Digital Signal Processing: A Computer Science Perspective
Jonathan Y. Stein

Print ISBN 0-471-29546-9 Online ISBN 0-471-20059-X

Part II

Signal Processing Systems

Digital Signal Processing: A Computer Science Perspective
Jonathan Y. Stein

Print ISBN 0-471-29546-9 Online ISBN 0-471-20059-X

6

Systems

The study of signals, their properties in time and frequency domains, their fundamental mathematical and physical limitations, the design of signals for specific purposes, and how to uncover a signal's capabilities through observation belong to *signal analysis*. We now turn to *signal processing*, which requires adding a new concept, that of the *signal processing system*.

A signal processing system is a device that processes input signals and/or produces output signals. Signal processing systems were once purely analog devices. Older household radio receivers input analog radio frequency signals from an antenna, amplify, filter, and extract the desired audio from them using analog circuits, and then output analog audio to speakers. The original telephone system consisted of analog telephone sets connected via copper wire lines, with just the switching (dialing and connecting to the desired party) discrete. Even complex radar and electronic warfare systems were once purely analog in nature.

Recent advances in microelectronics have made DSP an attractive alternative to analog signal processing. Digital signal processing systems are employed in a large variety of applications where analog processing once reigned, and of course newer purely digital applications such as modems, speech synthesis and recognition, and biomedical electronics abound. There still remain applications where analog signal processing systems prevail, mainly applications for which present-day DSP processors are not yet fast enough; yet the number of such applications is diminishing rapidly.

In this chapter we introduce systems analogously to our introduction of signals in Chapter 2. First we define analog and digital signal processing systems. Then we introduce the simplest possible systems, and important classes of systems. This will lead us to the definition of a *filter* that will become a central theme in our studies. Once the concept of filter is understood we can learn about MA, AR, and combined ARMA filters. Finally we consider the problem of *system identification* which leads us to the concepts of *frequency response*, *impulse response*, and *transfer function*.

6.1 *System* Defined

The first question we must ask when approaching the concept of signal processing is 'What exactly do we mean by a signal processing *system*?'

Definition: signal processing system
A signal processing system is any device that takes in zero or more signals as input, and returns zero or more signals as outputs. ■

According to this definition systems deal only with signals. Of course images may be considered two-dimensional signals and thus image processing *is* automatically included. Nevertheless, we will often extend the definition to include systems that may also input other entities, such as numeric or logical values. A system may output such other entities as well. An important output entity is a multiclass classification identifier, by which we mean that various signals may be input to the system as a function of time, and the system classifies them as they arrive as belonging to a particular class. The only practical requirement is that there should be at least one output, either signal, numeric, logical, or classification. Were one to build a system with no outputs, after possibly sophisticated processing of the input, the *system* would know the result (but you wouldn't).

What kind of system has no input signals? An example would be an *oscillator* or tone generator, which outputs a sinusoidal signal of constant frequency, irrespective of whatever may be happening around it. A simple modification would be to add a numeric input to control the amplitude of the sine, or a logical input to reset the phase. Such an oscillator is a basic building block in communications transmitters, radars, signaling systems, and music synthesizers.

What kind of system has no signal output? An example would be a *detector* that outputs a logical *false* until a signal of specified parameters is detected. A simple modification would be to output a numeric value that relates the time of detection to a reference time, while a more challenging extension would continually output the degree to which the present input matches the desired signal (with 0 standing for no match, 1 for perfect match). Such a system is the basis for modem demodulators, radar receivers, telephone switch signaling detectors, and pattern analyzers. Systems that output only multiclass classifications are the subject of a discipline known as *pattern recognition.*

EXERCISES

6.1.1 Which of the following are signal processing systems (we shall use x for inputs and y for outputs)? Explain.

1. The identity $y = x$
2. The constant $y = k$ irrespective of x
3. $y = \pm\sqrt{x}$
4. A device that inputs a pizza and outputs a list of its ingredients
5. $y = \sin(\frac{1}{t})$
6. $y(t) = \int_{-\infty}^{t} x(t)$
7. The Fourier transform
8. A television
9. A D/A converter

6.1.2 Given any two signals $x(t)$ and $y(t)$, is there always a system that inputs x and outputs y? Given a system that inputs $x(t)$ and outputs $y(t)$, is there always a system that inputs y and outputs x?

6.2 The Simplest Systems

Let us now present a few systems that will be useful throughout our studies. The simplest system with both an input signal x and an output signal y is the *constant*, $y(t) = k$ in analog time or $y_n = k$ in digital time. This type of system may model a power supply that strives to output a constant voltage independent of its input voltage. We can not learn much from this trivial system, which completely ignores its input. The next simplest system is the *identity*, whose output exactly replicates its input, $y(t) = x(t)$ or $y_n = x_n$.

The first truly nontrivial system is the *amplifier*, which in the analog world is $y(t) = Ax(t)$ and in the digital world $y_n = Ax_n$. A is called the *gain*. When $A > 1$ we say the system *amplifies* the input, since the output as a function of time looks like the input, only larger. For the same reason, when $A < 1$ we say the system *attenuates*. Analog amplifiers are vital for broadcast transmitters, music electronics (the reader probably has a stereo amplifier at home), public address systems, and measurement apparatus.

The ideal amplifier is a *linear system*, that is, the amplification of the sum of two signals is the sum of the amplifications, and the amplification of a constant times a signal is the constant times the amplification of the signal.

$$A\Big(x_1(t) + x_2(t)\Big) = Ax_1(t) + Ax_2(t) \qquad \text{and} \qquad A\Big(cx(t)\Big) = cAx(t)$$

Such perfect linear amplification can only be approximated in analog circuits; analog amplifiers saturate at high amplitudes, lose amplification at high frequencies, and do not respond linearly for very high amplitudes. Digitally amplification is simply multiplication by a constant, a calculation that may be performed reliably for all inputs, unless overflow occurs.

We can generalize the concept of the amplifier/attenuator by allowing deviations from linearity. For example, real analog amplifiers cannot output voltages higher than their power supply voltage, thus inducing *clipping*. This type of nonlinearity

$$y(t) = \mathrm{Clip}_\theta\left(Ax(t)\right) \qquad \mathbf{A}\,\Big|\,\mathbf{D} \qquad y_n = \mathrm{Clip}_\theta\left(Ax_n\right) \tag{6.1}$$

where

$$\mathrm{Clip}_\theta(x) \equiv \begin{cases} \theta & x \geq \theta \\ x & -\theta < x < \theta \\ -\theta & -\theta \leq x \end{cases}$$

is depicted in Figure 6.1. On the left side of the figure we see the output versus input for an ideal linear amplifier, and on the right side the output when

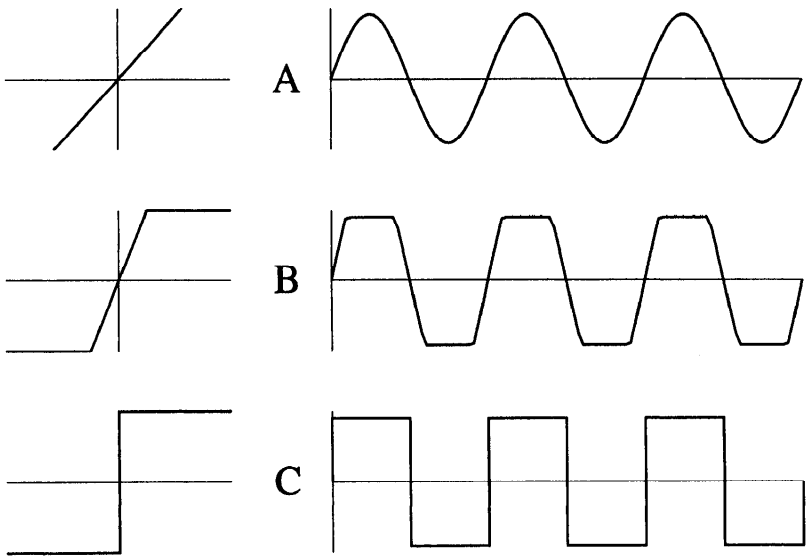


Figure 6.1: The effect of clipping amplifiers with different gains on sinusoidal signals. In (A) there is no clipping, in (B) intermediate gain and clipping, while (C) represents the infinite gain case (hard limiter).

a sinusoid is input. Figure 6.1.B represents an amplifier of somewhat higher gain, with a limitation on maximal output. The region where the output no longer increases with increasing input is called the region of *saturation.* Once the amplifier starts to saturate, we get 'flat-topping' of the output, as is seen on the ride side. The flat-topping gets worse as the gain is increased, until in 6.1.C the gain has become infinite and thus the system is always saturated (except for exactly zero input). This system is known as a 'hard limiter', and it essentially computes the sign of its input.

$$y(t) = \mathrm{sgn}\left(x(t)\right) \qquad \mathbf{A} \Big| \mathbf{D} \qquad y_n = \mathrm{sgn}\left(x_n\right) \tag{6.2}$$

Hard limiting changes sinusoids into square waves, and is frequently employed to obtain precisely this effect.

These clipping amplifiers deal symmetrically with positive and negative signal values; another form of nonlinearity arises when the sign explicitly affects the output. For example, the gain of an amplifier can depend on whether the signal is above or below zero. Extreme cases are the *half-wave rectifier*, whose output is nonzero only for positive signal values,

$$y(t) = \theta\left(x(t)\right) x(t) \qquad \mathbf{A} \Big| \mathbf{D} \qquad y_n = \theta\left(x_n\right) x_n \tag{6.3}$$

and the *full-wave rectifier*, whose output is always positive

$$y(t) = |x(t)| \qquad \mathbf{A} \Big| \mathbf{D} \qquad y_n = |x_n| \tag{6.4}$$

as depicted in Figure 6.2.

Yet another deviation from linearity is termed *power-law distortion*; for example, quadratic power distortion is

$$y(t) = x(t) + \epsilon x^2(t) \qquad \mathbf{A} \Big| \mathbf{D} \qquad y_n = x_n + \epsilon x_n^2 \tag{6.5}$$

for small $\epsilon > 0$. More generally higher powers may contribute as well. Real amplifiers always deviate from linearity to some degree, and power law distortion is a prevalent approximation to their behavior. Another name for power law distortion is *harmonic generation*; for example, quadratic power

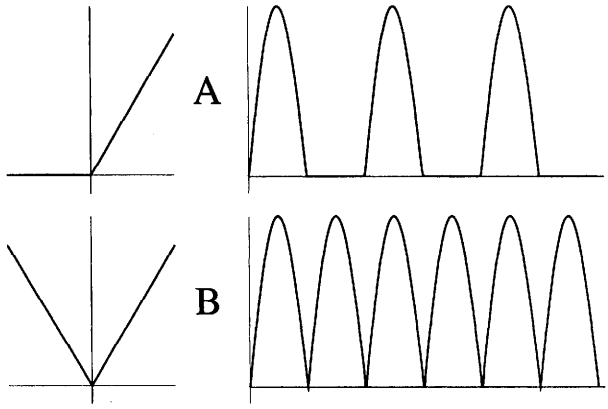


Figure 6.2: Half-wave and full-wave rectifiers. (A) depicts the output as a function of input of a half-wave rectifier, as well as its effect on a sinusoid. (B) depicts the same for a full-wave rectifier.

distortion is called second-harmonic generation. The reasoning behind this name will become clear in Section 8.1.

Let us summarize some of the systems we have seen so far:

constant	$y(t) = k$	$y_n = k$
identity	$y(t) = x(t)$	$y_n = x_n$
amplification	$y(t) = Ax(t)$	$y_n = Ax_n$
clipping	$y(t) = \text{Clip}_\theta\left(Ax(t)\right)$	$y_n = \text{Clip}_\theta\left(Ax_n\right)$
hard limiter	$y(t) = \text{sgn}\left(x(t)\right)$	$y_n = \text{sgn}\left(x_n\right)$
half-wave rectification	$y(t) = \theta\left(x(t)\right)x(t)$	$y_n = \theta\left(x_n\right)x_n$
full-wave rectification	$y(t) = \lvert x(t)\rvert$	$y_n = \lvert x_n\rvert$
quadratic distortion	$y(t) = x(t) + \epsilon x^2(t)$	$y_n = x_n + \epsilon x_n^2$

This is quite an impressive collection. The maximal extension of this type of system is the *general point transformation* $y(t) = f\left(x(t)\right)$ or $y_n = f\left(x_n\right)$. Here f is a completely general function, and the uninitiated to DSP might be led to believe that we have exhausted all possible signal processing systems. Notwithstanding, such a system is still extremely simple in at least two senses. First, the output at any time depends only on the input at that same time and nothing else. Such a system is *memoryless* (i.e., does not retain memory of previous inputs). Second, this type of system is *time-*

invariant (i.e., the behavior of the system does not change with time). Classical mathematical analysis and most non-DSP numerical computation deal almost exclusively with memoryless systems, while DSP almost universally requires combining values of the input signal at many different times. Time invariance, the norm outside DSP, is common in many DSP systems as well. However, certain important DSP systems *do* change as time goes on, and may even change in response to the input. We will see such systems mainly in Chapter 10.

EXERCISES

6.2.1 The digital amplifier is a linear system as long as no *overflow* or *underflow* occur. What is the effect of each of these computational problems? Which is worse? Can anything be done to prevent these problems?

6.2.2 Logarithmic companding laws are often used on speech signals to be quantized in order to reduce the required dynamic range. In North America the standard is called μ-law and is given by

$$y = \text{sgn}(x)\frac{\log(1+\mu|x|)}{\log(1+\mu)} \tag{6.6}$$

where x is assumed to be between -1 and $+1$ and $\mu = 255$. In Europe A-law is used

$$y = \begin{cases} \text{sgn}(x)\frac{\log(A|x|)}{\log(1+\log(A)} & 0 < |x| < \frac{1}{A} \\ \text{sgn}(x)\frac{1+\log(A|x|)}{1+\log(A)} & \frac{1}{A} < |x| < 1 \end{cases} \tag{6.7}$$

where $A = 87.6$. Why are logarithmic curves used? How much difference is there between the two curves?

6.2.3 Time-independent point transformations can nontrivially modify a signal's spectrum. What does squaring signal values do to the spectrum of a pure sinusoid? To the sum of two sinusoids? If point operations can modify a signal's spectrum why do you think systems with memory are needed?

6.3 The Simplest Systems with Memory

There are two slightly different ways of thinking about systems with memory. The one we will usually adopt is to consider the present output to be a function of the present input, previous inputs, and previous outputs.

$$y_n = f\left(x_n, x_{n-1}, x_{n-2}, \cdots y_{n-1}, y_{n-2} \cdots\right) \tag{6.8}$$

The other line of thought, called the *state-space description*, considers the output to be calculated based on the present input and the present *internal state* of the system.

$$y_n = f\left(x_n, \underline{S}\right) \tag{6.9}$$

In the state-space description the effect of the input on the system is twofold, it causes an output to be generated and it changes the state of the system. These two ways of thinking are clearly compatible, since we could always define the internal state to contain precisely the previous inputs and outputs. This is even the best way of defining the system's state for systems that explicitly remember these values. However, many systems do not actually remember this history; rather this history influences their behavior.

The simplest system with memory is the *simple delay*

$$y(t) = x(t-\tau) \qquad \mathbf{A} \Big| \mathbf{D} \qquad y_n = x_{n-m} \tag{6.10}$$

where the time τ or m is called the *lag*. From the signal processing point of view the simple delay is only slightly less trivial than the identity. The delay's output still depends on the input at only *one* time, that time just happens not to be the present time, rather the present time minus the lag.

We have said that the use of delays is one of the criteria for contrasting simple numeric processing from signal processing. Recall from Chapter 2 that what makes signal processing special is the schizophrenic jumping back and forth between the *time domain* and the *frequency domain*. It is thus natural to inquire what the simple delay does to the frequency domain representation of signals upon which it operates. One way to specify what any signal processing system does in the frequency domain is to input simple sinusoids of all frequencies of interest and observe the system's output for each. For the simple delay, when a sinusoid of amplitude A and frequency ω is input, a sinusoid of identical amplitude and frequency is output. We will see later on that a system that does not change the frequency of sinusoids and does not create new frequencies is called a *filter*. A filter that does not change the amplitude of arbitrary sinusoids, that is, one that passes all frequencies without attenuation or amplification, is called an *all-pass filter*. Thus the simple delay is an all-pass filter. Although an all-pass filter leaves the *power spectrum* unchanged, this does *not* imply that the spectrum remains unchanged. For the case of the delay it is obvious that the phase of the output sinusoid will usually be different from that of the input. Only if the lag is precisely a whole number of periods will the phase shift be zero; otherwise the phase may be shifted either positively or negatively.

After a little consideration we can deduce that the phase is shifted by the frequency times the delay lag. When the phase shift is proportional to the frequency, and thus is a straight line when plotted as a function of frequency, we say that the system is *linear-phase*. The identity system $y = x$ is also linear-phase, albeit with a trivial constant zero phase shift. Any time delay (even if unintentional or unavoidable such as a processing time delay) introduces a linear phase shift relation. Indeed any time-invariant linear-phase system is equivalent to a zero phase shift system plus a simple delay. Since simple delays are considered trivial in signal processing, linear-phase systems are to be considered 'good' or 'simple' in some sense. In contrast when the phase shift is not linear in frequency, some frequencies are delayed more than others, causing phase distortion. To appreciate the havoc this can cause imagine a nonlinear-phase concert hall. In any large concert hall a person in the balcony hears the music a short time after someone seated up front. When the room acoustics are approximately linear-phase this delay is not particularly important, and is more than compensated for by the reduction in ticket price. When nonlinear phase effects become important the situation is quite different. Although the music may be harmonious near the stage, the listener in the balcony hears different frequencies arriving after *different* time delays. Since the components don't arrive together they sum up to quite a different piece of music, generally less pleasant to the ear. Such a concert hall would probably have to pay people to sit in the balcony, and the noises of indignation made by these people would affect the musical experience of the people up front as well.

How can the simple delay system be implemented? The laws of relativity physics limit signals, like all information-carrying phenomena, from traveling at velocities exceeding that of light. Thus small analog delays can be implemented by *delay lines*, which are essentially appropriately chosen lengths of cable (see Figure 6.3.A). A voltage signal that exits such a delay line cable is delayed with respect to that input by the amount of time it took for the electric signal to travel the length of the cable. Since electric signals tend to travel quickly, in practice only very short delays can be implemented using analog techniques. Such short delays are only an appreciable fraction of a period for very high-frequency signals. The delay, which is a critical processing element for all signal processing, is difficult to implement for low-frequency analog signals.

Digital delays of integer multiples of the sampling rate can be simply implemented using a FIFO buffer (see Figure 6.3.B). The content of this FIFO buffer is precisely the system's internal state from the state-space point of view. The effect of the arrival of an input is to cause the oldest value

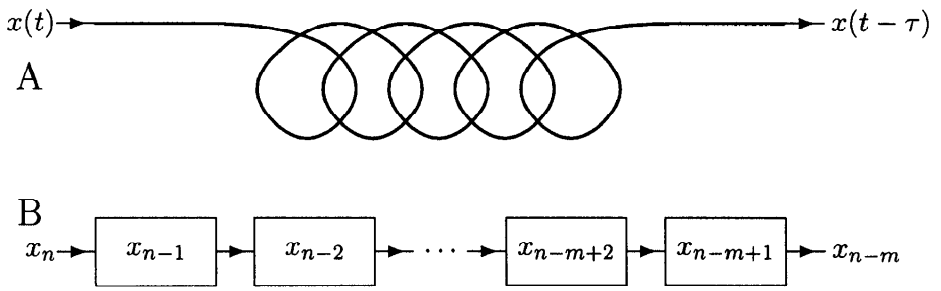


Figure 6.3: Implementation of the simple delay. In (A) we see how an analog delay of lag τ can be obtained by allowing a voltage of current signal to travel at finite velocity through a sufficiently long delay line. In (B) a digital delay of lag m is implemented using a FIFO buffer.

stored in the FIFO to be output and promptly discarded, for all the other values to 'move over', and for the present input to be placed in the buffer. Of course long delays will require large amounts of memory, but memory tends to drop in price with time, making DSP more and more attractive vis-a-vis analog processing. DSP does tend to break down at high frequencies, which is exactly where analog delay lines become practical.

Leaving the simple delay, we now introduce a somewhat more complex system. Think back to the last time you were in a large empty room (or a tunnel or cave) where there were strong echoes. Whenever you called out you heard your voice again after a delay (that we will call τ), which was basically the time it took for your voice to reach the wall from which it was reflected and return. If you tried singing or whistling a steady tone you would notice that some tones 'resonate' and seem very strong, while others seem to be absorbed. We are going to model such a room by a system whose output depends on the input at two different times, the present time and some previous time $t - \tau$. Our simple 'echo system' adds the signal values at the two times

$$y(t) = x(t) + x(t-\tau) \qquad \mathbf{A} \,\big|\, \mathbf{D} \qquad y_n = x_n + x_{n-m} \tag{6.11}$$

and is easily implemented digitally by a FIFO buffer and an adder.

In the frequency domain this system is *not* all-pass; the frequency dependence arising from the time lag τ (or m) corresponding to different phase differences at different frequencies. When we input a sinusoidal signal with

angular frequency ω such that τ corresponds to precisely one period (i.e., $\omega\tau = 2\pi$), the net effect of this system is to simply double the signal's amplitude. If, however, the input signal is such that τ corresponds to a half period ($\omega\tau = \pi$), then the output of the system will be zero. This is the reason some frequencies resonate while others seem to be absorbed.

More generally, we can find the *frequency response*, by which we mean the response of the system to any sinusoid as a function of its frequency. To find the frequency response we apply an input of the form $\sin(\omega t)$. The output, which is the sum of the input and its delayed copy, will be

$$\begin{aligned} \sin(\omega t) + \sin\Big(\omega(t-\tau)\Big) &= 2\cos\left(\frac{\omega\tau}{2}\right)\sin\left(\omega(t-\frac{\tau}{2})\right) \\ &= A(\omega\tau)\sin\left(\omega(t-\frac{\tau}{2})\right) \end{aligned}$$

which is easily seen to be a sinusoid of the same frequency as the input. It is, however, delayed by half the time lag (linear-phase!), and has an amplitude that depends on the product $\omega\tau$. This amplitude is maximal whenever $\omega\tau = 2k\pi$ and zero when it is an odd multiple of π. We have thus completely specified the frequency response; every input sine causes a sinusoidal output of the same frequency, but with a linear phase delay and a periodic amplification. A frequency that is canceled out by a system (i.e., for which the amplification of the frequency response is zero) is called a *zero* of the system. For this system all odd multiples of π are zeros, and all even multiples are maxima of the frequency response.

Our next system is only slightly more complex than the previous one. The 'echo system' we just studied assumed that the echo's amplitude was exactly equal to that of the original signal. Now we wish to add an echo or delayed version of the signal to itself, only this time we allow a multiplicative coefficient (a *gain* term).

$$y(t) = x(t) + hx(t-\tau) \qquad \mathbf{A} \,\Big|\, \mathbf{D} \qquad y_n = x_n + hx_{n-m} \tag{6.12}$$

When $h = 1$ we return to the previous case, while $h < 1$ corresponds to an attenuated echo, while $h > 1$ would be an amplified echo. We can also consider the case of negative h, corresponding to an echo that returns with phase reversal.

$$y(t) = x(t) - |h|x(t-\tau) \qquad \mathbf{A} \,\Big|\, \mathbf{D} \qquad y_n = x_n - |h|x_{n-m}$$

We leave the full mathematical derivation of the frequency response of our generalized echo system as an exercise. Still we can say a lot based on a little experimentation (using pen and paper or a computer graphing program). The first thing we notice is that a sinusoidal input will produce a sinusoidal output of the same frequency, but with amplitude between $1 - |h|$ and $1 + |h|$. Thus when $|h| \neq 1$ we can never perfectly cancel out a sinusoidal input signal, no matter what frequency we try, and thus the frequency response will have no zeros. Of course when $|h| < 1$ we can't double the amplitude either; the best we can do is to amplify the signal by $1 + |h|$. Yet this should be considered a mere quantitative difference, while the ability or inability to exactly zero out a signal is qualitative. The minima of the frequency response still occur when the echo is exactly out of phase with the input, and so for positive h occur whenever $\omega\tau$ is an odd multiple of π, while for negative h even multiples are needed. We present the graphs of amplification as a function of frequency for various positive h in Figure 6.4.

We can generalize our system even further by allowing the addition of multiple echoes. Such a system combines the input signal (possibly multiplied by a coefficient) with delayed copies, each multiplied by its own coefficient. Concentrating on digital signals, we can even consider having an echo from every possible time lag up to a certain maximum delay.

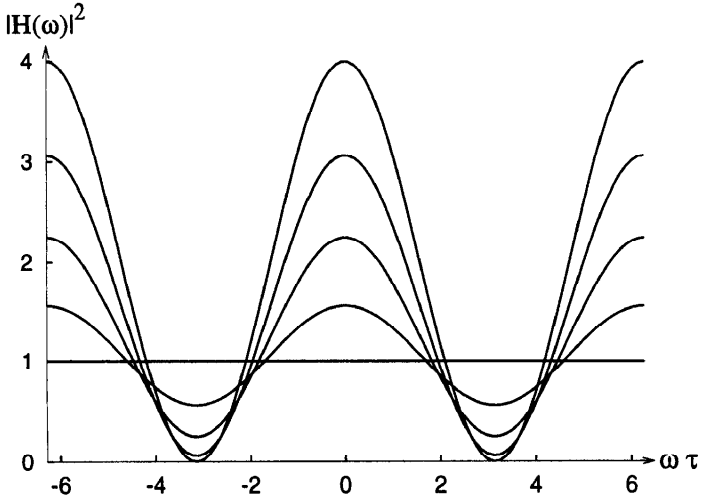


Figure 6.4: Amplitude of the frequency response for the echo system with positive coefficients. The amplitude is plotted as a function of $\omega\tau$. The coefficients are $h = 0$ (the straight line), $0.25, 0.5, 0.75$, and 1.0 (the plot with zeros).

$$y_n = h_0 x_n + h_1 x_{n-1} + h_2 x_{n-2} + \cdots + h_L x_{n-L} = \sum_{l=0}^{L} h_l x_{n-l} \qquad (6.13)$$

This system goes under many different names, including **M**oving **A**verage (MA) filter, *FIR* filter, and *all-zero* filter, the reasoning behind all of which will be elucidated in due course. The mathematical operation of summing over products of indexed terms with one index advancing and one retreating is called *convolution.*

Now this system may seem awesome at first, but it's really quite simple. It is of course linear (this you can check by multiplying x by a constant, and by adding $x_1 + x_2$). If the input signal is a pure sine then the output is a pure sine of the same frequency! Using linearity we conclude that if the input signal is the sum of sinusoids of certain frequencies, the output contains only these same frequencies. Although certain frequencies may be zeros of the frequency response, no new frequencies are ever created. In this way this system is simpler than the nonlinear point transformations we saw in the previous section. Although limited, the FIR filter will turn out to be one of the most useful tools in DSP.

What should be our next step in our quest for ever-more-complex digital signal processing systems? Consider what happens if echoes from the distant past are still heard—we end up with a nonterminating convolution!

$$y_n = \sum_{l=-\infty}^{\infty} h_l x_{n-l} \qquad (6.14)$$

In a real concert hall or cave the gain coefficients h_l get smaller and smaller for large enough l, so that the signal becomes imperceptible after a while. When an amplifier is involved the echoes can remain finite, and if they are timed just right they can all add up and the signal can become extremely strong. This is what happens when a microphone connected to an amplifier is pointed in the direction of the loudspeaker. The 'squeal' frequency depends on the time it takes for the sound to travel from the speaker to the microphone (through the air) and back (through the wires).

The FIR filter owes its strength to the idea of iteration, looping on all input signal values from the present time back to some previous time.

$y_n = 0$
`for` $i = 0$ `to` L
    $y_n \leftarrow y_n + h_l x_{n-l}$

More general than *iteration* is *recursion*,

$$y_n = f(x_n, x_{n-1}, x_{n-2}, \ldots, y_{n-1}, y_{n-2}, \ldots)$$

and the *IIR filter* exploits this by allowing y_n to be a weighted sum of all previous *outputs* as well as inputs.

$$\begin{aligned} y_n &= a_0 x_n + a_1 x_{n-1} + \cdots + a_L x_{n-L} + b_1 y_{n-1} + b_2 y_{n-2} + \cdots + b_M y_{n-M} \\ &= \sum_{l=0}^{L} a_l x_{n-l} + \sum_{m=1}^{M} b_m y_{n-m} \end{aligned} \tag{6.15}$$

We see here two convolution sums, one on the inputs and one on the (previous) outputs. Although even this system cannot create sinusoids of frequencies that do not exist in the input at all, it *can* magnify out of all proportion components that barely exist (see exercises). Of course even IIR filters are simple in a sense since the coefficients a_l and b_m do not vary with time. More complex systems may have coefficients that depend on time, on other signals, and even on the input signal itself. We will see examples of such systems when we discuss adaptive filters.

Are these time- and signal-dependent systems the most complex systems DSP has to offer? All I can say is 'I hope not.'

EXERCISES

6.3.1 Prove the following characteristics of the convolution.

existence of identity	$s_1 * \delta$	$= s_1$
commutative law	$s_1 * s_2$	$= s_2 * s_1$
associative law	$s_1 * (s_2 * s_3)$	$= (s_1 * s_2) * s_3$
distributive law	$s_1 * (s_2 + s_3)$	$= (s_1 * s_2) * (s_1 * s_3)$

6.3.2 We saw that a generalized echo system $y(t) = x(t) + hx(t-\tau)$ has no zeros in its frequency response for $|h| < 1$; i.e., there are no sinusoids that are exactly canceled out. Are there signals that are canceled out by this system?

6.3.3 Find the frequency response (both amplitude and phase) for the generalized echo system. Use the trigonometric identity for the sine of a sum, and then convert $a\sin(\omega t) + b\cos(\omega t)$ to $A\sin(\omega t + \phi)$. Check that you regain the known result for $h = 1$. Show that the amplitude is indeed between $1 - |h|$ and $1 + |h|$.

6.3.4 Plot the amplitude found in the previous exercise for positive coefficients and check that Figure 6.4 is reproduced. Now plot for negative h. Explain. Plot the phase found in the previous exercise. Is the system always linear-phase?

6.3.5 The digital generalized echo system $y_n = x_n + hx_{n-m}$ can only implement an echo whose delay is an integer number of sample intervals t_s. How can a fractional sample delay echo be accommodated?

6.3.6 Show that an IIR filter can 'blow up', that is, increase without limit even with constant input.

6.3.7 Show that the IIR filter

$$y_n = x_n - a_1 y_{n-1} - y_{n-2} \qquad y_n = 0 \text{ for } n < 0$$

when triggered with a unit impulse $x_n = \delta_{n,0}$ can sustain a sinusoid. What is its frequency?

6.3.8 The sound made by a plucked guitar string is almost periodic, but starts loud and dies out with time. This is similar to what we would get at the output of an IIR system with a delayed and attenuated echo of the output $y_n = x_n + gy_{n-m}$ with $0 < g < 1$. What is the frequency response of this system? (Hint: It is easier to use $x_n = e^{i\omega n}$ for $n > 0$ and zero for $n < 0$, rather than a real sinusoid.)

6.3.9 All the systems with memory we have seen have been *causal*, that is, the output at time T depends on the input at previous times $t \leq T$. What can you say about the output of a causal system when the input is a unit impulse at time zero? Why are causal systems *sensible*? One of the advantages of DSP over analog signal processing is the possibility of implementing noncausal systems. How (and when) can this be done?

6.4 Characteristics of Systems

Now that we have seen a variety of signal processing systems, both with memory and without, it is worthwhile to note some general characteristics a system might have. We will often use operator notation for systems with a single input and a single output signal.

$$y(t) = Hx(t) \qquad \mathbf{A} \,\Big|\, \mathbf{D} \qquad y_n = Hx_n \tag{6.16}$$

Here H is a *system* that converts one signal into another, not merely a function that changes numbers into numbers.

A memoryless system is called *invertible* if distinct input values lead to distinct output values. The system $y_n = 2x_n$ is thus invertible since every

finite value of x_n leads to a unique y_n. Such systems are called invertible since one can produce an *inverse system* H^{-1} such that $x_n = H^{-1}y_n$. For the system just mentioned it is obvious that $x_n = \frac{1}{2}y_n$. Since

$$x_n = H^{-1}\, y_n = H^{-1}\, H\, x_n \tag{6.17}$$

we can formally write

$$H^{-1}\, H = 1 \tag{6.18}$$

where 1 is the identity system. The system $y_n = x_n^2$ is noninvertible since both $x_n = -1$ and $x_n = +1$ lead to $y_n = +1$. Thus there is no system H^{-1} that maps y_n back to x_n.

The notion of invertibility is relevant for systems *with* memory as well. For example, the simple FIR filter

$$y_n = x_n - x_{n-1}$$

has an inverse system

$$x_n = y_n + x_{n-1}$$

which is an IIR filter. Unraveling this further we can write

$$\begin{aligned} x_n &= y_n + (y_{n-1} + x_{n-2}) \\ &= y_n + y_{n-1} + (y_{n-2} + x_{n-3}) \\ &= y_n + y_{n-1} + y_{n-2} + y_{n-3} + \cdots \end{aligned}$$

and assuming that the input signal was zero for $n = 0$ we get an infinite sum.

$$x_n = \sum_{i=0}^{\infty} y_i \tag{6.19}$$

Inverse systems are often needed when signals are distorted by a system and we are called upon to counteract this distortion. Such an inverse system is called an *equalizer*. An equalizer with which you may be familiar is the adjustable or preset equalizer for high-fidelity music systems. In order to reproduce the original music as accurately as possible, we need to cancel out distortions introduced by the recording process as well as resonances introduced by room acoustics. This is accomplished by dividing the audio spectrum into a small number of bands, the amplification of which can be individually adjusted. Another equalizer you may use a great deal, but without realizing it, is the equalizer in a modem. Phone lines terribly distort data signals and without equalization data transmission speeds would be around

2400 bits per second. By employing sophisticated adaptive equalization techniques to counteract the distortion, transmission speeds more than ten times faster can be attained.

In a Section 6.2 we mentioned *linearity*, although in the restricted context of memoryless systems. The definition remains the same in the general case, namely

$$H(x+y) = Hx + Hy \qquad \text{and} \qquad H(cx) = cHx \tag{6.20}$$

that is, H is a *linear* system if its output, when the input is a sum of two signals, is precisely the sum of the two signals that would have been the outputs had each signal been inputed to H separately. The second part states that when the input is a constant times a signal the output must be the constant times the output that would have been obtained were the unamplified signal input instead. We have already seen quite a few nonlinear systems, such as the squaring operation and the hard limiter. Nonlinear systems require special care since they can behave chaotically. We use the term chaos here in a technical sense—small changes to the input may cause major output changes.

This last remark leads us to the subject of *stability*. A system is said to be stable if bounded input signals induce bounded output signals. For example, the system

$$y_n = \tan\left(x_n - \frac{\pi}{2}\right)$$

is unstable near $x_n = 0$ since the output explodes there while the input is zero. However, even linear systems can be unstable according to the above definition. For instance, the system

$$y_n = \sum_{l=0}^{L} x_l$$

is linear, but when presented with a constant input signal the output grows (linearly) without limit.

We generally wish to avoid instability as much as possible, although the above definition is somewhat constraining. Systems with sudden singularities or exponentially increasing outputs should be avoided at all costs; but milder divergences are not as damaging. In any case true analog systems are always stable (since real power supplies can only generate voltages up to a certain level), and digital systems can not support signal values larger than the maximum representable number. The problem with this compelled stability is that it comes at the expense of nonlinearity.

The next characteristic of importance is *time-invariance*. A system H is said to be time-invariant if its operation is not time-dependent. This means that applying time delay or time advance operators to the input of a system is equivalent to applying them to the output.

$$y(t) = Hx(t) \longrightarrow y(t+T) = Hx(t+T) \tag{6.21}$$

A time-variant system has some internal clock that influences its behavior. For example,

$$y_n = \Big(1 + \sin(\Omega t)\Big)\, x_n$$

is time-variant, as is any system that is turned on at some time (i.e., that has zero output before this time no matter what the input, but output dependent on the input after this time).

The combination of linearity and time invariance is important enough to receive a name of its own. Some DSP engineers call a linear and time-invariant systems *LTI* systems, but most use the simpler name *filter*.

Definition: filter
A filter is a system H with a single input and single output signal that is both linear (obeys (6.20)) and time-invariant (obeys equation (6.21)). ∎

As usual we often deviate from the precise definition and speak of *nonlinear filters*, *time-variant filters*, and *multidimensional filters*, but when used without such qualifications the term 'filter' will be taken to be equivalent to LTI.

We already know about systems with memory and without. The output value of a system without memory depends only on the input value at the same time. Two weaker characteristics that restrict the time dependence of the output are *causality* and *streamability*. A system is termed causal if the output signal value at time T is only dependent on the input signal values for that time and previous times $t \leq T$. It is obvious that a memoryless system is always causal, and it is easy to show that a filter is causal if and only if a unit impulse input produces zero output for all negative times. Noncausal systems seem somewhat unreasonable, or at least necessitate time travel, since they require the system to correctly guess what the input signal is going to do at some future time. The philosophical aspects of this dubious behavior are explored in an exercise below. Streamable systems are either causal or can be made causal by adding an overall delay. For example, neither $y_n = x_{-n}$ nor $y_n = x_{n+1}$ are causal, but the latter is streamable while the former is not.

When working off-line, for instance with an input signal that is available as a file or known as an explicit function, one can easily implement noncausal systems. One need only *peek ahead* or precompute the needed input values, and then place the output value in the proper memory or file location. Analog systems can realize only causal systems since they must output values immediately without peeking forward in time, or going back in time to correct the output values. Since analog systems are also required to be stable, stable causal systems are called *realizable*, meaning simply that they may be built in analog electronics. Real-time digital systems can realize only stable streamable systems; the amount of delay allowed is application dependent, but the real-time constraint requires the required delay to be constant.

EXERCISES

6.4.1 Find the inverse system for the following systems. If this is in IIR form find the FIR form as well (take $x_n = 0$ for $n \leq 0$).

1. $y_n = x_n + x_{n-1}$
2. $y_n = x_n - \frac{1}{2}x_{n-1}$
3. $y_n = x_n - x_{n-1} - x_{n-2}$
4. $y_n = \sum_{i=-\infty}^{n} x_n$
5. $y_n = x_n + y_{n-1}$
6. $y_n = x_n - x_{n-1} + y_{n-1}$

6.4.2 What can you say about the FIR and IIR characteristics of inverse systems?

6.4.3 Which of the following systems are filters? Explain. 1. $y_n = x_{n-1} + k$

2. $y_n = x_{n+1}x_{n-1}$
3. $y_n = x_{ln}$
4. $y_n = 0$
5. $y_n = y_{n-1}$

6.4.4 Show that a filter is causal if and only if its output, when the input is a unit impulse centered on time zero, is nonzero only for positive times.

6.4.5 Show that if two signals are identical up to time t, then the output of a causal system to which these are input will be the same up to time t.

6.4.6 Show that the smoothing operation $y_n = \frac{1}{2}(x_n + x_{n-1})$ is causal while the similar $y_n = \frac{1}{2}(x_{n+1} + x_n)$ is not. The system $y_n = \frac{1}{2}(x_{n+1} - x_{n-1})$ is an approximation to the derivative. Show that it is not causal but is streamable. Find a causal approximation for the derivative.

6.4.7 Consider the philosophical repercussions of noncausal systems by reflecting on the following case. The system in question outputs -1 for two seconds if its input will be positive one second from now, but $+1$ for two seconds if its input will be negative. Now feed the output of the system back to its input.

6.4.8 Explain why streamable systems can be realized in DSP but not in analog electronics. What does the delay do to the phase response?

6.4.9 The systems $y_n = x_n + a$ (which adds a DC term) and $y_n = x_n^2$ (which squares its input) do not commute. Show that any two *filters* do commute.

6.4.10 Systems do not have to be deterministic. The **M**odulated **N**oise **R**eference **U**nit (MNRU) system, defined by $y_n = \left(1 + 10^{-\frac{Q}{20}}\nu_n\right) x_n$ (where ν is wideband noise) models audio quality degradation under logarithmic companding (exercise 6.2.2). Which of the characteristics defined in this section does the MNRU have? Can you explain how the MNRU works?

6.5 Filters

In the previous section we mentioned that the combination of linearity and time invariance is important enough to deserve a distinctive name, but did not explain why this is so. The explanation is singularly DSP, linking characteristics in the time domain with a simple frequency domain interpretation. We shall show shortly that the spectrum of a filter's output signal is the input signal's spectrum multiplied by a frequency-dependent weighting function. This means that some frequencies may be amplified, while others may be attenuated or even removed; the amplification as a function of frequency being determined by the particular filter being used. For example, an ideal low-pass filter takes the input signal spectrum, multiplies all frequency components below a cutoff frequency by unity, but multiplies all frequency components over that frequency by zero. It thus passes low frequencies while removing all high-frequency components. A band-pass filter may zero out all frequency components of the input signal except those in a range of frequencies that are passed unchanged.

Only filters (LTI systems) can be given such simple frequency domain interpretations. Systems that are not linear and time-invariant can create new frequency components where none existed in the input signal. For example, we mentioned at the end of Section 6.2 and saw in exercise 6.2.3 that the squaring operation generated harmonics when a sinusoidal signal was input, and generated combination frequencies when presented with the sum of two sinusoids. This is a general feature of non-LTI systems; the spectrum of the output will have frequency components that arise from complex combinations of input frequency components. Just as the light emerging from a optical filter does not contain colors lacking in the light impinging upon it,

just as when pouring water into a coffee filter brandy never emerges, just so you can be sure that the output of a signal processing filter does not contain frequencies absent in the input.

Let's prove this important characteristic of filters. First, we expand the input in the SUI basis (as the sum of unit impulses weighted by the signal value at that time).

$$x_n = \sum_{m=-\infty}^{\infty} x_m \delta_{n,m}$$

Next, using the linearity of the filter H, we can show that

$$y_n = Hx_n = H\left(\sum_{m=-\infty}^{\infty} x_m \delta_{n,m}\right) = \sum_{m=-\infty}^{\infty} H\left(x_m \delta_{n,m}\right)$$

but since the x_m are simply constants multiplying the SUIs, linearity also implies that we can move them outside the system operator.

$$y_n = \sum_{m=-\infty}^{\infty} x_m H\delta_{n,m}$$

Now the time has come to exploit the time invariance. The operation of the system on the SUI $H\delta_{n,m}$ is precisely the same as its operation on the unit impulse at time zero, only shifted m time units. The *impulse response* h_n is defined to be the response of a system at time n to the unit impulse.

$$h_n = H\delta_{n,0} \tag{6.22}$$

For causal systems $h_n = 0$ for $n < 0$, and for practical systems h_n must become small for large enough n. So time invariance means $H\delta_{n,m} = h_{n-m}$ and we have found the following expression for a filter's output.

$$y_n = \sum_{m=-\infty}^{\infty} x_m h_{n-m} \tag{6.23}$$

For causal filters future inputs cannot affect the output.

$$y_n = \sum_{m=-\infty}^{0} x_m h_{n-m} \tag{6.24}$$

We have seen this type of sum before! We called it a *convolution* sum and saw in Section 4.8 that its LTDFT was particularly simple. Taking the LTDFT of both sides of equation (6.23) and using equation (4.44) we find

$$Y_k = H_k X_k \tag{6.25}$$

which states that the output at frequency k is the input at that frequency multiplied by a frequency-dependent factor H_k. This factor is the digital version of what we previously called the frequency response $H(\omega)$.

Using terminology borrowed from linear algebra, what we have proved is that the sinusoids are *eigenfunctions* or, using more fitting terminology, *eigensignals* of filters. If we allow complex signals we can prove the same for the complex sinusoids $x_n = e^{i\omega n}$.

EXERCISES

6.5.1 In exercise 6.3.7 we saw that the system $y_n = x_n - a_1 y_{n-1} - y_{n-2}$ could sustain a sinusoidal oscillation even with no input. Yet this is a filter, and thus should not be able to create frequencies not in the input! Explain.

6.5.2 Show that the system $y_n = x_n + x_{-n}$ is not a filter. Show that it indeed doesn't act as a filter by considering the inputs $x_n = \sin(\omega n + \phi)$ and $x_n = \cos(\omega n + \phi)$.

6.5.3 Prove the general result that z^n are eigenfunctions of filters.

6.5.4 Prove the filter property for analog signals and filters.

6.6 Moving Averages in the Time Domain

We originally encountered the FIR filter as a natural way of modeling a sequence of echoes, each attenuated or strengthened and delayed in time. We now return to the FIR filter and ask why it is so popular in DSP applications. As usual in DSP there are two answers to this question, one related to the time domain and the other to the frequency domain. In this section we delve into the former and ask why it is natural for a system's output to depend on the input at more than one time. We will motivate this dependency in steps.

Consider the following problem. There is a signal z_n that is known to be constant $z_n = z$, and we are interested in determining this constant. We are not allowed to directly observe the signal z_n, only the signal

$$x_n = z_n + \nu_n$$

where ν_n is some noise signal. We know nothing about the noise save that its average is zero, and that its variance is finite.

Since the noise averages to zero and the observed signal is the sum of the desired constant signal and this noise, the observed signal's average value must be z. Our path is clear; we need to average the observed signal

$$\frac{1}{L}\sum_{l=0}^{L-1} x_l \approx z \tag{6.26}$$

with the sum approaching z more and more closely as we increase L. For finite L our estimate of z will be not be exact, but for large enough L (the required size depending on the noise variance) we will be close enough.

Now let us assume that z_n is not a constant, but a slowly varying signal. By *slowly varying* we mean that z_n is essentially the same for a great many consecutive samples. Once again we can only observe the noisy x_n, and are interested in recovering z_n. We still need to average somehow, but we can no longer average as much as we please, since we will start 'blurring' the desired nonconstant signal. We thus must be content with averaging over several x_n values,

$$y_n = \frac{1}{L}\sum_{l=0}^{L-1} x_{n+l} \approx z_n \tag{6.27}$$

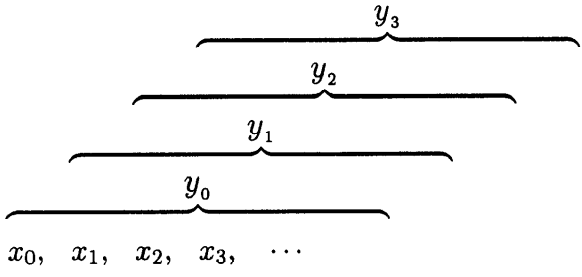
and repeating this operation every j samples in order to track z_n.

$$\overbrace{x_0, x_1, \ldots x_{L-1}}^{y_0}, x_L, x_{L+1}, \ldots \overbrace{x_j, x_{j+1}, \ldots x_{j+L-1}}^{y_j}, x_{j+L}, x_{j+L+1}, \cdots$$

We must take j small enough to track variations in z_n, while $L \leq j$ must be chosen large enough to efficiently average out the noise. Actually, unless there is some good reason not to, we usually take $L = j$ precisely.

$$\overbrace{x_0, x_1, \ldots x_{L-1}}^{y_0}, \overbrace{x_L, x_{L+1}, \ldots x_{2L-1}}^{y_L}, \overbrace{x_{2L}, x_{2L+1}, \ldots x_{3L-1}}^{y_{2L}}, \cdots$$

Now we assume that z_n varies a bit faster. We must reduce j in order to track z_n sufficiently well, but we cannot afford to reduce L this much unless the noise is very small. So why can't the averaging intervals overlap? Why can't we even calculate a new average every sample?



Well, we can; this type of averaging is called a *moving average*, which is often abbreviated MA. The moving average operation produces a new signal y_n which is an approximation to the original z_n. Upon closer inspection we discover that we have introduced a delay of $\frac{L}{2}$ in our estimates of z_n. We could avoid this by using

$$y_n = \frac{1}{2L+1} \sum_{l=-L}^{L} x_{n+l} \approx z_n \tag{6.28}$$

but this requires breaking of causality.

Our final step is to assume that z_n may vary very fast. Using the moving average as defined above will indeed remove the noise, but it will also intolerably average out significant variations in the desired signal itself. In general it may be impossible to significantly attenuate the noise without harming the signal, but we must strive to minimize this harm. One remedy is to notice that the above averaging applies equal weight to all L points in its sum. We may be able to minimize the blurring that this causes by weighting the center of the interval more than the edges. Consider the difference between the following noncausal moving averages.

$$y_n = \tfrac{1}{3}x_{n-1} + \tfrac{1}{3}x_n + \tfrac{1}{3}x_{n+1} \qquad \text{and} \qquad y_n = \tfrac{1}{4}x_{n-1} + \tfrac{1}{2}x_n + \tfrac{1}{4}x_{n+1}$$

The latter more strongly emphasizes the center term, de-emphasizing the influence of inputs from different times. Similarly we can define longer moving averages with coefficients becoming smaller as we move away from the middle (zero) terms.

The most general moving average (MA) filter is

$$y_n = \sum_{l=-L}^{L} h_l x_{n+l} \tag{6.29}$$

where the coefficients h_l need to be chosen to maximize the noise suppression while minimizing the signal distortion. If we are required to be realizable we use

$$y_n = \sum_{l=-L}^{0} h_l x_{n+l} = \sum_{l=0}^{L} h_{l-L} x_{n+l-L} \tag{6.30}$$

although L here will need to be about twice as large, and the output y_n will be delayed with respect to z_n.

EXERCISES

6.6.1 Experiment with the ideas presented in this section as practical techniques for removing noise from a signal. Start with a signal that is constant 1 to which a small amount of Gaussian white noise has been added $x_n = 1 + \epsilon\nu_n$. Try to estimate the constant by adding N consecutive signal values and dividing by N. How does the estimation error depend on ϵ and N?

6.6.2 Perform the same experiment again only this time take the clean signal to be a sinusoid rather than a constant. Attempt to reconstruct the original signal from the noisy copy by using a noncausal moving average with all coefficients equal. What happens when the MA filter is too short or too long?

6.6.3 Now use an MA filter with different coefficients. Take the center coefficient (that which multiplies the present signal value) to be maximal and the others to decrease linearly. Thus for length-three use $(\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$, for length-five use $\frac{1}{9}(1, 2, 3, 2, 1)$, etc. Does this perform better?

6.6.4 Find a noncausal MA differentiator filter, that is, one that approximates the signal's derivative rather than its value. How are this filter's coefficients different from those of the others we have discussed?

6.6.5 A parabola in digital time is defined by $p(n) = an^2 + bn + c$. Given any three signal values x_{-1}, x_0, x_{+1} there is a unique parabola that goes through these points. Given five values $x_{-2}, x_{-1}, x_0, x_{+1}, x_{+2}$ we can find coefficients a, b and c of the *best fitting* parabola $p(n)$, that parabola for which the squared error $\epsilon^2 = (p(-2) - x_{-2})^2 + (p(-1) - x_{-1})^2 + (p(0) - x_0)^2 + (p(+1) - x_{+1})^2 + (p(+2) - x_{+2})^2$ is minimized. We can use this best fitting parabola as a MA *smoothing filter*; for each n we find the best fitting parabola for the 5 signal values $x_{n-2}, x_{n-1}, x_n, x_{n+1}, x_{n+2}$ and output the center value of this parabola. Show that the five-point parabola smoothing filter is an MA filter. What are its coefficients?

6.6.6 After finding the best fitting parabola we can output the value of its derivative at the center. Find the coefficients of this five-point differentiator filter.

6.7 Moving Averages in the Frequency Domain

The operation of an MA filter in the time domain is simple to understand. The filter's input is a signal in the time domain, its output is once again a time domain signal, and the filter coefficients contain all the information needed to transform the former into the latter. What do we mean by the frequency domain description of a filter? Recall that the operation of a

filter on a signal has a simple frequency domain interpretation. The spectrum of a filter's output signal is the input signal's spectrum multiplied by a frequency-dependent weighting function. This weighting function is what we defined in Section 6.3 as the filter's *frequency response.* In Section 6.12 we will justify this identification of the frequency response as the fundamental frequency domain description. For now we shall just assume that the frequency response is the proper attribute to explore.

We originally defined the frequency response as the output of a filter given a real sinusoid of arbitrary frequency as input. In this section we extend our original definition by substituting *complex exponential* for *sinusoid.* As usual the main reason for this modification is mathematical simplicity; it is just easier to manipulate exponents than trigonometric functions. We know that at the end we can always extract the real part and the result will be mathematically identical to that we would have found using sinusoids.

Let's start with one of the simplest MA filters, the noncausal, equally weighted, three-point average.

$$y_n = \tfrac{1}{3}(x_{n-1} + x_n + x_{n+1}) \tag{6.31}$$

In order to find its frequency response $H(\omega)$ we need to substitute

$$x_n = e^{\mathrm{i}\omega n}$$

and since the moving average is a filter, we know that the output will be a complex exponential of the same frequency.

$$y_n = H(\omega)e^{\mathrm{i}\omega n}$$

Substituting

$$y_n = \tfrac{1}{3}\left(e^{\mathrm{i}\omega(n-1)} + e^{\mathrm{i}\omega n} + e^{\mathrm{i}\omega(n+1)}\right) = \tfrac{1}{3}\left(e^{-\mathrm{i}\omega} + 1 + e^{\mathrm{i}\omega}\right)e^{\mathrm{i}\omega n}$$

we immediately identify

$$H(\omega) = \tfrac{1}{3}\left(1 + e^{-\mathrm{i}\omega} + e^{\mathrm{i}\omega}\right) = \tfrac{1}{3}\left(1 + 2\cos(\omega)\right) \tag{6.32}$$

as the desired frequency response. If we are interested in the energy at the various frequencies, we need the square of this, as depicted in Figure 6.5. We see that this system is somewhat low-pass in character (i.e., lower frequencies are passed while higher frequencies are attenuated). However, the attenuation does not increase monotonically with frequency, and in fact the highest possible frequency $\frac{1}{2}f_s$ is not well attenuated at all!

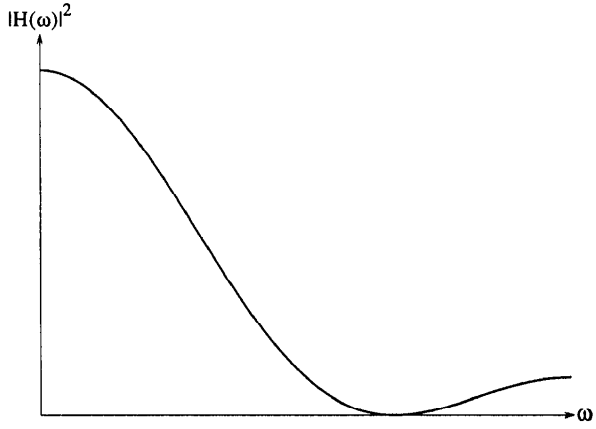


Figure 6.5: The (squared) frequency response of the simple three-point average filter. The response is clearly that of a low-pass filter, but not an ideal one.

At the end of the previous section we mentioned another three-point moving average.

$$y_n = \tfrac{1}{4}x_{n-1} + \tfrac{1}{2}x_n + \tfrac{1}{4}x_{n+1} \tag{6.33}$$

Proceeding as before we find

$$y_n = \tfrac{1}{4}e^{\mathrm{i}\omega(n-1)} + \tfrac{1}{2}e^{\mathrm{i}\omega n} + \tfrac{1}{4}e^{\mathrm{i}\omega(n+1)} = \left(\tfrac{1}{4}e^{-\mathrm{i}\omega} + \tfrac{1}{2} + \tfrac{1}{4}e^{\mathrm{i}\omega}\right)e^{\mathrm{i}\omega n}$$

and can identify

$$H(\omega) = \left(\tfrac{1}{4}e^{-\mathrm{i}\omega} + \tfrac{1}{2} + \tfrac{1}{4}e^{\mathrm{i}\omega}\right) = \tfrac{1}{2}\left(1 + \cos(\omega)\right) \tag{6.34}$$

a form known as a 'raised cosine'.

This frequency response, contrasted with the previous one in Figure 6.6 is also low-pass in character, and is more satisfying since it *does* go to zero at $\frac{1}{2}f_s$. However it is far from being an ideal low-pass filter that drops to zero response above some frequency; in fact it is wider than the frequency response of the simple average.

What happens to the frequency response when we average over more signal values? It is straightforward to show that for the simplest case

$$y_n = \frac{1}{2L+1}\sum_{l=-L}^{L} x_{n+l} \tag{6.35}$$

the frequency response is

$$\frac{\sin(\frac{Lx}{2})}{L\sin(\frac{x}{2})} \tag{6.36}$$

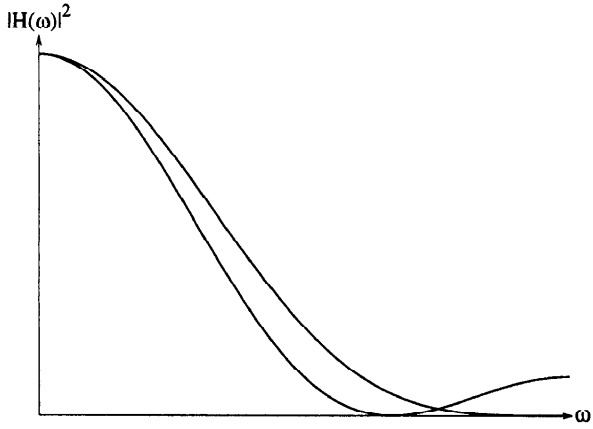


Figure 6.6: The (squared) frequency responses of two simple three-point average filters. Both responses are clearly low-pass but not ideal. The average with coefficients goes to zero at $\frac{1}{2}f_s$, but is 'wider' than the simple average.

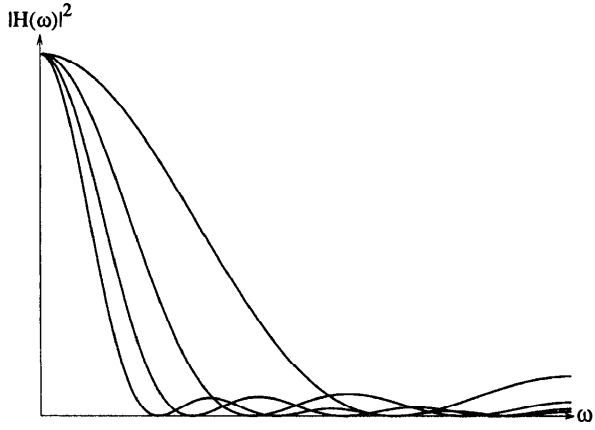


Figure 6.7: The (squared) frequency responses of simple averaging filters for $L = 3, 5, 7$ and 9. We see that as L increases the pass-band becomes narrower, but oscillations continue.

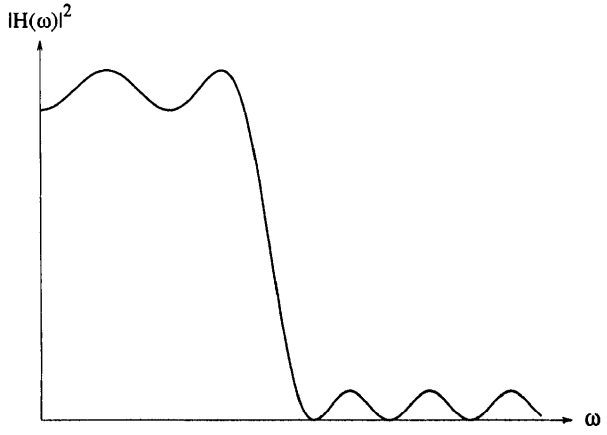


Figure 6.8: The (squared) frequency responses of a 16-coefficient low-pass filter. With these coefficients the lower frequency components are passed essentially unattenuated, while the higher components are strongly attenuated.

as is depicted in Figure 6.7 for $L = 3, 5, 7, 9$. We see that as L increases the filter becomes more and more narrow, so that for large L only very low frequencies are passed. However, this is only part of the story, since even for large L the oscillatory behavior persists. Filters with higher L have a narrower main lobe but more sidelobes.

By using different coefficients we can get different frequency responses. For example, suppose that we need to pass frequencies below half the Nyquist frequency essentially unattenuated, but need to block those above this frequency as much as possible. We could use a 16-point moving average with the following magically determined coefficients

$$\begin{array}{rrrr} 0.003936, & -0.080864, & 0.100790, & 0.012206, \\ -0.090287, & -0.057807, & 0.175444, & 0.421732, \\ 0.421732, & 0.175444, & -0.057807, & -0.090287, \\ 0.012206, & 0.100790, & -0.080864, & 0.003936 \end{array}$$

the frequency response of which is depicted in Figure 6.8. While some oscillation exists in both the pass-band and the stop-band, these coefficients perform the desired task relatively well.

Similarly we could find coefficients that attenuate low frequencies but pass high ones, or pass only in a certain range, etc. For example, another simple MA filter can be built up from the finite difference.

$$y_n = \Delta x_n = x_n - x_{n-1} \tag{6.37}$$

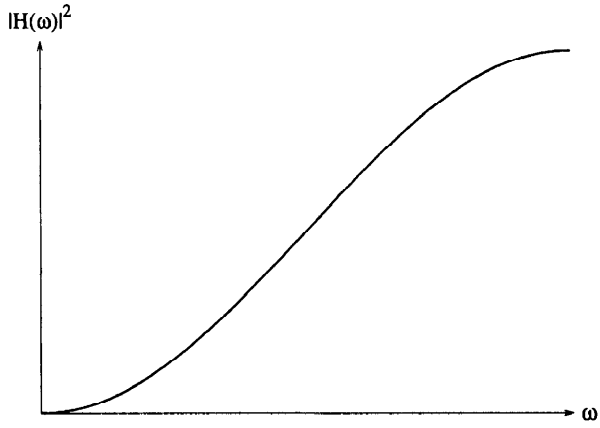


Figure 6.9: The (squared) frequency response of a finite difference filter. With these coefficients the lower frequency components are passed essentially unattenuated, while the higher components are strongly attenuated.

It is easy to show that its frequency response (see Figure 6.9) attenuates low and amplifies high frequencies.

EXERCISES

6.7.1 Calculate the frequency response for the simple causal moving average.

$$y_n = \frac{1}{L}\sum_{k=0}^{L-1} x_{n-k}$$

Express your result as the product of an amplitude response and a phase response. Compare the amplitude response to the one derived in the text for equally weighted samples? Explain the phase response.

6.7.2 Repeat the previous exercise for the noncausal case with an even number of signal values. What is the meaning of the phase response now?

6.7.3 Verify numerically that the 16-point MA filter given in the text has the frequency response depicted in Figure 6.8 by injecting sinusoids of various frequencies.

6.7.4 Find the squared frequency response of equation (6.37).

6.7.5 Find an MA filter that passes intermediate frequencies but attenuates highs and lows.

6.7.6 Find nontrivial MA filters that pass all frequencies unattenuated.

6.7.7 The second finite difference Δ^2 is the finite difference of the finite difference, i.e., $\Delta^2 x_n = \Delta(x_n - x_{n-1}) = x_n - 2x_{n-1} + x_{n-2}$. Give explicit formulas for the third and fourth finite differences. Generalize your results to the k^{th} order finite difference. Prove that $y_n = \Delta^k x_n$ is an MA filter with $k+1$ coefficients.

6.8 Why Convolve?

The first time one meets the convolution sum

$$x * y = \sum_i x_i \, y_{k-i}$$

one thinks of the algorithm

```
Given x, y, k
Initialize: conv ← 0; i, j
Loop:
     increment conv by x_i y_j
     increment i
     decrement j
```

and can't conceive of any good reason to have the two indices moving in opposite directions. Surely we can always redefine y_j and rephrase this as our original MA filter *moving average*

```
Given x, y
Initialize: conv ← 0; i, j
Loop:
     increment conv by x_i y_i
     increment i
```

saving a lot of confusion. There must be some really compelling reason for people to prefer this strange (or should I say *convoluted*) way of doing things. We will only fully understand why the convolution way of indexing is more prevalent in DSP in Section 6.12, but for now we can somewhat demystify the idea.

We'll start by considering two polynomials of the second degree.

$$\begin{aligned} A(x) &= a_0 + a_1 x + a_2 x^2 \\ B(x) &= b_0 + b_1 x + b_2 x^2 \end{aligned}$$

Their product is easily found,

$$\begin{aligned} A(x)B(x) &= a_0b_0 + (a_0b_1 + a_1b_0)x + (a_0b_2 + a_1b_1 + a_2b_0)x^2 \\ &\quad + (a_1b_2 + a_2b_1)x^3 + a_2b_2\,x^4 \end{aligned}$$

and the connection between the indices seems somewhat familiar. More generally, for any two polynomials

$$\begin{aligned} A(x) &= \sum_{i=0}^{N} a_i\,x^i \\ B(x) &= \sum_{j=0}^{M} b_j\,x^j \end{aligned}$$

we have

$$A(x)B(x) = \sum_{k=0}^{N+M} \left(\sum_{\substack{i,j \\ i+j=k}} a_ib_j \right) x^k = \sum_{k=0}^{N+M} (a * b)_k\,x^k$$

and the fundamental reason for these indices to run in opposite directions is obvious—the two exponents must sum to a constant!

Put this way the idea of indices running in opposite directions isn't so new after all. In fact you probably first came across it in grade school. Remember that an integer is represented in base-10 as a polynomial in 10, $A = \sum_{i=0}^{d_A} a_i 10^i$ (where d_A is the number of digits). Thus multiplication of two *integers* A and B is also really a convolution.

$$AB = \sum_{k=0}^{d_A+d_B} (a * b)_k 10^k$$

The algorithm we all learned as *long multiplication* is simply a tabular device for mechanizing the calculation of the convolution.

$$\begin{array}{cccccccc} & & & A_N & A_{N-1} & \cdots & A_1 & A_0 \\ & & * & B_N & B_{N-1} & \cdots & B_1 & B_0 \\ \hline & & & B_0A_N & B_0A_{N-1} & \cdots & B_0A_1 & B_0A_0 \\ & & B_1A_N & B_1A_{N-1} & & \cdots & B_1A_0 & \\ & & & & & \vdots & & \\ B_NA_N & \cdots & B_NA_1 & B_NA_0 & & & & \\ \hline C_{2N} & & C_{N+1} & C_N & C_{N-1} & \cdots & C_1 & C_0 \end{array}$$

We now understand why the symbol '$*$' is used for convolution; there is a simple connection between convolution and multiplication!

Anyone who is comfortable with multiplication of integers and polynomials is automatically at home with convolutions. There is even a formalism for turning *every* convolution into a polynomial multiplication, namely the *z transform* (see section 4.10). The basic idea is to convert every digital signal x_n into an equivalent polynomial

$$x_n \longleftarrow X(d) = \sum_n d^n$$

although it is conventional in DSP to use z^{-1} instead of d. Then the convolution of two digital signals can be performed by multiplying their respective z transforms. You can think of z transforms as being similar to logarithms. Just as logarithms transform multiplications into additions, z transforms transform convolutions into multiplications.

We have seen an isomorphism between convolution and *polynomial* products, justifying our statement that convolution is analogous to multiplication. There is also an isomorphism with yet another kind of multiplication that comes in handy. The idea is to view the signal x_n as a vector in N-dimensional space, and the process of convolving it with some vector h_n as an operator that takes x_n and produces some new vector y_n. Now since linear operators can always be represented as matrices, convolution is also related to *matrix* multiplication. To see this explicitly let's take the simple case of a signal x_n that is nonzero only between times $n = 0$ and $n = 4$ so that it is analogous to the vector $(x_0, x_1, x_2, x_3, x_4)$. Let the filter h_n have three nonzero coefficients h_{-1}, h_0, h_1 so that it becomes the vector (h_{-1}, h_0, h_1). The convolution $y = h * x$ can only be nonzero between times $n = -1$ and $n = 5$, but we will restrict our attention to times that correspond to nonzero x_n. These are given by

$$\begin{aligned}
y_0 &= h_0 x_0 + h_{-1} x_1 \\
y_1 &= h_{-1} x_0 + h_0 x_1 + h_{-1} x_2 \\
y_2 &= h_{-1} x_1 + h_0 x_2 + h_{-1} x_3 \\
y_3 &= h_{-1} x_2 + h_0 x_3 + h_{-1} x_4 \\
y_4 &= h_{-1} x_3 + h_0 x_4
\end{aligned}$$

and more compactly written in matrix form.

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} h_0 & h_1 & 0 & 0 & 0 \\ h_{-1} & h_0 & h_1 & 0 & 0 \\ 0 & h_{-1} & h_0 & h_1 & 0 \\ 0 & 0 & h_{-1} & h_0 & h_1 \\ 0 & 0 & 0 & h_{-1} & h_0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

The matrix has quite a distinctive form, all elements on each diagonal being equal. Such a matrix is said to be *Toeplitz* in structure, and Toeplitz matrices tend to appear quite a lot in DSP.

EXERCISES

6.8.1 N.G. Kneer, the chief DSP hardware engineer at NeverWorks Incorporated, purchases pre-owned DSP processors on an as-is basis from two suppliers, Alpha Numerics and Beta Million. From Alpha Numerics one gets a perfect lot 30% of the time, but 10% of the time all five chips are bad. From Beta one never gets a completely bad lot, but only gets a perfect lot 10% of the time. In fact, N.G. has come up with the following data regarding the probability of k defective chips out of a lot of five.

k	A_k	B_k
0	0.3	0.1
1	0.2	0.2
2	0.2	0.4
3	0.1	0.2
4	0.1	0.1
5	0.1	0.0

In order to reduce his risk, N.G. buys from both suppliers. Assuming he buys ten chips, five chips from each, what should he expect to be the distribution of defective chips? What is the connection between this and convolution?

6.8.2 Dee Espy has to purchase 100 DSPs, and decides that she should strive to minimize the number of defective chips she purchases. How many should she buy from each of the above suppliers?

6.8.3 Convolve the signal $x_n = \ldots, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0 \ldots$ with the filter $h = (1, 1, 1)$ and plot x_n, h_n and the output y_n. Convolve y_n with h_n resulting in $y^{[2]}1_n$ and again this new signal to get $y_n^{[3]}$, etc. Plot $x_n = y_n^{[0]}$, $y_n = y_n^{[1]}$, $y^{[2]}1_n$, $y^{[3]}1_n$, one under the other. What can be said about the effect of this consecutive filtering?

6.9 Purely Recursive Systems

In this section we will deal with purely recursive systems, that is systems for which y_n depends on previous y values and the present x_n, but not on previous x values. You should know that DSP engineers call such systems **A**uto**R**egressive (AR) systems, a name coined by G. Udny Yule in 1927. The word regression here refers to *regression analysis*, a well-known statistical method for finding the relationship of a variable to other variables. Yule was studying the number of sunspots observed as a function of time and decided to attempt to relate the present sunspot activity to previous values of the *same* quantity using regression analysis. He thus called this technique *autoregression* analysis. We prefer the name 'purely recursive' to autoregressive, but will nonetheless adopt the prevalent abbreviation 'AR'.

For AR systems the output y_n is obtained from the input x_n by

$$y_n = x_n + \sum_{m=1}^{M} b_m y_{n-m} \tag{6.38}$$

and to start up the recursion we have to make some assumption as to earlier outputs (e.g., take them to be zero). If the input signal was zero before time $n = 0$ then any causal system will have $y_n = 0$ for all negative n. However, if we choose to start the recursion at $n = 0$ but the input actually preexisted, the zeroing of the previous outputs is contrived.

Let's return to the problem introduced in Section 6.6 of finding the true value of a constant signal obscured by additive noise. Our first attempt was to simply average up some large number L of signal values.

$$y_{L-1} = \frac{1}{L} \sum_{l=0}^{L-1} x_l$$

Were we to determine that L values were not sufficient and we wished to try $L+1$, we do not have to add up all these values once again. It is easy to see that we need only multiply y_{L-1} by L to regain the sum, add the next input x_L, and divide by the new number of signal values $L+1$.

$$y_L = \frac{1}{L+1} (L\, y_{L-1} + x_L)$$

This manipulation has converted the original iteration into a recursion

$$y_L = \alpha\, x_L + \beta\, y_{L-1} \qquad \text{where } \alpha = \frac{1}{L},\ \beta = \frac{L}{L+1}$$

and we have the relation $\alpha + \beta = 1$. Changing the index to our more usual n we can now write

$$y_n = (1-\beta)x_n + \beta y_{n-1} \qquad 0 \leq \beta < 1 \tag{6.39}$$

which is of the AR form with $M = 1$.

The nice thing about equation (6.38) is that it is already suitable for rapidly varying signals. We needn't go through all the stages that lead to the MA filter with coefficients; by changing β this AR filter can be set to track rapidly varying signals or to do a better job of removing noise from slowly varying ones. When $\beta = 0$ (corresponding to $L = 0$) the AR filter output y_n is simply equal to the input, no noise is averaged out but no bandwidth lost either. As β increases the past values assume more importance, and the averaging kicks in at the expense of not losing the ability to track the input as rapidly. When $\beta \to 1$ (corresponding to infinite L) the filter paradoxically doesn't look at the current input at all!

Equation (6.39) is similar to the causal version of the moving average filter of equation (6.30) in that it moves along the signal immediately outputting the filtered signal. However, unlike the moving average filter, equation (6.39) never explicitly removes a signal value that it has seen from its consideration. Instead, past values are slowly 'forgotten' (at least for $\beta < 1$). For large β signal values from relatively long ago are still relatively important, while for small β past values lose their influence rapidly. You can think of this AR filter as being similar to an MA filter operating on L previous values, the times before $n - L$ having been forgotten. To see this, unravel the recursion in equation (6.39).

$$y_n = (1-\beta)x_n + \beta(1-\beta)x_{n-1} + \beta^2(1-\beta)x_{n-2} + \beta^3(1-\beta)x_{n-3} + \ldots \tag{6.40}$$

We see that the coefficient corresponding to x_{n-l} is smaller than that of x_n by a factor of β^l, and so for all practical purposes we can neglect the contributions for times before some l. For example, if $\beta = 0.99$ and we neglect terms that are attenuated by e^{-1}, we need to retain about 100 terms; however for $\beta = 0.95$ only about 20 terms are needed, for $\beta = 0.9$ we are down to ten terms, and for $\beta = 0.8$ to 5 terms. It is not uncommon to use $\beta = 0.5$ where only the x_n and x_{n-1} terms are truly relevant, the x_{n-2} term being divided by 4.

We should now explore the frequency response $H(\omega)$ of our AR filter. Using the technique of Section 6.7 we assume

$$x_n = e^{i\omega n}$$

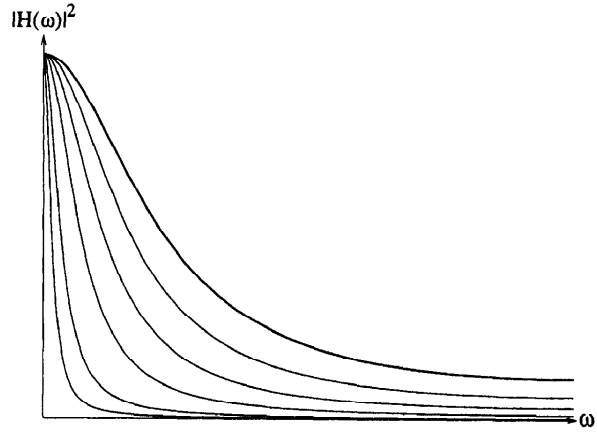


Figure 6.10: The (squared) frequency response of the simple AR low-pass filter for several different values of β. From top to bottom $\beta = 0.5, 0.6, 0.7, 0.8, 0.9, 0.95$.

and since the AR filter is a filter, we know that the output will be a complex exponential of the same frequency.

$$y_n = H(\omega)e^{i\omega n}$$

Substituting from equation (6.40) and using equation (A.47)

$$\begin{aligned} y_n &= (1-\beta)e^{i\omega n} + \beta(1-\beta)e^{i\omega(n-1)} + \beta^2(1-\beta)e^{i\omega(n-2)} + \ldots \\ &= (1-\beta)\sum_{k=0}^{\infty}(\beta e^{-i\omega})^k e^{i\omega n} \\ &= \frac{(1-\beta)}{(1-\beta e^{-i\omega})}e^{i\omega n} \end{aligned}$$

and we immediately identify

$$H(\omega) = \frac{(1-\beta)}{(1-\beta e^{-i\omega})} \tag{6.41}$$

as the desired frequency response, and

$$|H(\omega)|^2 = \frac{1-2\beta+\beta^2}{1-2\beta\cos(\omega)+\beta^2} \tag{6.42}$$

as its square. We plot this squared frequency response, for several values of β, in Figure 6.10.

Another useful AR filter is

$$y_n = x_n + y_{n-1} \tag{6.43}$$

which unravels to the following infinite sum:

$$y_n = x_n + x_{n-1} + x_{n-2} + \ldots = \sum_{m=0}^{\infty} x_{n-m}$$

We can write this in terms of the time delay operator

$$y = (1 + \mathrm{z}^{-1} + \mathrm{z}^{-2} + \ldots)x = \Upsilon x$$

where we have defined the infinite accumulator operator

$$\Upsilon \equiv \sum_{m=0}^{\infty} \mathrm{z}^{-m} x_n \tag{6.44}$$

which roughly corresponds to the integration operator for continuous signals. The finite difference $\Delta \equiv (1 - \mathrm{z}^{-1})$ and the infinite accumulator are related through $\Delta\Upsilon = 1$ and $\Upsilon\Delta = 1$, where 1 is the identity operator.

What happens when the infinite accumulator operates on a constant signal? Since we are summing the same constant over and over again the sum obviously gets larger and larger in absolute value. This is what we called *instability* in Section 6.4, since the output of the filter grows without limit although the input stays small. Such unstable behavior could never happen with an MA filter; and it is almost always an unwelcome occurrence, since all practical computational devices will eventually fail when signal values grow without limit.

EXERCISES

6.9.1 What is the exact relation between β in equation (6.39) and the amount of past time τ that is still influential? Define 'influential' until the decrease is by a factor of e^{-1}. Graph the result.

6.9.2 Quantify the bandwidth BW of the AR filter as a function of β and compare it with the influence time τ.

6.9.3 Contrast the squared frequency response of the AR filter (as depicted in Figure 6.10) with that of the simple averaging MA filter (Figure 6.7). What can you say about the amount of computation required for a given bandwidth?

6.9.4 Find an AR filter that passes high frequencies but attenuates low ones. Find an AR filter that passes intermediate frequencies but attenuates highs and lows.

6.9.5 Calculate the effect of the infinite accumulator operator Υ on the following signals, and then check by generating the first 10 values.

1. $x_n = u_n$ where u_n is the unit step of equation (2.4)
2. $x_n = (-1)^n u_n = 1, -1, 1, -1, \ldots$
3. $x_n = nu_n = 0, 1, 2, 3, \ldots$
4. $x_n = \alpha^n u_n$ where $\alpha \neq 1$

6.9.6 Apply the finite difference operator to the results obtained in the previous exercise, and show that $\Delta\Upsilon = 1$.

6.9.7 Prove that $\Upsilon\Delta = 1$. (Hint: Prove that the 'telescoping' series $x_1 - x_0 + x_2 - x_1 \ldots a_n - a_{n-1} = x_0 + x_n$.)

6.9.8 What is the condition for $y_n = \alpha x_n + \beta y_{n-1}$ to be stable? (Hint: Take a constant input $x_n = 1$ and compute y_n when $n \to \infty$.)

6.10 Difference Equations

We have seen that there are MA filters, with output dependent on the present and previous inputs, and AR filters, with output dependent on the present input and previous outputs. More general still are combined ARMA filters, with output dependent on the present input, L previous inputs, and M previous outputs.

$$y_n = \sum_{l=0}^{L} a_l x_{n-l} + \sum_{m=1}^{M} b_m y_{n-m} \tag{6.45}$$

When $M = 0$ (i.e., all b_m are zero), we have an MA filter $y_n = \sum_{l=0}^{L} a_l x_{n-l}$, while $L = 0$ (i.e., all $a_l = 0$ except a_0), corresponds to the AR filter $y_n = x_n + \sum_{m=1}^{M} b_m y_{n-m}$.

To convince yourself that ARMA relationships are natural consider the amount of money y_n in a bank account at the end of month n, during which x_n is the total amount deposited (if $x_n < 0$ more was withdrawn than deposited) and interest from the previous month is credited according to a rate of i.

$$y_n = y_{n-1} + x_n + iy_{n-2}$$

A slightly more complex example is that of a store-room that at the end of day n contains y_n DSP chips, after x_n chips have been withdrawn from

stock that day. The requisitions clerk orders new chips based on the average usage over the past two days $\frac{1}{2}(x_n + x_{n-1})$, but these are only delivered the next day. The number of chips in stock is thus given by an ARMA system.

$$y_n = y_{n-1} - x_n + \tfrac{1}{2}(x_{n-1} + x_{n-2})$$

Equation (6.45) can also be written in a more symmetric form

$$\sum_{m=0}^{M} \beta_m y_{n-m} = \sum_{l=0}^{L} \alpha_l x_{n-l} \tag{6.46}$$

where

$$\alpha_l = a_l \qquad \beta_0 \equiv 1 \qquad \beta_m = -b_m \quad \text{for} \quad m = 1 \ldots M$$

although this way of expressing the relationship between y and x hides the fact that y_n can be simply derived from previous x and y values. This form seems to be saying that the x and y signals are both equally independent, but happen to obey a complex relationship involving present and past values of both x and y. In fact the symmetric form equally well describes the inverse system.

$$x_n = y_n - \sum_{m=1}^{M} b_m y_{n-m} - \sum_{l=1}^{L} a_l x_{n-l}$$

We can formally express equation (6.46) using the time delay operator

$$\sum_{m=0}^{M} \beta_m \mathrm{z}^{-m} y_n = \sum_{l=0}^{L} \alpha_l \mathrm{z}^{-l} x_n \tag{6.47}$$

and (as you will demonstrate in the exercises) in terms of finite differences.

$$\sum_{m=0}^{M} B_m \, \Delta^m y_n = \sum_{l=0}^{L} A_l \, \Delta^l x_n \tag{6.48}$$

Recalling from Section 2.4 that the finite difference operator bears some resemblance to the derivative, this form bears some resemblance to a linear differential equation

$$\sum_{m=0}^{M} \beta_m y^{[n-m]}(t) = \sum_{l=0}^{L} \alpha_l x^{[n-l]}(t) \tag{6.49}$$

where $x^{[k]}$ is the k^{th} derivative of $x(t)$ with respect to t. For this reason ARMA systems are often called *difference equations.*

Derivatives are defined using a limiting process over differences. In DSP the time differences cannot be made smaller than the sampling interval T, and thus finite differences take the place of differentials, and difference equations replace differential equations.

There are many similarities between differential equations and difference equations. The recursion (6.45) is a prescription for generating y_n given initial conditions (e.g., all x_n and y_n are zero for $n < 0$); similarly solutions to differential equations need initial conditions to be specified. The most general solution of a difference equation can be written as a solution to the equation with zero input $x_n = 0$ and any particular solution with the actual input; readers familiar with differential equations know that general solutions to linear differential equations are obtained from the homogeneous solution plus a particular solution. Linear differential equations with constant coefficients can be solved by assuming solutions of the form $y(t) = Ae^{\lambda t}$; solutions to linear difference equations can be similarly found by assuming $y_n = z^n$, which is why we have been finding frequency responses by assuming $y_n = e^{i\omega t}$ all along.

Differential equations arise naturally in the analysis and processing of analog signals, because derivatives describe changes in signals over short time periods. For example, analog signals that have a limited number of frequency components have short time predictability, implying that only a small number of derivatives are required for their description. More complex signals involve more derivatives and extremely noisy analog signals require many derivatives to describe. Similarly digital signals that contain only a few sinusoids can be described by difference equations of low order while more complex difference equations are required for high-bandwidth signals.

EXERCISES

6.10.1 Difference equations are not the only tool for describing ARMA systems; the state-space description explicitly uses the system's memory (internal state). Denoting the input x_n, the output y_n, and the vector of internal state variables at time n by $\underline{s}_n$, the state equation description relates the output to the present input and system state and furthermore describes how to update the state given the input.

$$\begin{aligned} y_n &= \underline{f} \cdot \underline{s}_{n-1} + g\,x_n \\ \underline{s}_{n+1} &= \underline{\underline{A}}\,\underline{s}_n + x_n\,\underline{c} \end{aligned}$$

Relate the state equation parameters $\underline{f}$, g, $\underline{\underline{A}}$, and $\underline{c}$ to those of the ARMA description, $\underline{a}$ and $\underline{b}$.

6.10.2 Devise a circumstance that leads to an ARMA system with $L = 2$ and $M = 2$. $L = 1$ and $M = 3$.

6.10.3 For any digital signal s, we recursively define $s^{[m]}$, the *finite difference of order m*.

$$\begin{aligned} s_n^{[0]} &= s_n \\ s_n^{[n]} &= s_n^{[n-1]} - s_{n-1}^{[n-1]} \end{aligned}$$

For example, given the sequence $a_n = 2n + 1$ we find the following.

$$\begin{array}{lcccccccc} a_0 & = & 1 & & 3 & & 5 & & 7 & & 9 & \ldots \\ a_1 & = & & 2 & & 2 & & 2 & & 2 & \ldots & \\ a_2 & = & & & 0 & & 0 & & 0 & \ldots & & \\ a_3 & = & & & & 0 & & 0 & & \ldots & & \\ a_4 & = & & & & & 0 & & & \ldots & & \end{array}$$

We see here that the second and higher finite differences are all zero. In general, when the sequence is of the form $s_n = c_m n^m + c_{m-1} n^{m-1} + \cdots + c_1 n + c_0$, the $m + 1$ order finite differences are zero. This fact can be used to identify sequences. Find the finite differences for the following sequences, and then identify the sequence.

$$\begin{array}{cccccccc} 3 & 6 & 9 & 12 & 15 & 18 & \ldots \\ 3 & 6 & 11 & 18 & 27 & 38 & \ldots \\ 3 & 6 & 13 & 24 & 39 & 58 & \ldots \\ 3 & 6 & 17 & 42 & 87 & 158 & \ldots \\ 3 & 6 & 27 & 84 & 195 & 378 & \ldots \end{array}$$

6.10.4 Find the first, second, and third finite difference sequences for the following sequences.

1. $s_n = an$
2. $s_n = bn^2$
3. $s_n = cn^3$
4. $s_n = bn^2 + an$
5. $s_n = cn^3 + bn^2 + an$

6.10.5 Show that if $x_n = \sum_k^L a_k n^k$ then the $(L+1)^{\text{th}}$ finite difference is zero.

6.10.6 Plot the first, second and third differences of $s_n = \sin(2\pi f n)$ for frequencies $f = 0.1, 0.2, 0.3, 0.4$.

6.10.7 $a_0 x_n + a_1 x_{n-1}$ can be written $A_0 x_n + A_1 \Delta x_n$ where $a_1 = -A_1$ and $a_0 = A_0 + A_1$. What is the connection between the coefficients of $a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2}$ and $A_0 x_n + A_1 \Delta x_n + A_2 \Delta^2 x_n$? What about $a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} + a_3 x_{n-3}$ and $A_0 x_n + A_1 \Delta x_n + A_2 \Delta^2 x_n + A_3 \Delta^3 x_n$? Generalize and prove that all ARMA equations can be expressed as difference equations.

6.11 The Sinusoid's Equation

The usefulness of MA and AR filters can be clarified via a simple example. The analog sinusoid $s(t) = A\sin(\Omega t + \phi)$ not only has a simple spectral interpretation, but also obeys the second-order *differential* equation

$$\ddot{s}(t) + \Omega^2 s(t) = 0 \tag{6.50}$$

commonly called the equation of simple harmonic motion. Indeed this equation can be considered to be the defining equation for the family of analog sinusoidal signals, and its simplicity can be used as an alternate explanation of the importance of these signals.

In the digital domain we would expect digital sinusoids to obey a second-order *difference* equation. That this is indeed the case can be shown using the trigonometric identities (A.23)

$$\begin{aligned}
\sin\Big(\Omega(t-2T)\Big) &= \sin\Omega t\,\cos 2\Omega T - \cos\Omega t\,\sin 2\Omega T \\
&= \sin\Omega t\,\Big(2\cos^2\Omega T - 1\Big) + \cos\Omega t\,\Big(2\sin\Omega T\cos\Omega T\Big) \\
&= -\sin\Omega t + 2\cos\Omega T\,\Big(2\sin\Omega T\cos\Omega T\Big) \\
&= -\sin\Omega t + 2\cos\Omega T\,\sin\Omega(t-T)
\end{aligned}$$

which can easily be shown to be

$$s(t-2T) \;-\; 2\cos(\Omega T)s(t-T) \;+\; s(t) = 0 \tag{6.51}$$

or in digital form

$$s_n + c_1 s_{n-1} + c_2 s_{n-2} = 0 \tag{6.52}$$

where $c_1 = -2\cos\Omega T$ and $c_2 = 1$.

This difference equation, obeyed by all sinusoids, can be exploited in several different ways. In the most direct implementation it can be used as a *digital oscillator* or tone generator, i.e., an algorithm to generate sinusoidal signals. Given the desired digital oscillation frequency ω_d, amplitude A, and initial phase ϕ, we precompute the coefficient

$$c_1 = -2\cos\Omega T = -2\cos\frac{\Omega}{f_s} = -2\cos\Omega_d$$

and the first two signal values

$$\begin{aligned}
s_0 &= A\sin(\phi) \\
s_1 &= A\sin(\Omega T + \phi) = A\sin(\Omega_d + \phi)
\end{aligned}$$

where Ω_d is the digital frequency (we will omit the subscript from here on). The difference equations now recursively supply all signal values:

$$\begin{aligned} s_2 &= -(c_1 s_1 + s_0) \\ s_3 &= -(c_1 s_2 + s_1) \\ s_4 &= -(c_1 s_3 + s_2) \end{aligned}$$

and so on. This digital oscillator requires only one multiplication, one addition, and one sign reversal per sample point! This is remarkably efficient when compared with alternative oscillator implementations such as approximation of the sine function by a polynomial, table lookup and interpolation, or direct application of trigonometric addition formulas. The main problems with this implementation, like those of all purely recursive algorithms, are those of accuracy and stability. Since each result depends on the previous two, numerical errors tend to add up, and eventually swamp the actual calculation. This disadvantage can be rectified in practice by occasionally resetting with precise values.

Another application of the difference equation (6.51) is the removal of an interfering sinusoid. Given an input signal x_n contaminated with an interfering tone at known frequency Ω; we can subtract the sinusoidal component at this frequency by the following MA filter

$$y_n = x_n + c_1 x_{n-1} + x_{n-2}$$

where c_1 is found from Ω. The frequency response is found by substituting an arbitrary complex exponential $e^{i\omega n}$

$$\begin{aligned} y_n &= e^{i\omega n} - 2\cos\Omega e^{i\omega(n-1)} + e^{i\omega(n-2)} \\ &= \left(1 - 2\cos\Omega e^{-i\omega} + e^{-2i\omega}\right) e^{i\omega n} \\ &= e^{-i\omega}\left(e^{i\omega} - 2\cos\Omega + e^{-i\omega}\right) e^{i\omega n} \\ &= e^{-i\omega}\left(2\cos\omega - 2\cos\Omega\right) e^{i\omega n} \end{aligned}$$

which can be written in the form $y_n = H(\omega)x_n$. The square of $H(\omega)$ is depicted in Figure 6.11 for digital frequency $\frac{1}{2}$. Note that no energy remains at the interfering frequency; the system is a *notch filter.*

Finally the difference equation (6.52) can be used to estimate the frequency of a sine buried in noise. The idea is to reverse the equation, and using observed signal values to estimate the value of c_1. From this the frequency ω can be derived. Were no noise to be present we could guarantee

$$c_1 = -\frac{x_n + x_{n-2}}{x_{n-1}}$$

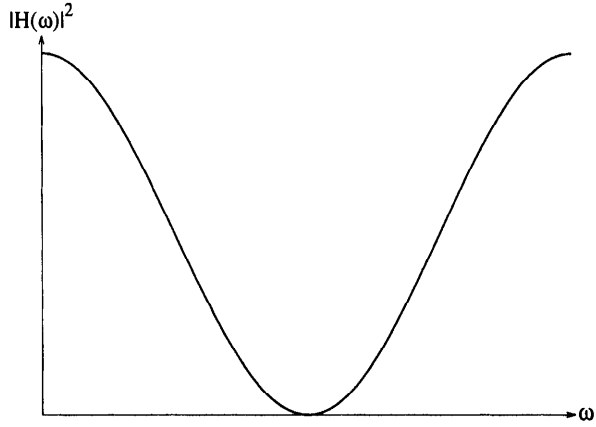


Figure 6.11: The (squared) frequency response of the MA notch filter set to one-half the Nyquist frequency. Note that no energy remains at the notched frequency.

but with noise this only holds on average

$$c_1 = -\left\langle \frac{x_n + x_{n-2}}{x_{n-1}} \right\rangle = -\left\langle \frac{(x_n + x_{n-2})x_{n-1}}{|x_{n-1}|^2} \right\rangle$$

the second form being that most commonly used.

EXERCISES

6.11.1 Show that the signal $s_n = e^{qn}$ obeys the equation $s_n = as_{n-1}$ where $a = e^q$.

6.11.2 Show that the signal $s_n = \sin(\Omega n)$ obeys the equation $s_n = a_1 s_{n-1} + a_2 s_{n-2}$ with coefficients a_i determined by the equation $1 - a_1 z^{-1} - a_2 z^{-2} = 0$ having solutions $z = e^{\pm i\Omega}$.

6.11.3 Show that if a signal is the sum of p exponentials

$$s_n = \sum_{i=1}^{p} A_i e^{q_i n}$$

then the equation $1 - \sum_k a_k z^{-k} = 0$ has roots $z = e^{q_i}$.

6.11.4 Generalize the previous exercises and demonstrate that the sum of p sines obeys a recursion involving $2p$ previous values. What is the equation and how are its coefficients determined?

6.12 System Identification—The Easy Case

Assume that someone brings you a signal processing system enclosed in a black box. The box has two connectors, one marked *input* and the other *output*. Other than these labels there are no identifying marks or documentation, and nothing else is known about what is hidden inside. What can you learn about such a system? Is there some set of measurements and calculations that will enable you to accurately predict the system's output when an arbitrary input is applied? This task is known as *system identification*.

You can consider system identification as a kind of game between yourself and an opponent. The game is played in the following manner. Your opponent brings you the black box (which may have been specifically fabricated for the purpose of the game). You are given a specified finite amount of time to experiment with the system. Next your opponent specifies a test input and asks you for your prediction—were this signal to be applied what output would result? The test input is now applied and your prediction put to the test.

Since your opponent is an antagonist you can expect the test input to be totally unlike any input you have previously tried (after all, you don't have time to try *every possible* input). Your opponent may be trying to trick you in many ways. Is it possible to win this game?

This game has two levels of play. In this section we will learn how to play the easy version; in the next section we will make a first attempt at a strategy for the more difficult level. The easy case is when you are given complete control over the black box. You are allowed to apply controlled inputs and observe the resulting output. The difficult case is when you are not allowed to control the box at all. The box is already hooked up and operating. You are only allowed to observe the input and output.

The latter case is not only more difficult, it may not even be possible to pass the prediction test. For instance, you may be unlucky and during the entire time you observe the system the input may be zero. Or the input may contain only a single sinusoid and you are asked to predict the output when the input is a sinusoid of a different frequency. In such cases it is quite unreasonable to expect to be able to completely identify the hidden system. Indeed, this case is so much harder than the first that the term *system identification* is often reserved for it.

However, even the easy case is far from trivial in general. To see this consider a system that is not time-invariant. Your opponent knows that precisely at noon the system will shut down and its output will be zero thereafter. You are given until 11:59 to observe the system and give your

prediction a few seconds before noon. Of course when the system is tested after noon your prediction turns out to be completely wrong! I think you will agree that the game is only fair if we limit ourselves to the identification of time-invariant systems.

Your opponent may still have a trick or two left! The system may have been built to be sensitive to a very specific trigger. For example, for almost every input signal the box may pass the signal unchanged; but for the trigger signal the output will be quite different! A signal that is different from the trigger signal in any way, even only having a slightly different amplitude or having an infinitesimal amount of additive noise, does not trigger the mechanism and is passed unchanged. You toil away trying a large variety of signals and your best prediction is that the system is simply an identity system. Then your opponent supplies the trigger as the test input and the system's output quite astounds you.

The only sensible way to avoid this kind of pitfall is to limit ourselves to linear systems. Linear systems may still be sensitive to specific signals. For example, think of a box that contains the identity system and in parallel a narrow band-pass filter with a strong amplifier. For most signals the output equals the input, but for signals in the band-pass filter's range the output is strongly amplified. However, for linear systems it is not possible to hide the trigger signal. Changing the amplitude or adding some noise will still allow triggering to occur, and once the effect is observed you may home in on it.

So the system identification game is really only fair for linear time-invariant systems, that is, for filters. It doesn't matter to us whether the filters are MA, AR, ARMA, or even without memory; that can be determined from your measurements. Of course since the black box is a real system, it is of necessity realizable as well, and in particular causal. Therefore from now on we will assume that the black box contains an unknown causal filter. If anyone offers to play the game without promising that the box contains a causal filter, don't accept the challenge!

Our task in this section is to develop a winning strategy for the easy case. Let's assume you are given one hour to examine the box in any way you wish (short of prying off the top). At the end of precisely one hour your opponent will reappear, present you with an input signal and ask you what you believe the box's response will be. The most straightforward way of proceeding would be to quickly apply as many different input signals as you can and to record the corresponding outputs. Then you win the game if your opponent's input signal turns out to be essentially one of the inputs you have checked. Unfortunately, there are very many possible inputs, and an hour is to short a time to test even a small fraction of them. To economize

we can exploit the fact that the box contains a linear time-invariant system. If we have already tried input x_n there is no point in trying ax_n or x_{n-m}, but this still leaves a tremendous number of signals to check.

Our job can be made more manageable in two different ways, one of which relies on the time domain description of the input signal, and the other on its frequency domain representation. The frequency domain approach is based on Fourier's theorem that every signal can be written as the weighted sum (or integral) of basic sinusoids. Assume that you apply to the unknown system not every possible signal, but only every possible sinusoid. You store the system's response to each of these and wait for your opponent to appear. When presented with the test input you can simply break it down to its Fourier components, and exploit the filter's linearity to add the stored system responses with the appropriate Fourier coefficients.

Now this task of recording the system outputs is not as hard as it appears, since sinusoids are eigensignals of filters. When a sinusoid is input to a filter the output is a single sinusoid of the same frequency, only the amplitude and phase may be different. So you need only record these amplitudes and phases and use them to predict the system output for the test signal. For example, suppose the test signal turns out to be the sum of three sinusoids

$$x_n = X_1 \sin(\omega_1 n) + X_2 \sin(\omega_2 n) + X_3 \sin(\omega_3 n)$$

the responses of which had been measured to be

$$H_1 \sin(\omega_1 n + \phi_1), \qquad H_2 \sin(\omega_2 n + \phi_2), \qquad \text{and} \qquad H_3 \sin(\omega_3 n + \phi_3)$$

respectively. Then, since the filter is linear, the output is the sum of the three responses, with the Fourier coefficients.

$$y_n = H_1 X_1 \sin(\omega_1 n + \phi_1) + H_2 X_2 \sin(\omega_2 n + \phi_2) + H_3 X_3 \sin(\omega_3 n + \phi_3)$$

More generally, any finite duration or periodic test digital signal can be broken down by the DFT into the sum of a denumerable number of complex exponentials

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{i\frac{2\pi k}{N}n}$$

and the response of the system to each complex exponential is the same complex exponential multiplied by a number H_k.

$$H_k e^{i\frac{2\pi k}{N}n}$$

Using these H_k we can predict the response to the test signal.

$$y_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k H_k e^{\mathrm{i}\frac{2\pi k}{N}n}$$

The H_k are in general complex (representing the gains and phase shifts) and are precisely the elements of the frequency response. A similar decomposition solves the problem for nonperiodic analog signals, only now we have to test a nondenumerable set of sinusoids.

The above discussion proves that the frequency response provides a complete description of a filter. Given the entire frequency response (i.e., the response of the system to all sinusoids), we can always win the game of predicting the response for an arbitrary input.

The frequency response is obviously a frequency domain quantity; the duality of time and frequency domains leads us to believe that there should be a complete description in the time domain as well. There is, and we previously called it the *impulse response*. To measure it we excite the system with a unit impulse (a Dirac delta function $\delta(t)$ for analog systems or a unit impulse signal $\delta_{n,0}$ for digital systems) and measure the output as a function of time (see equation 6.22). For systems without memory there will only be output for time $t = 0$, but in general the output will be nonzero over an entire time interval. A causal system will have its impulse response zero for times $t < 0$ but nonzero for $t \geq 0$. A system that is time-variant (and hence not a filter) requires measuring the response to all the SUIs, a quantity known as the Green's function.

Like the frequency response, the impulse response may be used to predict the output of a filter when an arbitrary input is applied. The strategy is similar to that we developed above, only this time we break down the test signal in the basis of SUIs (equation (2.26)) rather than using the Fourier expansion. We need only record the system's response to each SUI, expand the input signal in SUIs, and exploit the linearity of the system (as we have already done in Section 6.5). Unfortunately, the SUIs are not generally eigensignals of filters, and so the system's outputs will not be SUIs, and we need to record the entire output. However, unlike the frequency response where we needed to observe the system's output for an infinite number of basis functions, here we can capitalize on the fact that all SUIs are related by time shifts. Exploiting the time-invariance property of filters we realize that after measuring the response of an unknown system to a single SUI (e.g., the unit impulse at time zero), we may immediately deduce its response to all SUIs! Hence we need only apply a single input and record a single response

in order to be able to predict the output of a filter when an arbitrary input is applied! The set of signals we must test in order to be able to predict the output of the system to an arbitrary input has been reduced to a single signal! This is the strength of the impulse response.

The impulse response may be nonzero only over a finite interval of time but exactly zero for all times outside this interval. In this case we say the system has a *finite impulse response*, or more commonly we simply call it an FIR filter. The MA systems studied in Sections 6.6 and 6.7 are FIR filters. To see this consider the noncausal three-point averaging system of equation (6.33).

$$y_n = \tfrac{1}{4}x_{n-1} + \tfrac{1}{2}x_n + \tfrac{1}{4}x_{n+1}$$

As time advances so does this window of time, always staying centered on the present. What happens when the input is an impulse? At time $n = \pm 1$ we find a $\frac{1}{4}$ multiplying the nonzero signal value at the origin, returning $\frac{1}{4}$; of course, the $n = 0$ has maximum output $\frac{1}{2}$. At any other time the output will be zero simply because the window does not overlap any nonzero input signal values. The same is the case for any finite combination of input signal values. Thus all the systems that have the form of equation (6.13), which we previously called FIR filters, are indeed FIR.

Let's explicitly calculate the impulse response for the most general causal moving average filter. Starting from equation (6.30) (but momentarily renaming the coefficients) and using the unit impulse as input yields

$$\begin{aligned} y_n &= \sum_{l=0}^{L} g_l \delta_{n-L+l,0} \\ &= g_0\delta_{n-L,0} + g_1\delta_{n-L+1,0} + g_2\delta_{n-L+2,0} + \cdots + g_{L-1}\delta_{n-1,0} + g_L\delta_{n,0} \end{aligned}$$

which is nonzero only when $n = 0$ or $n = 1$ or ... or $n = L$. Furthermore, when $n = 0$ the output is precisely $h_0 = g_L$, when $n = 1$ the output is precisely $h_1 = g_{L-1}$, etc., until $h_L = g_0$. Thus the impulse response of a general MA filter consists exactly of the coefficients that appear in the moving average sum, but in reverse order!

The impulse response is such an important attribute of a filter that it is conventional to reverse the definition of the moving average, and define the FIR filter via the *convolution* in which the indices run in opposite directions, as we did in equation (6.13).

It is evident that were we to calculate the impulse response of the nonterminating convolution of equation (6.14) it would consist of the coefficients as well; but in this case the impulse response would never quite become zero.

If we apply a unit impulse to a system and its output never dies down to zero, we say that the system is **I**nfinite **I**mpulse **R**esponse (IIR). Systems of the form (6.15), which we previously called IIR filters, can indeed sustain an impulse response that is nonzero for an infinite amount of time. To see this consider the simple case

$$y_n = x_n + \tfrac{1}{2} y_{n-1}$$

which is of the type of equation (6.15). For negative times n the output is zero, $y_n = 0$, but at time zero $y_0 = 1$, at time one $y_1 = \frac{1}{2}$ and thereafter y_n is halved every time. It is obvious that the output at time n is precisely $y_n = 2^{-n}$, which for large n is extremely small, but never zero.

Suppose we have been handed a black box and measure its impulse response. Although there may be many systems with this response to the unit impulse, there will be only one filter that matches, and the coefficients of equation (6.14) are precisely the impulse response in reverse order. This means that if we know that the box contains a filter, then measuring the impulse response is sufficient to uniquely define the system. In particular, we needn't measure the frequency response since it is mathematically derivable from the impulse response.

It is instructive to find this connection between the impulse response (the time domain description) and the frequency response (the frequency domain description) of a filter. The frequency response of the nonterminating convolution system

$$y_n = \sum_{i=-\infty}^{\infty} h_i x_{n-i}$$

is found by substituting a sinusoidal input for x_n, and for mathematical convenience we will use a complex sinusoid $x_n = e^{i\omega n}$. We thus obtain

$$\begin{aligned} H(\omega)\, x_n = \quad y_n \quad &= \sum_{k=-\infty}^{\infty} h_k\, e^{i\omega(n-k)} \\ &= \sum_{k=-\infty}^{\infty} h_k\, e^{-i\omega k}\, e^{i\omega n} \\ &= \qquad H_k \qquad\quad x_n \end{aligned} \tag{6.53}$$

where we identified the Fourier transform of the impulse response h_k and the input signal. We have once again shown that when the convolution system has a sinusoidal input its output is the same sinusoid multiplied by a (frequency-dependent) gain. This gain is the frequency response, but

here we have found the FT of the impulse response; hence the frequency response and the impulse response are an FT pair. Just as the time and frequency domain representations of signals are connected by the Fourier transform, the simplest representations of filters in the time and frequency domains are related by the FT.

EXERCISES

6.12.1 Find the impulse response for the following systems.

1. $y_n = x_n$
2. $y_n = x_n + x_{n-2} + x_{n-4}$
3. $y_n = x_n + 2x_{n-1} + 3x_{n-2}$
4. $y_n = \sum_i a_i x_{n-i}$
5. $y_n = x_n + y_{n-1}$
6. $y_n = x_n + \frac{1}{2}(y_{n-1} + y_{n-2})$

6.12.2 An ideal low-pass filter (i.e., one that passes without change signals under some frequency but entirely blocks those above it) is unrealizable. Prove this by arguing that the Fourier transform of a step function is nonzero over the entire axis and then invoking the connection between frequency response and impulse response.

6.12.3 When determining the frequency response we needn't apply each sinusoidal input separately; sinusoid orthogonality and filter linearity allow us to apply multiple sinusoids at the same time. This is what is done in probe signals (cf. exercise 2.6.4). Can we apply all possible sinusoids at the same time and reduce the number of input signals to one?

6.12.4 Since white noise contains all frequencies with the same amplitude, applying white noise to the system is somehow equivalent to applying all possible sinusoids. The *white noise response* is the response of a system to white noise. Prove that for linear systems the spectral amplitude of the white noise response is the amplitude of the frequency response. What about the phase delay portion of the frequency response?

6.12.5 The fact that the impulse and frequency responses are an FT pair derives from the general rule that the FT relates convolution and multiplication $\mathrm{FT}(x * y) = \mathrm{FT}(x)\mathrm{FT}(y)$. Prove this general statement and relate it to the Wiener-Khintchine theorem.

6.12.6 Donald S. Perfectionist tries to measure the frequency response of a system by measuring the output power while injecting a slowly sweeping tone of constant amplitude. Unbeknownst to him the system contains a filter that passes most frequencies unattenuated, and amplifies a small band of frequencies. However, following the filter is a fast **A**utomatic **G**ain **C**ontrol (AGC) that causes all Donald's test outputs to have the same amplitude, thus completely masking the filter. What's wrong?

6.13 System Identification—The Hard Case

Returning to our system identification game, assume that your opponent presents you with a black box that is already connected to an input. We will assume first that the system is known to be an FIR filter of known length $L+1$. If the system is FIR of unknown length we need simply assume some extremely large $L+1$, find the coefficients, and discard all the zero coefficients above the true length.

The above assumption implies that the system's output at time n is

$$y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} + \cdots + a_L x_{n-L}$$

and your job is to determine these coefficients a_l by simultaneously observing the system's input and output. It is clear that this game is riskier than the previous one. You may be very unlucky and during the entire time we observe it the system's input may be identically zero; or you may be very lucky and the input may be a unit impulse and we readily derive the impulse response.

Let's assume that the input signal was zero for some long time (and the output is consequently zero as well) and then suddenly it is turned on. We'll reset our clock to call the time of the first nonzero input time zero (i.e., x_n is identically zero for $n < 0$, but nonzero at $n = 0$). According to the defining equation the first output must be

$$y_0 = a_0 x_0$$

and since we observe both x_0 and y_0 we can easily find

$$a_0 = \frac{y_0}{x_0}$$

which is well defined since by definition $x_0 \neq 0$. Next, observing the input and output at time $n = 1$, we have

$$y_1 = a_0 x_1 + a_1 x_0$$

which can be solved

$$a_1 = \frac{y_1 - a_0 x_1}{x_0}$$

since everything needed is known, and once again $x_0 \neq 0$.

Continuing in this fashion we can express the coefficient a_n at time n in terms of $x_0 \ldots x_n$, $y_0 \ldots y_n$, and $a_0 \ldots a_{n-1}$, all of which are known. To see

this explicitly write the equations

$$\begin{aligned} y_0 &= a_0 x_0 \\ y_1 &= a_0 x_1 + a_1 x_0 \\ y_2 &= a_0 x_2 + a_1 x_1 + a_2 x_0 \\ y_3 &= a_0 x_3 + a_1 x_2 + a_2 x_1 + a_3 x_0 \\ y_4 &= a_0 x_4 + a_1 x_3 + a_2 x_2 + a_3 x_1 + a_4 x_0 \end{aligned} \tag{6.54}$$

and so on, and note that these can be recursively solved

$$\begin{aligned} a_0 &= \frac{y_0}{x_0} \\ a_1 &= \frac{y_1 - a_0 x_1}{x_0} \\ a_2 &= \frac{y_2 - a_0 x_2 - a_1 x_1}{x_0} \\ a_3 &= \frac{y_3 - a_0 x_3 - a_1 x_2 - a_2 x_1}{x_0} \\ a_4 &= \frac{y_4 - a_0 x_4 - a_1 x_3 - a_2 x_2 - a_3 x_1}{x_0} \end{aligned} \tag{6.55}$$

one coefficient at a time.

In order to simplify the arithmetic it is worthwhile to use linear algebra notation. We can write equation (6.54) in matrix form, with the desired coefficients on the right-hand side

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} x_0 & 0 & 0 & 0 & \dots \\ x_1 & x_0 & 0 & 0 & \dots \\ x_2 & x_1 & x_0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \end{pmatrix} \tag{6.56}$$

and identify the matrix containing the input values as being lower triangular and Toeplitz. The solution of (6.55) is simple due to the matrix being lower triangular. Finding the l^{th} coefficient requires l multiplications and subtractions and one division, so that finding all $L+1$ coefficients involves $\frac{1}{2}L(L+1)$ multiplications and subtractions and $L+1$ divisions.

The above solution to the 'hard' system identification problem was based on the assumption that the input signal was exactly zero for $n < 0$. What can we do in the common case when we start observing the signals at an arbitrary time before which the input was not zero? For notational simplicity let's assume that the system is known to be FIR with $L = 2$. Since we

need to find three coefficients we will need three equations, so we observe three outputs, y_n, y_{n+1} and y_{n+2}. Now these outputs depend on five inputs, $x_{n-2}, x_{n-1}, x_n, x_{n+1}$, and x_{n+2} in the following way

$$\begin{aligned} y_n &= a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} \\ y_{n+1} &= a_0 x_{n+1} + a_1 x_n + a_2 x_{n-1} \\ y_{n+2} &= a_0 x_{n+2} + a_1 x_{n+1} + a_2 x_n \end{aligned} \tag{6.57}$$

which in matrix notation can be written

$$\begin{pmatrix} y_n \\ y_{n+1} \\ y_{n+2} \end{pmatrix} = \begin{pmatrix} x_n & x_{n-1} & x_{n-2} \\ x_{n+1} & x_n & x_{n-1} \\ x_{n+2} & x_{n+1} & x_n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} \tag{6.58}$$

or in other words $\underline{y} = \underline{\underline{X}}\,\underline{a}$, where $\underline{\underline{X}}$ is a nonsymmetric Toeplitz matrix. The solution is obviously $\underline{a} = \underline{\underline{X}}^{-1}\underline{y}$ but the three-by-three matrix is not lower triangular, and so its inversion is no longer trivial. For larger number of coefficients L we have to invert an $N = L + 1$ square matrix; although most direct N-by-N matrix inversion algorithms have computational complexity $O(N^3)$, it is possible to invert a general matrix in $O(N^{\log_2 7}) \sim O(N^{2.807})$ time. Exploiting the special characteristics of Toeplitz matrices reduces the computational load to $O(N^2)$.

What about AR filters?

$$y_n = x_n + \sum_{m=1}^{M} b_m y_{n-m}$$

Can we similarly find their coefficients in the hard system identification case? Once again, for notational simplicity we'll take $M = 3$. We have three unknown b coefficients, so we write down three equations,

$$\begin{aligned} y_n &= x_n + b_1 y_{n-1} + b_2 y_{n-2} + b_3 y_{n-3} \\ y_{n+1} &= x_{n+1} + b_1 y_n + b_2 y_{n-1} + b_3 y_{n-2} \\ y_{n+2} &= x_{n+2} + b_1 y_{n+1} + b_2 y_n + b_3 y_{n-1} \end{aligned} \tag{6.59}$$

or in matrix notation

$$\begin{pmatrix} y_n \\ y_{n+1} \\ y_{n+2} \end{pmatrix} = \begin{pmatrix} x_n \\ x_{n+1} \\ x_{n+2} \end{pmatrix} + \begin{pmatrix} y_{n-1} & y_{n-2} & y_{n-3} \\ y_n & y_{n-1} & y_{n-2} \\ y_{n+1} & y_n & y_{n-1} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \tag{6.60}$$

or simply $\underline{y} = \underline{x} + \underline{\underline{Y}}\,\underline{b}$. The answer this time is $\underline{b} = \underline{\underline{Y}}^{-1}(\underline{y} - \underline{x})$, which once again necessitates inverting a nonsymmetric Toeplitz matrix.

Finally, the full ARMA with $L = 2$ and $M = 3$

$$y_n = \sum_{l=0}^{L} a_l x_{n-l} + \sum_{m=1}^{M} b_m y_{n-m}$$

has six unknowns, and so we need to take six equations.

$$\begin{aligned}
y_n &= a_0x_n + a_1x_{n-1} + a_2x_{n-2} + b_1y_{n-1} + b_2y_{n-2} + b_3y_{n-3} \\
y_{n+1} &= a_0x_{n+1} + a_1x_n + a_2x_{n-1} + b_1y_n + b_2y_{n-1} + b_3y_{n-2} \\
y_{n+2} &= a_0x_{n+2} + a_1x_{n+1} + a_2x_n + b_1y_{n+1} + b_2y_n + b_3y_{n-1} \\
y_{n+3} &= a_0x_{n+3} + a_1x_{n+2} + a_2x_{n+1} + b_1y_{n+2} + b_2y_{n+1} + b_3y_n \\
y_{n+4} &= a_0x_{n+4} + a_1x_{n+3} + a_2x_{n+2} + b_1y_{n+3} + b_2y_{n+2} + b_3y_{n+1} \\
y_{n+5} &= a_0x_{n+5} + a_1x_{n+4} + a_2x_{n+3} + b_1y_{n+4} + b_2y_{n+3} + b_3y_{n+2}
\end{aligned}$$

This can be written compactly

$$\begin{pmatrix} y_n \\ y_{n+1} \\ y_{n+2} \\ y_{n+3} \\ y_{n+4} \\ y_{n+5} \end{pmatrix} = \begin{pmatrix} x_n & x_{n-1} & x_{n-2} & y_{n-1} & y_{n-2} & y_{n-3} \\ x_{n+1} & x_n & x_{n-1} & y_n & y_{n-1} & y_{n-2} \\ x_{n+2} & x_{n+1} & x_n & y_{n+1} & y_n & y_{n-1} \\ x_{n+3} & x_{n+2} & x_{n+1} & y_{n+2} & y_{n+1} & y_n \\ x_{n+4} & x_{n+3} & x_{n+2} & y_{n+3} & y_{n+2} & y_{n+1} \\ x_{n+5} & x_{n+4} & x_{n+3} & y_{n+4} & y_{n+3} & y_{n+2} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} \tag{6.61}$$

and the solution requires inverting a six-by-six nonsymmetric non-Toeplitz matrix. The ARMA case is thus more computationally demanding than the pure MA or AR cases.

Up to now we have assumed that we observe x_n and y_n with no noise whatsoever. In all practical cases there will be at least some quantization noise, and most of the time there will be many other sources of additive noise. Due to this noise we will not get precisely the same answers when solving equations (6.58), (6.60), or (6.61) for two different times. One rather obvious tactic is to solve the equations many times and average the resulting coefficients. However, the matrix inversion would have to be performed a very large number of times and the equations (especially (6.60) and (6.61)) often turn out to be rather sensitive to noise. A much more successful tactic is to average *before* solving the equations, which has the advantages of providing more stable equations and requiring only a single matrix inversion.

Let's demonstrate how this is carried out for the MA case.

$$y_n = \sum_{k=0}^{L} a_k x_{n-k} \tag{6.62}$$

In order to average we multiply both sides by x_{n-q} and sum over as many n as we can get our hands on.

$$\sum_n y_n x_{n-q} = \sum_{k=0}^{L} a_k \sum_n x_{n-k} x_{n-q}$$

We define the x autocorrelation and the x-y crosscorrelation (see Chapter 9)

$$C_x(k) = \sum_n x_n x_{n-k} \qquad C_{yx}(k) = \sum_n y_n x_{n-k}$$

and note the following obvious symmetry.

$$C_x(-k) = C_x(k)$$

The deconvolution equations can now be written simply as

$$C_{yx}(q) = \sum_k a_k C_x(q-k) \tag{6.63}$$

and are called the Wiener-Hopf equations. For $L = 2$ the Wiener-Hopf equations look like this:

$$\begin{pmatrix} C_{yx}(0) \\ C_{yx}(1) \\ C_{yx}(2) \end{pmatrix} = \begin{pmatrix} C_x(0) & C_x(-1) & C_x(-2) \\ C_x(1) & C_x(0) & C_x(-1) \\ C_x(2) & C_x(1) & C_x(0) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix}$$

and from the aforementioned symmetry we immediately recognize the matrix as *symmetric* Toeplitz, a fact that makes them more stable and even faster to solve.

For a black box containing an AR filter, there is a special case where the input signal dies out (or perhaps the input happens to be an impulse). Once the input is zero

$$y_n = \sum_{m=1}^{M} b_m y_{n-m}$$

multiplying by y_{n-q} and summing over n we find

$$\sum_n y_n y_{n-q} = \sum_{m=1}^{M} b_m \sum_n y_{n-m} y_{n-q}$$

in which we identify y autocorrelations.

$$\sum_{m=1}^{M} C_y(|m-q|)b_m = C_y(q) \tag{6.64}$$

For $M = 3$ these equations look like this.

$$\begin{pmatrix} C_y(0) & C_y(1) & C_y(2) \\ C_y(1) & C_y(0) & C_y(1) \\ C_y(2) & C_y(1) & C_y(0) \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} C_y(1) \\ C_y(2) \\ C_y(3) \end{pmatrix}$$

These are the celebrated Yule-Walker equations, which will turn up again in Sections 9.8 and 9.9.

EXERCISES

6.13.1 Write a program that numerically solves equation (6.55) for the coefficients of a causal MA filter given arbitrary inputs and outputs. Pick such a filter and generate outputs for a pseudorandom input. Run your program for several different input sequences and compare the predicted coefficients with the true ones (e.g., calculate the squared difference). What happens if you try predicting with too long a filter? Too short a filter? If the input is a sinusoid instead of pseudorandom?

6.13.2 Repeat the previous exercise for AR filters (i.e., solve equation (6.60)). If the filter seems to be seriously wrong, try exciting it with a new pseudorandom input and comparing its output with the output of the intended system.

6.13.3 In the text we assumed that we knew the order L and M. How can we find the order of the system being identified?

6.13.4 Assume that y_n is related to x_n by a noncausal MA filter with coefficients $a_{-M} \ldots a_M$. Derive equations for the coefficients in terms of the appropriate number of inputs and outputs.

6.13.5 In deriving the Wiener-Hopf equations we could have multiplied by y_{n-q} to get the equations

$$C_y(q) = \sum_k h_k C_{xy}(q-k)$$

rather than multiplying by x_{n-q}. Why didn't we?

6.13.6 In the derivation of the Wiener-Hopf equations we assumed that C_x and C_{yx} depend on k but not n. What assumption were we making about the noisy signals?

6.13.7 In an even harder system identification problem not only don't you have control over the system's input, you can't even observe it. Can the system be identified based on observation of the output only?

6.13.8 Assume that the observed input sequence x_n is white, i.e., that all autocorrelations are zero except for $C_x(0)$. What is the relationship between the crosscorrelation $C_{yx}(n)$ and the system's impulse response? How does this simplify the hard system identification task?

6.14 System Identification in the z Domain

In the previous section we solved the hard system identification problem in the time domain. The solution involved solving sets of linear equations, although for many cases of interest these equations turn out to be relatively simple. Is there a method of solving the hard system identification problem without the need for solving equations? For the easy problem we could inject an impulse as input, and simply measure the impulse response. For the hard problem we extended this technique by considering the input to be the sum of SUIs $x_n = \sum x_m \delta n - m$ and exploiting linearity. Realizing that each output value consists of intertwined contributions from many SUI inputs, we are forced to solve linear equations to isolate these individual contributions. There is no other way to disentangle the various contributions since although the SUI basis functions from which the input can be considered to be composed are orthogonal and thus easily separable by projection without solving equations, the time-shifted impulse responses are not.

This gives us an idea; we know that sinusoids *are* eigenfunctions of filters, and that they are mutually orthogonal. Hence the input at ω can be derived from the output at that same frequency, with no other frequencies interfering. We can thus recover the frequency response by converting the input and output signals into the frequency domain and merely dividing the output at every frequency by the input at that same frequency.

$$Y(\omega) = H(\omega)X(\omega) \qquad \longrightarrow \qquad H(\omega) = \frac{Y(\omega)}{X(\omega)}$$

What could be easier? If we wish we can even recover the impulse response from the frequency response, by using equation (6.53). So it would seem best to solve the easy system identification problem in the time domain and the hard problem in the frequency domain.

One must be careful when using this frequency domain approach to the hard system identification problem. To see why, think of an MA all-pass system that simply delays the input by L samples $y_n = x_{n-L}$. Had we observed the system when the input was of the form $x_n = \sin(2\pi\frac{kn}{L} + \phi)$ for any k we would conclude that we were observing the identity system $y_n = x_n$! This mistake is obviously due to the input being periodic and the system looking back in time by precisely an integer number of periods; equivalently in the frequency domain this input consists of a single line, and thus we can only learn about $H(\omega)$ at this single frequency. The lesson to be learned is more general than this simple example. In order to uniquely specify a system the input must excite it at all frequencies. There are often frequencies for which the system produces no output at all, and based on these we certainly would not be able to identify the system. The unit impulse is a single excitation that squeezes all possible information out of the system; due to orthogonality and the eigenfunction property a single sinusoid contributes only an infinitesimal amount of information about the system.

The frequency response is a great tool for FIR systems, but not as good for IIR systems since they may become unstable. When an IIR system's output increases without limit for a frequency ω, this is a sign that its frequency response is infinite there. For example, a typical frequency response is depicted in Figure 6.12. As usual, the horizontal axis is from DC to half the sampling rate. We see that the frequency response goes to zero for a digital frequency of 0.2. This means that when the input is a sinusoid of

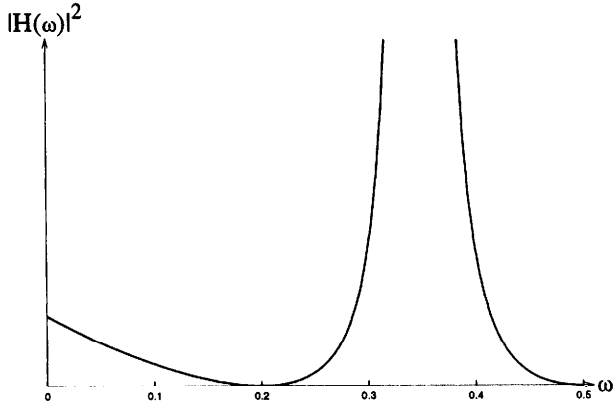


Figure 6.12: The frequency response of an ARMA filter with a zero at frequency $0.2f_s$ and a pole at $0.35f_s$.

this frequency there will be no output. We call such frequencies *zeros* of the frequency response. Around digital frequency 0.35 something different occurs. Input signals in this region are strongly amplified, and at 0.35 itself the output grows without limit. We recall from Section (4.11) that the Fourier transform is *not* the proper tool to describe this type of behavior; the z transform *is*.

So let's see how the zT can be used in system identification. We know that the effect of any filter can be expressed as convolution by the impulse response $y = h * x$, although for non-FIR systems this convolution is in principle infinite. In view of the connection between convolution and multiplication (see Section 6.8) we would like somehow to write $h = y/x$, a process known as *deconvolution*. The zT is the tool that transforms convolutions into algebraic multiplications, so we apply it now.

Similarly to the result for FT and DFT, convolution in the time domain becomes multiplication in the z domain

$$Y(z) = H(z)\,X(z) \tag{6.65}$$

and this is simple to prove.

$$\begin{aligned} Y(z) &= \sum_{n=-\infty}^{\infty} y_n z^{-n} \\ &= \sum_{n=-\infty}^{\infty} \left\{ \sum_{k=-\infty}^{\infty} h_k x_{n-k} \right\} z^{-n} \\ &= \sum_{k=-\infty}^{\infty} h_k \sum_{n=-\infty}^{\infty} x_{n-k} z^{-n} \\ &= \sum_{k=-\infty}^{\infty} h_k z^{-k} \sum_{m=-\infty}^{\infty} x_m z^{-m} \\ &= \quad H(z) \qquad\quad X(z) \end{aligned}$$

The z transform of the impulse response h is called the *transfer function*, and this name implies that we can think of the operation of the system as transferring $X(z)$ into $Y(z)$ by a simple multiplication. Of course, thinking of $z = re^{i\omega}$, the transfer function is seen to be a generalization of the frequency response. Evaluated on the unit circle $r = 1$ (i.e., $z = e^{i\omega}$) the transfer function is precisely the frequency response, while for other radii we obtain the response of the system to decaying ($r < 1$) or growing ($r > 1$) sinusoids as in equation (2.12).

So the complete solution of the hard system identification problem is easy. The transfer function of the system in question is

$$H(z) = \frac{Y(z)}{X(z)}$$

with frequency response obtainable by evaluating $H(z)$ on the unit circle $z = e^{i\omega}$, and impulse response derivable from the frequency response.

Let's use this method to identify a nontrivial system. Assume that the unknown system is equation (6.39) with $\beta = \frac{1}{2}$

$$y_n = \tfrac{1}{2}(y_{n-1} + x_n) \tag{6.66}$$

and that the input is observed to turn on at $n = 0$ and is DC from then on $x_n = u_n$. Observing the input and output you compose the following table:

n	0	1	2	$\cdots$	n	$\cdots$
x_n	1	1	1	$\cdots$	1	$\cdots$
y_n	$\frac{1}{2}$	$\frac{3}{4}$	$\frac{7}{8}$	$\cdots$	$\frac{2^{n+1}-1}{2^{n+1}}$	$\cdots$

Using the time domain method of equation (6.55) after some work you could deduce that the coefficients of the nonterminating convolution are $h_n = \frac{1}{2}^{-(n+1)}$, but some inspiration would still be required before the ARMA form could be discovered. So let's try the zT line of attack. The zTs of the input and output are easily found by using equation (4.63).

$$\begin{aligned} X(z) = \quad & \mathrm{zT}\, u_n \quad && = \frac{1}{1 - \mathrm{z}^{-1}} && \text{ROC } |z| > 1 \\ Y(z) = \quad & \mathrm{zT}\, \tfrac{2^{n+1}-1}{2^{n+1}} u_n \quad && = \frac{1}{1 - \mathrm{z}^{-1}} - \tfrac{1}{2} \frac{1}{1 - \frac{1}{2}\mathrm{z}^{-1}} \\ & && = \frac{\frac{1}{2}}{(1 - \mathrm{z}^{-1})(1 - \frac{1}{2}\mathrm{z}^{-1})} && \text{ROC } |z| > \tfrac{1}{2} \end{aligned}$$

Now the transfer function is the ratio

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\frac{1}{2}}{1 - \frac{1}{2}\mathrm{z}^{-1}}$$

so that the difference equation $Y(z) = H(z)X(z)$ is

$$\left(1 - \tfrac{1}{2}\mathrm{z}^{-1}\right)\, Y(z) \;=\; \tfrac{1}{2} X(z)$$

which from the meaning of z^{-1} in the time domain is simply

$$y_n - \tfrac{1}{2}y_{n-1} = \tfrac{1}{2}x_n$$

and equation (6.66) has magically appeared!

EXERCISES

6.14.1 As we shall learn in Chapter 18, modem signals are distorted by the telephone lines through which they travel. This distortion can be modeled as a filter, and removed by appropriate inverse filtering (equalizing). Can you explain why modems transmit known pseudonoise signals during their start-up procedures?

6.14.2 You observe a system when its input consists of the sum of two different sinusoids. Find two systems that cannot be distinguished based on this input. Do the same for an input composed of M sinusoids.

6.14.3 What is the transfer function of two systems connected in series (cascaded so that $y = H_2 w, w = H_1 x$)? Of two systems connected in parallel (i.e., so that $y = H_1 x + H_2 y$)?

6.14.4 Prove that ARMA systems commute.

6.14.5 Deconvolution is equivalent to finding the inverse system for a filter. Explain how to carry out deconvolution using the transfer function.

6.14.6 Prove (as in exercise 6.4.1) that the inverse system of an MA filter is AR and vice versa.

6.14.7 Many communication channels both distort the information carrying signal by an unknown filter and add nonwhite noise to it. The frequency characteristics of both the channel filter and the noise can be directly measured by inputting a signal consisting of a comb of equidistant sinusoids each with known amplitude and phase, and measuring the output at these frequencies. The input signal is conveniently generated and the output recovered using the DFT. In order to combat noise the procedure should be repeated N times and the output averaged. We denote the input in bin k by X_k and the measured output at repetition m by $Y_k^{[m]}$. Explain how to measure the SNR of bin k.

6.14.8 Continuing the previous exercise, a **F**requency **EQ**ualizer (FEQ) tries to remove the frequency distortion introduced by the channel filter by directly multiplying each output Y_k by complex number e_k in order to recover the input $X_k = e_k Y_k$. Explain how to find the FEQ coefficients e_k.

6.14.9 Continuing the previous exercises, the frequency magnitude response $|H_k|^2$ (the ratio of the output to input energies) as measured at repetition m is $|H_k^{[m]}|^2 = \frac{(Y_k^{[m]})^2}{X_k^2}$. Express $|H_k|^2$ in terms of e_k and SNR_k.

Bibliographical Notes

Signal processing systems are treated in all the standard signal processing texts, [186, 185, 187, 200, 189, 252, 159, 167], as well as books specifically on system design [126].

The word convolution was used as early as 1935 by mathematicians, but seems to have been picked up by the signal processing community rather later. Norbert Wiener, in his classic 1933 text [277], uses the German *Faltung* noting the lack of an appropriate English-language word. In his later book of the 1940s [278] there is a conspicuous absence of the word. Rice, in an influential 1944–45 article on noise [220] gives the FT of a product in an appendix, calling the convolution simply 'the integral on the right'. In 1958 Blackman and Tukey [19] use the word convolution freely, although they mention several other possible names as well.

The impulse response, known in other fields as the Green's function, was first published by Green in 1828 [86].

Amazingly, the Wiener-Hopf equations were originally derived in the early 1930s to solve a problem involving radiation equilibrium in stars [281]. While working on defense-related problems during World War II, Wiener discovered that these same equations were useful for prediction and filtering. Several years before Eberhard Hopf had returned to Nazi Germany in order to accept a professorship at Leipzig that had been vacated by a cousin of Wiener's who had fled Germany after the rise of Hitler ([280]). Despite this turn of events Wiener always referred to the 'Hopf-Wiener' equations.

The great Cambridge statistician George Udny Yule formulated the Yule-Walker equations for signals containing one or two sinusoidal components in the late 1920s, in an attempt to explain the periodicity of sunspot numbers [289]. A few years later Sir Gilbert Walker expanded on this work [267], discovering that the autocorrelations were much smoother than the noisy signal itself, and applying this technique to a meteorological problem.

Otto Toeplitz was one of the founders of operator theory, as well as a great teacher and historian of math [21, 14]. In operator theory he was one of Hilbert's principle students, emphasizing matrix methods and considering Banach's methods too abstract. In teaching he was a disciple of Felix Klein (who considered group theory too abstract). In Bonn he would lecture to packed audiences of over 200 students, and was said to recognize each student's handwriting and writing style. He indirectly influenced the development of Quantum Mechanics by teaching his friend Max Born matrix methods; Born later recognized that these were the mathematical basis of Heisenberg's theory. Toeplitz was dismissed from his university position on racial grounds after the Nürnberg laws of 1935, but stayed on in Germany until 1939 representing the Jewish community and helping minority students emigrate. He finally left Germany in 1939, traveling to Jerusalem where he assumed the post of scientific advisor to the Hebrew University, a position he held less than a year until his death in 1940.