A Design Study Approach to Classical Control

Randal W. Beard Timothy W. McLain
Brigham Young University

Updated: May 26, 2016

Homework A.14

(a) Modify your solution from HW A.13 so that the uncertainty parameter in `arm_dynamics.m` is $\alpha = 0.2$, representing 20% inaccuracy in the knowledge of the system parameters, and so that the input disturbance is 0.5 Newton-meters. Also, add noise to the output channel θ_m with standard deviation of 0.001. Before adding the disturbance observer, run the simulation and note that the controller is not robust to the large input disturbance.

(b) Add a disturbance observer to the controller, and verify that the steady state error in the estimator has been removed. Tune the system to get good response.

Solution

Matlab code used to design the observer based controller is shown below:

```
clear all

% 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;

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

%  tuning parameters for control
tr = 0.2;
zeta = 0.707;
integrator_pole = -1.0;

% 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

% observer design
% form augmented system for disturbance observer
A2 = [P.A, P.B; zeros(1,2), 0];
C2 = [P.C, 0];

% observer design
% tuning parameters for observer
wn_obs = 10;
zeta_obs = 1.707;
dist_pole = -5.5;

% desired observer poles
des_obsv_char_poly = conv([1,2*zeta*wn,wn^2],poly(dist_pole));
des_obsv_poles = roots(des_obsv_char_poly);

% is the system observable?
if rank(obsv(A2,C2))≠3,
    disp('System Not Observable');
else % if so, compute gains
    L2 = place(A2',C2',des_obsv_poles)';
    P.L = L2(1:2);
    P.Ld = L2(3);
end
```

Matlab code for the observer based control is shown below:

```
function out=arm_ctrl(in,P)
    theta_r = in(1);
    theta_m   = in(2);
    t       = in(3);

    % implement observer
    persistent xhat       % estimated state (for observer)
    persistent dhat       % estimate disturbance
    persistent tau        % delayed input (for observer)
    if t<P.Ts,
        xhat  = [0; 0];
        dhat  = 0;
        tau   = 0;
    end
```

```
15      % compute equilibrium torque tau_e
16      theta_e = 0;
17      theta = xhat(1);
18      tau_e = P.m*P.g*(P.ell/2)*cos(theta);
19      x_e = [theta_e; 0];
20      N = 10;
21      for i=1:N,
22          xhat = xhat + ...
23              P.Ts/N*(P.A*(xhat-x_e)+P.B*(tau-tau_e+dhat)...
24                      +P.L*(theta_m-P.C*xhat));
25          dhat = dhat + P.Ts/N*P.Ld*(theta_m-P.C*xhat);
26      end
27
28      % add integrator
29      error = theta_r - theta;
30      persistent integrator
31      persistent error_d1
32      % reset persistent variables at start of simulation
33      if t<P.Ts==1,
34          integrator  = 0;
35          error_d1    = 0;
36      end
37      integrator = integrator + (P.Ts/2)*(error+error_d1);
38      error_d1 = error;
39
40      % compute the state feedback controller
41      tau_unsat = tau_e-P.K*(xhat-x_e)-P.ki*integrator-dhat;
42      tau = sat( tau_unsat, P.tau_max);
43
44      % integrator anti-windup
45      if P.ki≠0,
46         integrator = integrator + P.Ts/P.ki*(tau-tau_unsat);
47      end
48
49      out = [tau; xhat];
50
51  end
52
53  function out = sat(in,limit)
54      if     in > limit,      out = limit;
55      elseif in < -limit,     out = -limit;
56      else                    out = in;
57      end
58  end
```

See the wiki for the complete solution.