A Design Study Approach to Classical Control

Randal W. Beard Timothy W. McLain
Brigham Young University

Updated: August 8, 2016

Homework A.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 vary by up to 20% 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 $T_s = 0.01$. The controller should only assume knowledge of the angle θ and the reference angle θ_r.

(c) Implement the PID controller designed in Problems A.8 using an m-function called arm_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=arm_ctrl(in,P)
    theta_c = in(1);
    theta   = in(2);
```

```
    t       = in(3);

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

    % compute equilibrium torque tau_e
    tau_e = P.m*P.g*(P.ell/2)*cos(theta);
    % compute the linearized torque using PID
    tau_tilde = PID_th(theta_c,theta,flag,P.kp,P.ki,P.kd,...
                       P.tau_max,P.Ts,P.sigma);
    % compute total torque
    tau = tau_e + tau_tilde;

end

%------------------------------------------------------------
% PID control for angle theta
function u = PID_th(theta_c,theta,flag,kp,ki,kd,limit,Ts,sigma)
    % declare persistent variables
    persistent integrator
    persistent thetadot
    persistent error_d1
    persistent theta_d1
    % reset persistent variables at start of simulation
    if flag==1,
        integrator  = 0;
        thetadot    = 0;
        error_d1    = 0;
        theta_d1    = 0;
    end

    % compute the error
    error = theta_c-theta;
    % update derivative of y
    thetadot = (2*sigma-Ts)/(2*sigma+Ts)*thetadot...
               + 2/(2*sigma+Ts)*(theta-theta_d1);
    % update integral of error
    if abs(thetadot)<0.05,
        integrator = integrator + (Ts/2)*(error+error_d1);
```

```
    end
    % update delayed variables for next time through the loop
    error_d1 = error;
    theta_d1 = theta;

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

    % integrator anti-windup
    if ki≠0,
        integrator = integrator + Ts/ki*(u-u_unsat);
    end
end

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