A Design Study Approach to Classical Control

Randal W. Beard   Timothy W. McLain
Brigham Young University

Updated: May 9, 2016

Homework C.10

The objective of this problem is to implement the PID controller using Matlab code using only measured outputs of the system.

(a) Modify the system dynamics file so that the parameters J_s, J_p, k and b vary by up to 20% of their nominal value each time they are run (uncertainty parameter = 0.2).

(b) Rearrange the block diagram so that the controller is implemented as an m-function implemented at the sample rate of $T_s = 0.01$. Assume that the controller only has knowledge of the angles ϕ and θ as well as the reference angle ϕ_r.

(c) Implement the nested PID loops designed in Problems **??**.**??** using an m-function called satellite_ctrl.m. Use the dirty derivative gain of $\tau = 0.05$. Tune the integrator to remove the steady state error caused by the uncertain parameters.

Solution

The solution is on the wiki page associated with the book.

The Matlab code for the controller is shown below.

```
function tau=satellite_ctrl(in,P)
    phi_r   = in(1);
```

```
    phi     = in(2);
    theta   = in(3);
    t       = in(4);

    % set persistent flag to initialize integrators and
    % differentiators at the start of the simulation
    persistent flag
    if t<P.Ts,
        flag = 1;
    else
        flag = 0;
    end

    % compute the desired angled angle using the outer loop control
    phi_r = phi_r/P.k_DC_phi;
    theta_r = PID_phi(phi_r,phi,flag,P.kp_phi,P.ki_phi,P.kd_phi,...
                      P.Ts,P.sigma);
    % compute the force using the inner loop
    tau     = PD_th(theta_r,theta,flag,P.kp_th,P.kd_th,P.taumax,...
                    P.Ts,P.sigma);

end

%------------------------------------------------------------------------
% PID control for position
function u = PID_phi(phi_r,phi,flag,kp,ki,kd,Ts,sigma)
    % declare persistent variables
    persistent integrator
    persistent error_d1
    persistent phidot
    persistent phi_d1
    % reset persistent variables at start of simulation
    if flag==1,
        integrator  = 0;
        error_d1    = 0;
        phidot      = 0;
        phi_d1      = 0;
    end

    % compute the error
    error = phi_r-phi;

    % update derivative of phi
    phidot = (2*sigma-Ts)/(2*sigma+Ts)*phidot...
             + 2/(2*sigma+Ts)*(phi-phi_d1);
```

```
    % update delayed variables for next time through the loop
    phi_d1 = phi;

    % update integral of error
    integrator = integrator + (Ts/2)*(error+error_d1);
    % update delayed variables for next time through the loop
    error_d1 = error;

    % compute the pid control signal
    u = kp*error + ki*integrator -kd*phidot;

end


%-----------------------------------------------------------
% PID control for angle theta
function u = PD_th(theta_r,theta,flag,kp,kd,limit,Ts,sigma)
    % declare persistent variables
    persistent thetadot
    persistent theta_d1
    % reset persistent variables at start of simulation
    if flag==1,
        thetadot    = 0;
        theta_d1    = 0;
    end

    % compute the error
    error = theta_r-theta;
    % update derivative of y
    thetadot = (2*sigma-Ts)/(2*sigma+Ts)*thetadot...
               + 2/(2*sigma+Ts)*(theta-theta_d1);
    % update delayed variables for next time through the loop
    theta_d1 = theta;

    % compute the pid control signal
    u_unsat = kp*error - kd*thetadot;
    u = sat(u_unsat,limit);

end

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

```
    else                                        out = in;
    end
end
```