the experiment, the temperature of the semiconductor will remain constant. This is because the electrons are not subject to scattering and are not undergoing any change in the magnitude of their velocities, as we will see in the ollowing sections. Figure 3 shows the temperature of the semiconductor remaining constant, as expected, as calculated from the unchanging electron velocities.

Question 2

In this section of the simulation, scattering is added to the electron movement (i.e.collisions with the background). A calculation of the mean free path based on scattering, rather than a rough calculation based on given parameters. In this simulation, the scattering is based on the formula:

$$P_{scat} = 1 - e^{(-dt/\tau_{mn})}$$

This is then compared to a random number generator in MATLab, to simulate the periodicity with which the electrons would scatter in real life, expected to be approximately 5% of the time. In this simulation, when an electron scatters, the electron "re-thermalizes", meaning it is given a new random velocity. The overall distribution of initial velocities for this simulatio nis expected to reflect a Maxwell-Boltzmann distribution (i.e. normal distribution), which is affected by including the standard deviation factor of the Maxwell-Boltzmann distribution in the velocity assignment command. Figure 4 shows the distribution of the initial velocities of the electrons. The upper left figure shows the x-axis velocity, the upper right figure shows the y-axis velocity, and the lower figure shows the distribution of the resultant speeds. Figure 5 shows a sample of the electron trajectories, with the individual color assignments. In this simulation, since there is scattering and re-thermalizing, the overall temperature of the semiconductor is not constant, but rather fluctuates with each step in the time loop. Figure 6 shows the instantaneous temperature readings over time, with the most recent reading displayed in the title. The average temperature of the material remains close to the initial value of the material, $T = 300K$.

The mean free path and the mean free time are calculated here as well. In an average run, the mean free path will be in the order of magnitude of approximately $2nm$, with a mean free time of approximately $0.13ps$. This is not a huge discrepancy between the assumed values at the beginning, though in the case of the mean free time it is one order of magnitude removed from the assumed value. It is worth noting that exact figures are not included here, as the simulation is based on random elements that will change the final outcome for this parameter with every simulation run.

Question 3

In this final section, we add some insulators to simulate electron motion through a barrier. This "bottle-neck" increases the electron activity, in terms of velocity changes. In this simulation, all boundaries are initialized as specular boundaries, with the exception of the right and left boundaries of the full region, which remain periodic. Electrons are re-thermalized upon scattering. Figure 7 shows the electron activity around the boundaries. Note that no electrons are initialized inside the boxes. Figure 8 shows the electron density map based on the positions of the electrons at the end of the simulation. As expected, it looks completely random, and no electrons have made it inside the dielectric "boxes". Figure 9 is similar in that it shows a temperature density map of the region. This figure takes the magnitude of the final velocies of the electrons and uses this value to calculate the temperature of each region of the grid. The grid used in this simulation is squares of $5nm$ by $5nm$. This yields a density grid of 40 by 20 boxes. Each box is the average temperature of the electrons within the box. The code for every section of the simulation is included below.

Question 1 Code

```
close all
clear all

L=200e-9;
W=100e-9;
n=10000; %change
nsteps =1000; %change
num = n;

ang=randn(1,n)*2*pi;

m0=9.109382e-31; %electron mass
mn=0.26*m0;
T=300; %Kelvin
k=physconst('Boltzman');

vth = sqrt(k*T/mn)
```

```
tau=0.2e-9; %seconds
mfp = tau*vth % meters

x=rand(1,n)*L;
y=rand(1,n)*W;
xp=x;
yp=y;

vx=vth*ones(1,n).*cos(ang);
vy=vth*ones(1,n).*sin(ang);

dt=(L/vth)/100;

col=hsv(10); %vector of colours for particle trajectories

f1 = figure;
f2 = figure;
set(0, 'CurrentFigure', f2)
  for p = 1:10
        plot([x(p); xp(p)],[y(p); yp(p)],'color',col(p,:));  hold on
    end
    xlim([0 L])
    ylim([0 W])

 f3 = figure;
for i=1:nsteps

    xp=x;
    yp=y;

    dx=vx*dt;
    dy=vy*dt;

    x=x+dx;
    y=y+dy;

    %periodic boundaries for walls
     for a=1:n
        if (x(a)>L)
            x(a)=x(a)-L;
            xp(a)=xp(a)-L;
        elseif x(a)<0
            x(a) = x(a)+L;
            xp(a) = xp(a)+L;
        end
    %specular boundaries for ceiling and floor
        if y(a)>=W
            vy(a) = -vy(a);
         elseif y(a)<=0
            vy(a) = -vy(a);
        end
     end
    velx = mean(abs(vx));
    vely = mean(abs(vy));
    v_inst=sqrt(velx*velx+vely*vely);

    Temp= v_inst*v_inst*mn/k ;

    set(0, 'CurrentFigure', f1)
    %plot trajectories
    plot(x,y,'*');
    xlim([0 L])
    ylim([0 W])
  title(sprintf('Monte Carlo Electron Simulation - Number of Electrons
 = %d', num))


    set(0, 'CurrentFigure', f2)
        %plot trajectories
    for p = 1:10
        plot([x(p); xp(p)],[y(p); yp(p)],'color',col(p,:));  hold on
    end
    title('Electron Simulation with Trajectories')
    xlim([0 L])
    ylim([0 W])


    set(0, 'CurrentFigure', f3)
    plot(i,Temp, 'o')
    hold on
    title(sprintf('Semiconductor Temperature = %s', Temp))

    pause(0.01);

end


vth =

   1.3224e+05


mfp =

   2.6449e-05
```

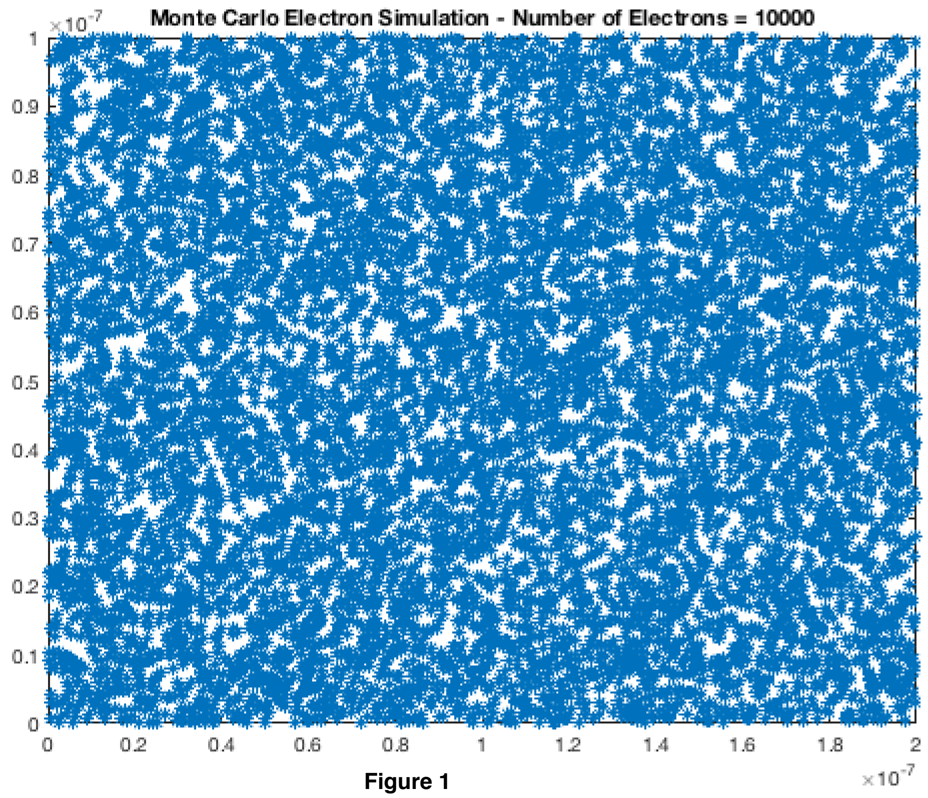

Figure 1

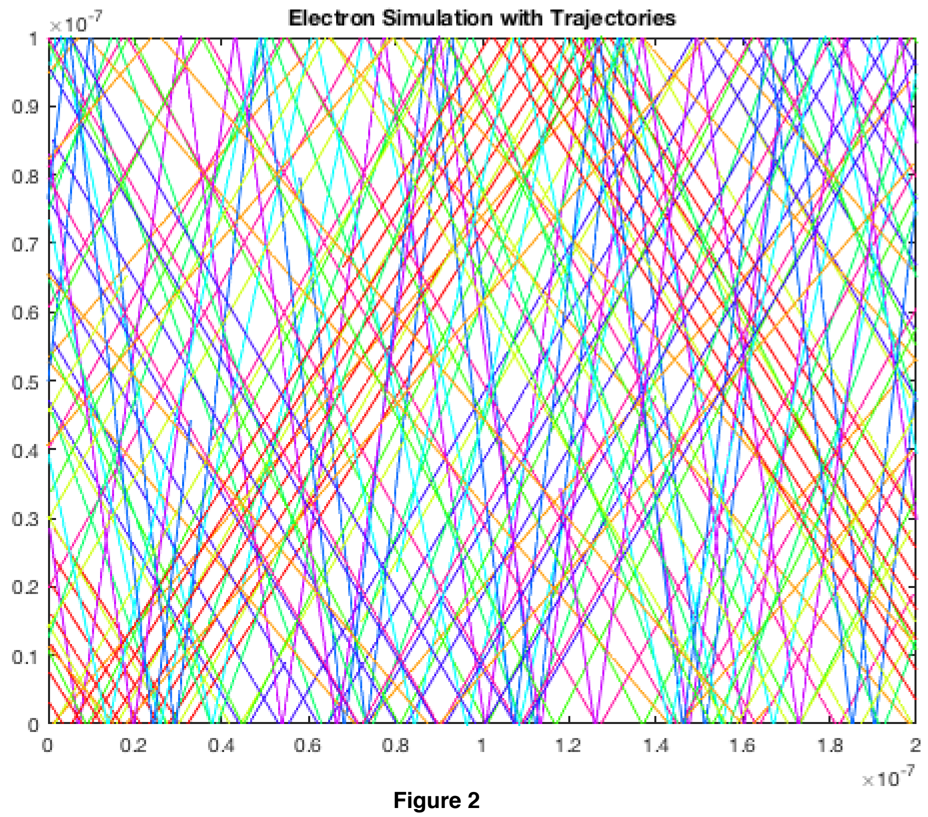

Figure 2

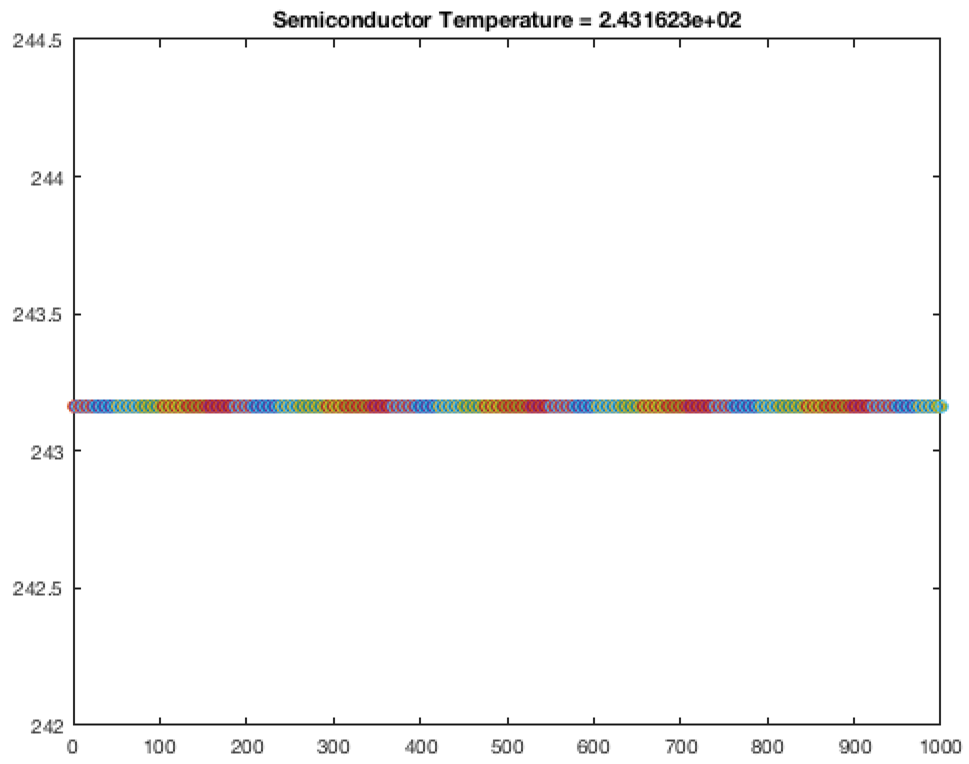

Figure 3

Question 2 Code

```
L=200e-9;
W=100e-9;
n=10000; %change
nsteps =1000; %change

tau_mn=0.2e-12 %seconds

m0=9.109382e-31; %electron mass
mn=0.26*m0;
T=300; %Kelvin
k=physconst('Boltzman');

vth = sqrt(k*T/mn);
mfp = tau_mn*vth % meters

%inititalize particle locations
x=rand(1,n)*L;
y=rand(1,n)*W;

%initialize previous location as first location just to get the plot
%started
xp = x;
yp = y;
```

```
%initialize random velocities
vx=vth*randn(1,n)/sqrt(2);
vy=vth*randn(1,n)/sqrt(2);
speed=sqrt(vx.*vx +vy.*vy);

f1 = figure;
f2 = figure;
f3=figure;

set(0, 'CurrentFigure', f1)
subplot(2,2,1);
histogram (vx);
title ('Distribution-Vx')
subplot(2,2,2);
histogram (vy);
title ('Distribution-vy')
subplot(2,2,[3,4]);
histogram (speed);
title ('Distribution-speed');

dt=(L/vth)/100;
av_temp = zeros(1,nsteps);

%vector of colours for particle trajectories
col=hsv(10);
set(0, 'CurrentFigure', f2)
  for p = 1:10
        plot([x(p); xp(p)],[y(p); yp(p)],'color',col(p,:));  hold on
    end
    xlim([0 L])
    ylim([0 W])

%main timeloop
for aa=1:nsteps
    xp=x;
    yp=y;

    %scattering
    pscat=1-exp(-dt/tau_mn);
    scatCount= 0;

    for bb=1:n
        if (pscat > rand())
            vx(bb)=vth*randn()/sqrt(2);
            vy(bb)=vth*randn()/sqrt(2);
            scatCount = scatCount+1;
        end
    end

    dx=vx*dt;
    dy=vy*dt;

    %increment every particle over dt
    x=x+dx;
    y=y+dy;

    %xpath calc before boundary adjustment
    xpath=abs(x-xp);

    %travelling restrictions (WALL)
     for a=1:n
        %periodic boundaries at x=0 and x=L
         if (xp(a)< L && x(a)>=L)
            x(a)=x(a)-L;
            xp(a)=xp(a)-L;
         elseif (xp(a)< 0 && x(a)<0)
            x(a) = x(a)+L;
            xp(a)=xp(a)+L;
         end

        %specular boundaries at y=0 and y=W
        if (y(a)>=W || y(a)<=0)
            vy(a) = -vy(a);
        elseif y(a)<=0
          vy(a) = -vy(a);
        end
     end %end travelling restrictions loop


     %ypath calc after boundary adjustment
    ypath=abs(y-yp);

    %calculate path
    path = sqrt(xpath.*xpath + ypath.*ypath);



% plot(x,y,'o');hold on
  set(0, 'CurrentFigure', f2)
    %plot trajectories
    for p = 1:10
        plot([x(p); xp(p)],[y(p); yp(p)],'color',col(p,:));  hold on
    end
    xlim([0 L])
    ylim([0 W])
    title ('Electron Collisions with Mean Free Path')
    pause(0.01);


    velx = mean(vx.^2);
    vely = mean(vy.^2);
    v_inst=sqrt(velx+vely);

    Temp= v_inst*v_inst*mn/k ;
   set(0, 'CurrentFigure', f3)
    plot(aa,Temp, 'o')
    hold on
    title(sprintf('Semiconductor Temperature = %s', Temp))

    av_temp(aa) = Temp;
end


AverageTemperature = mean(av_temp)
meanfreepath = mean(path)
meanfreetime = meanfreepath/(mean(v_inst))
```

```
tau_mn =

   2.0000e-13


mfp =

   2.6449e-08


AverageTemperature =

  299.9654


meanfreepath =

   1.7826e-09


meanfreetime =

   1.3409e-14
```

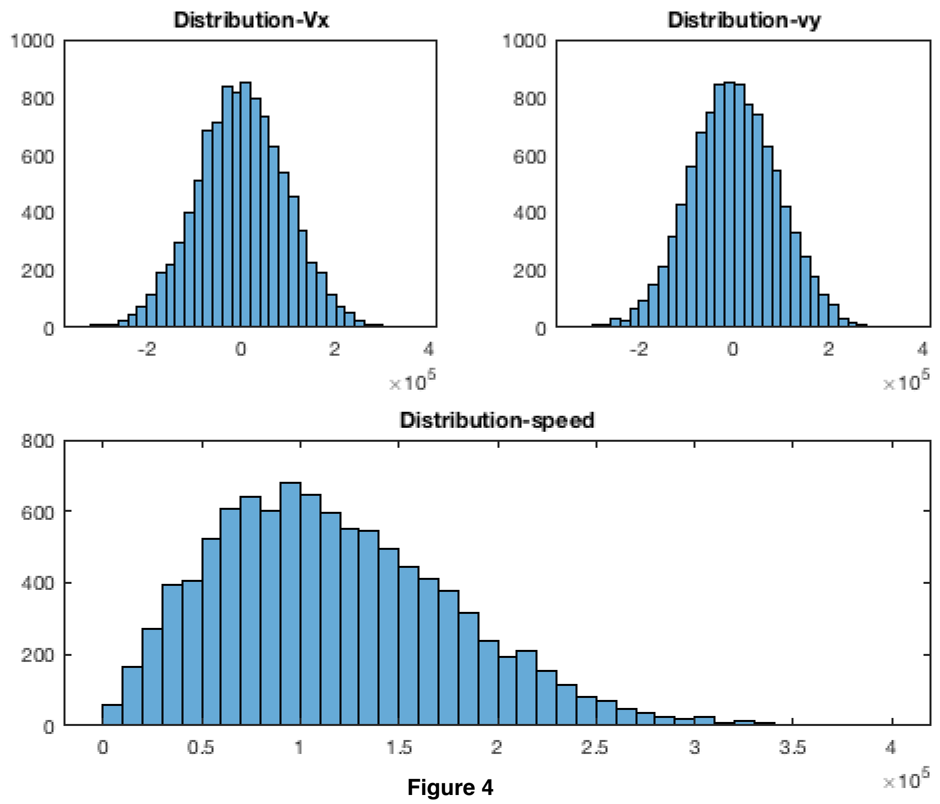

Figure 4

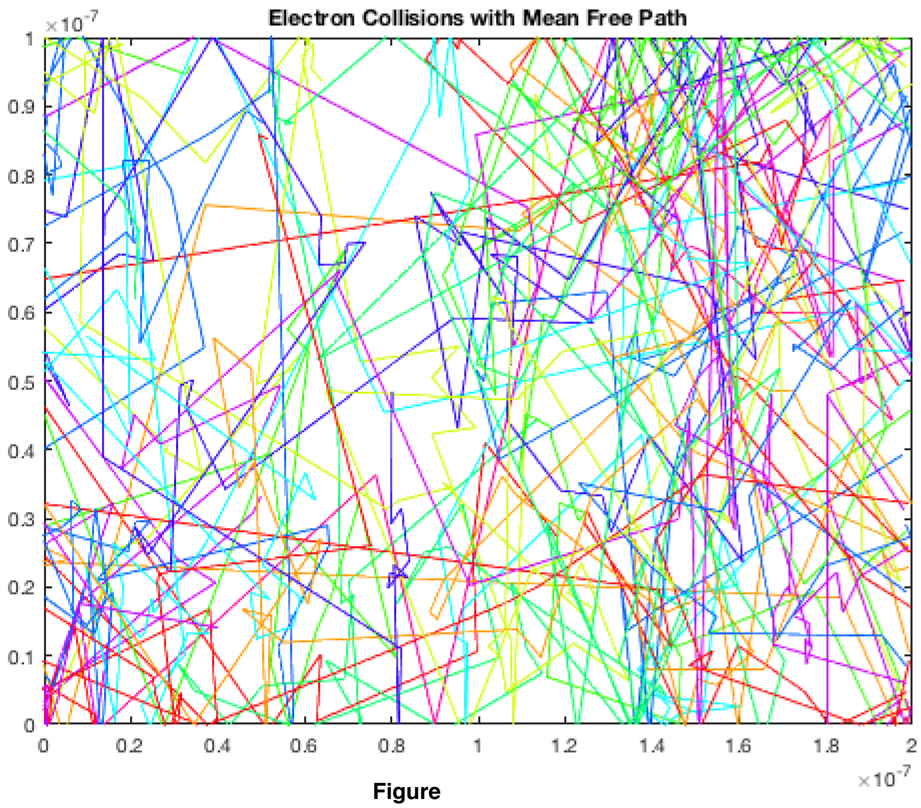

Figure 5

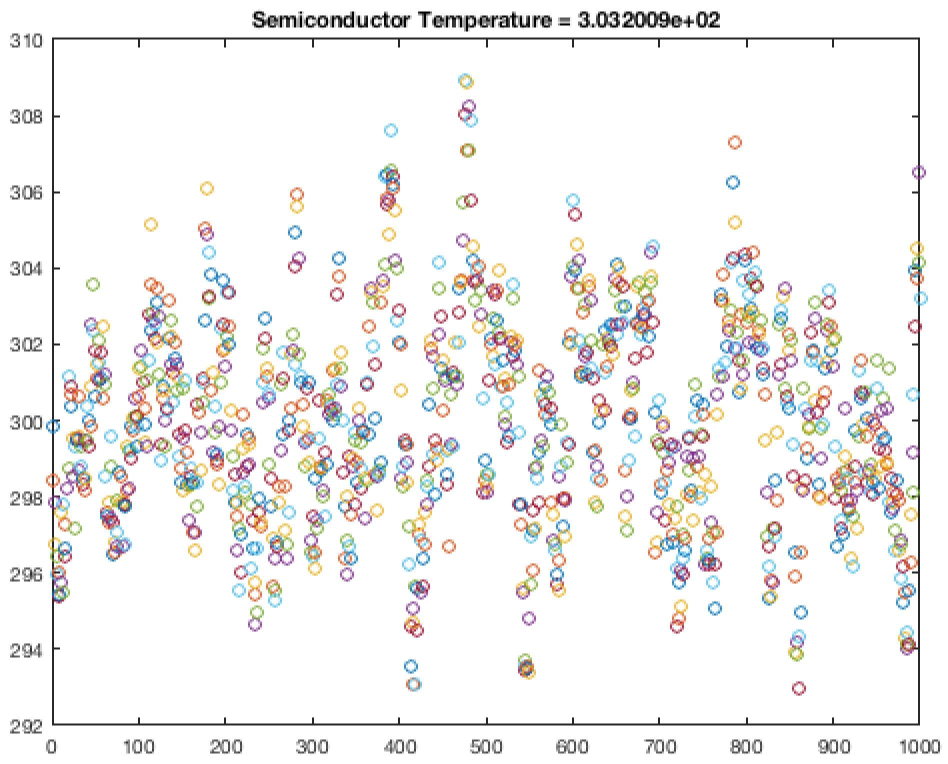

Figure 6

Question 3 Code

```
L=200e-9;
W=100e-9;
n=10000; %change
nsteps =1000; %change

ang=randn(1,n)*2*pi;

m0=9.109382e-31; %electron mass
mn=0.26*m0;
T=300; %Kelvin
k=physconst('Boltzman');
d=1e-18;
tau_mn=0.2e-12; %seconds


vth = sqrt(k*T/mn);

%inititalize particle locations
x=rand(1,n)*L;
y=rand(1,n)*W;

% create a bunch of electrons not in the boxes
% box 1  190e-9<x<210e-9 60e-9<y<100e-9
% box 2  190e-9<x<210e-9 0<y<40e-9
Cxlow = 80e-9;
Cxhigh= 120e-9;
Cylow =40e-9;
Cyhigh=60e-9;
Ibox = (y>Cyhigh | y<Cylow) & x<Cxhigh & x>Cxlow;

countrestarts =0;
% no starting in boxes
for a = 1:n
while (x(a)<Cxhigh && x(a)>Cxlow && (y(a)>Cyhigh || y(a)<Cylow))
    x(a) = rand()*L;
    y(a) = rand()*W;
    countrestarts = countrestarts+1;
end
end


%initialize previous location as first location just to get the plot
%started
xp = x;
yp = y;
%initialize random velocities
vx=vth*rand(1,n);
vy=vth*rand(1,n);

dt=(L/vth)/100;

col=hsv(10); %vector of colours for particle trajectories
figure(7)
  for p = 1:10
        plot([x(p); xp(p)],[y(p); yp(p)],'color',col(p,:));  hold on
  end
    xlim([0 L])
    ylim([0 W])

%display boxes
line([Cxlow,Cxlow,Cxhigh,Cxhigh], [0,Cylow,Cylow,0], 'color', 'k');
line([Cxlow,Cxlow,Cxhigh,Cxhigh], [W,Cyhigh,Cyhigh,W], 'color', 'k');

%main timeloop
for i=1:nsteps

    xp=x;
    yp=y;

    dx=vx*dt;
    dy=vy*dt;

    %increment every particle over dt
    x=x+dx;
    y=y+dy;

    %xpath calc before boundary adjustment
    xpath=abs(x-xp);

    %travelling restrictions (WALL)
     for a=1:n
       %no travelling through boxes
        if ( xp(a)<=Cxlow && x(a)>=Cxlow &&(y(a)>=Cyhigh ||
y(a)<=Cylow))
            x(a)=Cxlow ;
            vx(a)=-vx(a);
        elseif (xp(a)>=Cxhigh && x(a)<=Cxhigh&&(y(a)>=Cyhigh ||
y(a)<=Cylow))
            x(a)=Cxhigh;
            vx(a)=-vx(a);
        elseif (yp(a)<=Cyhigh && y(a)>=Cyhigh&&(x(a)>=Cxlow &&
 x(a)<=Cxhigh))
            y(a) = Cyhigh;
            vy(a) = -vy(a);
        elseif (yp(a)>=Cylow && y(a)<=Cylow&&(x(a)>=Cxlow &&
 x(a)<=Cxhigh))
              y(a) = Cylow;
            vy(a) = -vy(a);
        end

        %periodic boundaries at x=0 and x=L
         if (xp(a)< L && x(a)>=L)
            x(a)=x(a)-L;
            xp(a)=xp(a)-L;
         elseif (xp(a)< 0 && x(a)<0)
            x(a) = x(a)+L;
            xp(a)=xp(a)+L;
         end

        %specular boundaries at y=0 and y=W
        if (y(a)>=W || y(a)<=0)
            vy(a) = -vy(a);
        elseif y(a)<=0
          vy(a) = -vy(a);
        end
     end %end travelling restrictions loop

    %scattering
    pscat=1-exp(-dt/tau_mn);
    scatCount= 0;

    for bb=1:n
        if (pscat > rand())
            vx(bb)=vth*randn()/sqrt(2);
            vy(bb)=vth*randn()/sqrt(2);
            scatCount = scatCount+1;
        end
    end

     %ypath calc after boundary adjustment
    ypath=abs(y-yp);
    %calculate path - not sure if this is right
    path = sqrt(xpath.*xpath + ypath.*ypath);

%     plot(x,y,'o');hold on
  figure (7)
    %plot trajectories
    for p = 1:10
        plot([x(p); xp(p)],[y(p); yp(p)],'color',col(p,:));  hold on
    end
    xlim([0 L])
    ylim([0 W])
    title ('Monte Carlo Simulation of Electron Trajectories with
 Bottleneck')
    pause(0.01);


end

delta = 5e-9;
counta=0;
countelectrons=0;
Nmap = zeros(L/delta,W/delta);
Vmap = zeros(L/delta,W/delta);
Tmap = zeros(L/delta,W/delta);

%populate density maps
for  aa = delta:delta:L
    counta=counta+1;
    countb=0;
    for bb=delta:delta:W
        countb=countb+1;
        for cc=1:n
            %Populate Electron Density Map
            if (x(cc)<(counta*delta) & x(cc)>=((counta-1)*delta) &
 y(cc)<(countb*delta) & y(cc)>=((countb-1)*delta))
                Nmap(counta,countb) = Nmap(counta,countb)+1;
                Vmap(counta,countb) =
 Vmap(counta,countb)+sqrt(vx(cc)*vx(cc)+vy(cc)*vy(cc));
                map(counta,countb)=Vmap(counta,countb)/
Nmap(counta,countb);
                Tmap(counta,countb) =
 map(counta,countb)*map(counta,countb)*mn/k;
                countelectrons = countelectrons +1;
            elseif(x(cc)== L)
                Nmap(counta,countb) = Nmap(counta,countb)+1;
                Vmap(counta,countb) =
 Vmap(counta,countb)+sqrt(vx(cc)*vx(cc)+vy(cc)*vy(cc));
                map(counta,countb)=Vmap(counta,countb)/
Nmap(counta,countb);
               Tmap(counta,countb) =
 map(counta,countb)*map(counta,countb)*mn/k;
                countelectrons = countelectrons +1;
             elseif(y(cc)==W)
```

```
                Nmap(counta,countb) = Nmap(counta,countb)+1;
                Vmap(counta,countb) =
 Vmap(counta,countb)+sqrt(vx(cc)*vx(cc)+vy(cc)*vy(cc));
                map(counta,countb)=Vmap(counta,countb)/
Nmap(counta,countb);
               Tmap(counta,countb) =
 map(counta,countb)*map(counta,countb)*mn/k;
                countelectrons = countelectrons +1;
            elseif(x(cc)== L & y(cc)==W)
                Nmap(counta,countb) = Nmap(counta,countb)+1;
                Vmap(counta,countb) =
 Vmap(counta,countb)+sqrt(vx(cc)*vx(cc)+vy(cc)*vy(cc));
                map(counta,countb)=Vmap(counta,countb)/
Nmap(counta,countb);
               Tmap(counta,countb) =
 map(counta,countb)*map(counta,countb)*mn/k;
                countelectrons = countelectrons +1;
            end
        end
    end
end

for  dd = 1:L/delta
    for ee=1:W/delta
        if Tmap(dd,ee)== 0
            Tmap(dd,ee) = 300;
        end
    end
end

 figure(8)
surf(Nmap)
colormap('parula')
colorbar
shading interp;

figure (9)
surf(Tmap)
h=flipud(hsv);
colormap(h)
caxis([200 800])
colorbar
shading interp;
```

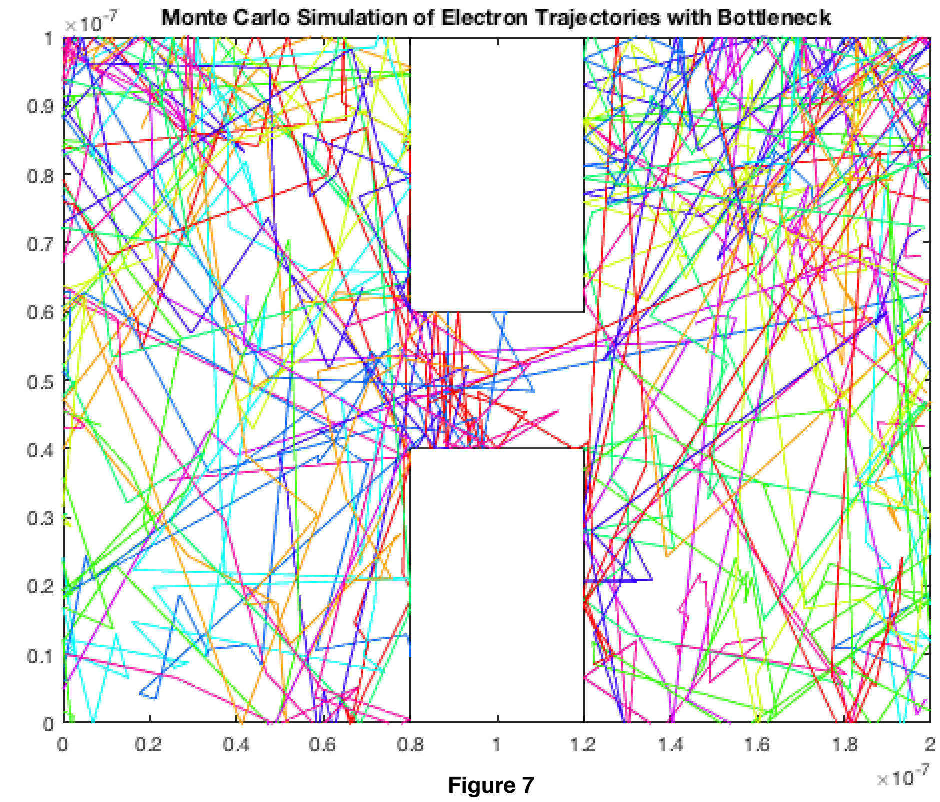

Figure 7

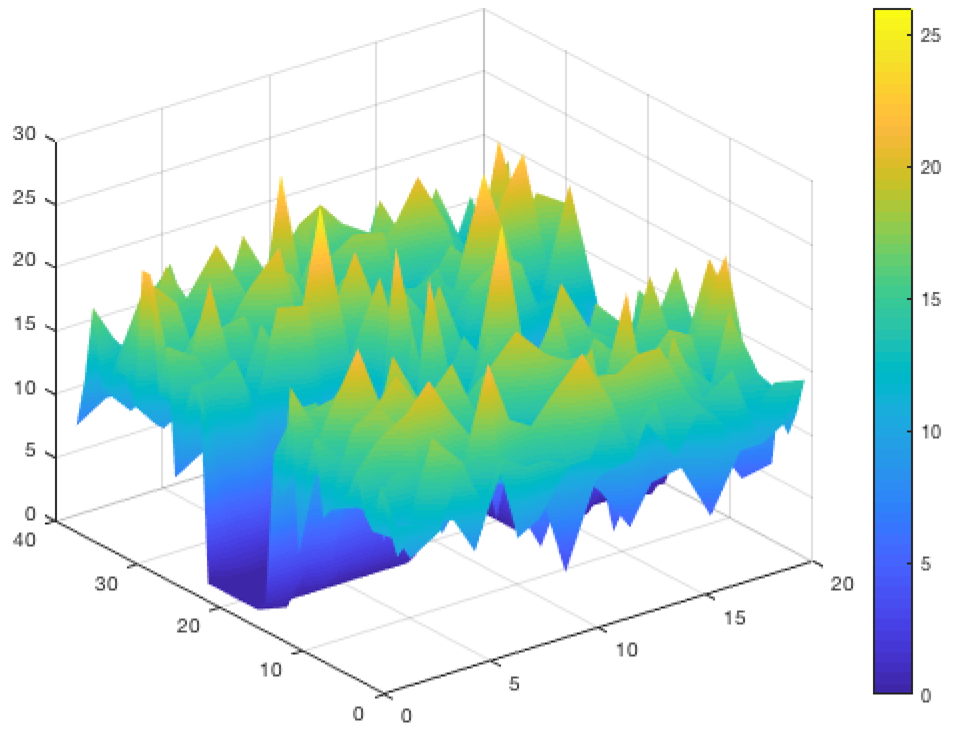

Figure 8

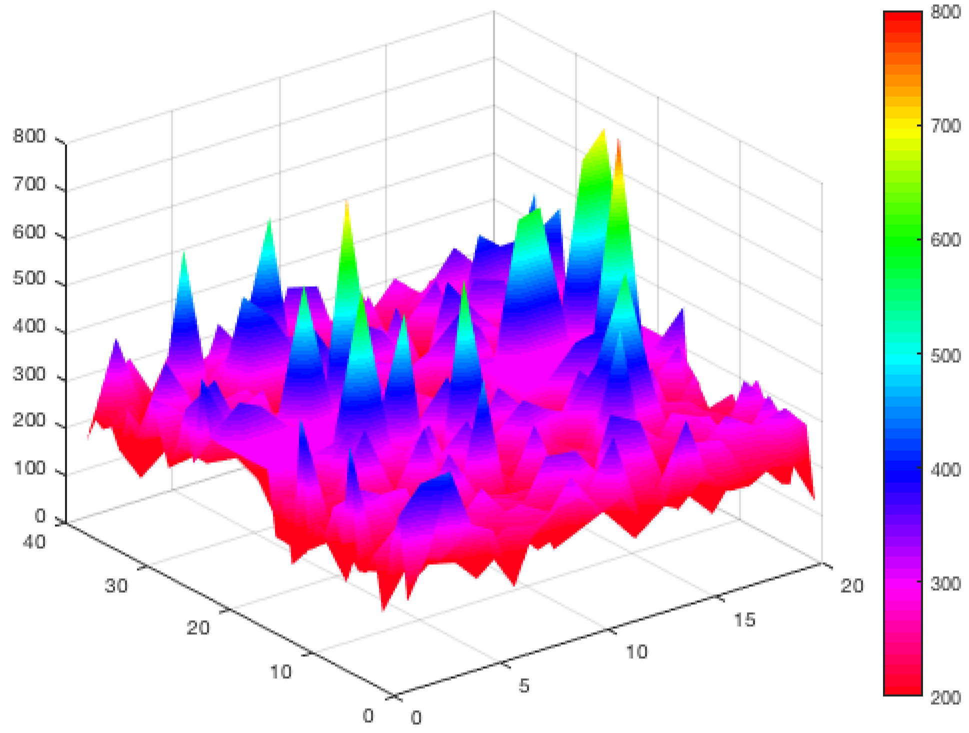

Figure 9

Published with MATLAB® R2017b