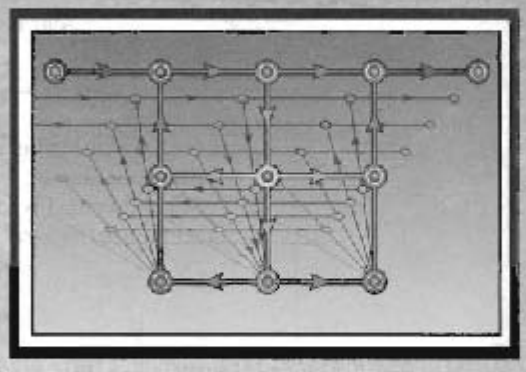
6 Structures for Discrete-Time Systems



6.0 INTRODUCTION

As we saw in Chapter 5, an LTI system with a rational system function has the property that the input and output sequences satisfy a linear constant-coefficient difference equation. Since the system function is the z-transform of the impulse response, and since the difference equation satisfied by the input and output can be determined by inspection of the system function, it follows that the difference equation, the impulse response, and the system function are equivalent characterizations of the input–output relation of an LTI discrete-time system. When such systems are implemented with discrete-time analog or digital hardware, the difference equation or the system function representation must be converted to an algorithm or structure that can be realized in the desired technology. As we will see in this chapter, systems described by linear constant-coefficient difference equations can be represented by structures consisting of an interconnection of the basic operations of addition, multiplication by a constant, and delay, the exact implementation of which is dictated by the technology to be used.

As an illustration of the computation associated with a difference equation, consider the system described by the system function

$$H(z) = \frac{b_0 + b_1 z^{-1}}{1 - az^{-1}}, \qquad |z| > |a|. \tag{6.1}$$

The impulse response of this system is

$$h[n] = b_0 a^n u[n] + b_1 a^{n-1} u[n \quad 1], \tag{6.2}$$

and the 1st-order difference equation that is satisfied by the input and output sequences is

$$y[n] - ay[n - 1] = b_0 x[n] + b_1 x[n - 1]. \tag{6.3}$$

Equation (6.2) gives a formula for the impulse response for this system. However, since the system impulse response has infinite duration, even if we only wanted to compute the output over a finite interval, it would not be efficient to do so by discrete convolution since the amount of computation required to compute $y[n]$ would grow with n. However, rewriting Eq. (6.3) in the form

$$y[n] = ay[n-1] + b_0x[n] + b_1x[n-1] \tag{6.4}$$

provides the basis for an algorithm for recursive computation of the output at any time n in terms of the previous output $y[n-1]$, the current input sample $x[n]$, and the previous input sample $x[n-1]$. As discussed in Section 2.5, if we further assume initial-rest conditions (i.e., if $x[n] = 0$ for $n < 0$, then $y[n] = 0$ for $n < 0$), and if we use Eq. (6.4) as a recurrence formula for computing the output from past values of the output and present and past values of the input, the system will be linear and time invariant. A similar procedure can be applied to the more general case of an N^{th}-order difference equation. However, the algorithm suggested by Eq. (6.4), and its generalization for higher-order difference equations is not the only computational algorithm for implementing a particular system, and often, it is not the best choice. As we will see, an unlimited variety of computational structures result in the same relation between the input sequence $x[n]$ and the output sequence $y[n]$.

In the remainder of this chapter, we consider the important issues in the implementation of LTI discrete-time systems. We first present the block diagram and signal flow graph descriptions of computational structures for linear constant-coefficient difference equations representing LTI causal systems.[1] Using a combination of algebraic manipulations and manipulations of block diagram representations, we derive a number of basic equivalent structures for implementing a causal LTI system including lattice structures. Although two structures may be equivalent with regard to their input–output characteristics for infinite-precision representations of coefficients and variables, they may have vastly different behavior when the numerical precision is limited. This is the major reason that it is of interest to study different implementation structures. The effects of finite-precision representation of the system coefficients and the effects of truncation or rounding of intermediate computations are considered in the latter sections of the chapter.

6.1 BLOCK DIAGRAM REPRESENTATION OF LINEAR CONSTANT-COEFFICIENT DIFFERENCE EQUATIONS

The implementation of an LTI discrete-time system by iteratively evaluating a recurrence formula obtained from a difference equation requires that delayed values of the output, input, and intermediate sequences be available. The delay of sequence values implies the need for storage of past sequence values. Also, we must provide means for multiplication of the delayed sequence values by the coefficients, as well as means for adding the resulting products. Therefore, the basic elements required for the implementation of an LTI discrete-time system are adders, multipliers, and memory for storing

[1] Such flow graphs are also called "networks" in analogy to electrical circuit diagrams. We shall use the terms flow graph, structure, and network interchangeably with respect to graphic representations of difference equations.

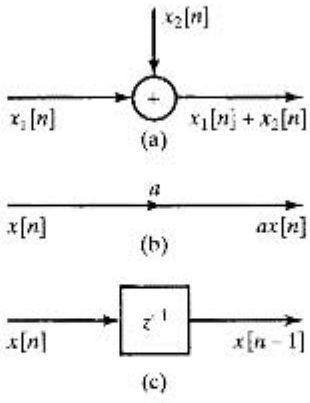


Figure 6.1 Block diagram symbols. (a) Addition of two sequences. (b) Multiplication of a sequence by a constant. (c) Unit delay.

delayed sequence values and coefficients. The interconnection of these basic elements is conveniently depicted by block diagrams composed of the basic pictorial symbols shown in Figure 6.1. Figure 6.1(a) represents the addition of two sequences. In general block diagram notation, an adder may have any number of inputs. However, in almost all practical implementations, adders have only two inputs. In all the diagrams of this chapter, we indicate this explicitly by limiting the number of inputs as in Figure 6.1(a). Figure 6.1(b) depicts multiplication of a sequence by a constant, and Figure 6.1(c) depicts delaying a sequence by one sample. In digital implementations, the delay operation can be implemented by providing a storage register for each unit delay that is required. For this reason, we sometimes refer to the operator of Figure 6.1(c) as a *delay register*. In analog discrete-time implementations such as switched-capacitor filters, the delays are implemented by charge storage devices. The unit delay system is represented in Figure 6.1(c) by its system function, z^{-1}. Delays of more than one sample can be denoted as in Figure 6.1(c), with a system function of z^{-M}, where M is the number of samples of delay; however, the actual implementation of M samples of delay would generally be done by cascading M unit delays. In an integrated-circuit implementation, these unit delays might form a shift register that is clocked at the sampling rate of the input signal. In a software implementation, M cascaded unit delays would be implemented as M consecutive memory registers.

Example 6.1 Block Diagram Representation of a Difference Equation

As an example of the representation of a difference equation in terms of the elements in Figure 6.1, consider the 2nd-order difference equation

$$y[n] = a_1 y[n-1] + a_2 y[n-2] + b_0 x[n]. \tag{6.5}$$

The corresponding system function is

$$H(z) = \frac{b_0}{1 - a_1 z^{-1} - a_2 z^{-2}}. \tag{6.6}$$

The block diagram representation of the system realization based on Eq. (6.5) is shown in Figure 6.2. Such diagrams give a pictorial representation of a computational algorithm for implementing the system. When the system is implemented on either a

general-purpose computer or a digital signal processing (DSP) chip, network structures such as the one shown in Figure 6.2 serve as the basis for a program that implements the system. If the system is implemented with discrete components or as a complete system with very large-scale integration (VLSI) technology, the block diagram is the basis for determining a hardware architecture for the system. In both cases, diagrams such as Figure 6.2 show explicitly that we must provide storage for the delayed variables (in this case, $y[n-1]$ and $y[n-2]$) and also the coefficients of the difference equation (in this case, a_1, a_2, and b_0). Furthermore, we see from Figure 6.2 that an output sequence value $y[n]$ is computed by first forming the products $a_1y[n-1]$ and $a_2y[n-2]$, then adding them, and, finally, adding the result to $b_0x[n]$. Thus, Figure 6.2 conveniently depicts the complexity of the associated computational algorithm, the steps of the algorithm, and the amount of hardware required to realize the system.

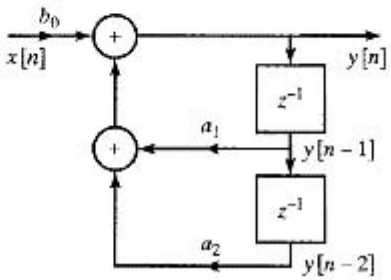


Figure 6.2 Example of a block diagram representation of a difference equation.

Example 6.1 can be generalized to higher-order difference equations of the form[2]

$$y[n] - \sum_{k=1}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k], \tag{6.7}$$

with the corresponding system function

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}. \tag{6.8}$$

Rewriting Eq. (6.7) as a recurrence formula for $y[n]$ in terms of a linear combination of past values of the output sequence and current and past values of the input sequence

[2]The form used in previous chapters for a general N^{th}-order difference equation was

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k].$$

In the remainder of the book, it will be more convenient to use the form in Eq. (6.7), where the coefficient of $y[n]$ is normalized to unity and the coefficients associated with the delayed output appear with a positive sign after they have been moved to the right-hand side of the equation. (See Eq. (6.9).)

leads to the relation

$$y[n] = \sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k]. \tag{6.9}$$

The block diagram of Figure 6.3 is an explicit pictorial representation of Eq. (6.9). More precisely, it represents the pair of difference equations

$$v[n] = \sum_{k=0}^{M} b_k x[n-k], \tag{6.10a}$$

$$y[n] = \sum_{k=1}^{N} a_k y[n-k] + v[n]. \tag{6.10b}$$

The assumption of a two-input adder implies that the additions are done in a specified order. That is, Figure 6.3 shows that the products $a_N y[n-N]$ and $a_{N-1}y[n-N+1]$ must be computed, then added, and the resulting sum added to $a_{N-2}y[n-N+2]$, and so on. After $y[n]$ has been computed, the delay variables must be updated by moving $y[n-N+1]$ into the register holding $y[n-N]$, and so on, with the newly computed $y[n]$ becoming $y[n-1]$ for the next iteration.

A block diagram can be rearranged or modified in a variety of ways without changing the overall system function. Each appropriate rearrangement represents a *different* computational algorithm for implementing the *same* system. For example, the block diagram of Figure 6.3 can be viewed as a cascade of two systems, the first representing the computation of $v[n]$ from $x[n]$ and the second representing the computation of $y[n]$ from $v[n]$. Since each of the two systems is an LTI system (assuming initial-rest conditions for the delay registers), the order in which the two systems are cascaded can be reversed, as shown in Figure 6.4, without affecting the overall system function. In Figure 6.4, for convenience, we have assumed that $M = N$. Clearly, there is no loss of generality, since if $M \neq N$, some of the coefficients a_k or b_k in the figure would be zero, and the diagram could be simplified accordingly.

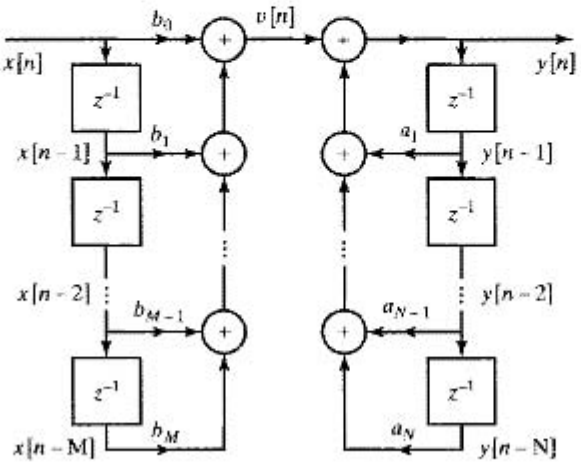


Figure 6.3 Block diagram representation for a general N^{th}-order difference equation.

In terms of the system function $H(z)$ in Eq. (6.8), Figure 6.3 can be viewed as an implementation of $H(z)$ through the decomposition

$$H(z) = H_2(z)H_1(z) = \left(\frac{1}{1 - \sum_{k=1}^{N} a_k z^{-k}}\right)\left(\sum_{k=0}^{M} b_k z^{-k}\right) \tag{6.11}$$

or, equivalently, through the pair of equations

$$V(z) = H_1(z)X(z) = \left(\sum_{k=0}^{M} b_k z^{-k}\right) X(z), \tag{6.12a}$$

$$Y(z) = H_2(z)V(z) = \left(\frac{1}{1 - \sum_{k=1}^{N} a_k z^{-k}}\right) V(z). \tag{6.12b}$$

Figure 6.4, on the other hand, represents $H(z)$ as

$$H(z) = H_1(z)H_2(z) = \left(\sum_{k=0}^{M} b_k z^{-k}\right)\left(\frac{1}{1 - \sum_{k=1}^{N} a_k z^{-k}}\right) \tag{6.13}$$

or, equivalently, through the equations

$$W(z) = H_2(z)X(z) = \left(\frac{1}{1 - \sum_{k=1}^{N} a_k z^{-k}}\right) X(z), \tag{6.14a}$$

$$Y(z) = H_1(z)W(z) = \left(\sum_{k=0}^{M} b_k z^{-k}\right) W(z). \tag{6.14b}$$

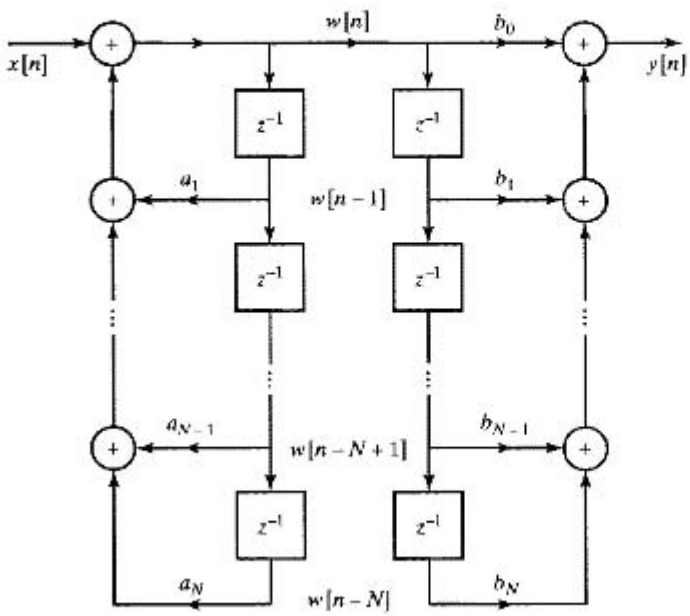


Figure 6.4 Rearrangement of block diagram of Figure 6.3. We assume for convenience that $N = M$. If $N \neq M$, some of the coefficients will be zero.

In the time domain, Figure 6.4 and, equivalently, Eqs. (6.14a) and (6.14b) can be represented by the pair of difference equations

$$w[n] = \sum_{k=1}^{N} a_k w[n-k] + x[n], \tag{6.15a}$$

$$y[n] = \sum_{k=0}^{M} b_k w[n-k]. \tag{6.15b}$$

The block diagrams of Figures 6.3 and 6.4 have several important differences. In Figure 6.3, the zeros of $H(z)$, represented by $H_1(z)$, are implemented first, followed by the poles, represented by $H_2(z)$. In Figure 6.4, the poles are implemented first, followed by the zeros. Theoretically, the order of implementation does not affect the overall system function. However, as we will see, when a difference equation is implemented with finite-precision arithmetic, there can be a significant difference between two systems that are equivalent with the assumption of infinite precision arithmetic in the real number system. Another important point concerns the number of delay elements in the two systems. As drawn, the systems in Figures 6.3 and 6.4 each have a total of $(N + M)$ delay elements. However, the block diagram of Figure 6.4 can be redrawn by noting that exactly the same signal, $w[n]$, is stored in the two chains of delay elements in the figure. Consequently, the two can be collapsed into one chain, as indicated in Figure 6.5.

The total number of delay elements in Figure 6.5 is less than or equal to the number required in either Figure 6.3 or Figure 6.4, and in fact it is the minimum number required to implement a system with system function given by Eq. (6.8). Specifically, the minimum number of delay elements required is, in general, $\max(N, M)$. An implementation with the minimum number of delay elements is commonly referred to as a *canonic form*

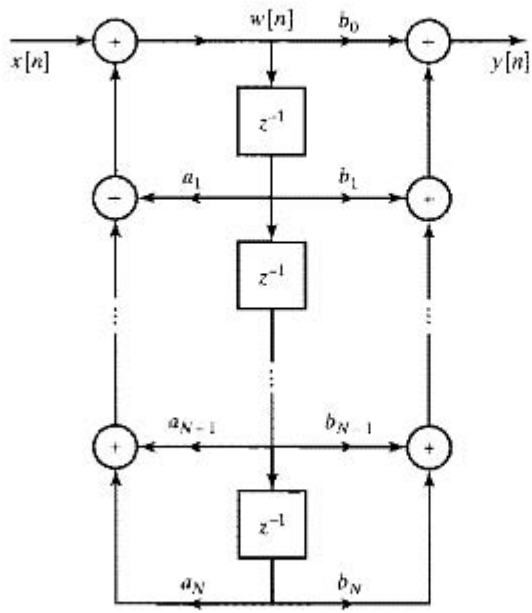


Figure 6.5 Combination of delays in Figure 6.4.

implementation. The noncanonic block diagram in Figure 6.3 is referred to as the *direct form I* implementation of the general Nth-order system because it is a direct realization of the difference equation satisfied by the input $x[n]$ and the output $y[n]$, which in turn can be written directly from the system function by inspection. Figure 6.5 is often referred to as the *direct form II* or *canonic direct form* implementation. Knowing that Figure 6.5 is an appropriate realization structure for $H(z)$ given by Eq. (6.8), we can go directly back and forth in a straightforward manner between the system function and the block diagram (or the equivalent difference equation).

Example 6.2 Direct Form I and Direct Form II Implementation of an LTI System

Consider the LTI system with system function

$$H(z) = \frac{1 + 2z^{-1}}{1 - 1.5z^{-1} + 0.9z^{-2}}. \tag{6.16}$$

Comparing this system function with Eq. (6.8), we find $b_0 = 1$, $b_1 = 2$, $a_1 = +1.5$, and $a_2 = -0.9$, so it follows from Figure 6.3 that we can implement the system in a direct form I block diagram as shown in Figure 6.6. Referring to Figure 6.5, we can also implement the system function in direct form II, as shown in Figure 6.7. In both cases, note that the coefficients in the feedback branches in the block diagram have opposite signs from the corresponding coefficients of z^{-1} and z^{-2} in Eq. (6.16). Although this change of sign is sometimes confusing, it is essential to remember that the feedback coefficients $\{a_k\}$ always have the opposite sign in the difference equation from their sign in the system function. Note also that the direct form II requires only two delay elements to implement $H(z)$, one less than the direct form I implementation.

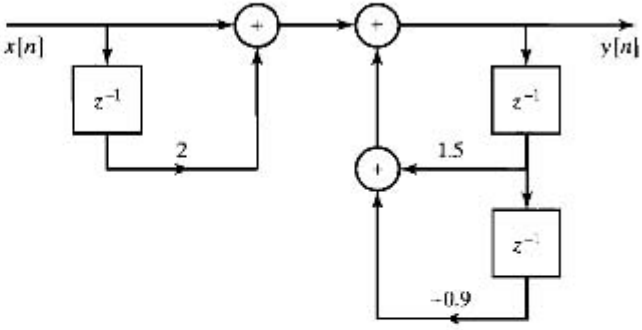


Figure 6.6 Direct form I implementation of Eq. (6.16).

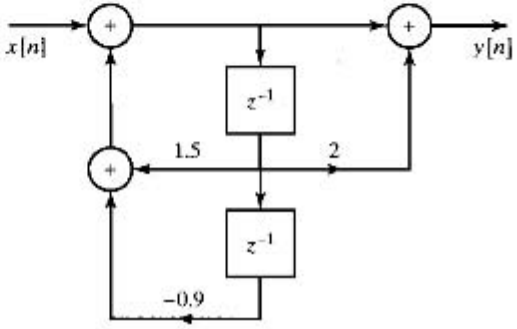


Figure 6.7 Direct form II implementation of Eq. (6.16).

In the preceding discussion, we developed two equivalent block diagrams for implementing an LTI system with system function given by Eq. (6.8). These block diagrams, which represent different computational algorithms for implementing the system, were obtained by manipulations based on the linearity of the system and the algebraic properties of the system function. Indeed, since the basic difference equations that represent an LTI system are linear, equivalent sets of difference equations can be obtained simply by linear transformations of the variables of the difference equations. Thus, there are an unlimited number of equivalent realizations of any given system. In Section 6.3, using an approach similar to that employed in this section, we will develop a number of other important and useful equivalent structures for implementing a system with system function as in Eq. (6.8). Before discussing these other forms, however, it is convenient to introduce signal flow graphs as an alternative to block diagrams for representing difference equations.

6.2 SIGNAL FLOW GRAPH REPRESENTATION OF LINEAR CONSTANT-COEFFICIENT DIFFERENCE EQUATIONS

A signal flow graph representation of a difference equation is essentially the same as a block diagram representation, except for a few notational differences. Formally, a

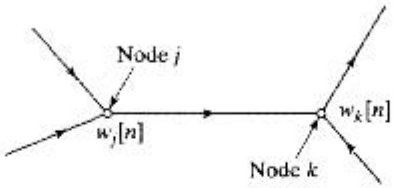


Figure 6.8 Example of nodes and branches in a signal flow graph.

signal flow graph is a network of directed branches that connect at nodes. Associated with each node is a variable or node value. The value associated with node k might be denoted w_k, or, since node variables for digital filters are generally sequences, we often indicate this explicitly with the notation $w_k[n]$. Branch (j, k) denotes a branch originating at node j and terminating at node k, with the direction from j to k being indicated by an arrowhead on the branch. This is shown in Figure 6.8. Each branch has an input signal and an output signal. The input signal from node j to branch (j, k) is the node value $w_j[n]$. In a linear signal flow graph, which is the only class we will consider, the output of a branch is a linear transformation of the input to the branch. The simplest example is a constant gain, i.e., when the output of the branch is simply a constant multiple of the input to the branch. The linear operation represented by the branch is typically indicated next to the arrowhead showing the direction of the branch. For the case of a constant multiplier, the constant is simply shown next to the arrowhead. When an explicit indication of the branch operation is omitted, this indicates a branch transmittance of unity, or the identity transformation. By definition, the value at each node in a graph is the sum of the outputs of all the branches entering the node.

To complete the definition of signal flow graph notation, we define two special types of nodes. *Source nodes* are nodes that have no entering branches. Source nodes are used to represent the injection of external inputs or signal sources into a graph. *Sink nodes* are nodes that have only entering branches. Sink nodes are used to extract outputs from a graph. Source nodes, sink nodes, and simple branch gains are illustrated in the signal flow graph of Figure 6.9. The linear equations represented by the figure are as follows:

$$\begin{aligned} w_1[n] &= x[n] + aw_2[n] + bw_2[n], \\ w_2[n] &= cw_1[n], \\ y[n] &= dx[n] + ew_2[n]. \end{aligned} \tag{6.17}$$

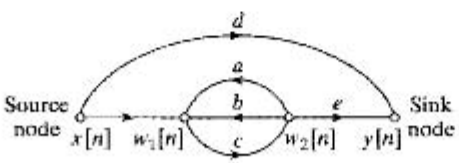


Figure 6.9 Example of a signal flow graph showing source and sink nodes.

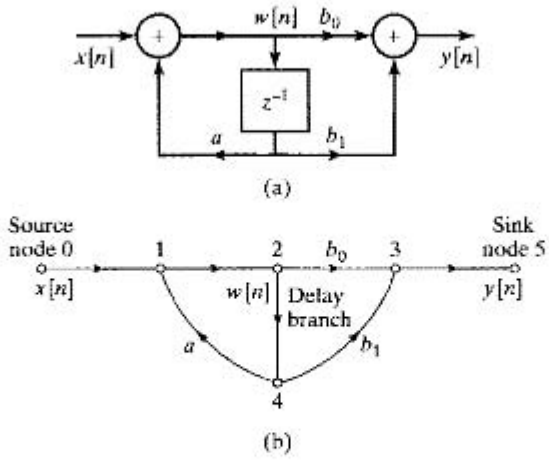


Figure 6.10 (a) Block diagram representation of a 1st-order digital filter. (b) Structure of the signal flow graph corresponding to the block diagram in (a).

Addition, multiplication by a constant, and delay are the basic operations required to implement a linear constant-coefficient difference equation. Since these are all linear operations, it is possible to use signal flow graph notation to depict algorithms for implementing LTI discrete-time systems. As an example of how the flow graph concepts just discussed can be applied to the representation of a difference equation, consider the block diagram in Figure 6.10(a), which is the direct form II realization of the system whose system function is given by Eq. (6.1). A signal flow graph corresponding to this system is shown in Figure 6.10(b). In the signal flow graph representation of difference equations, the node variables are sequences. In Figure 6.10(b), node 0 is a source node whose value is determined by the input sequence $x[n]$, and node 5 is a sink node whose value is denoted $y[n]$. Notice that the source and sink nodes are connected to the rest of the graph by unity-gain branches to clearly denote the input and output of the system. Obviously, nodes 3 and 5 have identical values. The extra branch with unity gain is simply used to highlight the fact that node 3 is the output of the system. In Figure 6.10(b), all branches except one (the delay branch (2, 4)) can be represented by a simple branch gain; i.e., the output signal is a constant multiple of the branch input. A delay cannot be represented in the time domain by a branch gain. However, the z-transform representation of a unit delay is multiplication by the factor z^{-1}. If we represented the difference equations by their corresponding z-transform equations, all the branches would be characterized by their system functions. In this case, each branch gain would be a function of z; e.g., a unit delay branch would have a gain of z^{-1}. By convention, we represent the variables in a signal flow graph as sequences rather than as z-transforms of sequences. However, to simplify the notation, we normally indicate a delay branch by showing its branch gain as z^{-1}, but it is understood that the output of such a branch is the branch input delayed by one sequence value. That is, the use of z^{-1} in a signal flow graph is in the sense of an operator that produces a delay of one sample. The graph of Figure 6.10(b) is shown in Figure 6.11 with this convention. The

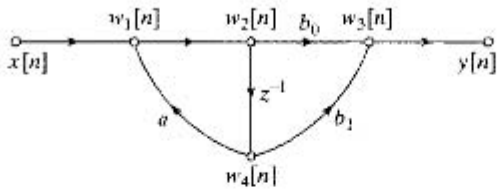


Figure 6.11 Signal flow graph of Figure 6.10(b) with the delay branch indicated by z^{-1}.

equations represented by Figure 6.11 are as follows:

$$w_1[n] = aw_4[n] + x[n], \tag{6.18a}$$

$$w_2[n] = w_1[n], \tag{6.18b}$$

$$w_3[n] = b_0 w_2[n] + b_1 w_4[n], \tag{6.18c}$$

$$w_4[n] = w_2[n-1], \tag{6.18d}$$

$$y[n] = w_3[n]. \tag{6.18e}$$

A comparison of Figure 6.10(a) and Figure 6.11 shows that there is a direct correspondence between branches in the block diagram and branches in the flow graph. In fact, the important difference between the two is that nodes in the flow graph represent both branching points and adders, whereas in the block diagram a special symbol is used for adders. A branching point in the block diagram is represented in the flow graph by a node that has only one incoming branch and one or more outgoing branches. An adder in the block diagram is represented in the signal flow graph by a node that has two (or more) incoming branches. In general, we will draw flow graphs with at most two inputs to each node, since most hardware implementations of addition have only two inputs. Signal flow graphs are therefore totally equivalent to block diagrams as pictorial representations of difference equations, but they are simpler to draw. Like block diagrams, they can be manipulated graphically to gain insight into the properties of a given system. A large body of signal flow graph theory exists that can be directly applied to discrete-time systems when they are represented in this form. (See Mason and Zimmermann, 1960; Chow and Cassignol, 1962; and Phillips and Nagle, 1995.) Although we will use flow graphs primarily for their pictorial value, we will use certain theorems relating to signal flow graphs in examining alternative structures for implementing linear systems.

Equations (6.18a)–(6.18e) define a multistep algorithm for computing the output of the LTI system from the input sequence $x[n]$. This example illustrates the kind of data precedence relations that generally arise in the implementation of IIR systems. Equations (6.18a)–(6.18e) cannot be computed in arbitrary order. Equations (6.18a) and (6.18c) require multiplications and additions, but Eqs. (6.18b) and (6.18e) simply rename variables. Equation (6.18d) represents the "updating" of the memory of the system. It would be implemented simply by replacing the contents of the memory register representing $w_4[n]$ by the value of $w_2[n]$, but this would have to be done consistently either *before* or *after* the evaluation of all the other equations. Initial-rest conditions would be imposed in this case by defining $w_2[-1] = 0$ or $w_4[0] = 0$. Clearly, Eqs. (6.18a)–(6.18e) must be computed in the order given, except that the last two could be interchanged or Eq. (6.18d) could be consistently evaluated first.

The flow graph represents a set of difference equations, with one equation being written at each node of the network. In the case of the flow graph of Figure 6.11, we can eliminate some of the variables rather easily to obtain the pair of equations

$$w_2[n] = aw_2[n-1] + x[n], \tag{6.19a}$$

$$y[n] = b_0 w_2[n] + b_1 w_2[n-1], \tag{6.19b}$$

which are in the form of Eqs. (6.15a) and (6.15b); i.e., in direct form II. Often, the manipulation of the difference equations of a flow graph is difficult when dealing with the time-domain variables, owing to feedback of delayed variables. In such cases, it is always possible to work with the z-transform representation, wherein all branches are simple gains since delay is represented in the z-transform by multiplication by z^{-1}. Problems 6.1–6.28 illustrate the utility of z-transform analysis of flow graphs for obtaining equivalent sets of difference equations.

Example 6.3 Determination of the System Function from a Flow Graph

To illustrate the use of the z-transform in determining the system function from a flow graph, consider Figure 6.12. The flow graph in this figure is not in direct form. Therefore, the system function cannot be written down by inspection of the graph. However, the set of difference equations represented by the graph can be written down by writing an equation for the value of each node variable in terms of the other node variables. The five equations are

$$w_1[n] = w_4[n] - x[n], \tag{6.20a}$$

$$w_2[n] = \alpha w_1[n], \tag{6.20b}$$

$$w_3[n] = w_2[n] + x[n], \tag{6.20c}$$

$$w_4[n] = w_3[n-1], \tag{6.20d}$$

$$y[n] = w_2[n] + w_4[n]. \tag{6.20e}$$

These are the equations that would be used to implement the system in the form described by the flow graph. Equations (6.20a)–(6.20e) can be represented by the z-transform equations

$$W_1(z) = W_4(z) - X(z), \tag{6.21a}$$

$$W_2(z) = \alpha W_1(z), \tag{6.21b}$$

$$W_3(z) = W_2(z) + X(z), \tag{6.21c}$$

$$W_4(z) = z^{-1} W_3(z), \tag{6.21d}$$

$$Y(z) = W_2(z) + W_4(z). \tag{6.21e}$$

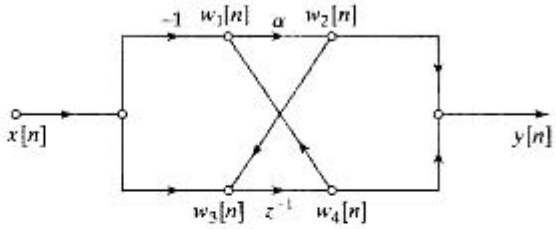


Figure 6.12 Flow graph not in standard direct form.

We can eliminate $W_1(z)$ and $W_3(z)$ from this set of equations by substituting Eq. (6.21a) into Eq. (6.21b) and Eq. (6.21c) into Eq. (6.21d), obtaining

$$W_2(z) = \alpha(W_4(z) - X(z)), \tag{6.22a}$$

$$W_4(z) = z^{-1}(W_2(z) + X(z)), \tag{6.22b}$$

$$Y(z) = W_2(z) + W_4(z). \tag{6.22c}$$

Equations (6.22a) and (6.22b) can be solved for $W_2(z)$ and $W_4(z)$, yielding

$$W_2(z) = \frac{\alpha(z^{-1} - 1)}{1 - \alpha z^{-1}} X(z), \tag{6.23a}$$

$$W_4(z) = \frac{z^{-1}(1 - \alpha)}{1 - \alpha z^{-1}} X(z), \tag{6.23b}$$

and substituting Eqs. (6.23a) and (6.23b) into Eq. (6.22c) leads to

$$Y(z) = \left(\frac{\alpha(z^{-1} - 1) + z^{-1}(1 - \alpha)}{1 - \alpha z^{-1}}\right) X(z) = \left(\frac{z^{-1} - \alpha}{1 - \alpha z^{-1}}\right) X(z). \tag{6.24}$$

Therefore, the system function of the flow graph of Figure 6.12 is

$$H(z) = \frac{z^{-1} - \alpha}{1 - \alpha z^{-1}}, \tag{6.25}$$

from which it follows that the impulse response of the system is

$$h[n] = \alpha^{n-1}u[n-1] - \alpha^{n+1}u[n]$$

and the direct form I flow graph is as shown in Figure 6.13.

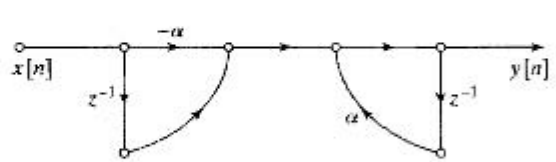


Figure 6.13 Direct form I equivalent of Figure 6.12.

Example 6.3 shows how the z-transform converts the time-domain expressions, which involve feedback and thus are difficult to solve, into linear equations that can be solved by algebraic techniques. The example also illustrates that different flow graph

representations define computational algorithms that require different amounts of computational resources. By comparing Figures 6.12 and 6.13, we see that the original implementation requires only one multiplication and one delay (memory) element, whereas the direct form I implementation would require two multiplications and two delay elements. The direct form II implementation would require one less delay, but it still would require two multiplications.

6.3 BASIC STRUCTURES FOR IIR SYSTEMS

In Section 6.1, we introduced two alternative structures for implementing an LTI system with system function as in Eq. (6.8). In this section we present the signal flow graph representations of those systems, and we also develop several other commonly used equivalent flow graph network structures. Our discussion will make it clear that, for any given rational system function, a wide variety of equivalent sets of difference equations or network structures exists. One consideration in the choice among these different structures is computational complexity. For example, in some digital implementations, structures with the fewest constant multipliers and the fewest delay branches are often most desirable. This is because multiplication is generally a time-consuming and costly operation in digital hardware and because each delay element corresponds to a memory register. Consequently, a reduction in the number of constant multipliers means an increase in speed, and a reduction in the number of delay elements means a reduction in memory requirements.

Other, more subtle, trade-offs arise in VLSI implementations, in which the area of a chip is often an important measure of efficiency. Modularity and simplicity of data transfer on the chip are also frequently very desirable in such implementations. In multiprocessor implementations, the most important considerations are often related to partitioning of the algorithm and communication requirements between processors. Other major considerations are the effects of a finite register length and finite-precision arithmetic. These effects depend on the way in which the computations are organized, i.e., on the structure of the signal flow graph. Sometimes it is desirable to use a structure that does not have the minimum number of multipliers and delay elements if that structure is less sensitive to finite register length effects.

In this section, we develop several of the most commonly used forms for implementing an LTI IIR system and obtain their flow graph representations.

6.3.1 Direct Forms

In Section 6.1, we obtained block diagram representations of the direct form I (Figure 6.3) and direct form II, or canonic direct form (Figure 6.5), structures for an LTI system whose input and output satisfy a difference equation of the form

$$y[n] - \sum_{k=1}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k], \tag{6.26}$$

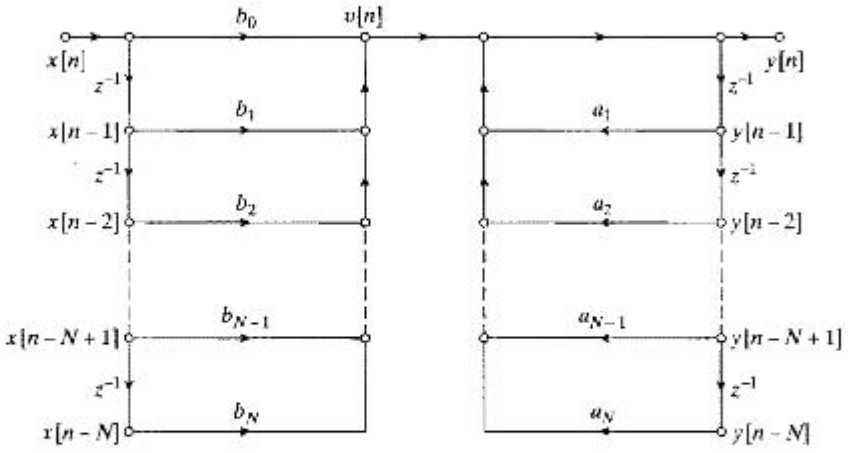


Figure 6.14 Signal flow graph of direct form I structure for an N^{th}-order system.

with the corresponding rational system function

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}. \tag{6.27}$$

In Figure 6.14, the direct form I structure of Figure 6.3 is shown using signal flow graph conventions, and Figure 6.15 shows the signal flow graph representation of the direct form II structure of Figure 6.5. Again, we have assumed for convenience that $N = M$. Note that we have drawn the flow graph so that each node has no more than two inputs. A node in a signal flow graph may have any number of inputs, but, as indicated earlier, this two-input convention results in a graph that is more closely related to programs and architectures for implementing the computation of the difference equations represented by the graph.

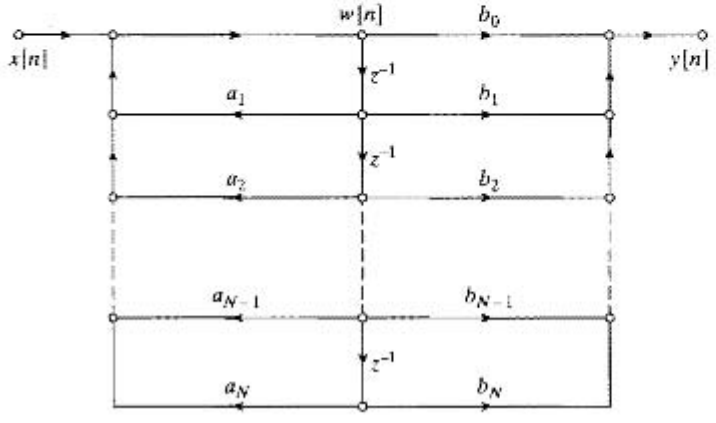


Figure 6.15 Signal flow graph of direct form II structure for an N^{th}-order system.

Example 6.4 Illustration of Direct Form I and Direct Form II Structures

Consider the system function

$$H(z) = \frac{1 + 2z^{-1} + z^{-2}}{1 - 0.75z^{-1} + 0.125z^{-2}}. \tag{6.28}$$

Since the coefficients in the direct form structures correspond directly to the coefficients of the numerator and denominator polynomials (taking into account the minus sign in the denominator of Eq. (6.27)), we can draw these structures by inspection with reference to Figures 6.14 and 6.15. The direct form I and direct form II structures for this example are shown in Figures 6.16 and 6.17, respectively.

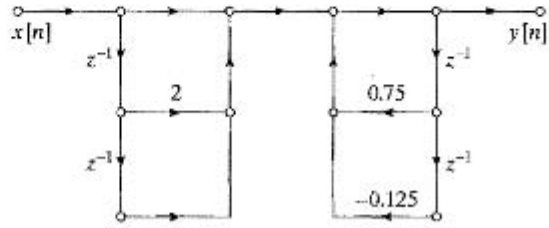


Figure 6.16 Direct form I structure for Example 6.4.

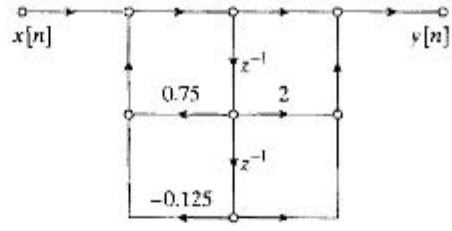


Figure 6.17 Direct form II structure for Example 6.4.

6.3.2 Cascade Form

The direct form structures were obtained directly from the system function $H(z)$, written as a ratio of polynomials in the variable z^{-1} as in Eq. (6.27). If we factor the numerator and denominator polynomials, we can express $H(z)$ in the form

$$H(z) = A \frac{\prod_{k=1}^{M_1}(1 - f_k z^{-1}) \prod_{k=1}^{M_2}(1 - g_k z^{-1})(1 - g_k^* z^{-1})}{\prod_{k=1}^{N_1}(1 - c_k z^{-1}) \prod_{k=1}^{N_2}(1 - d_k z^{-1})(1 - d_k^* z^{-1})}, \tag{6.29}$$

where $M = M_1 + 2M_2$ and $N = N_1 + 2N_2$. In this expression, the 1st-order factors represent real zeros at f_k and real poles at c_k, and the 2nd-order factors represent complex conjugate pairs of zeros at g_k and g_k^* and complex conjugate pairs of poles

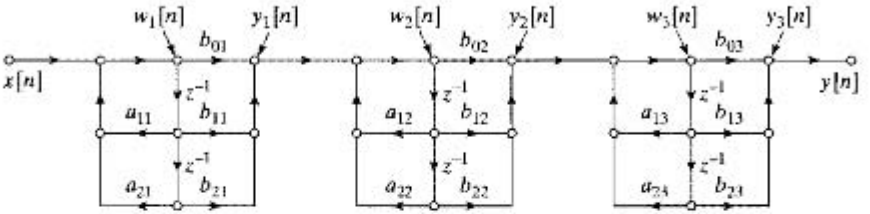


Figure 6.18 Cascade structure for a 6th-order system with a direct form II realization of each 2nd-order subsystem.

at d_k and d_k^*. This represents the most general distribution of poles and zeros when all the coefficients in Eq. (6.27) are real. Equation (6.29) suggests a class of structures consisting of a cascade of 1st- and 2nd-order systems. There is considerable freedom in the choice of composition of the subsystems and in the order in which the subsystems are cascaded. In practice, however, it is often desirable to implement the cascade realization using a minimum of storage and computation. A modular structure that is advantageous for many types of implementations is obtained by combining pairs of real factors and complex conjugate pairs into 2nd-order factors so that Eq. (6.29) can be expressed as

$$H(z) = \prod_{k=1}^{N_s} \frac{b_{0k} + b_{1k}z^{-1} + b_{2k}z^{-2}}{1 - a_{1k}z^{-1} - a_{2k}z^{-2}}, \tag{6.30}$$

where $N_s = \lfloor (N+1)/2 \rfloor$ is the largest integer contained in $(N+1)/2$. In writing $H(z)$ in this form, we have assumed that $M \leq N$ and that the real poles and zeros have been combined in pairs. If there are an odd number of real zeros, one of the coefficients b_{2k} will be zero. Likewise, if there are an odd number of real poles, one of the coefficients a_{2k} will be zero. The individual 2nd-order sections can be implemented using either of the direct form structures; however, the previous discussion shows that we can implement a cascade structure with a minimum number of multiplications and a minimum number of delay elements if we use the direct form II structure for each 2nd-order section. A cascade structure for a 6th-order system using three direct form II 2nd-order sections is shown in Figure 6.18. The difference equations represented by a general cascade of direct form II 2nd-order sections are of the form

$$y_0[n] = x[n], \tag{6.31a}$$

$$w_k[n] = a_{1k}w_k[n-1] + a_{2k}w_k[n-2] + y_{k-1}[n], \quad k = 1, 2, \ldots, N_s, \tag{6.31b}$$

$$y_k[n] = b_{0k}w_k[n] + b_{1k}w_k[n-1] + b_{2k}w_k[n-2], \quad k = 1, 2, \ldots, N_s, \tag{6.31c}$$

$$y[n] = y_{N_s}[n]. \tag{6.31d}$$

It is easy to see that a variety of theoretically equivalent systems can be obtained by simply pairing the poles and zeros in different ways and by ordering the 2nd-order sections in different ways. Indeed, if there are N_s 2nd-order sections, there are $N_s!$ (N_s factorial) pairings of the poles with zeros and $N_s!$ orderings of the resulting 2nd-order sections, or a total of $(N_s!)^2$ different pairings and orderings. Although these all have the same overall system function and corresponding input–output relation when infinite-

precision arithmetic is used, their behavior with finite-precision arithmetic can be quite different, as we will see in Sections 6.8–6.10.

Example 6.5 Illustration of Cascade Structures

Let us again consider the system function of Eq. (6.28). Since this is a 2nd-order system, a cascade structure with direct form II 2nd-order sections reduces to the structure of Figure 6.17. Alternatively, to illustrate the cascade structure, we can use 1st-order systems by expressing $H(z)$ as a product of 1st-order factors, as in

$$H(z) = \frac{1 + 2z^{-1} + z^{-2}}{1 - 0.75z^{-1} + 0.125z^{-2}} = \frac{(1 + z^{-1})(1 + z^{-1})}{(1 - 0.5z^{-1})(1 - 0.25z^{-1})}. \tag{6.32}$$

Since all of the poles and zeros are real, a cascade structure with 1st-order sections has real coefficients. If the poles and/or zeros were complex, only a 2nd-order section would have real coefficients. Figure 6.19 shows two equivalent cascade structures, each of which has the system function in Eq. (6.32). The difference equations represented by the flow graphs in the figure can be written down easily. Problem 6.22 is concerned with finding other, equivalent system configurations.

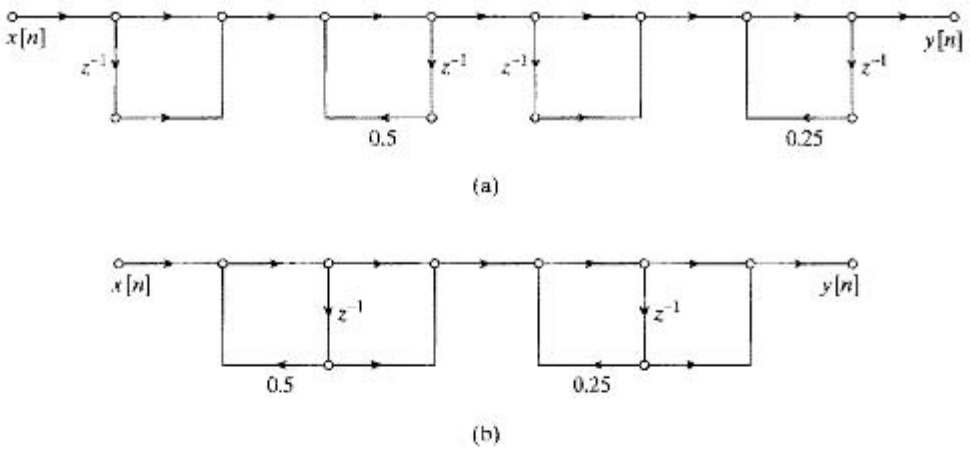


Figure 6.19 Cascade structures for Example 6.5. (a) Direct form I subsections. (b) Direct form II subsections.

A final comment should be made about our definition of the system function for the cascade form. As defined in Eq. (6.30), each 2nd-order section has five constant multipliers. For comparison, let us assume that $M = N$ in $H(z)$ as given by Eq. (6.27), and furthermore, assume that N is an even integer, so that $N_s = N/2$. Then, the direct form I and II structures have $2N + 1$ constant multipliers, while the cascade form structure suggested by Eq. (6.30) has $5N/2$ constant multipliers. For the 6th-order system in Figure 6.18, we require a total of 15 multipliers, while the equivalent direct forms would require a total of 13 multipliers. Another definition of the cascade form is

$$H(z) = b_0 \prod_{k=1}^{N_s} \frac{1 + \tilde{b}_{1k}z^{-1} + \tilde{b}_{2k}z^{-2}}{1 - a_{1k}z^{-1} - a_{2k}z^{-2}}, \tag{6.33}$$

where b_0 is the leading coefficient in the numerator polynomial of Eq. (6.27) and $\tilde{b}_{ik} = b_{ik}/b_{0k}$ for $i = 1, 2$ and $k = 1, 2, \ldots, N_s$. This form for $H(z)$ suggests a cascade of four-multiplier 2nd-order sections, with a single overall gain constant b_0. This cascade form has the same number of constant multipliers as the direct form structures. As discussed in Section 6.9, the five-multiplier 2nd-order sections are commonly used when implemented with fixed-point arithmetic, because they make it possible to distribute the overall gain of the system and thereby control the size of signals at various critical points in the system. When floating-point arithmetic is used and dynamic range is not a problem, the four-multiplier 2nd-order sections can be used to decrease the amount of computation. Further simplification results for zeros on the unit circle. In this case, $\tilde{b}_{2k} = 1$, and we require only three multipliers per 2nd-order section.

6.3.3 Parallel Form

As an alternative to factoring the numerator and denominator polynomials of $H(z)$, we can express a rational system function as given by Eq. (6.27) or (6.29) as a partial fraction expansion in the form

$$H(z) = \sum_{k=0}^{N_p} C_k z^{-k} + \sum_{k=1}^{N_1} \frac{A_k}{1 - c_k z^{-1}} + \sum_{k=1}^{N_2} \frac{B_k(1 - e_k z^{-1})}{(1 - d_k z^{-1})(1 - d_k^* z^{-1})}, \tag{6.34}$$

where $N = N_1 + 2N_2$. If $M \geq N$, then $N_p = M - N$; otherwise, the first summation in Eq. (6.34) is not included. If the coefficients a_k and b_k are real in Eq. (6.27), then the quantities A_k, B_k, C_k, c_k, and e_k are all real. In this form, the system function can be interpreted as representing a parallel combination of 1st- and 2nd-order IIR systems, with possibly N_p simple scaled delay paths. Alternatively, we may group the real poles in pairs, so that $H(z)$ can be expressed as

$$H(z) = \sum_{k=0}^{N_p} C_k z^{-k} + \sum_{k=1}^{N_s} \frac{e_{0k} + e_{1k} z^{-1}}{1 - a_{1k} z^{-1} - a_{2k} z^{-2}}, \tag{6.35}$$

where, as in the cascade form, $N_s = \lfloor (N+1)/2 \rfloor$ is the largest integer contained in $(N+1)/2$, and if $N_p = M - N$ is negative, the first sum is not present. A typical example for $N = M = 6$ is shown in Figure 6.20. The general difference equations for the parallel form with 2nd-order direct form II sections are

$$w_k[n] = a_{1k} w_k[n-1] + a_{2k} w_k[n-2] + x[n], \quad k = 1, 2, \ldots, N_s, \tag{6.36a}$$

$$y_k[n] = e_{0k} w_k[n] + e_{1k} w_k[n-1], \quad k = 1, 2, \ldots, N_s, \tag{6.36b}$$

$$y[n] = \sum_{k=0}^{N_p} C_k x[n-k] + \sum_{k=1}^{N_s} y_k[n]. \tag{6.36c}$$

If $M < N$, then the first summation in Eq. (6.36c) is not included.

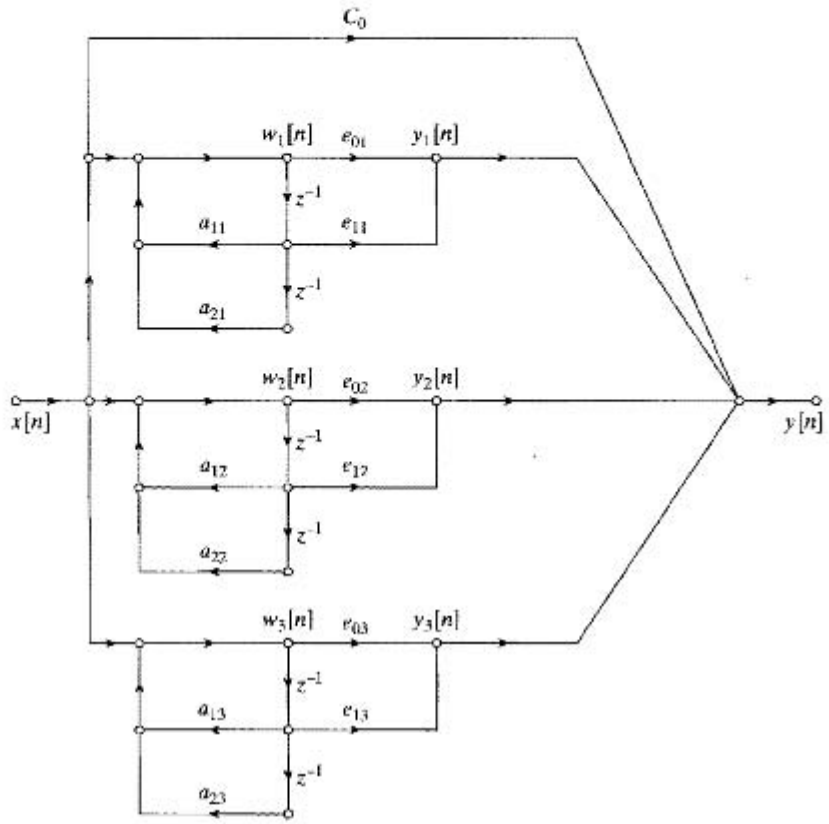


Figure 6.20 Parallel form structure for 6th-order system ($M = N = 6$) with the real and complex poles grouped in pairs.

Example 6.6 Illustration of Parallel Form Structures

Consider again the system function used in Examples 6.4 and 6.5. For the parallel form, we must express $H(z)$ in the form of either Eq. (6.34) or Eq. (6.35). If we use 2nd-order sections,

$$H(z) = \frac{1 + 2z^{-1} + z^{-2}}{1 - 0.75z^{-1} + 0.125z^{-2}} = 8 + \frac{-7 + 8z^{-1}}{1 - 0.75z^{-1} + 0.125z^{-2}}. \tag{6.37}$$

The parallel form realization for this example with a 2nd-order section is shown in Figure 6.21.

Since all the poles are real, we can obtain an alternative parallel form realization by expanding $H(z)$ as

$$H(z) = 8 + \frac{18}{1 - 0.5z^{-1}} - \frac{25}{1 - 0.25z^{-1}}. \tag{6.38}$$

The resulting parallel form with 1st-order sections is shown in Figure 6.22. As in the general case, the difference equations represented by both Figures 6.21 and 6.22 can be written down by inspection.

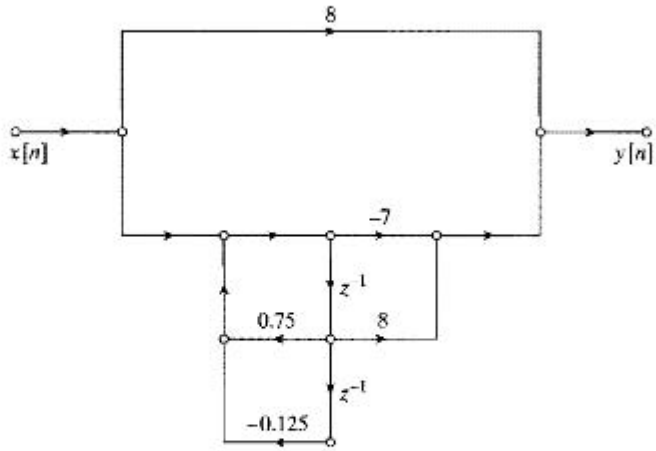


Figure 6.21 Parallel form structure for Example 6.6 using a 2nd-order system.

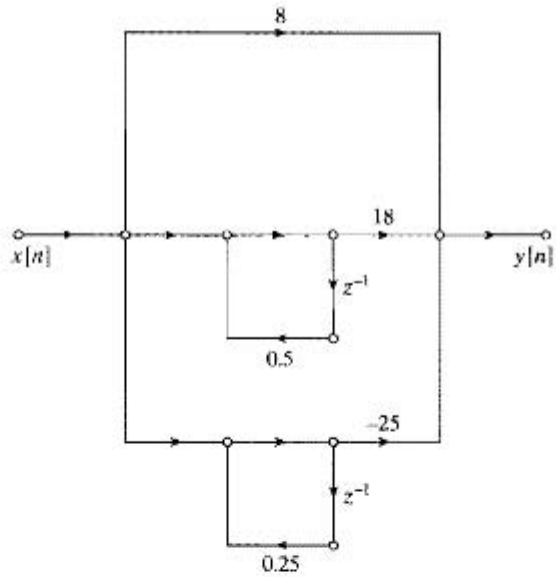


Figure 6.22 Parallel form structure for Example 6.6 using 1st-order systems.

6.3.4 Feedback in IIR Systems

All the flow graphs of this section have feedback loops; i.e., they have closed paths that begin at a node and return to that node by traversing branches only in the direction of their arrowheads. Such a structure in the flow graph implies that a node variable in a loop depends directly or indirectly on itself. A simple example is shown in Figure 6.23(a), which represents the difference equation

$$y[n] = ay[n-1] + x[n]. \tag{6.39}$$

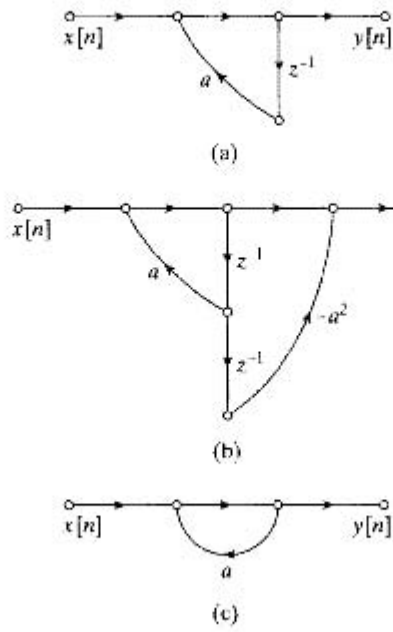


Figure 6.23 (a) System with feedback loop. (b) FIR system with feedback loop. (c) Noncomputable system.

Such loops are necessary (but not sufficient) to generate infinitely long impulse responses. This can be seen if we consider a network with no feedback loops. In such a case, any path from the input to the output can pass through each delay element only once. Therefore, the longest delay between the input and output would occur for a path that passes through all of the delay elements in the network. Thus, for a network with no loops, the impulse response is no longer than the total number of delay elements in the network. From this, we conclude that if a network has no loops, then the system function has only zeros (except for poles at $z = 0$), and the number of zeros can be no more than the number of delay elements in the network.

Returning to the simple example of Figure 6.23(a), we see that when the input is the impulse sequence $\delta[n]$, the single-input sample continually recirculates in the feedback loop with either increasing (if $|a| > 1$) or decreasing (if $|a| < 1$) amplitude owing to multiplication by the constant a, so that the impulse response is $h[n] = a^n u[n]$. This illustrates how feedback can create an infinitely long impulse response.

If a system function has poles, a corresponding block diagram or signal flow graph will have feedback loops. On the other hand, neither poles in the system function nor loops in the network are sufficient for the impulse response to be infinitely long. Figure 6.23(b) shows a network with a feedback loop, but with an impulse response of finite length. This is because the pole of the system function cancels with a zero; i.e., for Figure 6.23(b),

$$H(z) = \frac{1 - a^2 z^{-2}}{1 - az^{-1}} = \frac{(1 - az^{-1})(1 + az^{-1})}{1 - az^{-1}} = 1 + az^{-1}. \tag{6.40}$$

The impulse response of this system is $h[n] = \delta[n] + a\delta[n-1]$. The system is a simple example of a general class of FIR systems called *frequency-sampling systems*. This class of systems is considered in more detail in Problems 6.39 and 6.51.

Loops in a network pose special problems in implementing the computations implied by the network. As we have discussed, it must be possible to compute the node variables in a network in sequence such that all necessary values are available when needed. In some cases, there is no way to order the computations so that the node variables of a flow graph can be computed in sequence. Such a network is called *noncomputable* (see Crochiere and Oppenheim, 1975). A simple noncomputable network is shown in Figure 6.23(c). The difference equation for this network is

$$y[n] = ay[n] + x[n]. \tag{6.41}$$

In this form, we cannot compute $y[n]$ because the right-hand side of the equation involves the quantity we wish to compute. The fact that a flow graph is noncomputable does *not* mean the equations represented by the flow graph cannot be solved; indeed, the solution to Eq. (6.41) is $y[n] = x[n]/(1-a)$. It simply means that the flow graph does not represent a set of difference equations that can be solved successively for the node variables. The key to the computability of a flow graph is that all loops must contain at least one unit delay element. Thus, in manipulating flow graphs representing implementations of LTI systems, we must be careful not to create delay-free loops. Problem 6.37 deals with a system having a delay-free loop. Problem 7.51 shows how a delay-free loop can be introduced.

6.4 TRANSPOSED FORMS

The theory of linear signal flow graphs provides a variety of procedures for transforming such graphs into different forms while leaving the overall system function between input and output unchanged. One of these procedures, called *flow graph reversal* or *transposition*, leads to a set of transposed system structures that provide some useful alternatives to the structures discussed in the previous section.

Transposition of a flow graph is accomplished by reversing the directions of all branches in the network while keeping the branch transmittances as they were and reversing the roles of the input and output so that source nodes become sink nodes and vice versa. For single-input, single-output systems, the resulting flow graph has the same system function as the original graph if the input and output nodes are interchanged. Although we will not formally prove this result here,[3] we will demonstrate that it is valid with two examples.

Example 6.7 Transposed Form for a 1st-Order System with No Zeros

The 1st-order system corresponding to the flow graph in Figure 6.24(a) has system function

$$H(z) = \frac{1}{1 - az^{-1}}. \tag{6.42}$$

[3]The theorem follows directly from Mason's gain formula of signal flow graph theory. (See Mason and Zimmermann, 1960; Chow and Cassignol, 1962; or Phillips and Nagle, 1995.)

To obtain the transposed form for this system, we reverse the directions of all the branch arrows, taking the output where the input was and injecting the input where the output was. The result is shown in Figure 6.24(b). It is usually convenient to draw the transposed network with the input on the left and the output on the right, as shown in Figure 6.24(c). Comparing Figures 6.24(a) and 6.24(c) we note that the only difference is that in Figure 6.24(a), we multiply the *delayed* output sequence $y[n-1]$ by the coefficient a, whereas in Figure 6.24(c) we multiply the output $y[n]$ by the coefficient a and then delay the resulting product. Since the two operations can be interchanged, we can conclude by inspection that the original system in Figure 6.24(a) and the corresponding transposed system in Figure 6.24(c) have the same system function.

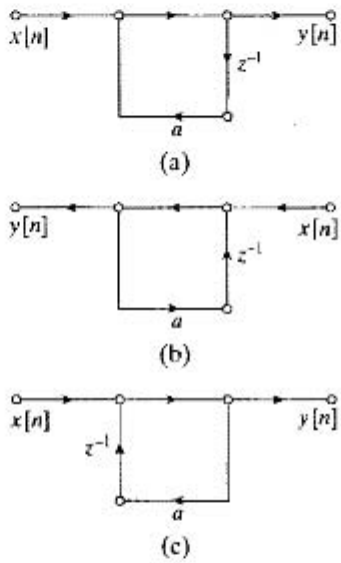


Figure 6.24 (a) Flow graph of simple 1st-order system. (b) Transposed form of (a). (c) Structure of (b) redrawn with input on left.

In Example 6.7, it is straightforward to see that the original system and its transpose have the same system function. However, for more complicated graphs, the result is often not so obvious. This is illustrated by the next example.

Example 6.8 Transposed Form for a Basic 2nd-Order Section

Consider the basic 2nd-order section depicted in Figure 6.25. The corresponding difference equations for this system are

$$w[n] = a_1 w[n-1] + a_2 w[n-2] + x[n], \tag{6.43a}$$

$$y[n] = b_0 w[n] + b_1 w[n-1] + b_2 w[n-2]. \tag{6.43b}$$

The transposed flow graph is shown in Figure 6.26; its corresponding difference equations are

$$v_0[n] = b_0x[n] + v_1[n-1], \tag{6.44a}$$

$$y[n] = v_0[n], \tag{6.44b}$$

$$v_1[n] = a_1y[n] + b_1x[n] + v_2[n-1], \tag{6.44c}$$

$$v_2[n] = a_2y[n] + b_2x[n]. \tag{6.44d}$$

Equations (6.43a)–(6.43b) and Eqs. (6.44a)–(6.44d) are different ways to organize the computation of the output samples $y[n]$ from the input samples $x[n]$, and it is not immediately clear that the two sets of difference equations are equivalent. One way to show this equivalence is to use the z-transform representations of both sets of equations, solve for the ratio $Y(z)/X(z) = H(z)$ in both cases, and compare the results. Another way is to substitute Eq. (6.44d) into Eq. (6.44c), substitute the result into Eq. (6.44a), and finally, substitute that result into Eq. (6.44b). The final result is

$$y[n] = a_1y[n-1] + a_2y[n-2] + b_0x[n] + b_1x[n-1] + b_2x[n-2]. \tag{6.45}$$

Since the network of Figure 6.25 is a direct form II structure, it is easily seen that the input and output of the system in Figure 6.25 also satisfies the difference Eq. (6.45). Therefore, for initial-rest conditions, the systems in Figures 6.25 and 6.26 are equivalent.

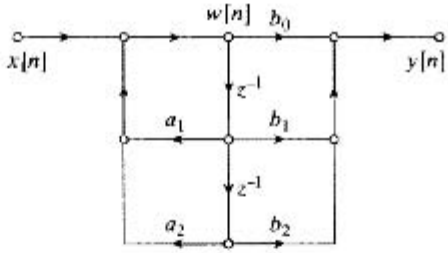


Figure 6.25 Direct form II structure for Example 6.8.

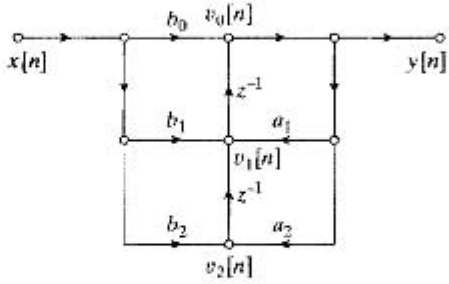


Figure 6.26 Transposed direct form II structure for Example 6.8.

The transposition theorem can be applied to any of the structures that we have discussed so far. For example, the result of applying the theorem to the direct form I structure of Figure 6.14 is shown in Figure 6.27, and similarly, the structure obtained by

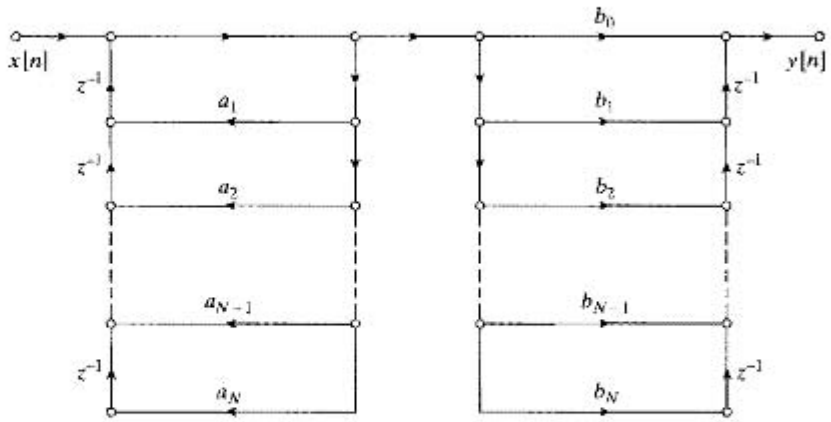


Figure 6.27 General flow graph resulting from applying the transposition theorem to the direct form I structure of Figure 6.14.

transposing the direct form II structure of Figure 6.15 is shown in Figure 6.28. If a signal flow graph configuration is transposed, the number of delay branches and the number of coefficients remain the same. Thus, the transposed direct form II structure is also a canonic structure. Transposed structures derived from direct forms are also "direct" in the sense that they can be constructed by inspection of the numerator and denominator of the system function.

An important point becomes evident through a comparison of Figures 6.15 and 6.28. Whereas the direct form II structure implements the poles first and then the zeros, the transposed direct form II structure implements the zeros first and then the poles. These differences can become important in the presence of quantization in finite-precision digital implementations or in the presence of noise in discrete-time analog implementations.

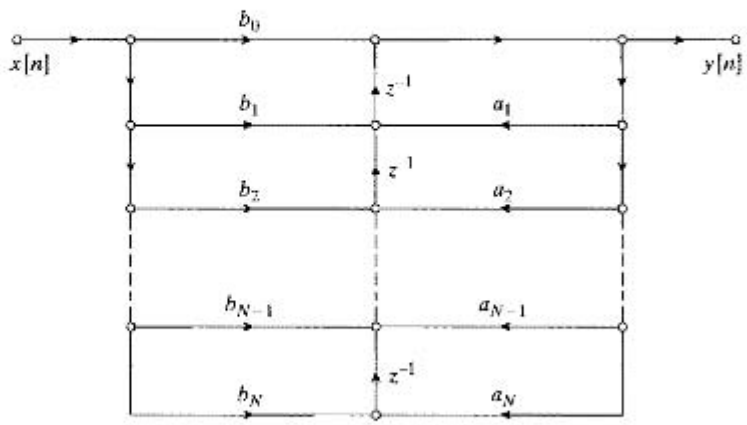


Figure 6.28 General flow graph resulting from applying the transposition theorem to the direct form II structure of Figure 6.15.

When the transposition theorem is applied to cascade or parallel structures, the individual 2^{nd}-order systems are replaced by transposed structures. For example, applying the transposition theorem to Figure 6.18 results in a cascade of three transposed direct form II sections (each like the one in Example 6.8) with the same coefficients as in Figure 6.18, but with the order of the cascade reversed. A similar statement can be made about the transposition of Figure 6.20.

The transposition theorem further emphasizes that an infinite variety of implementation structures exists for any given rational system function. The transposition theorem provides a simple procedure for generating new structures. The problems of implementing systems with finite-precision arithmetic have motivated the development of many more classes of equivalent structures than we can discuss here. Thus, we concentrate only on the most commonly used structures.

6.5 BASIC NETWORK STRUCTURES FOR FIR SYSTEMS

The direct, cascade, and parallel form structures discussed in Sections 6.3 and 6.4 are the most common basic structures for IIR systems. These structures were developed under the assumption that the system function had both poles and zeros. Although the direct and cascade forms for IIR systems include FIR systems as a special case, there are additional specific forms for FIR systems.

6.5.1 Direct Form

For causal FIR systems, the system function has only zeros (except for poles at $z = 0$), and since the coefficients a_k are all zero, the difference equation of Eq. (6.9) reduces to

$$y[n] = \sum_{k=0}^{M} b_k x[n-k]. \tag{6.46}$$

This can be recognized as the discrete convolution of $x[n]$ with the impulse response

$$h[n] = \begin{cases} b_n & n = 0, 1, \ldots, M, \\ 0 & \text{otherwise.} \end{cases} \tag{6.47}$$

In this case, the direct form I and direct form II structures in Figures 6.14 and 6.15 both reduce to the direct form FIR structure as redrawn in Figure 6.29. Because of the chain of delay elements across the top of the diagram, this structure is also referred to as a *tapped delay line* structure or a *transversal filter* structure. As seen from Figure 6.29, the signal at each tap along this chain is weighted by the appropriate coefficient (impulse-response value), and the resulting products are summed to form the output $y[n]$.

The transposed direct form for the FIR case is obtained by applying the transposition theorem to Figure 6.29 or, equivalently, by setting the coefficients a_k to zero in Figure 6.27 or Figure 6.28. The result is shown in Figure 6.30.

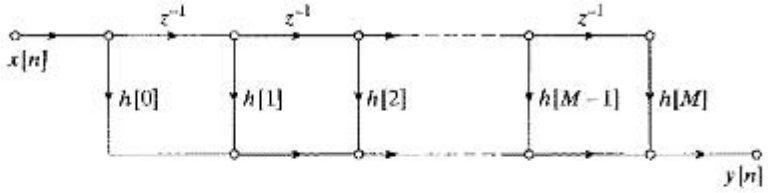


Figure 6.29 Direct form realization of an FIR system.

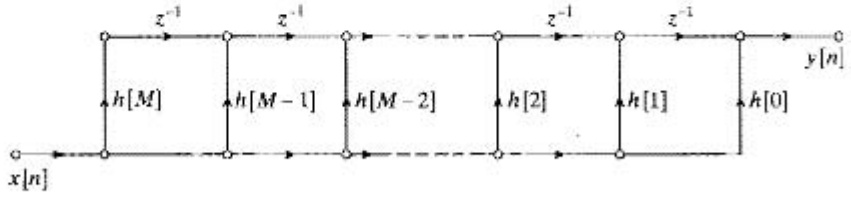


Figure 6.30 Transposition of the network of Figure 6.29.

6.5.2 Cascade Form

The cascade form for FIR systems is obtained by factoring the polynomial system function. That is, we represent $H(z)$ as

$$H(z) = \sum_{n=0}^{M} h[n]z^{-n} = \prod_{k=1}^{M_s} (b_{0k} + b_{1k}z^{-1} + b_{2k}z^{-2}), \tag{6.48}$$

where $M_s = \lfloor (M+1)/2 \rfloor$ is the largest integer contained in $(M+1)/2$. If M is odd, one of the coefficients b_{2k} will be zero, since $H(z)$ in that case would have an odd number of real zeros. The flow graph representing Eq. (6.48) is shown in Figure 6.31, which is identical in form to Figure 6.18 with the coefficients a_{1k} and a_{2k} all zero. Each of the 2nd-order sections in Figure 6.31 uses the direct form shown in Figure 6.29. Another alternative is to use transposed direct form 2nd-order sections or, equivalently, to apply the transposition theorem to Figure 6.31.

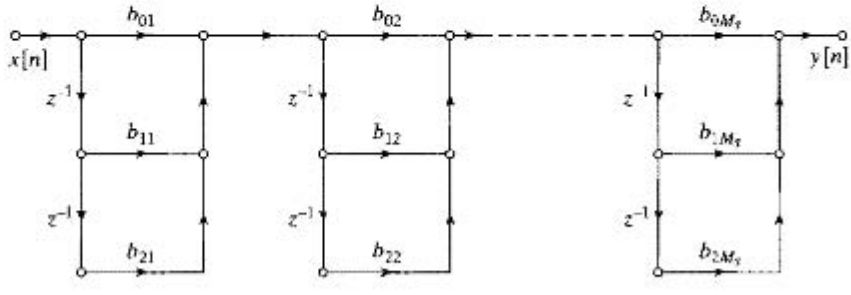


Figure 6.31 Cascade form realization of an FIR system.

6.5.3 Structures for Linear-Phase FIR Systems

In Chapter 5, we showed that causal FIR systems have generalized linear phase if the impulse response satisfies the symmetry condition

$$h[M-n] = h[n] \qquad n = 0, 1, \ldots, M \tag{6.49a}$$

or

$$h[M-n] = -h[n] \qquad n = 0, 1, \ldots, M. \tag{6.49b}$$

With either of these conditions, the number of coefficient multipliers can be essentially halved. To see this, consider the following manipulations of the discrete convolution equation, assuming that M is an even integer corresponding to type I or type III systems:

$$\begin{aligned} y[n] &= \sum_{k=0}^{M} h[k]x[n-k] \\ &= \sum_{k=0}^{M/2-1} h[k]x[n-k] + h[M/2]x[n-M/2] + \sum_{k=M/2+1}^{M} h[k]x[n-k] \\ &= \sum_{k=0}^{M/2-1} h[k]x[n-k] + h[M/2]x[n-M/2] + \sum_{k=0}^{M/2-1} h[M-k]x[n-M+k]. \end{aligned}$$

For type I systems, we use Eq. (6.49a) to obtain

$$y[n] = \sum_{k=0}^{M/2-1} h[k](x[n-k] + x[n-M+k]) + h[M/2]x[n-M/2]. \tag{6.50}$$

For type III systems, we use Eq. (6.49b) to obtain

$$y[n] = \sum_{k=0}^{M/2-1} h[k](x[n-k] - x[n-M+k]). \tag{6.51}$$

For the case of M an odd integer, the corresponding equations are, for type II systems,

$$y[n] = \sum_{k=0}^{(M-1)/2} h[k](x[n-k] + x[n-M+k]) \tag{6.52}$$

and, for type IV systems,

$$y[n] = \sum_{k=0}^{(M-1)/2} h[k](x[n-k] - x[n-M+k]). \tag{6.53}$$

Equations (6.50)–(6.53) imply structures with either $M/2+1$, $M/2$, or $(M+1)/2$ coefficient multipliers, rather than the M coefficient multipliers of the general direct form structure of Figure 6.29. Figure 6.32 shows the structure implied by Eq. (6.50), and Figure 6.33 shows the structure implied by Eq. (6.52).

In our discussion of linear-phase systems in Section 5.7.3, we showed that the symmetry conditions of Eqs. (6.49a) and (6.49b) cause the zeros of $H(z)$ to occur in mirror-image pairs. That is, if z_0 is a zero of $H(z)$, then $1/z_0$ is also a zero of $H(z)$. Furthermore, if $h[n]$ is real, then the zeros of $H(z)$ occur in complex-conjugate pairs.

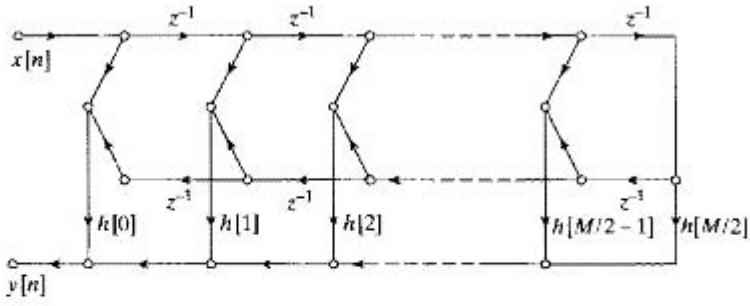


Figure 6.32 Direct form structure for an FIR linear-phase system when M is an even integer.

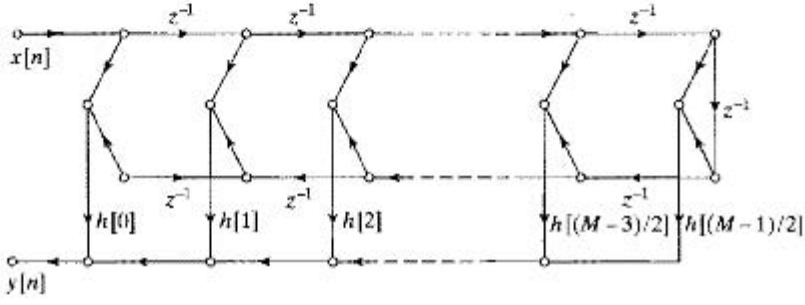


Figure 6.33 Direct form structure for an FIR linear-phase system when M is an odd integer.

As a consequence, real zeros not on the unit circle occur in reciprocal pairs. Complex zeros not on the unit circle occur in groups of four, corresponding to the complex conjugates and reciprocals. If a zero is on the unit circle, its reciprocal is also its conjugate. Consequently, complex zeros on the unit circle are conveniently grouped into pairs. Zeros at $z = \pm 1$ are their own reciprocal and complex conjugate. The four cases are summarized in Figure 6.34, where the zeros at z_1, z_1^*, $1/z_1$, and $1/z_1^*$ are considered as a group of four. The zeros at z_2 and $1/z_2$ are considered as a group of two, as are the zeros at z_3 and z_3^*. The zero at z_4 is considered singly. If $H(z)$ has the zeros shown in Figure 6.34, it can be factored into a product of 1^{st}-, 2^{nd}-, and 4^{th}-order factors. Each of these factors is a polynomial whose coefficients have the same symmetry as the coefficients of $H(z)$; i.e., each factor is a linear-phase polynomial in z^{-1}. Therefore, the system can be implemented as a cascade of 1^{st}-, 2^{nd}-, and 4^{th}-order systems. For example, the system function corresponding to the zeros of Figure 6.34 can be expressed as

$$\begin{aligned} H(z) = h[0](1+z^{-1})(1+az^{-1}+z^{-2})(1+bz^{-1}+z^{-2}) \\ \times (1+cz^{-1}+dz^{-2}+cz^{-3}+z^{-4}), \end{aligned} \tag{6.54}$$

where

$$a = (z_2 + 1/z_2), \qquad b = 2\mathcal{R}e\{z_3\}, \qquad c = -2\mathcal{R}e\{z_1 + 1/z_1\}, \qquad d = 2 + |z_1 + 1/z_1|^2.$$

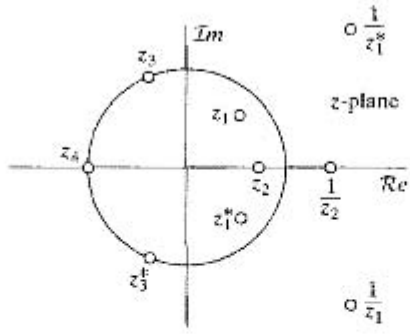


Figure 6.34 Symmetry of zeros for a linear-phase FIR filter.

This representation suggests a cascade structure consisting of linear-phase elements. It can be seen that the order of the system function polynomial is $M = 9$ and the number of different coefficient multipliers is five. This is the same number ($(M + 1)/2 = 5$) of constant multipliers required for implementing the system in the linear-phase direct form of Figure 6.32. Thus, with no additional multiplications, we obtain a modular structure in terms of a cascade of short linear-phase FIR systems.

6.6 LATTICE FILTERS

In Sections 6.3.2 and 6.5.2, we discussed cascade forms for both IIR and FIR systems obtained by factoring their system functions into 1st- and 2nd-order sections. Another interesting and useful cascade structure is based on a cascade (output to input) connection of the basic structure shown in Figure 6.35(a). In the case of Figure 6.35(a) the basic building block system has two inputs and two outputs, and is called a two-port flow graph. Figure 6.35(b) shows the equivalent flow graph representation. Figure 6.36 shows a cascade of M of these basic elements with a "termination" at each end of the cascade so that the overall system is a single-input single-output system with input $x[n]$ feeding both inputs of two-port building block (1) and output $y[n]$ defined to be $a^{(M)}[n]$, the upper output of the last two-port building block M. (The lower output of the M^{th} stage is generally ignored.) Although such a structure could take many different forms

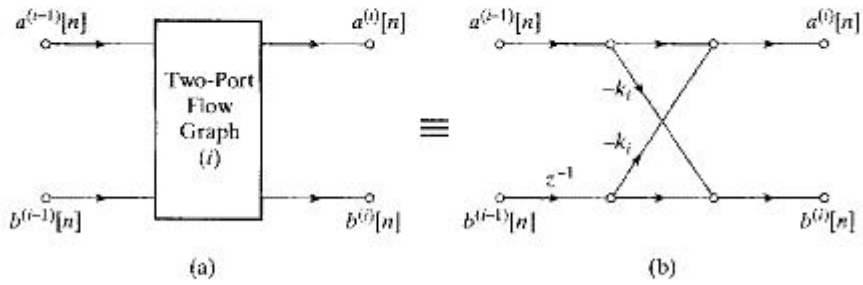


Figure 6.35 One section of the lattice structure for FIR lattice filters. (a) Block diagram representation of a two-port building block (b) Equivalent flow graph.

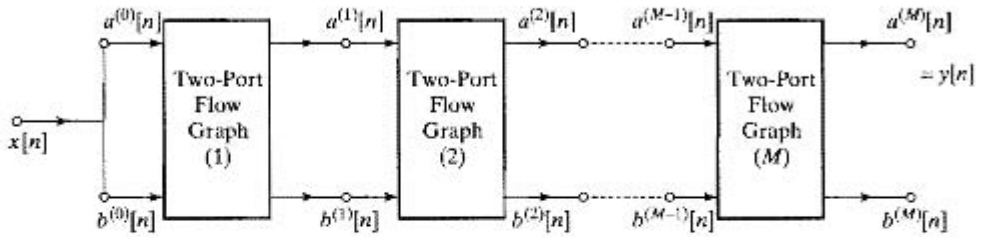


Figure 6.36 Cascade connection of M basic building block sections.

depending on the definition of the basic building block, we will limit our attention to the particular choice in Figure 6.35(b), which leads to a widely used class of FIR and IIR filter structures known as *lattice filters*.

6.6.1 FIR Lattice Filters

If the basic butterfly-shaped two-port building block in Figure 6.35(b) is used in the cascade of Figure 6.36, we obtain a flow graph like the one shown in Figure 6.37, whose lattice shape motivates the name *lattice filter*. The coefficients $k_1, k_2, \ldots, k_M$, are referred to generally as the k-parameters of the lattice structure. In Chapter 11, we will see that the k-parameters have a special meaning in the context of all-pole modeling of signals, and the lattice filter of Figure 6.37 is an implementation structure for a linear prediction of signal samples. In the current chapter, our focus is only on the use of lattice filters to implement FIR and all-pole IIR transfer functions.

The node variables $a^{(i)}[n]$ and $b^{(i)}[n]$ in Figure 6.37 are intermediate sequences that depend upon the input $x[n]$ through the set of difference equations

$$a^{(0)}[n] = b^{(0)}[n] = x[n] \tag{6.55a}$$

$$a^{(i)}[n] = a^{(i-1)}[n] - k_i b^{(i-1)}[n-1] \quad i = 1, 2, \ldots, M \tag{6.55b}$$

$$b^{(i)}[n] = b^{(i-1)}[n-1] - k_i a^{(i-1)}[n] \quad i = 1, 2, \ldots, M \tag{6.55c}$$

$$y[n] = a^{(M)}[n]. \tag{6.55d}$$

As we can see, the k-parameters are coefficients in this set of M coupled difference equations represented by Figure 6.37 and Eqs. (6.55a)–(6.55d). It should be clear that

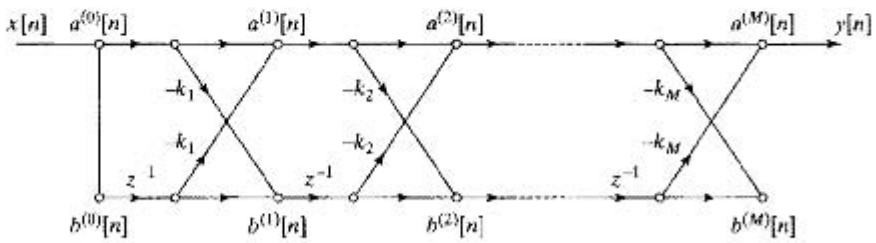


Figure 6.37 Lattice flow graph for an FIR system based on a cascade of M two-port building blocks of Figure 6.35(b).

these equations must be computed in the order shown ($i = 0, 1, \ldots, M$) since the output of stage $(i-1)$ is needed as input to stage (i), etc.

The lattice structure in Figure 6.37 is clearly an LTI system, since it is a linear signal flow graph with only delays and constant branch coefficients. Furthermore, note that there are no feedback loops, which implies that the system has a finite-duration impulse response. In fact, a straightforward argument is sufficient to show that the impulse response from the input to any internal node has finite length. Specifically, consider the impulse response from $x[n]$ to the node variable $a^{(i)}[n]$, i.e., from the input to the i^{th} upper node. It is clear that if $x[n] = \delta[n]$, then $a^{(i)}[0] = 1$ for every i, since the impulse propagates with no delay through the top branch of all the stages. All other paths to any node variable $a^{(i)}[n]$ or $b^{(i)}[n]$ pass through at least one delay, with the greatest delay being along the bottom path and then up to node variable $a^{(i)}[n]$ through the coefficient $-k_i$. This will be the last impulse that arrives at $a^{(i)}[n]$, so the impulse response will have length $i+1$ samples. All other paths to an internal node zigzag between the top and bottom of the graph, thereby passing through at least one, but not all, of the delays occurring before the outputs of section (i).

Note that in our introduction to lattice filters, $a^{(i)}[n]$ and $b^{(i)}[n]$ were used in Figure 6.37 and Eqs. (6.55a)–(6.55d) to denote the node variables of building block (i) for *any* input $x[n]$. However, for the remainder of our discussion, it is convenient to assume specifically that $x[n] = \delta[n]$ so that $a^{(i)}[n]$ and $b^{(i)}[n]$ are the resulting impulse responses at the associated nodes, and that the corresponding z-transforms $A^{(i)}(z)$ and $B^{(i)}(z)$ are the transfer functions between the input and the i^{th} nodes. Consequently, the transfer function between the input and the upper i^{th} node is

$$A^{(i)}(z) = \sum_{n=0}^{i} a^{(i)}[n]z^{-n} = 1 - \sum_{m=1}^{i} \alpha_m^{(i)} z^{-m}, \tag{6.56}$$

where in the second form, the coefficients $\alpha_m^{(i)}$ for $m \leq i$ are composed of sums of products of the coefficients k_j for $j \leq m$. As we have shown, the coefficient for the longest delay from the input to the upper node i is $\alpha_i^{(i)} = k_i$. In this notation, the impulse response from $x[n]$ to node variable $a^{(i)}[n]$ is

$$a^{(i)}[n] = \begin{cases} 1 & n = 0 \\ -\alpha_n^{(i)} & 1 \leq n \leq i \\ 0 & \text{otherwise} \end{cases} \tag{6.57}$$

Similarly, the transfer function from the input to the lower node i is denoted $B^{(i)}(z)$. Therefore, from Figure 6.35(b) or Eqs. (6.55b) and (6.55c), we see that

$$A^{(i)}(z) = A^{(i-1)}(z) - k_i z^{-1} B^{(i-1)}(z) \tag{6.58a}$$

$$B^{(i)}(z) = -k_i A^{(i-1)}(z) + z^{-1} B^{(i-1)}(z). \tag{6.58b}$$

Also, we note that at the input end $(i = 0)$

$$A_0(z) = B_0(z) = 1. \tag{6.59}$$

Using Eqs. (6.58a) and (6.58b) and starting with Eq. (6.59), we can calculate $A^{(i)}(z)$ and $B^{(i)}(z)$ recursively up to any value of i. If we continue, the pattern that emerges in the relationship between $B^{(i)}(z)$ and $A^{(i)}(z)$ is

$$B^{(i)}(z) = z^{-i} A^{(i)}(1/z) \tag{6.60a}$$

or by replacing z by $1/z$ in Eq. (6.60a) we have the equivalent relation

$$A^{(i)}(z) = z^{-i}B^{(i)}(1/z). \tag{6.60b}$$

We can verify these equivalent relationships formally by induction, i.e., by verifying that if they are true for some value $i-1$ then they will be true for i. Specifically, it is straightforward to see from Eq. (6.59) that Eqs. (6.60a) and (6.60b) are true for $i = 0$. Now note that for $i = 1$,

$$\begin{aligned} A^{(1)}(z) &= A^{(0)}(z) - k_1 z^{-1} B^{(0)}(z) = 1 - k_1 z^{-1} \\ B^{(1)}(z) &= -k_1 A^{(0)}(z) + z^{-1} B^{(0)}(z) = -k_1 + z^{-1} \\ &= z^{-1}(1 - k_1 z) \\ &= z^{-1} A^{(1)}(1/z) \end{aligned}$$

and for $i = 2$,

$$\begin{aligned} A^{(2)}(z) &= A^{(1)}(z) - k_2 z^{-1} B^{(1)}(z) = 1 - k_1 z^{-1} - k_2 z^{-2}(1 - k_1 z) \\ &= 1 - k_1(1 - k_2) z^{-1} - k_2 z^{-2} \\ B^{(2)}(z) &= -k_2 A^{(1)}(z) + z^{-1} B^{(1)}(z) = -k_2(1 - k_1 z^{-1}) + z^{-2}(1 - k_1 z) \\ &= z^{-2}(1 - k_1(1 - k_2) z - k_2 z^2) \\ &= z^{-2} A^{(2)}(1/z). \end{aligned}$$

We can prove the general result by assuming that Eq. (6.60a) and Eq. (6.60b) are true for $i-1$, and then substitute into Eq. (6.58b) to obtain

$$\begin{aligned} B^{(i)}(z) &= -k_i z^{-(i-1)} B^{(i-1)}(1/z) + z^{-1} z^{-(i-1)} A^{(i-1)}(1/z) \\ &= z^{-i} \left[A^{(i-1)}(1/z) - k_i z B^{(i-1)}(1/z) \right]. \end{aligned}$$

From Eq. (6.58a) it follows that the term in brackets is $A^{(i)}(1/z)$, so that in general,

$$B^{(i)}(z) = z^{-i} A^{(i)}(1/z),$$

as in Eq. (6.60a). Thus, we have shown that Eqs. (6.60a) and (6.60b) hold for any $i \geq 0$.

As indicated earlier, the transfer functions $A^{(i)}(z)$ and $B^{(i)}(z)$ can be computed recursively using Eq. (6.58a) and (6.58b). These transfer functions are i^{th}-order polynomials, and it is particularly useful to obtain a direct relationship among the coefficients of the polynomials. Toward this end, the right side of Eq. (6.57) defines the coefficients of $A^{(i)}(z)$ to be $-\alpha_m^{(i)}$, for $m = 1, 2, \ldots, i$ with the leading coefficient equal to one; i.e., as in Eq. (6.56),

$$A^{(i)}(z) = 1 - \sum_{m=1}^{i} \alpha_m^{(i)} z^{-m}, \tag{6.61}$$

and similarly,

$$A^{(i-1)}(z) = 1 - \sum_{m=1}^{i-1} \alpha_m^{(i-1)} z^{-m}. \tag{6.62}$$

To obtain a direct recursive relationship for the coefficients $\alpha_m^{(i)}$ in terms of $\alpha_m^{(i-1)}$ and k_i, we combine Eqs. (6.60a) and (6.62) from which it follows that

$$B^{(i-1)}(z) = z^{-(i-1)}A^{(i-1)}(1/z) = z^{-(i-1)}\left[1 - \sum_{m=1}^{i-1} \alpha_m^{(i-1)} z^{+m}\right]. \tag{6.63}$$

Substituting Eqs. (6.62) and (6.63) into Eq. (6.58a), $A^{(i)}(z)$ can also be expressed as

$$A^{(i)}(z) = \left(1 - \sum_{m=1}^{i-1} \alpha_m^{(i-1)} z^{-m}\right) - k_i z^{-1}\left(z^{-(i-1)}\left[1 - \sum_{m=1}^{i-1} \alpha_m^{(i-1)} z^{+m}\right]\right). \tag{6.64}$$

Re-indexing the second summation by reversing the ordering of the terms (i.e., replacing m by $i - m$ and re-summing) and combining terms in Eq. (6.64) leads to

$$A^{(i)}(z) = 1 - \sum_{m=1}^{i-1}\left[\alpha_m^{(i-1)} - k_i \alpha_{i-m}^{(i-1)}\right] z^{-m} - k_i z^{-i}, \tag{6.65}$$

where we see that, as indicated earlier, the coefficient of z^{-i} is $-k_i$. Comparing Eqs. (6.65) and (6.61) shows that

$$\alpha_m^{(i)} = \left[\alpha_m^{(i-1)} - k_i \alpha_{i-m}^{(i-1)}\right] \quad m = 1, \ldots, i-1 \tag{6.66a}$$

$$\alpha_i^{(i)} = k_i. \tag{6.66b}$$

Equations (6.66) are the desired direct recursion between the coefficients of $A^{(i)}(z)$ and the coefficients of $A^{(i-1)}(z)$. These equations, together with Eq. (6.60a) also determine the transfer function $B^{(i)}(z)$.

The recursion of Eqs. (6.66) can also be expressed compactly in matrix form. We denote by $\boldsymbol{\alpha}_{i-1}$ the vector of transfer function coefficients for $A^{(i-1)}(z)$ and by $\check{\boldsymbol{\alpha}}_{i-1}$ these coefficients in reverse order, i.e.,

$$\boldsymbol{\alpha}_{i-1} = \left[\alpha_1^{(i-1)}\ \alpha_2^{(i-1)}\ \cdots\ \alpha_{i-1}^{(i-1)}\right]^T$$

and

$$\check{\boldsymbol{\alpha}}_{i-1} = \left[\alpha_{i-1}^{(i-1)}\ \alpha_{i-2}^{(i-1)}\ \cdots\ \alpha_1^{(i-1)}\right]^T.$$

Then Eqs. (6.66) can be expressed as the matrix equation

$$\boldsymbol{\alpha}_i = \begin{bmatrix} \boldsymbol{\alpha}_{i-1} \\ \cdots\cdots \\ 0 \end{bmatrix} - k_i \begin{bmatrix} \check{\boldsymbol{\alpha}}_{i-1} \\ \cdots\cdots \\ -1 \end{bmatrix}. \tag{6.67}$$

The recursion in Eqs. (6.66) or Eqs. (6.67) is the basis for an algorithm for analyzing an FIR lattice structure to obtain its transfer function. We begin with the flow graph specified as in Figure 6.37 by the set of k-parameters $\{k_1, k_2, \ldots, k_M\}$. Then we can use Eqs. (6.66) recursively to compute the transfer functions of successively higher-order FIR filters until we come to end of the cascade giving us

$$A(z) = 1 - \sum_{m=1}^{M} \alpha_m z^{-m} = \frac{Y(z)}{X(z)}, \tag{6.68a}$$

k-Parameters-to-Coefficients Algorithm

Given $k_1, k_2, \ldots, k_M$
for $i = 1, 2, \ldots, M$
  $\alpha_i^{(i)} = k_i$ Eq. (6.66b)
  if $i > 1$ then for $j = 1, 2, \ldots, i-1$
    $\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)}$ Eq. (6.66a)
  end
end
$\alpha_j = \alpha_j^{(M)} \quad j = 1, 2, \ldots, M$ Eq. (6.68b)

Figure 6.38 Algorithm for converting from k-parameters to FIR filter coefficients.

where

$$\alpha_m = \alpha_m^{(M)} \quad m = 1, 2, \ldots, M. \tag{6.68b}$$

The steps of this algorithm are represented in Figure 6.38.

It is also of interest to obtain the k-parameters in the FIR lattice structure that realize a given desired transfer function from input $x[n]$ to the output $y[n] = a^{(M)}[n]$; i.e., we wish to go from $A(z)$ specified as a polynomial by Eqs. (6.68a) and (6.68b) to the set of k-parameters for the lattice structure in Figure 6.37. This can be done by reversing the recursion of Eqs. (6.66) or (6.67) to obtain successively the transfer function $A^{(i-1)}(z)$ in terms of $A^{(i)}(z)$ for $i = M, M-1, \ldots, 2$. The k-parameters are obtained as a by-product of this recursion.

Specifically, we assume that the coefficients $\alpha_m^{(M)} = \alpha_m$ for $m = 1, \ldots, M$ are specified and we want to obtain the k-parameters $k_1, \ldots, k_M$ to realize this transfer function in lattice form. We start with the last stage of the FIR lattice, i.e., with $i = M$. From Eq. (6.66b),

$$k_M = \alpha_M^{(M)} = \alpha_M \tag{6.69}$$

with $A^{(M)}(z)$ defined in terms of the specified coefficients as

$$A^{(M)}(z) = 1 - \sum_{m=1}^{M} \alpha_m^{(M)} z^{-m} = 1 - \sum_{m=1}^{M} \alpha_m z^{-m}. \tag{6.70}$$

Inverting Eqs. (6.66) or equivalently Eq. (6.67), with $i = M$ and $k_M = \alpha_M^{(M)}$ then determines $\boldsymbol{\alpha}_{M-1}$, the vector of transform coefficients at the next to last stage $i = M-1$. This process is repeated until we reach $A^{(1)}(z)$.

To obtain a general recursion formula for $\alpha_m^{(i-1)}$ in terms of $\alpha_m^{(i)}$ from Eq. (6.66a) note that $\alpha_{i-m}^{(i-1)}$ must be eliminated. To do this, replace m by $i-m$ in Eq. (6.66a) and multiply both sides of the resulting equation by k_i thereby obtaining

$$k_i \alpha_{i-m}^{(i)} = k_i \alpha_{i-m}^{(i-1)} - k_i^2 \alpha_m^{(i-1)}.$$

Adding this equation to Eq. (6.66a) results in

$$\alpha_m^{(i)} + k_i \alpha_{i-m}^{(i)} = \alpha_m^{(i-1)} - k_i^2 \alpha_m^{(i-1)}$$

from which it follows that

$$\alpha_m^{(i-1)} = \frac{\alpha_m^{(i)} + k_i \alpha_{i-m}^{(i)}}{1 - k_i^2} \quad m = 1, 2, \ldots, i-1. \tag{6.71a}$$

With $\alpha_m^{(i-1)}$ calculated for $m = 1, 2, \ldots, i-1$ we note from Eq. (6.66b) that

$$k_{i-1} = \alpha_{i-1}^{(i-1)}. \tag{6.71b}$$

Thus, starting with $\alpha_m^{(M)} = \alpha_m$, $m = 1, 2, \ldots M$ we can use Eqs. (6.71a) and (6.71b) to compute $\alpha_m^{(M-1)}$, for $m = 1, 2, \ldots, M-1$ and k_{M-1}, and then repeat this process recursively to obtain all of the transfer functions $A^{(i)}(z)$ and, as a by-product, all of the k-parameters needed for the lattice structure. The algorithm is represented in Figure 6.39.

Coefficients-to-k-Parameters Algorithm

Given $\alpha_j^{(M)} = \alpha_j \quad j = 1, 2, \ldots, M$

$k_M = \alpha_M^{(M)}$ Eq. (6.69)

for $i = M, M-1, \ldots, 2$

  for $j = 1, 2, \ldots, i-1$

$$\alpha_j^{(i-1)} = \frac{\alpha_j^{(i)} + k_i \alpha_{i-j}^{(i)}}{1 - k_i^2} \quad \text{Eq. (6.71a)}$$

  end

  $k_{i-1} = \alpha_{i-1}^{(i-1)}$ Eq. (6.71b)

end

Figure 6.39 Algorithm for converting from FIR filter coefficients to k-parameters.

Example 6.9 k-Parameters for a 3rd-Order FIR System

Consider the FIR system shown in Figure 6.40a whose system function is

$$A(z) = 1 - 0.9z^{-1} + 0.64z^{-2} - 0.576z^{-3}.$$

Consequently, $M = 3$ and the coefficients $\alpha_k^{(3)}$ in Eq. (6.70), are

$$\alpha_1^{(3)} = 0.9 \quad \alpha_2^{(3)} = 0.64 \quad \alpha_3^{(3)} = 0.576.$$

We begin by observing that $k_3 = \alpha_3^{(3)} = 0.576$.

Next we want to calculate the coefficients for transfer function $A^{(2)}(z)$ using Eq. (6.71a). Specifically, applying Eq. (6.71a), we obtain (rounded to three decimal places):

$$\alpha_1^{(2)} = \frac{\alpha_1^{(3)} + k_3 \alpha_2^{(3)}}{1 - k_3^2} = 0.795$$

$$\alpha_2^{(2)} = \frac{\alpha_2^{(3)} + k_3 \alpha_1^{(3)}}{1 - k_3^2} = -0.182$$

From Eq. (6.71b) we then identify $k_2 = \alpha_2^{(2)} = -0.182$

To obtain $A^{(1)}(z)$ we again apply Eq. (6.71a) obtaining

$$\alpha_1^{(1)} = \frac{\alpha_1^{(2)} + k_2 \alpha_1^{(2)}}{1 - k_2^2} = 0.673.$$

We then identify $k_1 = \alpha_1^{(1)} = 0.673$. The resulting lattice structure is shown in Figure 6.40b.

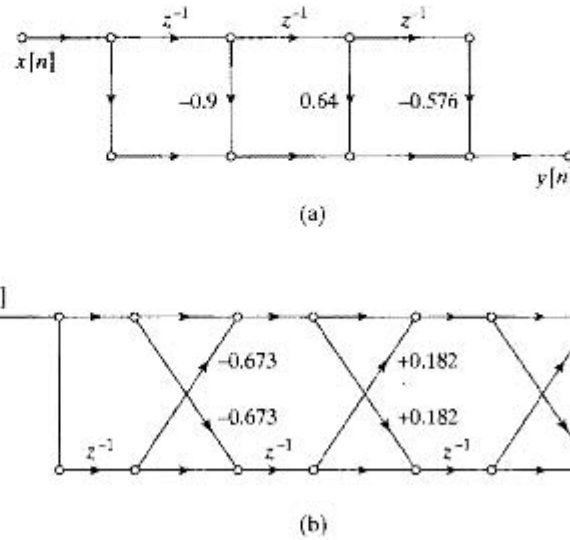


Figure 6.40 Flow graphs for example. (a) Direct form. (b) Lattice form (coefficients rounded).

6.6.2 All-Pole Lattice Structure

A lattice structure for implementing the all-pole system function $H(z) = 1/A(z)$ can be developed from the FIR lattice of the previous section by recognizing that $H(z)$ is the inverse filter for the FIR system function $A(z)$. To derive the all-pole lattice structure, assume that we are given $y[n] = a^{(M)}[n]$, and we wish to compute the input $a^{(0)}[n] = x[n]$. This can be done by working from right to left to invert the computations in Figure 6.37. More specifically, if we solve Eq. (6.58a) for $A^{(i-1)}(z)$ in terms of $A^{(i)}(z)$ and $B^{(i-1)}(z)$ and leave Eq. (6.58b) as it is, we obtain the pair of equations

$$A^{(i-1)}(z) = A^{(i)}(z) + k_i z^{-1} B^{(i-1)}(z) \tag{6.72a}$$

$$B^{(i)}(z) = -k_i A^{(i-1)}(z) + z^{-1} B^{(i-1)}(z), \tag{6.72b}$$

which have the flow graph representation shown in Figure 6.41. Note that in this case, the signal flow is from i to $i-1$ along the top of the diagram and from $i-1$ to i along the bottom. Successive connection of M stages of Figure 6.41 with the appropriate k_i in each section takes the input $a^{(M)}[n]$ to the output $a^{(0)}[n]$ as shown in the flow graph of Figure 6.42. Finally, the condition $x[n] = a^{(0)}[n] = b^{(0)}[n]$ at the terminals of the last stage in Figure 6.42 causes a feedback connection that provides the sequences $b^{(i)}[n]$ that propagate in the reverse direction. Such feedback is, of course, necessary for an IIR system.

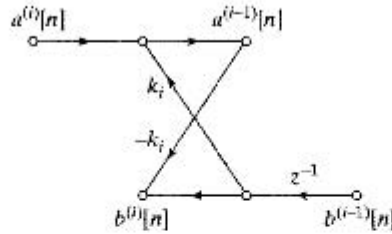


Figure 6.41 One stage of computation for an all-pole lattice system.

The set of difference equations represented by Figure 6.42 is[4]

$$a^{(M)}[n] = y[n] \tag{6.73a}$$

$$a^{(i-1)}[n] = a^{(i)}[n] + k_i b^{(i-1)}[n-1] \quad i = M, M-1, \ldots, 1 \tag{6.73b}$$

$$b^{(i)}[n] = b^{(i-1)}[n-1] - k_i a^{(i-1)}[n] \quad i = M, M-1, \ldots, 1 \tag{6.73c}$$

$$x[n] = a^{(0)}[n] = b^{(0)}[n]. \tag{6.73d}$$

Because of the feedback inherent in Figure 6.42 and these corresponding equations, initial conditions must be specified for all of the node variables associated with delays. Typically, we would specify $b^{(i)}[-1] = 0$ for initial rest conditions. Then, if Eq. (6.73b) is evaluated first, $a^{(i-1)}[n]$ will be available at times $n \geq 0$ for evaluation of Eq. (6.73c) with the values of $b^{(i-1)}[n-1]$ having been provided by the previous iteration.

Now we can state that all the analysis of Section 6.6.1 applies to the all-pole lattice system of Figure 6.42. If we wish to obtain a lattice implementation of an all-pole system with system function $H(z) = 1/A(z)$, we can simply use the algorithms in Figures 6.39 and 6.38 to obtain k-parameters from denominator polynomial coefficients or vice-versa.

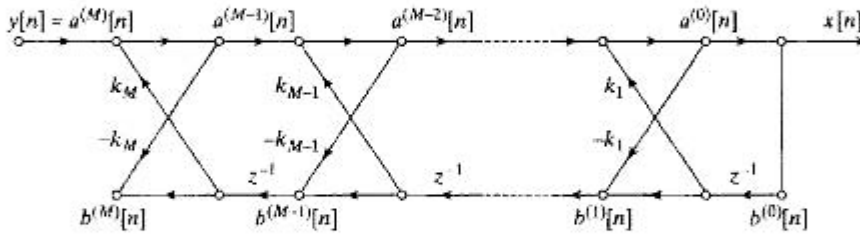


Figure 6.42 All-pole lattice system.

[4]Note that by basing our derivation of the all-pole lattice on the FIR lattice in Figure 6.37, we have ended up with the input denoted $y[n]$ and the output $x[n]$ in opposition to our normal convention. This labeling is, of course, arbitrary once the derivation has been completed.

Example 6.10 Lattice Implementation of an IIR System

As an example of an IIR system, consider the system function

$$H(z) = \frac{1}{1 - 0.9z^{-1} + 0.64z^{-2} - 0.576z^{-3}} \tag{6.74a}$$

$$= \frac{1}{(1 - 0.8jz^{-1})(1 + 0.8jz^{-1})(1 - 0.9z^{-1})} \tag{6.74b}$$

which is the inverse system for the system in Example 6.9. Figure 6.43(a) shows the direct form realization of this system, whereas Figure 6.43(b) shows the equivalent IIR lattice system using the k-parameters computed as in Example 6.9. Note that the lattice structure has the same number of delays (memory registers) as the direct form structure. However, the number of multipliers is twice the number of the direct form. This is obviously true for any order M.

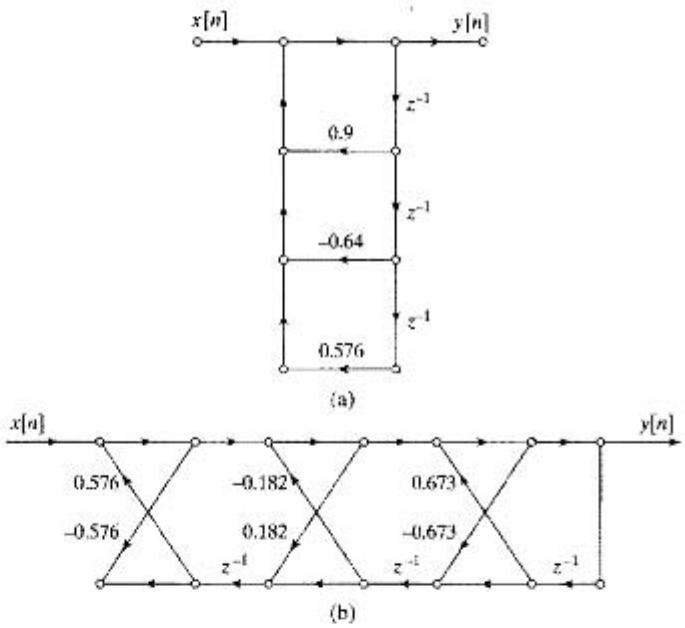


Figure 6.43 Signal flow graph of IIR filter; (a) direct form, (b) lattice form.

Since the lattice structure of Figure 6.42 is an IIR system, we must be concerned about its stability. We will see in Chapter 13 that a necessary and sufficient condition for all the zeros of a polynomial $A(z)$ to be inside the unit circle is $|k_i| < 1$, $i = 1, 2, \ldots, M$. (See Markel and Gray, 1976.) Example 6.10 confirms this fact since, as shown in Eq. (6.74b) the poles of $H(z)$ (zeros of $A(z)$) are located inside the unit circle of the z-plane and all the k-parameters have magnitude less than one. For IIR systems, the guarantee of stability inherent in the condition $|k_i| < 1$ is particularly important. Even though the lattice structure requires twice the number of multiplications per output sample as the direct form, it is insensitive to quantization of the k-parameters. This property accounts for the popularity of lattice filters in speech synthesis applications. (See Quatieri, 2002 and Rabiner and Schafer, 1978.)

6.6.3 Generalization of Lattice Systems

We have shown that FIR systems and all-pole IIR systems have a lattice structure representation. When the system function has both poles and zeros, it is still possible to find a lattice structure based upon a modification of the all-pole structure of Figure 6.42. The derivation will not be provided here (See Gray and Markel, 1973, 1976.), but it is outlined in Problem 11.27.

6.7 OVERVIEW OF FINITE-PRECISION NUMERICAL EFFECTS

We have seen that a particular LTI discrete-time system can be implemented by a variety of computational structures. One motivation for considering alternatives to the simple direct form structures is that different structures that are equivalent for infinite-precision arithmetic may behave differently when implemented with finite numerical precision. In this section, we give a brief introduction to the major numerical problems that arise in implementing discrete-time systems. A more detailed analysis of these finite word-length effects is given in Sections 6.8–6.10.

6.7.1 Number Representations

In theoretical analyses of discrete-time systems, we generally assume that signal values and system coefficients are represented in the real-number system. However, with analog discrete-time systems, the limited precision of the components of a circuit makes it difficult to realize coefficients exactly. Similarly, when implementing digital signal-processing systems, we must represent signals and coefficients in some digital number system that must always have finite precision.

The problem of finite numerical precision has already been discussed in Section 4.8.2 in the context of A/D conversion. We showed there that the output samples from an A/D converter are quantized and thus can be represented by fixed-point binary numbers. For compactness and simplicity in implementing arithmetic, one of the bits of the binary number is assumed to indicate the algebraic sign of the number. Formats such as *sign and magnitude, one's complement,* and *two's complement* are possible, but two's complement is most common.[5] A real number can be represented with infinite precision in two's-complement form as

$$x = X_m \left(-b_0 + \sum_{i=1}^{\infty} b_i 2^{-i} \right), \tag{6.75}$$

where X_m is an arbitrary scale factor and the b_is are either 0 or 1. The quantity b_0 is referred to as the *sign bit*. If $b_0 = 0$, then $0 \leq x \leq X_m$, and if $b_0 = 1$, then $-X_m \leq x < 0$. Thus, any real number whose magnitude is less than or equal to X_m can be represented by Eq. (6.75). An arbitrary real number x would require an infinite number of bits for its exact binary representation. As we saw in the case of A/D conversion, if we use only

[5] A detailed description of binary number systems and corresponding arithmetic is given by Knuth (1997).

a finite number of bits $(B+1)$, then the representation of Eq. (6.75) must be modified to

$$\hat{x} = Q_B[x] = X_m\left(-b_0 + \sum_{i=1}^{B} b_i 2^{-i}\right) = X_m \hat{x}_B. \tag{6.76}$$

The resulting binary representation is quantized, so that the smallest difference between numbers is

$$\Delta = X_m 2^{-B}. \tag{6.77}$$

In this case, the quantized numbers are in the range $-X_m \leq \hat{x} < X_m$. The fractional part of $\hat{x}$ can be represented with the positional notation

$$\hat{x}_B = b_0 \diamond b_1 b_2 b_3 \cdots b_B, \tag{6.78}$$

where $\diamond$ represents the binary point.

The operation of quantizing a number to $(B+1)$ bits can be implemented by rounding or by truncation, but in either case, quantization is a nonlinear memoryless operation. Figures 6.44(a) and 6.44(b) show the input–output relation for two's-complement rounding and truncation, respectively, for the case $B = 2$. In considering the effects of quantization, we often define the *quantization error* as

$$e = Q_B[x] - x. \tag{6.79}$$

For the case of two's-complement rounding, $-\Delta/2 < e \leq \Delta/2$, and for two's-complement truncation, $-\Delta < e \leq 0$.[6]

If a number is larger than X_m (a situation called an overflow), we must implement some method of determining the quantized result. In the two's-complement arithmetic system, this need arises when we add two numbers whose sum is greater than X_m. For example, consider the 4-bit two's-complement number 0111, which in decimal form is 7. If we add the number 0001, the carry propagates all the way to the sign bit, so that the result is 1000, which in decimal form is -8. Thus, the resulting error can be very large when overflow occurs. Figure 6.45(a) shows the two's-complement rounding quantizer, including the effect of regular two's-complement arithmetic overflow. An alternative, which is called *saturation overflow* or *clipping*, is shown in Figure 6.45(b). This method of handling overflow is generally implemented for A/D conversion, and it sometimes is implemented in specialized DSP microprocessors for the addition of two's-complement numbers. With saturation overflow, the size of the error does not increase abruptly when overflow occurs; however, a disadvantage of the method is that it voids the following interesting and useful property of two's-complement arithmetic: If several two's-complement numbers whose sum would not overflow are added, then the result of two's-complement accumulation of these numbers is correct, even though intermediate sums might overflow.

Both quantization and overflow introduce errors in digital representations of numbers. Unfortunately, to minimize overflow while keeping the number of bits the same, we must increase X_m and thus increase the size of quantization errors proportionately. Hence, to simultaneously achieve wider dynamic range and lower quantization error, we must increase the number of bits in the binary representation.

[6]Note that Eq. (6.76) also represents the result of rounding or truncating any (B_1+1)-bit binary representation, where $B_1 > B$. In this case Δ would be replaced by $(\Delta - X_m 2^{-B_1})$ in the bounds on the size of the quantization error.

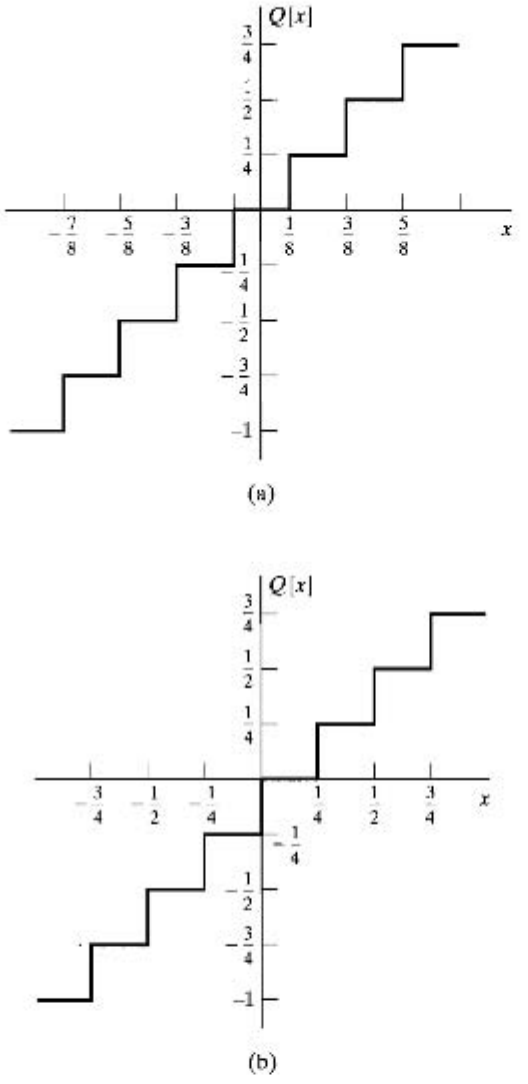


Figure 6.44 Nonlinear relationships representing two's-complement (a) rounding and (b) truncation for $B = 2$.

So far, we have simply stated that the quantity X_m is an arbitrary scale factor; however, this factor has several useful interpretations. In A/D conversion, we considered X_m to be the full-scale amplitude of the A/D converter. In this case, X_m would probably represent a voltage or current in the analog part of the system. Thus, X_m serves as a calibration constant for relating binary numbers in the range $-1 \leq \hat{x}_B < 1$ to analog signal amplitudes.

In digital signal-processing implementations, it is common to assume that all signal variables and all coefficients are binary fractions. Thus, if we multiply a $(B+1)$-bit signal variable by a $(B+1)$-bit coefficient, the result is a $(2B+1)$-bit fraction that can be conveniently reduced to $(B+1)$ bits by rounding or truncating the least significant bits. With this convention, the quantity X_m can be thought of as a scale factor that allows the

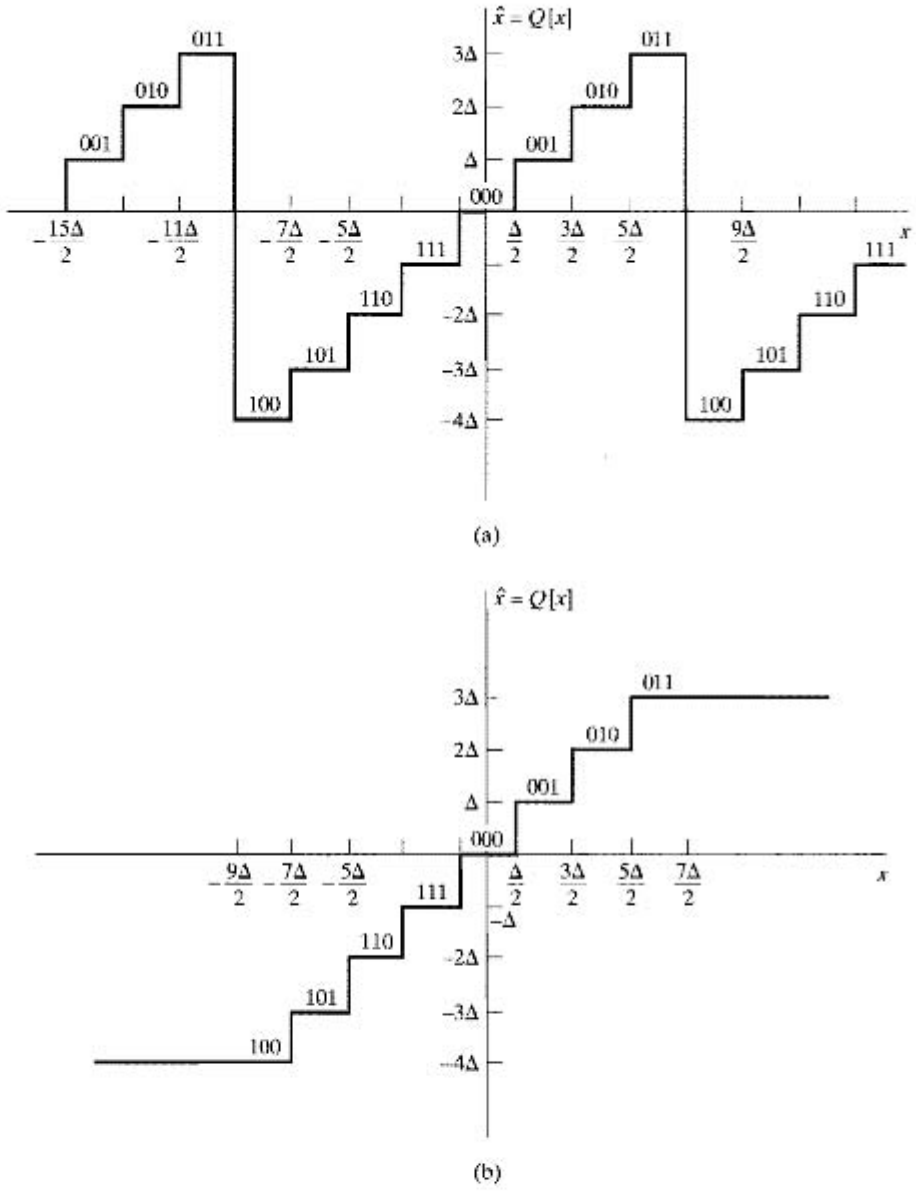


Figure 6.45 Two's-complement rounding. (a) Natural overflow. (b) Saturation.

representation of numbers that are greater than unity in magnitude. For example, in fixed-point computations, it is common to assume that each binary number has a scale factor of the form $X_m = 2^c$. Accordingly, a value $c = 2$ implies that the binary point is actually located between b_2 and b_3 of the binary word in Eq. (6.78). Often, this scale factor is not explicitly represented; instead, it is implicit in the implementation program or hardware architecture.

Still another way of thinking about the scale factor X_m leads to the *floating-point representations*, in which the exponent c of the scale factor is called the *characteristic* and the fractional part $\hat{x}_B$ is called the *mantissa*. The characteristic and the mantissa are each represented explicitly as binary numbers in floating-point arithmetic systems. Floating-point representations provide a convenient means for maintaining both a wide dynamic range and a small quantization noise; however, quantization error manifests itself in a somewhat different way.

6.7.2 Quantization in Implementing Systems

Numerical quantization affects the implementation of LTI discrete-time systems in several ways. As a simple illustration, consider Figure 6.46(a), which shows a block diagram

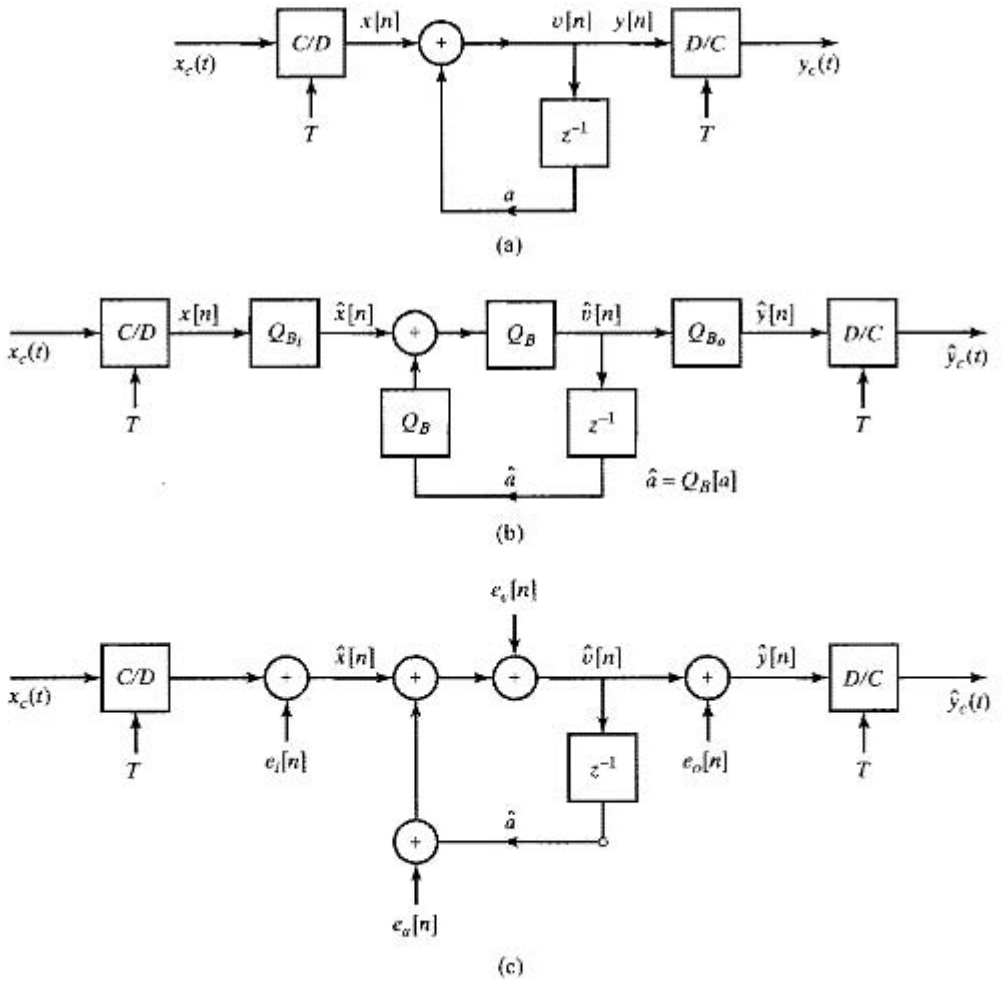


Figure 6.46 Implementation of discrete-time filtering of an analog signal. (a) Ideal system. (b) Nonlinear model. (c) Linearized model.

for a system in which a bandlimited continuous-time signal $x_c(t)$ is sampled to obtain the sequence $x[n]$, which is the input to an LTI system whose system function is

$$H(z) = \frac{1}{1 - az^{-1}}. \tag{6.80}$$

The output of this system, $y[n]$, is converted by ideal bandlimited interpolation to the bandlimited signal $y_c(t)$.

A more realistic model is shown in Figure 6.46(b). In a practical setting, sampling would be done with an A/D converter with finite precision of $(B_i + 1)$ bits. The system would be implemented with binary arithmetic of $(B + 1)$ bits. The coefficient a in Figure 6.46(a) would be represented with $(B + 1)$ bits of precision. Also, the delayed variable $\hat{v}[n-1]$ would be stored in a $(B + 1)$-bit register, and when the $(B + 1)$-bit number $\hat{v}[n-1]$ is multiplied by the $(B + 1)$-bit number $\hat{a}$, the resulting product would be $(2B + 1)$ bits in length. If we assume that a $(B + 1)$-bit adder is used, the product $\hat{a}\hat{v}[n-1]$ must be quantized (i.e., rounded or truncated) to $(B + 1)$ bits before it can be added to the $(B_i + 1)$-bit input sample $\hat{x}[n]$. When $B_i < B$, the $(B_i + 1)$ bits of the input samples can be placed anywhere in the $(B + 1)$-bit binary word with appropriate extension of the sign. Different choices correspond to different scalings of the input. The coefficient a has been quantized, so leaving aside the other quantization errors, the system response cannot in general be the same as in Figure 6.46(a). Finally, the $(B + 1)$-bit samples $\hat{v}[n]$, computed by iterating the difference equation represented by the block diagram, would be converted to an analog signal by a $(B_o + 1)$-bit D/A converter. When $B_o < B$, the output samples must be quantized further before D/A conversion.

Although the model of Figure 6.46(b) could be an accurate representation of a real system, it would be difficult to analyze. The system is nonlinear owing to the quantizers and the possibility of overflow at the adder. Also, quantization errors are introduced at many points in the system. The effects of these errors are impossible to analyze precisely, since they depend on the input signal, which we generally consider to be unknown. Thus, we are forced to adopt several different approximation approaches to simplify the analysis of such systems.

The effect of quantizing the system parameters, such as the coefficient a in Figure 6.46(a), is generally determined separately from the effect of quantization in data conversion or in implementing difference equations. That is, the ideal coefficients of a system function are replaced by their quantized values, and the resulting response functions are tested to see whether, in the absence of other quantization in the arithmetic, quantization of the filter coefficients has degraded the performance of the system to unacceptable levels. For the example of Figure 6.46, if the real number a is quantized to $(B + 1)$ bits, we must consider whether the resulting system with system function

$$\hat{H}(z) = \frac{1}{1 - \hat{a}z^{-1}} \tag{6.81}$$

is close enough to the desired system function $H(z)$ given by Eq. (6.80). Since there are only 2^{B+1} different $(B+1)$-bit binary numbers, the pole of $H(z)$ can occur only at 2^{B+1} locations on the real axis of the z-plane, and, while it is possible that $\hat{a} = a$, in most cases some deviation from the ideal response would result. This type of analysis is discussed in more general terms in Section 6.8.

The nonlinearity of the system of Figure 6.46(b) causes behavior that cannot occur in a linear system. Specifically, systems such as this can exhibit zero-input limit cycles, where the output oscillates periodically when the input becomes zero after having been nonzero. Limit cycles are caused both by quantization and by overflow. Although the analysis of such phenomena is difficult, some useful approximate results have been developed. Limit cycles are discussed briefly in Section 6.10.

If care is taken in the design of a digital implementation, we can ensure that overflow occurs only rarely and quantization errors are small. Under these conditions, the system of Figure 6.46(b) behaves very much like a linear system (with quantized coefficients) in which quantization errors are injected at the input and output and at internal points in the structure where rounding or truncation occurs. Therefore, we can replace the model of Figure 6.46(b) by the linearized model of Figure 6.46(c), where the quantizers are replaced by additive noise sources (see Gold and Rader, 1969; Jackson, 1970a, 1970b). Figure 6.46(c) is equivalent to Figure 6.46(b) if we know each of the noise sources exactly. However, as discussed in Section 4.8.3, useful results are obtained if we assume a random noise model for the quantization noise in A/D conversion. This same approach can be used in analyzing the effects of arithmetic quantization in digital implementations of linear systems. As seen in Figure 6.46(c), each noise source injects a random signal that is processed by a different part of the system, but since we assume that all parts of the system are linear, we can compute the overall effect by superposition. In Section 6.9, we illustrate this style of analysis for several important systems.

In the simple example of Figure 6.46, there is little flexibility in the choice of structure. However, for higher-order systems, we have seen that there is a wide variety of choices. Some of the structures are less sensitive to coefficient quantization than others. Similarly, because different structures have different quantization noise sources and because these noise sources are filtered in different ways by the system, we will find that structures that are theoretically equivalent sometimes perform quite differently when finite-precision arithmetic is used to implement them.

6.8 THE EFFECTS OF COEFFICIENT QUANTIZATION

LTI discrete-time systems are generally used to perform a filtering operation. Methods for designing FIR and IIR filters, which are discussed in Chapter 7, typically assume a particular form for the system function. The result of the filter design process is a system function for which we must choose an implementation structure (a set of difference equations) from an unlimited number of theoretically equivalent implementations. Although we are almost always interested in implementations that require the least hardware or software complexity, it is not always possible to base the choice of implementation structure on this criterion alone. As we will see in Section 6.9, the implementation structure determines the quantization noise generated internally in the system. Also, some structures are more sensitive than others to perturbations of the coefficients. As we pointed out in Section 6.7, the standard approach to the study of coefficient quantization and round-off noise is to treat them independently. In this section, we consider the effects of quantizing the system parameters.

6.8.1 Effects of Coefficient Quantization in IIR Systems

When the parameters of a rational system function or corresponding difference equation are quantized, the poles and zeros of the system function move to new positions in the z-plane. Equivalently, the frequency response is perturbed from its original value. If the system implementation structure is highly sensitive to perturbations of the coefficients, the resulting system may no longer meet the original design specifications, or an IIR system might even become unstable.

A detailed sensitivity analysis for the general case is complicated and usually of limited value in specific cases of digital filter implementation. Using powerful simulation tools, it is usually easy to simply quantize the coefficients of the difference equations employed in implementing the system and then compute the corresponding frequency response and compare it with the desired frequency-response function. Even though simulation of the system is generally necessary in specific cases, it is still worthwhile to consider, in general, how the system function is affected by quantization of the coefficients of the difference equations. For example, the system function representation corresponding to both direct forms (and their corresponding transposed versions) is the ratio of polynomials

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}, \tag{6.82}$$

The sets of coefficients $\{a_k\}$ and $\{b_k\}$ are the ideal infinite-precision coefficients in both direct form implementation structures (and corresponding transposed structures). If we quantize these coefficients, we obtain the system function

$$\hat{H}(z) = \frac{\sum_{k=0}^{M} \hat{b}_k z^{-k}}{1 - \sum_{k=1}^{N} \hat{a}_k z^{-k}}, \tag{6.83}$$

where $\hat{a}_k = a_k + \Delta a_k$ and $\hat{b}_k = b_k + \Delta b_k$ are the quantized coefficients that differ from the original coefficients by the quantization errors Δa_k and Δb_k.

Now consider how the roots of the denominator and numerator polynomials (the poles and zeros of $H(z)$) are affected by the errors in the coefficients. Each polynomial root is affected by *all* of the errors in the coefficients of the polynomial since each root is a function of all the coefficients of the polynomial. Thus, each pole and zero will be affected by all of the quantization errors in the denominator and numerator polynomials, respectively. More specifically, Kaiser (1966) showed that if the poles (or zeros) are tightly clustered, it is possible that small errors in the denominator (numerator) coefficients can cause large shifts of the poles (zeros) for the direct form structures. Thus, if the poles (zeros) are tightly clustered, corresponding to a narrow-bandpass filter or a narrow-bandwidth lowpass filter, then we can expect the poles of the direct form structure to be quite sensitive to quantization errors in the coefficients. Furthermore, Kaiser's

analysis showed that the larger the number of clustered poles (zeros), the greater is the sensitivity.

The cascade and parallel form system functions, which are given by Eqs. (6.30) and (6.35), respectively, consist of combinations of 2^{nd}-order direct form systems. However, in both cases, each pair of complex-conjugate poles is realized independently of all the other poles. Thus, the error in a particular pole pair is independent of its distance from the other poles of the system function. For the cascade form, the same argument holds for the zeros, since they are realized as independent 2^{nd}-order factors. Thus, the cascade form is generally much less sensitive to coefficient quantization than the equivalent direct form realization.

As seen in Eq. (6.35), the zeros of the parallel form system function are realized implicitly, through combining the quantized 2^{nd}-order sections to obtain a common denominator. Thus, a particular zero is affected by quantization errors in the numerator and denominator coefficients of *all* the 2^{nd}-order sections. However, for most practical filter designs, the parallel form is also found to be much less sensitive to coefficient quantization than the equivalent direct forms because the 2^{nd}-order subsystems are not extremely sensitive to quantization. In many practical filters, the zeros are often widely distributed around the unit circle, or in some cases they may all be located at $z = \pm 1$. In the latter situation, the zeros mainly provide much higher attenuation around frequencies $\omega = 0$ and $\omega = \pi$ than is specified, and thus, movements of zeros away from $z = \pm 1$ do not significantly degrade the performance of the system.

6.8.2 Example of Coefficient Quantization in an Elliptic Filter

As an illustration of the effect of coefficient quantization, consider the example of an IIR bandpass elliptic filter designed using approximation techniques to be discussed in Chapter 7. The filter was designed to meet the following specifications:

$$\begin{aligned} 0.99 \leq |H(e^{j\omega})| \leq 1.01, \quad & 0.3\pi \leq |\omega| \leq 0.4\pi, \\ |H(e^{j\omega})| \leq 0.01 (\text{i.e., } -40\,\text{dB}), \quad & |\omega| \leq 0.29\pi, \\ |H(e^{j\omega})| \leq 0.01 (\text{i.e., } -40\,\text{dB}), \quad & 0.41\pi \leq |\omega| \leq \pi. \end{aligned}$$

That is, the filter should approximate one in the passband, $0.3\pi \leq |\omega| \leq 0.4\pi$, and zero elsewhere in the base interval $0 \leq |\omega| \leq \pi$. As a concession to computational realizability, a transition (do not care) region of 0.01π is allowed on either side of the passband. In Chapter 7, we will see that specifications for frequency-selective filter design algorithms are often represented in this form. The MATLAB function for elliptic filter design produces the coefficients of a 12^{th}-order direct form representation of the system function of the form of Eq. (6.82), where the coefficients a_k and b_k were computed with 64-bit floating-point arithmetic and are shown in Table 6.1 with full 15-decimal-digit precision. We shall refer to this representation of the filter as "unquantized."

The frequency response $20\log_{10}|H(e^{j\omega})|$ of the unquantized filter is shown in Figure 6.47(a), which shows that the filter meets the specifications in the stopbands (at least 40 dB attenuation). Also, the solid line in Figure 6.47(b), which is a blow-up of the passband region $0.3\pi \leq |\omega| \leq 0.4\pi$ for the unquantized filter, shows that the filter also meets the specifications in the passband.

TABLE 6.1 UNQUANTIZED DIRECT-FORM COEFFICIENTS FOR A 12TH-ORDER ELLIPTIC FILTER

k	b_k	a_k
0	0.01075998066934	1.00000000000000
1	-0.05308642937079	-5.22581881365349
2	0.16220359377307	16.78472670299535
3	0.34568964826145	-36.88325765883139
4	0.57751602647909	62.39704677556246
5	-0.77113336470234	-82.65403268814103
6	0.85093484466974	88.67462886449437
7	-0.77113336470234	-76.47294840588104
8	0.57751602647909	53.41004513122380
9	-0.34568964826145	-29.20227549870331
10	0.16220359377307	12.29074563512827
11	-0.05308642937079	-3.53766014466313
12	0.01075998066934	0.62628586102551

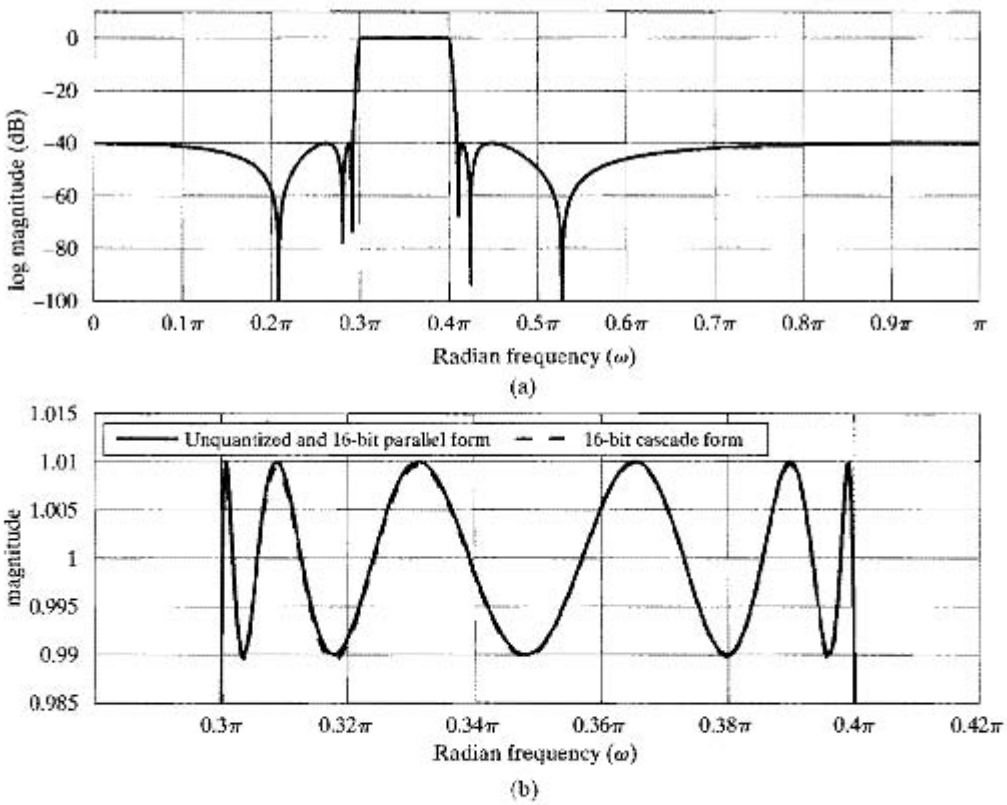


Figure 6.47 IIR coefficient quantization example. (a) Log magnitude for unquantized elliptic bandpass filter. (b) Magnitude in passband for unquantized (solid line) and 16-bit quantized cascade form (dashed line).

TABLE 6.2 ZEROS AND POLES OF UNQUANTIZED 12TH-ORDER ELLIPTIC FILTER.

k	$\lvert c_k\rvert$	$\angle c_k$	$\lvert d_k\rvert$	$\angle d_{1k}$
1	1.0	±1.65799617112574	0.92299356261936	±1.15956955465354
2	1.0	±0.65411612347125	0.92795010695052	±1.02603244134180
3	1.0	±1.33272553462313	0.96600955362927	±1.23886921536789
4	1.0	±0.87998582176421	0.97053510266510	±0.95722682653782
5	1.0	±1.28973944928129	0.99214245914242	±1.26048962626170
6	1.0	±0.91475122405407	0.99333628602629	±0.93918174153968

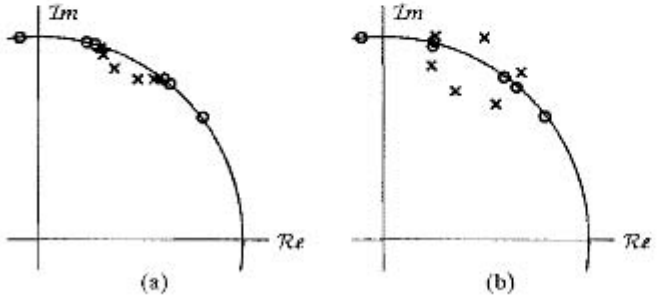


Figure 6.48 IIR coefficient quantization example. (a) Poles and zeros of $H(z)$ for unquantized coefficients. (b) Poles and zeros for 16-bit quantization of the direct form coefficients.

Factoring the numerator and denominator polynomials corresponding to the coefficients in Table 6.1 in Eq. (6.82) yields a representation

$$H(z) = \prod_{k=1}^{12} \frac{b_0(1 - c_k z^{-1})}{(1 - d_k z^{-1})}. \tag{6.84}$$

in terms of the zeros and poles, which are given in Table 6.2.

The poles and zeros of the unquantized filter that lie in the upper half of the z-plane are plotted in Figure 6.48(a). Note that the zeros are all on the unit circle, with their angular locations corresponding to the deep nulls in Figure 6.47. The zeros are strategically placed by the filter design method on either side of the passband to provide the desired stopband attenuation and sharp cutoff. Also note that the poles are clustered in the narrow passband, with two of the complex conjugate pole pairs having radii greater than 0.99. This finely tuned arrangement of zeros and poles is required to produce the narrowband sharp-cutoff bandpass filter frequency response shown in Figure 6.47(a).

A glance at the coefficients in Table 6.1 suggests that quantization of the direct form may present significant problems. Recall that with a fixed quantizer, the quantization error size is the same, regardless of the size of the number being quantized; i.e., the quantization error for coefficient $a_{12} = 0.62628586102551$ can be as large as the error for coefficient $a_6 = 88.67462886449437$, if we use the same number of bits and the same scale factor for both. For this reason, when the direct form coefficients in Table 6.1 were quantized with 16-bit precision, each coefficient was quantized independently of the other coefficients so as to maximize the accuracy for each coefficient; i.e., each 16-bit coefficient requires its own scale factor.[7] With this conservative approach, the

[7]To simplify implementation, it would be desirable, but far less accurate, if each coefficient had the same scale factor.

resulting poles and zeros are as depicted in Figure 6.48(b). Note that the zeros have shifted noticeably, but not dramatically. In particular, the closely-spaced pair of zeros toward the top of the circle has remained at about the same angle, but they have moved off of the unit circle into a group of four complex conjugate reciprocal zeros, whereas the other zeros are shifted angularly but remain on the unit circle. This constrained movement is a result of the symmetry of the coefficients of the numerator polynomial, which is preserved under quantization. However, the tightly clustered poles, having no symmetry constraints, have moved to much different positions, and, as is easily observed, some of the poles have moved outside the unit circle. Therefore, the direct form system cannot be implemented with 16-bit coefficients because it would be unstable.

On the other hand, the cascade form is much less sensitive to coefficient quantization. The cascade form of the present example can be obtained by grouping the complex conjugate pairs of poles and zeros in Eq. (6.84) and Table 6.2, to form six 2^{nd}-order factors as in

$$H(z) = \prod_{k=1}^{6} \frac{b_{0k}(1 - c_k z^{-1})(1 - c_k^* z^{-1})}{(1 - d_k z^{-1})(1 - d_k^* z^{-1})} = \prod_{k=1}^{6} \frac{b_{0k} + b_{1k} z^{-1} + b_{2k} z^{-2}}{1 - a_{1k} z^{-1} - a_{2k} z^{-2}}. \tag{6.85}$$

The zeros c_k and poles d_k and coefficients b_{ik} and a_{ik} of the cascade form can be computed with 64-bit floating-point accuracy so these coefficients can still considered to be unquantized. Table 6.3 gives the coefficients of the six 2^{nd}-order sections (as defined in Eq. (6.85). The pairing and ordering of the poles and zeros follows a procedure to be discussed in Section 6.9.3.

TABLE 6.3 UNQUANTIZED CASCADE-FORM COEFFICIENTS FOR A 12TH-ORDER ELLIPTIC FILTER

k	a_{1k}	a_{2k}	b_{0k}	b_{1k}	b_{2k}
1	0.737904	-0.851917	0.137493	0.023948	0.137493
2	0.961757	-0.861091	0.281558	-0.446881	0.281558
3	0.629578	-0.933174	0.545323	-0.257205	0.545323
4	1.117648	-0.941938	0.706400	-0.900183	0.706400
5	0.605903	-0.984347	0.769509	-0.426879	0.769509
6	1.173028	-0.986717	0.937657	-1.143918	0.937657

To illustrate how coefficients are quantized and represented as fixed-point numbers, the coefficients in Table 6.3 were quantized to 16-bit accuracy. The resulting coefficients are presented in Table 6.4. The fixed-point coefficients are shown as a decimal integer times a power-of-2 scale factor. The binary representation would be obtained by converting the decimal integer to a binary number. In a fixed-point implementation, the scale factor would be represented only implicitly in the data shifts that would be necessary to line up the binary points of products prior to their addition to other products. Notice that binary points of the coefficients are not all in the same location. For example, all the coefficients with scale factor 2^{-15} have their binary points between the sign bit, b_0, and the highest fractional bit, b_1, as shown in Eq. (6.78). However, numbers whose magnitudes do not exceed 0.5, such as the coefficient b_{02}, can be shifted left by one or more bit positions.[8] Thus, the binary point for b_{02} is actually to the left of the

[8]The use of different binary point locations retains greater accuracy in the coefficients, but it complicates the programming or system architecture.

sign bit as if the word length is extended to 17 bits. On the other hand, numbers whose magnitudes exceed 1 but are less than 2, such as a_{16}, must have their binary points moved one position to the right, i.e., between b_1 and b_2 in Eq. (6.78).

TABLE 6.4 SIXTEEN-BIT QUANTIZED CASCADE-FORM COEFFICIENTS FOR A 12TH-ORDER ELLIPTIC FILTER

k	a_{1k}	a_{2k}	b_{0k}	b_{1k}	b_{2k}
1	24196×2^{-15}	-27880×2^{-15}	17805×2^{-17}	3443×2^{-17}	17805×2^{-17}
2	31470×2^{-15}	-28180×2^{-15}	18278×2^{-16}	-29131×2^{-16}	18278×2^{-16}
3	20626×2^{-15}	-30522×2^{-15}	17556×2^{-15}	-8167×2^{-15}	17556×2^{-15}
4	18292×2^{-14}	-30816×2^{-15}	22854×2^{-15}	-29214×2^{-15}	22854×2^{-15}
5	19831×2^{-15}	-32234×2^{-15}	25333×2^{-15}	-13957×2^{-15}	25333×2^{-15}
6	19220×2^{-14}	-32315×2^{-15}	15039×2^{-14}	-18387×2^{-14}	15039×2^{-14}

The dashed line in Figure 6.47(b) shows the magnitude response in the passband for the quantized cascade form implementation. The frequency response is only slightly degraded in the passband region and negligibly in the stopband.

To obtain other equivalent structures, the cascade form system function must be rearranged into a different form. For example, if a parallel form structure is determined (by partial fraction expansion of the unquantized system function), and the resulting coefficients are quantized to 16 bits as before, the frequency response in the passband is so close to the unquantized frequency response that the difference is not observable in Figure 6.47(a) and barely observable in Figure 6.47(b).

The example just discussed illustrates the robustness of the cascade and parallel forms to the effects of coefficient quantization, and it also illustrates the extreme sensitivity of the direct forms for high-order filters. Because of this sensitivity, the direct forms are rarely used for implementing anything other than 2nd-order systems.[9] Since the cascade and parallel forms can be configured to require the same amount of memory and the same or only slightly more computation as the canonic direct form, these modular structures are the most commonly used. More complex structures such as lattice structures may be more robust for very short word lengths, but they require significantly more computation for systems of the same order.

6.8.3 Poles of Quantized 2nd-Order Sections

Even for the 2nd-order systems that are used to implement the cascade and parallel forms, there remains some flexibility to improve the robustness to coefficient quantization. Consider a complex-conjugate pole pair implemented using the direct form, as in Figure 6.49. With infinite-precision coefficients, this flow graph has poles at $z = re^{j\theta}$ and $z = re^{-j\theta}$. However, if the coefficients $2r\cos\theta$ and $-r^2$ are quantized, only a finite number of different pole locations is possible. The poles must lie on a grid in the z-plane defined by the intersection of concentric circles (corresponding to the quantization of r^2) and vertical lines (corresponding to the quantization of $2r\cos\theta$). Such a grid is

[9] An exception is in speech synthesis, where systems of 10th-order and higher are routinely implemented using the direct form. This is possible because in speech synthesis the poles of the system function are widely separated (see Rabiner and Schafer, 1978).

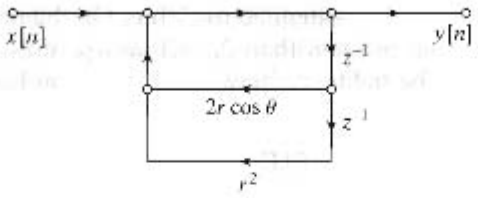
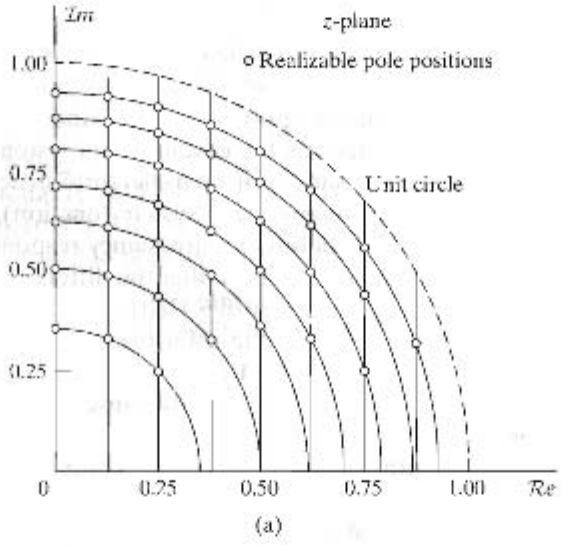
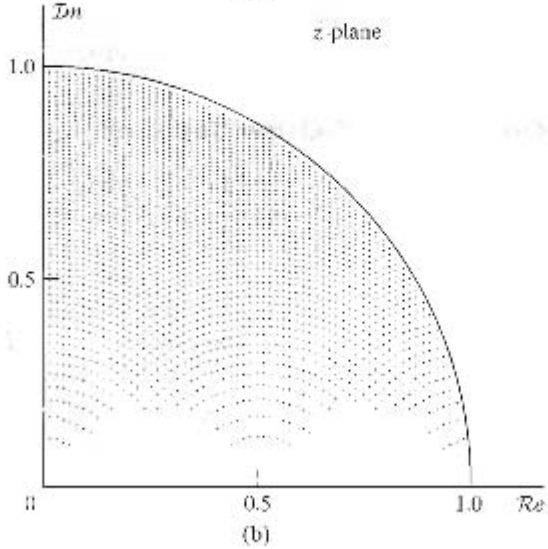


Figure 6.49 Direct form implementation of a complex-conjugate pole pair.

illustrated in Figure 6.50(a) for 4-bit quantization (3 bits plus 1 bit for the sign); i.e., r^2 is restricted to seven positive values and zero, whereas $2r\cos\theta$ is restricted to seven positive values, eight negative values, and zero. Figure 6.50(b) shows a denser grid obtained with 7-bit quantization (6 bits plus 1 bit for the sign). The plots of Figure 6.50 are, of course, symmetrically mirrored into each of the other quadrants of the z-plane.



(a)



(b)

Figure 6.50 Pole-locations for the 2nd-order IIR direct form system of Figure 6.49. (a) Four-bit quantization of coefficients. (b) Seven-bit quantization.

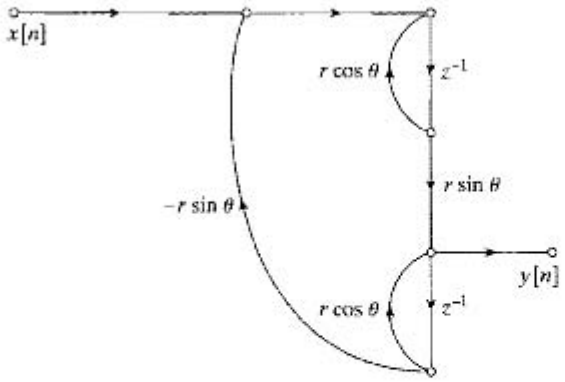


Figure 6.51 Coupled form implementation of a complex-conjugate pole pair.

Notice that for the direct form, the grid is rather sparse around the real axis. Thus, poles located around $\theta = 0$ or $\theta = \pi$ may be shifted more than those around $\theta = \pi/2$. Of course, it is always possible that the infinite-precision pole location is very close to one of the allowed quantized poles. In this case, quantization causes no problem whatsoever, but in general, quantization can be expected to degrade performance.

An alternative 2nd-order structure for realizing poles at $z = re^{j\theta}$ and $z = re^{-j\theta}$ is shown in Figure 6.51. This structure is referred to as the *coupled form* for the 2nd-order system (see Rader and Gold, 1967). It is easily verified that the systems of Figures 6.49 and 6.51 have the same poles for infinite-precision coefficients. To implement the system of Figure 6.51, we must quantize $r\cos\theta$ and $r\sin\theta$. Since these quantities are the real and imaginary parts, respectively, of the pole locations, the quantized pole locations are at intersections of evenly spaced horizontal and vertical lines in the z-plane. Figures 6.52(a) and 6.52(b) show the possible pole locations for 4-bit and 7-bit quantization, respectively. In this case, the density of pole locations is uniform throughout the interior of the unit circle. Twice as many constant multipliers are required to achieve this more uniform density. In some situations, the extra computation might be justified to achieve more accurate pole location with reduced word length.

6.8.4 Effects of Coefficient Quantization in FIR Systems

For FIR systems, we need only be concerned with the locations of the zeros of the system function, since, for causal FIR systems, all the poles are at $z = 0$. Although we have just seen that the direct form structure should be avoided for high-order IIR systems, it turns out that the direct form structure is commonly used for FIR systems. To understand why this is so, we express the system function for a direct form FIR system in the form

$$H(z) = \sum_{n=0}^{M} h[n]z^{-n}. \tag{6.86}$$

Now, suppose that the coefficients $\{h[n]\}$ are quantized, resulting in a new set of coefficients $\{\hat{h}[n] = h[n] + \Delta h[n]\}$. The system function for the quantized system is then

$$\hat{H}(z) = \sum_{n=0}^{M} \hat{h}[n]z^{-n} = H(z) + \Delta H(z), \tag{6.87}$$

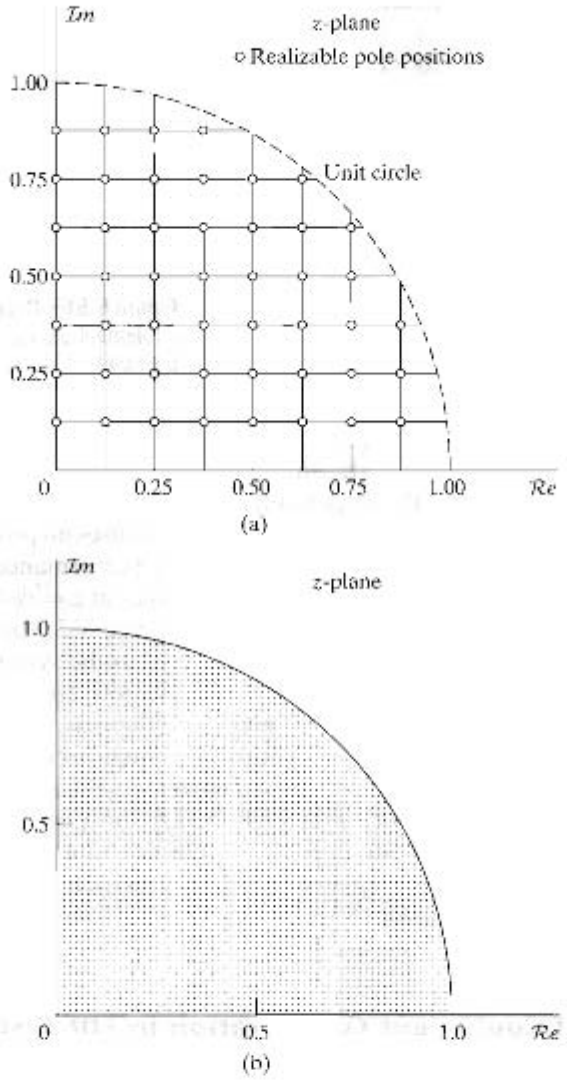


Figure 6.52 Pole locations for coupled form 2nd-order IIR system of Figure 6.51. (a) Four-bit quantization of coefficients. (b) Seven-bit quantization.

where

$$\Delta H(z) = \sum_{n=0}^{M} \Delta h[n] z^{-n}. \tag{6.88}$$

Thus, the system function (and therefore, also the frequency response) of the quantized system is linearly related to the quantization errors in the impulse-response coefficients. For this reason, the quantized system can be represented as in Figure 6.53, which shows the unquantized system in parallel with an error system whose impulse response is the sequence of quantization error samples $\{\Delta h[n]\}$ and whose system function is the corresponding z-transform, $\Delta H(z)$.

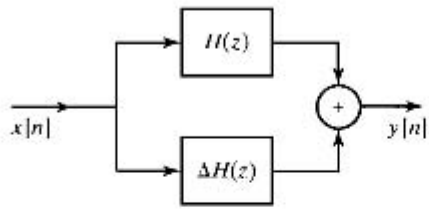


Figure 6.53 Representation of coefficient quantization in FIR systems.

TABLE 6.5 UNQUANTIZED AND QUANTIZED COEFFICIENTS FOR AN OPTIMUM FIR LOWPASS FILTER ($M = 27$)

Coefficient	Unquantized	16 bits	14 bits	13 bits	8 bits
$h[0] = h[27]$	1.359657×10^{-3}	45×2^{-15}	11×2^{-13}	6×2^{-12}	0×2^{-7}
$h[1] = h[26]$	-1.616993×10^{-3}	-53×2^{-15}	-13×2^{-13}	-7×2^{-12}	0×2^{-7}
$h[2] = h[25]$	-7.738032×10^{-3}	-254×2^{-15}	-63×2^{-13}	-32×2^{-12}	-1×2^{-7}
$h[3] = h[24]$	-2.686841×10^{-3}	-88×2^{-15}	-22×2^{-13}	-11×2^{-12}	0×2^{-7}
$h[4] = h[23]$	1.255246×10^{-2}	411×2^{-15}	103×2^{-13}	51×2^{-12}	2×2^{-7}
$h[5] = h[22]$	6.591530×10^{-3}	216×2^{-15}	54×2^{-13}	27×2^{-12}	1×2^{-7}
$h[6] = h[21]$	-2.217952×10^{-2}	-727×2^{-15}	-182×2^{-13}	-91×2^{-12}	-3×2^{-7}
$h[7] = h[20]$	-1.524663×10^{-2}	-500×2^{-15}	-125×2^{-13}	-62×2^{-12}	-2×2^{-7}
$h[8] = h[19]$	3.720668×10^{-2}	1219×2^{-15}	305×2^{-13}	152×2^{-12}	5×2^{-7}
$h[9] = h[18]$	3.233332×10^{-2}	1059×2^{-15}	265×2^{-13}	132×2^{-12}	4×2^{-7}
$h[10] = h[17]$	-6.537057×10^{-2}	-2142×2^{-15}	-536×2^{-13}	-268×2^{-12}	-8×2^{-7}
$h[11] = h[16]$	-7.528754×10^{-2}	-2467×2^{-15}	-617×2^{-13}	-308×2^{-12}	-10×2^{-7}
$h[12] = h[15]$	1.560970×10^{-1}	5115×2^{-15}	1279×2^{-13}	639×2^{-12}	20×2^{-7}
$h[13] = h[14]$	4.394094×10^{-1}	14399×2^{-15}	3600×2^{-13}	1800×2^{-12}	56×2^{-7}

Another approach to studying the sensitivity of the direct form FIR structure would be to examine the sensitivity of the zeros to quantization errors in the impulse-response coefficients, which are, of course the coefficients of the polynomial $H(z)$. If the zeros of $H(z)$ are tightly clustered, then their locations will be highly sensitive to quantization errors in the impulse-response coefficients. The reason that the direct form FIR system is widely used is that, for most linear phase FIR filters, the zeros are more or less uniformly spread in the z-plane. We demonstrate this by the following example.

6.8.5 Example of Quantization of an Optimum FIR Filter

As an example of the effect of coefficient quantization in the FIR case, consider a linear-phase lowpass filter designed to meet the following specifications:

$$0.99 \leq |H(e^{j\omega})| \leq 1.01, \qquad 0 \leq |\omega| \leq 0.4\pi,$$

$$|H(e^{j\omega})| \leq 0.001 (\text{i.e., } -60\,\text{dB}), \qquad 0.6\pi \leq |\omega| \leq \pi.$$

This filter was designed using the Parks–McClellan design technique, which will be discussed in Section 7.7.3. The details of the design for this example are considered in Section 7.8.1.

Table 6.5 shows the unquantized impulse-response coefficients for the system, along with quantized coefficients for 16-, 14-, 13-, and 8-bit quantization. Figure 6.54

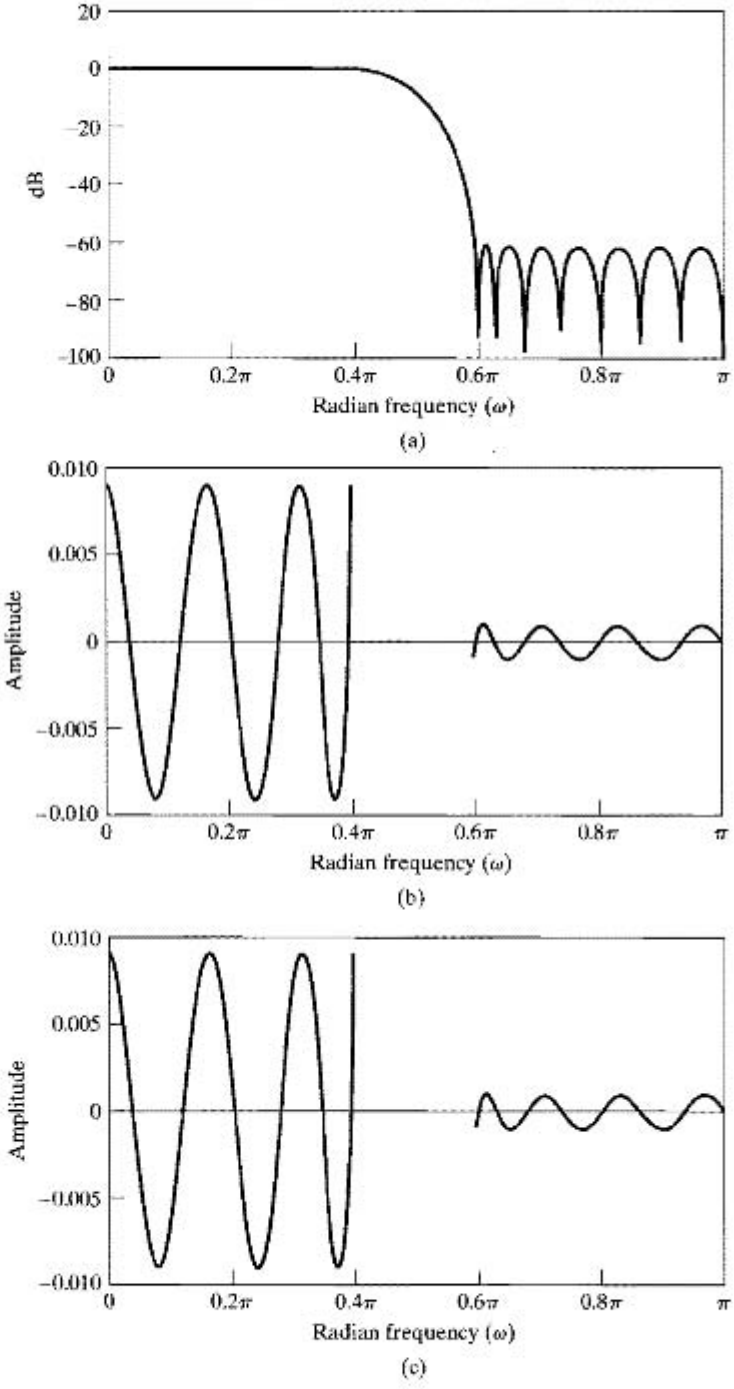
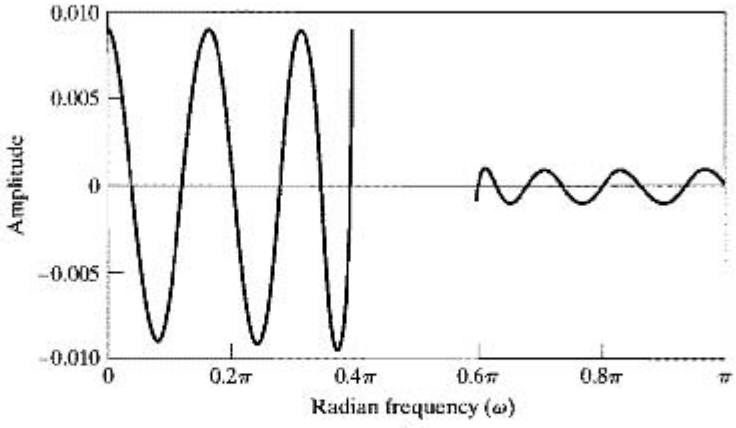
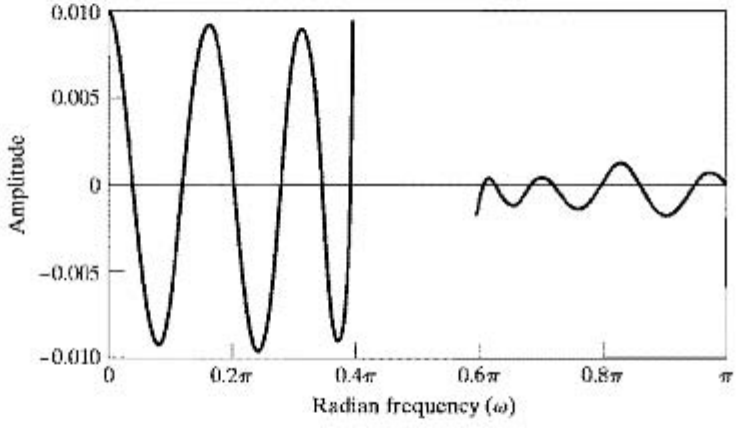


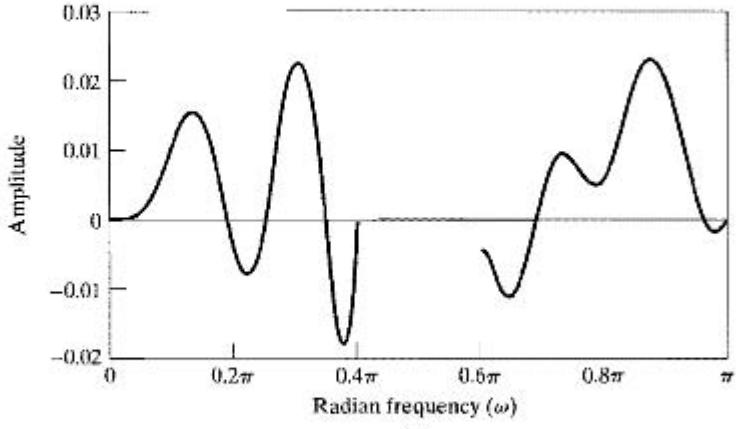
Figure 6.54 FIR quantization example. (a) Log magnitude for unquantized case. (b) Approximation error for unquantized case. (Error not defined in transition band.) (c) Approximation error for 16-bit quantization.



(d)



(e)



(f)

Figure 6.54 (*continued*) (d) Approximation error for 14-bit quantization. (e) Approximation error for 13-bit quantization. (f) Approximation error for 8-bit quantization.

gives a comparison of the frequency responses of the various systems. Figure 6.54(a) shows the log magnitude in dB of the frequency response for unquantized coefficients. Figures 6.54(b), (c), (d), (e), and (f) show the passband and stopband approximation errors (errors in approximating unity in the passband and zero in the stopband) for the unquantized, 16-, 14-, 13-, and 8-bit quantized cases, respectively. From Figure 6.54, we see that the system meets the specifications for the unquantized case and both the 16-bit and 14-bit quantized cases. However, with 13-bit quantization the stopband approximation error becomes greater than 0.001, and with 8-bit quantization the stopband approximation error is over 10 times as large as specified. Thus, we see that at least 14-bit coefficients are required for a direct form implementation of the system. However, this is not a serious limitation, since 16-bit or 14-bit coefficients are well matched to many of the technologies that might be used to implement such a filter.

The effect of quantization of the filter coefficients on the locations of the zeros of the filter is shown in Figure 6.55. Note that in the unquantized case, shown in Figure 6.55(a), the zeros are spread around the z-plane, although there is some clustering on the unit circle. The zeros on the unit circle are primarily responsible for developing the stopband attenuation, whereas those at conjugate reciprocal locations off the unit circle are primarily responsible for forming the passband. Note that little difference is observed in Figure 6.55(b) for 16-bit quantization, but in Figure 6.55(c), showing 13-bit quantization, the zeros on the unit circle have moved significantly. Finally, in Figure 6.55(d), we see that 8-bit quantization causes several of the zeros on the unit circle to pair up and move off the circle to conjugate reciprocal locations. This behavior of the zeros explains the behavior of the frequency response shown in Figure 6.54.

A final point about this example is worth mentioning. All of the unquantized coefficients have magnitudes less than 0.5. Consequently, if all of the coefficients (and therefore, the impulse response) are doubled prior to quantization, more efficient use of the available bits will result, corresponding in effect to increasing B by 1. In Table 6.5 and Figure 6.54, we did not take this potential for increased accuracy into account.

6.8.6 Maintaining Linear Phase

So far, we have not made any assumptions about the phase response of the FIR system. However, the possibility of generalized linear phase is one of the major advantages of an FIR system. Recall that a linear-phase FIR system has either a symmetric ($h[M-n] = h[n]$) or an antisymmetric ($h[M-n] = -h[n]$) impulse response. These linear-phase conditions are easily preserved for the direct form quantized system. Thus, all the systems discussed in the example of the previous subsection have a precisely linear phase, regardless of the coarseness of the quantization. This can be seen in the way in which the conjugate reciprocal locations are preserved in Figure 6.55.

Figure 6.55(d) suggests that, in situations where quantization is very coarse or for high-order systems with closely spaced zeros, it may be worthwhile to realize smaller sets of zeros independently with a cascade form FIR system. To maintain linear phase, each of the sections in the cascade must also have linear phase. Recall that the zeros of a linear-phase system must occur as illustrated in Figure 6.34. For example, if we use 2^{nd}-order sections of the form $(1+az^{-1}+z^{-2})$ for each complex-conjugate pair of zeros on the unit circle, the zero can move only on the unit circle when the coefficient a is

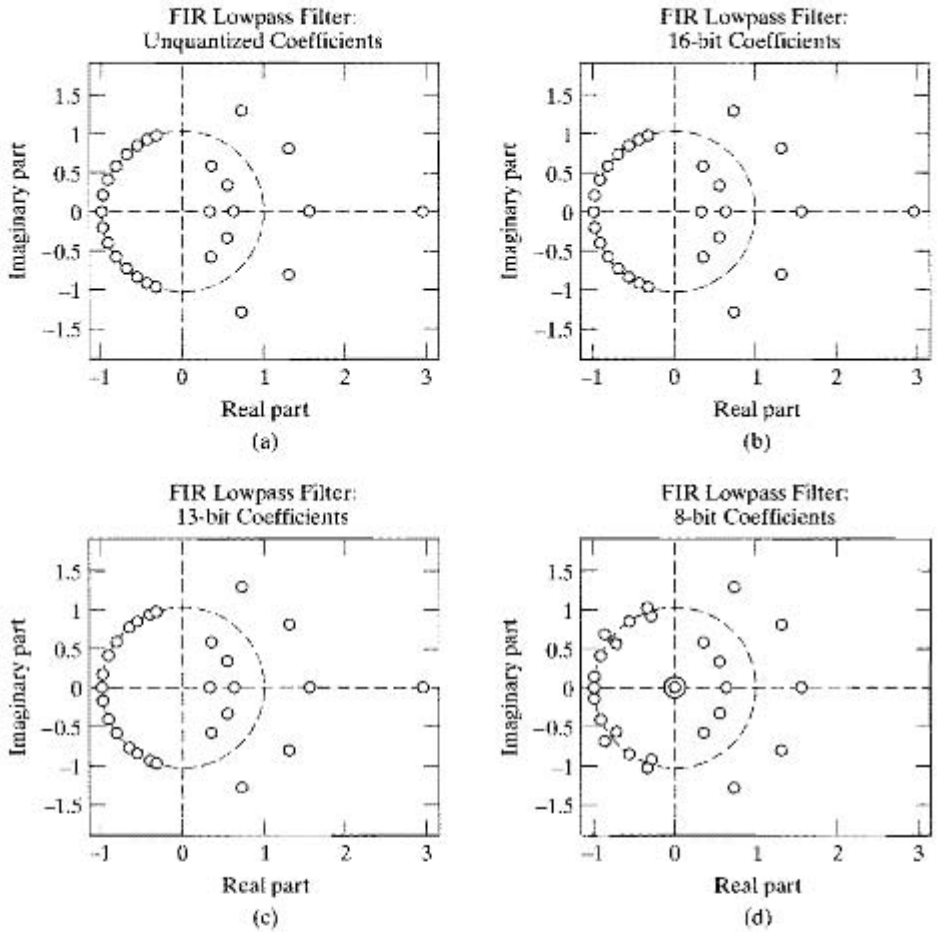


Figure 6.55 Effect of impulse response quantization on zeros of $H(z)$. (a) Unquantized. (b) 16-bit quantization. (c) 13-bit quantization. (d) 8-bit quantization.

quantized. This prevents zeros from moving away from the unit circle, thereby lessening their attenuating effect. Similarly, real zeros inside the unit circle and at the reciprocal location outside the unit circle would remain real. Also, zeros at $z = \pm 1$ can be realized exactly by 1st-order systems. If a pair of complex-conjugate zeros inside the unit circle is realized by a 2nd-order system rather than a 4th-order system, then we must ensure that, for each complex zero inside the unit circle, there is a conjugate reciprocal zero outside the unit circle. This can be done by expressing the 4th-order factor corresponding to zeros at $z = re^{j\theta}$ and $z = r^{-1}e^{-j\theta}$ as

$$\begin{aligned} &1 + cz^{-1} + dz^{-2} + cz^{-3} + z^{-4} \\ &\quad = (1 - 2r\cos\theta z^{-1} + r^2 z^{-2})\frac{1}{r^2}(r^2 - 2r\cos\theta z^{-1} + z^{-2}). \end{aligned} \tag{6.89}$$

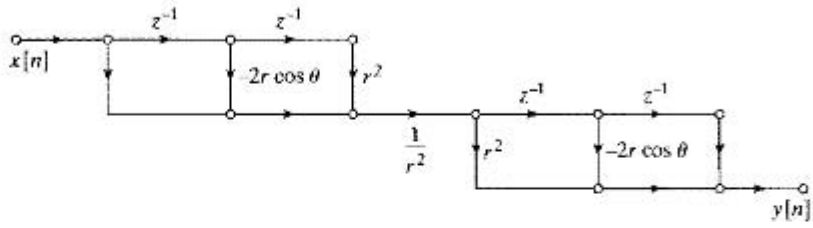


Figure 6.56 Subsystem to implement 4th-order factors in a linear-phase FIR system such that linearity of the phase is maintained independently of parameter quantization.

This condition corresponds to the subsystem shown in Figure 6.56. This system uses the same coefficients, $-2r\cos\theta$ and r^2, to realize both the zeros inside the unit circle and the conjugate reciprocal zeros outside the unit circle. Thus, the linear-phase condition is preserved under quantization. Notice that the factor $(1-2r\cos\theta z^{-1}+r^2z^{-2})$ is identical to the denominator of the 2nd-order direct form IIR system of Figure 6.49. Therefore, the set of quantized zeros is as depicted in Figure 6.50. More details on cascade realizations of FIR systems are given by Herrmann and Schüssler (1970b).

6.9 EFFECTS OF ROUND-OFF NOISE IN DIGITAL FILTERS

Difference equations realized with finite-precision arithmetic are nonlinear systems. Although it is important in general to understand how this nonlinearity affects the performance of discrete-time systems, a precise analysis of arithmetic quantization effects is generally not required in practical applications, where we are typically concerned with the performance of a specific system. Indeed, just as with coefficient quantization, the most effective approach is often to simulate the system and measure its performance. For example, a common objective in quantization error analysis is to choose the digital word length such that the digital system is a sufficiently accurate realization of the desired linear system and at the same time requires a minimum of hardware or software complexity. The digital word length can, of course, be changed only in steps of 1 bit, and as we have already seen in Section 4.8.2, the addition of 1 bit to the word length reduces the size of quantization errors by a factor of 2. Thus, the choice of word length is insensitive to inaccuracies in the quantization error analysis; an analysis that is correct to within 30 to 40 percent is often adequate. For this reason, many of the important effects of quantization can be studied using linear additive noise approximations. We develop such approximations in this section and illustrate their use with several examples. An exception is the phenomenon of zero-input limit cycles, which are strictly nonlinear phenomena. We restrict our study of nonlinear models for digital filters to a brief introduction to zero-input limit cycles in Section 6.10.

6.9.1 Analysis of the Direct Form IIR Structures

To introduce the basic ideas, let us consider the direct form structure for an LTI discrete-time system. The flow graph of a direct form I 2nd-order system is shown in

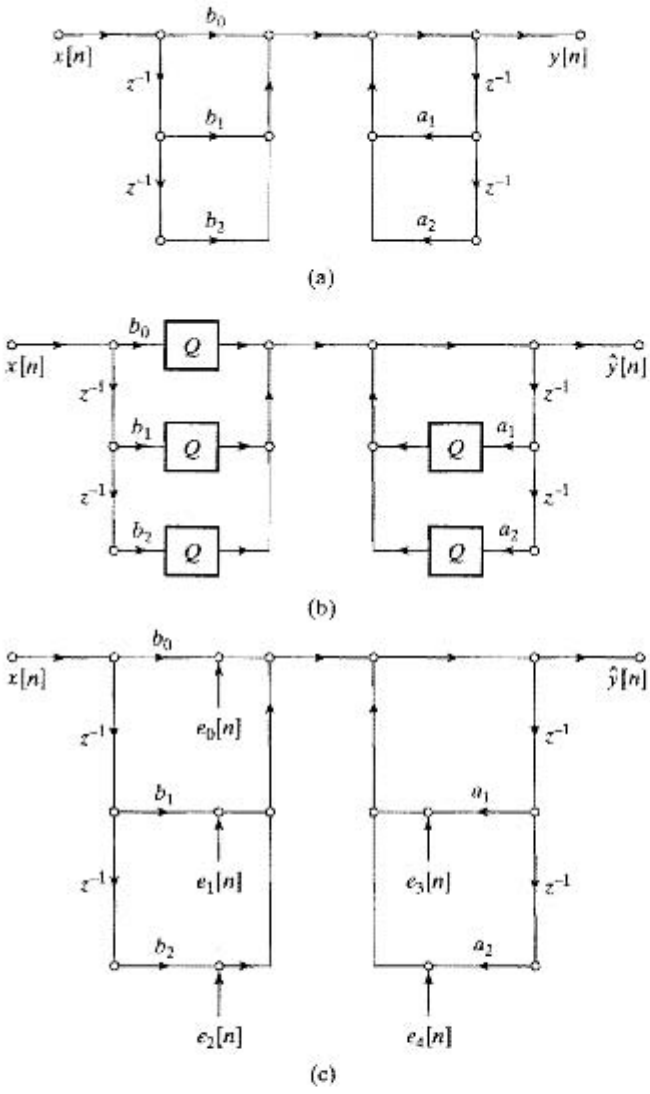


Figure 6.57 Models for direct form I system. (a) Infinite-precision model. (b) Nonlinear quantized model. (c) Linear-noise model.

Figure 6.57(a). The general N^{th}-order difference equation for the direct form I structure is

$$y[n] = \sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k], \tag{6.90}$$

and the system function is

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}} = \frac{B(z)}{A(z)}. \tag{6.91}$$

Let us assume that all signal values and coefficients are represented by $(B+1)$-bit fixed-point binary numbers. Then, in implementing Eq. (6.90) with a $(B+1)$-bit adder, it would be necessary to reduce the length of the $(2B+1)$-bit products resulting from multiplying two $(B+1)$-bit numbers back to $(B+1)$ bits. Since all numbers are treated as fractions, we would discard the least significant B bits by either rounding or truncation. This is represented by replacing each constant multiplier branch in Figure 6.57(a) by a constant multiplier followed by a quantizer, as in the nonlinear model of Figure 6.57(b). The difference equation corresponding to Figure 6.57(b) is the nonlinear equation

$$\hat{y}[n] = \sum_{k=1}^{N} Q[a_k \hat{y}[n-k]] + \sum_{k=0}^{M} Q[b_k x[n-k]]. \tag{6.92}$$

Figure 6.57(c) shows an alternative representation in which the quantizers are replaced by noise sources that are equal to the quantization error at the output of each quantizer. For example, rounding or truncation of a product $bx[n]$ is represented by a noise source of the form

$$e[n] = Q[bx[n]] - bx[n]. \tag{6.93}$$

If the noise sources are known exactly, then Figure 6.57(c) is exactly equivalent to Figure 6.57(b). However, Figure 6.57(c) is most useful when we assume that each quantization noise source has the following properties:

1. Each quantization noise source $e[n]$ is a wide-sense-stationary white-noise process.
2. Each quantization noise source has a uniform distribution of amplitudes over one quantization interval.
3. Each quantization noise source is *uncorrelated* with the input to the corresponding quantizer, all other quantization noise sources, and the input to the system.

These assumptions are identical to those made in the analysis of A/D conversion in Section 4.8. Strictly speaking, our assumptions here cannot be valid, since the quantization error depends directly on the input to the quantizer. This is readily apparent for constant and sinusoidal signals. However, experimental and theoretical analyses have shown (see Bennett, 1948; Widrow, 1956, 1961; Widrow and Kollár, 2008) that in many situations the approximation just described leads to accurate predictions of measured statistical averages such as the mean, variance, and correlation function. This is true when the input signal is a complicated wideband signal such as speech, in which the signal fluctuates rapidly among all the quantization levels and traverses many of those levels in going from sample to sample (see Gold and Rader, 1969). The simple linear-noise approximation presented here allows us to characterize the noise generated in the system by averages such as the mean and variance and to determine how these averages are modified by the system.

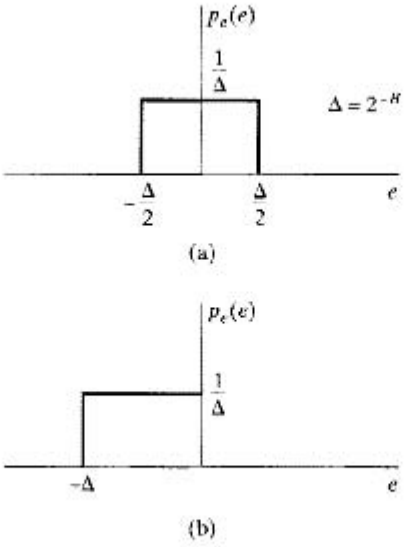


Figure 6.58 Probability density function for quantization errors. (a) Rounding. (b) Truncation.

For $(B+1)$-bit quantization, we showed in Section 6.7.1 that, for rounding,

$$-\frac{1}{2}2^{-B} < e[n] \leq \frac{1}{2}2^{-B}, \tag{6.94a}$$

and for two's-complement truncation,

$$-2^{-B} < e[n] \leq 0. \tag{6.94b}$$

Thus, according to our second assumption, the probability density functions for the random variables representing quantization error are the uniform densities shown in Figure 6.58(a) for rounding and in Figure 6.58(b) for truncation. The mean and variance for rounding are, respectively,

$$m_e = 0, \tag{6.95a}$$

$$\sigma_e^2 = \frac{2^{-2B}}{12}. \tag{6.95b}$$

For two's-complement truncation, the mean and variance are

$$m_e = -\frac{2^{-B}}{2}, \tag{6.96a}$$

$$\sigma_e^2 = \frac{2^{-2B}}{12}. \tag{6.96b}$$

In general, the autocorrelation sequence of a quantization noise source is, according to the first assumption,

$$\phi_{ee}[n] = \sigma_e^2\,\delta[n] + m_e^2. \tag{6.97}$$

In the case of rounding, which we will assume for convenience henceforth, $m_e = 0$, so the autocorrelation function is $\phi_{ee}[n] = \sigma_e^2\,\delta[n]$, and the power spectrum is $\Phi_{ee}(e^{j\omega}) = \sigma_e^2$ for $|\omega| \leq \pi$. In this case, the variance and the average power are identical. In the

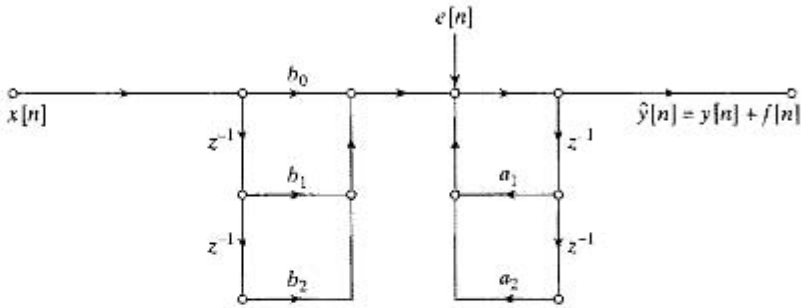


Figure 6.59 Linear-noise model for direct form I with noise sources combined.

case of truncation, the mean is not zero, so average-power results derived for rounding must be corrected by computing the mean of the signal and adding its square to the average-power results for rounding.

With this model for each of the noise sources in Figure 6.57(c), we can now proceed to determine the effect of the quantization noise on the output of the system. To aid us in doing this, it is helpful to observe that all of the noise sources in that figure are effectively injected between the part of the system that implements the zeros and the part that implements the poles. Thus, Figure 6.59 is equivalent to Figure 6.57(c) if $e[n]$ in Figure 6.59 is

$$e[n] = e_0[n] + e_1[n] + e_2[n] + e_3[n] + e_4[n]. \tag{6.98}$$

Since we assume that all the noise sources are independent of the input and independent of each other, the variance of the combined noise sources for the 2nd-order direct form I case is

$$\sigma_e^2 = \sigma_{e_0}^2 + \sigma_{e_1}^2 + \sigma_{e_2}^2 + \sigma_{e_3}^2 + \sigma_{e_4}^2 = 5 \cdot \frac{2^{-2B}}{12}, \tag{6.99}$$

and for the general direct form I case, it is

$$\sigma_e^2 = (M + 1 + N)\frac{2^{-2B}}{12}. \tag{6.100}$$

To obtain an expression for the output noise, we note from Figure 6.59 that the system has two inputs, $x[n]$ and $e[n]$, and since the system is now assumed to be linear, the output can be represented as $\hat{y}[n] = y[n] + f[n]$, where $y[n]$ is the response of the ideal unquantized system to the input sequence $x[n]$ and $f[n]$ is the response of the system to the input $e[n]$. The output $y[n]$ is given by the difference equation (6.90), but since $e[n]$ is injected after the zeros and before the poles, the output noise satisfies the difference equation

$$f[n] = \sum_{k=1}^{N} a_k f[n-k] + e[n]; \tag{6.101}$$

i.e., the properties of the output noise in the direct form I implementation depend only on the poles of the system.

To determine the mean and variance of the output noise sequence, we can use some results from Section 2.10. Consider a linear system with system function $H_{ef}(z)$

with a white-noise input $e[n]$ and corresponding output $f[n]$. Then, from Eqs. (2.184) and (2.185), the mean of the output is

$$m_f = m_e \sum_{n=-\infty}^{\infty} h_{ef}[n] = m_e H_{ef}(e^{j0}). \tag{6.102}$$

Since $m_e = 0$ for rounding, the mean of the output will be zero, so we need not be concerned with the mean value of the noise if we assume rounding. From Eqs. (6.97) and (2.190), it follows that, because, for rounding, $e[n]$ is a zero-mean white-noise sequence, the power density spectrum of the output noise is simply

$$P_{ff}(\omega) = \Phi_{ff}(e^{j\omega}) = \sigma_e^2 |H_{ef}(e^{j\omega})|^2. \tag{6.103}$$

From Eq. (2.192), the variance of the output noise can be shown to be

$$\sigma_f^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} P_{ff}(\omega)d\omega = \sigma_e^2 \frac{1}{2\pi}\int_{-\pi}^{\pi} |H_{ef}(e^{j\omega})|^2 d\omega. \tag{6.104}$$

Using Parseval's theorem in the form of Eq. (2.162), we can also express σ_f^2 as

$$\sigma_f^2 = \sigma_e^2 \sum_{n=-\infty}^{\infty} |h_{ef}[n]|^2. \tag{6.105}$$

When the system function corresponding to $h_{ef}[n]$ is a rational function, as it will always be for difference equations of the type considered in this chapter, we can use Eq. (A.66) in Appendix A for evaluating infinite sums of squares of the form of Eq. (6.105).

We will use the results summarized in Eqs. (6.102)–(6.105) often in our analysis of quantization noise in linear systems. For example, for the direct form I system of Figure 6.59, $H_{ef}(z) = 1/A(z)$; i.e., the system function from the point where all the noise sources are injected to the output consists only of the poles of the system function $H(z)$ in Eq. (6.91). Thus, we conclude that, in general, the total output variance owing to internal round-off or truncation is

$$\begin{aligned}\sigma_f^2 &= (M+1+N)\frac{2^{-2B}}{12}\frac{1}{2\pi}\int_{-\pi}^{\pi}\frac{d\omega}{|A(e^{j\omega})|^2} \\ &= (M+1+N)\frac{2^{-2B}}{12}\sum_{n=-\infty}^{\infty}|h_{ef}[n]|^2,\end{aligned} \tag{6.106}$$

where $h_{ef}[n]$ is the impulse response corresponding to $H_{ef}(z) = 1/A(z)$. The use of the preceding results is illustrated by the following examples.

Example 6.11 Round-off Noise in a 1st-Order System

Suppose that we wish to implement a stable system having the system function

$$H(z) = \frac{b}{1-az^{-1}}, \qquad |a| < 1. \tag{6.107}$$

Figure 6.60 shows the flow graph of the linear-noise model for the implementation in which products are quantized before addition. Each noise source is filtered by the system from $e[n]$ to the output, for which the impulse response is $h_{ef}[n] = a^n u[n]$.

Since $M = 0$ and $N = 1$ for this example, from Eq. (6.103), the power spectrum of the output noise is

$$P_{ff}(\omega) = 2\frac{2^{-2B}}{12}\left(\frac{1}{1+a^2-2a\cos\omega}\right), \tag{6.108}$$

and the total noise variance at the output is

$$\sigma_f^2 = 2\frac{2^{-2B}}{12}\sum_{n=0}^{\infty}|a|^{2n} = 2\frac{2^{-2B}}{12}\left(\frac{1}{1-|a|^2}\right). \tag{6.109}$$

From Eq. (6.109), we see that the output noise variance increases as the pole at $z = a$ approaches the unit circle. Thus, to maintain the noise variance below a specified level as $|a|$ approaches unity, we must use longer word lengths. The following example also illustrates this point.

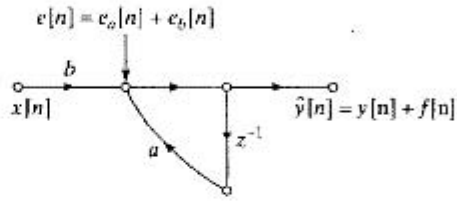


Figure 6.60 1st-order linear noise model.

Example 6.12 Round-off Noise in a 2nd-Order System

Consider a stable 2nd-order direct form I system with system function

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{(1 - re^{j\theta}z^{-1})(1 - re^{-j\theta}z^{-1})}. \tag{6.110}$$

The linear-noise model for this system is shown in Figure 6.57(c), or equivalently, Figure 6.59, with $a_1 = 2r\cos\theta$ and $a_2 = -r^2$. In this case, the total output noise power can be expressed in the form

$$\sigma_f^2 = 5\frac{2^{-2B}}{12}\frac{1}{2\pi}\int_{-\pi}^{\pi}\frac{d\omega}{|(1 - re^{j\theta}e^{-j\omega})(1 - re^{-j\theta}e^{-j\omega})|^2}. \tag{6.111}$$

Using Eq. (A.66) in Appendix A, the output noise power is found to be

$$\sigma_f^2 = 5\frac{2^{-2B}}{12}\left(\frac{1+r^2}{1-r^2}\right)\frac{1}{r^4 + 1 - 2r^2\cos 2\theta}. \tag{6.112}$$

As in Example 6.11, we see that as the complex conjugate poles approach the unit circle ($r \to 1$), the total output noise variance increases, thus requiring longer word lengths to maintain the variance below a prescribed level.

The techniques of analysis developed so far for the direct form I structure can also be applied to the direct form II structure. The nonlinear difference equations for the

direct form II structure are of the form

$$\hat{w}[n] = \sum_{k=1}^{N} Q[a_k \hat{w}[n-k]] + x[n], \tag{6.113a}$$

$$\hat{y}[n] = \sum_{k=0}^{M} Q[b_k \hat{w}[n-k]]. \tag{6.113b}$$

Figure 6.61(a) shows the linear-noise model for a 2^{nd}-order direct form II system. A noise source has been introduced after each multiplication, indicating that the products are quantized to $(B+1)$ bits before addition. Figure 6.61(b) shows an equivalent linear model, wherein we have moved the noise sources resulting from implementation of the poles and combined them into a single noise source $e_a[n] = e_3[n] + e_4[n]$ at the input. Likewise, the noise sources due to implementation of the zeros are combined into the single noise source $e_b[n] = e_0[n] + e_1[n] + e_2[n]$ that is added directly to the output. From this equivalent model, it follows that for M zeros and N poles and rounding ($m_e = 0$), the power spectrum of the output noise is

$$P_{ff}(\omega) = N\frac{2^{-2B}}{12}|H(e^{j\omega})|^2 + (M+1)\frac{2^{-2B}}{12}, \tag{6.114}$$

and the output noise variance is

$$\begin{aligned}\sigma_f^2 &= N\frac{2^{-2B}}{12}\frac{1}{2\pi}\int_{-\pi}^{\pi}|H(e^{j\omega})|^2 d\omega + (M+1)\frac{2^{-2B}}{12} \\ &= N\frac{2^{-2B}}{12}\sum_{n=-\infty}^{\infty}|h[n]|^2 + (M+1)\frac{2^{-2B}}{12},\end{aligned} \tag{6.115}$$

That is, the white noise produced in implementing the poles is filtered by the entire system, whereas the white noise produced in implementing the zeros is added directly to the output of the system. In writing Eq. (6.115), we have assumed that the N noise sources at the input are independent, so that their sum has N times the variance of a single quantization noise source. The same assumption was made about the $(M+1)$ noise sources at the output. These results are easily modified for two's-complement truncation. Recall from Eqs. (6.95a)–(6.95b) and Eqs. (6.96a)–(6.96b) that the variance of a truncation noise source is the same as that of a rounding noise source, but the mean of a truncation noise source is not zero. Consequently, the formulas in Eqs. (6.106) and (6.115) for the total output noise variance also hold for truncation. However, the output noise will have a nonzero average value that can be computed using Eq. (6.102).

A comparison of Eq. (6.106) with Eq. (6.115) shows that the direct form I and direct form II structures are affected differently by the quantization of products in implementing the corresponding difference equations. In general, other equivalent structures such as cascade, parallel, and transposed forms will have a total output noise variance different from that of either of the direct form structures. However, even though Eqs. (6.106) and (6.115) are different, we cannot say which system will have the smaller output noise variance unless we know specific values for the coefficients of the system. In other words, it is not possible to state that a particular structural form will always produce the least output noise.

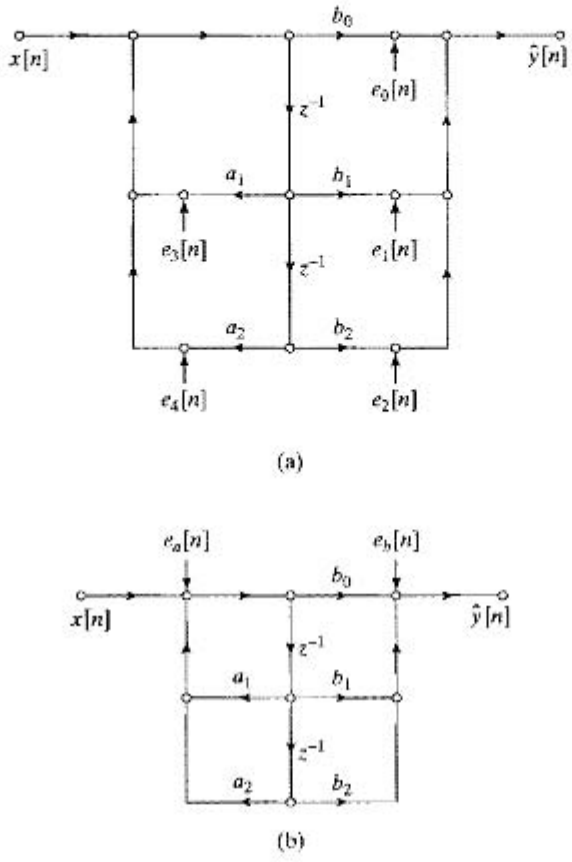


Figure 6.61 Linear-noise models for direct form II. (a) Showing quantization of individual products. (b) With noise sources combined.

It is possible to improve the noise performance of the direct form systems (and therefore cascade and parallel forms as well) by using a $(2B+1)$-bit adder to accumulate the sum of products required in both direct form systems. For example, for the direct form I implementation, we could use a difference equation of the form

$$\hat{y}[n] = Q\left[\sum_{k=1}^{N} a_k \hat{y}[n-k] + \sum_{k=0}^{M} b_k x[n-k]\right]; \tag{6.116}$$

i.e., the sums of products are accumulated with $(2B+1)$- or $(2B+2)$-bit accuracy, and the result is quantized to $(B+1)$ bits for output and storage in the delay memory. In the direct form I case, this means that the quantization noise is still filtered by the poles, but the factor $(M+1+N)$ in Eq. (6.106) is replaced by unity. Similarly, for the direct form II realization, the difference equations (6.113a)–(6.113b) can respectively be replaced by

$$\hat{w}[n] = Q\left[\sum_{k=1}^{N} a_k \hat{w}[n-k] + x[n]\right] \tag{6.117a}$$

and

$$\hat{y}[n] = Q\left[\sum_{k=0}^{M} b_k \hat{w}[n-k]\right]. \tag{6.117b}$$

This implies a single noise source at both the input and output, so the factors N and $(M+1)$ in Eq. (6.115) are each replaced by unity. Thus, the double-length accumulator provided in most DSP chips can be used to significantly reduce quantization noise in direct form systems.

6.9.2 Scaling in Fixed-Point Implementations of IIR Systems

The possibility of overflow is another important consideration in the implementation of IIR systems using fixed-point arithmetic. If we follow the convention that each fixed-point number represents a fraction (possibly times a known scale factor), each node in the structure must be constrained to have a magnitude less than unity to avoid overflow. If $w_k[n]$ denotes the value of the k^{th} node variable, and $h_k[n]$ denotes the impulse response from the input $x[n]$ to the node variable $w_k[n]$, then

$$|w_k[n]| = \left| \sum_{m=-\infty}^{\infty} x[n-m]h_k[m] \right|. \tag{6.118}$$

The bound

$$|w_k[n]| \leq x_{\max} \sum_{m=-\infty}^{\infty} |h_k[m]| \tag{6.119}$$

is obtained by replacing $x[n-m]$ by its maximum value $x_{\max}$ and using the fact that the magnitude of a sum is less than or equal to the sum of the magnitudes of the summands. Therefore, a sufficient condition for $|w_k[n]| < 1$ is

$$x_{\max} < \frac{1}{\displaystyle\sum_{m=-\infty}^{\infty} |h_k[m]|} \tag{6.120}$$

for all nodes in the flow graph. If $x_{\max}$ does not satisfy Eq. (6.120), then we can multiply $x[n]$ by a scaling multiplier s at the input to the system so that $sx_{\max}$ satisfies Eq. (6.120) for all nodes in the flow graph; i.e.,

$$sx_{\max} < \frac{1}{\displaystyle\max_k \left[\sum_{m=-\infty}^{\infty} |h_k[m]| \right]}. \tag{6.121}$$

Scaling the input in this way guarantees that overflow never occurs at any of the nodes in the flow graph. Equation (6.120) is necessary as well as sufficient, since an input always exists such that Eq. (6.119) is satisfied with equality. (See Eq. (2.70) in the discussion of stability in Section 2.4.) However, Eq. (6.120) leads to a very conservative scaling of the input for most signals.

Another approach to scaling is to assume that the input is a narrowband signal, modeled as $x[n] = x_{\max} \cos \omega_0 n$. In this case, the node variables will have the form

$$w_k[n] = |H_k(e^{j\omega_0})| x_{\max} \cos(\omega_0 n + \angle H_k(e^{j\omega_0})). \tag{6.122}$$

Therefore, overflow is avoided for *all* sinusoidal signals if

$$\max_{k,|\omega| \leq \pi} |H_k(e^{j\omega})| x_{\max} < 1 \tag{6.123}$$

or if the input is scaled by the scale factor s such that

$$sx_{\max} < \frac{1}{\max\limits_{k,|\omega|\le\pi} |H_k(e^{j\omega})|}. \tag{6.124}$$

Still another scaling approach is based on the energy $E = \Sigma_n |x[n]|^2$ of the input signal. We can derive the scale factor in this case by applying the Schwarz inequality (see Bartle, 2000) to obtain the following inequality relating the square of the node signal to the energies of the input signal and the node impulse response:

$$\begin{aligned} |w_k[n]|^2 &= \left| \frac{1}{2\pi} \int_{-\pi}^{\pi} H_k(e^{j\omega}) X(e^{j\omega}) e^{j\omega n} d\omega \right|^2 \\ &\le \left(\frac{1}{2\pi} \int_{-\pi}^{\pi} |H_k(e^{j\omega})|^2 d\omega \right) \left(\frac{1}{2\pi} \int_{-\pi}^{\pi} |X(e^{j\omega})|^2 d\omega \right). \end{aligned} \tag{6.125}$$

Therefore, if we scale the input sequence values by s and apply Parseval's theorem, we see that $|w_k[n]|^2 < 1$ for all nodes k if

$$s^2 \left(\sum_{n=-\infty}^{\infty} |x[n]|^2 \right) = s^2 E < \frac{1}{\max\limits_k \left[\sum\limits_{n=-\infty}^{\infty} |h_k[n]|^2 \right]}. \tag{6.126}$$

Since it can be shown that for the k^{th} node,

$$\left\{ \sum_{n=-\infty}^{\infty} |h_k[n]|^2 \right\}^{1/2} \le \max_{\omega} |H_k(e^{j\omega})| \le \sum_{n=-\infty}^{\infty} |h_k[n]|, \tag{6.127}$$

it follows that (for most input signals) Eqs. (6.121), (6.124), and (6.126) give three decreasingly conservative ways of scaling the input to a digital filter (equivalently decreasing the gain of the filter). Of the three, Eq. (6.126) is generally the easiest to evaluate analytically because the partial fraction method of Appendix A can be used; however use of Eq. (6.126) requires an assumption about the mean-squared value of the signal, E. On the other hand, Eq. (6.121) is difficult to evaluate analytically, except for the simplest systems. Of course, if the filter coefficients are fixed numbers, the scale factors can be estimated by computing the impulse response or frequency response numerically.

If the input must be scaled down ($s < 1$), the signal-to-noise ratio (SNR) at the output of the system will be reduced because the signal power is reduced, but the noise power is dependent only on the rounding operation. Figure 6.62 shows 2^{nd}-order direct form I and direct form II systems with scaling multipliers at the input. In determining the scaling multiplier for these systems, it is not necessary to examine each node in the flow graph. Some nodes do not represent addition and thus cannot overflow. Other nodes represent partial sums. If we use nonsaturation two's-complement arithmetic, such nodes are permitted to overflow if certain key nodes do not. For example, in Figure 6.62(a), we can focus on the node enclosed by the dashed circle. In the figure, the scaling multiplier is shown combined with the b_ks, so that the noise source is the same

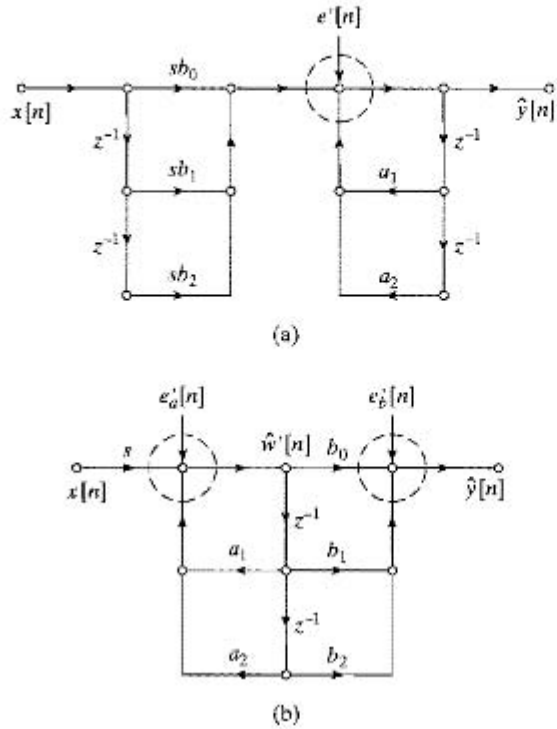


Figure 6.62 Scaling of direct form systems. (a) Direct form I. (b) Direct form II.

as in Figure 6.59; i.e., it has five times the power of a single quantization noise source.[10] Since the noise source is again filtered only by the poles, the output noise power is the same in Figures 6.59 and 6.62(a). However, the overall system function of the system in Figure 6.62(a) is $sH(z)$ instead of $H(z)$, so the unquantized component of the output $\hat{y}[n]$ is $sy[n]$ instead of $y[n]$. Since the noise is injected after the scaling, the ratio of signal power to noise power in the scaled system is s^2 times the SNR for Figure 6.59. Because $s < 1$ if scaling is required to avoid overflow, the SNR is reduced by scaling.

The same is true for the direct form II system of Figure 6.62(b). In this case, we must determine the scaling multiplier to avoid overflow at both of the circled nodes. Again, the overall gain of the system is s times the gain of the system in Figure 6.61(b), but it may be necessary to implement the scaling multiplier explicitly in this case to avoid overflow at the node on the left. This scaling multiplier adds an additional noise component to $e_a[n]$, so the noise power at the input is, in general, $(N + 1)2^{-2B}/12$. Otherwise, the noise sources are filtered by the system in exactly the same way in both Figure 6.61(b) and Figure 6.62(b). Therefore, the signal power is multiplied by s^2, and the noise power at the output is again given by Eq. (6.115), with N replaced by $(N + 1)$. The SNR is again reduced if scaling is required to avoid overflow.

[10]This eliminates a separate scaling multiplication and quantization noise source. However, scaling (and quantizing) the b_ks can change the frequency response of the system. If a separate input scaling multiplier precedes the implementation of the zeros in Figure 6.62(a), then an additional quantization noise source would contribute to the output noise after going through the entire system $H(z)$.

Example 6.13 Interaction Between Scaling and Round-off Noise

To illustrate the interaction of scaling and round-off noise, consider the system of Example 6.11 with system function given by Eq. (6.107). If the scaling multiplier is combined with the coefficient b, we obtain the flow graph of Figure 6.63 for the scaled system. Suppose that the input is white noise with amplitudes uniformly distributed between -1 and $+1$. Then the total signal variance is $\sigma_x^2 = 1/3$. To guarantee no overflow in computing $\hat{y}[n]$, we use Eq. (6.121) to compute the scale factor

$$s = \frac{1}{\sum_{n=0}^{\infty} |b|\,|a|^n} = \frac{1-|a|}{|b|}. \tag{6.128}$$

The output noise variance was determined in Example 6.11 to be

$$\sigma_f^2 = 2\frac{2^{-2B}}{12}\frac{1}{1-a^2} \tag{6.129}$$

and since we again have two $(B+1)$-bit rounding operations, the noise power at the output is the same, i.e., $\sigma_{f'}^2 = \sigma_f^2$. The variance of the output $y'[n]$ due to the scaled input $sx[n]$ is

$$\sigma_{y'}^2 = \left(\frac{1}{3}\right)\frac{s^2b^2}{1-a^2} = s^2\sigma_y^2. \tag{6.130}$$

Therefore, the SNR at the output is

$$\frac{\sigma_{y'}^2}{\sigma_{f'}^2} = s^2\frac{\sigma_y^2}{\sigma_f^2} = \left(\frac{1-|a|}{|b|}\right)^2\frac{\sigma_y^2}{\sigma_f^2}. \tag{6.131}$$

As the pole of the system approaches the unit circle, the SNR decreases because the quantization noise is amplified by the system and because the high gain of the system forces the input to be scaled down to avoid overflow. Again, we see that overflow and quantization noise work in opposition to decrease the performance of the system.

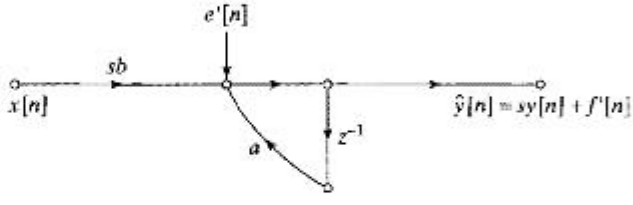


Figure 6.63 Scaled 1st-order system.

6.9.3 Example of Analysis of a Cascade IIR Structure

The previous results of this section can be applied directly to the analysis of either parallel or cascade structures composed of 2nd-order direct form subsystems. The interaction of scaling and quantization is particularly interesting in the cascade form. Our general

comments on cascade systems will be interwoven with a specific example.

An elliptic lowpass filter was designed to meet the following specifications:

$$0.99 \le |H(e^{j\omega})| \le 1.01, \qquad |\omega| \le 0.5\pi,$$

$$|H(e^{j\omega})| \le 0.01, \qquad 0.56\pi \le |\omega| \le \pi.$$

The system function of the resulting system is

$$H(z) = 0.079459 \prod_{k=1}^{3} \left(\frac{1 + b_{1k}z^{-1} + z^{-2}}{1 - a_{1k}z^{-1} - a_{2k}z^{-2}} \right) = 0.079459 \prod_{k=1}^{3} H_k(z). \tag{6.132}$$

where the coefficients are given in Table 6.6. Notice that all the zeros of $H(z)$ are on the unit circle in this example; however, that need not be the case in general.

Figure 6.64(a) shows a flow graph of a possible implementation of this system as a cascade of 2nd-order transposed direct form II subsystems. The gain constant, 0.079459, is such that the overall gain of the system is approximately unity in the passband, and it is assumed that this guarantees no overflow at the output of the system. Figure 6.64(a) shows the gain constant placed at the input to the system. This approach reduces the amplitude of the signal immediately, with the result that the subsequent filter sections must have high gain to produce an overall gain of unity. Since the quantization noise sources are introduced after the gain of 0.079459 but are likewise amplified by the rest of the system, this is not a good approach. Ideally, the overall gain constant, being less than unity, should be placed at the very end of the cascade, so that the signal and noise will be attenuated by the same amount. However, this creates the possibility of overflow along the cascade. Therefore, a better approach is to distribute the gain among the three stages of the system, so that overflow is just avoided at each stage of the cascade. This distribution is represented by

$$H(z) = s_1 H_1(z) s_2 H_2(z) s_3 H_3(z), \tag{6.133}$$

where $s_1 s_2 s_3 = 0.079459$. The scaling multipliers can be incorporated into the coefficients of the numerators of the individual system functions $H_k'(z) = s_k H_k(z)$, as in

$$H(z) = \prod_{k=1}^{3} \left(\frac{b_{0k}' + b_{1k}'z^{-1} + b_{2k}'z^{-2}}{1 - a_{1k}z^{-1} - a_{2k}z^{-2}} \right) = \prod_{k=1}^{3} H_k'(z), \tag{6.134}$$

where $b_{0k}' = b_{2k}' = s_k$ and $b_{1k}' = s_k b_{1k}$. The resulting scaled system is depicted in Figure 6.64(b).

Also shown in Figure 6.64(b) are quantization noise sources representing the quantization of the products before addition. Figure 6.64(c) shows an equivalent noise model,

TABLE 6.6 COEFFICIENTS FOR ELLIPTIC LOWPASS FILTER IN CASCADE FORM

k	a_{1k}	a_{2k}	b_{1k}
1	0.478882	−0.172150	1.719454
2	0.137787	−0.610077	0.781109
3	−0.054779	−0.902374	0.411452

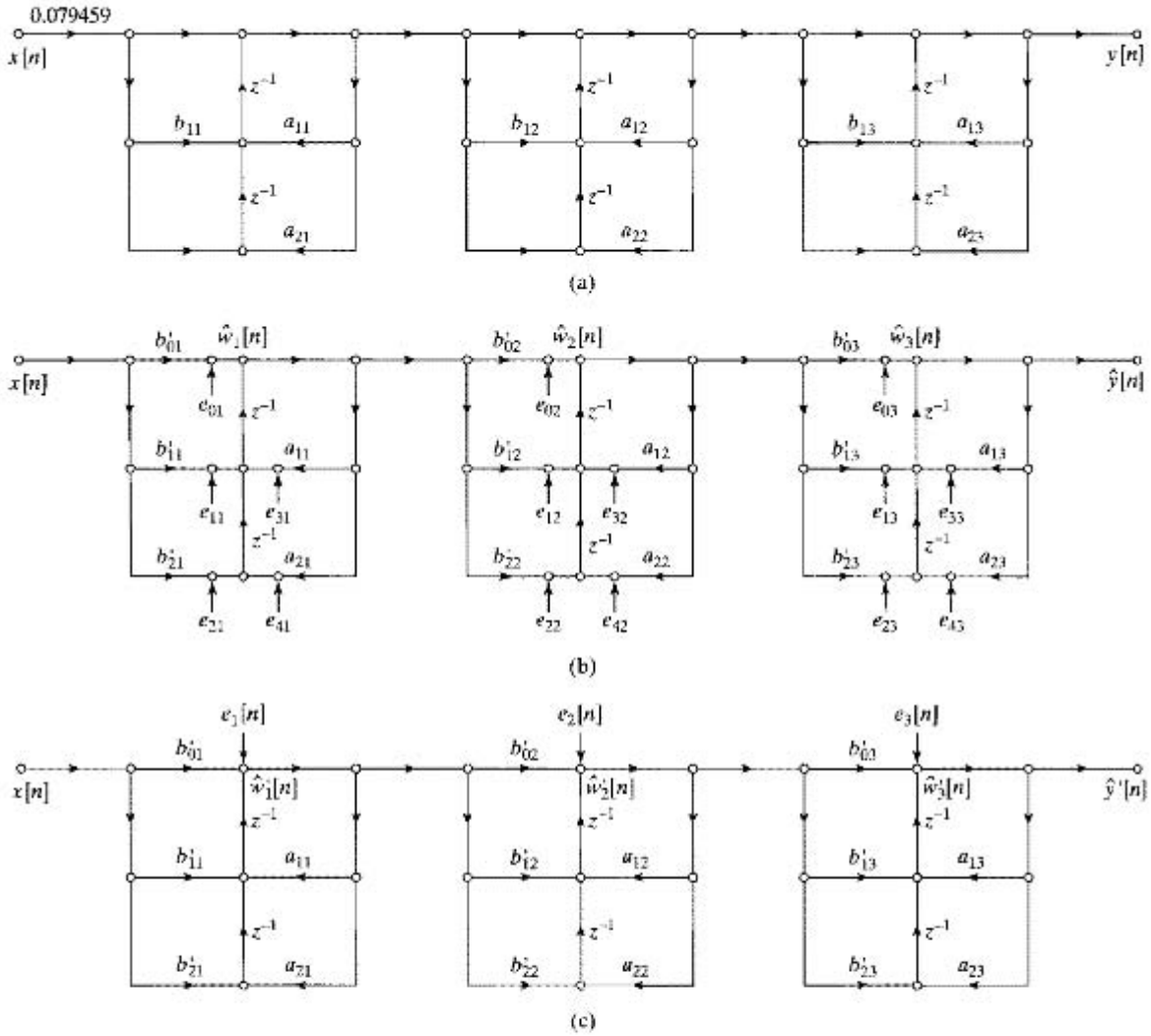


Figure 6.64 Models for 6th-order cascade system with transposed direct form II subsystems. (a) Infinite-precision model. (b) Linear-noise model for scaled system, showing quantization of individual multiplications. (c) Linear-noise model with noise sources combined.

for which it is recognized that all the noise sources in a particular section are filtered only by the poles of that section (and the subsequent subsystems). Figure 6.64(c) also uses the fact that delayed white-noise sources are still white noise and are independent of all the other noise sources, so that all five sources in a subsection can be combined into a single noise source having five times the variance of a single quantization noise source.[11] Since the noise sources are assumed independent, the variance of the output

[11] This discussion can be generalized to show that the transposed direct form II has the same noise behavior as the direct form I system.

noise is the sum of the variances owing to the three noise sources in Figure 6.64(c). Therefore, for rounding, the power spectrum of the output noise is

$$P_{f'f'}(\omega) = 5\frac{2^{-2B}}{12}\left[\frac{s_2^2|H_2(e^{j\omega})|^2 s_3^2|H_3(e^{j\omega})|^2}{|A_1(e^{j\omega})|^2} + \frac{s_3^2|H_3(e^{j\omega})|^2}{|A_2(e^{j\omega})|^2} + \frac{1}{|A_3(e^{j\omega})|^2}\right], \tag{6.135}$$

and the total output noise variance is

$$\begin{aligned}\sigma_{f'}^2 = 5\frac{2^{-2B}}{12}\Bigg[&\frac{1}{2\pi}\int_{-\pi}^{\pi}\frac{s_2^2|H_2(e^{j\omega})|^2 s_3^2|H_3(e^{j\omega})|^2}{|A_1(e^{j\omega})|^2}d\omega \\ &+ \frac{1}{2\pi}\int_{-\pi}^{\pi}\frac{s_3^2|H_3(e^{j\omega})|^2}{|A_2(e^{j\omega})|^2}d\omega + \frac{1}{2\pi}\int_{-\pi}^{\pi}\frac{1}{|A_3(e^{j\omega})|^2}d\omega\Bigg].\end{aligned} \tag{6.136}$$

If a double-length accumulator is available, it would be necessary to quantize only the sums that are the inputs to the delay elements in Figure 6.64(b). In this case the factor of 5 in Eqs. (6.135) and (6.136) would be changed to 3. Furthermore, if a double-length register were used to implement the delay elements, only the variables $\hat{w}_k[n]$ would have to be quantized, and there would be only one quantization noise source per subsystem. In that case, the factor of 5 in Eqs. (6.135) and (6.136) would be changed to unity.

The scale factors s_k are chosen to avoid overflow at points along the cascade system. We will use the scaling convention of Eq. (6.124). Therefore, the scaling constants are chosen to satisfy

$$s_1 \max_{|\omega|\le\pi}|H_1(e^{j\omega})| < 1, \tag{6.137a}$$

$$s_1 s_2 \max_{|\omega|\le\pi}|H_1(e^{j\omega})H_2(e^{j\omega})| < 1, \tag{6.137b}$$

$$s_1 s_2 s_3 = 0.079459. \tag{6.137c}$$

The last condition ensures that there will be no overflow at the output of the system for unit-amplitude sinusoidal inputs, because the maximum overall gain of the filter is unity. For the coefficients of Table 6.6, the resulting scale factors are $s_1 = 0.186447$, $s_2 = 0.529236$, and $s_3 = 0.805267$.

Equations (6.135) and (6.136) show that the shape of the output noise power spectrum and the total output noise variance depends on the way that zeros and poles are paired to form the 2nd-order sections and on the order of the 2nd-order sections in the cascade form realization. Indeed, it is easily seen that, for N sections, there are $(N!)$ ways to pair the poles and zeros, and there are likewise $(N!)$ ways to order the resulting 2nd-order sections, a total of $(N!)^2$ different systems. In addition, we can choose either direct form I or direct form II (or their transposes) for the implementation of the 2nd-order sections. In our example, this implies that there are 144 different cascade systems to consider, if we wish to find the system with the lowest output noise variance. For five cascaded sections, there would be 57,600 different systems. Clearly, the complete analysis of even low-order systems is a tedious task, since an expression like Eq. (6.136) must be evaluated for each pairing and ordering. Hwang (1974) used dynamic programming and Liu and Peled (1975) used a heuristic approach to reduce the amount of computation.

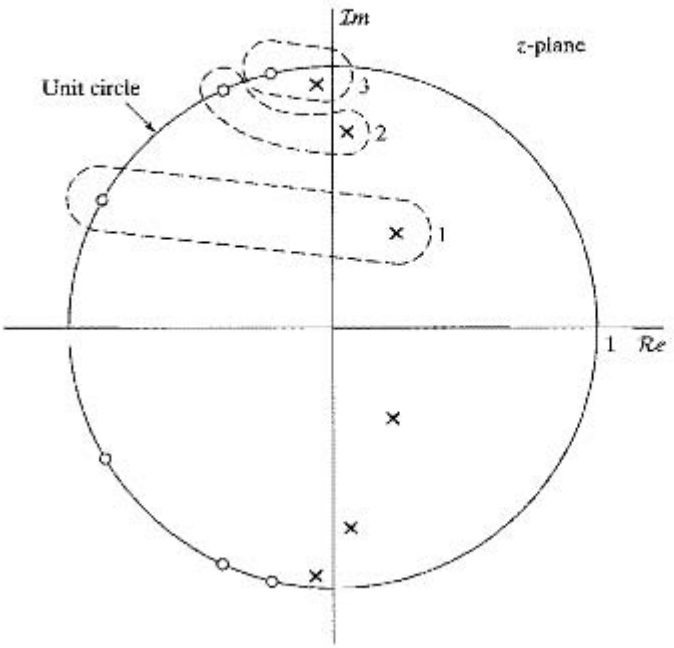


Figure 6.65 Pole–zero plot for 6th-order system of Figure 6.64, showing pairing of poles and zeros.

Even though finding the best pairing and ordering may require computer optimization, Jackson (1970a, 1970b, 1996) found that good results are almost always obtained by applying simple rules of the following form:

1. The pole that is closest to the unit circle should be paired with the zero that is closest to it in the z-plane.
2. Rule 1 should be repeatedly applied until all the poles and zeros have been paired.
3. The resulting 2nd-order sections should be ordered according to either increasing closeness to the unit circle or decreasing closeness to the unit circle.

The pairing rules are based on the observation that subsystems with high peak gain are undesirable because they can cause overflow and because they can amplify quantization noise. Pairing a pole that is close to the unit circle with an adjacent zero tends to reduce the peak gain of that section. These heuristic rules are implemented in design and analysis tools such as the MATLAB function `zp2sos`.

One motivation for rule 3 is suggested by Eq. (6.135). We see that the frequency responses of some of the subsystems appear more than once in the equation for the power spectrum of the output noise. If we do not want the output noise variance spectrum to have a high peak around a pole that is close to the unit circle, then it is advantageous to have the frequency-response component owing to that pole not appear frequently in Eq. (6.135). This suggests moving such "high Q" poles to the beginning of the cascade. On the other hand, the frequency response from the input to a particular node in the flow graph will involve a product of the frequency responses of the subsystems that precede the node. Thus, to avoid excessive reduction of the signal level in the early stages of the cascade, we should place the poles that are close to the unit circle last in order.

Clearly then, the question of ordering hinges on a variety of considerations, including total output noise variance and the shape of the output noise spectrum. Jackson (1970a, 1970b) used L_p norms to quantify the analysis of the pairing-and-ordering problem and gave a much more detailed set of "rules of thumb" for obtaining good results without having to evaluate all possibilities.

The pole–zero plot for the system in our example is shown in Figure 6.65. The paired poles and zeros are circled. In this case, we have chosen to order the sections from least peaked to most peaked frequency response. Figure 6.66 illustrates how the frequency responses of the individual sections combine to form the overall frequency response. Figures 6.66(a)–(c) show the frequency responses of the individual unscaled subsystems. Figures 6.66(d)–(f) show how the overall frequency response is built up. Notice that Figures 6.66(d)–(f) demonstrate that the scaling Eqs. (6.137a)–(6.137c) ensure that the maximum gain from the input to the output of any subsystem is less than unity. The solid curve in Figure 6.67 shows the power spectrum of the output noise for the ordering 123 (least peaked to most peaked). We assume that $B + 1 = 16$ for the plot. Note that the spectrum peaks in the vicinity of the pole that is closest to the unit circle. The dotted curve shows the power spectrum of the output noise when the section order is reversed (i.e., 321). Since section 1 has high gain at low frequencies, the noise spectrum is appreciably larger at low frequencies and slightly lower around the peak. The high Q pole still filters the noise sources of the first section in the cascade, so it still tends to dominate the spectrum. The total noise power for the two orderings turns out to be almost the same in this case.

The example we have just presented shows the complexity of the issues that arise in fixed-point implementations of cascade IIR systems. The parallel form is somewhat simpler because the issue of pairing and ordering does not arise. However, scaling is still required to avoid overflow in individual 2^{nd}-order subsystems and when the outputs of the subsystems are summed to produce the overall output. The techniques that we have developed must therefore be applied for the parallel form as well. Jackson (1996) discusses the analysis of the parallel form in detail and concludes that its total output noise power is typically comparable to that of the best pairings and orderings of the cascade form. Even so, the cascade form is more common, because, for widely used IIR filters such that the zeros of the system function are on the unit circle, the cascade form can be implemented with fewer multipliers and with more control over the locations of the zeros.

6.9.4 Analysis of Direct-Form FIR Systems

Since the direct form I and direct form II IIR systems include the direct form FIR system as a special case (i.e., the case where all coefficients a_k in Figures 6.14 and 6.15 are zero), the results and analysis techniques of Sections 6.9.1 and 6.9.2 apply to FIR systems if we eliminate all reference to the poles of the system function and eliminate the feedback paths in all the signal flow graphs.

The direct form FIR system is simply the discrete convolution

$$y[n] = \sum_{k=0}^{M} h[k]x[n-k]. \tag{6.138}$$

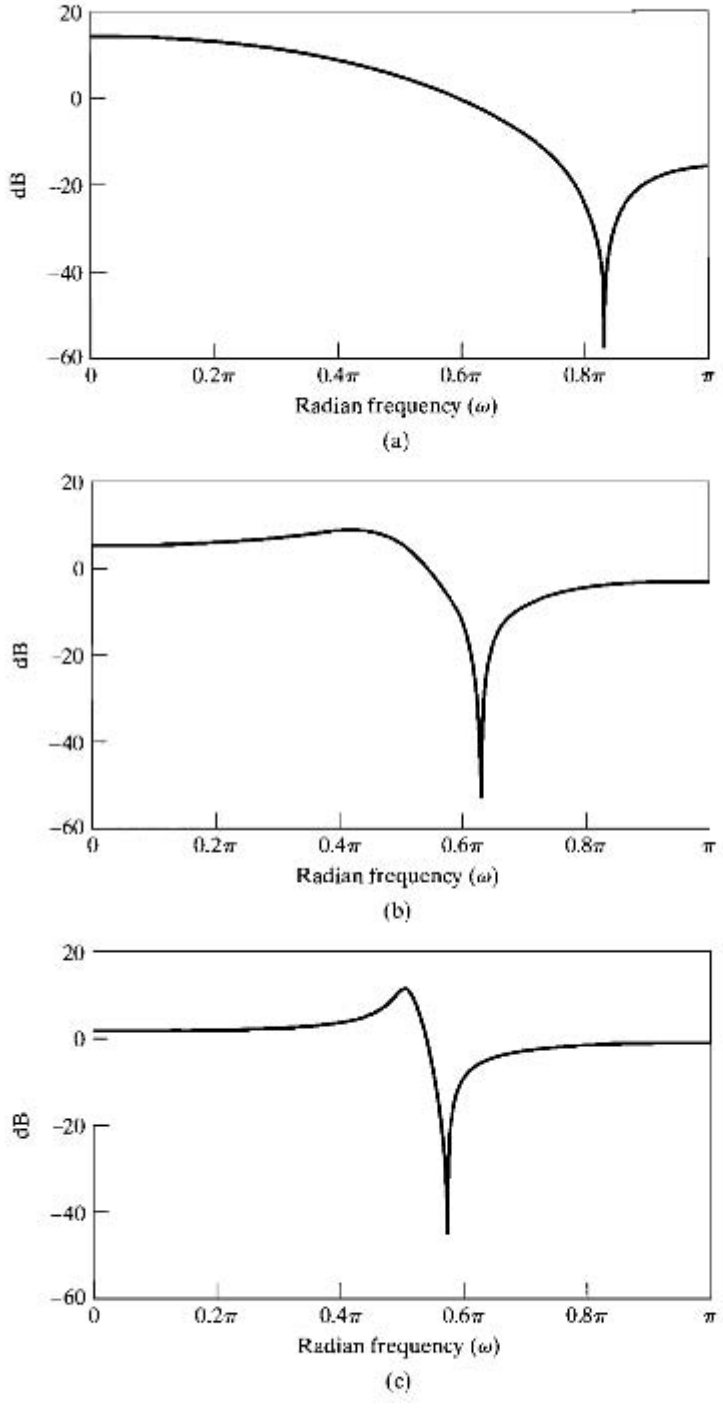
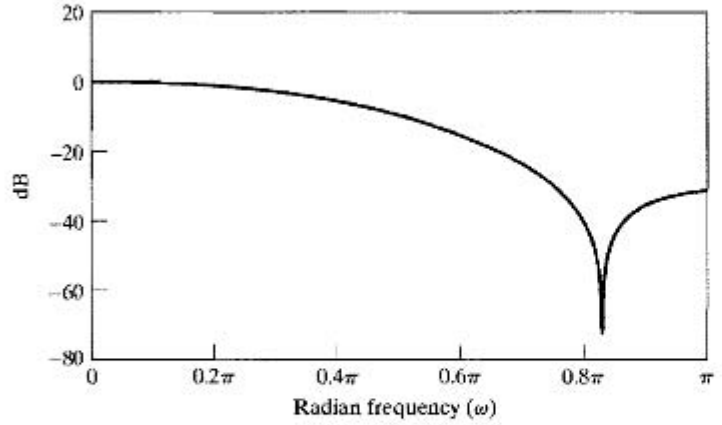
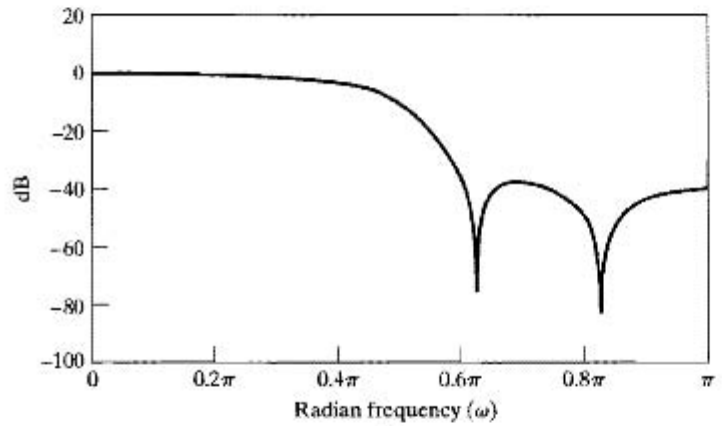


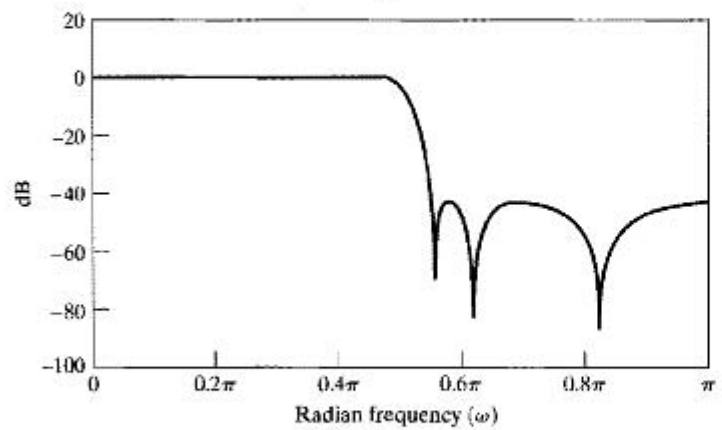
Figure 6.66 Frequency-response functions for example system.
(a) $20 \log_{10} |H_1(e^{j\omega})|$.
(b) $20 \log_{10} |H_2(e^{j\omega})|$.
(c) $20 \log_{10} |H_3(e^{j\omega})|$.



(d)



(e)



(f)

Figure 6.66 *(continued)*
(d) $20 \log_{10} |H_1'(e^{j\omega})|$.
(e) $20 \log_{10} |H_1'(e^{j\omega}) H_2'(e^{j\omega})|$.
(f) $20 \log_{10} |H_1'(e^{j\omega}) H_2'(e^{j\omega}) H_3'(e^{j\omega})|$
$= 20 \log_{10} |H'(e^{j\omega})|$.

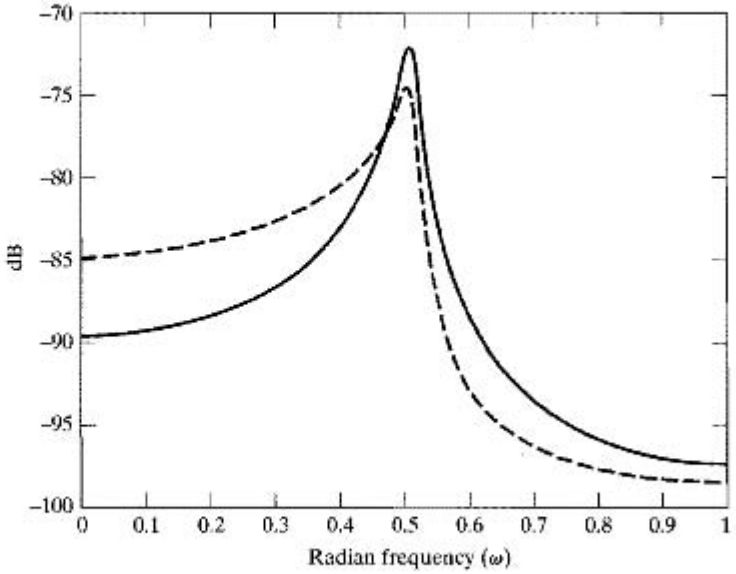


Figure 6.67 Output noise power spectrum for 123 ordering (solid line) and 321 ordering (dashed line) of 2^{nd}-order sections.

Figure 6.68(a) shows the ideal unquantized direct form FIR system, and Figure 6.68(b) shows the linear-noise model for the system, assuming that all products are quantized before additions are performed. The effect is to inject $(M + 1)$ white-noise sources directly at the output of the system, so that the total output noise variance is

$$\sigma_f^2 = (M+1)\frac{2^{-2B}}{12}. \tag{6.139}$$

This is exactly the result we would obtain by setting $N = 0$ and $h_{ef}[n] = \delta[n]$ in Eqs. (6.106) and (6.115). When a double-length accumulator is available, we would need to quantize only the output. Therefore, the factor $(M+1)$ in Eq. (6.139) would be replaced by unity. This makes the double-length accumulator a very attractive hardware feature for implementing FIR systems.

Overflow is also a problem for fixed-point realizations of FIR systems in direct form. For two's-complement arithmetic, we need to be concerned only about the size of the output, since all the other sums in Figure 6.68(b) are partial sums. Thus, the impulse-response coefficients can be scaled to reduce the possibility of overflow. Scaling multipliers can be determined using any of the alternatives discussed in Section 6.9.2. Of course, scaling the impulse response reduces the gain of the system, and therefore the SNR at the output is reduced as discussed in that section.

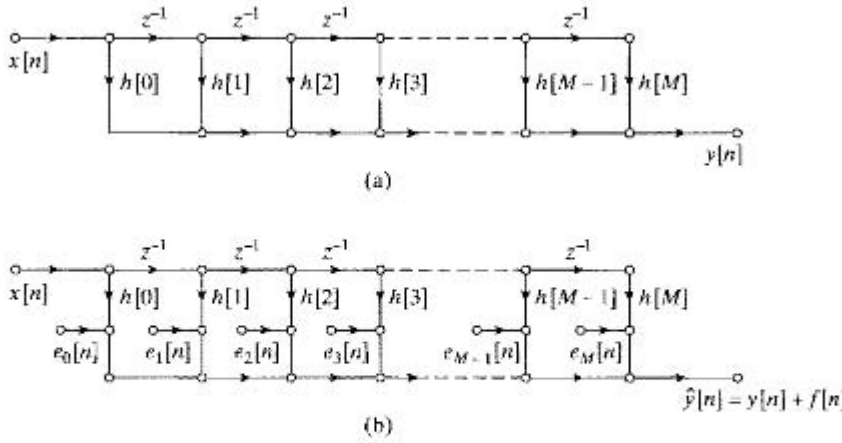


Figure 6.68 Direct form realization of an FIR system. (a) Infinite-precision model. (b) Linear-noise model.

Example 6.14 Scaling Considerations for the FIR System in Section 6.8.5

The impulse-response coefficients for the system in Section 6.8.5 are given in Table 6.5. Simple calculations show, and from Figure 6.54(b) we see, that

$$\sum_{n=0}^{27} |h[n]| = 1.751352,$$

$$\left(\sum_{n=0}^{27} |h[n]|^2 \right)^{1/2} = 0.679442,$$

$$\max_{|\omega| \le \pi} |H(e^{j\omega})| \approx 1.009.$$

These numbers satisfy the ordering relationship in Eq. (6.127). Thus, the system, as given, is scaled so that overflow is theoretically possible for a sinusoidal signal whose amplitude is greater than $1/1.009 = 0.9911$, but even so, overflow is unlikely for most signals. Indeed, since the filter has a linear phase, we can argue that, for wideband signals, since the gain in the passband is approximately unity, and the gain elsewhere is less than unity, the output signal should be smaller than the input signal.

In Section 6.5.3, we showed that linear-phase systems like the one in Example 6.14 can be implemented with about half the number of multiplications of the general FIR system. This is evident from the signal flow graphs of Figures 6.32 and 6.33. In these cases, it should be clear that the output noise variance would be halved if products were quantized before addition. However, the utilization of such structures involves a more complicated indexing algorithm than the direct form. The architecture of most DSP chips combines a double-length accumulator with an efficient pipelined multiply–

accumulate operation and simple looping control to optimize for the case of the direct form FIR system. For this reason, direct form FIR implementations are often most attractive, even compared with IIR filters that meet frequency-response specifications with fewer multiplications, since cascade or parallel structures do not permit long sequences of multiply-accumulate operations.

In Section 6.5.3, we discussed cascade realizations of FIR systems. The results and analysis techniques of Section 6.9.3 apply to these realizations; but for FIR systems with no poles, the pairing and ordering problem reduces to just an ordering problem. As in the case of IIR cascade systems, the analysis of all possible orderings can be very difficult if the system is composed of many subsystems. Chan and Rabiner (1973a, 1973b) studied this problem and found experimentally that the noise performance is relatively insensitive to the ordering. Their results suggest that a good ordering is an ordering for which the frequency response from each noise source to the output is relatively flat and for which the peak gain is small.

6.9.5 Floating-Point Realizations of Discrete-Time Systems

From the preceding discussion, it is clear that the limited dynamic range of fixed-point arithmetic makes it necessary to carefully scale the input and intermediate signal levels in fixed-point digital realizations of discrete-time systems. The need for such scaling can be essentially eliminated by using floating-point numeric representations and floating-point arithmetic.

In floating-point representations, a real number x is represented by the binary number $2^c\hat{x}_M$, where the exponent c of the scale factor is called the *characteristic* and $\hat{x}_M$ is a fractional part called the *mantissa*. Both the characteristic and the mantissa are represented explicitly as fixed-point binary numbers in floating-point arithmetic systems. Floating-point representations provide a convenient means for maintaining both a wide dynamic range and low quantization noise; however, quantization error manifests itself in a somewhat different way. Floating-point arithmetic generally maintains its high accuracy and wide dynamic range by adjusting the characteristic and normalizing the mantissa so that $0.5 \leq \hat{x}_M < 1$. When floating-point numbers are multiplied, their characteristics are added and their mantissas are multiplied. Thus, the mantissa must be quantized. When two floating-point numbers are added, their characteristics must be adjusted to be the same by moving the binary point of the mantissa of the smaller number. Hence, addition results in quantization, too. If we assume that the range of the characteristic is sufficient so that no numbers become larger than 2^c, then quantization affects only the mantissa, but the error in the mantissa is also scaled by 2^c. Thus, a quantized floating-point number is conveniently represented as

$$\hat{x} = x(1 + \varepsilon) = x + \varepsilon x, \tag{6.140}$$

By representing the quantization error as a fraction ε of x, we automatically represent the fact that the quantization error is scaled up and down with the signal level.

The aforementioned properties of floating-point arithmetic complicate the quantization error analysis of floating-point implementations of discrete-time systems. First, noise sources must be inserted both after each multiplication and after each addition. An important consequence is that, in contrast to fixed-point arithmetic, the *order* in

which multiplications and additions are performed can sometimes make a big difference. More important for analysis, we can no longer justify the assumption that the quantization noise sources are white noise and are independent of the signal. In fact, in Eq. (6.140), the noise is expressed explicitly in terms of the signal. Therefore, we can no longer analyze the noise without making assumptions about the nature of the input signal. If the input is assumed to be known (e.g., white noise), a reasonable assumption is that the relative error ε is independent of x and is uniformly distributed white noise.

With these types of assumptions, useful results have been obtained by Sandberg (1967), Liu and Kaneko (1969), Weinstein and Oppenheim (1969), and Kan and Aggarwal (1971). In particular, Weinstein and Oppenheim, comparing floating-point and fixed-point realizations of 1st- and 2nd-order IIR systems, showed that if the number of bits representing the floating-point mantissa is equal to the length of the fixed-point word, then floating-point arithmetic leads to higher SNR at the output. Not surprisingly, the difference was found to be greater for poles close to the unit circle. However, additional bits are required to represent the characteristic, and the greater the desired dynamic range, the more bits are required for the characteristic. Also, the hardware to implement floating-point arithmetic is much more complex than that for fixed-point arithmetic. Therefore, the use of floating-point arithmetic entails an increased word length and increased complexity in the arithmetic unit. Its major advantage is that it essentially eliminates the problem of overflow, and if a sufficiently long mantissa is used, quantization also becomes much less of a problem. This translates into greater simplicity in system design and implementation.

Nowadays, digital filtering of multi-media signals is often implemented on personal computers or workstations that have very accurate floating point numerical representations and high speed arithmetic units. In such cases, the quantization issues discussed in Sections 6.7–6.9 are generally of little or no concern. However, in high volume systems, fixed point arithmetic is generally required to achieve low cost.

6.10 ZERO-INPUT LIMIT CYCLES IN FIXED-POINT REALIZATIONS OF IIR DIGITAL FILTERS

For stable IIR discrete-time systems implemented with infinite-precision arithmetic, if the excitation becomes zero and remains zero for n greater than some value n_0, the output for $n > n_0$ will decay asymptotically toward zero. For the same system, implemented with finite-register-length arithmetic, the output may continue to oscillate indefinitely with a periodic pattern while the input remains equal to zero. This effect is often referred to as *zero-input limit cycle behavior* and is a consequence either of the nonlinear quantizers in the feedback loop of the system or of overflow of additions. The limit cycle behavior of a digital filter is complex and difficult to analyze, and we will not attempt to treat the topic in any general sense. To illustrate the point, however, we will give two simple examples that will show how such limit cycles can arise.

6.10.1 Limit Cycles Owing to Round-off and Truncation

Successive round-off or truncation of products in an iterated difference equation can create repeating patterns. This is illustrated in the following example.

Example 6.15 Limit Cycle Behavior in a 1st-Order System

Consider the 1st-order system characterized by the difference equation

$$y[n] = ay[n-1] + x[n], \qquad |a| < 1. \tag{6.141}$$

The signal flow graph of this system is shown in Figure 6.69(a). Let us assume that the register length for storing the coefficient a, the input $x[n]$, and the filter node variable $y[n-1]$ is 4 bits (i.e., a sign bit to the left of the binary point and 3 bits to the right of the binary point). Because of the finite-length registers, the product $ay[n-1]$ must be rounded or truncated to 4 bits before being added to $x[n]$. The flow graph representing the actual realization based on Eq. (6.141) is shown in Figure 6.69(b). Assuming rounding of the product, the actual output $\hat{y}[n]$ satisfies the nonlinear difference equation

$$\hat{y}[n] = Q[a\hat{y}[n-1]] + x[n], \tag{6.142}$$

where $Q[\cdot]$ represents the rounding operation. Let us assume that $a = 1/2 = 0_\circ 100$ and that the input is $x[n] = (7/8)\delta[n] = (0_\circ 111)\delta[n]$. Using Eq. (6.142), we see that for $n = 0$, $\hat{y}[0] = 7/8 = 0_\circ 111$. To obtain $\hat{y}[1]$, we multiply $\hat{y}[0]$ by a, obtaining the result $\hat{a}y[0] = 0_\circ 011100$, a 7-bit number that must be rounded to 4 bits. This number, 7/16, is exactly halfway between the two 4-bit quantization levels 4/8 and 3/8. If we choose always to round upward in such cases, then $0_\circ 011100$ rounded to 4 bits is $0_\circ 100 = 1/2$. Since $x[1] = 0$, it follows that $\hat{y}[1] = 0_\circ 100 = 1/2$. Continuing to iterate the difference equation gives $\hat{y}[2] = Q[a\hat{y}[1]] = 0_\circ 010 = 1/4$ and $\hat{y}[3] = 0_\circ 001 = 1/8$. In both these cases, no rounding is necessary. However, to obtain $\hat{y}[4]$, we must round the 7-bit number $a\hat{y}[3] = 0_\circ 000100$ to $0_\circ 001$. The same result is obtained for all values of $n \geq 3$. The output sequence for this example is shown in Figure 6.70(a). If $a = -1/2$, we can carry out the preceding computation again to demonstrate that the output is as shown in Figure 6.70(b). Thus, because of rounding of the product $a\hat{y}[n-1]$, the output reaches a constant value of 1/8 when $a = 1/2$ and a periodic steady-state oscillation between $+1/8$ and $-1/8$ when $a = -1/2$. These are periodic outputs similar to those that would be obtained from a 1st-order pole at $z = \pm 1$ instead of at $z = \pm 1/2$.

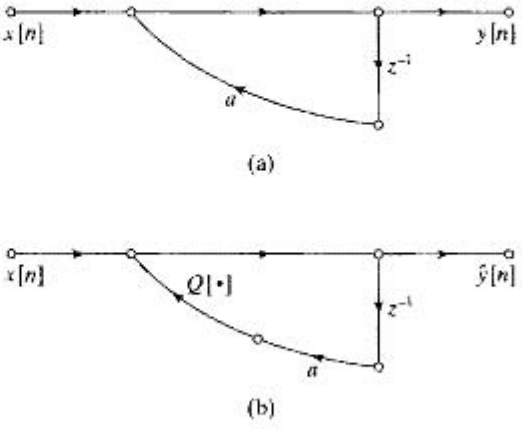


Figure 6.69 1st-order IIR system. (a) Infinite-precision linear system. (b) Nonlinear system due to quantization.

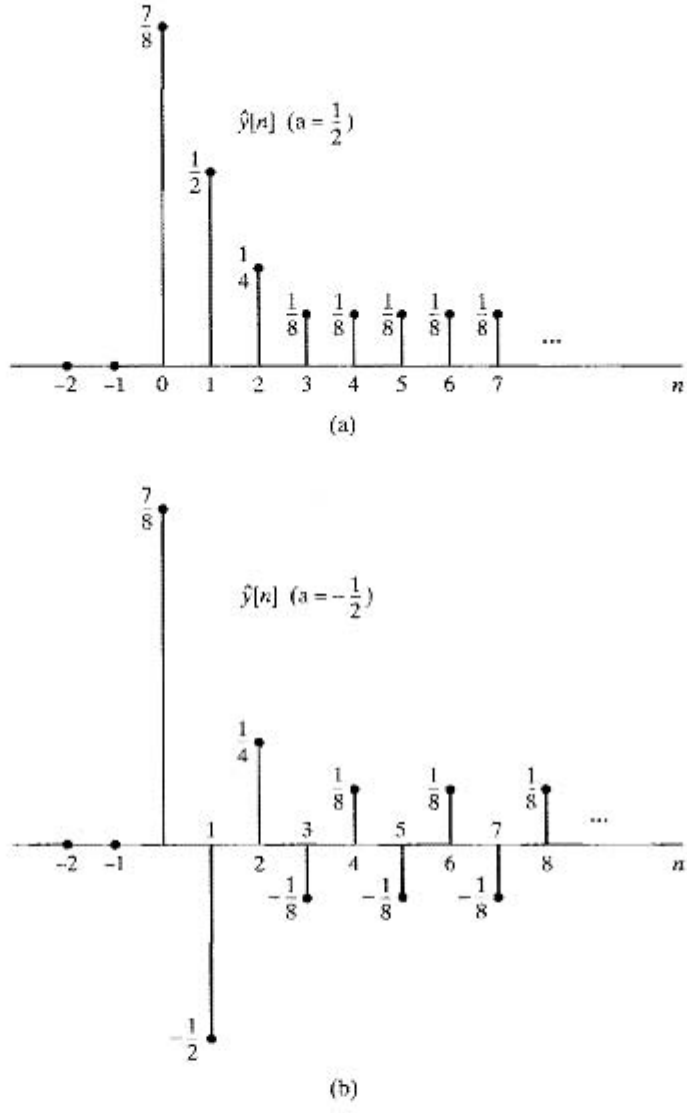


Figure 6.70 Response of the 1st-order system of Figure 6.69 to an impulse. (a) $a = \frac{1}{2}$. (b) $a = -\frac{1}{2}$.

When $a = +1/2$, the period of the oscillation is 1, and when $a = -1/2$, the period of oscillation is 2. Such steady-state periodic outputs are called *limit cycles*, and their existence was first noted by Blackman (1965), who referred to the amplitude intervals to which such limit cycles are confined as *dead bands*. In this case, the dead band is $-2^{-B} \leq \hat{y}[n] \leq +2^{-B}$, where $B = 3$.

The foregoing example has illustrated that a zero-input limit cycle can result from rounding in a 1st-order IIR system. Similar results can be demonstrated for truncation. 2nd-order systems can also exhibit limit cycle behavior. In the case of parallel realizations

of higher-order systems, the outputs of the individual 2^{nd}-order systems are independent when the input is zero. In this case, one or more of the 2^{nd}-order sections could contribute a limit cycle to the output sum. In the case of cascade realizations, only the first section has zero input; succeeding sections may exhibit their own characteristic limit cycle behavior, or they may appear to be simply filtering the limit cycle output of a previous section. For higher-order systems realized by other filter structures, the limit cycle behavior becomes more complex, as does its analysis.

In addition to giving an understanding of limit cycle effects in digital filters, the preceding results are useful when the zero-input limit cycle response of a system is the desired output. This is the case, for example, when one is concerned with digital sine wave oscillators for signal generation or for the generation of coefficients for calculation of the discrete Fourier transform.

6.10.2 Limit Cycles Owing to Overflow

In addition to the classes of limit cycles discussed in the preceding section, a more severe type of limit cycle can occur owing to overflow. The effect of overflow is to insert a gross error in the output, and in some cases the filter output thereafter oscillates between large-amplitude limits. Such limit cycles have been referred to as *overflow oscillation*. The problem of oscillations caused by overflow is discussed in detail by Ebert et al. (1969). Overflow oscillations are illustrated by the following example.

Example 6.16 Overflow Oscillations in a 2^{nd}-Order System

Consider a 2^{nd}-order system realized by the difference equation

$$\hat{y}[n] = x[n] + Q[a_1\hat{y}[n-1]] + Q[a_2\hat{y}[n-2]], \tag{6.143}$$

where $Q[\cdot]$ represents two's-complement rounding with a word length of 3 bits plus 1 bit for the sign. Overflow can occur with two's-complement addition of the rounded products. Suppose that $a_1 = 3/4 = 0_\diamond 110$ and $a_2 = -3/4 = 1_\diamond 010$, and assume that $x[n]$ remains equal to zero for $n \geq 0$. Furthermore, assume that $\hat{y}[-1] = 3/4 = 0_\diamond 110$ and $\hat{y}[-2] = -3/4 = 1_\diamond 010$. Now the output at sample $n = 0$ is

$$\hat{y}[0] = 0_\diamond 110 \times 0_\diamond 110 + 1_\diamond 010 \times 1_\diamond 010.$$

If we evaluate the products using two's-complement arithmetic, we obtain

$$\hat{y}[0] = 0_\diamond 100100 + 0_\diamond 100100,$$

and if we choose to round upward when a number is halfway between two quantization levels, the result of two's-complement addition is

$$\hat{y}[0] = 0_\diamond 101 + 0_\diamond 101 = 1_\diamond 010 = -\frac{3}{4}.$$

In this case the binary carry overflows into the sign bit, thus changing the positive sum into a negative number. Repeating the process gives

$$\hat{y}[1] = 1_\diamond 011 + 1_\diamond 011 = 0_\diamond 110 = \frac{3}{4}.$$

The binary carry resulting from the sum of the sign bits is lost, and the negative sum is mapped into a positive number. Clearly, $\hat{y}[n]$ will continue to oscillate between $+3/4$ and $-3/4$ until an input is applied. Thus, $\hat{y}[n]$ has entered a periodic limit cycle with a period of 2 and an amplitude of almost the full-scale amplitude of the implementation.

The preceding example illustrates how overflow oscillations occur. Much more complex behavior can be exhibited by higher-order systems, and other frequencies can occur. Some results are available for predicting when overflow oscillations can be supported by a difference equation (see Ebert et al., 1969). Overflow oscillations can be avoided by using the saturation overflow characteristic of Figure 6.45(b) (see Ebert et al., 1969).

6.10.3 Avoiding Limit Cycles

The possible existence of a zero-input limit cycle is important in applications where a digital filter is to be in continuous operation, since it is generally desired that the output approach zero when the input is zero. For example, suppose that a speech signal is sampled, filtered by a digital filter, and then converted back to an acoustic signal using a D/A converter. In such a situation it would be very undesirable for the filter to enter a periodic limit cycle whenever the input is zero, since the limit cycle would produce an audible tone.

One approach to the general problem of limit cycles is to seek structures that do not support limit cycle oscillations. Such structures have been derived by using state-space representations (see Barnes and Fam, 1977; Mills, Mullis and Roberts, 1978) and concepts analogous to passivity in analog systems (see Rao and Kailath, 1984; Fettweis, 1986). However, these structures generally require more computation than an equivalent cascade or parallel form implementation. By adding more bits to the computational wordlength, we can generally avoid overflow. Similarly, since round-off limit cycles usually are limited to the least significant bits of the binary word, additional bits can be used to reduce the effective amplitude of the limit cycle. Also, Claasen et al. (1973) showed that if a double-length accumulator is used so that quantization occurs after the accumulation of products, then limit cycles owing to round-off are much less likely to occur in 2^{nd}-order systems. Thus, the trade-off between word length and computational algorithm complexity arises for limit cycles just as it does for coefficient quantization and round-off noise.

Finally, it is important to point out that zero-input limit cycles due to both overflow and round-off are a phenomenon unique to IIR systems: FIR systems cannot support zero-input limit cycles, because they have no feedback paths. The output of an FIR system will be zero no later than $(M + 1)$ samples after the input goes to zero and remains there. This is a major advantage of FIR systems in applications wherein limit cycle oscillations cannot be tolerated.

6.11 SUMMARY

In this chapter, we have considered many aspects of the problem of implementing an LTI discrete-time system. The first half of the chapter was devoted to basic implementation structures. After introducing block diagram and signal flow graphs as pictorial representations of difference equations, we discussed a number of basic structures for

IIR and FIR discrete-time systems. These included the direct form I, direct form II, cascade form, parallel form, lattice form, and transposed version of all the basic forms. We showed that these forms are all equivalent when implemented with infinite-precision arithmetic. However, the different structures are most significant in the context of finite-precision implementations. Therefore, the remainder of the chapter addressed problems associated with finite precision or quantization in fixed-point digital implementations of the basic structures.

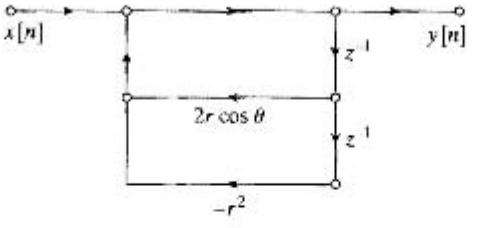
We began the discussion of finite precision effects with a brief review of digital number representation and an overview showing that the quantization effects that are important in sampling (discussed in Chapter 4) are also important in representing the coefficients of a discrete-time system and in implementing systems using finite-precision arithmetic. We illustrated the effect of quantization of the coefficients of a difference equation through several examples. This issue was treated independently of the effects of finite-precision arithmetic, which we showed introduces nonlinearity into the system. We demonstrated that in some cases this nonlinearity was responsible for limit cycles that may persist after the input to a system has become zero. We also showed that quantization effects can be modeled in terms of independent random white-noise sources that are injected internally into the flow graph. Such linear-noise models were developed for the direct form structures and for the cascade structure. In all of our discussion of quantization effects, the underlying theme was the conflict between the desire for fine quantization and the need to maintain a wide range of signal amplitudes. We saw that in fixed-point implementations, one can be improved at the expense of the other, but to improve one while leaving the other unaffected requires that we increase the number of bits used to represent coefficients and signal amplitudes. This can be done either by increasing the fixed-point word length or by adopting a floating-point representation.

Our discussion of quantization effects serves two purposes. First, we developed several results that can be useful in guiding the design of practical implementations. We found that quantization effects depend greatly on the structure used and on the specific parameters of the system to be implemented, and even though simulation of the system is generally necessary to evaluate its performance, many of the results discussed are useful in making intelligent decisions in the design process. A second, equally important purpose of this part of the chapter was to illustrate a style of analysis that can be applied in studying quantization effects in a variety of digital signal-processing algorithms. The examples of the chapter indicate the types of assumptions and approximations that are commonly made in studying quantization effects. In Chapter 9, we will apply the analysis techniques developed here to the study of quantization in the computation of the discrete Fourier transform.

Problems

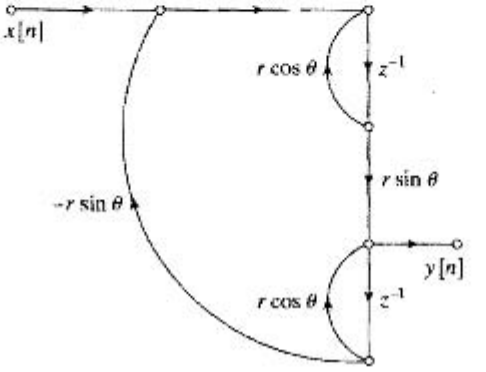
Basic Problems with Answers

6.1. Determine the system function of the two flow graphs in Figure P6.1, and show that they have the same poles.



Network 1

(a)



Network 2

(b)

Figure P6.1

6.2. The signal flow graph of Figure P6.2 represents a linear difference equation with constant coefficients. Determine the difference equation that relates the output $y[n]$ to the input $x[n]$.

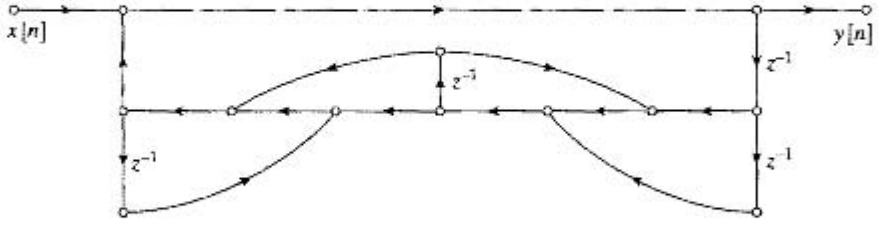
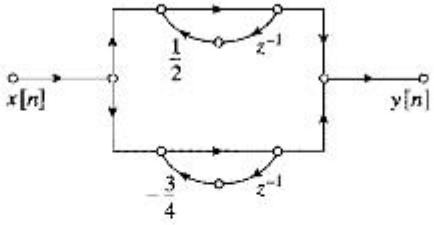
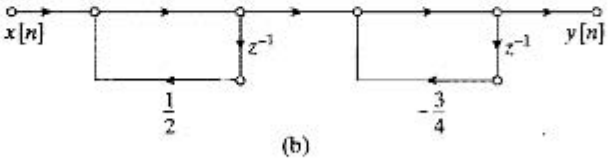


Figure P6.2

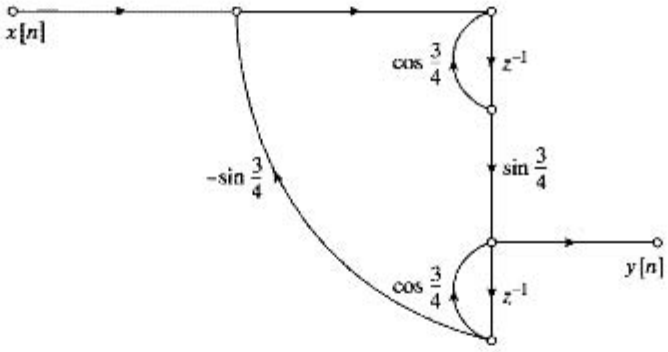
6.3. Figure P6.3 shows six systems. Determine which one of the last five, (b)–(f), has the same system function as (a). You should be able to eliminate some of the possibilities by inspection.



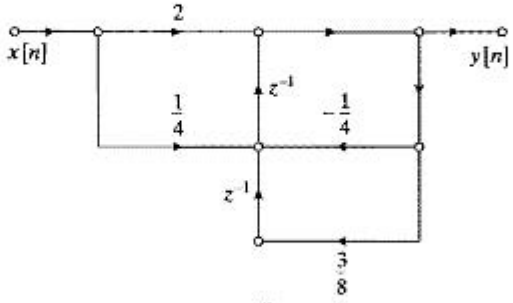
(a)



(b)

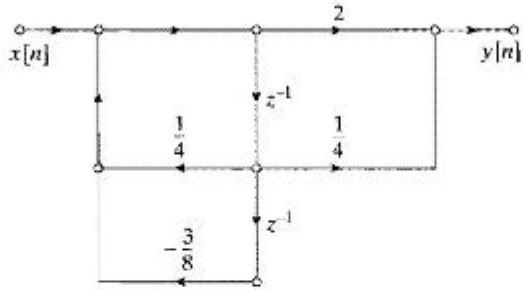


(c)

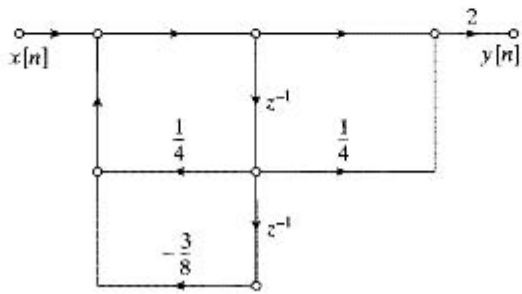


(d)

Figure P6.3



(e)



(f)

Figure P6.3 *(continued)*

6.4. Consider the system in Figure P6.3(d).

(a) Determine the system function relating the z-transforms of the input and output.

(b) Write the difference equation that is satisfied by the input sequence $x[n]$ and the output sequence $y[n]$.

6.5. An LTI system is realized by the flow graph shown in Figure P6.5.

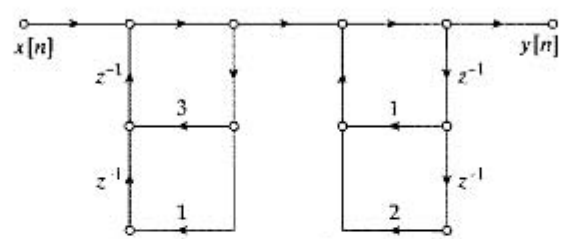


Figure P6.5

(a) Write the difference equation relating $x[n]$ and $y[n]$ for this flow graph.

(b) What is the system function of the system?

(c) In the realization of Figure P6.5, how many real multiplications and real additions are required to compute each sample of the output? (Assume that $x[n]$ is real, and assume that multiplication by 1 does not count in the total.)

(d) The realization of Figure P6.5 requires four storage registers (delay elements). Is it possible to reduce the number of storage registers by using a different structure? If so, draw the flow graph; if not, explain why the number of storage registers cannot be reduced.

6.6. Determine the impulse response of each of the systems in Figure P6.6.

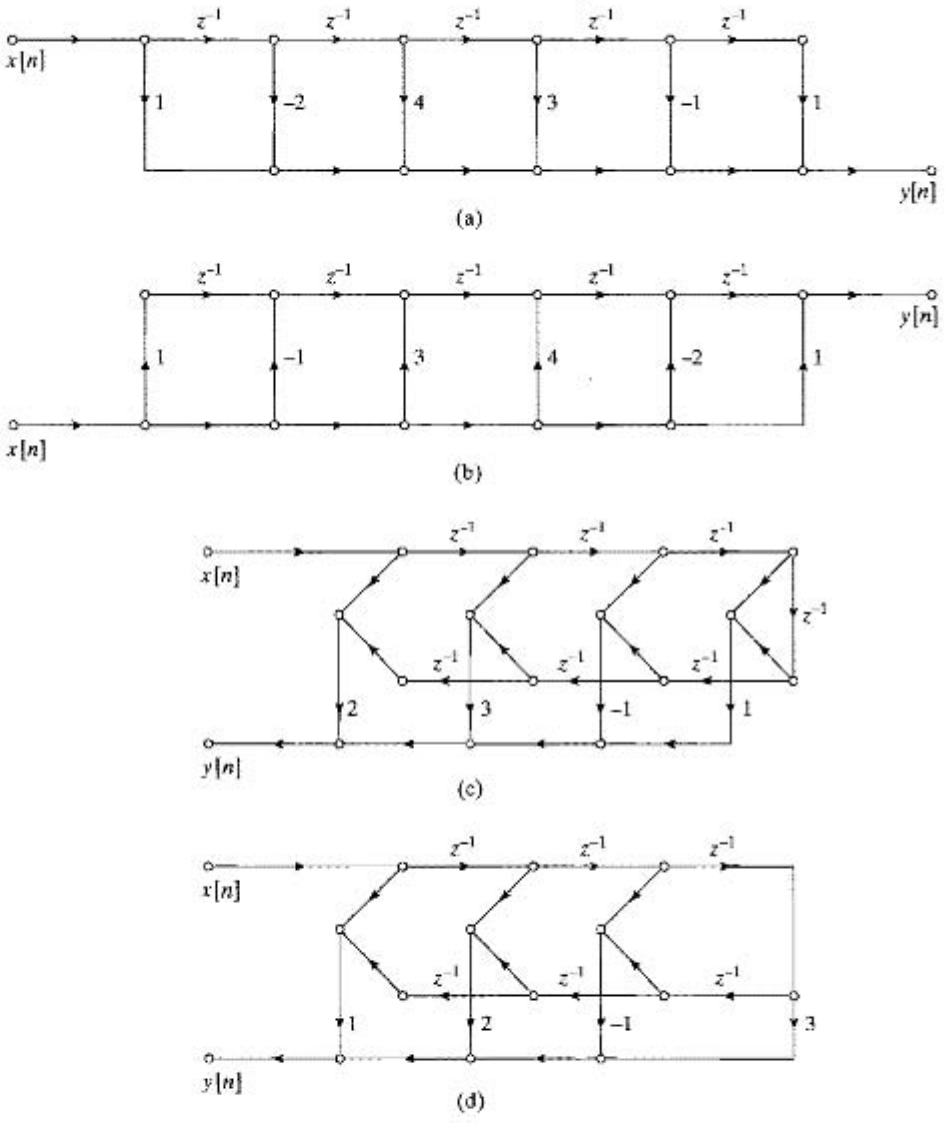


Figure P6.6

6.7. Let $x[n]$ and $y[n]$ be sequences related by the following difference equation:

$$y[n] - \frac{1}{4}y[n-2] = x[n-2] - \frac{1}{4}x[n].$$

Draw a direct form II signal flow graph for the causal LTI system corresponding to this difference equation.

6.8. The signal flow graph in Figure P6.8 represents an LTI system. Determine a difference equation that gives a relationship between the input $x[n]$ and the output $y[n]$ of this system. As usual, all branches of the signal flow graph have unity gain unless specifically indicated otherwise.

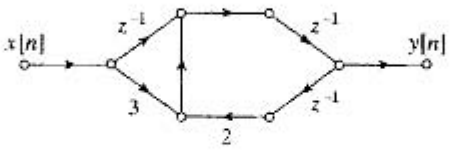


Figure P6.8

6.9. Figure P6.9 shows the signal flow graph for a causal discrete-time LTI system. Branches without gains explicitly indicated have a gain of unity.

(a) By tracing the path of an impulse through the flowgraph, determine $h[1]$, the impulse response at $n = 1$.

(b) Determine the difference equation relating $x[n]$ and $y[n]$.

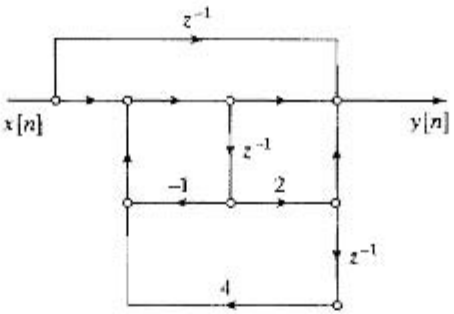


Figure P6.9

6.10. Consider the signal flow graph shown in Figure P6.10.

(a) Using the node variables indicated, write the set of difference equations represented by this flow graph.

(b) Draw the flow graph of an equivalent system that is the cascade of two 1st-order systems.

(c) Is the system stable? Explain.

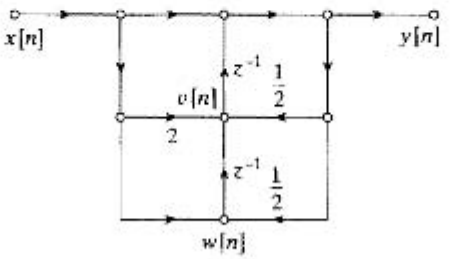


Figure P6.10

6.11. Consider a causal LTI system with impulse response $h[n]$ and system function

$$H(z) = \frac{(1-2z^{-1})(1-4z^{-1})}{z\left(1-\frac{1}{2}z^{-1}\right)}$$

(a) Draw a direct form II flow graph for the system.
(b) Draw the transposed form of the flow graph in part (a).

6.12. For the LTI system described by the flow graph in Figure P6.12, determine the difference equation relating the input $x[n]$ to the output $y[n]$.

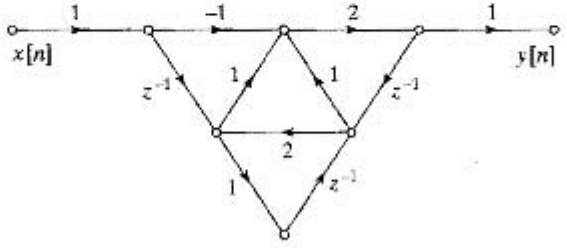


Figure P6.12

6.13. Draw the signal flow graph for the direct form I implementation of the LTI system with system function

$$H(z) = \frac{1-\frac{1}{2}z^{-2}}{1-\frac{1}{4}z^{-1}-\frac{1}{8}z^{-2}}.$$

6.14. Draw the signal flow graph for the direct form II implementation of the LTI system with system function

$$H(z) = \frac{1+\frac{5}{6}z^{-1}+\frac{1}{6}z^{-2}}{1-\frac{1}{2}z^{-1}-\frac{1}{2}z^{-2}}.$$

6.15. Draw the signal flow graph for the transposed direct form II implementation of the LTI system with system function

$$H(z) = \frac{1-\frac{7}{6}z^{-1}+\frac{1}{6}z^{-2}}{1+z^{-1}+\frac{1}{2}z^{-2}}.$$

6.16. Consider the signal flow graph shown in Figure P6.16.

(a) Draw the signal flow graph that results from applying the transposition theorem to this signal flow graph.
(b) Confirm that the transposed signal flow graph that you found in (a) has the same system function $H(z)$ as the original system in the figure.

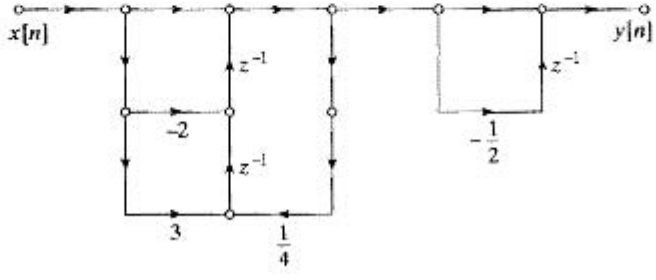


Figure P6.16

6.17. Consider the causal LTI system with system function

$$H(z) = 1 - \frac{1}{3}z^{-1} + \frac{1}{6}z^{-2} + z^{-3}.$$

(a) Draw the signal flow graph for the direct form implementation of this system.
(b) Draw the signal flow graph for the transposed direct form implementation of the system.

6.18. For some nonzero choices of the parameter a, the signal flow graph in Figure P6.18 can be replaced by a 2nd-order direct form II signal flow graph implementing the same system function. Give one such choice for a and the system function $H(z)$ that results.

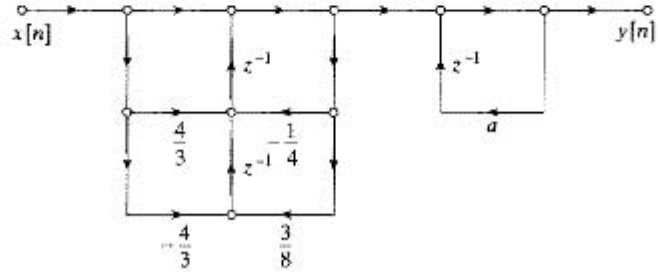


Figure P6.18

6.19. Consider the causal LTI system with the system function

$$H(z) = \frac{2 - \frac{8}{3}z^{-1} - 2z^{-2}}{\left(1 - \frac{1}{3}z^{-1}\right)\left(1 + \frac{2}{3}z^{-1}\right)}.$$

Draw a signal flow graph that implements this system as a parallel combination of 1st-order transposed direct form II sections.

6.20. Draw a signal flow graph implementing the system function

$$H(z) = \frac{(1 + (1 - j/2)z^{-1})(1 + (1 + j/2)z^{-1})}{(1 + (j/2)z^{-1})(1 - (j/2)z^{-1})(1 - (1/2)z^{-1})(1 - 2z^{-1})}$$

as a cascade of 2nd-order transposed direct form II sections with real coefficients.

Basic Problems

6.21. For many applications, it is useful to have a system that will generate a sinusoidal sequence. One possible way to do this is with a system whose impulse response is $h[n] = e^{j\omega_0 n}u[n]$. The real and imaginary parts of $h[n]$ are therefore $h_r[n] = (\cos\omega_0 n)u[n]$ and $h_i[n] = (\sin\omega_0 n)u[n]$, respectively.

In implementing a system with a complex impulse response, the real and imaginary parts are distinguished as separate outputs. By first writing the complex difference equation required to produce the desired impulse response and then separating it into its real and imaginary parts, draw a flow graph that will implement this system. The flow graph that you draw should have only real coefficients. This implementation is sometimes called the *coupled form oscillator*, since, when the input is excited by an impulse, the outputs are sinusoidal.

6.22. For the system function

$$H(z) = \frac{1 + 2z^{-1} + z^{-2}}{1 - 0.75z^{-1} + 0.125z^{-2}},$$

draw the flow graphs of all possible realizations for this system as cascades of 1st-order systems.

6.23. We want to implement a causal system $H(z)$ with the pole-zero diagram shown in Figure P6.23. For all parts of this problem, z_1, z_2, p_1, and p_2 are real, and a gain constant that is independent of frequency can be absorbed into a gain coefficient in the output branch of each flow graph.

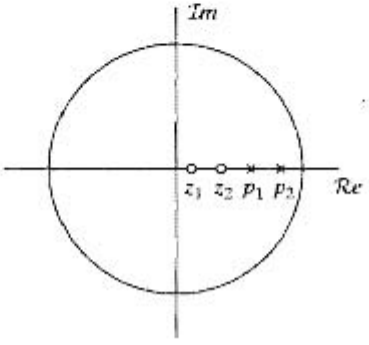


Figure P6.23

(a) Draw the flow graph of the direct form II implementation. Determine an expression for each of the branch gains in terms of the variables z_1, z_2, p_1, and p_2.

(b) Draw the flow graph of an implementation as a cascade of 2nd-order direct form II sections. Determine an expression for each of the branch gains in terms of the variables z_1, z_2, p_1, and p_2.

(c) Draw the flow graph of a parallel form implementation with 1st-order direct form II sections. Specify a system of linear equations that can be solved to express the branch gains in terms of the variables z_1, z_2, p_1, and p_2.

6.24. Consider a causal LTI system whose system function is

$$H(z) = \frac{1 - \frac{3}{10}z^{-1} + \frac{1}{3}z^{-2}}{\left(1 - \frac{4}{5}z^{-1} + \frac{2}{3}z^{-2}\right)\left(1 + \frac{1}{5}z^{-1}\right)} = \frac{\frac{1}{2}}{1 - \frac{4}{5}z^{-1} + \frac{2}{3}z^{-2}} + \frac{\frac{1}{2}}{1 + \frac{1}{5}z^{-1}}$$

(a) Draw the signal flow graphs for implementations of the system in each of the following forms:

(i) Direct form I
(ii) Direct form II
(iii) Cascade form using 1st- and 2nd-order direct form II sections
(iv) Parallel form using 1st- and 2nd-order direct form I sections
(v) Transposed direct form II.

(b) Write the difference equations for the flow graph of part (v) in (a) and show that this system has the correct system function.

6.25. A causal LTI system is defined by the signal flow graph shown in Figure P6.25, which represents the system as a cascade of a 2nd-order system with a 1st-order system.

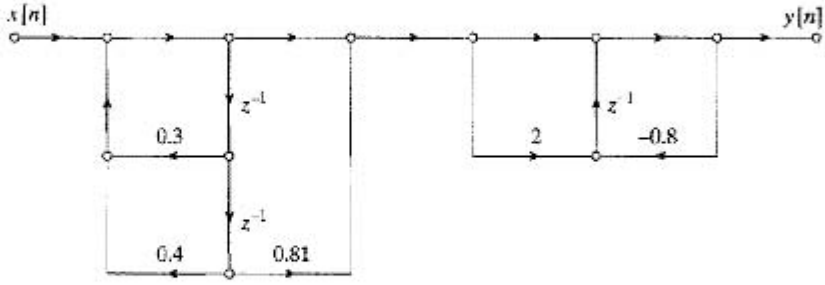


Figure P6.25

(a) What is the system function of the overall cascade system?
(b) Is the overall system stable? Explain briefly.
(c) Is the overall system a minimum-phase system? Explain briefly.
(d) Draw the signal flow graph of a transposed direct form II implementation of this system.

6.26. A causal LTI system has system function given by the following expression:

$$H(z) = \frac{1}{1 - z^{-1}} + \frac{1 - z^{-1}}{1 - z^{-1} + 0.8z^{-2}}.$$

(a) Is this system stable? Explain briefly.
(b) Draw the signal flow graph of a parallel form implementation of this system.
(c) Draw the signal flow graph of a cascade form implementation of this system as a cascade of a 1st-order system and a 2nd-order system. Use a transposed direct form II implementation for the 2nd-order system.

6.27. An LTI system with system function

$$H(z) = \frac{0.2(1 + z^{-1})^6}{\left(1 - 2z^{-1} + \frac{7}{8}z^{-2}\right)\left(1 + z^{-1} + \frac{1}{2}z^{-2}\right)\left(1 - \frac{1}{2}z^{-1} + z^{-2}\right)}$$

is to be implemented using a flow graph of the form shown in Figure P6.27.

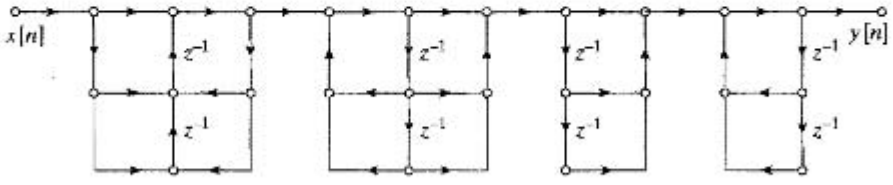


Figure P6.27

(a) Fill in all the coefficients in the diagram of Figure P6.27. Is your solution unique?

(b) Define appropriate node variables in Figure P6.27, and write the set of difference equations that is represented by the flow graph.

6.28. (a) Determine the system function, $H(z)$, from $x[n]$ to $y[n]$ for the flow graph shown in Figure P6.28-1 (note that the location where the diagonal lines criss-cross is not a single node).

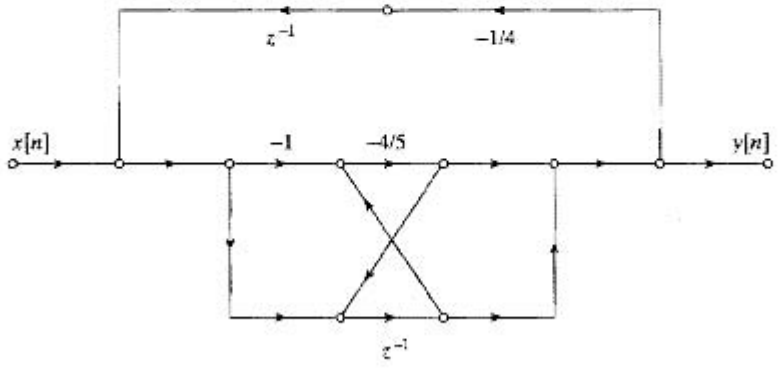


Figure P6.28-1

(b) Draw the direct form (I and II) flow graph of systems having the system function $H(z)$.

(c) Design $H_1(z)$ such that $H_2(z)$ in Figure P6.28-2 has a causal stable inverse and $|H_2(e^{j\omega})| = |H(e^{j\omega})|$. Note: Zero-pole cancellation is permitted.

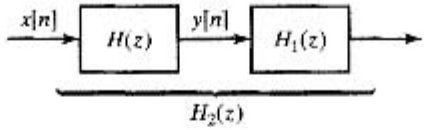


Figure P6.28-2

(d) Draw the transposed direct form II flow graph for $H_2(z)$.

6.29. (a) Determine the system function $H(z)$ relating the input $x[n]$ to the output $y[n]$ for the FIR lattice filter depicted in Figure P6.29.

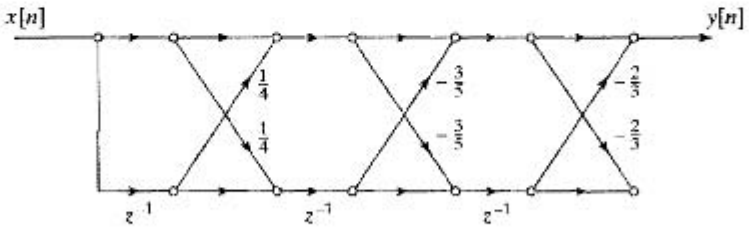


Figure P6.29

(b) Draw the lattice filter structure for the all-pole filter $1/H(z)$.

6.30. Determine and draw the lattice filter implementation of the following causal all-pole system function:

$$H(z) = \frac{1}{1 + \frac{3}{2}z^{-1} - z^{-2} + \frac{3}{4}z^{-3} + 2z^{-4}}$$

Is the system stable?

6.31. An IIR lattice filter is shown in Figure P6.31.

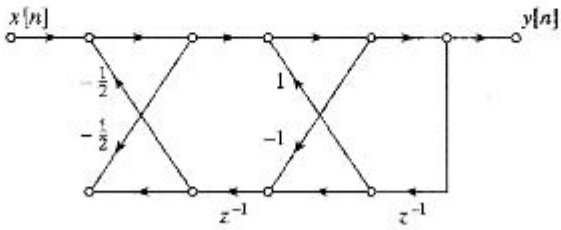


Figure P6.31

(a) By tracing the path of an impulse through the flowgraph, determine $y[1]$ for input $x[n] = \delta[n]$.
(b) Determine a flow graph for the corresponding inverse filter.
(c) Determine the transfer function for the IIR filter in Figure P6.31.

6.32. The flow graph shown in Figure P6.32 is an implementation of a causal, LTI system.

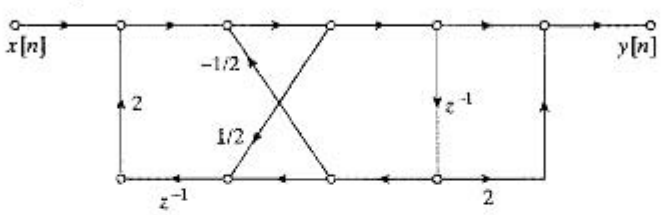


Figure P6.32

(a) Draw the transpose of the signal flow graph.
(b) For either the original system or its transpose, determine the difference equation relating the input $x[n]$ to the output $y[n]$. (*Note:* The difference equations will be the same for both structures.)
(c) Is the system BIBO stable?
(d) Determine $y[2]$ if $x[n] = (1/2)^n u[n]$.

Advanced Problems

6.33. Consider the LTI system represented by the FIR lattice structure in Figure P6.33-1.

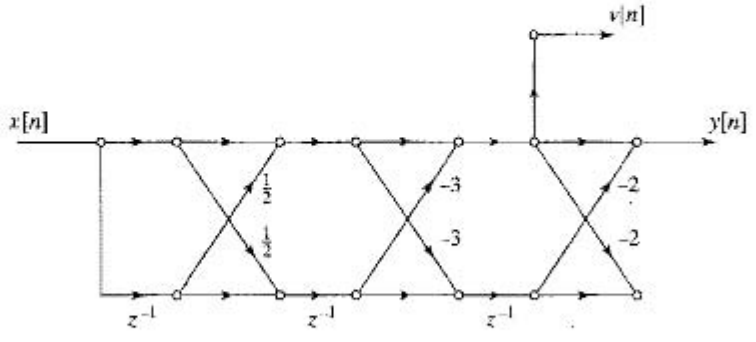


Figure P6.33-1

(a) Determine the system function from the input $x[n]$ to the output $v[n]$ (NOT $y[n]$).

(b) Let $H(z)$ be the system function from the input $x[n]$ to the output $y[n]$, and let $g[n]$ be the result of expanding the associated impulse response $h[n]$ by 2 as shown in Figure P6.33-2.

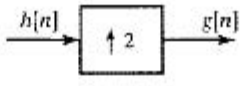


Figure P6.33-2

The impulse response $g[n]$ defines a new system with system function $G(z)$. We would like to implement $G(z)$ using an FIR lattice structure. Determine the k-parameters necessary for an FIR lattice implementation of $G(z)$. *Note:* You should think carefully before diving into a long calculation.

6.34. Figure P6.34-1 shows an impulse response $h[n]$, specified as

$$h[n] = \begin{cases} \left(\frac{1}{2}\right)^{n/4} u[n], & \text{for } n \text{ an integer multiple of 4} \\ \text{constant in between as indicated} \end{cases}$$

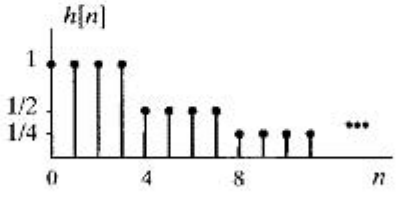


Figure P6.34-1

(a) Determine a choice for $h_1[n]$ and $h_2[n]$ such that

$$h[n] = h_1[n] * h_2[n],$$

where $h_1[n]$ is an FIR filter and where $h_2[n] = 0$ for $n/4$ not an integer. Is $h_2[n]$ an FIR or IIR filter?

(b) The impulse response $h[n]$ is to be used in a downsampling system as indicated in Figure P6.34-2.

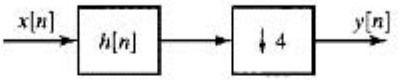


Figure P6.34-2

Draw a flow graph implementation of the system in Figure P6.34-2 that requires the minimum number of nonzero and nonunity coefficient multipliers. You may use unit delay elements, coefficient multipliers, adders and compressors. (Multiplication by a zero or a one does not require a multiplier.)

(c) For your system, state how many multiplications per input sample and per output sample are required, giving a brief explanation.

6.35. Consider the system shown in Figure P6.35-1.

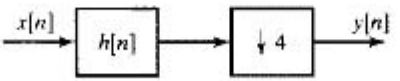


Figure P6.35-1

We want to implement this system using the polyphase structure shown in Figure P6.35-2.

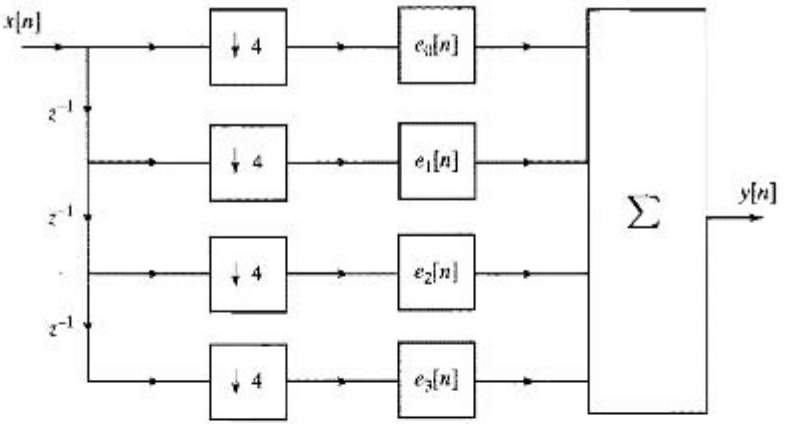


Figure P6.35-2 Polyphase structure of the system.

For parts (a) and (b) only, assume $h[n]$ is defined in Figure P6.35-3

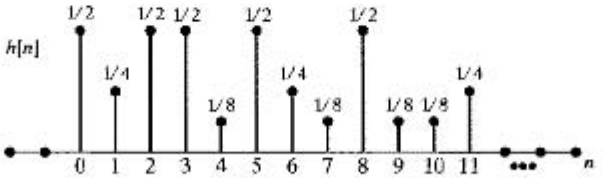


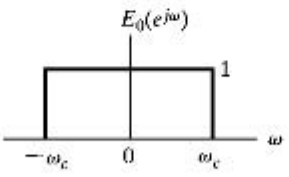
Figure P6.35-3

($h[n] = 0$ for all $n < 0$ and $n \geq 12$).

(a) Give the sequences $e_0[n]$, $e_1[n]$, $e_2[n]$, and $e_3[n]$ that result in a correct implementation.

(b) We want to minimize the total number of multiplies per output sample for the implementation of the structure in Figure P6.35-2. Using the appropriate choice of $e_0[n]$, $e_1[n]$, $e_2[n]$, and $e_3[n]$ from part (a), determine the minimum number of multiplies per output sample for the overall system. Also, determine the minimum number of multiplies per input sample for the overall system. Explain.

(c) Instead of using the sequences $e_0[n]$, $e_1[n]$, $e_2[n]$, and $e_3[n]$ identified in part (a), now assume that $E_0(e^{j\omega})$ and $E_2(e^{j\omega})$ the DTFTs of $e_0[n]$ and $e_2[n]$, respectively, are as given in Figure P6.35-4, and $E_1(e^{j\omega}) = E_3(e^{j\omega}) = 0$.

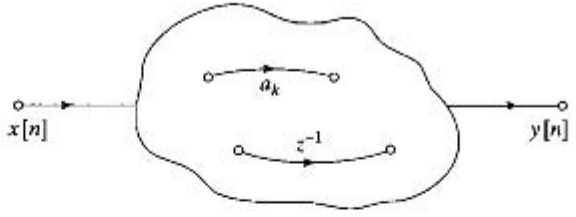


$$E_2(e^{j\omega}) = \sum_{r=-\infty}^{\infty} \delta(\omega - 2\pi r)$$

Figure P6.35-4

Sketch and label $H(e^{j\omega})$ from $(-\pi, \pi)$.

6.36. Consider a general flow graph (denoted Network A) consisting of coefficient multipliers and delay elements, as shown in Figure P6.36-1. If the system is initially at rest, its behavior is completely specified by its impulse response $h[n]$. We wish to modify the system to create a new flow graph (denoted Network A_1) with impulse response $h_1[n] = (-1)^n h[n]$.



Network A

Figure P6.36-1

(a) If $H(e^{j\omega})$ is as given in Figure P6.36-2, sketch $H_1(e^{j\omega})$.

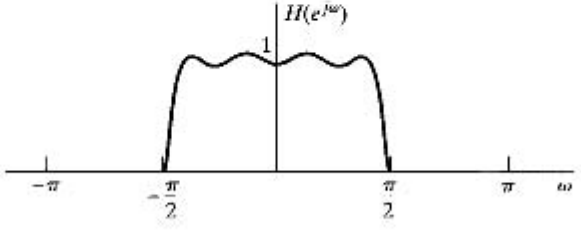


Figure P6.36-2

(b) Explain how to modify Network A by simple modifications of its coefficient multipliers and/or the delay branches to form the new Network A_1 whose impulse response is $h_1[n]$.

(c) If Network A is as given in Figure P6.36-3, show how to modify it by simple modifications to *only the coefficient multipliers* so that the resulting Network A_1 has impulse response $h_1[n]$.

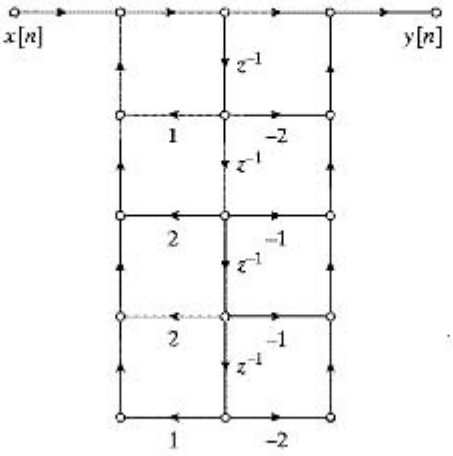


Figure P6.36-3

6.37. The flow graph shown in Figure P6.37 is noncomputable; i.e., it is not possible to compute the output using the difference equations represented by the flow graph because it contains a closed loop having no delay elements.

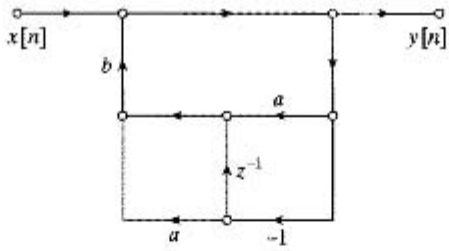


Figure P6.37

(a) Write the difference equations for the system of Figure P6.37, and from them, find the system function of the flow graph.

(b) From the system function, obtain a flow graph that is computable.

6.38. The impulse response of an LTI system is

$$h[n] = \begin{cases} a^n, & 0 \le n \le 7, \\ 0, & \text{otherwise.} \end{cases}$$

(a) Draw the flow graph of a direct form nonrecursive implementation of the system.

(b) Show that the corresponding system function can be expressed as

$$H(z) = \frac{1 - a^8 z^{-8}}{1 - az^{-1}}, \quad |z| > |a|.$$

(c) Draw the flow graph of an implementation of $H(z)$, as expressed in part (b), corresponding to a cascade of an FIR system (numerator) with an IIR system (denominator).

(d) Is the implementation in part (c) recursive or nonrecursive? Is the overall system FIR or IIR?

(e) Which implementation of the system requires
(i) the most storage (delay elements)?
(ii) the most arithmetic (multiplications and additions per output sample)?

6.39. Consider an FIR system whose impulse response is

$$h[n] = \begin{cases} \frac{1}{15}(1 + \cos[(2\pi/15)(n - n_0)]), & 0 \le n \le 14, \\ 0, & \text{otherwise.} \end{cases}$$

This system is an example of a class of filters known as frequency-sampling filters. Problem 6.51 discusses these filters in detail. In this problem, we consider just one specific case.

(a) Sketch the impulse response of the system for the cases $n_0 = 0$ and $n_0 = 15/2$.

(b) Show that the system function of the system can be expressed as

$$H(z) = (1 - z^{-15}) \cdot \frac{1}{15}\left[\frac{1}{1 - z^{-1}} + \frac{\frac{1}{2}e^{-j2\pi n_0/15}}{1 - e^{j2\pi/15}z^{-1}} + \frac{\frac{1}{2}e^{j2\pi n_0/15}}{1 - e^{-j2\pi/15}z^{-1}}\right].$$

(c) Show that if $n_0 = 15/2$, the frequency response of the system can be expressed as

$$H(e^{j\omega}) = \frac{1}{15}e^{-j\omega 7}\left\{\frac{\sin(\omega 15/2)}{\sin(\omega/2)} + \frac{1}{2}\frac{\sin[(\omega - 2\pi/15)15/2]}{\sin[(\omega - 2\pi/15)/2]} + \frac{1}{2}\frac{\sin[(\omega + 2\pi/15)15/2]}{\sin[(\omega + 2\pi/15)/2]}\right\}.$$

Use this expression to sketch the magnitude of the frequency response of the system for $n_0 = 15/2$. Obtain a similar expression for $n_0 = 0$. Sketch the magnitude response for $n_0 = 0$. For which choices of n_0 does the system have a generalized linear phase?

(d) Draw a signal flow graph of an implementation of the system as a cascade of an FIR system whose system function is $1 - z^{-15}$ and a parallel combination of a 1st- and 2nd-order IIR system.

6.40. Consider the discrete-time system depicted in Figure P6.40-1.

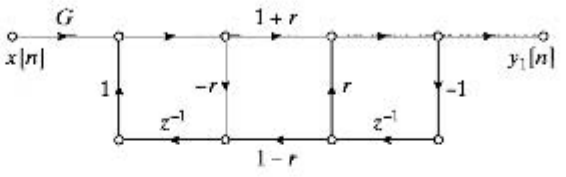


Figure P6.40-1

(a) Write the set of difference equations represented by the flow graph of Figure P6.40-1.

(b) Determine the system function $H_1(z) = Y_1(z)/X(z)$ of the system in Figure P6.40-1, and determine the magnitudes and angles of the poles of $H_1(z)$ as a function of r for $-1 < r < 1$.

(c) Figure P6.40-2 shows a different flow graph obtained from the flow graph of Figure P6.40-1 by moving the delay elements to the opposite top branch. How is the system function $H_2(z) = Y_2(z)/X(z)$ related to $H_1(z)$?

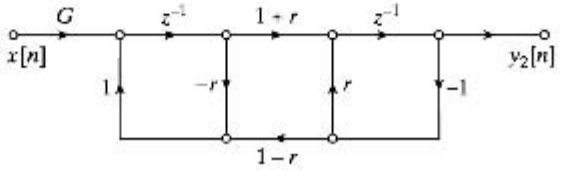


Figure P6.40-2

6.41. The three flow graphs in Figure P6.41 are all equivalent implementations of the same two-input, two-output LTI system.

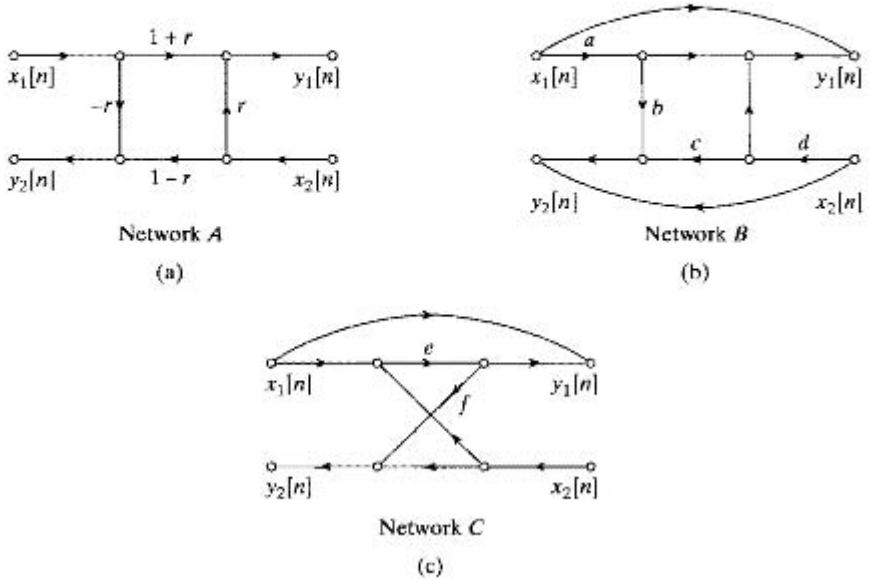


Figure P6.41

(a) Write the difference equations for Network A.

(b) Determine values of a, b, c, and d for Network B in terms of r in Network A such that the two systems are equivalent.

(c) Determine values of e and f for Network C in terms of r in Network A such that the two systems are equivalent.

(d) Why might Network B or C be preferred over Network A? What possible advantage could Network A have over Network B or C?

6.42. Consider an all-pass system with system function

$$H(z) = -0.54\frac{1 - (1/0.54)z^{-1}}{1 - 0.54z^{-1}}.$$

A flow graph for an implementation of this system is shown in Figure P6.42.

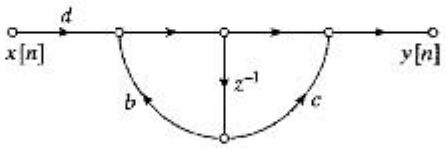


Figure P6.42

(a) Determine the coefficients b, c, and d such that the flow graph in Figure P6.42 is a direct realization of $H(z)$.

(b) In a practical implementation of the network in Figure P6.42, the coefficients b, c, and d might be quantized by rounding the exact value to the nearest tenth (e.g., 0.54 will be rounded to 0.5 and $1/0.54 = 1.8518\ldots$ will be rounded to 1.9). Would the resulting system still be an all-pass system?

(c) Show that the difference equation relating the input and output of the all-pass system with system function $H(z)$ can be expressed as

$$y[n] = 0.54(y[n-1] - x[n]) + x[n-1].$$

Draw the flow graph of a network that implements this difference equation with two delay elements, but only one multiplication by a constant other than +1.

(d) With quantized coefficients, would the flow graph of part (c) be an all-pass system?

The primary disadvantage of the implementation in part (c) compared with the implementation in part (a) is that it requires two delay elements. However, for higher-order systems, it is necessary to implement a cascade of all-pass systems. For N all-pass sections in cascade, it is possible to use all-pass sections in the form determined in part (c) while requiring only $(N + 1)$ delay elements. This is accomplished by sharing a delay element between sections.

(e) Consider the all-pass system with system function

$$H(z) = \left(\frac{z^{-1} - a}{1 - az^{-1}}\right)\left(\frac{z^{-1} - b}{1 - bz^{-1}}\right).$$

Draw the flow graph of a "cascade" realization composed of two sections of the form obtained in part (c) with one delay element shared between the sections. The resulting flow graph should have only three delay elements.

(f) With quantized coefficients a and b, would the flow graph in part (e) be an all-pass system?

6.43. All branches of the signal flow graphs in this problem have unity gain unless specifically indicated otherwise.

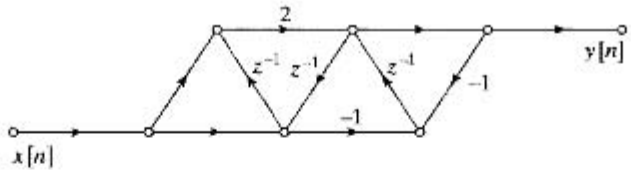


Figure P6.43-1

(a) The signal flow graph of System A, shown in Figure P6.43-1, represents a causal LTI system. Is it possible to implement the same input–output relationship using fewer delays? If it is possible, what is the minimum number of delays required to implement an equivalent system? If it is not possible, explain why not.

(b) Does the System B shown in Figure P6.43-2 represent the same input–output relationship as System A in Figure P6.43-1? Explain clearly.

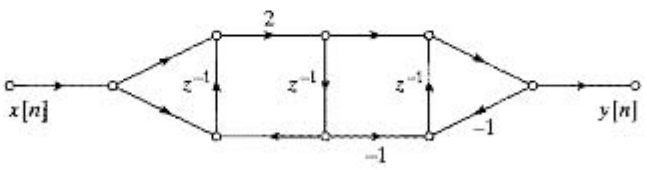


Figure P6.43-2

6.44. Consider an all-pass system whose system function is

$$H(z) = \frac{z^{-1} - \frac{1}{3}}{1 - \frac{1}{3}z^{-1}}.$$

(a) Draw the direct form I signal flow graph for the system. How many delays and multipliers do you need? (Do not count multiplying by ± 1.)

(b) Draw a signal flow graph for the system that uses one multiplier. Minimize the number of delays.

(c) Now consider another all-pass system whose system function is

$$H(z) = \frac{(z^{-1} - \frac{1}{3})(z^{-1} - 2)}{(1 - \frac{1}{3}z^{-1})(1 - 2z^{-1})}.$$

Determine and draw a signal flow graph for the system with two multipliers and three delays.

6.45. With infinite-precision arithmetic, the flow graphs shown in Figure P6.45 have the same system function, but with quantized fixed-point arithmetic they behave differently. Assume that a and b are real numbers and $0 < a < 1$.

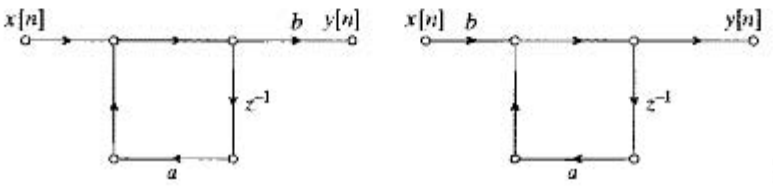


Figure P6.45

(a) Determine x_{max}, the maximum amplitude of the input samples so that the maximum value of the output $y[n]$ of either of the two systems is guaranteed to be less than one.

(b) Assume that the above systems are implemented with two's-complement fixed-point arithmetic, and that in both cases all products are immediately rounded to $B + 1$ bits (*before* any additions are done). Insert round-off noise sources at appropriate locations in the above diagrams to model the rounding error. Assume that each of the noise sources inserted has average power equal to $\sigma_B^2 = 2^{-2B}/12$.

(c) If the products are rounded as described in part (b), the outputs of the two systems will differ; i.e., the output of the first system will be $y_1[n] = y[n] + f_1[n]$ and the output of the second system will be $y_2[n] = y[n] + f_2[n]$, where $f_1[n]$ and $f_2[n]$ are the outputs due to the noise sources. Determine the power density spectra $\Phi_{f_1 f_1}(e^{j\omega})$ and $\Phi_{f_2 f_2}(e^{j\omega})$ of the output noise for both systems.

(d) Determine the total noise powers $\sigma_{f_1}^2$ and $\sigma_{f_2}^2$ at the output for both systems.

6.46. An allpass system is to be implemented with fixed-point arithmetic. Its system function is

$$H(z) = \frac{(z^{-1} - a^*)(z^{-1} - a)}{(1 - az^{-1})(1 - a^*z^{-1})}$$

where $a = re^{j\theta}$.

(a) Draw the signal flow graphs for both the direct form I and direct form II implementations of this system as a 2nd-order system using only real coefficients.

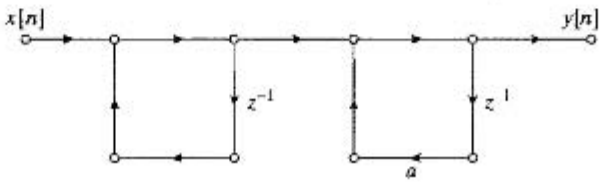
(b) Assuming that the products are each rounded *before* additions are performed, insert appropriate noise sources into the networks drawn in part (a), combining noise sources where possible, and indicating the power of each noise source in terms of σ_B^2, the power of a single rounding noise source.

(c) Circle the nodes in your network diagrams where overflow may occur.

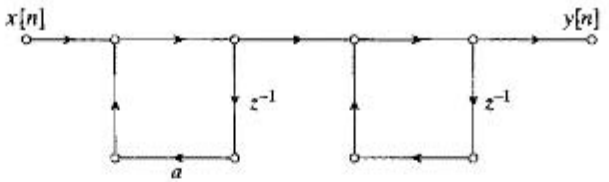
(d) Specify whether or not the output noise power of the direct form II system is independent of r, while the output noise power for the direct form I system increases as $r \to 1$. Give a convincing argument to support your answer. Try to answer the question *without* computing the output noise power of either system. Of course, such a computation would answer the question, but you should be able to see the answer without computing the noise power.

(e) Now determine the output noise power for both systems.

6.47. Assume that a in the flow graphs shown in Figure P6.47 is a real number and $0 < a < 1$. Note that under infinite-precision arithmetic, the two systems are equivalent.



Flow Graph #1



Flow Graph #2

Figure P6.47

(a) Assume that the two systems are implemented with two's-complement fixed-point arithmetic, and that in both cases all products are immediately rounded (*before* any additions are done). Insert round-off noise sources at appropriate locations in both flow graphs to model the rounding error (multiplications by unity do not introduce noise). Assume that each of the noise sources inserted has average power equal to $\sigma_B^2 = 2^{-2B}/12$.

(b) If the products are rounded as described in part (a), the outputs of the two systems will differ; i.e., the output of the first system will be $y_1[n] = y[n] + f_1[n]$ and the output of the second system will be $y_2[n] = y[n] + f_2[n]$, where $y[n]$ is the output owing to $x[n]$ acting alone, and $f_1[n]$ and $f_2[n]$ are the outputs owing to the noise sources. Determine the power density spectrum of the output noise $\Phi_{f_1 f_1}(e^{j\omega})$. Also determine the total noise power of the output of flow graph #1; i.e., determine $\sigma_{f_1}^2$.

(c) Without actually computing the output noise power for flow graph #2, you should be able to determine which system has the largest total noise power at the output. Give a brief explanation of your answer.

6.48. Consider the parallel form flow graph shown in Figure P6.48

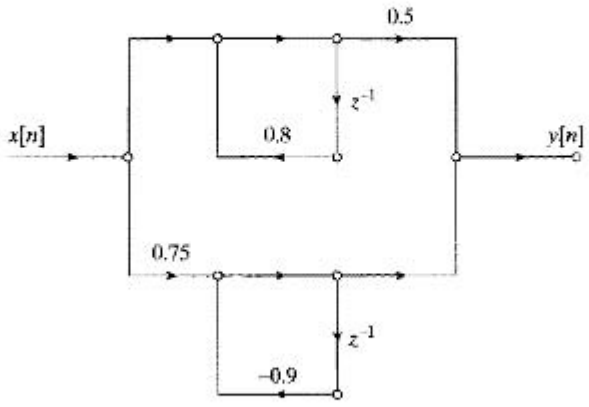


Figure P6.48

(a) Assume that the system is implemented with two's-complement fixed-point arithmetic, and that all products (multiplications by 1 do not introduce noise) are immediately rounded (*before* any additions are done). Insert round-off noise sources at appropriate locations in the flow graph to model the rounding error. Indicate the size (average power) of each noise source in terms of σ_B^2, the average power of one $(B+1)$-bit rounding operation.

(b) If the products are rounded as described in part (a), the output can be represented as $\hat{y}[n] = y[n] + f[n]$ where $y[n]$ is the output owing to $x[n]$ acting alone, and $f[n]$ is the total output due to all the noise sources acting independently. Determine the power density spectrum of the output noise $\Phi_{ff}(e^{j\omega})$.

(c) Also determine the total noise power σ_f^2 of the noise component of the output.

6.49. Consider the system shown in Figure P6.49, which consists of a 16-bit A/D converter whose output is the input to an FIR digital filter that is implemented with 16-bit fixed-point arithmetic.

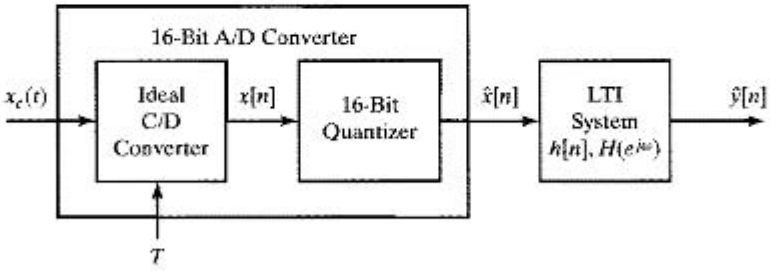


Figure P6.49

The impulse response of the digital filter is

$$h[n] = -.375\delta[n] + .75\delta[n-1] - .375\delta[n-2].$$

This system is implemented with 16-bit two's-complement arithmetic. The products are rounded to 16-bits *before* they are accumulated to produce the output. In anticipation of using the linear noise model to analyze this system, we define $\hat{x}[n] = x[n] + e[n]$ and $\hat{y}[n] = y[n] + f[n]$, where $e[n]$ is the quantization error introduced by the A/D converter and $f[n]$ is the *total* quantization noise at the output of the filter.

(a) Determine the maximum magnitude of $\hat{x}[n]$ such that no overflow can possibly occur in implementing the digital filter; i.e., determine $x_{\max}$ such that $\hat{y}[n] < 1$ for all $-\infty < n < \infty$ when $\hat{x}[n] < x_{\max}$ for all $-\infty < n < \infty$.

(b) Draw the linear noise model for the complete system (including the linear noise model of the A/D). Include a detailed flow-graph for the digital filter including all noise sources due to quantization.

(c) Determine the total noise power at the output. Denote this σ_f^2.

(d) Determine the power spectrum of the noise at the output of the filter; i.e., determine $\Phi_{ff}(e^{j\omega})$. Plot your result.

Extension Problems

6.50. In this problem, we consider the implementation of a causal filter with system function

$$H(z) = \frac{1}{(1 - .63z^{-1})(1 - .83z^{-1})} = \frac{1}{1 - 1.46z^{-1} + 0.5229z^{-2}}$$

This system is to be implemented with $(B+1)$-bit two's-complement rounding arithmetic with products rounded before additions are performed. The input to the system is a zero-mean, white, wide-sense stationary random process, with values uniformly distributed between $-x_{\max}$ and $+x_{\max}$.

(a) Draw the direct form flow graph implementation for the filter, with all coefficient multipliers rounded to the nearest tenth.

(b) Draw a flow graph implementation of this system as a cascade of two 1^{st}-order systems, with all coefficient multipliers rounded to the nearest tenth.

(c) Only one of the implementations from parts (a) and (b) above is usable. Which one? Explain.

(d) To prevent overflow at the output node, we must carefully choose the parameter $x_{\max}$. For the implementation selected in part (c), determine a value for $x_{\max}$ that guarantees the output will stay between -1 and 1. (Ignore any potential overflow at nodes other than the output.)

(e) Redraw the flow graph selected in part (c), this time including linearized noise models representing quantization round-off error.

(f) Whether you chose the direct form or cascade implementation for part (c), there is still at least one more design alternative:

(i) If you chose the direct form, you could also use a transposed direct form.

(ii) If you chose the cascade form, you could implement the smaller pole first or the larger pole first.

For the system chosen in part (c), which alternative (if any) has lower output quantization noise power? Note you do not need to explicitly calculate the total output quantization noise power, but you must justify your answer with some analysis.

6.51. In this problem, we will develop some of the properties of a class of discrete-time systems called frequency-sampling filters. This class of filters has system functions of the form

$$H(z) = (1 - z^{-N}) \cdot \sum_{k=0}^{N-1} \frac{\tilde{H}[k]/N}{1 - z_k z^{-1}},$$

where $z_k = e^{j(2\pi/N)k}$ for $k = 0, 1, \ldots, N-1$.

(a) System functions such as $H(z)$ can be implemented as a cascade of an FIR system whose system function is $(1 - z^{-N})$ with a parallel combination of 1st-order IIR systems. Draw the signal flow graph of such an implementation.

(b) Show that $H(z)$ is an $(N-1)$st-degree polynomial in z^{-1}. To do this, it is necessary to show that $H(z)$ has no poles other than $z = 0$ and that it has no powers of z^{-1} higher than $(N-1)$. What do these conditions imply about the length of the impulse response of the system?

(c) Show that the impulse response is given by the expression

$$h[n] = \left(\frac{1}{N} \sum_{k=0}^{N-1} \tilde{H}[k] e^{j(2\pi/N)kn} \right) (u[n] - u[n-N]).$$

Hint: Determine the impulse responses of the FIR and the IIR parts of the system, and convolve them to find the overall impulse response.

(d) Use l'Hôpital's rule to show that

$$H(z_m) = H(e^{j(2\pi/N)m}) = \tilde{H}[m], \qquad m = 0, 1, \ldots, N-1;$$

i.e., show that the constants $\tilde{H}[m]$ are samples of the frequency response of the system, $H(e^{j\omega})$, at equally spaced frequencies $\omega_m = (2\pi/N)m$ for $m = 0, 1, \ldots, N-1$. It is this property that accounts for the name of this class of FIR systems.

(e) In general, both the poles z_k of the IIR part and the samples of the frequency response $\tilde{H}[k]$ will be complex. However, if $h[n]$ is real, we can find an implementation involving only real quantities. Specifically, show that if $h[n]$ is real and N is an even integer, then $H(z)$ can be expressed as

$$H(z) = (1 - z^{-N}) \left\{ \frac{H(1)/N}{1 - z^{-1}} + \frac{H(-1)/N}{1 + z^{-1}} + \sum_{k=1}^{(N/2)-1} \frac{2|H(e^{j(2\pi/N)k})|}{N} \cdot \frac{\cos[\theta(2\pi k/N)] - z^{-1}\cos[\theta(2\pi k/N) - 2\pi k/N]}{1 - 2\cos(2\pi k/N)z^{-1} + z^{-2}} \right\},$$

where $H(e^{j\omega}) = |H(e^{j\omega})|e^{j\theta(\omega)}$. Draw the signal flow graph representation of such a system when $N = 16$ and $H(e^{j\omega_k}) = 0$ for $k = 3, 4, \ldots, 14$.

6.52. In Chapter 4, we showed that, in general, the sampling rate of a discrete-time signal can be reduced by a combination of linear filtering and time compression. Figure P6.52 shows a block diagram of an M-to-1 decimator that can be used to reduce the sampling rate by an integer factor M. According to the model, the linear filter operates at the high sampling rate. However, if M is large, most of the output samples of the filter will be discarded by the compressor. In some cases, more efficient implementations are possible.

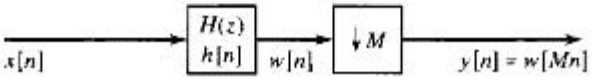


Figure P6.52

(a) Assume that the filter is an FIR system with impulse response such that $h[n] = 0$ for $n < 0$ and for $n > 10$. Draw the system in Figure P6.52, but replace the filter $h[n]$ with an equivalent signal flow graph based on the given information. Note that it is not possible to implement the M-to-1 compressor using a signal flow graph, so you must leave this as a box, as shown in Figure P6.52.

(b) Note that some of the branch operations can be commuted with the compression operation. Using this fact, draw the flow graph of a more efficient realization of the system of part (a). By what factor has the total number of computations required in obtaining the output $y[n]$ been decreased?

(c) Now suppose that the filter in Figure P6.52 has system function

$$H(z) = \frac{1}{1 - \frac{1}{2}z^{-1}}, \qquad |z| > \frac{1}{2}.$$

Draw the flow graph of the direct form realization of the complete system in the figure. With this system for the linear filter, can the total computation per output sample be reduced? If so, by what factor?

(d) Finally, suppose that the filter in Figure P6.52 has system function

$$H(z) = \frac{1 + \frac{7}{8}z^{-1}}{1 - \frac{1}{2}z^{-1}}, \qquad |z| > \frac{1}{2}.$$

Draw the flow graph for the complete system of the figure, using each of the following forms for the linear filter:

(i) direct form I
(ii) direct form II
(iii) transposed direct form I
(iv) transposed direct form II.

For which of the four forms can the system of Figure P6.52 be more efficiently implemented by commuting operations with the compressor?

6.53. Speech production can be modeled by a linear system representing the vocal cavity, which is excited by puffs of air released through the vibrating vocal cords. One approach to synthesizing speech involves representing the vocal cavity as a connection of cylindrical acoustic tubes of equal length, but with varying cross-sectional areas, as depicted in Figure P6.53. Let us assume that we want to simulate this system in terms of the volume velocity representing airflow. The input is coupled into the vocal tract through a small constriction, the vocal cords. We will assume that the input is represented by a change in volume velocity at the left end, but that the boundary condition for traveling waves at the left end is that the net volume velocity must be zero. This is analogous to an electrical transmission line driven by a current source at one end and with an open circuit at the far end. Current in the transmission line is then analogous to volume velocity in the acoustic tube, whereas voltage is analogous to acoustic pressure. The output of the acoustic tube is the volume velocity at the right end. We assume that each section is a lossless acoustic transmission line.

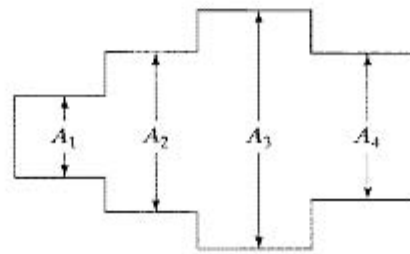


Figure P6.53

At each interface between sections, a forward-traveling wave f^+ is transmitted to the next section with one coefficient and reflected as a backward-traveling wave f^- with a different coefficient. Similarly, a backward-traveling wave f^- arriving at an interface is transmitted with one coefficient and reflected with a different coefficient. Specifically, if we consider a forward-traveling wave f^+ in a tube with cross-sectional area A_1 arriving at the interface with a tube of cross-sectional area A_2, then the forward-traveling wave transmitted is $(1+r)f^+$ and the reflected wave is rf^+, where

$$r = \frac{A_2 - A_1}{A_2 + A_1}.$$

Consider the length of each section to be 3.4 cm, with the velocity of sound in air equal to 34,000 cm/s. Draw a flow graph that will implement the four-section model in Figure P6.53, with the output sampled at 20,000 samples/s.

In spite of the lengthy introduction, this a reasonably straightforward problem. If you find it difficult to think in terms of acoustic tubes, think in terms of transmission-line sections with different characteristic impedances. Just as with transmission lines, it is difficult to express the impulse response in closed form. Therefore, draw the flow graph directly from physical considerations, in terms of forward- and backward-traveling pulses in each section.

6.54. In modeling the effects of round-off and truncation in digital filter implementations, quantized variables are represented as

$$\hat{x}[n] = Q[x[n]] = x[n] + e[n],$$

where $Q[\cdot]$ denotes either rounding or truncation to $(B+1)$ bits and $e[n]$ is the *quantization error*. We assume that the quantization noise sequence is a stationary white-noise sequence such that

$$\mathcal{E}\{(e[n] - m_e)(e[n+m] - m_e)\} = \sigma_e^2 \delta[m]$$

and that the amplitudes of the noise sequence values are uniformly distributed over the quantization step $\Delta = 2^{-B}$. The 1st-order probability densities for rounding and truncation are shown in Figures P6.54(a) and (b), respectively.

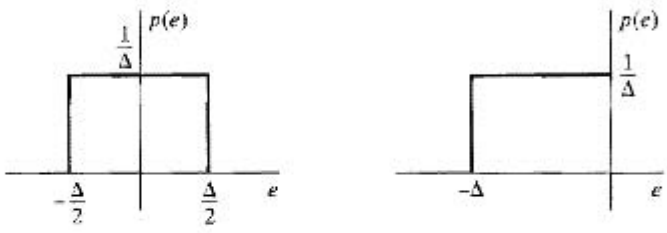


Figure P6.54

(a) Determine the mean m_e and the variance σ_e^2 for the noise owing to rounding.

(b) Determine the mean m_e and the variance σ_e^2 for the noise owing to truncation.

6.55. Consider an LTI system with two inputs, as depicted in Figure P6.55. Let $h_1[n]$ and $h_2[n]$ be the impulse responses from nodes 1 and 2, respectively, to the output, node 3. Show that if $x_1[n]$ and $x_2[n]$ are uncorrelated, then their corresponding outputs $y_1[n]$ and $y_2[n]$ are also uncorrelated.

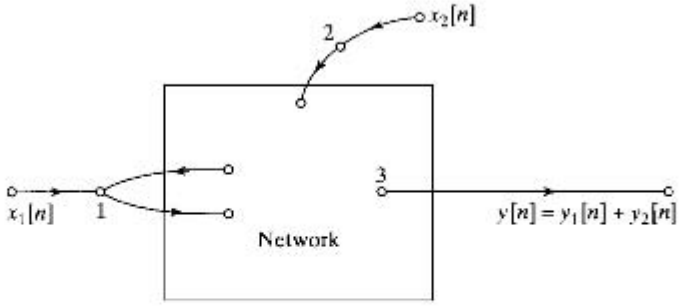


Figure P6.55

6.56. The flow graphs in Figure P6.56 all have the same system function. Assume that the systems in the figure are implemented using $(B + 1)$-bit fixed-point arithmetic in all the computations. Assume also that all products are rounded to $(B + 1)$ bits *before* additions are performed.

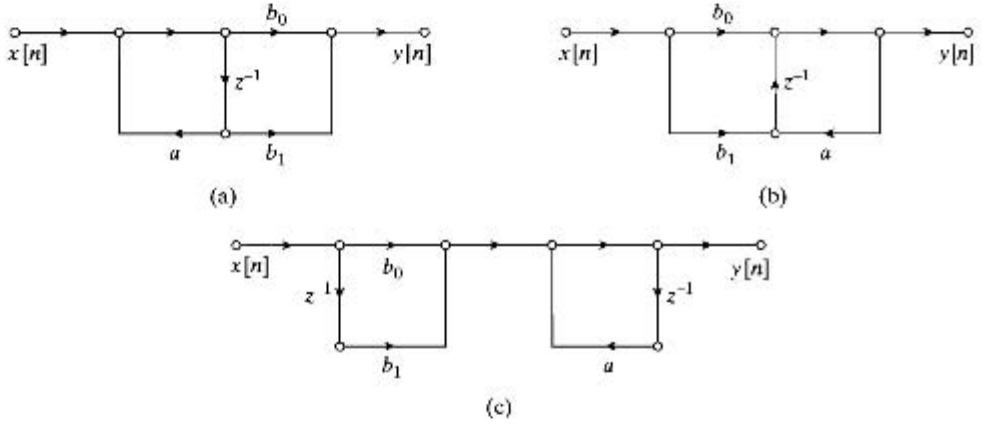


Figure P6.56

(a) Draw linear-noise models for each of the systems in Figure P6.56.

(b) Two of the flow graphs in Figure P6.56 have the *same* total output noise power owing to arithmetic round-off. Without explicitly computing the output noise power, determine which two have the same output noise power.

(c) Determine the output noise power for each of the flow graphs in Figure P6.56. Express your answer in terms of σ_B^2, the power of a single source of round-off noise.

6.57. The flow graph of a 1st-order system is shown in Figure P6.57.

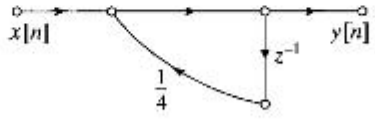


Figure P6.57

(a) Assuming infinite-precision arithmetic, find the response of the system to the input

$$x[n] = \begin{cases} \frac{1}{2}, & n \geq 0, \\ 0, & n < 0. \end{cases}$$

What is the response of the system for large n?

Now suppose that the system is implemented with fixed-point arithmetic. The coefficient and all variables in the flow graph are represented in sign-and-magnitude notation with 5-bit registers. That is, all numbers are to be considered signed fractions represented as

$$b_0 b_1 b_2 b_3 b_4,$$

where b_0, b_1, b_2, b_3, and b_4 are either 0 or 1 and

$$|\text{Register value}| = b_1 2^{-1} + b_2 2^{-2} + b_3 2^{-3} + b_4 2^{-4}.$$

If $b_0 = 0$, the fraction is positive, and if $b_0 = 1$, the fraction is negative. The result of a multiplication of a sequence value by a coefficient is truncated before additions occur; i.e., only the sign bit and the most significant four bits are retained.

(b) Compute the response of the quantized system to the input of part (a), and plot the responses of both the quantized and unquantized systems for $0 \le n \le 5$. How do the responses compare for large n?

(c) Now consider the system depicted in Figure P6.57, where

$$x[n] = \begin{cases} \frac{1}{2}(-1)^n, & n \ge 0, \\ 0, & n < 0. \end{cases}$$

Repeat parts (a) and (b) for this system and input.

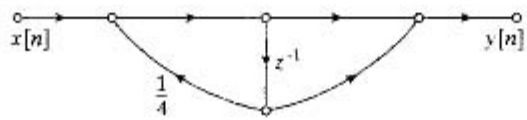


Figure P6.57

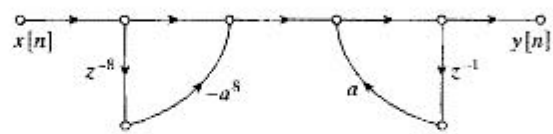
6.58. A causal LTI system has a system function

$$H(z) = \frac{1}{1 - 1.04z^{-1} + 0.98z^{-2}}.$$

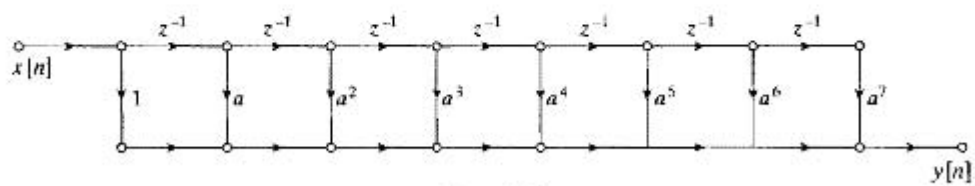
(a) Is this system stable?

(b) If the coefficients are rounded to the nearest tenth, would the resulting system be stable?

6.59. When implemented with infinite-precision arithmetic, the flow graphs in Figure P6.59 have the same system function.



Network 1



Network 2

Figure P6.59

(a) Show that the two systems have the same overall system function from input $x[n]$ to output $y[n]$.

(b) Assume that the preceding systems are implemented with two's complement fixed-point arithmetic and that products are rounded *before* additions are performed. Draw signal flow graphs that insert round-off noise sources at appropriate locations in the signal flow graphs of Figure P6.59.

(c) Circle the nodes in your figure from part (b) where overflow can occur.

(d) Determine the maximum size of the input samples such that overflow cannot occur in either of the two systems.

(e) Assume that $|a| < 1$. Determine the total noise power at the output of each system, and determine the maximum value of $|a|$ such that Network 1 has lower output noise power than Network 2.