EE 5322 Intelligent Control- Exam 1
Spring 2014

1. This is a take home exam. YOU MUST WORK ALONE.

Any cheating or collusion will be severely punished.

2. To obtain full credit, show all your work. No partial credit will be given without the supporting work.

3. The three problems are weighted equally

4. Please sign this form and include it as the first page of your submitted exam.

....

Typed Name:___________M.S. Raghavendra Sriram____________________

Pledge of honor:

"On my honor I have neither given nor received aid on this examination."

e-Signature: ___________M.S. Raghavendra Sriram____________________

Mobile Robot Control & Potential Fields

1. **Potential Field.** Use MATLAB to make a 3-D plot of the potential fields described below. You will need to use plot commands and maybe the mesh function. The work area is a square from (0,0) to (10,10) in the (x,y) plane. The goal is at (10,10). There are obstacles at (3,2) and (7,6). Use a repulsive potential of K_i / r_i for each obstacle, with r_i the vector to the i-th obstacle. For the target use an attractive potential of $K_T r_T$, with r_T the vector to the target. Adjust the gains to get a decent plot. Plot the sum of the three potential fields in 3-D.

2. **Potential Field Navigation.** For the same scenario as in Problem 1, a mobile robot starts at (0,0). The front wheel steered mobile robot has dynamics $\dot{x} \square V$

$$\cos\square\sin\square$$

$$\dot{y} \square V \quad \cos\square\cos\square$$

$$\dot{\square} \square \frac{V}{L} \sin\square$$

with (x,y) the position, □ the heading angle, V the wheel speed, L the wheel base, and □ the steering angle. Set L= 2.

   a. Compute forces due to each obstacle and goal. Compute total force on the vehicle at point (x,y).
   b. Design a feedback control system for force-field control. Draw your control system.
   c. Use MATLAB to simulate the nonlinear dynamics assuming a constant velocity V and a steerable front wheel. The wheel should be steered so that the vehicle always goes downhill in the force field plot. Plot the resulting trajectory in the (x,y) plane. Use a square from (0,0) to (12,12).

3. **Platoon of Mobile Robots.**

There are 5 robots in a platoon. Robot 1 is the leader. For each robot i take the simplified Newton's law dynamics (with mass=1)

$$\ddot{x}_i \square F_{xi}$$

$$\ddot{y}_i \square F_{yi}$$

with (x,y) the position of the vehicle and F_x,F_y the forces in the x and y direction respectively.

   a. Program the forces for the leader node 1 to avoid the obstacles and go to the target. Same scenario as above (but with these simplified dynamics).
   b. Program the force on each follower to stay ½ unit from the leader and not to run into each other. Use repulsive POTENTIAL between followers as K_i / r_{ij}^2 with r_{ij}= distance between followers i and j. For the potential to the leader, use something like

$$V_{iL} \square \tfrac{1}{2} (r_{iL} \square r_D)^2$$

with r_{iL} the distance from follower i to the leader, and r_D the desired separation. Play with the potentials to make it work properly. Compute forces properly using calculus.

c. Simulate. Plots trajectories in (x,y) plane. Start all robots at (0,0).

4. **20% EXTRA CREDIT-** Make a routine to plot the robots as points on the screen in realtime as the robots move. Then, you can see them move.

   If you feel like it, make a movie too. Use MATLAB fns moviein, getframe, etc.

DO NOT DO THE FOLLOWING HOMEWORKS YET.

THEY WILL CHANGE

Date: 11 March, 2014 M.S. Raghavendra Sriram
1000854840
EE 5322 Intelligent Control Systems

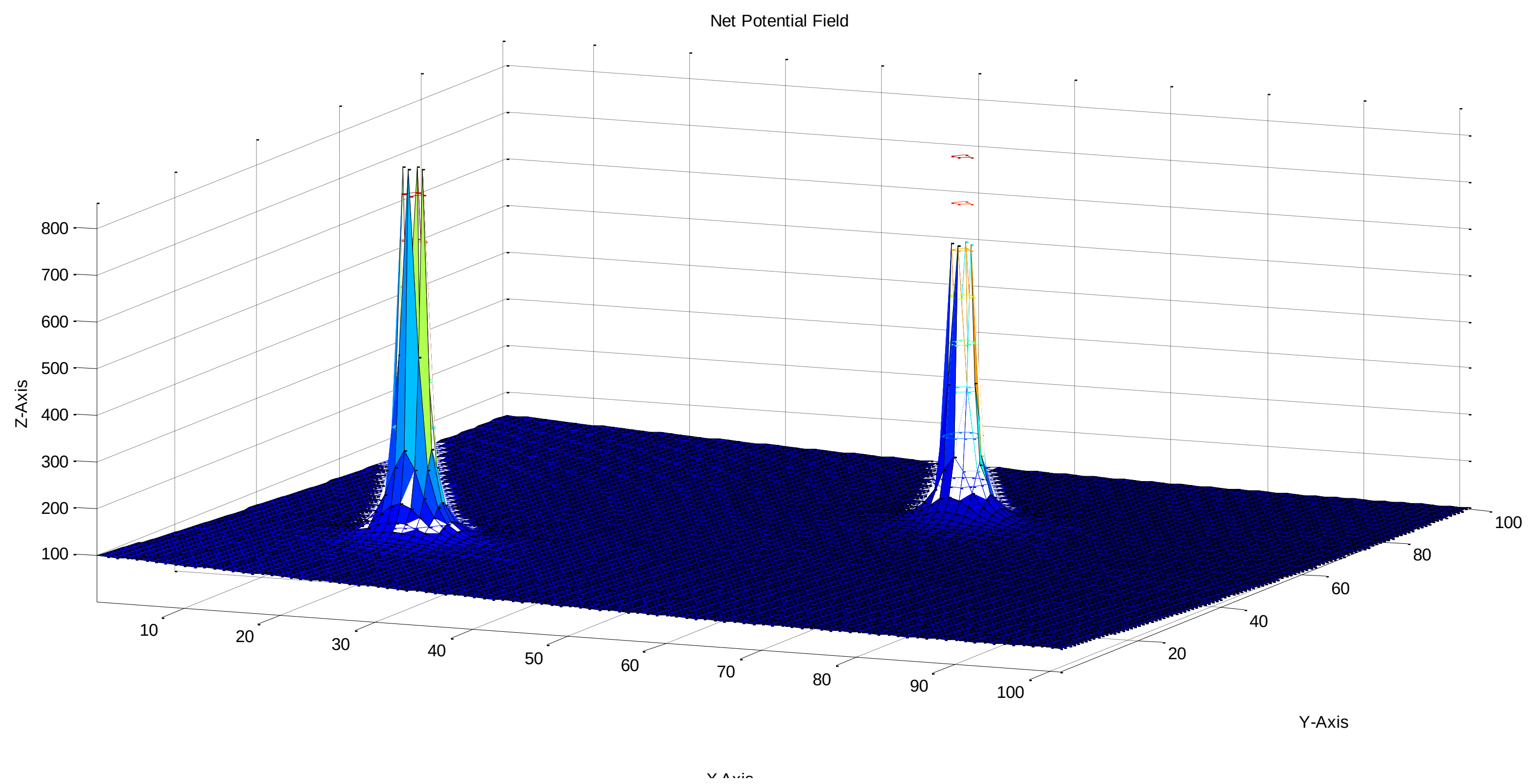
Take Home Exam - 1

A1.a)

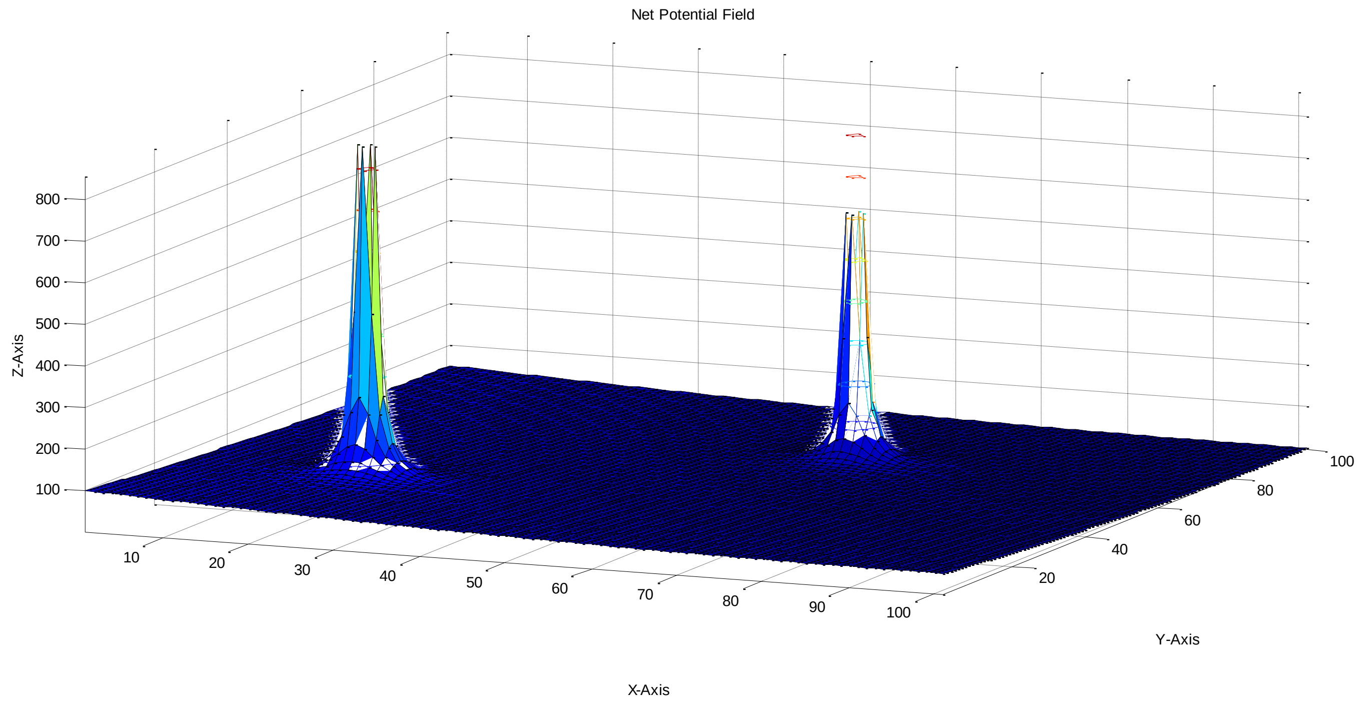
CODE:

```
clc;
clear all;
close all;

x=0:0.1:10;%x axis plot range plotting samples from 0 to 12 with samples every 0.3 sec
y=0:0.1:10;%y axis plot range plotting samples from 0 to 12 with samples every 0.3 sec
lx=length(x);%length of x co-ordinates
ly=length(y);%length of y co-ordinates
V=zeros(lx,ly);%Potential matrix

xg=10;yg=10; %Co-ordinates of the goal in x and y planes
Kg=1; %Goal potential constant

x1=3;y1=2; %obstacle 1 - Co-ordinates
K1=4; %Obstacle 1 potential constant

x2=7;y2=6; %obstacle 2 - Co-ordinates
K2=3; %Obstacle 2 potential constant

%Calculation of potential on the target due to the goal, obstacle 1 &
%obstacle 2
for i=1:lx
 for j=1:ly
 rg=((xg-x(i))^2+(yg-y(j))^2)/2; %Distance from goal(10,10)
 r1=((x1-x(i))^2+(y1-y(j))^2)/2; %Distance from obstacle 1(3,2)
 r2=((x2-x(i))^2+(y2-y(j))^2)/2; %Distance from obstacle 2(7,6)

 V(i,j)=Kg.*rg+K1./r1+K2./r2;%Potential field equation.
 end
end
mesh(V)
surf(V,'FaceColor','interp','EdgeColor','none','FaceLighting','phong')
hold on
contour3(V)
hold on;
axis tight
camlight left
grid on;
title('Net Potential Field ');
xlabel('X-Axis')
ylabel('Y-Axis')
zlabel('Z-Axis')
```

Plots



Net Potential Field
Z-Axis
800
700
600
500
400
300
200
100
10
20
30
40
50
60
70
80
90
100
20
40
60
80
100
Y-Axis



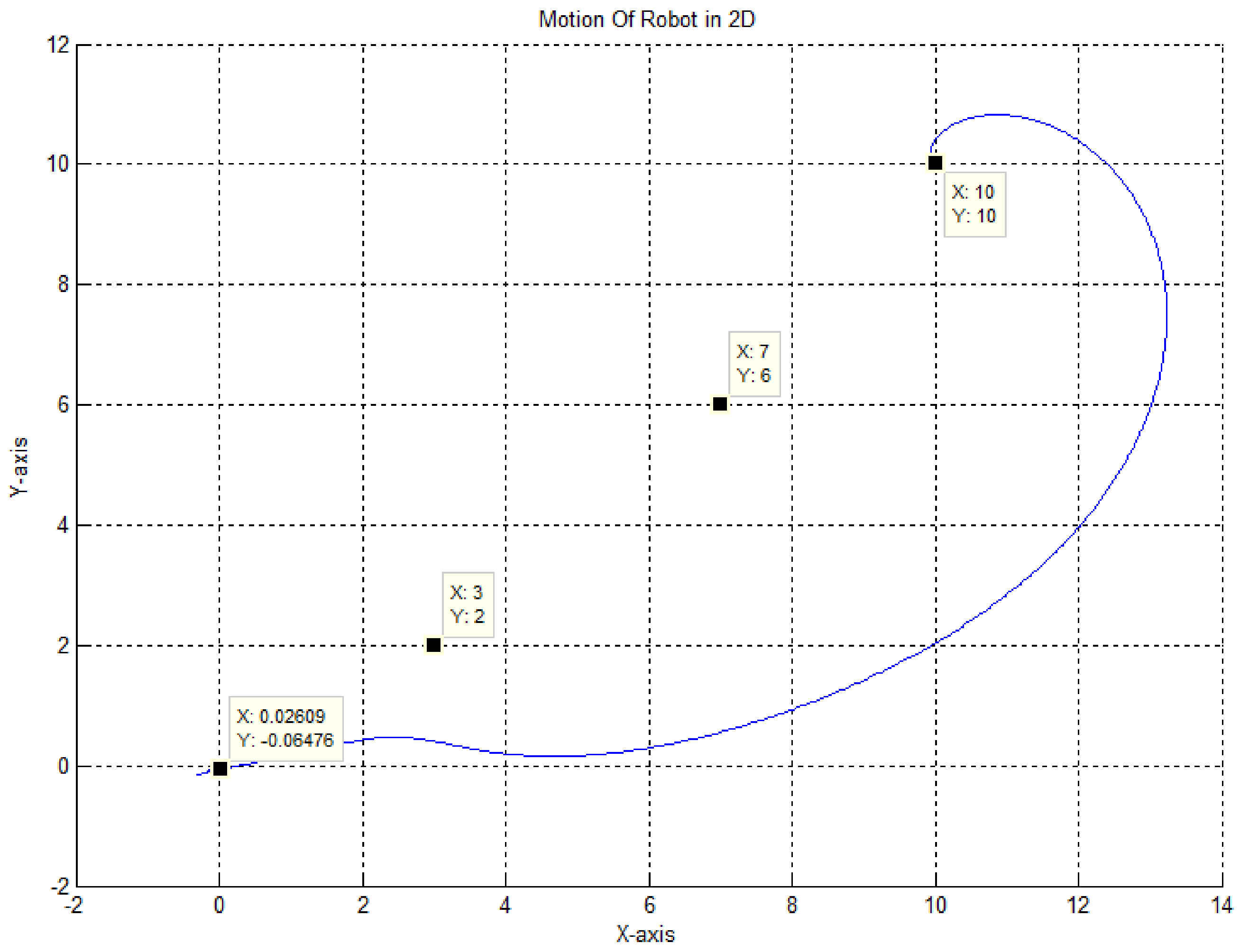
Net Potential Field
Z-Axis
800
700
600
500
400
300
200
100
10
20
30
40
50
60
70
80
90
100
20
40
60
80
100
Y-Axis
X-Axis

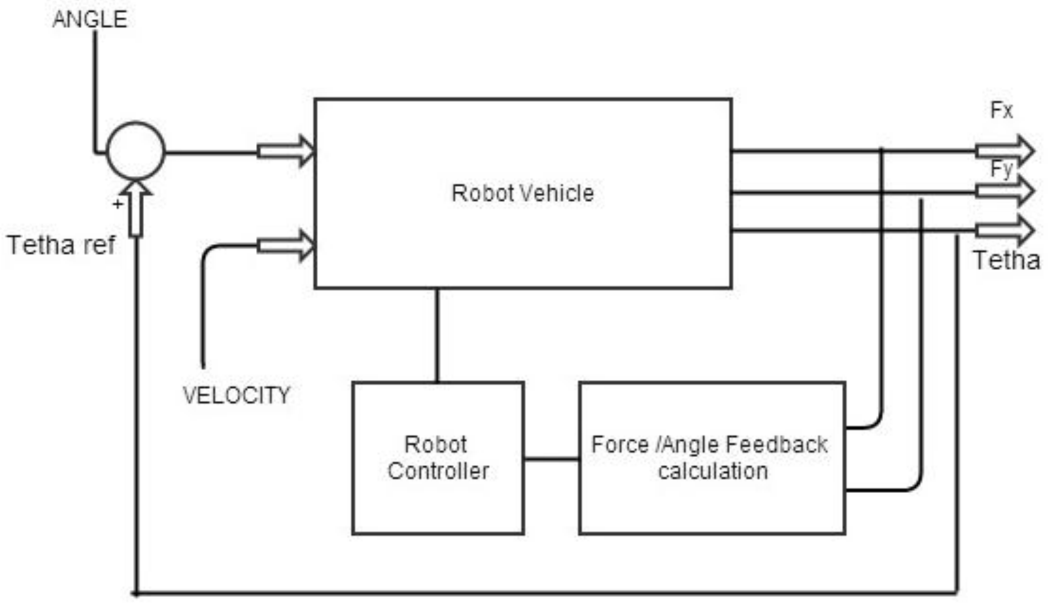
Q2).

CODE :

```
clc;
clear all;
close all;
xg=10;yg=10;%goal co-ordinates
z0=[pi/6;0;0];

x=0;y=0;%initial points
ti = 0;
tf = 6.1;
t = ti:0.01:tf;%from initial 0 to final point tf
[t,z]=ode23('para',t,z0);
hold on;
%mesh(z);%how do i get a 3D rendition of this plot...
figure(1)
plot(z(:,2),z(:,3))
grid on;
hold on;
plot(3,2,'*g',7,6,'*b',10,10,'*r');
%axes(0,0,12,12);
grid on;
hold on;

title('Motion Of Robot in 2D');
xlabel('X-axis')
ylabel('Y-axis')


 function alf1=alfa(x,y)
xg=10;yg=10; % goal
x1=3;y1=2; %obstacle 1
x2=7;y2=6; %obstacle 2
Kg=20; %Goal force constant
K1=15; %Obstacle 1 force constant
K2=25; %Obstacle 2 force constant
rg=((xg-x)^2+(yg-y)^2)^0.5; %Distance from the goal(xg,yg)
r1=((x1-x)^2+(y1-y)^2)^0.5; %Distance from obstacle 1(xo1,yo1)
r2=((x2-x)^2+(y2-y)^2)^0.5; %Distance from obstacle 2(xo2,yo2)

Fgx=Kg*(xg-x)/rg; %Force due to goal in x-direction
Fgy=Kg*(yg-y)/rg; %Force due to goal in y-direction

F1x=-K1*(x1-x)/(r1^2); %Force due to obstacle 1 in x-direction
F1y=-K1*(y1-y)/(r1^2); %Force due to obstacle 1 in y-direction

F2x=-K2*(x2-x)/(r2^2); %Force due to obstacle 2 in x-direction
F2y=-K2*(y2-y)/(r2^2); %Force due to obstacle 2 in y-direction

Fx=Fgx+F1x+F2x;
Fy=Fgy+F1y+F2y;
alf1=atan2(Fy,Fx);

end


function zdot=para(t,z)
alf=alfa(z(2),z(3));
K=10;V=5;L=2;
phi=K*(alf-z(1));
zdot=[V/L*sin(phi);V*cos(phi)*cos(z(1));V*cos(phi)*sin(z(1))];
end
```

Plot :


Motion Of Robot in 2D
X: 10
Y: 10
X: 7
Y: 6
X: 3
Y: 2
X: 0.02609
Y: -0.06476
Y-axis
X-axis
12
10
8
6
4
2
0
-2
-2
0
2
4
6
8
10
12
14


ANGLE
+
Tetha ref
VELOCITY
Robot Vehicle
Fx
Fy
Tetha
Robot Controller
Force /Angle Feedback calculation

Q3.

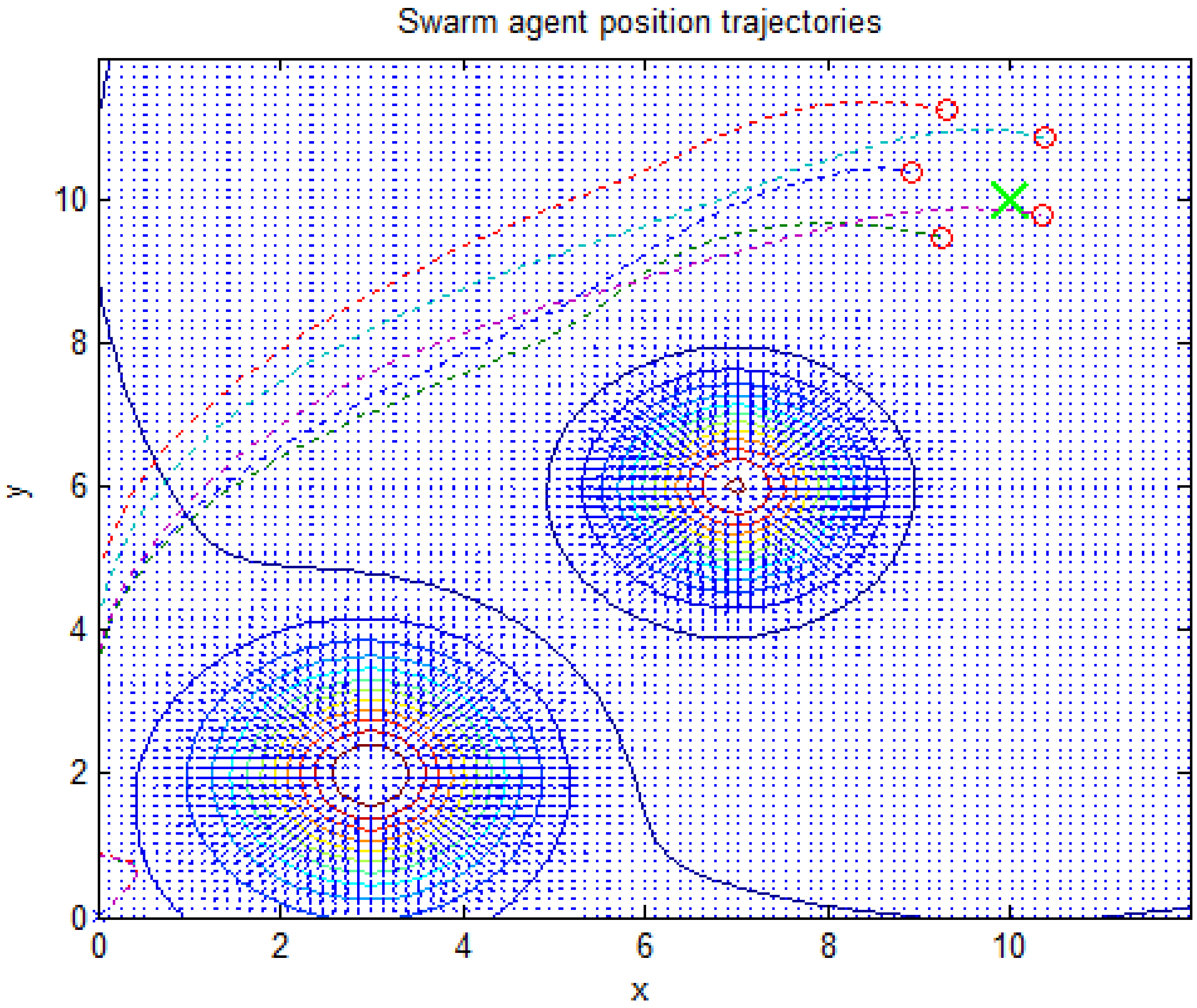
Code:

```
clc
clear all
close all

N=5 % number of agents present in the system.
k1=3;
k2=k1;
kv=1.1; % Velocity damping ('k')
kf=2;   % Profile following gain
b=5;    % repel function parameters
c=1;    % rulpulsion region parameter


% simulation parameters:
Tf=10;              % Total time in seconds
T_it=0.05;          % Time iteration slices
Tspan=0:T_it:Tf+T_it;  %Total time span

%initial conditions:
X0=0.01*rand(1)          %initial positions in X and Y dimensions
Y0=0.01*rand(1)
Vx0=0.01*rand(1,N)       %initial velocities in X and Y dimensions
Vy0=0.01*rand(1,N)

% Initialization of position and velocity
X(1,1:N)=X0;             % 1st dimension is time, second is N values of X (Y) position
Y(1,1:N)=Y0;
Vx(1,1:N)=Vx0;
Vy(1,1:N)=Vy0;

% Final Goal position of vehicle
xg=[10; 10];
w1=120;        % goal function and obstacle function multiplication factor
w2=0.1;        % w1=120 and w2=0.1

value0=0;
value1=0;
Dp1=0;
Dp2=0;
Dv1=0 ;
Dv2=0 ;
Df=0;

Dmax=1;
ScaleU=1; % This is used to change the magnitude of the control input ux and uy.

for n=1:Tf/T_it-1

  dp1=2*rand(3,N)-1;
  dv1=2*rand(3,N)-1;
  dp2=2*rand(3,N)-1;
  dv2=2*rand(3,N)-1;
  df0=2*rand(3,N)-1;

  % Calculate the average position and velocity of the swarm at current n.
  if (N>1)                         % Test to see if there is more than one swarm member
    xbar=mean([X(n,:)' Y(n,:)']);      % 2x1 vector of means in X and Y dimensions
    vbar=mean([Vx(n,:)' Vy(n,:)']);    % and for velocity also
  else
    xbar=[X(n,:)' Y(n,:)'];          % If only one swarm member then mean is just that member's position
    vbar=[Vx(n,:)' Vy(n,:)'];
  end


  ErrorMatrix=[X(n,:)-xbar(1);
          Y(n,:)-xbar(2);
```

```
            Vx(n,:)-vbar(1);
            Vy(n,:)-vbar(2)];

    normEi=0;
    dp=0*((ones(3,1)*normEi));
    dv=0*((ones(3,1)*normEi));
    df=0*(df0/Dmax*Df);

    EP_hat=[X(n,:)-dp(1,:); Y(n,:)-dp(2,:)]; % EP_hat is the position error of agent i with sensing error.


    % The 'for' loop below caculates the effect of the repel term on each agent
    for i=1:N
        Ediff=EP_hat(:,i)*ones(1,N)-EP_hat; % Column j (1<=j<=N) contains the error position difference of agent i and agent j in [x;y] direction,
respectively.
        dist=sqrt(sum(Ediff.*Ediff)); % The jth component is the norm of the error difference of agent i and j. It's equal to the distance from
agent i to agent j; or .
        xrepel(i)=sum(b*exp(-dist.^2/c).*(X(n,i)-X(n,:)));
        yrepel(i)=sum(b*exp(-dist.^2/c).*(Y(n,i)-Y(n,:)));

    end
    % The 'for' loop below calculates the discrete gradient for each agent at current position.
    A=[];
    for i=1:N
        J_current = goal_func([X(n,i);Y(n,i)],xg,w2) + obs_func([X(n,i);Y(n,i)],w1);
        F_x=Vx(n,i)*T_it;
        F_y=Vy(n,i)*T_it;
        FJx=goal_func([X(n,i)+F_x;Y(n,i)],xg,w2) + obs_func([X(n,i)+F_x;Y(n,i)],w1) - J_current;
        FJy=goal_func([X(n,i);Y(n,i)+F_y],xg,w2) + obs_func([X(n,i);Y(n,i)+F_y],w1) - J_current;
        A(i,:)=[FJx/F_x FJy/F_y]
    end

    % Calculate the control input on two dimension x,y. Each u (i.e., ux, uy) is a 1xN vector.
    ux=-k1*(X(n,:)-mean(X(n,:))-dp(1,:)) - k2*(Vx(n,:)-mean(Vx(n,:))-dv(1,:)) - kv*Vx(n,:) + xrepel - kf*(A(:,1)'-df(1,:));
    uy=-k1*(Y(n,:)-mean(Y(n,:))-dp(2,:)) - k2*(Vy(n,:)-mean(Vy(n,:))-dv(2,:)) - kv*Vy(n,:) + yrepel - kf*(A(:,2)'-df(2,:));


    % Calculates the position and velocity in the next time step (Euler's method).
    X(n+1,:)=X(n,:)+Vx(n,:)*T_it;
    Y(n+1,:)=Y(n,:)+Vy(n,:)*T_it;

    Vx(n+1,:)=Vx(n,:) + ux*ScaleU*T_it;
    Vy(n+1,:)=Vy(n,:) + uy*ScaleU*T_it;

end

t=[1:length(X)]'*T_it;

% Plot the Agent trajectories

% Plot the functions:
kk=12;
xx=0:(kk+1)/100:kk;   % For our function the range of values we are considering
yy=xx;

% Compute the obstacle and goal functions

for jj=1:length(xx)
    for ii=1:length(yy)
        zz(ii,jj)=obs_func([xx(jj);yy(ii)],w1);
    end
end
for jj=1:length(xx)
    for ii=1:length(yy)
        zzz(ii,jj)=goal_func([xx(jj);yy(ii)],xg,w2);
    end
end

[temp1,temp2]=size(X);
if (Tf>=20)
    temparray = 1:(Tf/20)*5:temp1;
else
```

```
  temparray = 1:temp1;
end

figure(1)
clf
contour(xx,yy,zz+zzz,12)
colormap(jet)
xlabel('x');
ylabel('y');
hold on
[DX, DY] = gradient(zz+zzz, 1, 1);
quiver(xx, yy, -DX, -DY);
hold on;
% Plot initial and final positions
plot(xg(1),xg(2),'gx','MarkerSize',16,'linewidth',2);
plot(X(temparray,:),Y(temparray,:),'LineStyle',':')
xlabel('x')
ylabel('y')
title('Swarm agent position trajectories')
hold on
plot(X0,Y0,'bx')
hold on
plot(X(temp1,:),Y(temp1,:),'ro');

% Make the Movie :

Xd=[];
Yd=[];
figure(2)
for i=1:N
  Xd(:,i)=decimate(X(:,i),10);       % Decimate data to speed up movie % Set decimate factor
  Yd(:,i)=decimate(Y(:,i),10);
end
  [temp1d,temp2]=size(Xd);

  for j=1:temp1d
      clf;
      contour(xx,yy,zz+zzz,10)
      colormap(jet);
      hold on;
      plot(xg(1),xg(2),'gx','MarkerSize',16,'linewidth',2);
      hold on;
      plot(Xd(j,:),Yd(j,:),'bo','MarkerSize',10,'linewidth',3);
      axis([min(min(X)) max(max(X)) min(min(Y)) max(max(Y))]);
      xlabel('x')
      ylabel('y')
      hold on;
      [DX, DY] = gradient(zz+zzz, 1, 1);
      quiver(xx, yy, -DX, -DY);
      hold on;
      title('Swarm agent position trajectories')
      M(:,j) = getframe;

   end
   movie(M)
  % save Movie
  filename = 'Potential Field - 5 Agents movie';
  movie2avi(M, filename, 'fps', 20);


function goal_value=goal_func(x,xgoal,w2)

[m,n]=size(x);

Err0= x-xgoal*ones(1,n);
goal_value=w2*sum(Err0.*Err0);


function obs=obs_func(x,w1)
obs=w1*max([exp(-0.5*((x(1,1)-3)^2+(x(2,1)-2)^2)),exp(-0.7*((x(1,1)-7)^2+(x(2,1)-6)^2))]);
Solution :

N =
      5

X0 =

    0.0081

Y0 =

    0.0062

Vx0 =

    0.0096    0.0035    0.0082    0.0029    0.0051

Vy0 =

    0.0006    0.0067    0.0091    0.0040    0.0098
```

Plot :


Swarm agent position trajectories
y
x