Umeh Ifeanyi

Table of Contents

101045786, Part 1

```
% Translation is only considered in the plane, therefore 2 degrees of
% freedom. The following equation holds:
%
%
% Where m is 0.26*9.11*10^-31. Effective Mass of the electrons m
%
%
```

Solving for vth gives:

```
vth = (\frac{2*k*T}{m})^\frac{1}{2}
```

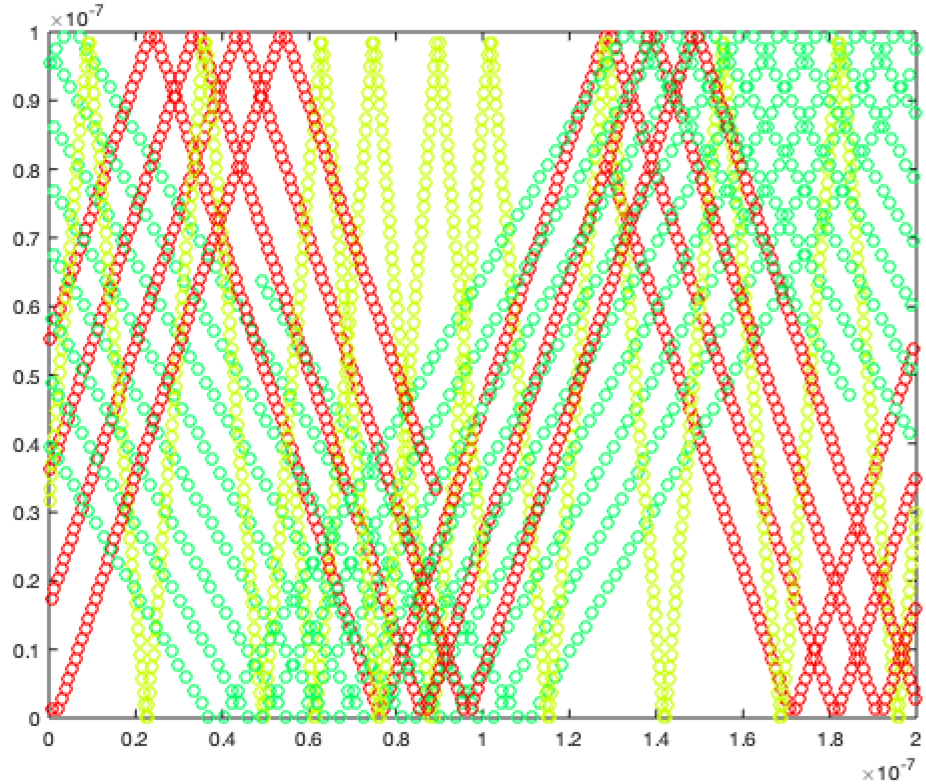
Mean free path is given by:

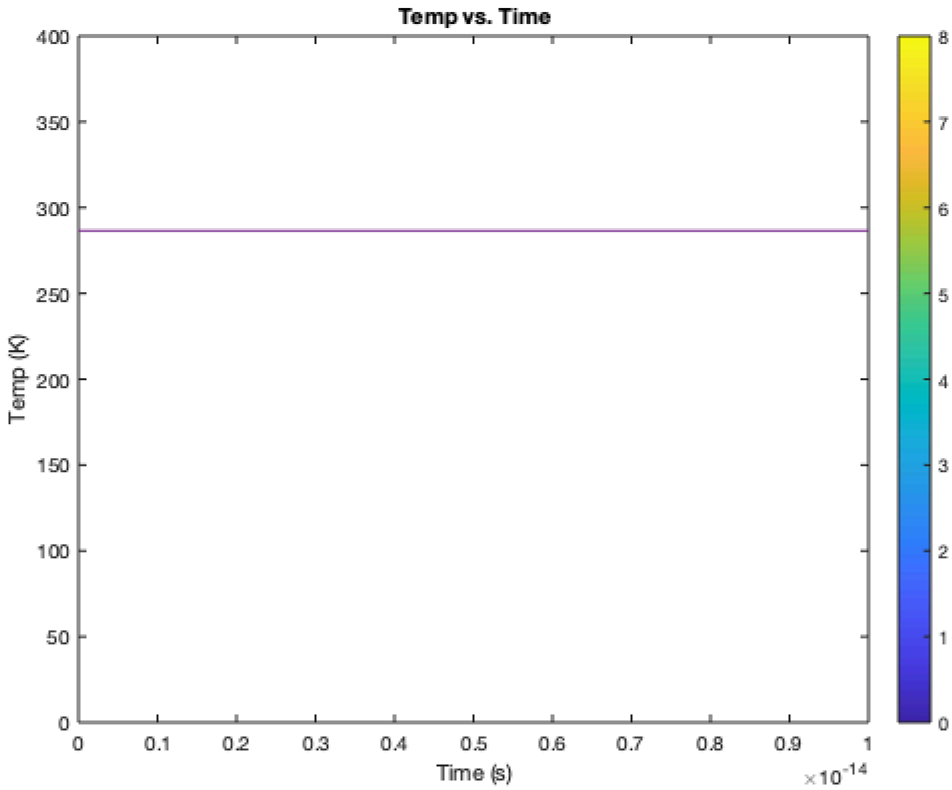
tau = vth*0.2ps

```
clear
clf

%%CONSTANT DECLARATIONS

me=0.26*(9.11*10^-31); %% eff mass of Electrons

dt = 10*(10^-15); %sim for 1000 dt's
kb = 1.3806*10^-23;
vth = sqrt((2*300*kb)/(me)); %thermal velocity, 2D
N = 1000; %number of particles
steps = 1000; %number of steps
p1 = 1; %lower bound
p2 = 3; % upper bound
```

```
vnet = zeros(1,N);
x = zeros(1,N);
y = zeros(1,N);
vx = zeros(1,N);
vy = zeros(1,N);
T = zeros(1,N);
for i = 1:N
    %initial conditions


    x(i) = rand*200*(10^-9);

    y(i) = rand*100*(10^-9);

    vx(i) = (vth)*cos(2*pi*rand);
    vy(i) = (vth)*sin(2*pi*rand);

    vnet(i) = sqrt(vx(i)^2 + vy(i)^2);
end

for j = 1:steps     %amount of timesteps
    x(1:N) = x(1:N) + (vx(1:N).*dt);
    y(1:N) = y(1:N) + (vy(1:N).*dt);

    for k = 1:N   %boundary conditions
        if(y(k)<=0 || y(k)>=100*10^-9)
            vy(k) = -vy(k);
        end
        if(x(k)<=0)
            x(k) = x(k) + 200*10^-9;
        end
        if(x(k)>=200*10^-9)
            x(k) = x(k) - 200*10^-9;
        end

    end

    for q=p1:p2
        colorVec = hsv(5);
        plot(x(q),y(q),'o','color', colorVec(q,:));
    end

    T(j) = (me/(2*kb))*( mean(abs(vx).^2)+ mean(abs(vy).^2) );

    axis([0,200*10^-9,0,100*10^-9]);
    pause(0.00001);
    hold on;

end


figure(2)
h=linspace(0,1000*dt,1000);
```

```
plot(h,T)
title('Temp vs. Time');
xlabel('Time (s)');
ylabel('Temp (K)');
axis([0,10^-14,0,400]);
```





Part 2

COLLISIONS WITH MEAN FREE PATH

```
% Temperature varies for each scattering of all particles. Average for
 all
% time steps is rough 300K. More with more particles averge speeds
 tends to
% vth. Most populous bin in histogram approaches vth. Mean free path
 and
% mean time between collisions is calculated based on steps/count of
% scattering. With more timesteps, MFP and MTBC converges to analytic
% result.
```

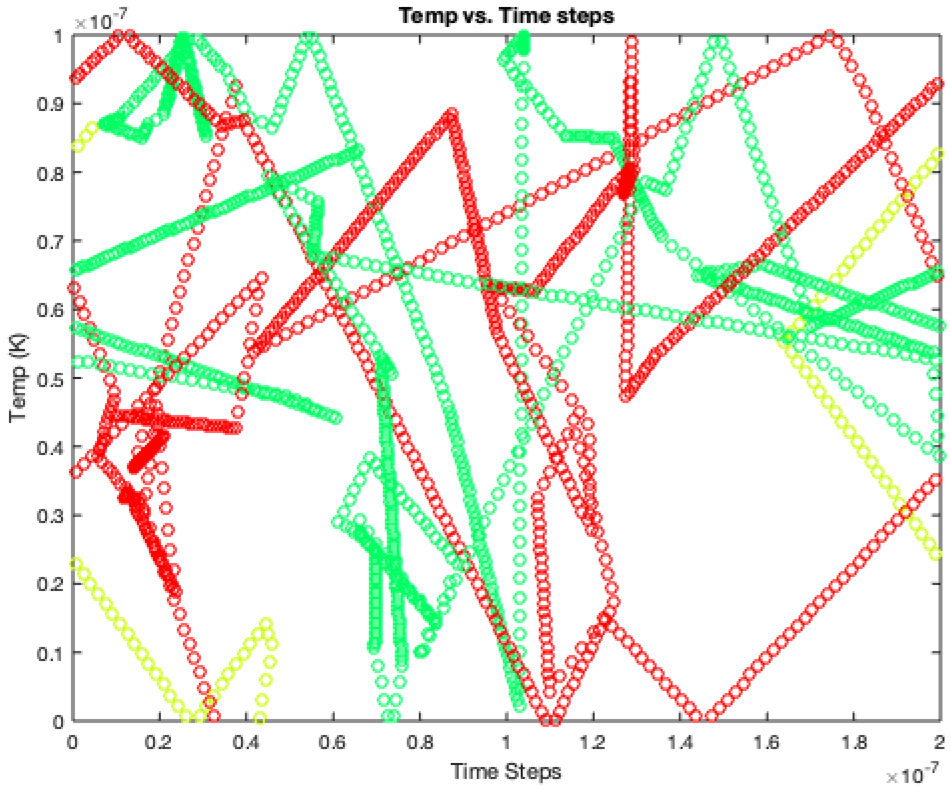
```
clear
clf
```

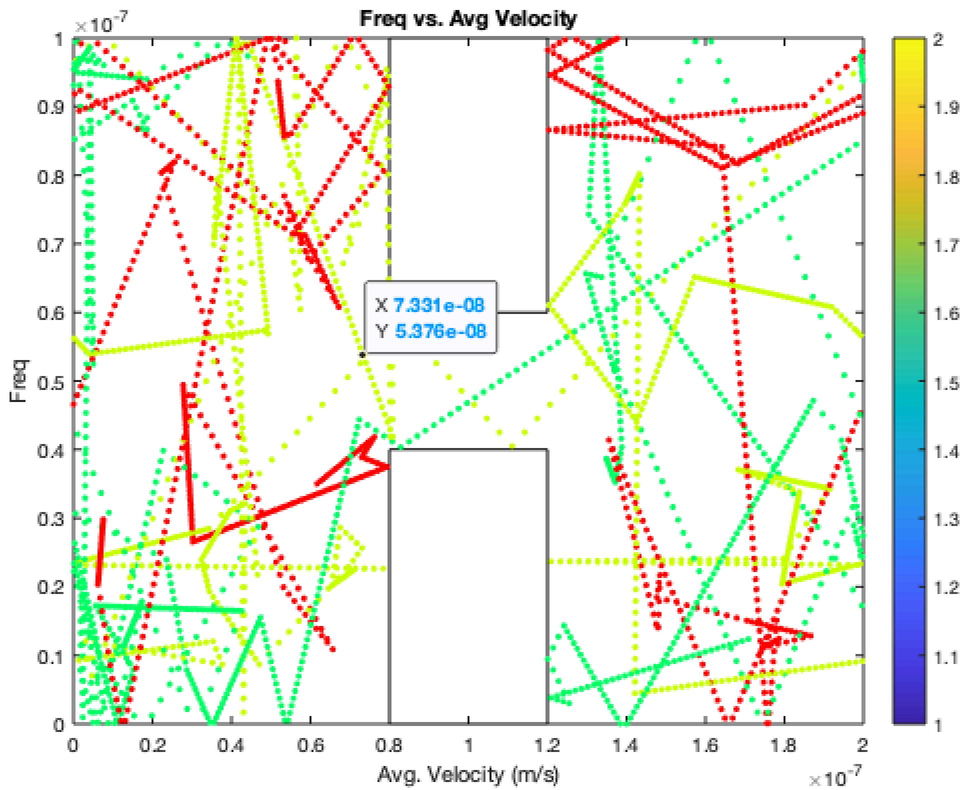
CONSTANT DECLARATIONS

```
me=0.26*(9.11*10^-31); %effective mass
dt = 10*(10^-15); %sim for 1000 dt's
pscat = 1-exp(-(dt)/(0.2*10^(-12)));
kb = 1.3806*10^-23;
vth = sqrt((2*300*kb)/(me)); %thermal velocity, 2D
N = 1000; %no of particles
steps = 1000; %no of steps
p1 = 1; %partition lower bound
p2 = 3; %partition upper bound
countscat=0;

pcomp = zeros(1,steps);
vnet = zeros(1,N);
x = zeros(1,N);
y = zeros(1,N);
vx = zeros(1,N);
vy = zeros(1,N);
T = zeros(1,N);
for i = 1:N
    x(i) = rand*200*(10^-9);
    y(i) = rand*100*(10^-9);                %initial conditions
    vx(i) = (vth).*randn(1,1)*(1/sqrt(2));
    vy(i) = (vth).*randn(1,1)*(1/sqrt(2));
```

```
    vnet(i) = sqrt(vx(i)^2 + vy(i)^2);
end

for s = 1:steps
    pcomp(s) = rand;
end

for j = 1:steps     %amount of timesteps
    x(1:N) = x(1:N) + (vx(1:N).*dt);
    y(1:N) = y(1:N) + (vy(1:N).*dt);

    if(pscat > pcomp(j))
        countscat = countscat+1;
        for u = 1:N
            vx(u) = (vth).*randn(1,1)*(1/sqrt(2));
            vy(u) = (vth).*randn(1,1)*(1/sqrt(2));

        end
    end

    for k = 1:N   %boundary conditions
        if(y(k)<=0 || y(k)>=100*10^-9)
            vy(k) = -vy(k);
        end
        if(x(k)<=0)
            x(k) = x(k) + 200*10^-9;
        end
        if(x(k)>=200*10^-9)
            x(k) = x(k) - 200*10^-9;
        end

    end

    for q=p1:p2
        colorVec = hsv(5);
        plot(x(q),y(q),'o','color', colorVec(q,:));

    end

    T(j) = (me/(2*kb))*( mean(abs(vx).^2)+ mean(abs(vy).^2) );

    axis([0,200*10^-9,0,100*10^-9]);
    pause(0.00001);
    hold on;

end

figure(2);
h=linspace(0,1000*dt,1000);
plot(T);
% axis([0,1000,280,320]);
title('Temp vs. Time steps');
xlabel('Time Steps');
ylabel('Temp (K)');
```

```
figure(3);
hist(vnet,20);
title('Freq vs. Avg Velocity');
xlabel('Avg. Velocity (m/s)');
ylabel('Freq');
```





Part 2 Questions

```
mfp = (steps/countscat)*dt*mean(vnet); %%Calculation for mean free
 path
meantime = dt*(steps/countscat);  %Mean time between collisions

sprintf('avg. velocity is %0.5e m/s' ,mean(vnet))
sprintf('mean free path is %0.5e m and mean time between collisions is
 %0.5e s' ,mfp,meantime)
```

```
ans =

    'avg. velocity is 1.70979e+05 m/s'


ans =

    'mean free path is 3.97626e-08 m and mean time between collisions
 is 2.32558e-13 s'
```

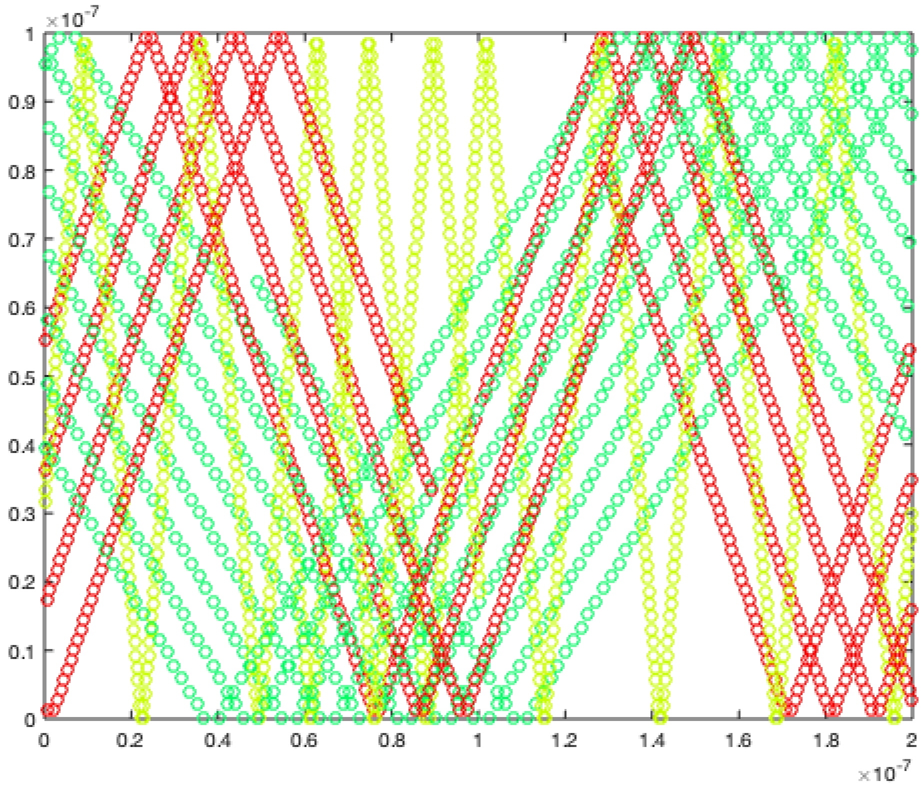
Part 3

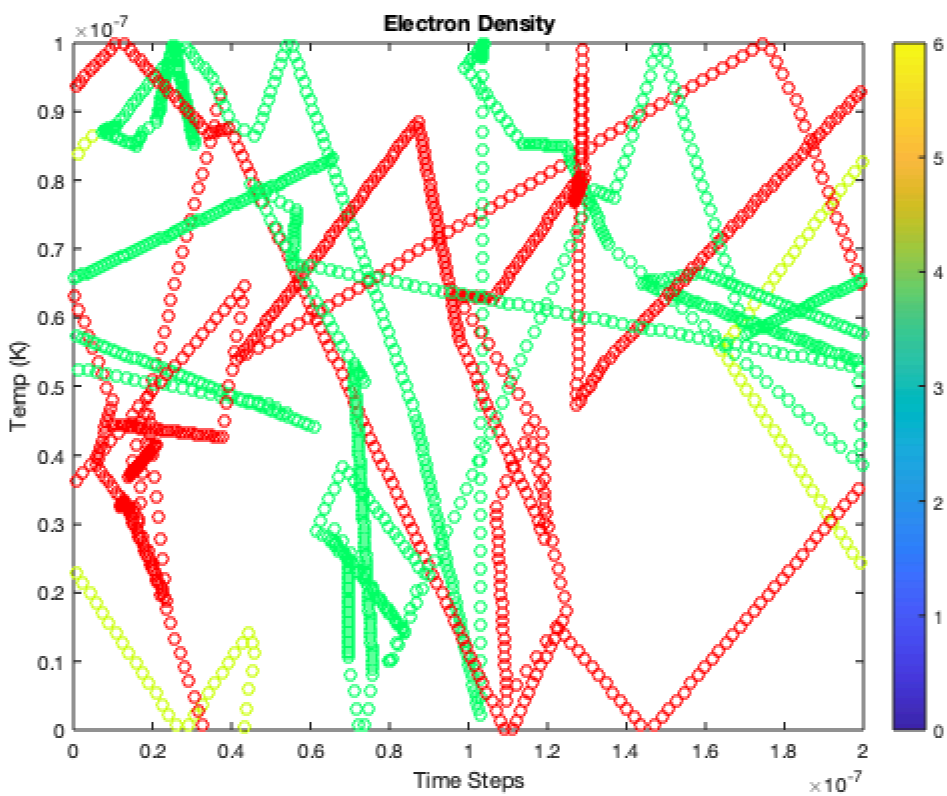
ENHANCEMENTS

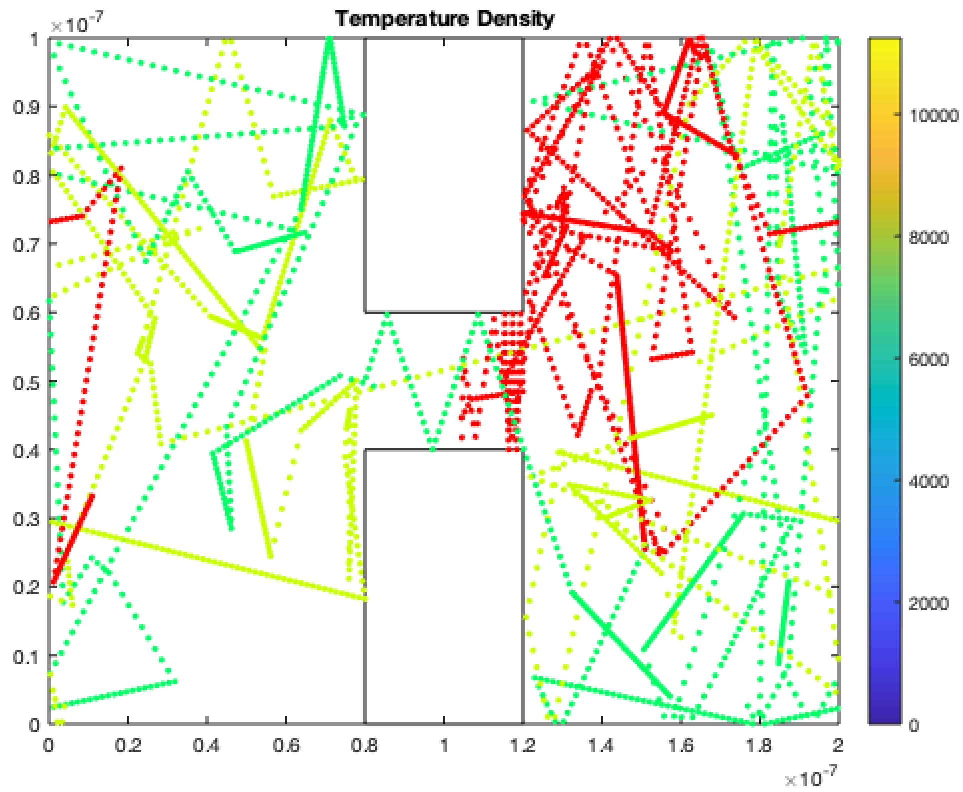
```
% Desnity map parition into 100x100 grid. Rectangles chosen to be
 specular
% always. Some regions of area have higher temperatures, while some
 are
% lower. More particles means avg. temp converges to 300K.




clear
clf


%CONSTANT DECLATARIONS

me=0.26*(9.11*10^-31); %eff mass
dt = 10*(10^-15); %sim for 1000 dt's
pscat = 1-exp(-(dt)/(0.2*10^(-12)));
kb = 1.3806*10^-23;
vth = sqrt((2*300*kb)/(me)); %thermal velocity, 2D
N = 10000; %no of particles
steps = 1000; %no of steps
p1 = 1; %partition lower bound
p2 = 3; %partition upper bound



partition=100;
count = zeros(100,100);
T = zeros(partition,partition);
velx = zeros(partition,partition);
vely = zeros(partition,partition);
pcomp = zeros(1,steps);
vnet = zeros(1,N);
x = zeros(1,N);
y = zeros(1,N);
vx = zeros(1,N);
vy = zeros(1,N);

for i = 1:N
    x(i) = rand*200*(10^-9);
    y(i) = rand*100*(10^-9);                 %initial conditions
    if(x(i)>=80*10^-9 && x(i)<=120*10^-9 && y(i)<=40*10^-9)
        x(i) = x(i) + 41*10^-9;
    end                    %if randomness puts particle in rectangle,
 then add 41nm to put it out
    if(x(i)>=80*10^-9 && x(i)<=120*10^-9 && y(i)>=40*10^-9)
```

```
        x(i) = x(i) + 41*10^-9;
    end

    vx(i) = (vth).*randn(1,1)*(1/sqrt(2));
    vy(i) = (vth).*randn(1,1)*(1/sqrt(2));
    vnet(i) = sqrt(vx(i)^2 + vy(i)^2);
end

for d = 1:steps
    pcomp(d) = rand;
end

for j = 1:steps     %amount of timesteps
    x(1:N) = x(1:N) + (vx(1:N).*dt);
    y(1:N) = y(1:N) + (vy(1:N).*dt);
    if(pscat > pcomp(j))

        for u = 1:N
            vx(u) = (vth).*randn(1,1)*(1/sqrt(2));
            vy(u) = (vth).*randn(1,1)*(1/sqrt(2));
        end

    end

    for k = 1:N   %boundary conditions
        if( y(k)<=0 || y(k)>=100*10^-9)
            vy(k) = -vy(k);
        end
        if(x(k)<=0)
            x(k) = x(k) + 200*10^-9;
        end
        if(x(k)>=200*10^-9)
            x(k) = x(k) - 200*10^-9;
        end

        if( (y(k)>=60*10^-9 || y(k)<=40*10^-9) &&
 x(k)+dt*vx(k)>=80*10^-9 && x(k)<=80*10^-9)%left side
            vx(k) = -vx(k);
        end
        if( (y(k)>=60*10^-9 || y(k)<=40*10^-9) &&
 x(k)+dt*vx(k)<=120*10^-9 && x(k)>=120*10^-9)%right side
            vx(k) = -vx(k);
        end

        if( (y(k)+dt*vy(k)>=60*10^-9 || y(k)+dt*vy(k)<=40*10^-9) &&
 x(k)<=120*10^-9 && x(k)>=80*10^-9)
            vy(k) = -vy(k);  %Horizontal components of rectangles
        end

    end
    for q=p1:p2
        colorVec = hsv(5);
        plot(x(q),y(q),'.','color', colorVec(q,:));
```

```
    end

    axis([0,200*10^-9,0,100*10^-9]);
    pause(0.00001);
    hold on;

end
rectangle('Position',[80*10^-9 60*10^-9 40*10^-9 40*10^-9]);
rectangle('Position',[80*10^-9 0 40*10^-9 40*10^-9]);

for s=1:partition
    for r=1:partition
        for c=1:N
            if( x(c)<r*2*10^-9 && x(c)>(r-1)*2*10^-9 && y(c)<s*1*10^-9
 && y(c)>(s-1)*1*10^-9)
                count(s,r) = count(s,r)+1;
                velx(s,r) = velx(s,r) + vx(c);
                vely(s,r) = vely(s,r) + vy(c);
            end
        end
    end
end

for v=1:partition
    for z=1:partition
        T(v,z) = (me/(2*kb))*( mean(velx(v,z).^2)+
 mean(vely(v,z).^2) );
    end
end

figure(2);
surf(count);
title('Electron Density');
view(2);
colorbar;
caxis([min(min(count)),  max(max(count))]);

figure(3);
surf(T);
title('Temperature Density');
view(2);
colorbar;
caxis([min(min(T)),  max(max(T))]);

sprintf('Avg temp is %0.5e K' ,mean(T(:)))


ans =

    'Avg temp is 2.64904e+02 K'
```


×10^-7
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
×10^-7


Electron Density
Temp (K)
Time Steps
×10^-7
×10^-7
0
1
2
3
4
5
6



Published with MATLAB® R2019b