ELEC ENG 4CL4:

Control System Design

Notes for Lecture #25

Friday, March 12, 2004

Dr. Ian C. Bruce
Room: CRL-229
Phone ext.: 26984
Email: ibruce@mail.ece.mcmaster.ca

Chapter 12

Models for Sampled Data Systems

Motivation

Up to this point in the book, we have assumed that the control systems we have studied operate in continuous time and that the control law is implemented in analogue fashion. Certainly in the early days of control, all control systems were implemented via some form of analogue equipment. Typically controllers were implemented using one of the following formats:-

- hydraulic
- pneumatic
- analogue electronic

However, in recent times, almost all analogue controllers have been replaced by some form of computer control.

This is a very natural move since control can be conceived as the process of making computations based on past observations of a system's behaviour so as to decide how one should change the manipulated variables to cause the system to respond in a desirable fashion.

The most natural way to make these computations is via some form of computer.

A huge array of control orientated computers are available in the market place.

A typical configuration includes:

- some form of central processing unit (*to make the necessary computations*)

- analogue to digital converters (*to read the analogue process signals into the computer*).

  (We call this the process of **SAMPLING**)

- digital to analogue converters (*to take the desired control signals out of the computer and present them in a form whereby they can be applied back onto the physical process*).

  (We call this the process of **SIGNAL RECONSTRUCTION**)

Types of Control Orientated Computer

Depending upon the application, one could use many different forms of control computer. Typical control orientated computers are:

DCS (Distributed Control System) These are distributed computer components aimed at controlling a large plant.

PLC (Programmable Logic Controller) These are special purpose control computers aimed at simple control tasks - especially those having many on-off type functions.

PC (Personal Computer) There is an increasing trend to simply use standard PC's for control. They offer many advantages including minimal cost, flexibility and familiarity to users.

Embedded Controller. In special purpose applications, it is quite common to use special computer hardware to execute the control algorithm. Indeed, the reader will be aware that many commonly used appliances (CD players, automobiles, motorbikes, etc.) contain special microprocessors which enable various control functions.

Why Study Digital Control?

A simple (*engineering*) approach to digital control is to sample quickly and then to make some reasonable approximation to the derivatives of the digital data. For example, we could approximate the derivative of an analogue signal, $y(t)$, as follows:

$$\frac{d}{dt}y(t) \approx \frac{y(t)-y(t-\Delta)}{\Delta}$$

where Δ is the sampling period.

The remainder of the design might then proceed exactly as for continuous time signals and systems using the continuous model.

Actually, the above strategy turns out to be quite good and it is certainly very commonly used in practice.

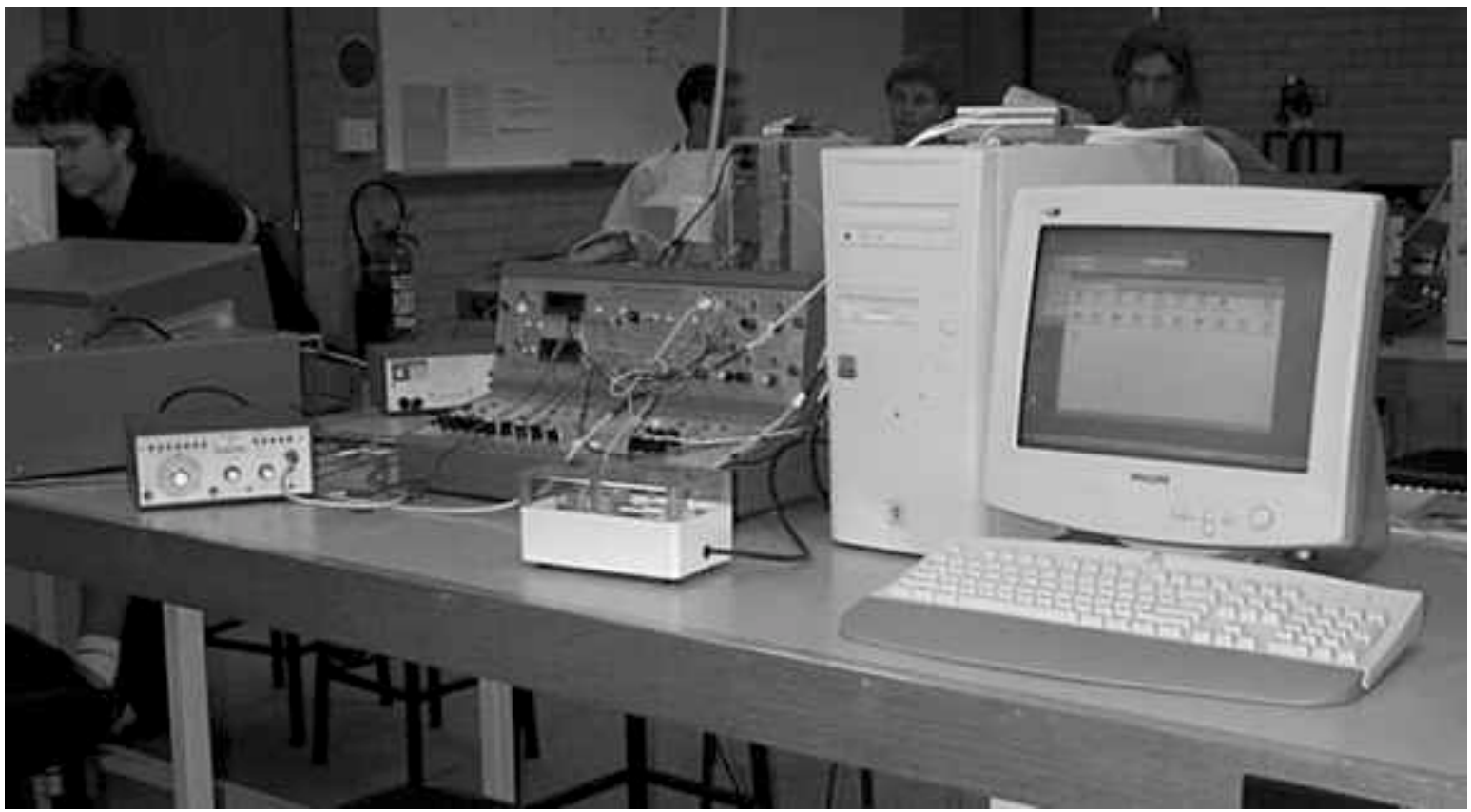
However, there are some unexpected traps for the unwary. These traps have lead to negative experiences for people naively trying to do digital control by simply mimicking analogue methods. Thus it is important to know when such simple strategies make sense and what can go wrong. We will illustrate by a simple example below.

D.C. Servo Motor Control

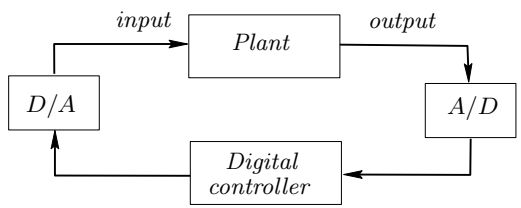
We consider the control of a d.c. servo system via a computer. This is a very simple example. Yet we will show that this simple example can (when it is fully understood) actually illustrate almost an entire course on control.

A photo of a typical d.c. servo system is shown on the next slide.

Photo of Servo Laboratory System with Digital Control via a PC



The set-up for digital control of this system is shown schematically below:



The objective is to cause the output shaft position, $y(t)$, to follow a given reference signal, $y^*(t)$.

Modelling

Since the control computations will be done inside the computer, it seems reasonable to first find a model relating the sampled output, $\{y(k\Delta);\ k = 0, 1, \ldots\}$ to the sampled input signals generated by the computer, which we denote by $\{u(k\Delta),\ k = 0, 1, \ldots\}$. (Here Δ is the sample period).

We will see later in this chapter that the output at time $k\Delta$ can be modelled as a linear function of past outputs and past controls. (We ask that the reader accept this for the moment).

Thus the (*discrete time*) model for the servo takes the form:

$$y\left(\overline{k+1}\Delta\right)=\bar{a}_1 y\left(k\Delta\right)+\bar{a}_0 y\left(\overline{k-1}\Delta\right)+\bar{b}_1 u\left(k\Delta\right)+\bar{b}_0 u\left(\overline{k-1}\Delta\right).$$

A Prototype Control Law

Conceptually, we want $y\left(\overline{k+1}\Delta\right)$ to go to the desired value y^*. This suggests that we could simply set the right hand side of the equation on the previous slide equal to y^*. Doing this we see that $u(k\Delta)$ becomes a function of $y(k\Delta)$ (as well as $y\left(\overline{k-1}\Delta\right)$ and $u\left(\overline{k-1}\Delta\right)$. At first glance this looks reasonable but on reflection we have left no time to make the necessary calculations. Thus, it would be better if we could reorganize the control law so that $u(k\Delta)$ becomes a function of $y\left(\overline{k-1}\Delta\right)$, … . Actually this can be achieved by changing the model slightly as we show on the next slide.

Model Development

Substituting the model into itself to yield:

$$
\begin{aligned}
y(\overline{k+1}\Delta) &= \bar{a}_1\{y(k\Delta)\} + \bar{a}_0 y(\overline{k-1}\Delta) \\
&\quad + \bar{b}_1 u(k\Delta) + \bar{b}_0 u(\overline{k-1}\Delta) \\
&= \bar{a}_1\{\bar{a}_1 y(\overline{k-1}\Delta) + \bar{a}_0 y(\overline{k-2}\Delta) \\
&\quad + \bar{b}_1 u(\overline{k-1}\Delta) + \bar{b}_0 u(\overline{k-2}\Delta)\} \\
&\quad + \bar{a}_0 y(\overline{k-1}\Delta) + \bar{b}_1 u(k\Delta) + \bar{b}_0 u(\overline{k-1}\Delta)
\end{aligned}
$$

We see that $y\left(\overline{k+1}\Delta\right)$ takes the following form:

$$y\left(\overline{k+1}\Delta\right)=\alpha_1 y\left(\overline{k-1}\Delta\right)+\alpha_2 y\left(\overline{k-2}\Delta\right)$$
$$+\beta_1 u\left(\overline{k}\Delta\right)+\beta_2 u\left(\overline{k-1}\Delta\right)+\beta_3 u\left(\overline{k-2}\Delta\right)$$

where $\alpha_1=\overline{a}_1^{\,2}+\overline{a}_0$ etc.

Actually, $\alpha_1, \alpha_2, \beta_1, \beta_2, \beta_3$, can be estimated from the physical system. We will not go in to details here. However, for the system shown earlier the values turn out to be as follows for $\Delta = 0.05$ seconds:

$$\alpha_1 = 0.03554$$

$$\alpha_2 = 0.03077$$

$$\beta_1 = 1$$

$$\beta_2 = -1.648$$

$$\beta_3 = 0.6483$$

A Modified Prototype Control Law

Now we want the output to go to the reference y^*.

Recall we have the model:

$$y(\overline{k+1}\Delta) = \alpha_1 y(\overline{k-1}\Delta) + \alpha_2 y(\overline{k-2}\Delta)$$
$$+ \beta_1 u(\overline{k}\Delta) + \beta_2 u(\overline{k-1}\Delta) + \beta_3 u(\overline{k-2}\Delta)$$

This suggests that all we need do is set $y\left(\overline{k+1}\Delta\right)$ equal to the desired set-point $y^*\left(\overline{k+1}\Delta\right)$ and solve for $u(k\Delta)$. The answer is

$$u(k\Delta) = \frac{y^*(\overline{k+1}\Delta) - \alpha_1 y(\overline{k-1}\Delta) - \alpha_2 y(\overline{k-2}\Delta) - \beta_2 u(\overline{k-}\Delta) - \beta_3 u(\overline{k-2}\Delta)}{\beta_1}$$

Notice that the above control law expresses the current control $u(k\Delta)$ as a function of

- the reference, $y^*(\overline{k+1}\Delta)$
- past output measurements, $y(\overline{k-1}\Delta), y(\overline{k-2}\Delta)$
- past control signals, $u(\overline{k-1}\Delta), u(\overline{k-2}\Delta)$

Also notice that 1 sampling interval exists between the measurement of $y\left(\overline{k-1}\Delta\right)$ and the time needed to apply $u(k\Delta)$; i.e. we have specifically allowed time for the computation of $u(\mathrm{k}\Delta)$ to be performed after $y\left(\overline{k+1}\Delta\right)$ is measured!

Recap

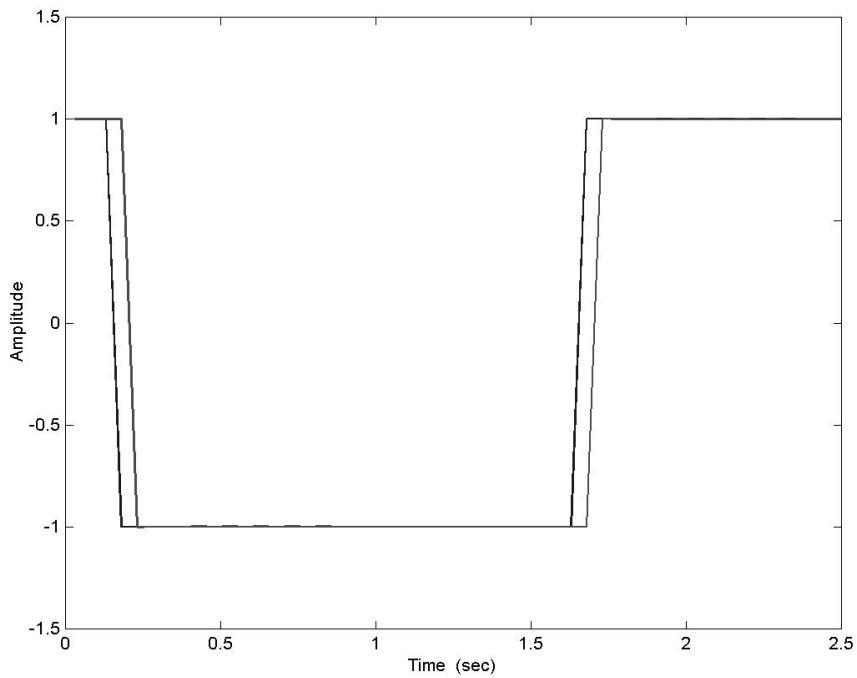
All of this is very plausible so far. We have obtained a simple *digital* control law which causes $y\left(\overline{k+1}\Delta\right)$ to go to the desired value $y^*\left(\overline{k+1}\Delta\right)$ in one step !

Of course, the real system evolves in continuous time (*readers may care to note this point for later consideration*).

Simulation Results

To check the above idea, we run a computer simulation. The results are shown on the next slide. Here the reference is a square wave. Notice that, as predicted, the output follows the reference with a delay of just 2 samples.

Simulation Results with Sampling Period 0.05 seconds



Experimental results

However, when we try this on a real system, the results are extremely poor! Indeed, the system essentially goes unstable.

- Can the reader guess some of the causes for the difference between the ideal simulation results and the very poor real results?

Causes of the Poor Response

It turns out that there are many reasons for the poor response. Some of these are:

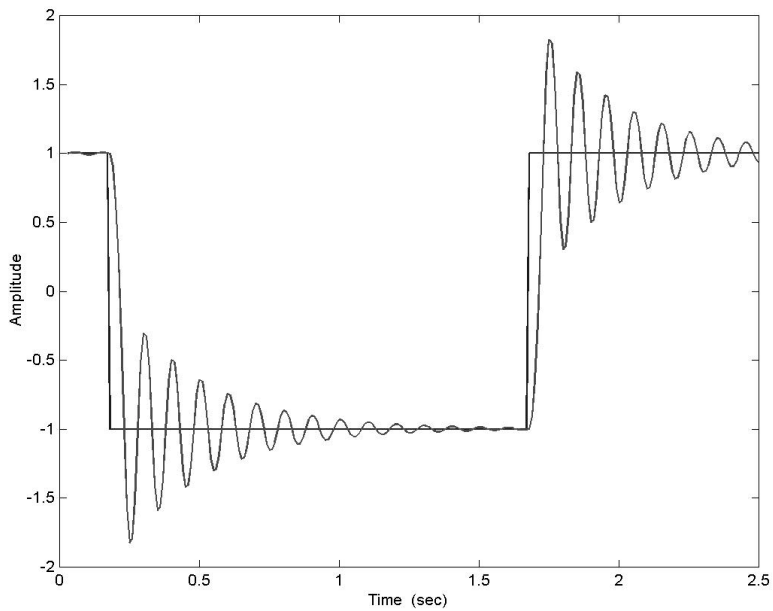
1. *Intersample issues*
2. *Input saturation*
3. *Noise*
4. *Timing jitter*

The purpose of this chapter and the following two chapters is to understand these issues. To provide motivation for the reader we will briefly examine these issues for this simple servo example.

1. Intersample Issues

If we look at the output response at a rate faster than the control sampling rate then we see that the actual response is as shown on the next slide.

Simulation result showing full continuous output response



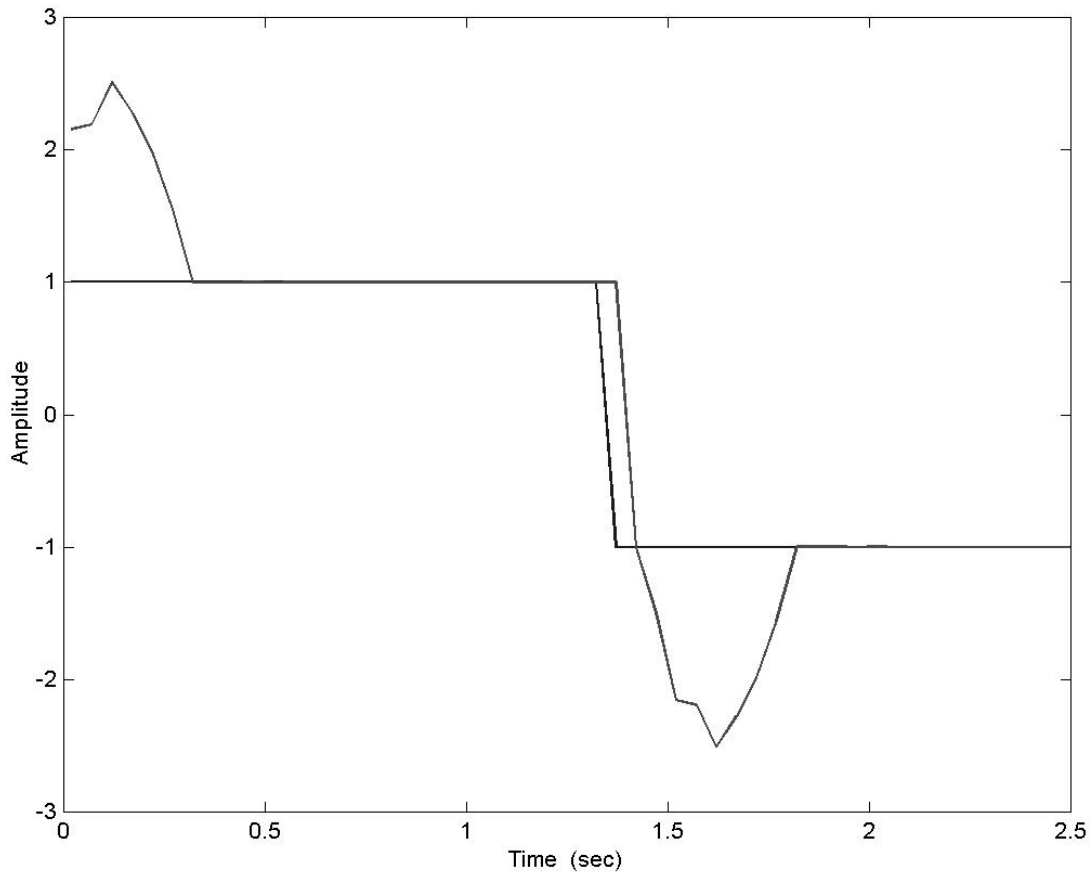
This is rather surprising! However, if we think back to the original question, we only asked that the sampled output go to the desired reference. Indeed it has. However, we said nothing about the intersample response!

A full explanation of this phenomenon will be given in Chapter 14.

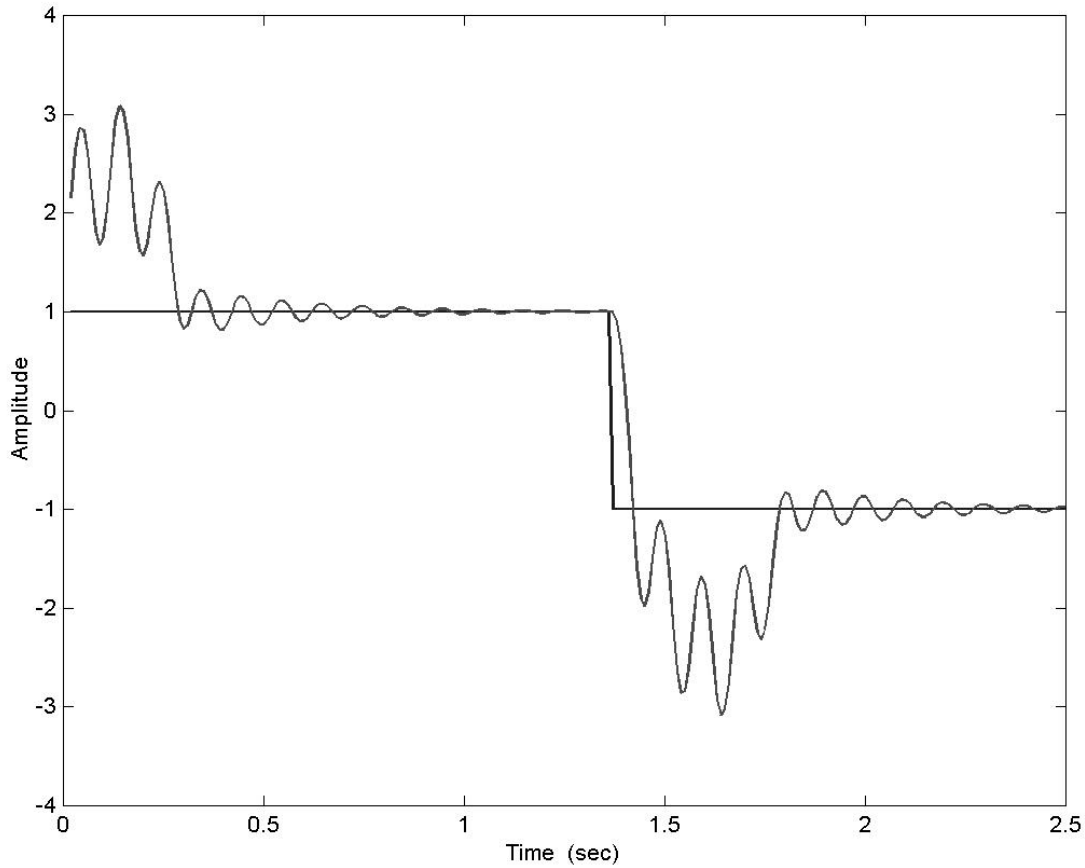
2. Input Saturation

Looking again at the simulations, we see that this particular control law is calling for very large input signals. However, the D/A converter on the real servo kit only operates on a range of ±10 volts.

We thus repeat the simulation but clamp the voltage at ±10 volts. The result is an unstable response. Indeed, clamping at ±100 volts still gives very poor results as shown on the next slide.

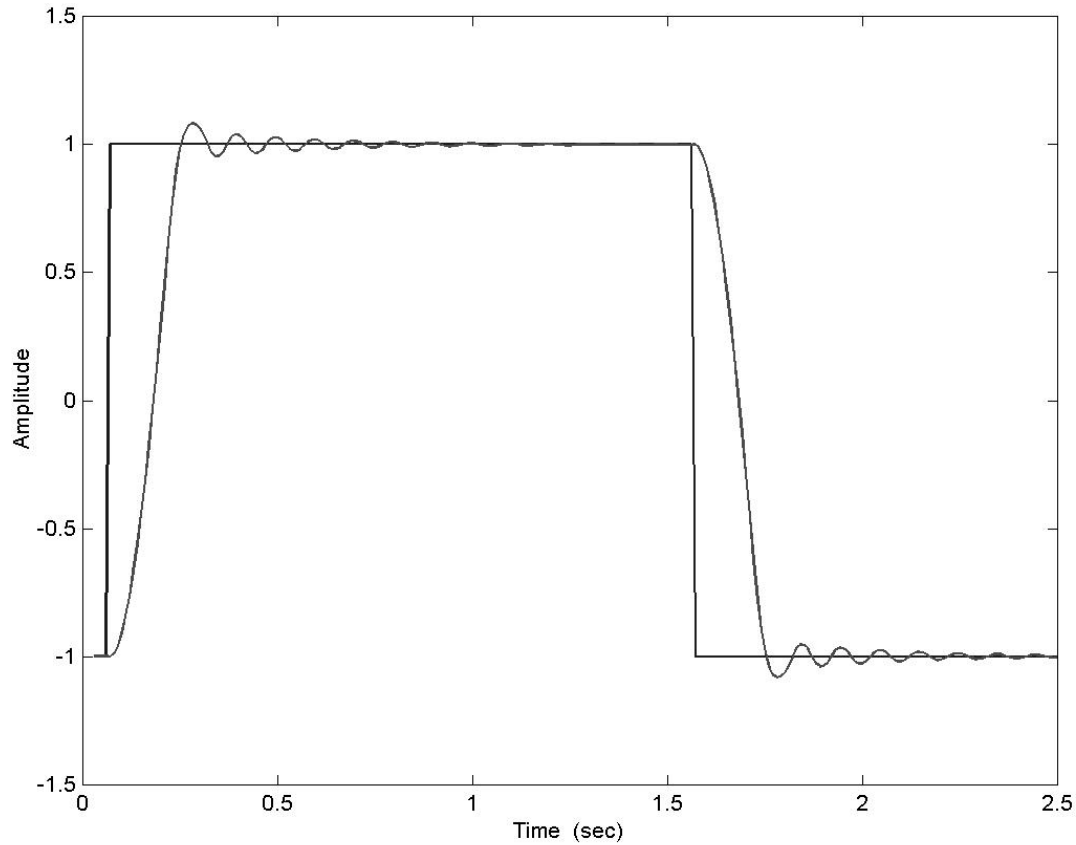

3
2
1
0
-1
-2
-3
Amplitude
0
0.5
1
1.5
2
2.5
Time (sec)

Looking between the samples reveals even more structure to the result shown above. See the next slide.


4
3
2
1
0
-1
-2
-3
-4
Amplitude
0
0.5
1
1.5
2
2.5
Time (sec)

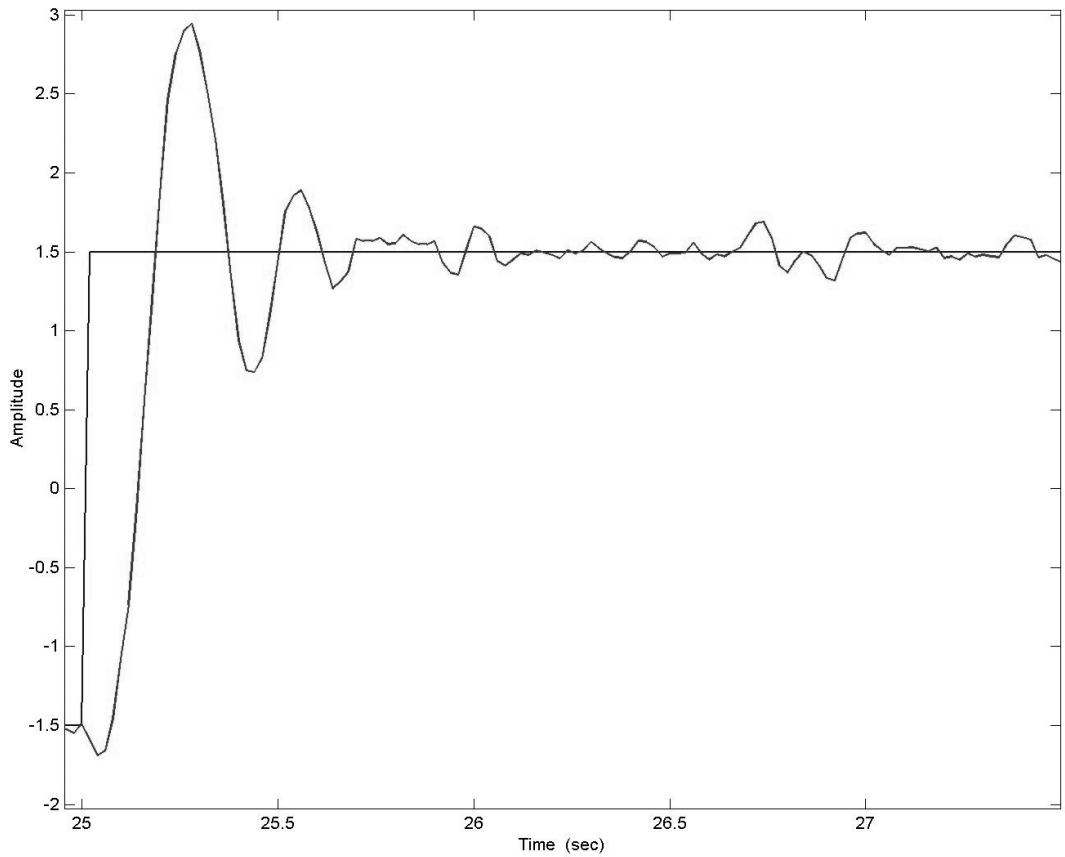
The reader may recall that we studied windup and input saturation in Chapter 11. Maybe we should try the ideas presented in Chapter 11 here.

We recall from Chapter 11 that the essential trick in anti-windup schemes is to ensure that the states of the control law are *told* that the input has saturated. This means that all we need do is to ensure that the saturated past input signals are stored in the computer to be used in subsequent control law calculations. Making the test gives the result on the next slide.


1.5
1
0.5
0
-0.5
-1
-1.5
Amplitude
0
0.5
1
1.5
2
2.5
Time (sec)

Experimental results revisited

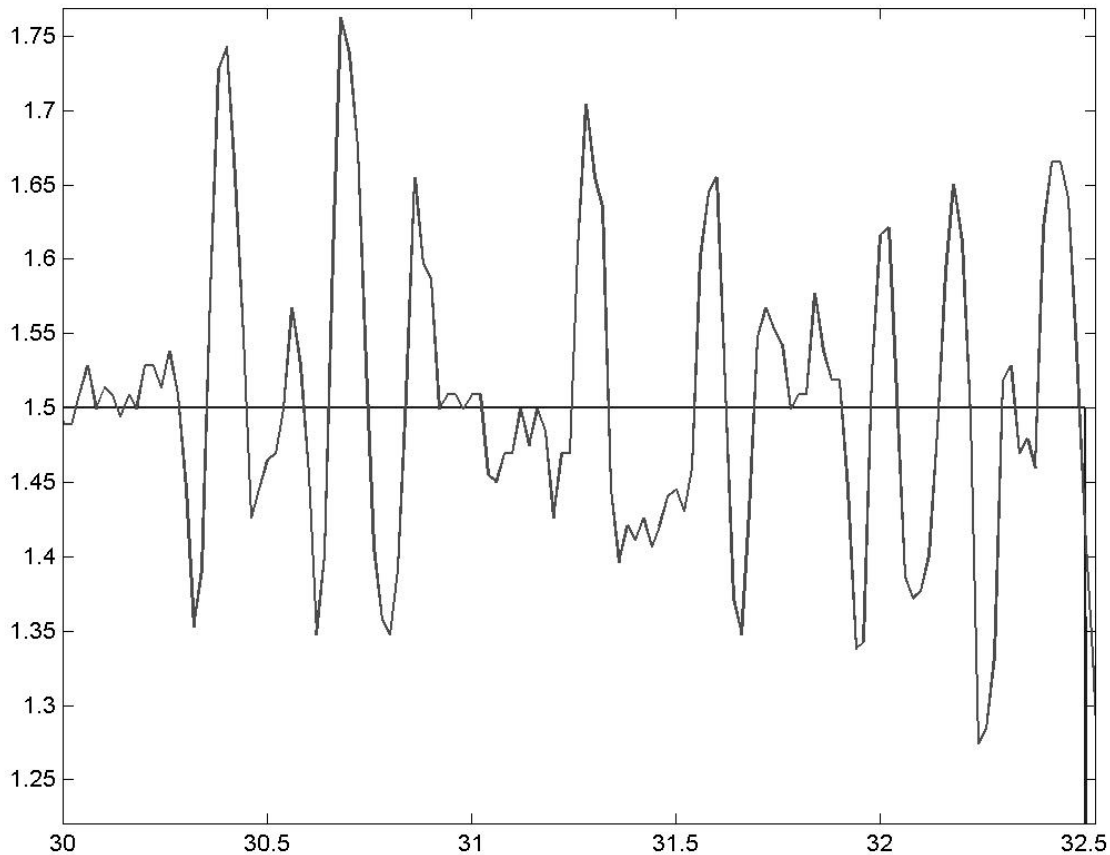
Going back to the real servo kit and applying the above idea with anti-windup protection at ±10 volts gives the results on the next slide.


3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Amplitude
25
25.5
26
26.5
27
Time (sec)

We see that we now, at least, have achieved stable operation. However, the results are nowhere near as good as those predicted by simulation.

3. Noise

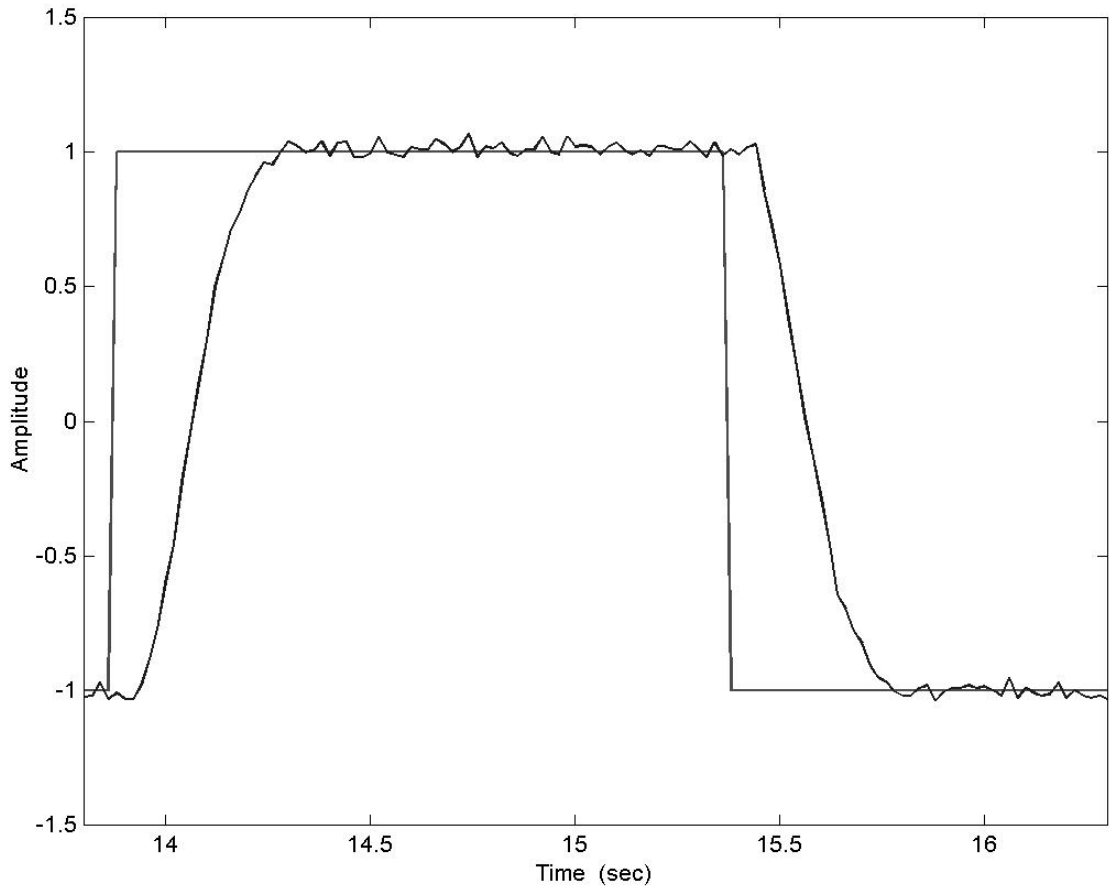
One further point that we have overlooked is that causing $y(t)$ to approach y^* as quickly as possible gives a very wide bandwidth controller. However, we saw in Chapter 8 that such a controller will necessarily magnify noise. Indeed, if we look at the steady response of the system (*see the next slide*) then we can see that noise is indeed causing problems.


1.75
1.7
1.65
1.6
1.55
1.5
1.45
1.4
1.35
1.3
1.25
30
30.5
31
31.5
32
32.5

4. Timing Jitter

Finally we realize that this particular real controller has been implemented in a computer that does not have a real-time operating system. This means that the true sampling rate actually varies around the design value. We call this *timing jitter*. This can be thought of as introducing modelling errors. Yet we are using a wideband controller. Thus, we should expect significant degradation in performance relative to the idealized simulations.

Finally, we make a much less demanding design and try a simple digital PID controller on the real system. The results are entirely satisfactory as can be seen on the next slide. Of course, the design bandwidth is significantly less than was attempted with the previous design.


1.5
1
0.5
0
-0.5
-1
-1.5
Amplitude
14
14.5
15
15.5
16
Time (sec)

Hopefully the above example has motivated the reader to say - "*let's study digital control*".

By the time you have studied the next three chapters you will understand all of the features of the above simple problem, e.g.

- how to build the model;
- what are the special features of the one-step-ahead control law we have used; and
- why funny things can (*and sometimes do*) happen between samples.