CHAPTER 13. A FEW ELECTRONIC IMPLEMENTATION ISSUES

In this book a good deal of effort has been directed at describing what is required to attain active attenuation of some unwanted disturbance, which includes deriving a number of algorithms for control system design. In this chapter we will consider some of the issues associated with implementation of some of these algorithms in an electronic (digital) controller. This consideration will cover both what is physically required in terms of hardware, and performance-related issues in hardware selection. The discussion will be largely qualitative, as there are no fixed rules for optimising hardware arrangements. The discussion will also be brief, concentrating on a few issues which are commonly encountered in active control system implementation. A detailed discussion of hardware issues is beyond the scope of this book, and would probably be out of date by the time it is read.

The discussion here will be largely directed at the implementation of feedforward control systems, as it is these systems which have been of principal interest in this book. The hardware requirements for feedforward and feedback implementations are, for the most part, the same. What will be lacking is discussion of strictly feedback implementation issues, such as integrator windup. For discussion of strictly feedback control issues, the reader is directed towards any good digital control text, such as Middleton and Goodwin (1990) and Franklin et al (1990).

Figure 13.1 here

Illustrated in figure 13.1 is a block diagram of the main components of a digital active control system. As was outlined in chapter 5, digital implementation of control systems involves the inclusion of a number of components to the standard "textbook" control arrangement. These additions include an anti-aliasing filter, sample and hold circuitry, and an analog to digital converter (ADC) on the input to the controller, the inclusion of a digital to analog converter (DAC), sample and hold circuitry, a reconstruction filter on the output of the controller, and the addition of a clock to synchronize events such as sampling. Referring to figure 13.2, for the purposes of this chapter, this arrangement can be separated into two parts: the analog / digital interface, and the main controller "body", on which the control algorithms will run. This body is typically a microprocessor. Discussion in this chapter will also be separated along the lines of figure 13.2: design of the analog to digital interface, and implementation of active control software on a microprocessor.

Figure 13.2 here

13.1 THE ANALOG / DIGITIAL INTERFACE

Recall from chapter 5 that the ADCs and DACs provide an interface between the real (continuous) world and the world of a digital system. ADCs take some physical variable, usually an electrical voltage, and convert it to a stream of numbers which are sent to controller for use in the control algorithm. These numbers usually arrive at increments of some fixed time period, or sampling period. The numbers arriving from the ADC are usually representative of the value of the signal at the start of the sampling period, as the data input to the ADC is normally sampled and then held constant during the conversion process to enable an accurate conversion, as will be discussed shortly. Commonly, the sampling period is implicitly referred to by a sampling rate, which is the number of samples taken in one second.

The digital signal coming from the ADC is quantised in level. This simply means that the stream of numbers sent to the digital control system has some finite number of digits, hence finite accuracy. This accuracy is normally quantified by the number of bits used by the ADC to represent the measured signal. For example, a 16 bit ADC converter will represent a sampled physical system variable as a set of 16 bits, each with a value of 0 or 1. It follows that the accuracy of the digital representation of the analog value is limited by the "quantum size", given by

$$quantum\ size = \frac{full\ scale\ range}{2^n}, \qquad (13.1.1)$$

where n is the number of bits. For example, if the full scale range of the ADC is ±10 volts, the accuracy of the 16 bit digital representation is limited to better than (20 volts)/(2^{16})=0.305 millivolts. The difference between the actual analog value and its digital representation is referred to as the quantisation error. The dynamic range of the ADC is also determined by the number of bits used to digitally represent the analog value, and is usually expressed in decibels, or dB. For example, a 16 bit ADC has a dynamic range of (20 log (2^{16})), or 96.3, dB.

(One point which should be mentioned here is that the "ideal" dynamic range of a converter, as defined above, is often significantly less than the effective dynamic range which is achieved in implementation. This is partly because it is unlikely that the high end of the voltage range is consistently used, rendering some of the more significant bits useless. Also, measurement noise often corrupts the signal, effectively rendering one or more of the less significant bits ineffective.)

The DAC works in an opposite fashion to the ADC in the sense that it provides a continuous output signal in response to an input stream of numbers. This continuous output is achieved using the sample and hold circuit, normally

incorporated "on-chip". This circuit is designed to progressively extrapolate the output signal between successive samples in some prescribed manner. The most commonly used hold circuit is the zero order hold, which simply holds the output voltage constant between successive samples. With the zero order hold circuit the output of the DAC / sample and hold circuitry is continuous in time, but quantised in level. To smooth out this pattern, a reconstruction filter is normally placed at the output of the DAC / sample and hold circuitry. This filter is low pass, which has the effect of removing the high frequency "corners" from the stepped signal.

There are a wide range of variables associated with the design of the analog / digital interface. Discussion here will concentrate on the most general parameters, such as sample rate selection, and avoid package-specific issues such as how the data is made available to the microprocessor (parallel versus serial chips). Further discussion of these issues can be found in application notes available from most component manufacturers, such as Analog Devices.

13.1.1. Sample rate selection

Figure 13.3 here

The first issue of interest here is selection of a suitable sampling rate. An absolute limit on the lower value of the sampling rate can be derived by studying the effect of sampling on the (continuous) system transfer function. To do this, it is useful to consider the simplified ADC / DAC arrangement shown in figure 13.3, comprising two parts: the sampler, and the hold. The signal coming from the sampler can be viewed as a set of impulses, or an impulse train,

$$x^*(t) = \sum_{k=-\infty}^{\infty} x(t)\ \delta(t-kT), \tag{13.1.2}$$

where T is the sampling period, taken here to be fixed. The Laplace transform of this is

$$\mathcal{L}\{x^*(t)\} = x^*(s) = \sum_{k=-\infty}^{\infty} x(kT)e^{-skT}. \tag{13.1.3}$$

For the zero-order hold considered here the output at time t is equal to

$$x_n(t) = x(kT) \qquad kT \le t < kT+T. \tag{13.1.4}$$

This can be expressed in terms of a unit step function as

$$x_n(t) = x(kT)(1(t)-1(t-T)), \tag{13.1.5}$$

which enables the Laplace transform to be calculated

$$\mathcal{L}\{x_n(t)\} = x_n(s) = \mathcal{L}\{x(kT)\}\frac{1-e^{-sT}}{s}. \tag{13.1.6}$$

Combining the sample of equation (13.1.3) and the hold of equation (13.1.6), the continuous time transfer function of the sample and hold is

$$y(s) = \sum_{k=-\infty}^{\infty} x(kT)e^{-skT}\frac{1-e^{-sT}}{s} \tag{5.2.7}$$

Therefore, the digital control system can be modelled as in block diagram form as shown in figure 13.4.

Figure 13.4 here

Consider now the frequency domain representation of equation (13.1.2). The impulse train can be re-expressed as a Fourier series

$$\sum_{k=-\infty}^{\infty} \delta(t-kT) = \sum_{n=-\infty}^{\infty} h_n e^{j(\frac{2\pi n}{T})t}, \tag{13.1.8}$$

where the coefficients H_n are found by integrating over a single sampling period,

$$h_n = \frac{1}{T}\int_{-T/2}^{T/2} \delta(t)e^{-j\omega_s t}\,dt. \tag{13.1.9}$$

Here ω_s is the sampling frequency in rad s^{-1}, defined by

$$\omega_s = \frac{2\pi}{T}. \tag{13.1.10}$$

Evaluation of this integral results in

$$h_n = \frac{1}{T} \tag{13.1.11}$$

Therefore, the impulse train can be expressed in frequency domain format as

$$\sum_{k=-\infty}^{\infty} \delta(t-kT) = \frac{1}{T}\sum_{n=-\infty}^{\infty} x(s-jn\omega_s) \qquad (13.1.12)$$

Substituting this result back into equation (13.1.2), and taking the Laplace transform, it is found that

$$x^*(s) = \frac{1}{T}\sum_{n=-\infty}^{\infty} x(s-jn\omega_s). \qquad (13.1.13)$$

The significance of this result of equation (13.1.13) is that it states that images of the true value of the sampled spectrum repeat themselves at infinite numbers of intervals of ω_s. This phenomena is termed aliasing.

Figure 13.5 here

Practically, the phenomena of aliasing means that it is impossible to tell the difference between two (or more) sinusoids based upon the sampled signal. This effect is illustrated in figure 13.5, where two sinusoids have exactly the same sampled values, and can therefore not be distinguished from one another based upon the sampled data. Aliasing can have a significant detrimental effect upon control system performance if there are substantial levels of "high" frequency data, $\omega>\omega_s/2$, which are allowed to alias onto the "low" frequency data, $\omega<\omega_s/2$. To combat this problem, anti-aliasing filters placed in front of the input to the sample and hold / ADC arrangement. These are analog low-pass filters which remove frequency components greater than half the sampling frequency, $\omega>\omega_s/2$, from the input spectrum.

From the above discussion, it can be surmised that the absolute lower value of sampling rate for a given problem is twice the highest frequency of interest. However, actually implementing a system with this sampling rate is not advisable. First, while it is theoretically possible to reconstruct a harmonic signal sampled at twice its frequency, the filter required to do so is of infinite length, and not BIBO stable (see the discussion of Shannon's reconstruction theory in Middleton and Goodwin (1990)). Second, there is no margin for error in the upper frequency limit; any slight change in upper frequency results in aliasing.

So what is a good sampling rate? It was noted in chapter 6 that for tonal excitation, synchronising the sampling to be at four times the excitation has some beneficial effects from the standpoint of adaptive algorithm performance (the possibility of a uniform error surface, or equal eigenvalues, which leads to the "best"

compromise between algorithm speed and stability). In general, however, we cannot expect to be able to synchronise the sample rate of the controller with the unwanted disturbance, even if it is harmonic.

To paint a qualitative picture of a "good" sampling rate, consider the problem of sampling a step response of a system. If, for ease of computation, we assume the system has a bandwidth of 1 Hz, then the continuous signal will be defined by

$$y(t) = 1 - e^{-2\pi t}. \qquad (13.1.14)$$

Figure 13.6 here

Referring to figure 13.6, if the step response is sampled at 2 Hz, the step is indistinguishable to the viewer. Sampled at 5 Hz, the characteristics begin to appear. At 10 Hz, the step is apparent. In fact, if the samples are connected with straight lines, the reconstruction of the step is in error by less than 4% (Middleton and Goodwin, 1990). Intuitively then we can postulate that the filtering exercise, which is analogous the reconstruction of a signal, becomes "easier" as the sampling rate increases.

There is, however, a limit to this process. At high sampling rates, tens or even hundreds of times the disturbance frequency, there are numerical problems. In chapter 6 it was shown that as the level of oversampling increased the disparity in the eigenvalues of the autocorrelation matrix of the input samples increased. These eigenvalues to a large extent govern the convergence behaviour of the algorithm. A large disparity in eigenvalues generally translates into slow algorithm convergence, and reduced algorithm stability. Fast sampling also enhances the numerical inaccuracies associated with finite word length (integer) calculations (Middleton and Goodwin, 1990).

Based on the above discussion, it can be surmised that the optimum sampling rate is a compromise between fast and slow; both of these extremes lead to problems with adaptive algorithm convergence and, especially with fast sampling, stability. The "optimum" sampling rate compromise is often sited as ten times the frequency of interest. In practise, this sampling rate provides for rapid convergence of the adaptive algorithm and reasonable levels of stability. In implementing active control systems we often find that for a given sampling rate ω_s the system will work reasonably well from frequencies approaching $\omega_s/100$ to frequencies up to $\omega_s/4$. On the low end of the scale, adaptation of the controller with frequencies below $\omega_s/100$ is often (extremely) slow, and not a particularly stable operation. While this is sometimes improved by increasing the length of the digital (control) filter, the only real solution is a reduction in sampling rate. On the high end of the scale, the

adaptive algorithm appears ineffective with excitation frequencies above $\omega_s/4$.

13.1.2. Converter type and group delay considerations

A second important variable in the design of analog / digital interface is the converter type, a selection which is influenced by group delay considerations. For our discussion here, group delay is a measure of the time it takes for a signal to pass from the input of the analog / digital interface to the output. For analog to digital conversion, what is of interest is the group delay through the anti-alias filters and the ADC. For digital to analog conversion, what is of interest is the group delay through the DAC and the reconstruction filters.

Before being a discussion of factors related to group delay, it is necessary to understand why group delay is important. Group delay has a significant influence upon (at least) three variables in active control system design: physical size, level of control, and stability. Physical size is most obviously influenced by group delay in a causal feedforward active control system, such as a system designed to control (broadband) sound propagation in an air handling duct. If an upstream microphone is being used to supply a reference signal to the active control system, the group delay will limit the proximity of the reference sensor and control source, and hence the physical size of the system. If the group delay through each converter and filter is a few milliseconds, then for sound travelling at 343 ms^{-1} to be controlled the active control system must be at least two meters in length (once sensed, the sound will travel two meters by the time the controlling disturbance is introduced into the system). For compact systems, group delay minimisation is desirable.

If a feedback control system is used, the effect of group delay is slightly different. Firstly, group delay through the analog / digital interface will be responsible for a phase shift in the signal provided to the controller; the phase of the signal provided to the control at time *t* will not be the same as the signal at the sensor at time *t*. If at all significant (say, $>10^\circ$) this phase shift must be taken into account at the time of controller design. Even if it is taken into account, group delay often reduces the "average" level of control which is achieved. This is because feedback systems are often implemented to attenuate the reverberant response of a system. If it take "longer" to sense the disturbance, it takes longer to control it. Averaged over the operating cycle of the system, this results in reduced levels of attenuation with longer group delays.

Group delay can also influence the stability of adaptive algorithms used in

feedforward active control systems. As is a finite time delay between the derivation of a control signal and its "appearance" in the error signal, there is a delay between a change in filter weights and its effect being measured. As was discussed in chapter 6, when the weight adaptation is done at time intervals shorter than this delay the stability of the adaptive algorithm is reduced. With long group delays this is often the case.

In active control system implementation, perhaps the two most common types of ADCs employed are successive approximation converters and sigma delta converters. Successive approximation converters, which are probably the most common used at the time of writing this book, function by a sequence of comparisons between the measured signal and known voltage levels. These converters often require a separate sample and hold amplifier to "capture" the signal for conversion, and external anti-aliasing filters. They range widely in price, but can be relatively expensive for highly accurate devices. The group delay through the converter (governed by the conversion time) can vary, but is often significantly less than the group delay through the anti-alias filter which precedes it.

The group delay through the anti-alias filter is influenced by a number of variables. The two most significant are the number of poles in the filter and the cutoff frequency. Increasing the number of poles, and decreasing the cutoff frequency, both increase the group delay in an approximately linear fashion.

Sigma delta converters are functionally different than successive approximation converters, and are often significantly cheaper. These converters sample internally at a rate significantly faster than the "observed" sampling rate, such as 128 times per delivered sample. An internal conversion with an accuracy of only 1-bit is followed by a "decimation" filtering process to derive the final sampled signal. Because of the high internal sampling rates, the anti-aliasing filter cutoff can be an extremely high frequency. This means that a single pole filter can be used, which is usually on-chip. The high sampling rate also means that the sample and hold amplifier is not required. The overall result is often a (significantly) cheaper converter with no additional (sample and hold and anti-aliasing filter) circuitry required.

Sigma delta converters also have some negative features. The most significant of these is group delay. The filtering process inherent in the sigma delta conversion technique requires time. At the time of writing this book, the time was typically 30 samples at the nominal (no internal) sampling rate. Therefore, if a sampling rate of 5000 Hz was used, the group delay on the input to the controller is in the order of 5-6 milliseconds. This is significantly longer than the group delay which could be expected using a successive approximation converter and an anti-

aliasing filter with a cutoff at 2000 Hz. For harmonic excitation this is not a problem, as the controller can be non-causal (if an adaptive algorithm is used stability will be reduced, but this can often be compensated for). However, for a causal system this may be unacceptable.

13.2. MICROPROCESSOR SELECTION

In a digital active control system implementation, the actual control algorithms, including the control filters, will run on a microprocessor of some type. Advances in microprocessor technology are coming at such a rapid pace that it is somewhat pointless to outlined specific "chips", as by the time this book is published it is likely that the components will be out of date. It is no coincidence that interest in active noise control has paralleled advances in microprocessor technology; it is these advances which has made active noise and vibration control practically realisable. As microprocessor technology continues to improve, the applicability of active control will expand.

In this section we will briefly describe the types of microprocessors which are most commonly used at this stage, and the differences between them. Specific brands will not be discussed, only broad categories. The three types of microprocessors of interest here are "general" microprocessors, digital signal processors, and microcontrollers. In the future it is quite possible that the differences which separate these types of microprocessors will begin to evaporate, and the chip "classes" described here will begin to merge as the technology advances.

"General" microprocessors, for the purposes of the discussion here, refer to microprocessors which have large instruction sets and can do a wide range of tasks. Such a microprocessor can be viewed as a "jack of all trades but master of none". There are several classes of chips which fall into this category. Perhaps the most common is a complex instruction set computer, or CISC chip. These are general purpose microcomputers which have assembly language instruction sets with complex instructions which take several microprocessor clock cycles to implement. These are typically the microprocessors found in personal computers.

A more advanced version of this type of chip is the reduced instruction set computer, or RISC chip. RISC chips have only a simple instruction set, where each instruction can be implemented in one cycle of the microprocessor clock. While they are typically faster than their CISC cousins, RISC chips tend to be more expensive and more complex. Their compilers are also more complex, as all complex routines must be broken down into a series of simple instructions.

In active control implementations, general purpose microprocessors often have the disadvantage that they are relatively slow to do an important task: multiply. It is not uncommon for a general purpose microprocessor to take tens of clock cycles to do a single multiplication. When implementing digital filters this can be a serious problem, greatly limiting the length of filters which can be implemented. They do, in general, have greater flexibility than the other two types of microprocessors being discussed here. This alone, however, is usually not enough to make them the most attractive option.

Digital signal processors (DSPs) are microprocessors with architectures which are optimised for digital signal processing and numeric computation. They can typically perform a multiply and accumulate (add) operation in a single microprocessor clock cycle. This makes them extremely attractive for active control system implementation, which relies heavily on multiplications and additions (the functions of a digital filter). These microprocessors tend not to support "other" operations, such as division, as well as general purpose microprocessors, but are often still the microprocessor of choice.

Microcontrollers are essentially complete computer systems on one chip. They come with a wide variety of on-board options, including ADCs and DACs. They are also very cheap, commonly finding use in mass-produced products such as whitegoods. They are, however, lacking in some of the functions important in active control. They generally has less accuracy than the other devices discussed here, being, for example, 8 bit devices instead of 16 or 32 bit devices (which the other devices typically are). The ADCs and DACs are also typically 8 bits. Microcontrollers are also slow to multiply, as they lack the specialised architecture of the DSP. Their main asset is low cost, although at the time of writing this book DSPs were rapidly approaching the cost of microcontrollers in quantity.

13.3 SOFTWARE CONSIDERATIONS

Once a microprocessor and analog / digital interface has been chosen, some though must be given to the layout of the software. The issues of interest here are division of the control code, language use, and the choice of fixed or floating point implementation.

In a typical adaptive feedforward active control system, the controller software can be divided into two parts: that which must be done in real time (at each sample period), and that which can be done at a rate slower than real time (over several sample periods). Typically, calculation of the control filter output(s) must be done in real time, while all other operations can be done "off-line". Real time code is typically "interrupt driven". With this, when a data sample is ready, a signal is sent to the chip which causes it to stop what it is doing and go perform a specified operation (in this instance, calculation of the filter output). Splitting controller code into an interrupt-driven real time part and an off-line background part often leads to the most efficient use of the microprocessor capabilities, especially for large implementations.

With most microprocessors, a number of programming languages can be used to generate controller code. These usually include a low-level assembly language and a variety of high-level languages, such as C. While it is generally possible to program tasks in any of the available languages, the speed of execution of the compiled code can vary significantly. In general, code written using the microprocessor assembly language executes faster (in other words, is more efficient) than code written using a higher level language. This means that it is advantageous to write the real time portion of the control code in assembly language, to give it maximum speed. The off-line code may then be written in a higher level language, where speed is not as important as the speed in with which a program can be written and debugged (higher level languages tend to be far simpler to program in that assembly language).

Finally, there is the choice of fixed point and floating point code. Basically, fixed point code is implemented using integer number, while floating point code is implemented using real numbers. Microprocessors, especially DSPs, can be purchased which explicit support fixed point and floating point calculations. In general, fixed point processors are (significantly) less expensive than floating point processors. However, fixed point calculations are more prone to a range of numerical problems than floating point calculations. As most fixed point devices have floating point "libraries" which allow them to emulate the functions of fixed point processors, it is not uncommon to implement a controller on a fixed point chip with fixed point real time filtering code and floating point weight adaptation.

Chapter 13 References

Franklin, G.F., Powell, J.D. and Workman, M.L. (1990). *Digital Control of Dynamic Systems*. Addison-Wesley: Reading, Massachusetts.

Middleton, R.H. and Goodwin, G.C. (1990). **Digital Control and Estimation, A Unified Approach**. Prentice Hall: Englewood Cliffs, New Jersey.

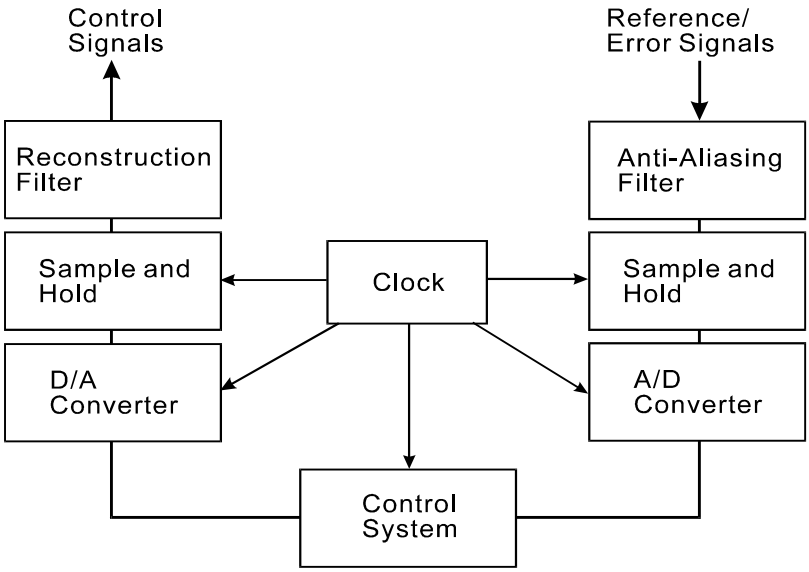

Fig 13.1. Block diagram of the main components of a digital control system.

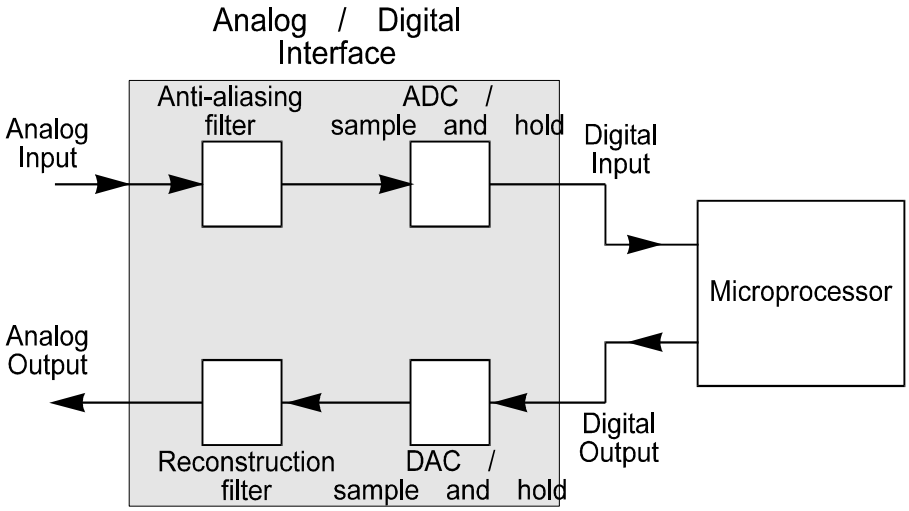

Fig 13.2. Digital control system separated into two parts: an analog/digital interface, and the main controller body.

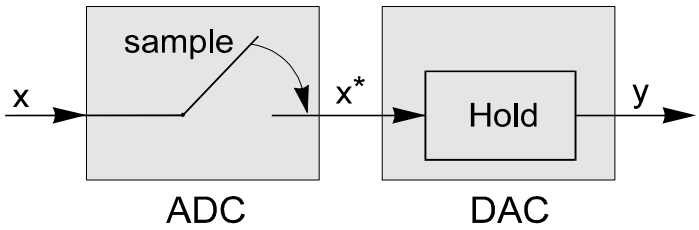

Fig 13.3. Simple ADC/DAC arrangement viewed as two parts: A sampler and a hold.

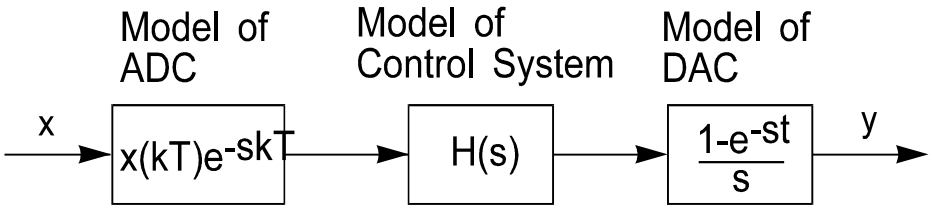

Fig 13.4. Block diagram of the simple ADC/DAC arrangement.

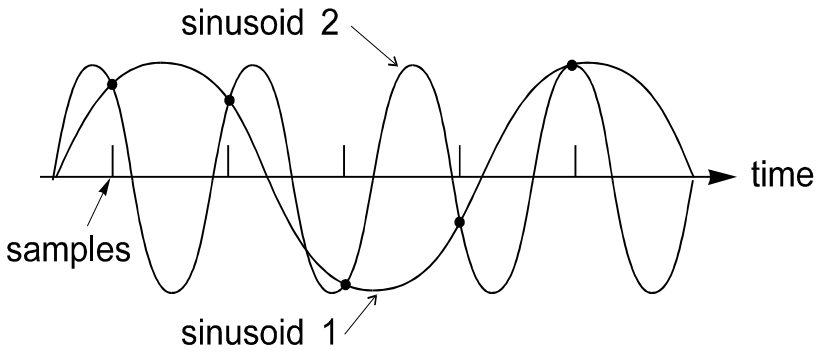

Fig 13.5 A demonstration of aliasing, where two sinusoids have the same sampled values.

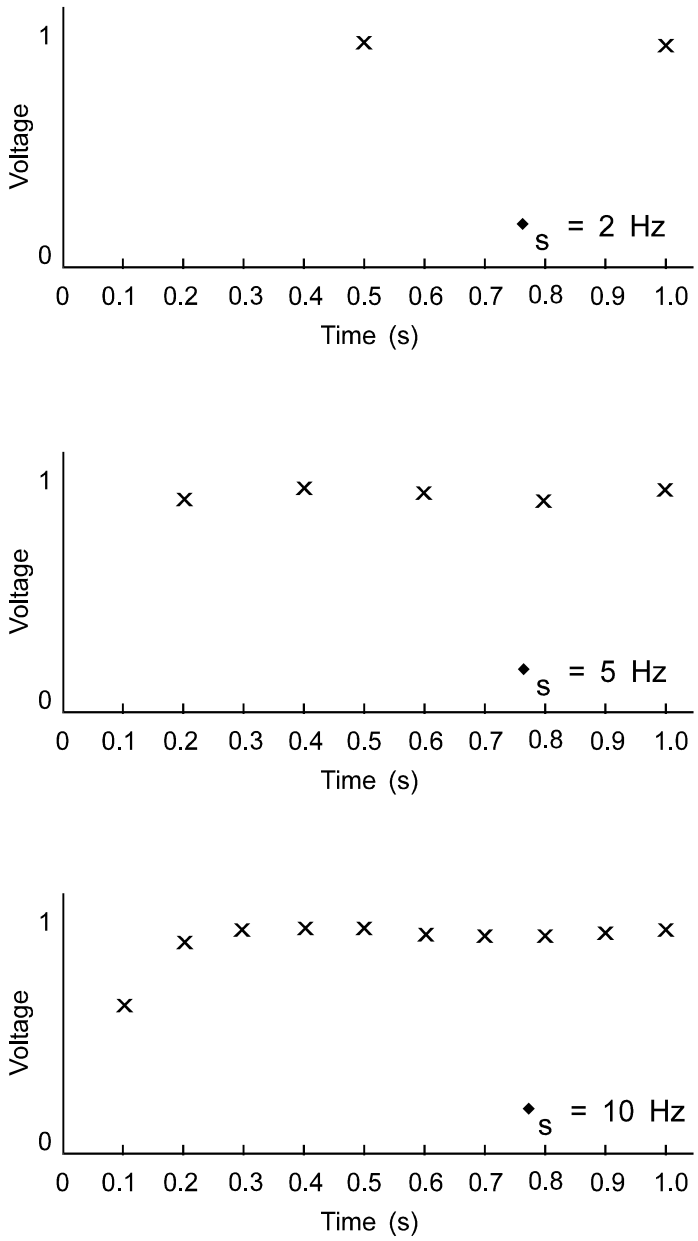

Fig 13.6. Step response of a system with a 1 Hz bandwidth, sampled at 2 Hz, 5 Hz, and 10 Hz.