29 Flight Control of a Hovercraft

You are tasked with developing simple control systems for two types of hovercraft moving in the horizontal plane. As you know, a hovercraft rests on a cushion of air, with very little ground resistance to motion in the surge (body-referenced forward), sway (body-referenced port), and yaw (taken positive counterclockwise viewed from above) degrees of freedom. The simplified dynamic equations of motion are:

$$\begin{aligned} \dot{u} &= -u + F_u \\ \dot{v} &= F_v \\ \dot{r} &= M, \end{aligned}$$

where the surge and sway velocities are u and v and the control forces in the u- and v-directions are F_u and F_v, respectively. The yaw rate is r and the control moment is M.

There are two types of craft we will consider: 1) one where F_u, F_v, and M are all commanded independently, e.g., using independent thrusters, and 2) one where $F_v = p_v \delta F_u$ and $M = p_\phi \delta F_u$. This second case is encountered if there is a surge thruster (or set), whose exhaust pushes on a rudder with (body-referenced) angle δ. To create a lateral force or moment, you have to have some F_u and some rudder action. We will use the values $p_v = 0.2$ and $p_\phi = -0.2$ in this problem. Notice negative sign in p_ϕ; it means that a positive rudder action, counterclockwise if viewed from above, leads to a negative body torque, that is, clockwise. This is typical when the rudder is behind the thruster, and near the back of the craft.

The dynamic equations are augmented with some kinematic relations to evolve the location of the craft in a fixed frame:

$$\begin{aligned} \dot{X} &= u\cos\phi - v\sin\phi \\ \dot{Y} &= u\sin\phi + v\cos\phi \\ \dot{\phi} &= r, \end{aligned}$$

where X and Y denote the location in Cartesian coordinates, and ϕ is the yaw angle. See the figure below.

Create a six-state model for each vehicle type from the above equations, and perform the following tasks:

1. By putting in different settings and combinations of F_u, F_v, M, δ, convince yourself that for each of the two cases, the behavior is like a hovercraft. In the first case, turns occur completely independently of translational motion in the X, Y plane, whereas for the second, lateral force and turning torque occur together, and they scale with both δ and F_u.

2. For Case 1: From a full-zero starting condition $\vec{s} = [u, v, r, X, Y, \phi] = [0, 0, 0, 0, 0, 0]$, the objective is to move to to $\vec{s}_{desired} = [u, v, r, X, Y, \phi]_{desired} = [0, 0, 0, 1m, 1m, \pi rad]$

under closed-loop control. To do this, inside the derivative call for your simulation, make three error signals:

$$\begin{aligned} e_X &= X - X_{desired} \\ e_Y &= Y - Y_{desired} \\ e_\phi &= \phi - \phi_{desired}. \end{aligned}$$

Then position errors in the body-referenced frame are a simple rotational transformation

$$\begin{aligned} e_u &= \cos\phi\ e_X - \sin\phi\ e_Y \\ e_v &= \sin\phi\ e_X + \cos\phi\ e_Y. \end{aligned}$$

Your control law then concludes with:

$$\begin{aligned} F_u &= -k_{p,u}e_u - k_{d,u}u \\ F_v &= -k_{p,v}e_v - k_{d,v}v \\ M &= -k_{p,\phi}e_\phi - k_{d,\phi}r, \end{aligned}$$

where the k_p's and k_d's are gains proportional to the error and to the derivative of the error, in each channel, and choosing these is your main job. Start with small positive values and you will see your feedback system start to work! We say that this system is fully actuated, since you can independently control all the forces and the moment.

To see why the above rules work in a basic sense, consider the yaw direction only. With the feedback, the complete governing equation is

$$\begin{aligned} J\dot{r} &= -k_{p,\phi}(\phi - \phi_{desired}) - k_{d,\phi}r, \text{ or,} \\ J\ddot{\phi} + k_{d,\phi}\dot{\phi} + k_{p,\phi}\phi &= k_{p,\phi}\phi_{desired}. \end{aligned}$$

You see that this is a stable second-order oscillator whose parameters you get to tune, and that the steady-state solution is $\phi = \phi_{desired}$. For the yaw control problem, an important practical note is: don't let e_ϕ get outside of the range $[-\pi, \pi]$, or you will be turning the long way around. A couple of if/then's can make sure of this.

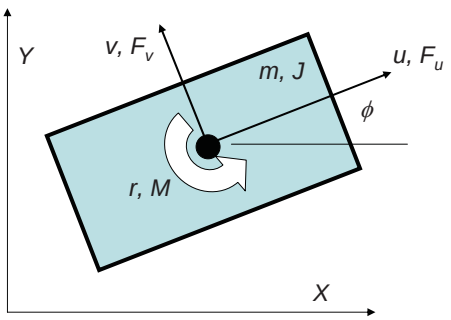
Provide a clear, marked listing of your control code (in one block please), time plots of all six channels of $\vec{s}$ through time, and an X, Y plot of the vehicle trajectory in the plane. These will show that the vehicle actually did what we wanted - to move from the origin to $\vec{s}_{desired}$.

3. Case 2: Develop a mission controller that maneuvers the vehicle from an all-zero starting condition to the state $[u, v, r, X, Y, \phi] = [1m/s, 0m/s, 0rad/s, 20m, 20m, \pi/2rad]$. In other words, the vehicle moves while rotating a quarter-turn, and passes through the desired X, Y location at speed and with no yaw rate.

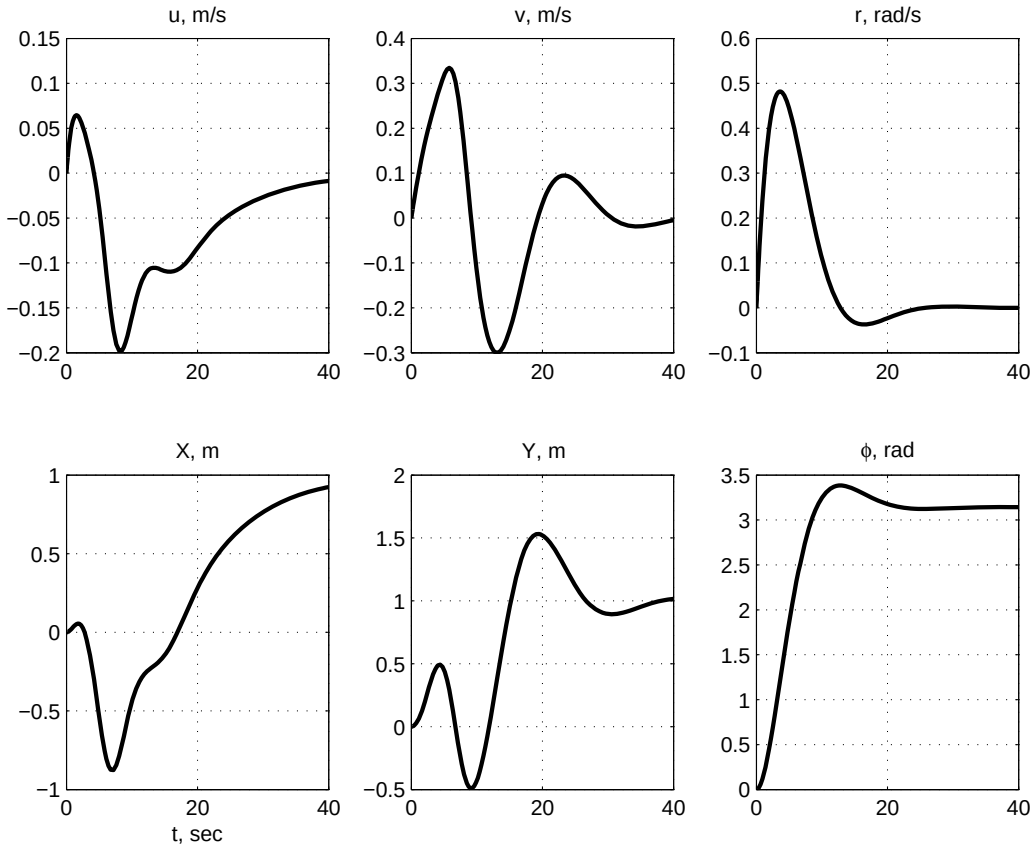
You can't use the same strategy as for Case 1, because the craft can only actuate sway and yaw together. We say that this system is *under-actuated*. It is not acceptable to create by hand a sequence of control actions that achieves the move, or to create an algorithm that uses specific times; they will never work in practice due to modeling errors and external disturbances. What I do recommend is that you use an intermediate *waypoint* on the way to the goal. This is a target X, Y that you go to first, so that the vehicle will then line up well with the goal. Airplanes are doing this sort of maneuver when they circle an airport in preparation for landing. Your mission procedure is to head for the intermediate waypoint until the craft arrives within a small radius of it (several meters, say), and then head to the goal. One well-chosen intermediate waypoint should be enough for this problem, but you could use more if you like.

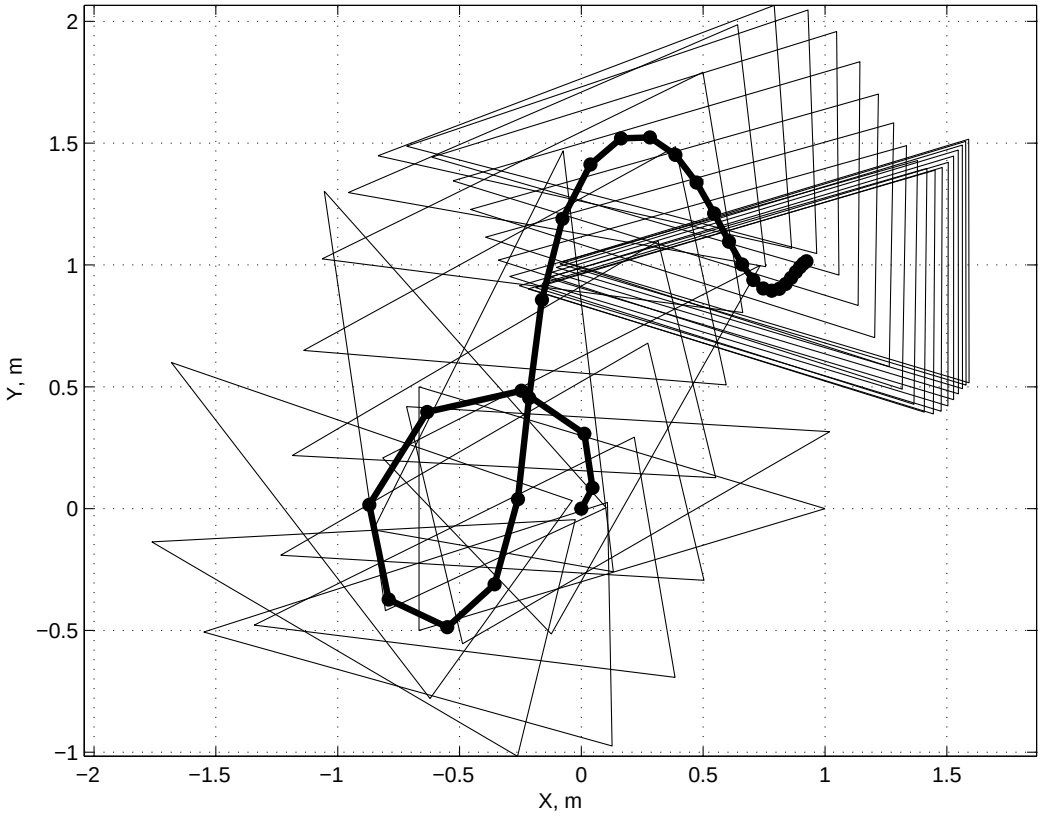
The low-level control strategy is also different from Case 1. Here, you will set $F_u = 1$ to move at a steady speed of $1m/s$, and then command δ so as to drive the vehicle to the desired X, Y. How to do this? Try setting $\phi_{desired} = atan2(Y - Y_{desired}, X - X_{desired})$. Remember that positive δ gives a negative moment.

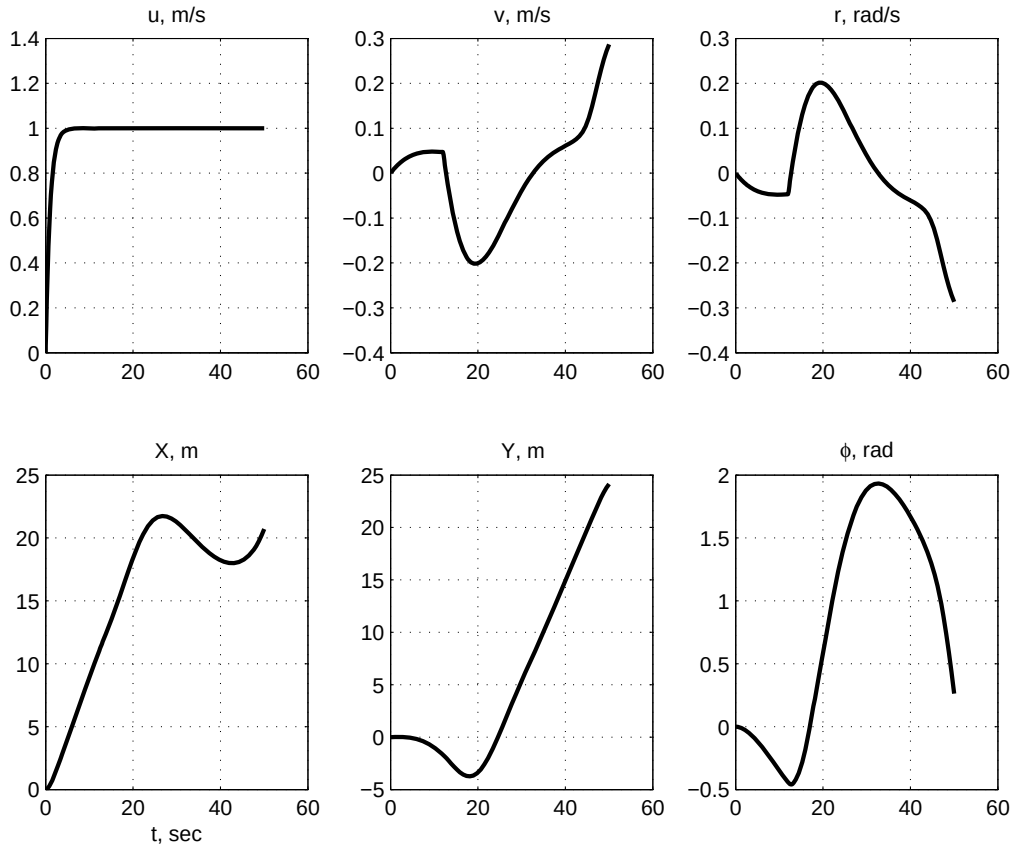
Show the same plots and listing here that you did for Case 1.

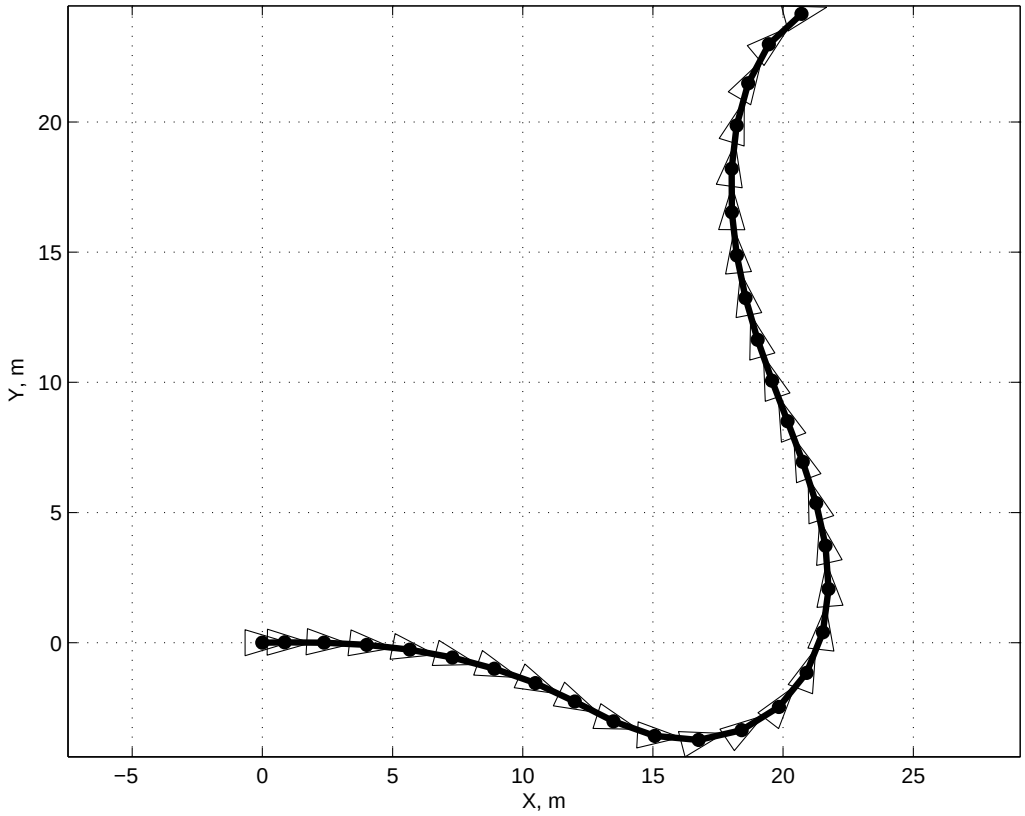


Plots and code are below; these generally play out the strategy described. In Case 1, one finds that setting the heading closed-loop behavior to be faster than that for X, Y may lead to better performance, because there is some coupling. Case 2 is effectively a point-and-go approach, with the critical use of an intermediate waypoint at $[X, Y] = [15, -5]m$ to line up the vehicle for the final approach to the goal.


u, m/s
v, m/s
r, rad/s
X, m
Y, m
ϕ, rad
t, sec


2
1.5
1
0.5
0
−0.5
−1
Y, m
−2
−1.5
−1
−0.5
0
0.5
1
1.5
X, m


u, m/s
v, m/s
r, rad/s
X, m
Y, m
ϕ, rad
t, sec


Y, m
X, m
20
15
10
5
0
−5
0
5
10
15
20
25

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Study dynamics and control of a hovercraft.
% FSH MIT 2.017 Oct 2009

clear all;
global captureRadius flag ; % for Case 2's intermediate waypoint

xCase = input('Case 1 or Case 2?  ');

captureRadius = 5 ; % used for switching waypoints in Case 2
flag = 0 ; % toggle for switching waypoints in Case 2 ;

if xCase == 1,
    tfinal = 40;
    [t,s] = ode45('hoverCraftDeriv1',tfinal,zeros(6,1));
else,
    tfinal = 50 ; % for Case 2
    [t,s] = ode45('hoverCraftDeriv2',tfinal,zeros(6,1));
end;

figure(1);clf;hold off;
for i = 1:6,
    subplot(2,3,i);
    plot(t,s(:,i),'LineWidth',2);
    grid;
end;
subplot(2,3,4);
xlabel('t, sec');
subplot(2,3,1);title('u, m/s');   subplot(2,3,2);title('v, m/s');
subplot(2,3,3);title('r, rad/s'); subplot(2,3,4);title('X, m');
subplot(2,3,5);title('Y, m');     subplot(2,3,6);title('\phi, rad');

tr = 0:tfinal/30:tfinal ; % regularly-spaced time
sr = spline(t,s',tr)';    % spline onto regularly-spaced time

figure(2);clf;hold off;
cphi = cos(sr(:,6)) ; sphi = sin(sr(:,6));
xbox = [1 -2/3 -2/3 1] ;
ybox = [0 1/2 -1/2 0];
center = [1 1 1 1];
plot(sr(:,4),sr(:,5),'k','LineWidth',3);
hold on;
for i = 1:length(cphi),
    plot(sr(i,4)*center + cphi(i)*xbox - sphi(i)*ybox, ...
```

```
        sr(i,5)*center + sphi(i)*xbox + cphi(i)*ybox);
    plot(sr(i,4),sr(i,5),'.k','MarkerSize',20);
end;
axis('equal');
grid;
xlabel('X, m');
ylabel('Y, m');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [sdot] = hoverCraftDeriv1(t,s);

u = s(1) ; v = s(2) ; r = s(3) ; X = s(4) ; Y = s(5) ; phi = s(6) ;
cphi = cos(phi) ;  sphi = sin(phi) ;

%%%%%%%%%%%%%%%%%%%%%%%%%%
% here is the control logic

desPhi = pi ;
errPhi = phi-desPhi ;
if errPhi < -pi, errPhi = errPhi + 2*pi ; end ;
if errPhi > pi, errPhi = errPhi - 2*pi ; end ;

desX = 1 ;
errX = X - desX ;
desY = 1 ;
errY = Y - desY ;

Fu = -.1*(errX*cphi - errY*sphi) ;
Fv = -.1*(errX*sphi + errY*cphi) - .3*v ;
M = -.1*(phi-desPhi)-.4*r ;

%%%%%%%%%%%%%%%%%%%%%%%%%%

udot = -u + Fu ;
vdot = Fv ;
rdot = M ;
Xdot = cphi*u - sphi*v ;
Ydot = sphi*u + cphi*v ;
phidot = r ;
sdot = [udot ; vdot ; rdot ; Xdot ; Ydot ; phidot] ;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [sdot] = hoverCraftDeriv2(t,s);
global flag captureRadius ;

u = s(1) ; v = s(2) ; r = s(3) ; X = s(4) ; Y = s(5) ; phi = s(6) ;
cphi = cos(phi) ;  sphi = sin(phi) ;

%%%%%%%%%%%%%%%%%%%%%%%%%%
% here is the control logic
if ~flag,
    waypointX = 15 ;
    waypointY = -5 ;
else,
    waypointX = 20 ;
    waypointY = 20 ;
end;

desPhi = atan2(waypointY-Y,waypointX-X);
errPhi = phi-desPhi ;
if errPhi < -pi, errPhi = errPhi + 2*pi ; end ;
if errPhi > pi, errPhi = errPhi - 2*pi ; end ;

del = 0.2*errPhi + r ;

if norm([X-waypointX Y-waypointY]) < captureRadius,
    flag = 1 ;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%

Fu = 1 ;
Fv = .2*del*Fu ;
M = -.2*del*Fu ;

udot = -u + Fu ;
vdot = Fv ;
rdot = M ;
Xdot = cphi*u - sphi*v ;
Ydot = sphi*u + cphi*v ;
phidot = r ;
sdot = [udot ; vdot ; rdot ; Xdot ; Ydot ; phidot] ;
```

%%

2.017J Design of Electromechanical Robotic Systems
Fall 2009