44 Feedback on a Highly Maneuverable Vessel

The directional behavior of a highly maneuverable vessel is modeled by

$$\frac{\theta}{\delta} = \frac{0.1s + 1}{s^2 + 0.6s - 0.05}$$

where the rudder deflection is δ and the heading is θ. You note that because the transfer function denominator has a negative coefficient, the system is unstable without a control system. Also, the numerator shows that the hydrodynamic lift on the rudder has a component that scales rudder position as well as one that scales rudder rotation *rate*.

1. Find the *poles* (the roots of the denominator polynomial) of this system. Are the roots complex (indicating an oscillatory behavior), or real (indicating two exponential growth/decay modes)? At what rate does the system go unstable - that is, over what time does the response grow by a factor of e? (You can confirm this with a simulation of the system behavior from some nonzero initial condition.)

   The poles are given by the roots of the denominator polynomial equation in s, i.e., the roots of $s^2 + 0.6s - 0.05 = 0$. These are -0.674 and 0.074, both real. The negative one pertains to a stable mode whereas the positive one indicates an unstable mode. The time scale (or time constant) of the unstable mode is 1/0.074 or about 13.5 seconds. This is the amount of time it takes for the output to increase by a factor of $e = 2.718$. It is confirmed in an impulse response, for example, after the transient response of the first, stable mode has died away.

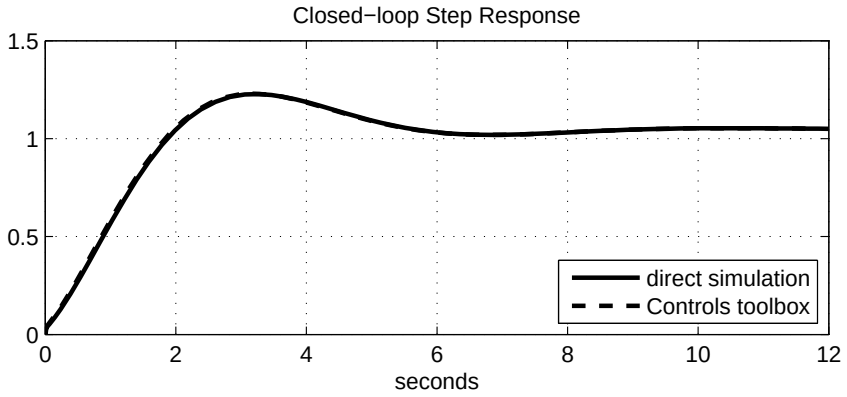
2. Design a proportional-derivative controller for the boat (PD), so as to achieve a closed-loop bandwidth of $\omega_c = 1$ rad/s and a damping ratio of $\zeta = 0.5$. The idea here is to choose k_p and k_d so as to put the closed-loop poles - that is, the roots of $1+P(s)C(s) = 0$ - at the specific locations $\omega_c\zeta \pm j\omega_c\sqrt{1-\zeta^2}$. The easiest way to carry this out is to write out the polynomial with k_d and k_p as free variables, and recognize that this same polynomial can be written as $s^2 + 2\zeta\omega_c s + \omega_c^2 = 0$. Some algebra will give you a set of two equations in two unknowns.

   The poles of the closed-loop system are to be the same as those of $s^2 + 2\zeta\omega_c s + \omega_c^2 = 0$; the closed-loop poles satisfy $1+P(s)C(s) = 0$, where $P(s)$ is the transfer function of the plant, and $C(s)$ is that of the controller. If $P(s) = n_p(s)/d_p(s)$ and $C(s) = n_c(s)/d_c(s)$ (numerators and denominators), then the condition $1 + P(s)C(s) = 0$ is the same as $d_P(s)d_C(s) + n_P(s)n_C(s) = 0$. Working this out, we obtain:

$$\begin{aligned} 0 &= s^2 + 0.6s - 0.05 + (k_p + k_d s)(0.1s + 1) \rightarrow \\ 2\zeta\omega_c &= \frac{0.6 + 0.1k_p + k_d}{1 + 0.1k_d} \quad \textit{and} \\ \omega_c^2 &= \frac{k_p - 0.05}{1 + 0.1k_d}. \end{aligned}$$

Setting ω_c and ζ to the desired values and solving this for the unknown variables we get $k_p = 1.082$ and $k_d = 0.324$.

3. Simulate the plant and controller, combined as a negative feedback system, in a step response, and show a plot. Note the step is applied to the commanded position; so you have to combine the plant and the controller, using $\delta = (k_p + sk_d)(\theta_{command} - \theta)$. Approximate the step function input as a ramp that rises over 0.02 seconds or longer (this will avoid artifacts of MATLAB's adaptive step-size algorithm). Does the response have the time scale and damping properties that you designed for?



The step response can be computed using calls to the MATLAB Control System Toolbox, but it is instructive to perform the calculations as a continuous-time simulation. After all, what is MATLAB doing?

The closed-loop dynamics are given as

$$\begin{aligned} (s^2 + 0.6s - 0.05)\theta(s) &= (0.1s + 1)\ \delta \\ \ddot{\theta} + 0.6\dot{\theta} - 0.05\theta &= \left(0.1\frac{d}{dt} + 1\right)\left(k_d\frac{d}{dt} + k_p\right)(r - \theta), \end{aligned}$$

where r is the reference heading. Collecting terms, we find

$$(1 + 0.1k_d)\ddot{\theta} + (0.6 + 0.1k_p + k_d)\dot{\theta} + (k_p - 0.05)\theta = 0.1k_d\ddot{r} + (0.1k_p + k_d)\dot{r} + k_p r.$$

The left-hand side looks like a regular mass-spring-damper, which will certainly have positive coefficients if the gains have been picked correctly. The right-hand side is more interesting because it involves the first and second derivatives of the commanded heading; for a step input in r, both of these derivatives are infinite! If r is a unit step, then $\dot{r}$ is a unit impulse, and $\ddot{r}$ is a (scaled) positive, then negative impulse.

A control action generated in the case of a step input would be impulsive here, and this is generally to be avoided in practice - unless some saturation occurs elsewhere in the system, or you know that the actuator can handle this kind of input. One solution is to never make r a step function, or even a ramp, so that $e = r - \theta$ always has two finite derivatives. A curve that is quadratic in time, for example, would suffice; so would the output of a separate second-order stable system, without any differentiation of the input (i.e., no factors of s in its transfer function numerator).

After all this, we still have to simulate the system one way or another. We extend a trick used in a prior question - the construction of an approximate impulse, of duration ϵ, and area one. This will be $\dot{r}$; its derivative will be $\ddot{r}$, and its integral will be r. Consider a very tall symmetric triangle with a peak at time $\epsilon/2$, and height $2/\epsilon$. The area under it is clearly one, and its derivative is $\pm 4/\epsilon^2$. With this technique, the plot shows that the step response computed via simulation matches that created by the Control System Toolbox.

Getting back to the question now, we expect a damped natural frequency in the closed loop of $\omega_d = \sqrt{1 - 0.5^2} = 0.866$, or a damped period of about 7.25 seconds. This is what we see in the figure. The damping ratio of 0.5 corresponds with an overshoot of around fifteen percent, and this is also confirmed by the step response, when the final value - which is not one - is taken into account.

4. After the transients of the step response have died out, what is the steady-state error?

   The final value can be computed by setting $s = 0$ in the closed-loop transfer function; this captures the effects at zero frequency, i.e., the steady-state after transients have died out. We have

$$\textit{final value} = \frac{k_p}{-0.05 + k_p} = 1.048.$$

   Clearly when k_p is large, this final value will be closer to one, as desired. As you know, however, gains cannot be turned up indiscriminately because the closed loop natural frequency will follow.

```
%-------------------------------------------------------------
% Feedback Control for Maneuvering

clear all;

global xepsilon kp kd n1 n0 d2 d1 d0;

% epsilon for computing step function; if we make this really small, the
% integrator will be forced to take tiny steps at the beginning; this
% conflicts with the defaults in MATLAB's adaptive stepsize routine.
```

```
xepsilon = .02 ;

% P numerator and denominator coefficients
n1 = .1 ;
n0 = 1 ;
d2 = 1 ;
d1 = .6 ;
d0 = -.05 ;

disp(sprintf('Open-loop poles: %g %g',roots([d2 d1 d0])));

% find the two gains for the specified crossover frequency and damping
% ratio
wc = 1 ;
z = .5 ;
mat = [2*z*wc*n1-n0 , -n1 ; wc^2*n1 , -n0] ;
vec = [d1 - 2*z*wc*d2 ; d0 - d2*wc^2] ;
vec = inv(mat)*vec ;
kd = vec(1) ;
kp = vec(2) ;
disp(sprintf('The gains needed are [kp kd] = [%g %g].', kp,kd));

% A.  here's the analysis using control system toolbox calls

% construct the plant and look at the unstable impulse response
sysP = tf([0 n1 n0],[d2 d1 d0]);
figure(1);clf;hold off;
impulse(sysP);

% close the loop and look at step response
sysC = tf([kd kp],[0 1]);
sysPC = sysC*sysP ;
sysCL = feedback(sysPC,1);
[y1,t1] = step(sysCL);

% B.  here's the simulation using regular integration

[t,y] = ode45('maneuveringFeedback_dxdt',[0 12],[0 0]) ;

figure(2);clf;hold off;
subplot('Position',[.2 .2 .6 .3]);
plot(t,y(:,2),t1,y1,'--','LineWidth',2);
grid;
title('Closed-loop Step Response');
```

```
xlabel('seconds');
legend('direct simulation','Controls toolbox',4);

%-----------------------------------------------------------------


%-----------------------------------------------------------------------
function [xdot] = maneuveringFeedback_dxdt(t,x) ;

global xepsilon kp kd n1 n0 d2 d1 d0 ;

% construct the artificial step input, with its two derivatives
if t < xepsilon/2,
    rddot = 4/xepsilon^2 ;
    rdot = 4/xepsilon^2*t ;
    r = 2/xepsilon^2*t^2 ;
elseif t < xepsilon,
    rddot = -4/xepsilon^2 ;
    rdot = 2/xepsilon - 4/xepsilon^2*(t-xepsilon/2) ;
    r = 0.5 + + 2/xepsilon*(t-xepsilon/2) - ...
        2/xepsilon^2*(t-xepsilon/2)^2;
else,
    r = 1 ;
    rdot = 0 ;
    rddot = 0 ;
end;

% (an option to the above is just to ignore the derivatives in the
% numerator!  For this, one would use r = 1, rdot = 0, rddot = 0 in
% the line below.  )

% compute the derivatives
xdot(1,1) = -(d1 + n1*kp + n0*kd)*x(1,1) - (n0*kp+d0)*x(2,1) + ...
        n1*kd*rddot + (n1*kp + n0*kd)*rdot + n0*kp*r ;
xdot(1,1) = xdot(1,1)/(d2 + n1*kd) ;

xdot(2,1) = x(1,1) ;

%-----------------------------------------------------------------------
```

2.017J Design of Electromechanical Robotic Systems
Fall 2009