Chapter 8

PID Controller for Process Control

PID (proportional-integral-derivative) controller is the most widely used controller in industry [1]. PID controller has three tunable control parameters K_P: proportional gian, K_I: integral gain, and K_D: derivative gain. These three parameters can be conveniently tuned for many industrial control systems, even without knowledge of the plant model [12]. PID controller calculates the weighted sum of the instantaneous error $E(s) = R(s) - Y(s) = L\{e(t)\}$, error rate $sE(s) = L\{\frac{de(t)}{dt}\}$, and accumulated error $\frac{1}{s}E(s) = L\{\int e(t)dt\}$, using respective gains K_p, K_d, K_I, and use this summation as the control input to the actuator. The control input $u_{PID}(t)$ of a PID controller is as follows.

$$U_{PID}(s) = K_P e(s) + K_I \frac{E(s)}{s} + K_D s E(s) \tag{8.1}$$

PID controller is alternatively presented as follows

$$U_{PID}(s) = K_P \left\{ E(s) + \frac{1}{T_i}\frac{E(s)}{s} + T_d s E(s) \right\} \tag{8.2}$$

where $T_i = \frac{K_P}{K_I}$, and $T_d = \frac{K_D}{K_P}$ are the time constants of the integrator and differentiator. A plant controlled by a PID controller is shown in Fig.8.1.

8.1 Proportional Controller

The proportional controller commands the actuator to respond to the instantaneous error $e(t) = r(t) - y(t)$. It exerts a positive command when $r(t) > y(t)$, a negative command when $r(t) < y(t)$, and a zero command

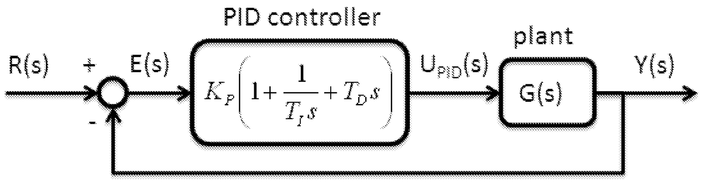


Figure 8.1: A plant controlled by a PID controller

when $r(t) = y(t)$. P-control action is proportional to the error with the constant of proportionality K_P. If K_P is too high the P-controller is very sensitive and it aggressively responds even for very small errors. On the other hand, if K_P is very small P-controller is very inactive, and only responds for larger errors. Both these behaviors are unacceptable, and it is necessary to adjust K_P to an appropriate strength so that P-controller responds properly to the error. The P-controller is primarily responsible for the responsiveness of the PID controller.

8.2 Integral Controller

The integral controller keeps accumulating the error, and generates a control action proportional to the value of accumulated error at any moment. Therefore, even when there is no error $y(t) = r(t)$ at time t, the integral controller produces a control action if the net accumulated error at time t is not zero. If the net accumulated error when $e(t) = 0$ is positive, the integral command will drive the response to overshoot. On the other hand, if the net accumulated error is negative, it causes the response to undershoot. Both these behaviors are apparently unsatisfactory because it reduces the stability of the control system. However, I-controller is required to eliminate steady state error in presence of persistent disturbances shown in Fig. 8.2. In order to maintain stability, I-controller is restricted error accumulation only within predefine limits, which is called the anti-windup protection of the I-controller.
Lets assume that $e(t) = y(t) - r(t) = 0$ at time t. Then, $u_P(t) = K_P e(t) = 0$, and $u(t) = u_P(t) + d = d$. This nonzero control action will deflect the plant away from the desired state $y(t) = r(t)$. Therefore, the plant will not stabilize at zero error with P-controller alone in view of a persistent disturbance. The response deflects till the stable condition $u(t) = u_P(t) + d = 0$ is reached with some error. This error at stable response is $e(t) = -d/K_P$. In order to drive the response back to the reference value, an I-controller can be introduced as shown in Fig.8.3. Under PI control, response will stabilize

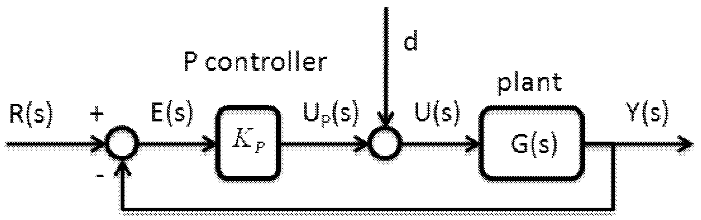


Figure 8.2: A disturbance in a control system controlled by a P controller

when $u_P(t) + u_I(t) + d = K_p e(t) + K_I \int e(t)dt + d = 0$, in which case $e(t) = 0$ and $\int e(t)dt = -d$. Therefore, zero steady state error can be achieved with the I-controller, which accumulated a finite error and counter the action of the disturbance.

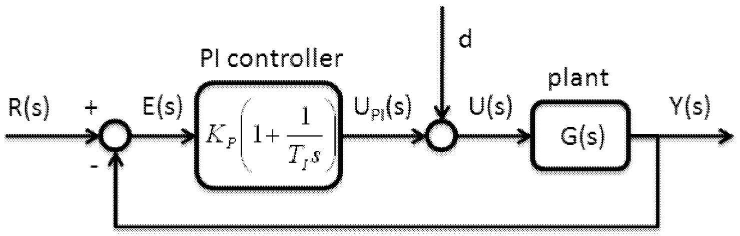


Figure 8.3: A disturbance in a control system controlled by a PI controller

8.2.1 Example

Figure 8.4 shows a P-controller in a robot arm position control system. The P-controller generates the torque command $u_p(t) = K_P e(t)$ for the motor, which adjusts the arm position $\theta(t)$.

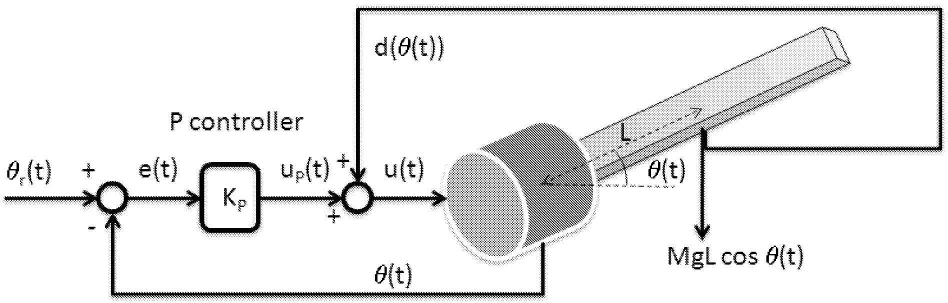


Figure 8.4: P-controller for robot arm position control

The weight of the arm creates a clockwise (-ve) torque, and this torque acts as a disturbance $d(\theta(t))$ as shown in Fig.8.4. Lets assume $K_p = 25$, $M = 3$kg, $L = 1$m, $g = 9.8\text{ms}^{-2}$, and reference position $\theta_r = 1$rad. When the arm is at position $\theta(t)$, net torque on the shaft to turn the arm is $u_P(t) + d(\theta(t))$, where $u_P(t) = K_P(\theta_r - \theta(t))$, and $d(\theta(t)) = -MgL\cos(\theta(t))$. When the arm stabilizes at some position θ_s, the net torque on the shaft is zero as follows.

$$\begin{aligned} K_p(\theta_r - \theta_s) - MgL\cos\theta_s &= 0 \\ \theta_s + \frac{MgL\cos\theta_s}{K_p} - \theta_r &= 0 \\ \theta_s + \frac{3 \times 9.8 \times 1\cos\theta_s}{25} - 1 &= 0 \\ \theta_s + 0.85\cos\theta_s - 1 &= 0 \end{aligned} \tag{8.3}$$

The numerical solution of (8.3) is $\theta_s = 0.15$rad. Therefore, the arm stabilizes leaving a steady state error of $1 - 0.15 = 0.85$rad=48.9^o. In this position, the clockwise torque due to the weight of the arm is countered by the proportional controller by exerting an equal and opposite (anticlockwise) torque, for which it needs this error of 48.9^o.
In order to stabilize the arm with zero steady state error an I-controller can be introduced to work together with the existing P-controller. However, I-controller should be carefully introduced and used intermittently during the control period as otherwise it could potentially reduces relative stability of the system. Generally, due to error accumulation, I-controller is enabled intermittently during certain parts of the response. Lets assume that the control system is turned on at $t = 0$ with a step reference input, while the response remains at zero. If the integral controller is also turned on at $t = 0$, it will start accumulating a +ve error ($e(t) = r(t) - y(t) > 0$) till the response reaches the reference input.At this state, the integrator will have been charged with a huge accumulated error. This accumulated error will continue to drive the response causing a substantial overshoot ($y(t) > r(t)$). Therefore, I-controller should be used carefully.

8.3 Derivative Controller

Derivative controller generates a control command proportional to the rate of change of error $\dot{e}(t) = \dot{r}(t) - \dot{y}(t)$ at any moment. Assuming that the reference is held constant, i.e. $\dot{r}(t) = 0$ the D-control command is $u_D(t) = -K_D\dot{y}(t)$, which is a command against the motion. Therefore, the D-controller acts against the motion, similar to the damper behavior in the mass-spring-damper system described earlier. The D-control action is dominant when the response undergoes rapid or sudden changes, which shows that the D-controller improves stability of the control system. At steady state, when the response stabilizes, the D-controller ceases to exert any control effort.

8.4 Tuning PID Controller

Tunning PID controller is the process of determining the proper match between the gains K_P, K_I, K_D of the controller. As shown above, K_P makes the system responsive to errors, however, a bigger value of K_P will make the system too sensitive, and responsive to even noise in the control loop. K_I reduces the steady state error, however, it increases overshoot and reduces stability. K_D, stabilize the system, and in the same time it slows down the response. In order to realize desirable response the three individual controllers have to be properly adjusted [1],[10]. There are three main techniques for gain tuning of PID controllers namely, Zeigler-Nichols, Cohen-Coon, and ITAE based methods. A performance comparision of these methods are available in [16]

8.4.1 Ziegler-Nichols Method

Ziegler-Nichols method of PID tuning [18] is a historical work by Zeigler and Nichols, and it is well known even today. This method assumes that the plant is of the form $Ke^{Ts}/(s+a)$, and the controller gains are tuned heuristically. This method can be implemented in two ways. If the frequency response of the plant is known and it shows that there is a crossover frequency f_u as shown in Fig.8.5.

If frequency response of the plant is not known, a P-controller is introduced to control the plant while the response is monitored as K_P increases. When the response shows stable oscillations the value of the proportional gain is equalto the gain margin GM and the frequency of oscillation is the crossover frequency. Either by way of frequency response or by this P-controller test it is possible to obtain the gain margin GM and crossover frequency f_{co}. The crossover period $T_{co} = 1/f_{co}$. With GM and T_{co} PID controller gains can be determined using Table 8.1 [18].

According to the Table 8.1, K_P is weaken when controller configuration changes from P to PI. This way, system bandwidth is slightly reduced to maintain a healthy gain margin while I-control action reduces phase margin. Without this modification, I-controller could reduce more of the stability

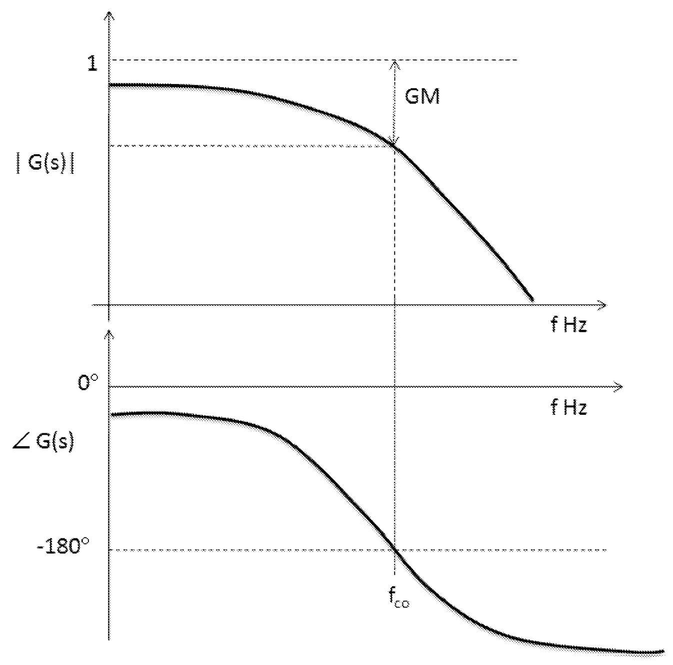


Figure 8.5: Frequency response of a plant with a crossover frequency and gain margin

margin. However, when D-controller is also introduced, the additional phase lead of the D-controller allows to increase the bandwidth again. This gain tunning method is a heuristic method of tuning the controller to achieve a 30^0 phase margin and a gain margin of 2. This method is not useful in designing a PD controller as it will over-damp the response and that is why it is not included in the Table 8.1. This method is not applicable if the plant is stable for all frequencies in which case there is no crossover frequency. The Ziegler-Nichols method provides few different controller settings as shown in the Table above, and each controller can be tested to verify which one is most appropriate for a given plant. Sometimes, Zeigler-Nichols method may not be able to tune PID controller for some planst at all [10].

8.5 Cohen-Coon Method

A decade after Zeigler and Nochols, another PID tuning method was proposed by G. H. Cohen and G. A. Coon [5]. This method is also based on a delayed first order rise, and the method tunes the PID gains to achieve quarter-amplitude damping, i.e. each peak in the transient is one quarter of the immediate preceding peak. The open loop plant is assumed to be

controller	K_P	K_I	K_D	T_I	T_D
P	$0.5GM$	-	-	-	-
PI	$0.45GM$	$1.2\frac{K_P}{T_{co}}$	-	$0.8T_{co}$	-
PID	$0.6GM$	$2\frac{K_P}{T_{co}}$	$0.125\ K_P T_{co}$	$0.5\ T_{co}$	$0.125\ T_{co}$
Pessen	$0.7GM$	$2.5\frac{K_P}{T_{co}}$	$0.15\ K_P T_{co}$	$0.4\ T_{co}$	$0.15\ T_{co}$
wos	$0.33GM$	$2\frac{K_P}{T_{co}}$	$0.33\ K_P T_{co}$	$0.5\ T_{co}$	$0.33\ T_{co}$
woos	$0.2GM$	$2\frac{K_P}{T_{co}}$	$0.33\ K_P T_{co}$	$0.5\ T_{co}$	$0.33\ T_{co}$

Table 8.1: Ziegler-Nichols method, Pessen Integral rule, with and without overshoot tunning of PID controller

$Ke^{-\tau_d s}/(\tau s+1)$, which leads to unit step response $K[1-e^{-\frac{1}{\tau}(t-\tau_d)}]$, where τ_d is the apparent dead time, τ is apparent time constant, and K is the DC gain as shown in Fig.8.6. If the unit step response of the open loop plant is available, then, K, τ_d and τ can be determined. These three parameters are used to tune PID gains as shown in Table 8.2.

8.6 Integral-based Method

Minimization of error integral with respect to controller gains is employed in this method [2], where the Integral of Time-weighted Absolute Error (ITAE) $\int t|e(t)|dt$, is the basis for optimally determininig the PID gains as follows.

$$\begin{aligned} K_P &= \frac{1}{K}\Gamma \\ \frac{1}{T_I} &= \frac{1}{\tau_a}\Gamma \\ T_D &= \tau_a\Gamma \end{aligned} \tag{8.4}$$

where K is the DC gain shown in Fig.8.6, and $\Gamma = \sigma_1\left(\frac{\tau_{ad}}{\tau_a}\right)^{\sigma_2}$ is calculated for a given controller type (PI, PD, PID etc.) referring the Table 8.3 and

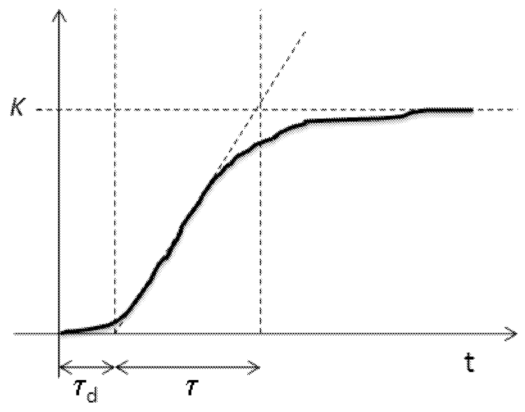


Figure 8.6: Experimental unit step response of a plant

controller	K_P	T_I	T_D
P	$\frac{\tau}{DCG\tau_d}\{1+\frac{\tau_d}{3\tau}\}$	-	-
PI	$\frac{\tau}{DCG\tau_d}\{\frac{9}{10}+\frac{\tau_d}{12\tau}\}$	$\tau_d\{\frac{30+3\tau_d/\tau}{9+20\tau_d/\tau}\}$	-
PD	$\frac{\tau}{DCG\tau_d}\{\frac{5}{4}+\frac{\tau_d}{6\tau}\}$	-	$\tau_d\{\frac{6-2\tau_d/\tau}{22+3\tau_d/\tau}\}$
PID	$\frac{\tau}{DCG\tau_d}\{\frac{4}{3}+\frac{\tau_d}{4\tau}\}$	$\tau_d\{\frac{32+6\tau_d/\tau}{13+8\tau_d/\tau}\}$	$\tau_d\{\frac{4}{11+2\tau_d/\tau}\}$

Table 8.2: Cohen-Coon PID gain tuning table

step response attributes τ, and τ_d.

8.7 Example

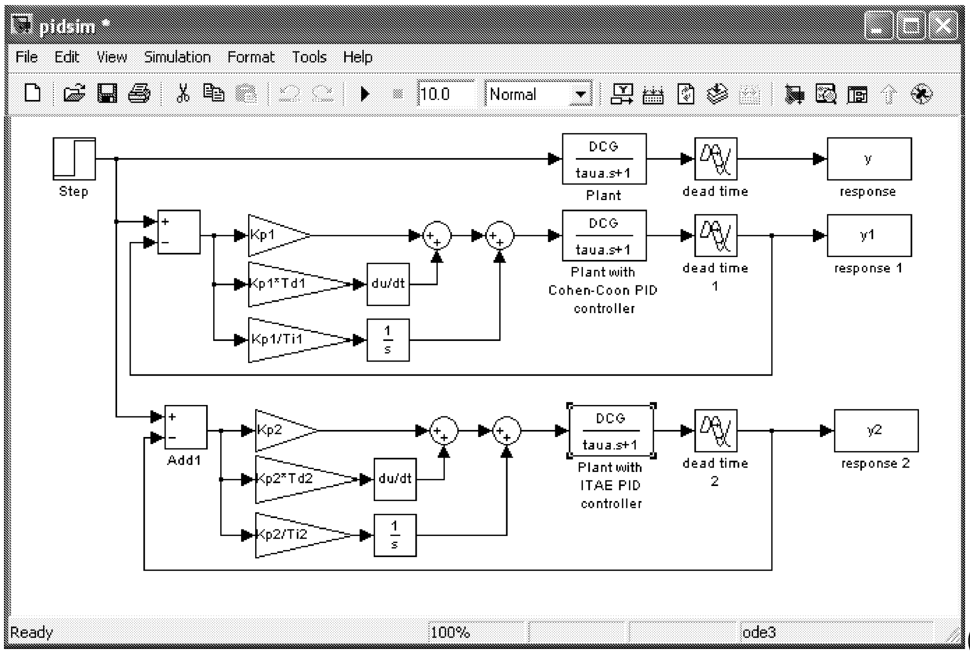
A plant shows an apparent dead time $\tau_d = 0.6$s, and apparent time constant $\tau = 2$s, and a DC gain $K = 0.7$. Using Cohen-Coon PID tunning method in Table. 8.2 PID gains are calculated as $Kp1 = 6.71$, $Ti1 = 1.32$, and $Kd1 = 0.21$. Similarly, the ITAE PID gains are calculated using Table.8.3 as $Kp2 = 4.24$, $Ti2 = 2.05$, and $Kd2 = 0.16$. The following `MatLab` code will simulate this plant under the control of Cohen-Coon and ITAE PID controller. The plant is modelled as a `Simulink` block, and it is run from the `MatLab` code. The `Simulink` model of the open loop plant $G(s)$, Cohen-Coon

controller	Γ_D σ_1, σ_2	Γ_I σ_1, σ_2	Γ_D σ_1, σ_2
P	0.490,-1.084	-	-
PI	0.859,-0.977	0.674, -0.680	-
PID	1.357,-0.947	0.842, -0.738	0.381,0.995

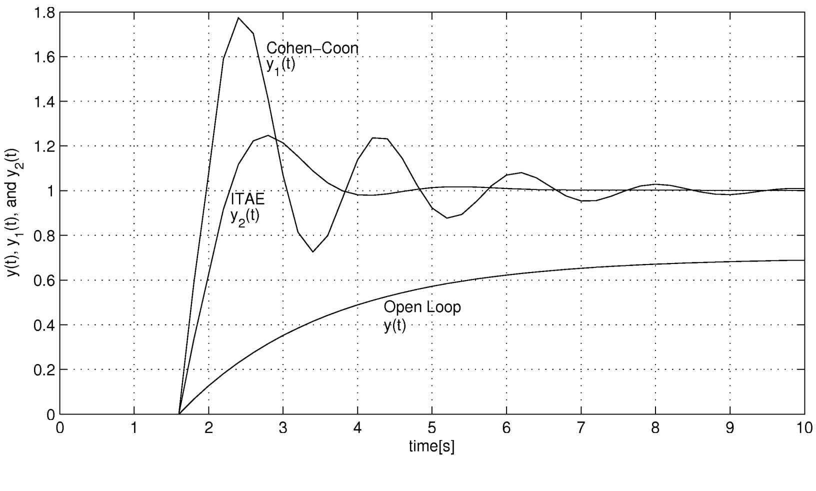
Table 8.3: ITAE based PID gain tuning table

PID controller, and ITAE PID controller are shown in Fig.8.7(a), whereas Fig.8.7(b) shows the unit step response under Cohen-Coon and ITAE PID controllers.

```
% PID Tuning
DCG=0.7; tau=2; taud=0.6; % OL plant response

% Cohen-Coon PID
Kp1=(tau/(DCG*taud)) * ( 4/3 + taud/(4*tau))
Ti1=taud*((32+6*taud/tau)/(13+8*taud/tau))
Td1=taud*(4/(11+2*taud/tau))

% ITAE PID
sp1=1.357; sp2=-0.947;
si1=0.842; si2=-0.738;
sd1=0.381; sd2=0.995;
r=tauad/taua;


Kp2=sp1*r^sp2
Ti2=si1*r^si2
Td2=sd1*r^sd2

sim pidsim; % calls simulink block

plot(tout,y,'r-.',tout,y1,'b--',tout,y2,'k'); % step response
xlabel('time[s]'); ylabel('y(t), y1(t), and y2(t)'); grid on;
```



(a)



(b)

Figure 8.7: (a)`MatLab Simulink` block of the open loop plant, Cohen-Coon PID controller, and ITAE PID controller, (b) step response of the open loop plant, Cohen-Coon PID, and ITAE PID

8.8 Summary

A PID controller can be quite easily designed and employed for industrial plant control even without any knowledge about plant model and behavior. PID controller tuning methods such as Zeigler-Nochols, Cohen-coon, and Integral-based (such as ITAE) are widely used in industrial plant control. These methods try to minimize oscillations by way of adjusting the three parameters K_P, T_I, and T_D of the PID controller. Zeigler-Nicholas method can be used if the open loop response show sustained oscillations for some gain, whereas Cohen-coon and Integral-based methods can be used if the open loop response shows a delayed first order rise. These methods of tunning PID gains generally provide satisfactory performance, however, the best performance for a given plant might not be directly achieved by these methods. In such situations, more interactive tunning is required.