PID Controller Tuning Guide

Richard Poley *C2000 System Applications*

ABSTRACT

PID (Proportional-Integral-Derivative) control first appeared in 1922[1] and has become the most common type of control strategy[2]. In recent years, the availability of large amounts of computing power at low cost has led to the adoption of digital PID control in high sample rate systems such as motor drives and switched electrical power systems. This document presents a brief guide to digital linear PID control, and suggests a step-by-step procedure for tuning such a controller.

Contents

Figures

1 Introduction

PID (Proportional-Integral-Derivative) control first appeared in 1922[1] and has become the most common type of control strategy[2]. In recent years, the availability of large amounts of computing power at low cost has led to the adoption of digital PID control in high sample rate systems such as motor drives and switched electrical power systems. This document presents a brief guide to digital linear PID control, and suggests a step-by-step procedure for tuning such a controller.

In principle, the PID controller comprises three parallel paths, each of which influences the control action in a different way. The effects are most clearly visualized in the features of the transient part of the closed loop response; for example, in the output response following a step change in the control reference or the output load; and for this reason the use of PID control for control loop tuning in the time domain is well established.

In general, proportional action directly influences open loop gain: an increase in proportional gain reduces the response rise time, but does so at the expense of increased over-shoot and possibly oscillation. The purpose of integral action is to eliminate steady-state error; large integral gain resulting in a more rapid settling of the response. However integral control accentuates any oscillatory characteristics already present in the response. Derivative action is a predictive type of control, its' influence being to introduce a damping effect into the response which counter-acts oscillatory effects arising from proportional and integral action.

A user would typically proceed to tune the controller by adjusting each term in sequence while monitoring the transient response. Gains adjustments would be made iteratively, in a cyclic fashion, the process often being supported by the use of a performance index[3]. It is the simplicity of the PID structure, together with the correlation between properties of the transient response and controller parameters which accounts for its popularity.

It is common to encounter PI and PID controllers in situations where the control loop is tuned in the frequency domain. One such application is field oriented motor control, where two inner current loops each contain a PI controller, the parameters of which are selected to position the controller zero to cancel the motor pole formed by the stator inductance and resistance. Other examples may be found in power electronics of the use of PID control to fix a pair of controller zeros. Since these design requirements can be met using other controller structures they will not be described further in this document.

Over the ninety-five years since its invention the PID controller has been the subject of much research, and enhancements are now routinely made to avoid well-known practical control issues [1]. Foremost among these are measures to avoid integrator "windup" during saturation of the control loop, and filtering to prevent high frequency noise amplification through the derivative path.

Section 2 describes the structure of the basic linear PID controller. Section 3 presents a brief overview of the PID implementation in the C2000 Digital Control Library. Section 4 describes utilities to measure performance based on transient response. Section 5 steps the reader through a suggested PID tuning procedure; this being illustrated by an example in the following chapter. Finally, a list of literature references and suggestions for further reading can be found in section 7.

2 The Linear PID Controller

In the simplest negative feedback control loop, the controller is situated between the error junction and the plant. The input to the controller is the loop error $e(t)$, and its' output $u(t)$ represents a corrective action to be applied to the plant such that $e(t)$ is always driven towards zero as rapidly as possible. In what follows, the reference input to the control loop will be denoted $r(t)$, and the output of the plant $y(t)$. This type of configuration is known as cascade control. Other control loop configurations are possible, but to keep this document to a manageable size only cascade control is considered here.

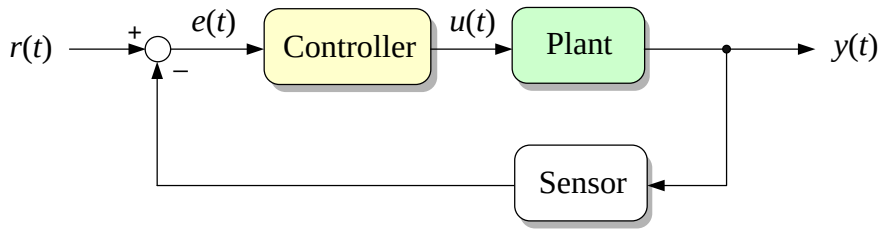

Figure 1. The cascade control loop

The simplest type of control action is "proportional": either gain or attenuation applied in series with the forward path. Denoting the proportional gain as k_p, we write

$$u_p(t) = k_p e(t) \quad \ldots(1)$$

Adjustment of the proportional controller gain directly changes the open loop gain, but has no effect on open loop phase. In general, increasing proportional gain reduces rise time and steady state error, however large proportional gain can induce over-shoot and oscillation in the transient response, so in practice there is an upper limit on the feasible value of k_p.

Depending on the plant, the steady state output may or may not converge on the desired value. In situations where it does not, integral control action can be applied. The integral path control law is

$$u_i(t) = k_i \int_{-\infty}^{t} e(\tau)\, d\tau \quad \ldots(2)$$

Adjustment of the integral gain k_i changes the rate at which the response converges on steady state: the larger the gain, the faster the rate of convergence. However the integral gain tends to amplify any over-shoot and oscillation which may be present, so again there is an upper practical limit on the value of k_i.

Notice that the input to the integral path is the loop error. The integrator acts to drive $e(t)$ to zero, however this does not imply that $u(t)$ will also converge on zero. Depending on the plant, it is quite possible that the integrator output will be non-zero even when the steady state error has been removed.

The derivative path has the formula

$$u_d(t) = k_d \frac{de(t)}{dt} \quad \ldots(3)$$

By differentiating the error, this type of control induces a predictive action in which the faster the rate of change of the error, the larger corrective effort is applied. The derivative gain k_d acts as a sort of time constant which fixes the time interval over which the error is predicted.

Combining the three terms we have a formula, or "control law", for the PID:

$$u(t) = k_p e(t) + k_i \int_{-\infty}^{t} e(\tau) d\tau + k_d \frac{de(t)}{dt} \quad \ldots(4)$$

The influence of each of these three terms is shown in terms of a typical transient response in the figure below.

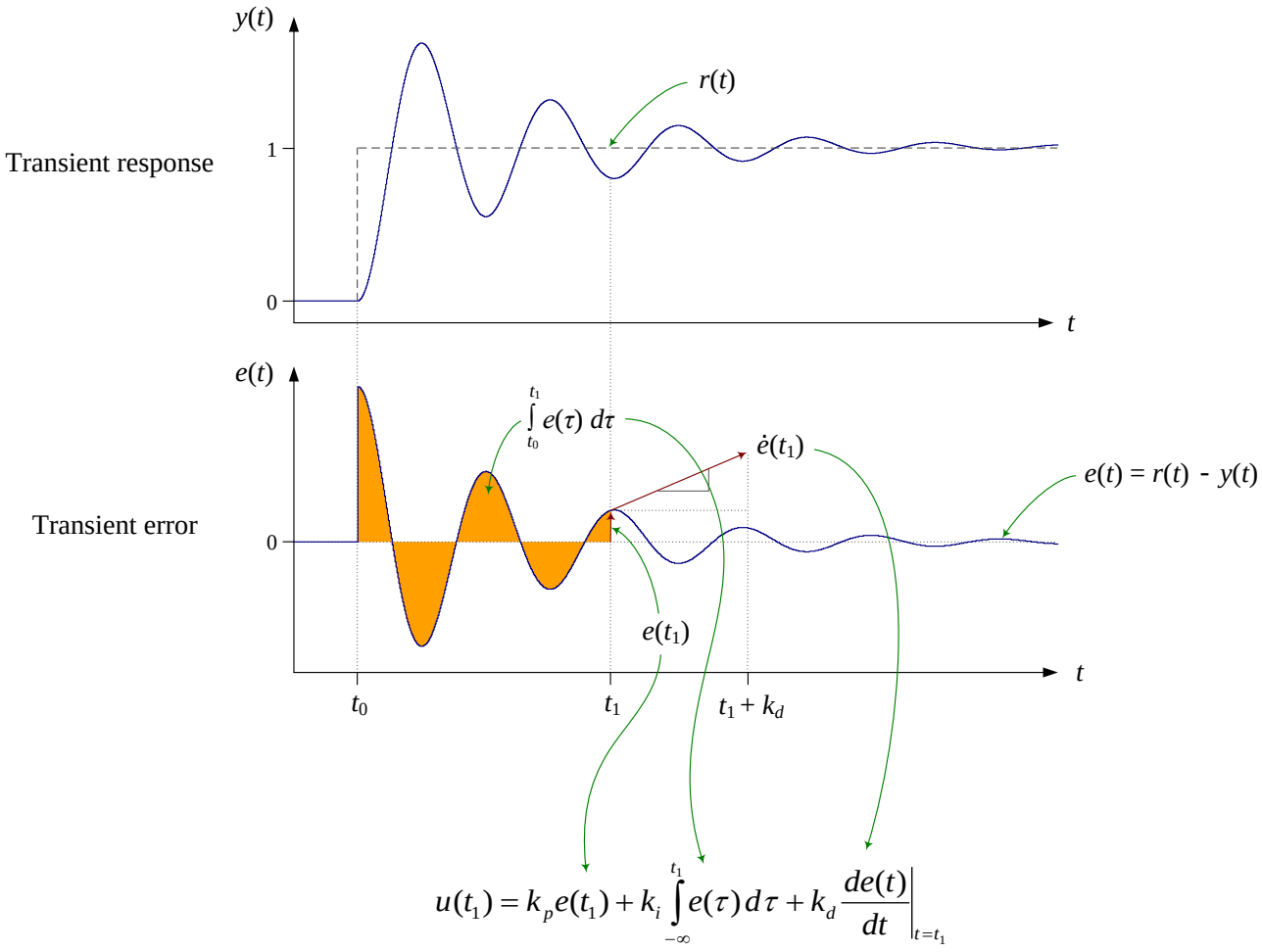

Figure 2. PID control action

This type of PID is referred to as the parallel form, since each of the three control actions appears in a separate parallel path, and the output of the controller is their parallel sum

$$u(t) = u_p(t) + u_i(t) + u_d(t)$$

In a slightly different PID configuration, the proportional term appears in series with the parallel connection of integral and derivative paths, so the control law becomes

$$u(t) = k_p\left(e(t) + k_i\int_{-\infty}^{t} e(\tau)d\tau + k_d\frac{de(t)}{dt}\right) \quad \ldots(5)$$

This configuration is referred to as a "series" PID. It is claimed that the series controller is easier to tune than the parallel form because adjustment of k_i and k_d do not affect the optimum choice of k_p.

3 DCL Implementation

The C2000 Digital Control Library (DCL) contains a collection of closed loop controller algorithms optimized for the C2000 micro-controller, among which are several linear PID controllers. The library is supplied in the form of C and assembly source code, and distributed in the "C2000Ware" software package[7]. New users of the library should begin by reading the chapters 1 & 2 in the DCL User's Guide[4]. A detailed description of the linear PID controllers is given in section 3.1 of that document and will not be repeated here.

The linear PID controllers in the DCL include the following features:

- Parallel and ideal forms
- Adjustable output saturation
- Independent reference weighting on proportional path
- Anti-windup integrator reset
- Programmable low-pass derivative filter
- External saturation input for integrator anti-windup

Figure 3 shows an implementation of parallel PID controller known as "C4". The three parallel paths are clearly visible, as are the associated gain terms. The "i14" variable is set by the output clamp block and determines whether integration is active.

The lower path contains the filtered differentiator, with the low-pass filter bandwidth set by coefficients "c1" and "c2". The purpose of the filter is to avoid amplification of un-wanted high frequency noise. The filter in the C4 controller is a simple first order lag with differentiator, converted into discrete form using the Tustin transform. Refer to the DCL User's Guide[4] for more information.

All PID type controllers in the library implement integrator anti-windup reset in a similar way. A clamp is present at the controller output which allows the user to set upper and lower limits on the control effort. If either limit is exceeded, an internal floating-point controller variable changes from 1 to 0. This variable is multiplied by the integrator input, such that the integrator accumulates zero data. In this way the integrator is frozen when the output is saturated, thus preventing the well-known “windup” phenomenon. Integration resumes when the controller output returns to its allowable range.

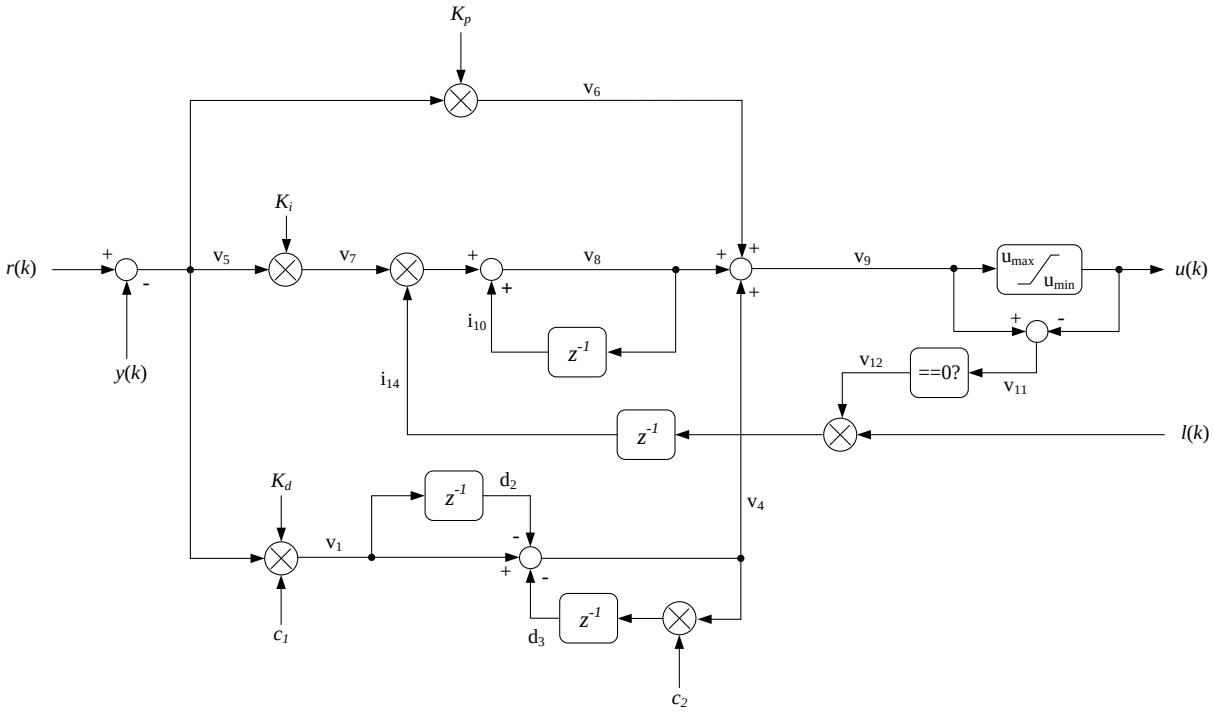

Figure 3. The DCL_PID_C4 controller

Some PID controllers in the library make provision for anti-windup reset to be triggered from an external part of the loop. This is useful in situations where a component outside the controller may be saturated. The floating-point variable “lk” is expected to be either 1.0 or 0.0 in the normal and saturated conditions respectively. Note that all the controllers here require non-zero proportional gain to recover from loop saturation.

4 Performance Measurement

Before commencing the tuning operation, the user should understand the various performance specifications which can be applied to the transient response[3,6]. A summary of the most common specifications is shown below.

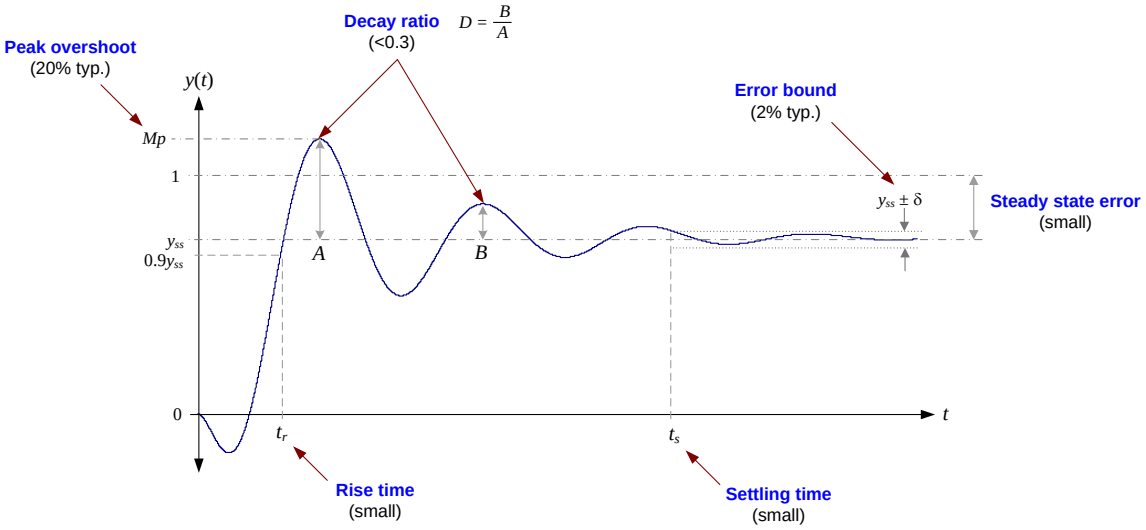

Figure 4. Summary of transient response specifications

A degree of optimality is possible in by assigning a performance index or cost function. Such functions are typically based on integrating the transient error over a fixed time interval. Various indices are in commons use. An example is the "ISE", which integrates the square of the servo error over a fixed interval following application of the transient stimulus. For a continuous time system with measurement interval T, the ISE index is

$$\int_0^T e^2(t)dt \qquad \ldots(5)$$

The lower the ISE value, the better the transient response. The DCL contains functions to capture and compute this and other performance indices. Refer to section 4.5 of the User's Guide for more information on the performance index methods and options.

5 Tuning the PID

This section suggests a manual tuning procedure for the linear PID controller. It applies to both continuous time and discrete time controllers when tuning against a transient response such as a step change of reference.

Several established procedures for fixing the controller gains based on plant response measurements can be found in the literature (e.g. Ziegler-Nichols[5], Cohen-Coon, etc.) but these are rarely used outside of process control. In power electronics, tuning is most often performed by iteratively adjusting one parameter at a time to optimize one or two performance objectives before moving to a different parameter. This process continues until the user judges that a satisfactory response has been achieved.

The following list of steps constitutes a procedure for manually tuning the PID controller described in section 3. It is assumed that the user understands the general features of the transient response (see section 4) and that realistic performance objectives have been set.

Step 1. Initialize the controller gains

Set the proportional gain (k_p) to a known safe initial value. Be sure to select a value significantly lower than the expected optimum value so that the gain can be safely increased without risk of the control loop becoming unstable. Ensure both the integral and derivative gains are set to zero.

Step 2. Initialize the control limits

Determine the physical range of control output and set appropriate limits at the output of the controller. This step is important to ensure anti-windup reset is effective. In some DCL implementations, it is possible to bring in limits from remote parts of the control system.

It can be instructive to monitor the saturation variable inside the controller. In DCL v3, each controller has a supporting structure (CSS) which contains a "testpoint" variable intended to bring out internal controller variables for this purpose.

Step 3. Configure the derivative filter

Oscillatory effects in the transient response can sometimes be reduced by applying derivative gain. To do this, first decide on a bandwidth for the derivative low-pass filter. The choice of filter bandwidth should be low enough to remove most of any sensor and process noise present, but not too low that the operation of the differentiator is compromised. More general guidelines are difficult to give and some degree of trial-and-error may be necessary. If filtering is not required, use infinite cut-off frequency (i.e. zero time constant).

Apply the equations in the DCL User's Guide[4] to find the derivative filter coefficients c1 & c2, and load these values into the PID controller structure.

Step 4. Apply a test stimulus

Apply a stimulus to the control loop in such a way as to induce an observable transient at the system output. In many cases the disturbance will be a sudden change in reference set-point, in others, a change in output load may be easier to apply. Observe the transient part of the output response. The DCL data logger utility will be useful for this task and code examples can be found in the library illustrating its use.

Step 5. Adjust the proportional gain

If the response does not meet specifications, adjust k_p and repeat the transient test. Repeat this step until the optimum value of k_p is found.

In general, if a steady state error is present increasing k_p will reduce (but not eliminate) it. Increasing k_p typically reduces the rise time (makes the response faster), but can introduce overshoot and oscillation into the response. Keep in mind that changes to k_p should be made in small steps, and that some systems will become unstable if the proportional loop gain exceeds some value. There may also be a lower limiting value of k_p which yields a stable response.

If a value of k_p can be found which meets all the performance objectives the tuning procedure can be terminated at this point.

Step 6. Adjust the integral gain

Depending on the nature of the control loop, steady state error can sometimes be eliminated with the introduction of integral control action. The effect of integral control depends on the open loop transfer function and the type of test stimulus applied[1,2,3]. In industrial applications the test stimulus is often a step change of reference input, and zero steady state error is achieved when at least one integrator is present inside the loop. For illustrative purposes, this is the situation we will assume here.

If necessary, gradually increase the integral gain term (k_i) to reduce steady state output error. Increasing the k_i increases the rate at which the response converges on steady state, but may introduce or amplify overshoot and oscillation. If this happens, it may be useful to slightly decrease k_p and then re-adjust k_i. Repeat this step until an optimum value of k_i is found. Again, if all performance specifications are met, terminate the procedure here.

Depending on the plant dynamics, the response may be very sensitive to the integral term and may become unstable, so be sure to start with a very small gain. In general, the faster the response of the plant, the greater will be the sensitivity of the control loop to integral gain.

Step 7. Adjust the derivative gain

Apply a small amount of derivative gain (k_d) and repeat the transient test. As with the other gains, changes should be made in small steps. Depending on the nature of the plant, the control may be strongly or weakly dependent on this parameter. In general, a system which is sensitive to k_i is insensitive to k_d and vice-versa.

If small amount of derivative action makes no significant difference it may be necessary to use progressively larger increments until a difference is seen. In general, the faster the plant the less sensitive is the closed loop response to derivative action.

It may be necessary to re-adjust k_p and k_i gains at this point. Typically, tuning involves repeated adjustments to the gain terms to find the best achievable response.

Step 8. Adjust the set-point weight

Some PID implementations in the DCL include a set-point "weight", k_r. Depending on the plant, it may be advantageous to weight the control error input to the proportional path differently from that of the integral path, and this is achieved by changing k_r. Examples include systems with time delay or right half plane zeros. The majority of control systems will not benefit materially from set-point weighting; however where improvement can be made the optimum value of k_r gain is typically a little less than unity.

Step 9. Repeat the procedure

At this point, the user will typically return to step 5 and re-adjust the proportional gain, again proceeding in small incremental steps. The introduction of integral or derivative action will change the optimum value of k_p and it is likely some improvement can be made. He or she will then move on to re-adjust k_i and k_d. This cyclic procedure continues until either the performance specifications have been met, or the user judges that no further improvement can be made.

In summary, the process above is nothing more than a series of adjustments of the three controller gains. The key points are that adjustments must be made in small steps, and that the use of a performance index provides a valuable, non-subjective indication of performance.

6 Tuning Example

To illustrate the above process, this section presents an example of transient tuning applied to a simple linear system. The plant is a model of a dynamical system having a third order transfer function. Sensor noise has been included in the simulation, and the control variable is limited by the ±15 V voltage swing of a power amplifier stage. The DCL_PID_C4 controller is used in a feedback control loop with 16-bit ADC and DAC resolution. A similar model (using Simulink) can be found in the DCL download.

We begin by setting the proportional controller gain to 1, and the integral and derivative gains to 0 (step 1). The output limits are loaded into the controller Umax & Umin parameters (step 2), and a bandwidth of 150 Hz used to compute the derivative filter parameters (step 3). A step input stimulus of amplitude 0.1 is applied and the following data plotted (step 4).

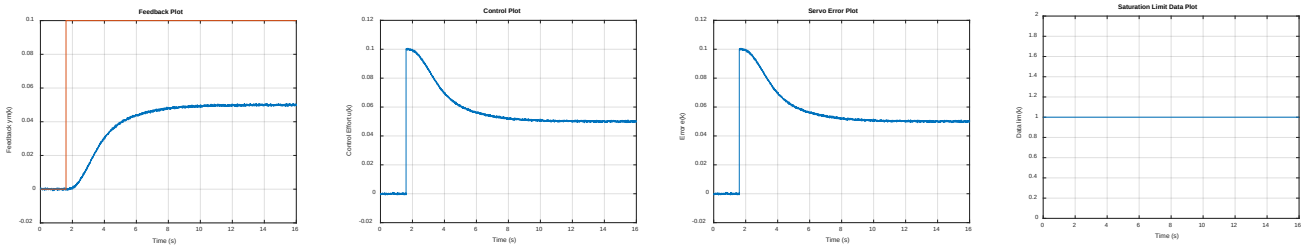

Figure 5. Response plots: kp = 1, ki = 0, kd = 0, IES = 5.2537

Clearly the response is quite slow and there is a steady state error of approximately 50%. The proportional gain is now increased steadily to the point where over-shoot reaches an unacceptable level.

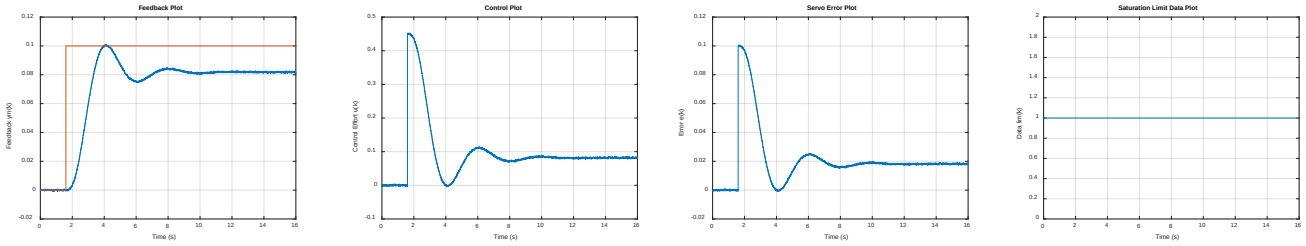

Figure 6. Response plots: kp = 4.5, ki = 0, kd = 0, IES = 1.3743

Observe that the rise time is now much shorter; the speed of the control system has increased. The IES index has fallen considerably; indicating much better transient performance, however there is now an undesirable over-shoot of approximately 20%. Although the steady state error is much reduced, k_p cannot be increased further without exacerbating the over-shoot.

We now direct attention to removing the steady state error through integral control action. The plots below show the result of changing k_i to 0.01.

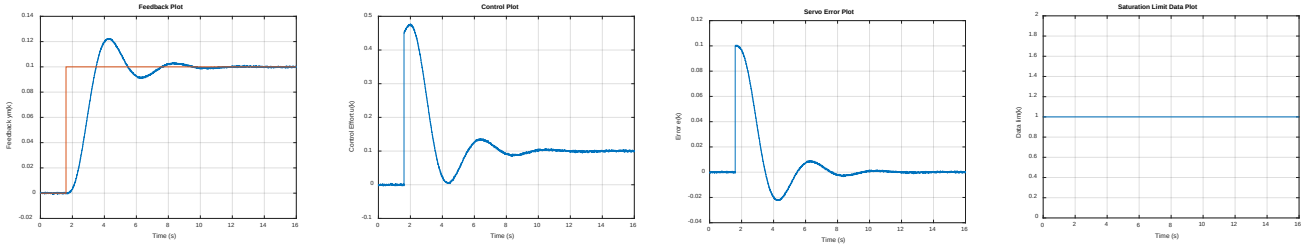

Figure 7. Response plots: kp = 4.5, ki = 0.01, kd = 0, IES = 0.9971

With this amount of integral control, steady state error is removed quite quickly; however the over-shoot has increased to about 30%. Derivative action is now added in an attempt to reduce this. The plots below show the effect of increasing k_d to 1.5.

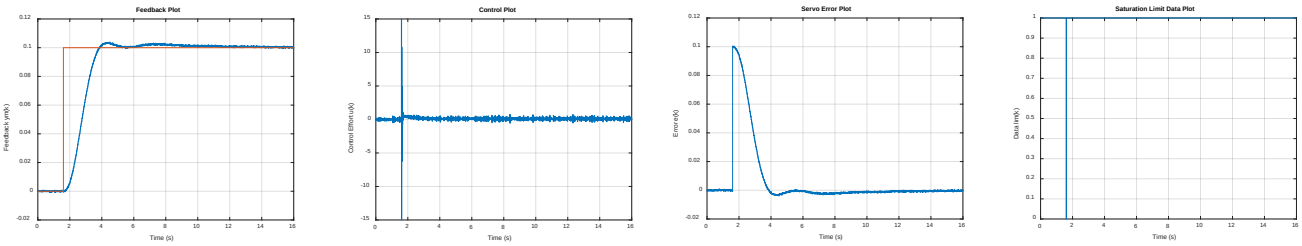

Figure 8. Response plots: kp = 4.5, ki = 0.01, kd = 1.5, IES = 0.8876

As expected, over-shoot has greatly diminished as a result of derivative action. However there is a large 'spike' in the control effort as the edge of stimulus step function propagates through the derivative path and this has caused the controller output to briefly saturate. This effect is known as derivative 'kick', and is visible in the control plot and in the brief excursion in the saturation plot. Although not serious here, derivative kick can be avoided by differentiating the feedback rather than the error, and this is done in the C1 & C2 PID controllers.

At this point the user might decide to re-adjust the proportional and integral gains in an attempt to further improve performance, although the return on the time invested is unlikely to be large.

7 References

[1] N. Minorsky, "*Automatic Steering Tests*", J. Amer. Soc. of Naval Engineers, pp. 285-310, Vol. 42, 1930.

[2] L. Desborough and R. Miller, "*Increasing customer value of industrial control performance monitoring - Honeywell's experience*". Sixth International Conference on Chemical Process Control. AIChE Symposium Series Number 326 (Vol. 98), 2002.

[3] R. Poley, "*Control Theory Fundamentals*", CreateSpace, 3rd Ed., 2015

[4] C2000 Digital Control Library v3.0, User's Guide

[5] Z. Ziegler and J.G. Nichols, "*Optimum settings for automatic controllers*", Transactions of the ASME. 64: 759–768, 1942

[6] Control Theory Fundamentals seminar, section 3: https://www.youtube.com/watch?v=9v1uK5wF3mA

[7] C2000Ware software package: http://www.ti.com/tool/c2000ware