eet

Digital Signal Processing

White

chapter 1

Introduction to Digital Signal Processing and Digital Filtering

1.1 Introduction

Digital signal processing (DSP) refers to anything that can be done to a signal using code on a computer or DSP chip. To reduce certain sinusoidal frequency components in a signal in amplitude, **digital filtering** is done. One may want to obtain the integral of a signal. If the signal comes from a tachometer, the integral gives the position. If the signal is noisy, then filtering the signal to reduce the amplitudes of the noise frequencies improves signal quality. For example, noise may occur from wind or rain at an outdoor music presentation. Filtering out sinusoidal components of the signal that occur at frequencies that cannot be produced by the music itself results in recording the music with little wind and rain noise. Sometimes the signal is corrupted not by noise, but by other signal frequencies that are of no present interest. If the signal is an electronic measurement of a brain wave obtained by using probes applied externally to the head, other electronic signals are picked up by the probes, but the physician may be interested only in signals occurring at a particular frequency. By using digital filtering, the signals of interest only can be presented to the physician.

1.2 Historical Perspective

Originally signal processing was done only on **analog or continuous time signals** using **analog signal processing (ASP)**. Until the late 1950s digital

computers were not commercially available. When they did become commercially available they were large and expensive, and they were used to simulate the performance of analog signal processing to judge its effectiveness. These simulations, however, led to digital processor code that simulated or performed nearly the same task on samples of the signals that the analog systems did on the signal. After a while it was realized that the simulation coding of the analog system was actually a DSP system that worked on samples of the input and output at discrete time intervals.

But to implement signal processing digitally instead of using analog systems was still out of the question. The first problem was that an analog input signal had to be represented as a sequence of samples of the signal, which were then converted to the computer's numerical representation. The same process would have to be applied in reverse to the output of the digitally processed signal. The second problem was that because the processing was done on very large, slow, and expensive computers, practical real-time processing between samples of the signal was impossible. Finally, as we will see in Chapter 9, even if digital processing could be done quickly enough between input samples in order to adequately represent the input signal, high sample rates require more bits of precision than slower ones.

The development of faster, cheaper, and smaller input signal samplers (ADCs) and output converters from digital data to analog data (DACs) began to make real-time DSP practical. Also, the processors were becoming smaller, faster, and cheaper and used more bits. Real-time replacements for analog systems may be just as small, cheap, and accurate and be able to process at a sample rate adequate for many analog signals.

However, testing and modification of the coding for DSP systems led to DSP systems that have no analog signal processing equivalents, yet sometimes perform the signal processing better than the DSP coding developed to replace analog systems. For digital filtering, these processing methods are discussed in Chapters 10 and 11.

1.3 Simple Examples of Digital Signal Processing

Digital signal processing entails anything that can be done to a signal using coding on a computer or DSP chip. This includes digital filtering

of signals as well as digital integration and digital correlation of signals. This text concentrates on constant rate digital filtering, with references to where the material is applicable to DSP in general. At the end of the text the techniques developed for digital filtering will be used for the DSP task of integration to show how the concepts and techniques are not limited to digital filtering.

The concepts are very simple. A signal is sampled in time at a constant rate in order to input its magnitude value at periodic intervals into the computer. The sample value of the analog magnitude is converted into a binary number. The sampling and conversion are done with an **analog to digital converter (ADC)**. Now the computer code can work on the signal. The computer code computes an output value, which is converted to an analog magnitude from a binary number and then held constant until a new output is computed to replace it. This is done by a **digital to analog converter (DAC)**. The basic DSP system described here is shown in Figure 1.1.

To illustrate the concept of DSP and to see where more study and analysis are needed, let's look at a few simple things that can be done to a sampled signal. If a signal is sampled every T seconds by an ADC and in the computer the sample value is just multiplied by a constant and then sent to the DAC, you have a digital amplifier. The gain of the amplifier is equal to the coded value of the constant. The following equation describes this digital amplifier, where x is the current input sample value from the ADC and y is the corresponding computer output to the DAC.

$$y = ax \qquad \text{A simple digital amplifier}$$

If the sampled value of the input signal is multiplied by T, you have computed an approximation to the area under the signal between samples, as long as the signal doesn't change too much between samples. If this value is added to the previous input sample multiplied by T, you

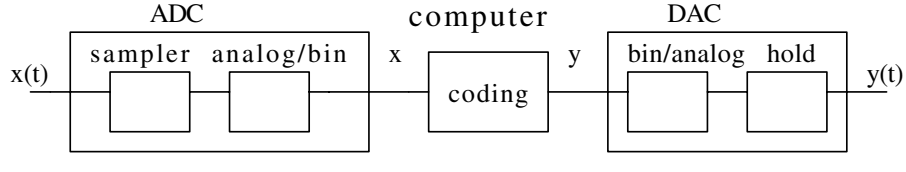

Figure 1.1. Basic DSP system

have approximated the area under the signal over two sample times. This could be repeated endlessly to approximate the area under the signal from when sampling started, as shown in Figure 1.2. The area under a signal or function is its integral. Thus you have performed very simple digital integration using the current sample of the input multiplied by T and then adding the result to the previous output. This process is described by the following equations after two input samples (the –1 subscripts indicate they are previous input or output values).

$$y_{-1} = Tx_{-1}$$ Simple digital integration after one input sample (the previous sample)

$$y = y_{-1} + Tx$$ Simple digital integration after two input samples (the current sample)

By using looping, such as a "While" or "For" loop, the preceding equations could be repeated endlessly by looping about one equation.

If the current input sample value is multiplied by one-half and added to half the previous sample value of the input, a current change in the input signal is reduced, while if the signal is changing slowly the output is very close to the input, since it is just the sum of two half values. Thus the computer is doing a very simple lowpass filtering of the input signal. This simple process is represented by the following equation. The result of using this equation on a string of input samples from the ADC is the input to the DAC shown in Table 1.1. As can be seen, the results, y, are smoothed or lowpass filtered versions of x, the ADC output.

$$y = 0.5x + 0.5x_{-1}$$ Simple digital lowpass filtering

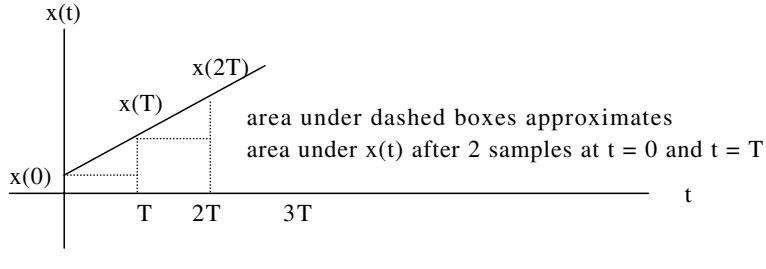

Figure 1.2. Example of digital integration

Table 1.1
Example of digital filtering (smoothing)

ADC sample time	0	T	2T	3T	4T
ADC output x	1.2	0.7	1.4	1.1	0.6
DAC input y	xxxxxx	0.95	1.05	1.25	0.85

1.4 The Common DSP Equation

The simple DSP examples just discussed were carried out using some input sample values stored in the computer or received currently from the ADC, multiplying them by appropriate constants, and summing the results. Sometimes the previous output values are multiplied by appropriate constants and also added to the first sum to give a new output, as was done in the digital integration example. Almost all digital signal processing by a computer involves adding the signal input sample just obtained, multiplied by a constant, to the sum of a few previous input samples, each multiplied by their corresponding constants, and sometimes adding all of this to a few previous outputs, each multiplied by their constants, to obtain a new output. This leads to the common equation used for almost all DSP:

$$y = (b_{-1}y_{-1} + \cdots + b_{-m}y_{-m}) + (ax + a_{-1}x_{-1} + \cdots + a_{-n}x_{-n}) \quad \text{(Equation 1.1)}$$

In Equation 1.1, the *x*s are the sampled input values, the *y*s are the output samples going to a DAC. The subscripts indicate how many previous sample periods ago are referred to. The *a*s and *b*s are just constants stored in the computer or DSP chip. A flowchart showing how Equation 1.1 might be implemented by code in the computer shown in Figure 1.1 is given in Figure 1.3.

It may seem strange that almost all DSP tasks are carried out by solving the preceding equation each time a new value of *x* is input from the ADC, but you must remember that all a computer can do mathematically is add, subtract, multiply, and divide; which is just what this equation requires. If you choose any values for any of the *a* and *b* constants and repeat the equation for every new input sample from an ADC, you will be doing DSP! But what DSP have you done and how well? The answers to these questions and more will be given in the rest of this text.

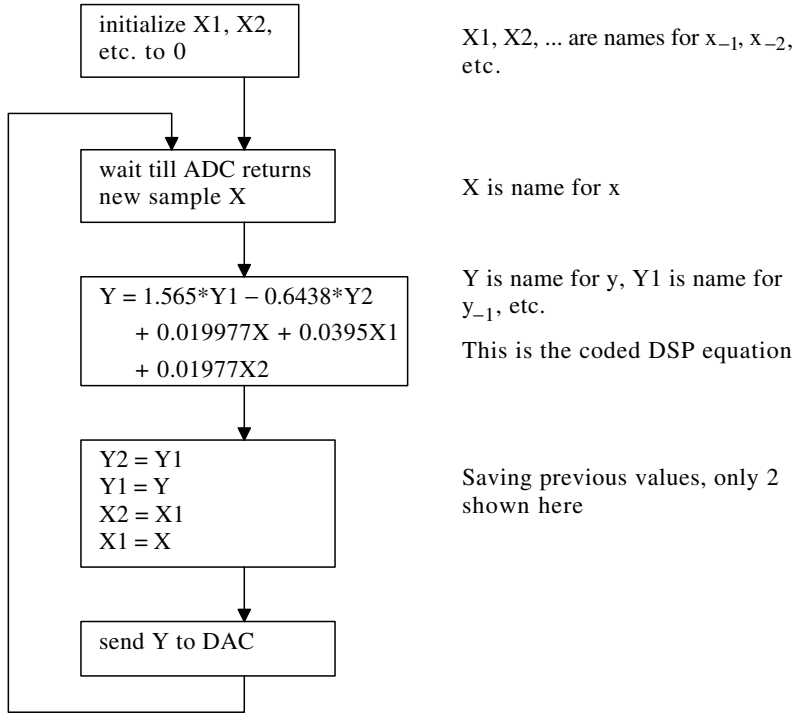

Figure 1.3. Flowchart using the common DSP equation

1.5 What the DSP Equation Shows

The common DSP equation will be used to show that many DSP questions need further study if one is to understand digital signal processing and do analysis or design of a DSP system. These questions include the following:

- How do you choose the *a* and *b* coefficients to perform a specific DSP task, such as doing second-order lowpass Butterworth filtering?
- How many coefficients are needed, and what is the effect of using fewer than required?
- Are the *b* coefficients always needed, and what is the effect if they are not used?

- The a and b coefficients are represented as binary numbers in the computer; how many bits should be used to meet the filter specifications?
- The x values are sample values of the input signal; how often should the signal be sampled?
- What is the effect of different sample rates, and does the filter coding need to be changed if the sample rate changes?
- How many bits should be used in the ADC and DAC to obtain a specific precision?

The answers to these questions and how they are obtained are subjects of the following chapters of this text. In order to fully make use of this text, the student should have a background in college algebra, trigonometry, first-semester calculus, analog filtering, and AC circuits. The only *required* background is in algebra and analog filtering; the others will increase the speed of learning and give a deeper understanding of the subject.

chapter 2

Effect of Signal Sampling

Introduction

In this chapter we examine the effects of sampling on signals and DSP systems. All DSP input signals are sampled, usually at equal intervals of time, in order to input numbers representative of the signal into a computer or DSP chip. We need to determine the effect of this sampling on the signal, as it produces unexpected and critical side effects; these need to be understood before effective filter design can be carried out. In order to simplify the demonstration of the effects of sampling, we will use an analog input signal composed of a single cosine wave. This signal will illustrate and quantitatively show the effects of sampling an analog signal by means of the ADC.

2.1 Periodic Sampling of a Cosine Signal

All signals worked on by DSP systems must be sampled at discrete values of time in order to be input into a DSP chip or computer. This sampling and its conversion to binary values is done by the ADC. This periodic sampling creates very special signal and DSP system characteristics. These characteristics are used in the specification and design of digital filters and DSP systems. In order to see and analyze these characteristics, we will look at the effects of periodic sampling on a cosine signal at different frequencies. We will often refer to signals at, above, or below a certain frequency, rather than to sinusoids with frequencies at, above, or below

those of sinusoids at a certain frequency. This shorthand English is used almost universally in industry and the literature. As an example, "reducing the frequencies above 100 rad/s" means to reduce the amplitudes of all sinusoids with frequencies above 100 rad/s.

If the signal into an ADC is a cosine wave at the radian frequency of w rad/s, its equation is given by Equation 2.1.

$$x(t) = \cos(wt) \qquad \text{(Equation 2.1)}$$

The continuous time variable is t, and A is the peak value. If the signal is sampled every T seconds, its value at the sample times is given by Equation 2.2, where n is an integer.

$$x(nT) = \cos(wnT) \qquad \text{(Equation 2.2)}$$

This sampling process is illustrated in Figure 2.1, where $T = 0.1$ and w is 2π rad/s. Column 2 of Table 2.1 gives nT, which is the sample time for every integer n, since the samples occur at $t = 0, T, 2T, \ldots$ only. A few sample values are computed in column 3 and can easily be checked using a calculator. All that has been done is to substitute nT for t, as shown in Equation 2.2. This is a valid way to get the equation of any signal after sampling, not just that of the cosine signal used here. For example, the following equations are the input and sampled output signals of an ADC for a decaying sinusoidal signal.

$$x(t) = Ae^{-3t}\cos(7t)$$

$$x(n) = x(nT) = Ae^{-3nT}\cos(7nT)$$

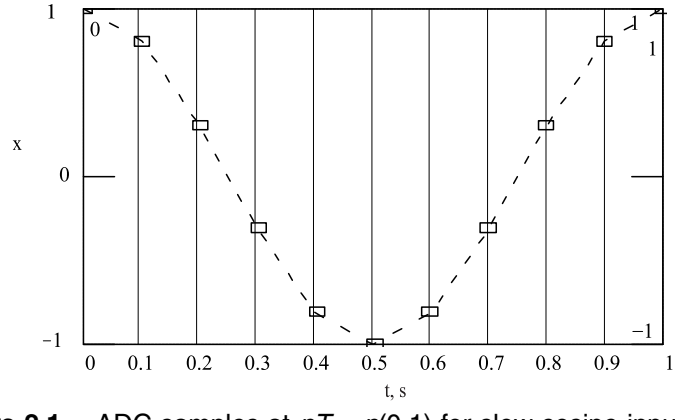

Figure 2.1. ADC samples at $nT = n(0.1)$ for slow cosine input

Table 2.1
Showing the effects of sampling the signals $x(t)$, $x1(t)$, and $x2(t)$

n	t = nT	x(n)	x1(n)	x2(n)
0	0	1	1	1
1	0.1	0.809	0.809	0.809
2	0.2	0.309	0.309	0.309
3	0.3	- 0.309	- 0.309	- 0.309
4	0.4	- 0.809	- 0.809	- 0.809
5	0.5	- 1	- 1	- 1
6	0.6	- 0.809	- 0.809	- 0.809
7	0.7	- 0.309	- 0.309	- 0.309
8	0.8	0.309	0.309	0.309
9	0.9	0.809	0.809	0.809
10	1	1	1	1

Notice that for notational convenience the sampled signal argument is usually written without the sample period T, but T or its value is never left out of the sampled signal equation (or it would be a different equation).

2.2 Periodicity of Any DSP System Frequency Response

Let's look at the values of the sampled sinusoid when its frequency w is increased by the sampling frequency w_s, as shown in the following equation.

$$x1(t) = \cos[(w + w_s)t] \qquad \text{(Equation 2.3)}$$

The sampling frequency in Hz is $1/T$ and in rad/s is $2\pi/T$, so Equation 2.3 gives

$$x1(n) = \cos[(w + 2\pi/T)nT]$$

$$= \cos(wnT + 2n\pi)$$

$$= \cos(wnT)$$

Notice that after sampling, the signal in Equation 2.3 looks just like the original sampled signal in Equation 2.2. You can see that the sampled values of Equation 2.3 shown in column 4 of Table 2.1 are the same as the values in column 3. The preceding equations make use of the fact that the sine or cosine of an angle offset by 2π is not changed. You can easily see that any angles offset by multiples of 2π are the same by turning around 2π radians in a room and finding that you are facing in the original direction again.

If the cosine signal frequency is increased by integer multiples of the sampling frequency, its sampled values out of the ADC are indistinguishable from the corresponding values of the samples of the unshifted cosine input to the ADC. This is true even if the original frequency is decreased by multiples of the sampling frequency such that a negative argument for the sampled cosine is obtained, since $\cos(-a)$ is $\cos(a)$, from trigonometry. This is also apparent to all students in electronics , since on a scope the only difference between a cosine signal and a negative cosine signal on an oscilloscope is when the trace starts.

Why was it useful to point out that two different cosine signals into an ADC have the same sample values out of the ADC if they differ in frequency by the sampling frequency of $2\pi/T$ rad/s? The reason is that all signals worked on by a DSP system are represented in a computer as outputs from an ADC. If two signals have the same values out of the ADC, then any DSP system will do exactly the same thing to both signals. One example is a lowpass digital filter. The purpose of a lowpass filter is to reduce the amplitude of sinusoidal signals above a specified frequency, while not reducing the amplitude below a specified frequency. From the preceding discussion, we know that above a certain frequency the digital lowpass filter will start acting like a highpass filter because a high frequency sinusoidal will have the same sample values as a lower frequency sinusoidal signal. This important fact will be used in Chapter 3 in drawing graphical digital filter specifications.

The preceding equations show that all ADC outputs look alike if separated by $2\pi/T$ radians per second. There is no way around this. Whatever a digital filter does, its characteristics repeat themselves every $2\pi/T$ radians per second, because after going through an ADC the inputs look the same, as Table 2.1 shows. The math is just an explicit way to show this, but it can be seen in Figure 2.2, where the same sinusoid shown in Figure 2.1 is sampled at the original rate after its frequency is increased by $2\pi/T$ radians per second. The cosine wave is shown by dashed lines,

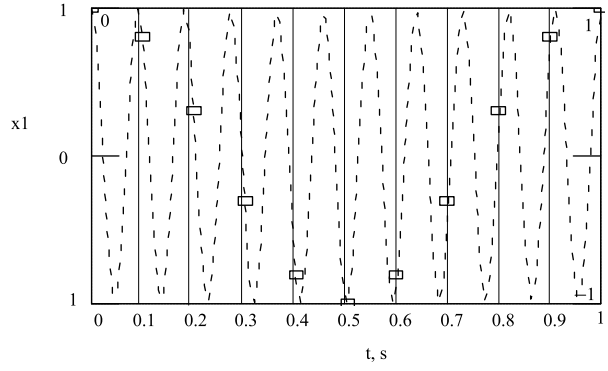

Figure 2.2. ADC samples at $nT = n(0.1)$ for fast cosine input

and the sample values by boxes. If you want to filter an analog signal, you must sample at a high enough rate or eliminate the high frequency content of the signal first.

We have shown that a cosine signal increased in frequency by the ADC sampling rate has the same sample values as if its frequency were not increased. Next it will be shown that its sample values will be the same at an even lower increased frequency. As stated earlier, from trigonometry we have $\cos(a) = \cos(-a)$. Again using the property that any trigonometric function is identical if its angle is changed by 2π radians, we will show that the signal $x2(t)$ in Equation 2.4 has the same value out of the ADC as the original signal $x(t)$.

$$x2(t) = \cos[(-w + w_s)t] \qquad \text{(Equation 2.4)}$$

This cosine signal $x2(t)$ is just the original signal $x(t)$ with its frequency at the sample frequency minus the original signal's frequency. When samples of $x2(t)$ are taken every T seconds by the ADC, the equation for the sample values is derived as in the following set of equations.

$$x2(n) = \cos[(-w + 2\pi/T)nT]$$

$$= \cos(-wnT + 2\pi n)$$

$$= \cos(-wnT)$$

$$= \cos(wnT)$$

Again it is seen that at a higher frequency even less than the sample frequency the sampled values out of an ADC and into a computer look identical to those at a lower frequency. This can be verified by computing the values in column 5 of Table 2.1 at the sample times on a calculator, using Equation 2.4 with $T = 0.1$ and $w = 2\pi$ rad/s. This new higher frequency is not the original increased by the sampling frequency, but the original frequency subtracted from the sample frequency. This result is true for all integer multiples of the sample frequency.

Thus, using the simple algebraic substitution of nT for t to obtain the value of a cosine signal at the sample times separated by T seconds, we have seen that all DSP systems must do the same thing to sinusoids of frequency w as they do to sinusoids at frequencies w above and below the sample frequency, since their sample values into the computer are the same. This is shown in Figure 2.3, where a cosine of amplitude A is plotted at w, $w_s + w$, and $w_s - w$. Remember that w_s is just the sampling frequency in rad/s. This significant result must be taken into account when designing a DSP system. If the DSP system is the previously mentioned lowpass filter, its filtering characteristics repeat, as shown in Figure 2.4. When specifying the frequency characteristics of a DSP system such as a digital filter, you must be aware that they will repeat above half the sample rate at π/T in rad/s, as seen in Figure 2.4, and this repetition is periodic.

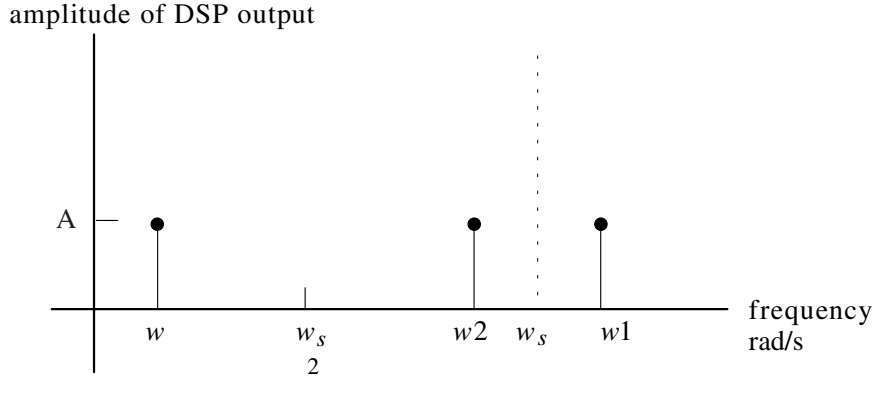

Figure 2.3. Equivalence of DSP output for sinusoids at *w*, *w*1, *w*2

2.3 Aliasing and Nyquist Limit

The condition where the highest input signal frequency content is equal to half the sample rate is called the **Nyquist limit**, and it leads to the **Nyquist criterion**. The Nyquist criterion is not violated if the sampling rate is more than twice as high as the highest frequency of the sinusoids in the signal, that is, if the highest input signal frequency content is less than the Nyquist limit. This is shown in Figure 2.4. The figure is drawn for an arbitrary input signal frequency spectrum; the signal might be composed of only one cosine wave or many cosine or sine waves.

In Figure 2.4 the Nyquist criterion is not violated, but it can be seen that if the sampling frequency is not greater than twice the highest frequency of any sinusoid in the signal, frequency components in the original signal would look like lower frequency signal components. Figure 2.5 shows the same signal spectrum sampled at a lower rate, so that the Nyquist limit is violated. The DSP system not only treats sinusoids above the Nyquist limit as if they were lower frequency sinusoids, but actually includes them with the actual lower frequency sinusoids. It is important to be aware of this double whammy. The DSP system not only has a periodic frequency spectrum for its output signal, but it also modifies the spectrum you have tried to design by including higher frequency sinusoids for processing as if they were the corresponding lower frequency sinusoids. There is no way to undo this effect of a DSP system. You must either be aware of the damage and accept the consequences, avoid violating the Nyquist criterion by sampling faster, or eliminate frequency components in the signal above the Nyquist limit.

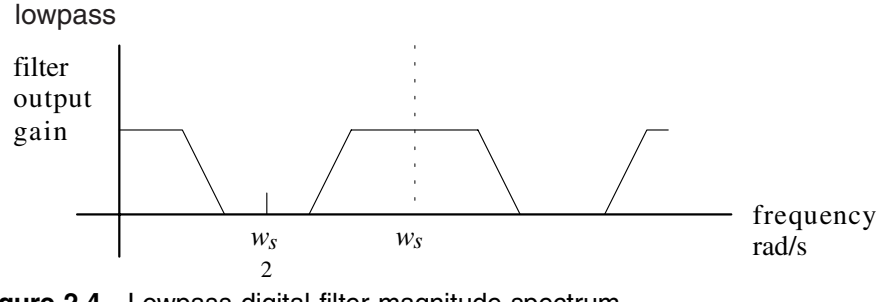

Figure 2.4 Lowpass digital filter magnitude spectrum

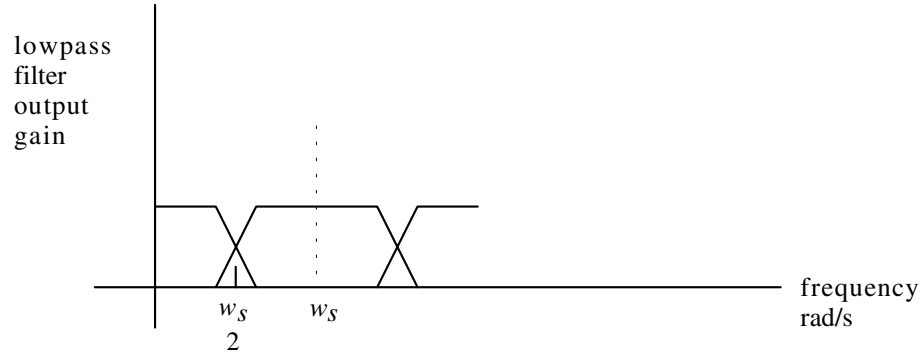

Figure 2.5. Lowpass digital filter showing Nyquist violated

2.4 Anti-aliasing Filters

Because the frequency content of most signals is unknown to some degree, especially when noise is considered, most DSP systems use a lowpass analog filter called an **anti-aliasing filter** in front of the ADC. This filter must be an analog filter, since it is in front of the ADC. If it were after the ADC it would itself be a digital filter, with the same problems you want to eliminate from the original DSP system! Figure 2.6 shows a typical DSP system using an anti-aliasing filter.

The specifications on the anti-aliasing filter depend on the input signal sinusoidal frequency content and the proposed sampling rate specified by the sample period T. As mentioned earlier, it must be an analog lowpass filter. It seems strange that almost all DSP systems and especially digital filters include an analog filter. However, this is usually a very simple lowpass filter to build. All it needs to do is to reduce the amplitudes of sinusoids in the signal into the ADC below an acceptable level above a frequency at which they look like significant lower frequency sinusoids.

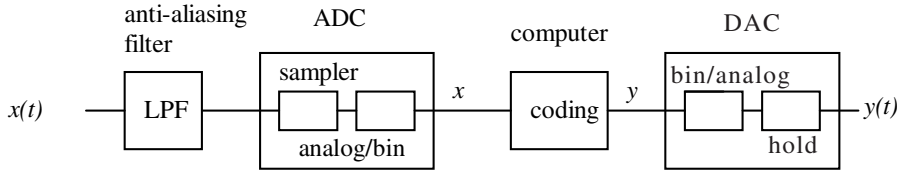

Figure 2.6. DSP system with anti-aliasing filter

Figure 2.7 shows this process for a digital lowpass filter, with the much less stringent anti-aliasing lowpass analog filter response shown in dashed lines.

Example 2.1. Determining the requirements for an anti-aliasing filter

Problem: Assume that a digital lowpass filter is to be designed to pass all frequencies below 100 rad/s and reduce all frequencies above 500 rad/s by 32. Let's assume the sample period is 0.001 s. An anti-aliasing filter must be designed for this digital filter.

Solution: The sample rate in rad/s is 2000π. From Figure 2.7 it can be seen that frequencies above 2000π minus 100 rad/s must be reduced by 32 (by the anti-aliasing filter) or else the digital filter will pass them as if they were the corresponding low frequency signals in the passband.

The requirements on the anti-aliasing filter are seen to be that it reduces the amplitude of the signal into the ADC by 32 or more above 5783 rad/s while not significantly reducing the frequencies below 100 rad/s. From an analog filtering course, it can be learned that this is easily achieved by a first-order lowpass analog filter. A first-order filter reduces the signal by 2 every time the frequency doubles beyond the corner frequency. If the corner frequency is set at 100 rad/s, by the time the frequency is 3200 rad/s (which is five doublings of the corner frequency), the analog signal is reduced by 32. This first-order lowpass filter could even be a simple two-component RC filter.

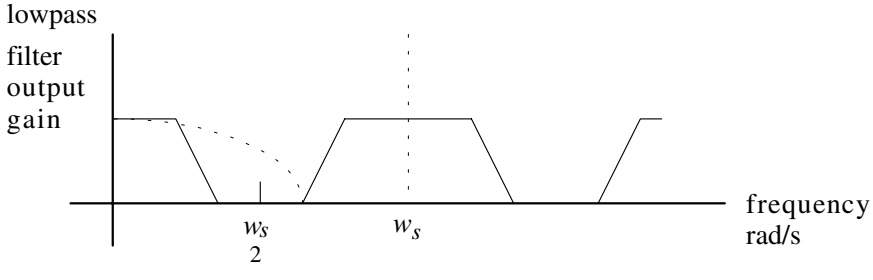

Figure 2.7 Lowpass digital filter and anti-aliasing filter

2.5 The Nyquist Limit and DSP Output Periodicity by Mathematical Means

Most DSP texts and most engineers have been trained to think of aliasing and the Nyquist criterion in terms of the frequency of a sampled signal. A simple derivation is given in the following discussion; most other derivations are much more complex. It is not necessary to determine the magnitude spectrum of a sampled signal, since we have determined aliasing and the Nyquist criterion without it. Inside the computer or DSP chip, there are just a bunch of numbers being manipulated, not a signal with a sampled analog spectrum. But to illustrate the other approach, the following derivation is given, using knowledge from first-semester calculus and Laplace transform theory. The output of an ADC is a sampled signal and we will show that this leads to it having a periodic spectrum with a period of $1/T$ Hz or $2\pi/T$ rad/s. Again, as can be seen in Figure 2.4 and Figure 2.5, this leads to the Nyquist criterion on the sampling rate.

We will use the delta or impulse function $\delta(t)$ used in analog signal processing class, a very narrow and tall signal with area (or strength) = 1 at $t = 0$ and zero value anywhere else. Then $\delta(t - T)$ is just a spike of strength 1 at $t = T$ and zero everywhere else. The sum of $\delta(t)$ and $\delta(t - T)$ is just spikes of strength 1 at $t = 0$ and another at $t = T$. Using this approach, the output of an ADC is given by the following equations.

$$x(nT) = x(n) = x(0)\delta(t) + x(T)\delta(t - T) + x(2T)\delta(t - 2T) + \ldots$$

$$x(n) = \sum_{n=0}^{\infty} x(nT)\delta(t - nT) = x(t)\sum_{n=0}^{\infty} \delta(t - nT)$$

Now it can be seen that $\sum_{n=0}^{\infty} \delta(t - nT)$ is periodic (plot it), so it has a Fourier series, as given by the following equation.

$$\sum_{n=0}^{\infty} \delta(t - nT) = a_0 + \sum_{k=1}^{\infty} a_k \cos(k\omega_s t) + \sum_{k=1}^{\infty} b_k \sin(k\omega_s t), \text{ where } \omega_s = 2\pi/T$$

The coefficients for the preceding Fourier series are computed as follows, using the standard formulas for computing Fourier series coefficients.

$$a_0 = \frac{1}{T}\int_{-T/2}^{T/2} \sum_{n=0}^{\infty} \delta(t - nT)\mathrm{dt} = 1/T$$

$$a_k = \frac{2}{T}\int_{-T/2}^{T/2} \sum_{n=0}^{\infty} \delta(t - nT)\cos(k\omega_s t)\mathrm{dt} = \frac{2}{T}$$

$$b_k = \frac{2}{T}\int_{-T/2}^{T/2} \sum_{n=0}^{\infty} \delta(t - nT)\sin(k\omega_s t)\mathrm{dt} = 0$$

Using the preceding coefficients in the Fourier series equation, we can write the equation for the sum of the impulses in the following form.

$$\sum_{n=0}^{\infty} \delta(t - nT) = \frac{1}{T} + \frac{2}{T}\sum_{k=1}^{\infty} \cos(k\omega_s t)$$

Using this in the equation for $x(nT) = x(n)$, as a discrete-time signal, gives the following equation.

$$x(n) = x(t)[\frac{1}{T} + \frac{2}{T}\sum_{k=1}^{\infty} \cos(k\omega_s t)]$$

Now let $x(t)$ be any sinusoid of amplitude A at frequency w. Then the preceding equation for $x(n)$ can be written in the following form.

$$x(n) = x(nT) = \mathrm{A}\cos(\omega t)[\frac{1}{T} + \frac{2}{T}\sum_{k=1}^{\infty} \cos(k\omega_s t)]$$

Using the trigonometric identity for the product of cosines, we finally have the result we need in the following equation.

$$x(n) = x(nT) = \frac{\mathrm{A}}{\mathrm{T}}\cos(\omega t) + \frac{2\,\mathrm{A}}{T}\sum_{k=1}^{\infty} [0.5\cos(\omega - k\omega_s)t + 0.5\cos(\omega + k\omega_s)t]$$

From the preceding equation it is obvious that the sampled signal frequency response as a discrete time signal is periodic and symmetric