# A Design Study Approach to Classical Control

Randal W. Beard Timothy W. McLain
Brigham Young University

Updated: May 21, 2016

## Homework B.12

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

**(b)** Add a constant input disturbance of 0.5 Newtons to the input of the plant 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

$$\dot{x} = \begin{pmatrix} 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \\ 0 & -2.2167 & -0.0500 & 0 \\ 0 & 24.5238 & 0.1020 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \\ 0.9524 \\ -1.9436 \end{pmatrix} u$$
$$y = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} x.$$

The integrator will only be on $z = x_1$, therefore

$$C_r = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}.$$

The the augmented system is therefore

$$A_1 = \begin{pmatrix} A & \mathbf{0} \\ -C_r & \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 \\ 0 & -2.2167 & -0.0500 & 0 & 0 \\ 0 & 24.5238 & 0.1020 & 0 & 0 \\ -1.0000 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$B_1 = \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0.9524 \\ -1.9436 \\ 0 \end{pmatrix}$$

**Step 2.** After the design in HW B.11, the closed loop poles were located at $p_{1,2} = -1.4140 \pm j1.4144$, $p_{3,4} = -0.8000 \pm j0.6000$. We will add the integrator pole at $p_I = -10$. The new controllability matrix

$$\begin{aligned} \mathcal{C}_{A_1,B_1} &= [B_1, A_1B_1, A_1^2B_1, A_1^3B_1, A_1^4B_1] \\ &= 1000 \begin{pmatrix} 0 & 0.0010 & -0.0000 & 0.0043 & -0.0004 \\ 0 & -0.0019 & 0.0001 & -0.0477 & 0.0028 \\ 0.0010 & -0.0000 & 0.0043 & -0.0004 & 0.1057 \\ -0.0019 & 0.0001 & -0.0477 & 0.0028 & -1.1691 \\ 0 & 0 & 0.0010 & -0.0000 & 0.0043 \end{pmatrix}. \end{aligned}$$

The determinant is nonzero, therefore the system is controllable.

The open loop characteristic polynomial

$$\begin{aligned} \Delta_{ol}(s) &= \det(sI - A_1) \\ &= \det \begin{pmatrix} s & 0 & -1.0000 & 0 & 0 \\ 0 & s & 0 & -1.0000 & 0 \\ 0 & 2.2167 & s+0.0500 & 0 & 0 \\ 0 & -24.5238 & -0.1020 & s & 0 \\ 1.0000 & 0 & 0 & 0 & s \end{pmatrix} \\ &= s^5 + 0.0500s^4 - 24.5238s^3 - s^2, \end{aligned}$$

which implies that

$$
\mathbf{a}_{A_1} = \begin{pmatrix} 0.0500, & -24.5238, & -1, & 0, & 0 \end{pmatrix}
$$

$$
\mathcal{A}_{A_1} = \begin{pmatrix} 1 & 0.05 & -24.5238 & -1 & 0 \\ 0 & 1 & 0.05 & -24.5238 & -1 \\ 0 & 0 & 1 & 0.05 & -24.5238 \\ 0 & 0 & 0 & 1 & 0.05 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.
$$

The desired closed loop polynomial

$$
\begin{aligned}
\Delta_{cl}^{d}(s) &= (s + 1.4140 - j1.4144)(s + 1.4140 + j1.4144)\ldots \\
&\quad (s + 0.8 - j0.6)(s + 0.8 + j0.6)(s + 10) \\
1.0000 &= s^5 + 18.2955s^4 + 117.3685s^3 + 397.6722s^2 + 576.9796s + 416.4551,
\end{aligned}
$$

which implies that

$$
\boldsymbol{\alpha} = \begin{pmatrix} 18.2955, & 117.3685, & 397.6722, & 576.9796, & 416.4551 \end{pmatrix}.
$$

The augmented gains are therefore given as

$$
\begin{aligned}
K_1 &= (\boldsymbol{\alpha} - \mathbf{a}_{A_1})\mathcal{A}_{A_1}^{-1}\mathcal{C}_{A_1,B_1}^{-1} \\
&= \begin{pmatrix} -29.4377, & -85.6531, & -21.4235, & -19.8345, & 21.2477 \end{pmatrix}
\end{aligned}
$$

**Step 3.** The feedback gains are therefore given by

$$
\begin{aligned}
K &= K_1(1:4) = \begin{pmatrix} -29.4377, & -85.6531, & -21.4235, & -19.8345 \end{pmatrix} \\
k_I &= K_1(5) = 21.2477
\end{aligned}
$$

Alternatively, we could have used the following Matlab script

```
% inverted Pendulum parameter file
clear all

% system parameters known to controller
P.m1 = 0.25;  % kg
P.m2 = 1;     % kg
P.ell = 0.5; % m
```

```
P.b = 0.05; % N m
P.g = 9.8; % m/s^2

% initial conditions
P.z0 = 0;
P.zdot0 = 0;
P.theta0 = 0;
P.thetadot0 = 0;

% input constraint
P.F_max = 5;

% sample rate for controller
P.Ts = 0.01;

% gain for dirty derivative
P.sigma = 0.05;


% state space design
A = [...
    0, 0, 1, 0;...
    0, 0, 0, 1;...
    0, -P.m1*P.g/P.m2, -P.b/P.m2, 0;...
    0, (P.m1+P.m2)*P.g/P.m2/P.ell, P.b/P.m2/P.ell, 0;...
];
B = [0; 0; 1/P.m2; -1/P.m2/P.ell ];
C = [...
    1, 0, 0, 0;...
    0, 1, 0, 0;...
    ];

% form augmented system
Cout = [1,0,0,0];
A1 = [A, zeros(4,1); -Cout, 0];
B1 = [B; 0];

% tunning parameters
tr_z = 1.5;  % rise time for position
tr_theta = .5; % rise time for angle
zeta_z   = 0.707; % damping ratio position
zeta_th  = 0.707; % damping ratio angle
integrator_pole = -10;

% compute gains
```

```
wn_th     = 2.2/tr_theta; % natural frequency for angle
wn_z      = 2.2/tr_z; % natural frequency for position
des_char_poly = conv(conv([1,2*zeta_z*wn_z,wn_z^2],...
                     [1,2*zeta_th*wn_th,wn_th^2]),...
                     poly(integrator_pole));
des_poles = roots(des_char_poly);

% is the system controllable?
if rank(ctrb(A1,B1))≠5,
    disp('System Not Controllable');
else % if so, compute gains
    K1   = place(A1,B1,des_poles);
    P.K  = K1(1:4);
    P.ki = K1(5);
end
```

Matlab code that implements the associated controller listed below.

```
function F=pendulum_ctrl(in,P)
    z_r   = in(1);
    z     = in(2);
    theta = in(3);
    t     = in(4);

    % use a digital differentiator to find zdot and thetadot
    persistent zdot
    persistent z_d1
    persistent thetadot
    persistent theta_d1
    % reset persistent variables at start of simulation
    if t<P.Ts,
        zdot        = 0;
        z_d1        = 0;
        thetadot    = 0;
        theta_d1    = 0;
    end
    zdot = (2*P.sigma-P.Ts)/(2*P.sigma+P.Ts)*zdot...
        + 2/(2*P.sigma+P.Ts)*(z-z_d1);
    thetadot = (2*P.sigma-P.Ts)/(2*P.sigma+P.Ts)*thetadot...
        + 2/(2*P.sigma+P.Ts)*(theta-theta_d1);
    z_d1 = z;
    theta_d1 = theta;

    % integrator
```

```
    error = z_r - z;
    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
    x = [z; theta; zdot; thetadot];
    % compute the state feedback controller
    F_unsat = -P.K*x - P.ki*integrator;
    F = sat( F_unsat, P.F_max);

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

end

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