```
% ELEC4700 Assignment 3

% By Huanyu Liu 100986552

% Part 3

clear

clc

L=40;

W=60;

sigma1=1;

sigma2=0.01;

k=L*W;

G=sparse(k,k);

Z = zeros(k, 1);

Vo=1; % voltage drop across the whole area = the beginning voltage

S = zeros(L, W);    % sigma

for x = 1 : L

    for y = 1 : W

        if x >= 0.4*L && x <= 0.6*L && (y <= 0.4*W || y >= 0.6*W)

            % area in the blocks

            S(x, y) = sigma2;

        else

            % area outside the blocks

            S(x, y) = sigma1;
```

```
        end
    end
end
for i = 1:L
    for j = 1:W
        n = j + (i-1)*W;
        nxm = j + (i-2)*W;
        nxp = j + i*W;
        nym = j-1+(i-1)*W;
        nyp = j+1+(i-1)*W;
        if i == 1
            G(n, n) = 1;
            % assume the current flows from left to right
            Z(n) = Vo;
        elseif i == L
            G(n, n) = 1;
            % by default Z(n)=0 here
        elseif j == 1  % lower bound
            if i > 0.4*L && i < 0.6*L % inside the blocks
                G(n, n) = -3;
                G(n, nyp) = sigma2;
                G(n, nxp) = sigma2;
```

```
            G(n, nxm) = sigma2;
        else
            G(n, n) = -3;
            G(n, nyp) = sigma1;
            G(n, nxp) = sigma1;
            G(n, nxm) = sigma1;
        end
    elseif j == W % upper bound
        if i > 0.4*L && i < 0.6*L % inside the block
            G(n, n) = -3;
            G(n, nym) = sigma2;
            G(n, nxp) = sigma2;
            G(n, nxm) = sigma2;
        else
            G(n, n) = -3;
            G(n, nym) = sigma1;
            G(n, nxp) = sigma1;
            G(n, nxm) = sigma1;
        end
    else
        if i > 0.4*L && i < 0.6*L && (j < 0.4*W||j > 0.6*W)
            % inside the blocks
```

```
                G(n, n) = -4;

                G(n, nyp) = sigma2;

                G(n, nym) = sigma2;

                G(n, nxp) = sigma2;

                G(n, nxm) = sigma2;

            else

                G(n, n) = -4;

                G(n, nyp) = sigma1;

                G(n, nym) = sigma1;

                G(n, nxp) = sigma1;

                G(n, nxm) = sigma1;

            end

        end

    end

end

% G*V=Z

V3 = G\Z;

V4=reshape(V3,L,W);

[Ex, Ey] = gradient(V4);

Jx = S.*Ex;

Jy = S.*Ey;

J = sqrt(Jx.^2 + Jy.^2);
```

```
% assume R=1

V=zeros(L,W);

V(:,1)=1;

for x=1:L

    for y=2:W

    V(x,y)=V(x,y-1)-J(x,y)/L;

    end

end

[E1,E2]=gradient(V);



 % Initialize the parameters

n=1000; % number of particles

Length=200e-9;

xs=Length/L;  % step size in length

Width=100e-9;

ys=Width/W;  % step size in width

T=300; % temperture of the backgound

tao=0.2e-12; % the given mean time between collisions

m0=9.109e-31; % mass of a particle

mn=0.26*m0; % effective mass

kb=1.38e-23; % constant coeffient

vth=sqrt(2*kb*T/mn); % average speed of each particle
```

```
con=1e11; % The electron concentration

% Initialize the positions of each particle

Pox = Length*rand(1,n);

Poy = Width*rand(1,n);



op1 = Pox >= 0.4*Length;

op2 = Pox <= 0.6*Length;

op = op1&op2; %specify the locations of the blocks

count = sum(op(:)==1); % number of particles that may be in the blocks

Poy(op) = 0.4*Width.*ones(1,count) + 0.2*Width.*rand(1,count); % limit the range so no
particles can exist in the blocks



% New parameters for assignment3

e=1.60217662e-19;

F1=E1.*e;

a1=F1./m0;

F2=E2.*e;

a2=F2./m0;



% Initialize the speed of each particle and measure the initial temperature

for num=1:n

Vx(num) = randn()*vth/sqrt(2);

Vy(num) = randn()*vth/sqrt(2);
```

```
end

% draw the first locations of the particles and the blocks

figure(1)

plot(Pox,Poy,'.');

xlim([0 Length]);

ylim([0 Width]);

line([0.4*Length 0.4*Length], [0 0.4*Width]);

line([0.4*Length 0.6*Length], [0.4*Width 0.4*Width]);

line([0.6*Length 0.6*Length], [0 0.4*Width]);

line([0.4*Length 0.4*Length], [Width 0.6*Width]);

line([0.4*Length 0.6*Length], [0.6*Width 0.6*Width]);

line([0.6*Length 0.6*Length], [0.6*Width Width]);

hold on

% more parameters that will be used in the loop

TStop = 1e-12; % max running time

t=0; % start time

dt=1e-14; % step time

ddt = 0; % time since last timestop

 while t < TStop

     Pscat = 1-exp(-ddt/tao); % scattering posibility

     if Pscat > rand % if scatter

         ddt=0; % reset the parameter for the possibility as required
```

```
        Vx = randn(1,n).*vth/sqrt(2);

        Vy = randn(1,n).*vth/sqrt(2); % velocity changes (in maxwell-boltzmann
distribution)

    else % nothing happens, same speed the next duration of time step

        ddt=ddt+dt; % add the timestep size to the parameter

    end



    xp=round(Pox./xs);  % get the indice of the positions for the accelerate

    yp=round(Poy./ys);

    for m=1:n

        if xp(m)<=0   % fix the rounding

            xp(m)=1;

        elseif xp(m)>=41

            xp(m)=40;

        end

        if yp(m)<=0

            yp(m)=1;

        elseif yp(m)>=61

            yp(m)=60;

        end

       Vx(m) = Vx(m) + a1(xp(m),yp(m)).*t;

       Vy(m) = Vy(m) + a2(xp(m),yp(m)).*t;

    end
```

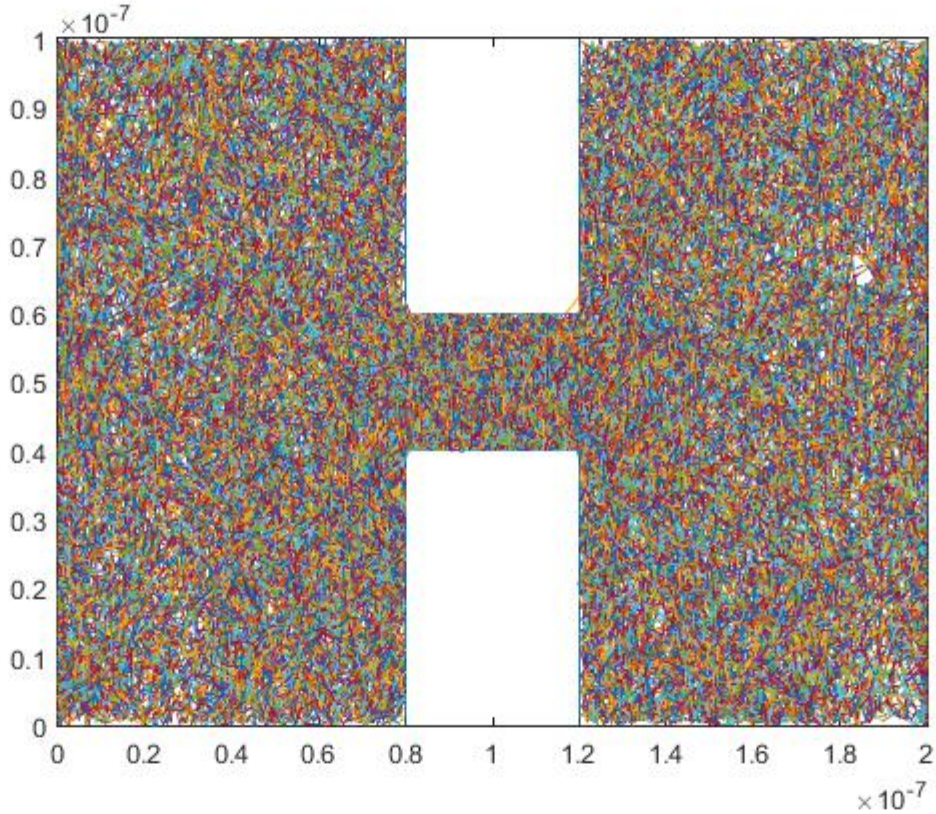
```
    tPx = Pox + Vx.*dt; % predict the position

    tPy = Poy + Vy.*dt;

% when the particles go to the right and left border

    px1 = Pox >= Length;

    Pox(px1) = Pox(px1) - Length;

    px2 = Pox <= 0;

    Pox(px2) = Pox(px2) + Length;

     % when the particles will go across a border

    a=tPy<=0.4*Width;

    b=tPy>=0.6*Width;

    x=a|b;

    e=tPx>=0.4*Length;

    % but now it it outside the blocks

    f=Pox<=0.4*Length;

    px3=x&e&f;

    % then it will be reflected

        Vx(px3) = Vx(px3).*(-1); % hit boarder 0.4*L



    g=tPx<=0.6*Length;

    h=Pox>=0.6*Length;

    px4=x&g&h;

        Vx(px4) = Vx(px4).*(-1); % hit boarder 0.6*L
```

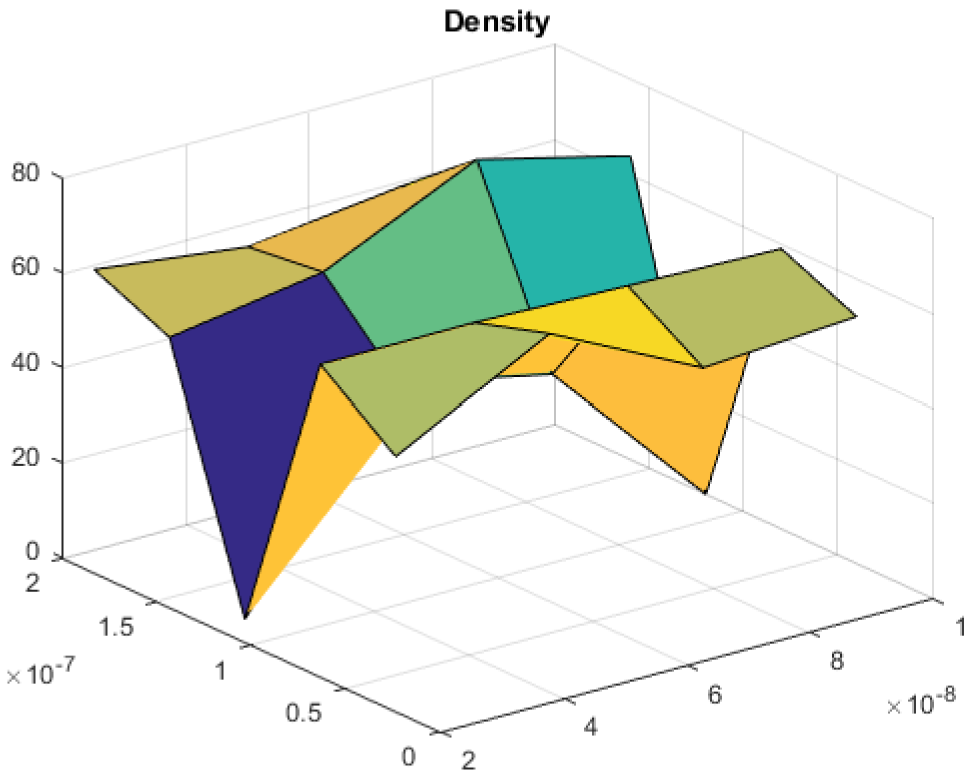
```
py1 = tPy <= 0;

Vy(py1) = Vy(py1) .* (-1);

py2 = tPy >= Width;

Vy(py2) = Vy(py2) .* (-1);

c=tPx>=0.4*Length;

d=tPx<=0.6*Length;

y=c&d;

i=tPy<=0.4*Width;

j=Poy>=0.4*Width;

py3=y&i&j;

    Vy(py3) = Vy(py3) .* (-1); % hit boarder 0.4*H

k=tPy>=0.6*Width;

l=Poy<=0.6*Width;

py4=y&k&l;

    Vy(py4) = Vy(py4) .* (-1); % hit boarder 0.6*H



    % now all velocity have been modified to the correct direction,

    % update the position

PreviousPox = Pox;

PreviousPoy = Poy;

Pox = Pox + Vx.*dt;
```

```
    Poy = Poy + Vy.*dt;



    figure(1)

    for i=1:n

    plot([PreviousPox(i),Pox(i)],[PreviousPoy(i),Poy(i)]);

    end

    xlim([0 Length]);

    ylim([0 Width]);

    hold on





    pause(0.01)

    t=t+dt;

end



n1=Pox<0.2*Length;

n2=Pox<0.4*Length;

n3=Pox<0.6*Length;

n4=Pox<0.8*Length;

n5=Poy<0.25*Width;

n6=Poy<0.5*Width;

n7=Poy<0.75*Width;
```

```
Den=zeros(5,4);

Den(1,1)=sum(n1&n5);

Den(1,2)=sum(n1&n6&(~n5));

Den(1,3)=sum(n1&n7&(~n6));

Den(1,4)=sum(n1&(~n7));

Den(2,1)=sum((~n1)&n2&n5);

Den(2,2)=sum((~n1)&n2&n6&(~n5));

Den(2,3)=sum((~n1)&n2&n7&(~n6));

Den(2,4)=sum((~n1)&n2&(~n7));

Den(3,1)=sum((~n2)&n3&n5);

Den(3,2)=sum((~n2)&n3&n6&(~n5));

Den(3,3)=sum((~n2)&n3&n7&(~n6));

Den(3,4)=sum((~n2)&n3&(~n7));

Den(4,1)=sum((~n3)&n4&n5);

Den(4,2)=sum((~n3)&n4&n6&(~n5));

Den(4,3)=sum((~n3)&n4&n7&(~n6));

Den(4,4)=sum((~n3)&n4&(~n7));

Den(5,1)=sum((~n4)&n5);

Den(5,2)=sum((~n4)&n6&(~n5));

Den(5,3)=sum((~n4)&n7&(~n6));

Den(5,4)=sum((~n4)&(~n7));

[X,Y]=meshgrid(Width/4:Width/4:Width,Length/5:Length/5:Length);
```

```
figure(2)

surf(X,Y,Den);

title('Density');
```





% comment：The particles tend to leave the centre line between the blocks.

% To see whether this is the correct rule, the next step should be cancelling the blocks or dividing the flame into more areas.

Published with MATLAB® R2015a