The Design of the PID Controller

Robert Paz
New Mexico State University
32 PUBLICATIONS **178** CITATIONS

The Design of the PID Controller

Robert A. Paz
Klipsch School of Electrical and Computer Engineering

June 12, 2001

Abstract

The PID controller enjoys the honor of being the most commonly used dynamic control technique. Over 85% of all dynamic (low-level) controllers are of the PID variety. The purpose of this report is to provide a brief overview of the PID controller.

1 Introduction

In the control of dynamic systems, no controller has enjoyed both the success and the failure of the PID control. Of all control design techniques, the PID controller is the most widely used. Over 85% of all dynamic controllers are of the PID variety. There is actually a great variety of types and design methods for the PID controller.

What is a PID controller? The acronym PID stands for **P**roportio-**I**ntegro-**D**ifferential control. Each of these, the **P** , the **I** and the **D** are terms in a control algorithm, and each has a special purpose. Sometimes certain of the terms are left out because they are not needed in the control design. This is possible to have a PI, PD or just a P control. It is very rare to have a ID control.

2 The Problem Setup

The standard PID control configuration is as shown. It is also sometimes called the "PID parameter form."

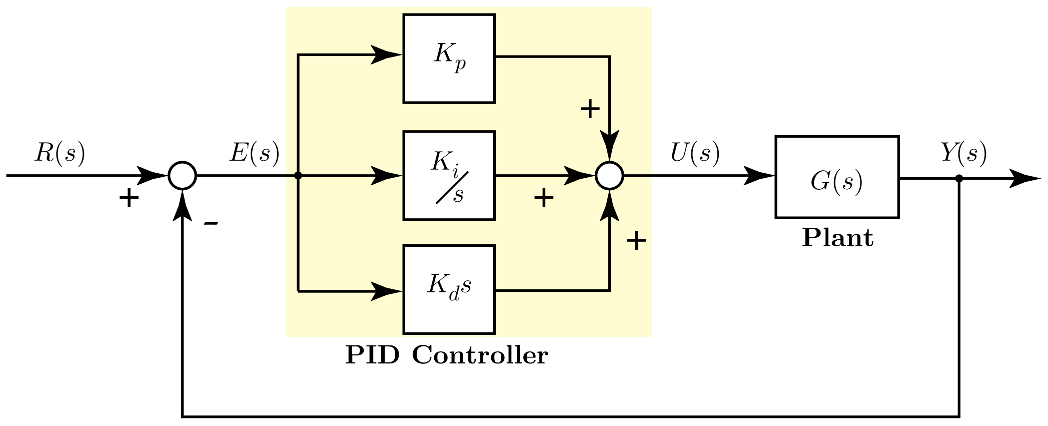

Figure 1: **PID Controlled System**

In this configuration, the control signal $u(t)$ is the sum of three terms. Each of these terms is a function of the tracking error $e(t)$. The term K_p indicates that this term is proportional to the error. The term K_i/s is an integral term, and the term $K_d s$ is a derivative term. Each of the terms works "independently"[1] of the other.

2.1 Interpretation of the Terms

We now consider each of the terms, assuming that the others are zero. With $K_i = K_d = 0$, we simply have $u(t) = K_p e(t)$. Thus at any instant in time, the control is proportional to the error. It is a function of the *present value* of the error. The larger the error, the larger the control signal. One way to look at this term is that the farther away from the desired point we are, the harder we try. As we get closer, we don't try quite as hard. If we are right on the target, we stop trying. As can be seen by this analogy, when we are close to the target, the control essentially does nothing. Thus, if the system drifts a bit from the target, the control does almost nothing to bring it back. Thus enters the integral term.

Assuming now that $K_p = K_d = 0$, we simply have

$$u(t) = K_i \int_0^t e(\tau)\, d\tau$$

The addition of this integral makes the open-loop forward path of Type I. Thus, the system, if stable, is guaranteed to have zero steady-state error to a step input. This can also be viewed as an application of the *internal model principle*. If $e(t)$ is non-zero for any length of time (for example, positive), the control signal gets larger and large as time goes on. It thus forces the plant to react in the event that the plant output starts to drift. We can think of the integral term as an accumulation of the *past values* of the error. It is not uncommon for the integral gain to be related to the proportional gain by

$$K_i = \frac{K_p}{\tau_i}$$

where τ_i is the *integral time*. Generally, by itself, the I term is not used. It is more commonly used with the P term to give a PI control.. The I term tends to slow the system reactions down. In order to speed up the system responses, we add the derivative term.

Assuming now that $K_p = K_i = 0$, we have

$$u(t) = K_d \frac{de(t)}{dt}$$

that is that the control is based on the rate of change of the error. The more quickly the error responds, the larger the control effort. This changing of the error indicates where the error is going. We can thus think of the derivative term as being a function

[1]This is not exactly true since the whole thing operates in the context of a closed-loop. However, at any instant in time, this is true and makes working with the PID controller much easier than other controller designs.

of the *future values* of the error. In general, a true differentiator is not available. This is because true differentiation is a “wide-band” process, i.e., the gain of this term increases linearly with frequency. It is a non-causal process. It tends to amplify high-frequency noises. In addition, if the error undergoes a sharp transition, for example, when a step input is applied, the derivative skyrockets, requiring an unreasonably large control effort. Generally, the control signal will saturate all amplifiers etc. To deal with the impractical nature of the D term, it is not uncommon to use the modified derivative

$$\dot{x}(t) = \frac{1}{\tau_1}\left(e(t) - x(t)\right), \quad u(t) = K_d \dot{x}(t) \tag{1.1}$$

where τ_1 limits the bandwidth of the differentiator. In terms of Laplace transform variables, we have

$$\mathcal{L}\{\dot{e}(t)\} = sE(s) \approx \left(\frac{s}{\tau_1 s + 1}\right) E(s) \tag{1.2}$$

Thus, for $\tau_1 - small$, the approximation is better. This corresponds to the “pole” of the differentiator getting larger. It is common to let

$$\tau_1 = \frac{K_d}{N K_p} \tag{1.3}$$

where N is typically in the range from 10 to 20. This matter of the differentiator could also be viewed using the Taylor expansion of $e(t)$.

An alternate form of the PID controller is called the “non-interacting form.” In this form, we have $K_i = \frac{K_p}{\tau_i}$, and $K_d = K_p \tau_d$. In this case, we can factor K_p out of the overall controller

$$u(t) = K_p\left(e(t) + \frac{1}{\tau_i}\int_0^t e(\tau)\,d\tau + \tau_d \frac{de(t)}{dt}\right) \tag{1.4}$$

The PID control thus considers past, present and future values of the error in assigning its control value. This partly explains its success in usage.

2.2 Setpoint Weighting

It is common for the closed-loop system to track a constant reference input. In this case, the input is called a *setpoint.* This being the case, it is often advantageous to consider an alteration of the overall control law for the sake of this problem. We noted, for instance, that it is not particularly good to differentiate step changes in the error signal. Changing the configuration will give basically the same behavior, without having to do such things. The setpoint weighted PID is thus a generalization of the PID, and has

$$u(t) = K_p e_p(t) + K_i \int_0^t e_i(\tau)\,d\tau + K_d \frac{de_d(t)}{dt} \tag{1.5}$$

where

$$e_p = a_p r(t) - y(t), \quad e_i(t) = r(t) - y(t), \quad e_d(t) = a_d r(t) - y(t) \tag{1.6}$$

where constants a_p and a_d are as yet undetermined. Each of the terms thus has a different "error" associated with it. Note that when $a_p = a_d = 1$, that we have the original PID design. Note also that when $r(t)$ is a piecewise constant signal (only step changes), then for all time (except at the actual step locations), $\dot{r}(t) = 0$, and thus,

$$\frac{de_d(t)}{dt} = \frac{d}{dt}\left(a_d r(t) - y(t)\right) = -\dot{y}(t) \tag{1.7}$$

which is independent of $r(t)$ and a_d. In general, since y is the output of the plant, it will be a smooth function and thus $\dot{y}$ will be bounded. It is thus not uncommon to let $a_d = 0$. This eliminates spikes in the term $K_d \frac{de_d(t)}{dt}$, without substantially affecting the overall control performance.

A block diagram for this is shown. As it appears, it seems much more complicated. However, it is actually not much more complicated.

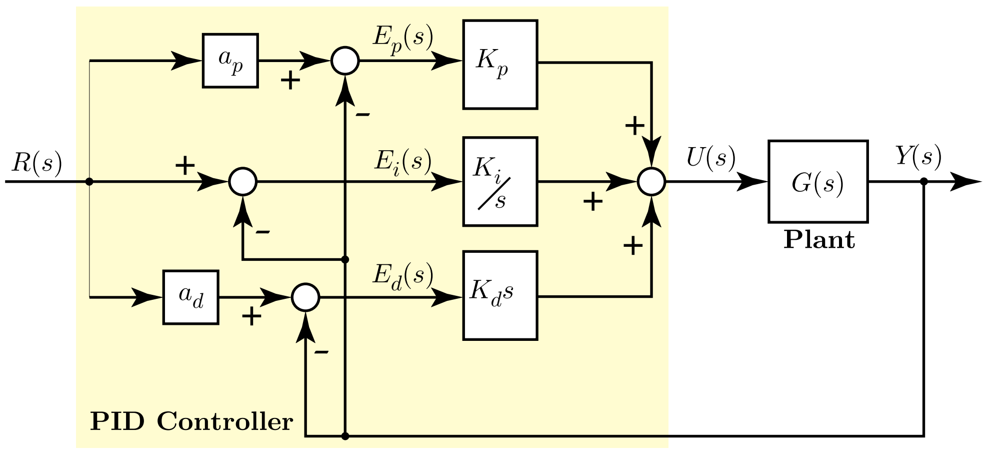

Figure 2: **The Setpoint Weighting Configuration**

As we have seen, changing a_d doesn't change the overall design. Changing a_p, however, may change the design. The rationale behind a_p is that if, for example, $a_p = 0.5$, then a large step change in $r(t)$ doesn't create such a large control magnitude. However, in general, e_p does not go to zero when $y = r$. Thus, there is a persistent control applied even when it is not necessary. The use of this, then is of questionable value. Setting $a_p = 1$, however, brings us back to the original error.

2.3 Overall Usage

The PID controller performs especially well when the system has first order dynamics (a single pole). Actually, in this case, the P controller is a state-feedback control! In general, for the system with first-order dynamics the PI control is sufficient, and the D is not needed.

For the system with essentially second-order dynamics, the PD control corresponds to state feedback. The PID control generally works well for these systems.

The PID controller can also work for some systems of higher order. Generally speaking, the derivative term is beneficial when the time constants of the system differ by orders of magnitude. It is sometimes helpful when tight control of higher-order dynamics is required. This is because the higher order dynamics prevent the use of high proportional gain. The D provides damping and speeds up the transient response.

The PID controller, of course, is not the end-all of controllers. Sometimes the controller just won't work very well. The following are cases when the PID control doesn't perform well. In general, these require the use of more sophisticated methods of control.

- Tight control of higher order process
- Systems with long delay times. In this case, the derivative term is not helpful. A "Smith predictor" is often used in this case.
- Systems with lightly damped oscillatory modes
- Systems with large uncertainties or variations.
- Systems with harmonic disturbances
- Highly coupled multi-input, multi-output systems—especially where coordination is important.

2.4 Example

Consider the system given by the transfer function

$$G(s) = \frac{20(s+2)}{\left(s+\frac{1}{2}\right)(s^2+s+4)} \tag{1.8}$$

If we use the (modified) PID controller

$$\begin{aligned} C(s) &= K_p + \frac{K_i}{s} + \frac{K_d s}{\tau_1 s + 1} \\ &= \frac{(K_p\tau_1 + K_d)s^2 + (K_p + K_i\tau_1)s + K_1}{s(\tau_1 s + 1)} \end{aligned}$$

where

$$K_p = 9, K_i = 4, K_d = 1, \tau_1 = \tfrac{1}{100} \tag{1.9}$$

We use the following MATLAB code to simulate the system:

```
num=20*[1 2];den=conv([1 .5],[1 1 4]);
Kp=9;Ki=4;Kd=2;t1=1/100;
numc=[Kp*t1+Kd,Kp+Ki*t1,Ki]/t1;denc=[t1,1,0]/t1;
numf=conv(num,numc);denf=conv(den,denc);
dencl=denf+[0 0 numf];
numy=numf;
syso=tf(num/20,den);sysy=tf(numy,dencl);
t=[0:.002:10]';
yo=step(syso,t);yc=step(sysy,t);
figure(1);plot(t,[yo yc]);grid
xlabel('time (sec)');ylabel('y(t)')
```

We thus obtain the plot shown. This plot compares the open-loop response with that of the PID controlled response. For the open-loop, we simply scaled the step input so that the overall system output settled out to 1.

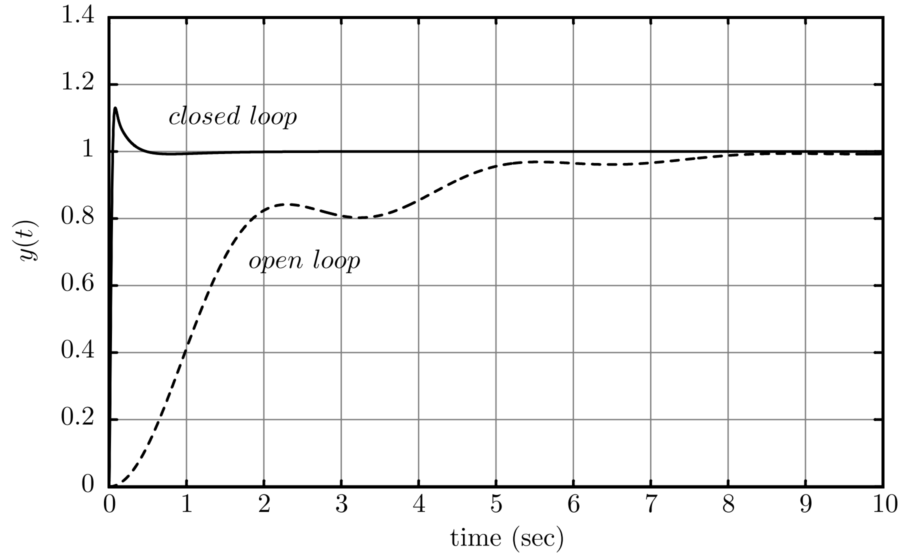

Figure 3: **Step Response for PID and Open-Loop Systems**

We see that the PID controlled system performs much better than the open loop. It is often advantageous to make a comparison with the open-loop system, since that shows what would happen using the simplest kind of control. Indeed, the open loop requires no sensors and is thus much less costly to build and maintain than the PID system. If the PID system did not perform so much better, it might not be worthwhile.

3 Plant Modeling

In general, the more we know about a system, the better we will be able to control it. If we know *nothing* about a system, we better not be trying to *control* it. A logical question, then, is how much (or little) do we need to know, and how will we obtain that data?

3.1 Information Required

The more information available about the system, the better it can be controlled[2]. Many processes to be controlled are either difficult to model or else it is difficult to obtain the parameters for the model. For example, the dynamic inductance of a dc motor is very difficult to obtain. Generally, to obtain a model, it is necessary to perform some experiments to obtain actual parameter values. The type of experiments will determine what data is obtained. Here, we consider three types of experiments. Each will give a different type of information. That information will be used to design the PID controller.

In general, the experiments to be performed require that the system be stable. If not, the experiments won't work. This, of course, is a great restriction. Unless the exact model of the unstable system is known, there is no known way to tune a PID controller. Some controllers, such as adaptive controllers are able to compensate for unstable systems, however their complexity excludes them from consideration in this course.

3.2 Step Response Information

A simple experiment to perform is the step response test. Here, we assume a unit step is applied to the system, although it is also possible to scale the whole experiment is necessary. From the results of this test, the data may be used to obtain approximations of the system.

Figure 4 shows example step responses for a number of different possibilities. The tuning rules were essentially developed for a step response of type **A**. The rules, however, may work for responses of type **B** and **E**. The responses **C** and **D** are for unstable systems. If the plant is exactly known, it might be possible to design a PID controller, but there is no guarantee. As for the system with response **F**, the system has a zero in the RHP (unstable zero). A PID controller may reduce the settling time of the system, but is likely to accentuate the undershoot.

The PID controller rules are thus essentially designed for a first or second-order, non-oscillatory system with possibly some time-delay. An example plot is shown in Figure 5.

The step response has several values that are of importance in obtaining an approximate transfer function for the system. First, we see y_{ss} which is the steady-state value for the step input. Next, we see y_0, τ_0, and m, which gives a linear approximation of the

[2]There are limits, of course, as to what can be achieved.

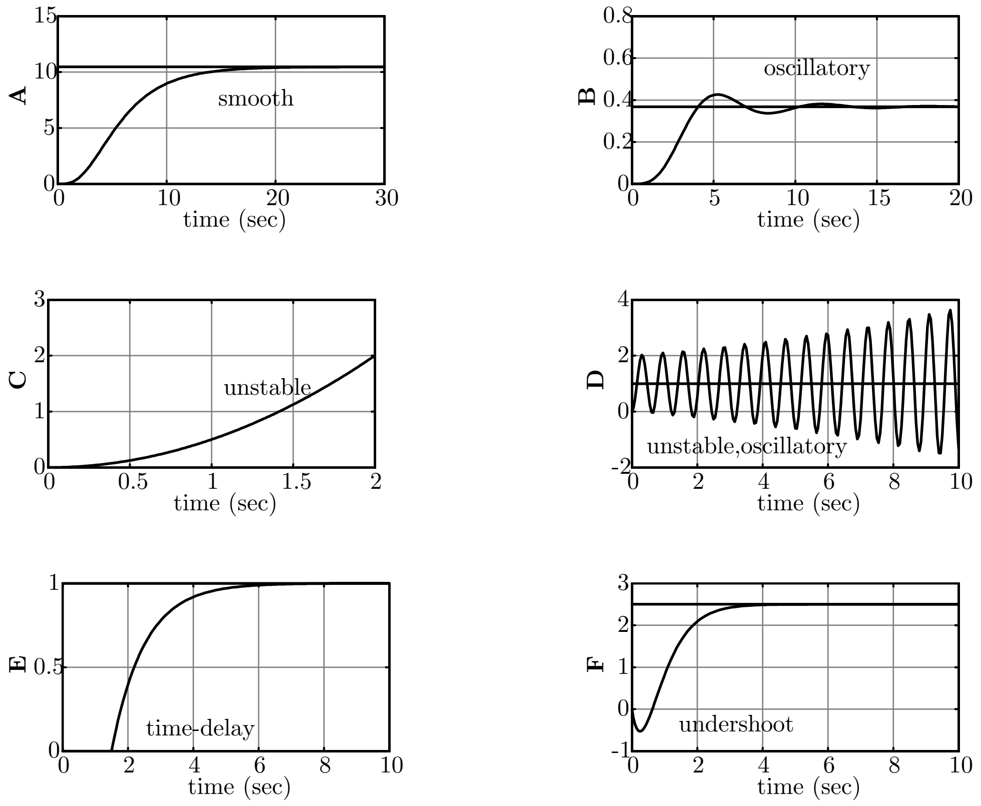

Figure 4: **Six Possible Step Responses**

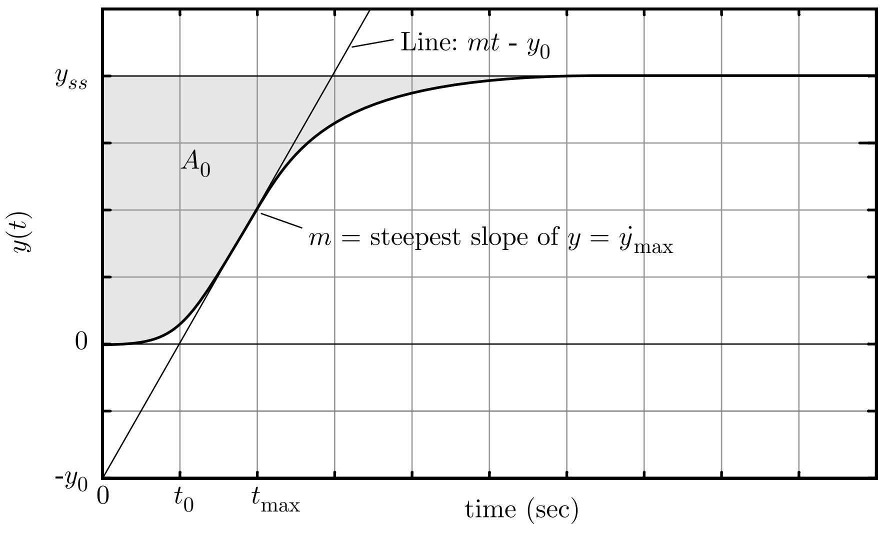

Figure 5: **A Typical Step Response**

transient portion of the step response. Given only the step response data, we first find the point at which the derivative of y is at a maximum. This value is $\dot{y}_{\max}$ and occurs at time $t_{\max}$. The equation of the line that passes through the point $y(t_{\max})$ with slope $m = \dot{y}_{\max}$ is given by

$$\text{Line:} \quad (\dot{y}_{\max})\, t + y(t_{\max}) - (\dot{y}_{\max})(t_{\max}) \tag{1.10}$$

which is indeed a linear function of time t. From this, we may determine

$$y_0 = (\dot{y}_{\max})(t_{\max}) - y(t_{\max}) \tag{1.11}$$

$$\tau_0 = \frac{y_0}{\dot{y}_{\max}} \tag{1.12}$$

Here, τ_0 is often called the *apparent dead time.* In this case, we may approximate the plant by the model

$$G_{a1}(s) = \left(\frac{y_0}{\tau_0}\right) \frac{e^{-s\tau_0}}{s} \tag{1.13}$$

which doesn't match the steady-state characteristics very well, but does match the transient portion in a linear fashion. A model that approximates the system's steady-state behavior is given by

$$G_{a2}(s) = \frac{y_{ss}}{1 + (\tau_{ar})\, s} \tag{1.14}$$

where y_{ss} is the steady-state value of y and τ_{ar} is the so-called *average residence time*, and is given by

$$\tau_{ar} = \frac{A_0}{y_{ss}} \tag{1.15}$$

and where A_0 is the area between y_{ss} and $y(t)$

$$A_0 = \int_0^\infty |y_{ss} - y(t)|\, dt \tag{1.16}$$

Note that even though the integral is all the way to ∞, actually the integral may be approximated for a shorter time since the difference $|y_{ss} - y(t)|$ becomes small after a certain point. The approximation G_{a2} generally doesn't match the transient portion extremely well, but is a good first-order approximation. It has been observed that the quantity

$$\kappa_1 = \frac{\tau_0}{\tau_{ar}} \tag{1.17}$$

is a measure of the difficulty of control. Generally $0 \leq \kappa_1 \leq 1$, and the closer κ_1 is to 1, the more difficult the system is to control using the PID. Still another approximation is given by

$$G_{a3}(s) = (y_{ss}) \frac{e^{-\tau_0 s}}{1 + \tau_{at} s} \tag{1.18}$$

where

$$\tau_{at} = \tau_{ar} - \tau_0 \tag{1.19}$$

is called the *apparent time constant.* This is the better of the three approximations. Generally speaking, τ_{at} is less susceptible to noise in the step response.

3.3 Frequency Response Information

Another type of experiment that actually provides more useful (in terms of designing the PID controller) information is a frequency response test. Here, we do a simple test. Again, we assume that the system is stable. We also let $r(t) = \varepsilon$, a very small value.

There are two variations on the Frequency response tests. Both yield essentially the same information. The first test, illustrated in Figure 6(a) is that of a **P** controller. Here, the gain K_p is increased from zero until the system begins to oscillate. When K_p is set such that there is a constant oscillation (neither increasing nor decreasing), that value is called the *ultimate gain* and is denoted K_u. The oscillation will generally be periodic with some period T_u.

This approach is generally risky since the plant is operated near instability. Also, generally, it is difficult to keep the control amplitude bounded (important for safety!). Thus, in general, this method is difficult to automate.

The second variation is based on the diagram in Figure 6(b). Here, a relay is employed, giving the control signal

$$u(t) = \begin{cases} \ell, & e(t) \geq 0 \\ -\ell, & e(t) < 0 \end{cases} \tag{1.20}$$

This control, for most systems of interest, will result in an oscillation. Eventually the control input will be a square wave of amplitude ℓ, and have a period approximately

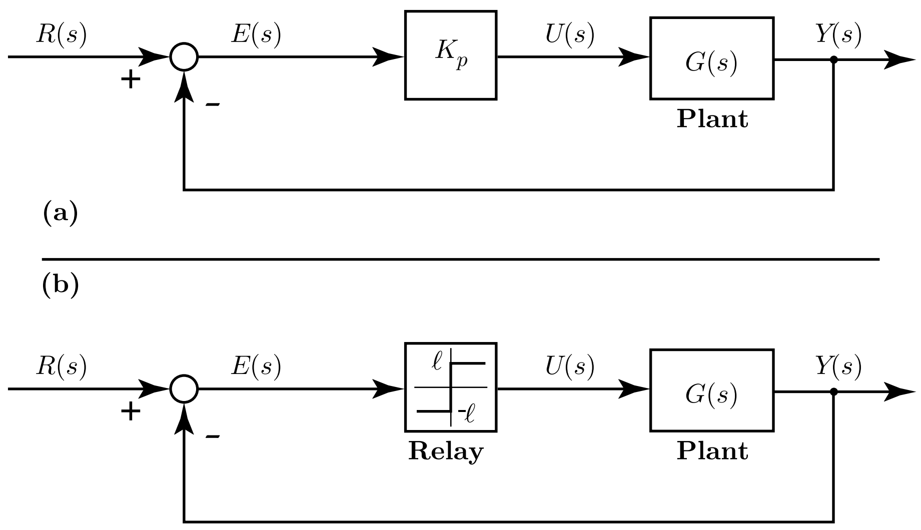

Figure 6: **Frequency Response Test Setups**

equal to T_u from the previous version of the test. The output of the system will settle out to be a sinusoid of amplitude α. In this case, we may obtain the ultimate gain from the formula[3]

$$K_u = \frac{4\ell}{\pi\alpha} \tag{1.21}$$

This K_u and T_u are close to the actual values of K_u and T_u found in the previous variation, the previous values being actually more accurate.

Thus, regardless of which variation we choose, we end up with T_u and K_u. The nice thing about this test is that the control signal is bounded for all time. The threat of instability is also reduced.

3.4 System Identification

System Identification is a method of obtaining a transfer function by applying a known test signal to a system, observing the output, and comparing it with the input to determine the transfer function of the system. This test is generally quite a bit more involved than either the step response test or the frequency response test. It yields, however, an actual transfer function, rather than just a few values of an approximate transfer function.

Because of the complexity, this topic will not be considered here, but will be relegated to another course.

[3]This value is obtained by a type of nonlinear analysis called *describing functions*.

3.5 Example

Consider the plant

$$G(s) = \frac{10000}{s^4 + 126s^3 + 2725s^2 + 12600s + 10000} \tag{1.22}$$

We use the following SIMULINK model to perform the step response experiment. Note that in this setup, we have blocks added to help compute the necessary constants. Performing the simulation once, we obtain

$$\dot{y}_{\max} = 0.6632 \tag{1.23}$$

This value is plugged back into the diagram.

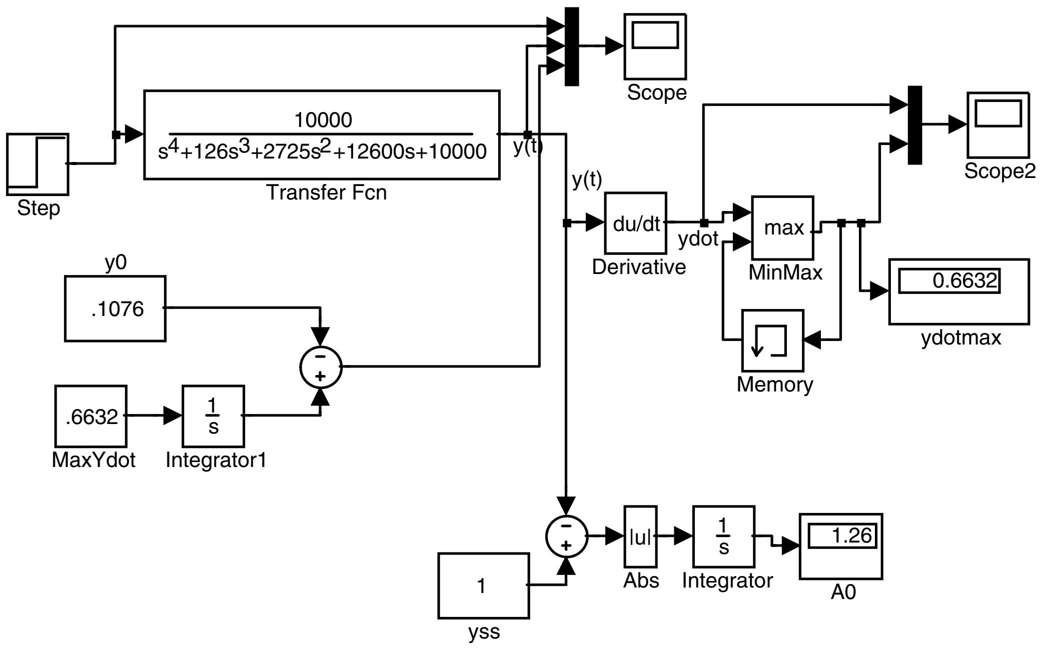

Figure 7: **SIMULINK Diagram for the Step Response Test**

In addition, looking at `Scope2`, we can find the time at which the maximum slope occurs. Zooming in, we obtain

$$t_{\max} = 0.5 \tag{1.24}$$

Zooming into `Scope`, we find that

$$y(t_{\max}) = y(0.5) = 0.224 \tag{1.25}$$

From this we may calculate

$$\begin{aligned} y_0 &= (\dot{y}_{\max})(t_{\max}) - y(t_{\max}) \\ &= (0.6632)(0.5) - (0.224) = 0.1076 \qquad (1.26) \end{aligned}$$

This is also plugged back into the diagram. We also compute

$$\tau_0 = \frac{y_0}{\dot{y}_{\max}} = \frac{0.1076}{0.6632} = 0.1622 \qquad (1.27)$$

We note from `Scope` that $y_{ss} = 1$. This is plugged back into the diagram and the simulation is performed again. Doing so, we find the line displayed as in Figure 5 in `Scope`. We also obtain the integral value

$$A_0 = 1.26 \qquad (1.28)$$

We thus may compute

$$\begin{aligned} \tau_{ar} &= \frac{A_0}{y_{ss}} = \frac{1.26}{1} = 1.26 \qquad (1.29) \\ \tau_{at} &= \tau_{ar} - \tau_0 = 1.26 - 0.1622 = 1.0978 \qquad (1.30) \end{aligned}$$

and find that

$$\kappa_1 = \frac{0.1622}{1.26} = 0.1287 \qquad (1.31)$$

indicating that this system is not too hard to control using a PID controller.

For the Frequency Response Test, we use the following SIMULINK diagram.

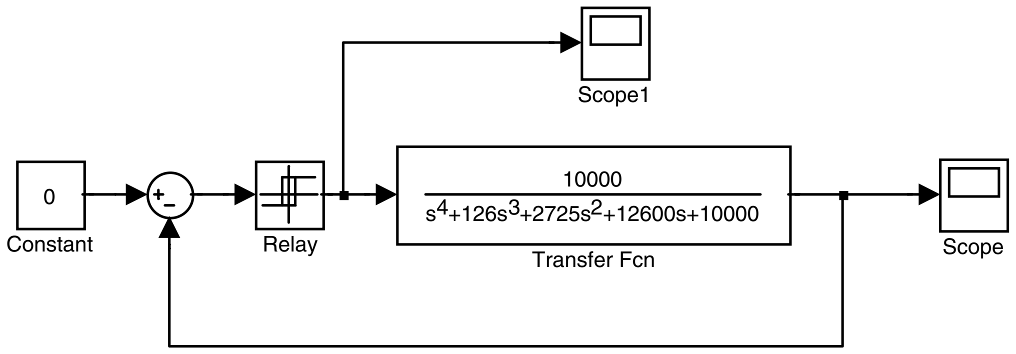

Figure 8: **SIMULINK Frequency Response Test Configuration**

This is simulated for 100 seconds. The relay has amplitude $\ell = 1$. Looking at either `Scope` or `Scope1`, we find the period of oscillation to be

$$T_u \approx 0.64 \qquad (1.32)$$

and looking near $t = 100$, at `Scope`, we find that the steady-state amplitude is

$$\alpha = 0.0527 \tag{1.33}$$

From this we find the ultimate gain

$$K_u \approx \frac{4\ell}{\pi\alpha} = \frac{4\,(1)}{\pi\,(0.0527)} = 24.16 \tag{1.34}$$

Note that the actual values for this are $T_u = \frac{2\pi}{10} = 0.6283$, and $K_u = \frac{101}{4} = 25.25$.[4]

4 PID Tuning

Having obtained a basic system information, from either of the tests, we are now in a position to perform some designs on the system. These designs are based upon the data obtained by the above experiments.

4.1 Ziegler-Nichols Step Response Method

This is the earliest design method for the PID controller. It was originally developed in 1942, so its not exactly state-of-the-art. It does, however, work effectively for many systems. This method is based on the approximate model $G_{a1}(s)$. Once this model has been determined by the step response test, we may design controllers according to the following table. Note that only y_0 and t_0 are used.

Controller	K_p	K_i	K_d
PI	$\frac{9}{10y_0}$	$\frac{3}{10y_0\tau_0}$	0
PID	$\frac{6}{5y_0}$	$\frac{3}{5y_0\tau_0}$	$\frac{3\tau_0}{5y_0}$

These formulae are heuristic rules based upon the response of many different systems. The design criteria was to ensure that the amplitude of closed-loop oscillation decay at a rate of 1/4. This is actually often too lightly damped. It is also often too sensitive.

4.2 Ziegler-Nichols Frequency Response Method

From the Frequency Response Test (or Relay Test), we obtained the ultimate gain K_u and T_u. From these values, we obtain the following table.

Controller	K_p	K_i	K_d
PI	$\frac{2K_u}{5}$	$\frac{K_u}{2T_u}$	0
PID	$\frac{3K_u}{5}$	$\frac{6K_u}{5T_u}$	$\frac{3K_uT_u}{40}$

[4]These were obtained analytically because we actually know the system exactly. They are close enough to the approximations which don't require exact knowledge to obtain.

We note that in this case as in the previous case,

$$\frac{K_i K_d}{K_p^2} = \frac{1}{4} \tag{1.35}$$

This appears to be common in PID tuning. Both methods also often give too high a K_p, which is related to the designed decay ratio. The ZNSR method generally gives a higher K_p than the ZNFR method.

4.3 Chein-Hrones-Reswick Method

A modification of the Ziegler-Nichols method is the Chein-Hrones-Reswick (CHR) method. This is based more on a setpoint response. This is a step-response method and uses y_0, τ_0 and τ_{at}. In this case, we have the following table.

Controller	K_p	K_i	K_d
PI	$\frac{7}{20y_0}$	$\frac{7}{24y_0\tau_{at}}$	0
PID	$\frac{3}{5y_0}$	$\frac{3}{5y_0\tau_{at}}$	$\frac{3\tau_0}{10y_0}$

4.4 Example

Consider the example given in the previous section. Here, we consider only PID controllers. There, we obtained the values $y_0 = 0.1076$, $\tau_0 = 0.1622$, $\tau_{at} = 1.0978$, $T_u \approx 0.64$, and $K_u \approx 24.16$. We thus obtain the controller designs

Controller	K_p	K_i	K_d
ZNSR	11.1524	34.3786	0.9045
CHR	5.5762	5.0794	0.4522
ZNFR	14.496	45.300	1.1597

MATLAB is used to perform the simulations of the systems. We thus obtain the plot shown below. We note that the CHR design is the best overall. That is partially due to the fact that it is tailored to the setpoint problem.

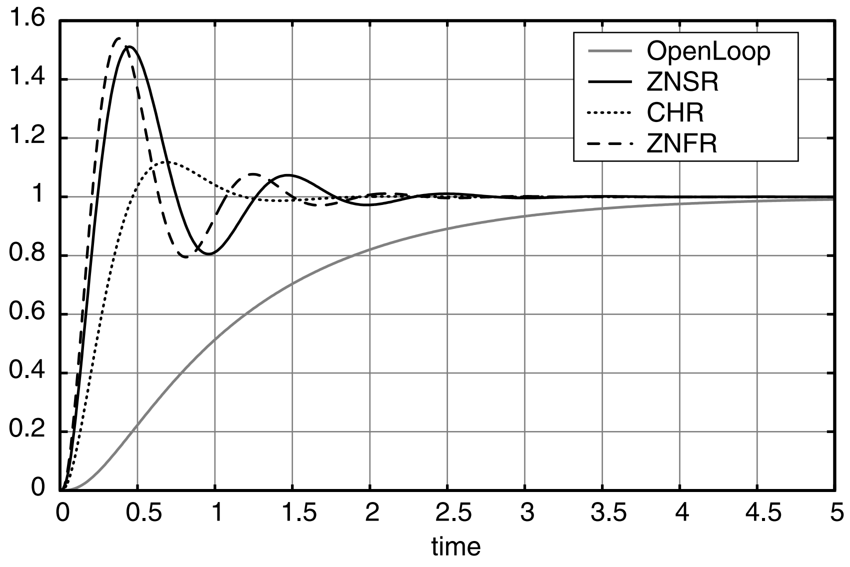

Figure 9: **Example Step Responses**

5 Integrator Windup

In general, the actuator for any system is limited, i.e., there is a maximum exertion that the actuator can accomplish. If the actuator is a power amplifier, it generally has “rails.” Thus, the design for a linear system is not sufficient since the real system is not linear, but has a nonlinear saturation term. A controller designed for a linear system will often not work on a nonlinear system. This configuration is illustrated below. Here, a disturbance input is also added.

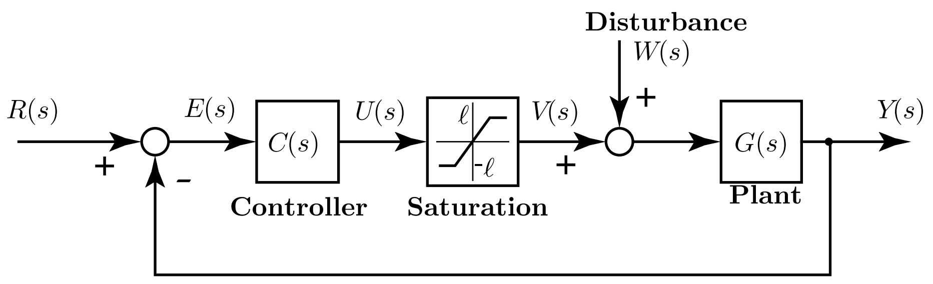

Figure 10: **Nonlinear System with Disturbance**

When the actuator saturates, the loop is effectively open. In particular, if the control is such that a long period of saturation exists, it is possible for the integral term to keep integrating even when it is not reasonable to do so. the integrator itself is open-loop unstable, and the integrator output will drift higher. This is called *integrator windup.* In order for the integrator to "unwind," the error signal must actually change sign before the output of the integrator will *start* to return to zero. Thus, the actuator remains saturated even after error changes sign since the controller's output stays high until the integrator resets.

This is difficult to visualize, so we give our example next.

5.1 Example

Consider the problem of the previous example, using the ZNSR method. Suppose we have a saturation limit ℓ where we may have $\ell = 1, 1.5, 2$ or 5. Suppose also that the disturbance $w(t)$ is a negative unit step input that turns on at time $t = 15$. A plot of this is shown below, and is compared with the open-loop system. Note that the open-loop system goes to zero after the disturbance enters because the disturbance "cancels" the reference signal.

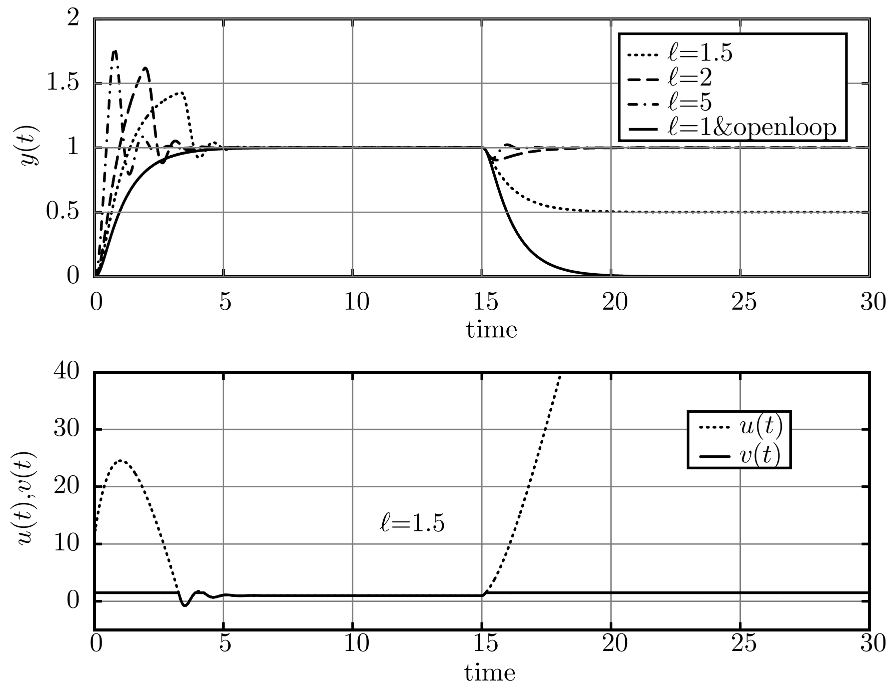

Figure 11: **Integrator Windup Effects when** $\ell = 1, 1.5, 2, 5$

When $\ell = 1$, the control signal quickly saturates and thus the system behaves essentially as the open-loop system. We have shown the control signal $u(t)$ and the actuation signal $v(t)$ for $\ell = 1.5$ in the bottom graph. Here, the input immediately saturates, but finally resets at around $t = 3$, in which case the system begins to act like a linear system. When the disturbance comes, the integrator again saturates the actuator, and the integrator value drives $u(t)$ off to infinity, never to return. The net effect on the output is that the output does not return to the setpoint value, but settles at 0.5. This is better than the case of $\ell = 1$, however. For $\ell \geq 2$, the system recovers from the disturbance and continues tracking. When $\ell = 5$, the output looks identical to the original ZNSR response.

5.2 Back Calculation

The idea of an anti-windup technique is to mitigate the effects of the integrator continuing to integrate due to the nonlinear saturation effect. In the Back-Calculation method, when the actuator output saturates, the integral is recomputed such that its output keeps the control at the saturation limit. This is actually done through a filter so that anti-windup is not initiated by short periods of saturation such as those induced by noise.

This method can be viewed as supplying a supplementary feedback path around the integrator that only becomes active during saturation. This stabilizes the integrator when the main feedback loop is open due to saturation. In this case, we have the setpoint-weighted control law

$$u(t) = K_p e_p(t) + \int_0^t \bar{e}_i(\tau)\, d\tau + K_d \frac{de_d(t)}{dt} \tag{1.36}$$

where

$$e_p = a_p r(t) - y(t), \quad e_d(t) = a_d r(t) - y(t) \tag{1.37}$$

and where we have the integral error

$$\bar{e}_i(t) = K_i (r(t) - y(t)) + \frac{1}{T_t}(v(t) - u(t)) \tag{1.38}$$

Note that if there is no saturation, then $v(t) = u(t)$ and $\bar{e}_i$ corresponds to e_i given previously. When $v(t) \neq u(t)$, then the *tracking time constant* T_t determines how quickly after saturation the integrator is reset.

In general, the small T_t is better as it gives a quicker reset. However, if it is too small, noises may prevent it from actually resetting the integrator. As a general rule, we choose

$$\tau_d < T_t < \tau_i \tag{1.39}$$

where τ_d is the derivative time and τ_i is the integral time referred to earlier. A typical choice will be

$$T_t = \sqrt{\tau_d \tau_i} \tag{1.40}$$

We revisit the previous example. In this case, we use the Back-Calculation in performing the integration, with

$$T_t = \sqrt{\tau_d \tau_i} = \sqrt{\frac{K_d}{K_i}} \tag{1.41}$$

again with $\ell = 1, 1.5, 2$ and 5. A plot of the response is shown below.

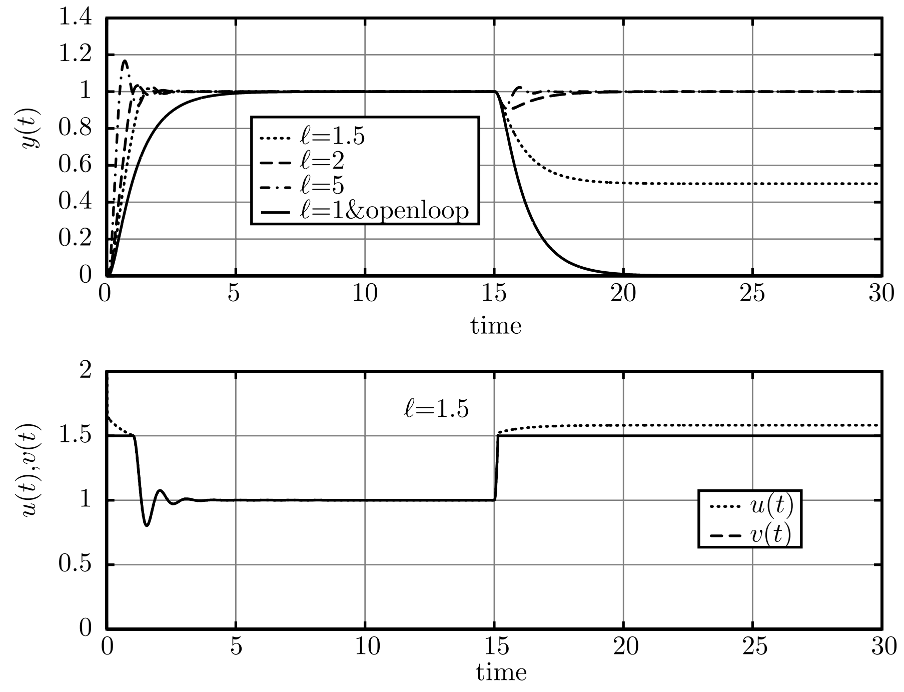

Figure 12: **Back-Calculation Responses**

Here, we see that for $\ell = 1$, there is effectively no change in the response. This is because the saturation is so close to the boundary for the step input that there is no "headroom."

For $\ell = 1.5$, we also have a comparable disturbance response to that above. This is due to the fact that the combination of the setpoint and the disturbance are past the saturation limit. We note, however, two beneficial aspects of the back-calculation. The first is that the response to the initial setpoint change is significantly better than when the back-calculation is not used. We have virtually no overshoot, and a much faster settling time. This is because the back-calculation resists the tendency of the controller to initially saturate. The second benefit is that when the disturbance hits, the control signal $u(t)$ does not head off to infinity, but remains bounded.

For $\ell \geq 2$, we see a comparable disturbance rejection to that of the original problem. However, as with the $\ell = 1.5$ case, we see a significantly improved transient response to the setpoint change. Overall, the back-calculation has improved the performance of the system.

5.3 Conditional Integration

In this case, we switch the input to the integrator to zero whenever a “crisis” condition exists. A crisis may be

- when the error signal is large
- when the actuator is already saturated
- when the actuator is close to saturation and the present value of the control signal is driving it toward saturation.

Any of these definitions may be used to define when the integrator is integrating the error and when it is not. Thus,

$$u(t) = K_p e_p(t) + \int_0^t \tilde{e}_i(\tau)\, d\tau + K_d \frac{de_d(t)}{dt} \tag{1.42}$$

where e_p and e_d are as before, and we have the integral error

$$\tilde{e}_i(t) = \begin{cases} K_i(r(t) - y(t)), & \text{no “crisis”} \\ 0, & \text{“crisis”} \end{cases} \tag{1.43}$$

One example of this is to use the “crisis” error function

$$\tilde{e}_i(t) = K_i(r(t) - y(t))\, e^{-(100(u(t)-v(t)))^2} \tag{1.44}$$

which is equal to $K_i(r(t) - y(t))$ when $u(t) = v(t)$. When $u(t) \neq v(t)$, $K_i(r(t) - y(t))$ is multiplied by a number that is closer to zero the further $u(t)$ and $v(t)$ are from each other. If $u(t)$ is far enough from $v(t)$, then $\tilde{e}_i(t)$ effectively becomes zero. This function does this in a “smooth” way that doesn’t create “chattering” associated with nonsmooth nonlinear (switching) functions.

Using this function on the previous example, in place of the Back-Calculation method, we obtain essentially the same plots. One difference, however, is that unlike the Back-Calculation method, this method may have a larger control value when the system is saturated and the anti-windup method doesn’t do as much to control this.

6 Conclusion

PID controllers can work surprisingly well, especially considering how little information is provided for the design. Several methods for tuning the controllers have been presented. There can be vast differences in the results produced by different tuning procedures. The quality of the tuning is very much dependent on the compatibility of the tuning method with the plant behavior and the performance goals.

Bibliography

[1] Mathworks, (1999), *Control System Toolbox User's Guide*, Natick, MA: The Mathworks Inc.

[2] Åström, K. and T. Hagglund, (1995) *PID Controllers: Theory, Design, and Tuning*, 2nd Edition, Instrument Society of America

[3] Knospe, C.R. (2000), *PID Control: Tuning and Anti-Windup Techniques*, "Practical Control Techniques for Control Engineering" workshop at the 2000 American Control Conference.