A Design Study Approach to Classical Control

Randal W. Beard Timothy W. McLain
Brigham Young University

Updated: May 21, 2016

Homework A.12

(a) Modify the state feedback solution developed in Homework A.11 to add an integrator with anti-windup.

(b) Add a disturbance to the system and allow the system parameters to change up to 20%.

(c) Tune the integrator pole (and other gains if necessary) to get good tracking performance.

Solution

Step 1. The original state space equations are

$$
\begin{aligned}
\dot{x} &= \begin{pmatrix} 0 & 1.0000 \\ 0 & -0.667 \end{pmatrix} x + \begin{pmatrix} 0 \\ 66.667 \end{pmatrix} u \\
y &= \begin{pmatrix} 1 & 0 \end{pmatrix} x,
\end{aligned}
$$

therefore the augmented system is

$$A_1 = \begin{pmatrix} A & \mathbf{0} \\ -C & \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 & 1.0000 & 0 \\ 0 & -0.667 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$
$$B_1 = \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 \\ 66.667 \\ 0 \end{pmatrix}$$

Step 2. After the design in HW A.11, the closed loop poles were located at $p_{1,2} = -3.1191 \pm j3.1200$. We will add the integrator pole at $p_I = -5$. The new controllability matrix

$$\mathcal{C}_{A_1,B_1} = [B_1, A_1B_1, A_1^2B_1] = \begin{pmatrix} 0 & 63.6512 & -39.2993 \\ 63.6512 & -39.2993 & 24.2640 \\ 0 & 0 & 63.6512 \end{pmatrix}.$$

The determinant is $det(\mathcal{C}_{A_1,B_1}) = -2.5788e+05 \neq 0$, therefore the system is controllable.

The open loop characteristic polynomial

$$\Delta_{ol}(s) = \det(sI - A_1) = \det \begin{pmatrix} s & -1.0000 & 0 \\ 0 & s+0.667 & 0 \\ -1 & 0 & s \end{pmatrix} = s^3 + 0.6174s^2,$$

which implies that

$$\mathbf{a}_{A_1} = \begin{pmatrix} 0.6174, & 0, & 0 \end{pmatrix}$$
$$\mathcal{A}_{A_1} = \begin{pmatrix} 1 & 0.6174 & 0 \\ 0 & 1 & 0.6174 \\ 0 & 0 & 1 \end{pmatrix}.$$

The desired closed loop polynomial

$$\begin{aligned} \Delta_{cl}^d(s) &= (s + 3.1191 - j3.12)(s + 3.1191 + j3.12)(s+5) \\ &= s^3 + 12.7770s^2 + 69.1350s + 151.2500, \end{aligned}$$

which implies that

$$\boldsymbol{\alpha} = \begin{pmatrix} 12.7770, & 69.1350, & 151.2500 \end{pmatrix}.$$

The augmented gains are therefore given as

$$\begin{aligned} K_1 &= (\boldsymbol{\alpha} - \mathbf{a}_{A_1})\mathcal{A}_{A_1}^{-1}\mathcal{C}_{A_1,B_1}^{-1} \\ &= \begin{pmatrix} 1.0370, & 0.1817, & -2.2687 \end{pmatrix} \end{aligned}$$

Step 3. The feedback gains are therefore given by

$$\begin{aligned} K &= K_1(1:2) = \begin{pmatrix} 1.0370, & 0.1817 \end{pmatrix} \\ k_I &= K_1(3) = -2.2687 \end{aligned}$$

Alternatively, we could have used the following Matlab script

```
clear all

% actual system parameters
% initial conditions
P.theta0 = 0;
P.thetadot0 = 0;

% system parameters known to controller
P.m = 0.5;    % kg
P.ell = 0.3; % m
P.b = 0.01;  % N m s
P.g = 9.8;    % m/s^2

% sample rate
P.Ts = 0.01;

% dirty derivative gain
P.sigma = 0.05;

% equalibrium torque
P.theta_e = 0*pi/180;
P.tau_e   = P.m*P.g*P.ell/2*cos(P.theta_e);

% saturation constraint
P.tau_max = 1;
tau_max = P.tau_max-P.tau_e;

%----------------------
% state space design
A = [...
```

```
    0, 1;...
    0, -3*P.b/P.m/(P.ell^2);...
    ];
B = [0; 3/P.m/(P.ell^2) ];
C = [...
    1, 0;...
    ];
% form augmented system
A1 = [A, zeros(2,1); -C, 0];
B1 = [B; 0];

%  tuning parameters
tr = 0.4;
zeta = 0.707;
integrator_pole = -5;

% desired closed loop polynomial
wn = 2.2/tr;
% gains for pole locations
des_char_poly = conv([1,2*zeta*wn,wn^2],poly(integrator_pole));
des_poles = roots(des_char_poly);


% is the system controllable?
if rank(ctrb(A1,B1))≠3,
    disp('System Not Controllable');
else % if so, compute gains
    K1   = place(A1,B1,des_poles);
    P.K  = K1(1:2);
    P.ki = K1(3);
end
```

Matlab code that implements the associated controller listed below.

```
function tau=arm_ctrl(in,P)
    theta_r = in(1);
    theta   = in(2);
    t       = in(3);

    % use a digital differentiator to find thetadot
    persistent thetadot
    persistent theta_d1
    % reset persistent variables at start of simulation
    if t<P.Ts,
```

```
        thetadot    = 0;
        theta_d1    = 0;
    end
    thetadot = (2*P.sigma-P.Ts)/(2*P.sigma+P.Ts)*thetadot...
        + 2/(2*P.sigma+P.Ts)*(theta-theta_d1);
    theta_d1 = theta;

    % implement integrator
    error = theta_r - theta;
    persistent integrator
    persistent error_d1
    % reset persistent variables at start of simulation
    if t<P.Ts==1,
        integrator  = 0;
        error_d1    = 0;
    end
    integrator = integrator + (P.Ts/2)*(error+error_d1);
    error_d1 = error;


    % construct the state
    theta_e = 0;
    x = [theta-theta_e; thetadot];
    % compute equilibrium torque tau_e
    tau_e = P.m*P.g*(P.ell/2)*cos(theta);
    % compute the state feedback controller with integrator
    tau_tilde = - P.K*x - P.ki*integrator;
    % compute total torque
    tau_unsat = tau_e + tau_tilde;
    tau = sat( tau_unsat, P.tau_max);

    % integrator anti-windup
    if P.ki≠0,
       integrator = integrator + P.Ts/P.ki*(tau-tau_unsat);
    end
end

function out = sat(in,limit)
    if      in > limit,      out = limit;
    elseif in < -limit,      out = -limit;
    else                     out = in;
    end
end
```