NRL Report 7870

Overview of Digital Signal Processing Theory

LAWRENCE M. LEIBOWITZ

Digital Applications Branch
Office of the Director of Research

May 20, 1975

NAVAL RESEARCH LABORATORY
Washington, D.C.

SECURITY CLASSIFICATION OF THIS PAGE *(When Date Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: NRL Report 7870	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: OVERVIEW OF DIGITAL SIGNAL PROCESSING THEORY	5. TYPE OF REPORT & PERIOD COVERED: Interim report
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Lawrence M. Leibowitz Code 4150	8. CONTRACT OR GRANT NUMBER(s): See back
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Research Laboratory, Washington, D.C. 20375	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE: May 20, 1975
	13. NUMBER OF PAGES: 91
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Digital signal processing	Transpose
Discrete signals	Digital filters
Linear shift-invariant systems	Infinite impulse response (IIR)
z Transform	Finite impulse response (FIR)
Digital networks	Discrete Fourier transform (DFT) (Continued)

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

An overview of the theory of digital signal processing is presented here using key portions of the applicable technical literature. This can serve either as an introduction for those who desire to pursue the subject further or as a concise summary for those for whom more detailed investigation would be impractical. The discussion begins with a consideration of discrete systems and signals as well as their relationship to the continuous case. The realization of digital signal processing systems in the form of digital networks is presented. Theory and design of digital filters are discussed along with their relation to continuous filter characteristics. The discrete Fourier transform (DFT) and

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE *(When Date Entered)*

8. This report represents a part of the research performed under the NRL Edison Memorial Fellowship in partial fulfillment of the requirements for the degree of Doctor of Science at the George Washington University School of Engineering and Applied Science.

19.

Fast Fourier transform (FFT)	Input quantization
Discrete convolution	Coefficient quantization
High-speed convolution	Roundoff error
Sectioning	Dynamic range limitations
Quantization effects	Limit cycles

20. its efficient implementation in the form of the fast Fourier transform (FFT) are presented. Discrete convolution and correlation and the application of the FFT to their efficient implementation are described. Finally the quantization effects inherent in all digital signal-processing realizations are reviewed with respect to their influence on digital-filter and FFT outputs.

CONTENTS

OVERVIEW OF DIGITAL SIGNAL PROCESSING THEORY

1. INTRODUCTION

1.1 Signal Processing

Electricity, or the flow of electrons, has enabled man to satisfy some of his most important needs. Among these are the storage, processing, and transmission of a practical and useful form of energy and the storage, processing, and transmission of information. This latter application of electron flow comes under the more specific description *electronics*, and includes voice communications, data communications, and various types of control systems, timing systems, and detection systems used in radar, sonar, and seismological technology.

This handling and manipulation of information is accomplished by a direct correspondence between the natural variation in the information and some characteristic of the flow of electrical energy. The characteristics of the electrical energy available for this purpose include amplitude, frequency, and relative time delay or phase. The overall electrical waveform, or signal used for information handling, normally consists of a component due directly to information content as well as a component due to undesired effects described generally as *noise*. In the manipulation of information in the form of electrical waveforms, it is often necessary to change one waveform into another more desirable waveform. It may be desired to modify a waveform component or characteristic, separate two or more previously combined waveforms, or even eliminate a waveform component entirely. Such modifications of waveforms or signals, come under the general classification of *signal-processing* techniques which are implemented by means of signal-processing systems. The most significant areas of signal processing include filters, which are used for waveform shaping as well as spectral and correlation measurement.

The signals to be considered by a signal-processing system are classified as either one dimensional or multidimensional depending on the number of independent variables. Electrical signals are generally one-dimensional functions of time; a picture, for example, with its spatial variables, represents a two-dimensional signal. In this report, only one-dimensional signal processing will be discussed except where otherwise noted. For ease of discussion, it will be assumed here that the independent variable is time, although other interpretations such as distance would serve equally well.

Note: This report represents a part of the research performed under the Edison Memorial Fellowship in partial fulfillment of the requirements for the degree of Doctor of Science at the George Washington University School of Engineering and Applied Science.
Manuscript submitted February 7, 1975.

1.2 Continuous Systems

Signals are usually generated at their source and used at their final destination in a form in which the dependent variable, signal amplitude, can take on a continuous range of values as a continuous function of time. The class of such signals are referred to as *analog* or *continuous* signals, with the latter being a more desirable term. Examples of such signals are those generated and received in normal AM and FM radio systems. Mathematically, sin ωt would be such a signal. The class of systems in which these continuous or analog signals are used are known as continuous or analog systems. The general analysis of such systems is covered in Refs. 1, 2, and 3, and the synthesis of these systems is covered in Ref. 4. As opposed to strictly continuous signals there are signals in which only the independent variable, time, assumes a continuum of values; these are referred to as *continuous-time* signals.

1.3 Discrete Systems

Discrete-time signals are defined over a continuous range of amplitude values but only for discrete values of the independent variable, time. These discrete-time or *sampled-data* signals are used in sampled-data systems as described in Ref. 5. When the signal amplitude, or dependent variable, is restricted to a discrete set of values defined only at a discrete set of values of the independent variable, the signal is referred to as a *digital signal*. Thus systems that handle signals which are represented as a sequence of discrete values are *digital systems*. Such digital systems designed for accomplishing waveform manipulation by spectrum modifications are defined as *digital filters*. A digital signal could be produced for presentation at an input to a digital system by means of an analog-to-digital converter which produces discrete samples of a continuous time signal. In a digital signal the noise component, mentioned earlier, can be represented as a sequence of undesirable discrete values. This noise sequence would generally be a sequence of random values. It is the manipulation of digital signals by digital systems which is classified under the description of *digital signal processing* which will be reviewed in this report. The definitions and terminology used in this report are generally consistent with that recommended by the IEEE Group on Audio and Electroacoustics [6]. Many definitions are presented here for the benefit of those not previously versed in the relatively new field of digital signal processing. For others, this will serve for review and consistency of terminology.

The components which are interconnected to form continuous system networks are resistors, inductors, and capacitors. The parameter values of these resistance-inductance-capacitance (RLC) devices determine the signal-processing characteristics of a continuous system. Digital systems are composed of digital adders, multipliers, and unit delays or delay registers. In binary digital circuitry, these components are formed by networks of nonlinear logic gates and flipflop bit-storage devices. The interconnection or block diagrams of these components determine the characteristics of the digital system.

1.4 Digital Signal Processing

A digital system used for signal processing can in general be implemented in either of two ways. The first is a machine-, assembly-, or higher-level-language computer

program which is used in a general-purpose computer to implement the algorithmic procedure indicated by the block diagram. This is a true mathematical representation of the digital system and should not be referred to as a simulation. In the other approach a special-purpose computing system, interconnecting the digital devices as indicated in the block diagram, can be physically implemented. The increasing speed and decreasing size and cost of digital integrated-circuit hardware elements along with their extremely high reliability, maintainability, and repeatability of performance have resulted within the past dozen years in an increasing desire to perform more and more signal-processing tasks by digital rather than analog means. That an isomorphism between digital and analog signal processing exists in a large class of practical problems, and thus that the trend toward digital signal processing is justified, has been shown by Steiglitz [7]. Digital systems suffer less than continuous systems from parameter-value repeatability inaccuracies and performance sensitivity to environment. A major problem area however that must continually be considered in the design and application of digital systems is the inherent quantization effects due to the necessarily finite representation of all parameters in a system.

Since most signals to be processed occur naturally in a continuous form, the input to a digital signal processing system is usually preceded by an analog-to-digital converter that, under the control of a trigger signal, generates digital samples of the continuous signal. These digital samples are processed within the digital signal processing system according to the required algorithms and presented at its output. With the exception of cases in which the results can be accepted in digital form, such as for presentation to other digital systems or output devices, the output must be presented to a digital-to-analog converter that provides the final output in the form of a continuous signal. These converters, which operate between continuous and discrete signal representations, generate further inaccuracies due to finite operating speeds as well as finite quantization. The theory and implementation of analog/digital conversions has been covered extensively by Schmid in Ref. 8.

The basic linear algorithms used in digital signal processing are the digital filter and the discrete Fourier transform (DFT). A digital filter is usually accomplished by recursion described by linear difference equations, although other realizations use discrete convolution and DFT techniques. The DFT is almost always applied in the form of one of an extremely efficient set of algorithms collectively referred to as fast-Fourier-transform or FFT techniques. Such techniques, which were first disclosed by Cooley and Tukey [9] in 1965 reduce the computation time by a large factor so as to make previously inefficient, long DFT procedures practical. This development has had the most significant effect on digital signal processing. The DFT in the form of the FFT is used for frequency spectral processing and measurement, correlation measurement, system realization by high-speed convolution, and the realization of digital filters.

1.5 Objectives

This report will review the theory and techniques by which signal-processing procedures, both those previously accomplished by continuous systems as well as those previously impractical, can be accomplished by digital means. The major areas of digital signal processing will be explored.

This discussion will begin with the theory of discrete systems, with emphasis on discrete signals, the z transform, discrete-time linear shift-invariant systems, discrete convolution, system functions, causality, and stability. The relationships between discrete and continuous system theory will be presented with respect to the Laplace transform in the continuous case and the z transform in the discrete case, with consideration of mappings between the s and z planes. The Fourier transforms in the continuous and discrete cases will be considered as well as the sampling of continuous signals for digital processing and the reconstruction of continuous signals from discrete representations. The methods of realization of digital signal processing systems will be presented along with a description of digital network elements, signal flow graphs, and various forms of digital networks as derived from the system function in the form of a ratio of polynomials in z^{-1}. The theory of digital filtering will be discussed with relation to continuous filter theory in terms of such characteristics as the bandwidth, ripple, and the magnitude-squared function. The design of digital filters, to satisfy frequency-domain characteristics, by various techniques such as using transformations to translate proven continuous filter designs into digital filters will be considered. Digital filter design techniques in the time-domain are also discussed. The theory of the DFT and its implementation problems will be presented prior to a description of the theory and realization of the FFT. Particularly powerful applications of the FFT such as high-speed convolution and correlation will be discussed along with a description of the techniques required to correctly use the FFT algorithm. Finally the quantization effects inherent in all phases of digital signal processing will be reviewed with respect to their influence on the outputs from digital filters and FFT applications.

2. DISCRETE SIGNALS AND SYSTEMS

2.1 Representation of Discrete Signals

To be able to represent and analyze digital systems and gain further insight into their operation, a host of definitions and techniques have been developed. These definitions and techniques have counterparts in continuous system theory. A signal can be represented in continuous system theory as a function of time $x(t)$, where the domain of t can be $-\infty < t < \infty$ or any subset of that interval. In a discrete time system, a signal is represented as a sequence of values $x(nT)$ where n is an integer and, in general, $-\infty < n < +\infty$. Thus the function is defined only at discrete time intervals of length T. The discrete signal can be thought of as the result of sampling $x(t)$ at uniform intervals of duration T. This sampling process will be discussed later. For the purpose of representing a discrete signal as a sequence of values, it can be assumed that T is equal to unity without effecting the validity of the theory to follow. Thus a discrete signal can be represented as the sequence $x(n)$, where $-\infty < n < \infty$, and for the purpose of discussion the nth value of the sequence can be thought of as the nth sample.

The unit impulse function, and the response of continuous systems to such an input, play an important role in the representation and analysis of signals and systems in continuous system theory. An analogous situation exists in discrete system theory with a *discrete-time impulse* or *unit sample* $\delta(n)$ input and the corresponding output or *unit-sample response*. The unit sample has the value 0 for all values of n except $n = 0$, for which $\delta(0) = 1$. The sequence $\delta(n - n_0)$ is 0 for all n except $n = n_0$, for which it has a value 1. The product of a constant $x(n_0)$ and the unit-sample function delayed by n_0,

$x(n_0)\delta(n - n_0)$, represents a sample of magnitude $x(n_0)$ at the n_0th sample of a sequence. Thus the unit sample can be used to represent a sequence $x(n)$ as a weighted sum of unit samples, that is,

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n - k).$$

A comprehensive and general description of the discrete representation of signals has been presented by Oppenheim and Johnson in Ref. 10. In that paper several alternatives to periodically sampled representations are discussed along with the representation of discrete sequences by other discrete sequences.

2.2 Linear Shift-Invariant Systems

An important class of discrete-time systems which are used to perform many signal processing functions are *linear shift-invariant systems.* This class can be easily handled mathematically and can be readily designed to particular specifications. The conditions pertaining to these systems are described in Ref. 11. The transformation $T[\ldots]$ can be used to represent the output $y(n)$ of a system in response to an input $x(n)$, where $y(n) = T[x(n)]$. If a system has responses $y_1(n)$ and $y_2(n)$ corresponding to inputs $x_1(n)$ and $x_2(n)$, then the system is linear only if

$$T[ax_1(n) + bx_2(n)] = aT[x_1(n)] + bT[x_2(n)] = ay_1(n) + by_2(n)$$

with a and b arbitrary constants. For the class of shift-invariant systems, if the response $y(n)$ corresponds to input $x(n)$, then $y(n - k)$ is the response corresponding to input $x(n - k)$, k being any integer. The class of systems possessing both the linearity and shift-invariant restraints are linear shift-invariant systems. These systems are analogous to linear time-invariant systems, which are the most used class of continuous systems, as described in Refs. 1 through 4. Unless otherwise stated, all systems to be considered here possess the linear shift-invariant properties.

A subclass of linear shift-invariant systems used in many signal-processing systems and particularly in digital filtering are those described by *linear constant-coefficient difference equations.* These difference equations can be used, as in Refs. 11 and 12, to describe the behavior of linear shift-invariant discrete systems in the same manner that linear constant-coefficient differential equations are used for the analysis of linear time-invariant continuous systems. These difference equations are of the form

$$\sum_{k=0}^{N} a_k\, y(n - k) = \sum_{r=0}^{M} b_r\, x(n - r),$$

where $x(n)$ and $y(n)$ are the system input and output sequences respectively. The nth value of the output can therefore be expressed as

$$y(n) = -\sum_{k=1}^{N}\left(\frac{a_k}{a_0}\right) y(n-k) + \sum_{r=0}^{M}\left(\frac{b_r}{a_0}\right) x(n-r)$$

and is thus a function of the nth input value and the M and N past values of the input and output. If the unit-sample response is of finite duration the system is an *FIR* (finite impulse response) system, whereas a system with a unit sample response of infinite duration is an *IIR* (infinite impulse response) system. For an FIR system $N = 0$ and $y(n)$ is a function only of the present and past M inputs. For an IIR system N must be greater than zero.

2.3 The z Transform

The Laplace transform [13] permits the differential equations which describe the operations of continuous systems to be transformed into algebraic equations which can be more easily manipulated and solved. As a direct extension of this transform technique, components of a continuous signal processing system can be represented for analysis directly in their s-plane or frequency-domain equivalents, thus permitting ease of analysis. The z transform techniques, initially introduced by DeMoivre [14] via the concept of the "generating function" of probability theory, likewise permit algebraic manipulation and frequency-domain representation for discrete systems and the linear difference equations which describe their operation. The application of z transforms to sampled-data systems is described by Ragazzini and Franklin [5], and a complete development of the z transform and its properties is provided by Jury [15]. A discussion of the z transform requires application of some results from complex variable theory such as provided in Ref. 16. The z transform $X(z)$ of a discrete sequence $x(n)$ is defined as

$$X(z) = \sum_{n=-\infty}^{+\infty} x(n)z^{-n},$$

which due to the extent of the summation index to all negative as well as positive integers is known as the *two-sided z transform*. The complex variable z^{-1} is termed the *unit delay* operator and z is the *unit advance* operator. If, as in practical systems, the sequence $x(n)$ starts at $n = 0$ with $x(n) = 0$ for all $n < 0$, the z transform can then be expressed in its one-sided form:

$$X(z) = \sum_{n=0}^{+\infty} x(n)z^{-n}.$$

In like fashion, one can describe a z transform for a finite-length sequence $x(n)$ which is nonzero from n_1 to n_2 and can describe z transforms for right-sided and left-sided sequences defined for summation indices $n_1 \leqslant n < +\infty$ and $-\infty < n \leqslant n_2$ respectively. Just as the Laplace transform can be represented by its behavior in the complex s plane, $X(z)$ can be represented graphically in the complex z plane. The z transform operation can be represented symbolically as $X(z) = z[x(n)]$.

To completely specify the z transform, it is necessary to express the defined series along with a description of the region of convergence of $X(z)$ in the z plane. The region of convergence of a z transform $X(z)$ is that set of values of the complex variable z for which $X(z)$ converges. In general this region will be the annular region $R_- < |z| < R_+$, where R_- can be as small as 0 and R_+ as large as ∞.

The z transforms that occur in the analysis of linear shift-invariant systems can be expressed as ratios of polynomials in z or z^{-1}, as will be shown later. Those values of z for which the numerator, and thus the z transform $X(z)$, are 0 are the *zeros* of $X(z)$. Those values of z for which the denominator is 0, and thus $X(z)$ is infinite, are known as the *poles* of $X(z)$. Additionally poles may occur at $z = \infty$. There are several relationships between the poles and zeros of a z transform and its region of convergence which can be derived from arguments presented in Ref. 11. First the region of convergence cannot contain any poles, and second the region of convergence must be bounded by poles or by 0 and ∞.

2.4 The Inverse z Transform

From the z transform $X(z)$ the corresponding sequence $x(n)$ can be found by means of one of several methods defined in Ref. 15. This process is referred to as the *inverse z transform* and can be denoted symbolically as $x(n) = z^{-1}[X(z)]$. In its most general form the inverse transform can be expressed as a complex integral formula,

$$x(n) = \frac{1}{2\pi j} \oint_C X(z) z^{n-1} dz,$$

where C is a counterclockwise closed contour in the region of convergence of $X(z)$ and enclosing the origin as well as all singularities of $X(z)$. For rational z transforms, Cauchy's integral formula can be used to evaluate $x(n)$ as

$$x(n) = \frac{1}{2\pi j} \oint_C X(z) z^{n-1}\, dz = \text{sum of the residues of } X(z) z^{n-1}.$$

When $X(z)$ can be expressed in a power series expansion (Taylor's series) of $X(z)$ as a function of z^{-1}, the value of $x(n)$ will be the coefficient of the z^{-n} term in the power series

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n}.$$

In practical problems an $X(z)$ given in closed form can be expressed in a power series by the use of an existing expansion such as those for the sine or logarithm. For rational z transforms a power-series expansion can be derived by long division with consideration for convergence at $z = \infty$.

The partial-fraction expansion method of z transform inversion can be applied to a rational $X(z)$ analytic at infinity. The partial-fraction expansion of $X(z)$ can be expressed as

$$X(z) = X_1(z) + X_2(z) + \cdots .$$

The inverse of $X(z)$ can then be obtained as the sum of the inverses of each partial fraction in the above expansion, that is,

$$x(n) = z^{-1}[X(z)] = z^{-1}[X_1(z)] + z^{-1}[X_2(z)] + \cdots .$$

The inverse of each of the simpler forms $X_k(z)$ can then be found from tables or power series and summed to give $x(n)$.

2.5 Application of the z Transform

As described previously, the utility of the z transform is in the representation of discrete systems and the solution of the linear difference equations which describe the operation of a significant class of such systems. The solution of linear difference equations by z transforms is covered in detail in chapter 2 of Ref. 15. For the linear difference equation, whose general form was described previously, consider the case with $N = 1$, $M = 0$, $a_0 = 1$, $a_1 = -K$, and $b_0 = 1$. Thus $y(n) = Ky(n - 1) + x(n)$ with initial condition $y(-1) = 0$, which is a first-order linear difference equation with zero initial conditions and with $K < 1$, represents a digital feedback integrator often used in radar signal processing [17]. Using the z transform, and the inverse transform, the unit sample response can be derived. Taking the z transform of both sides of the above equation yields

$$Y(z) = Kz^{-1} Y(z) + X(z),$$

and solving for $Y(z)$ yields

$$Y(z) = \frac{X(z)}{1 - Kz^{-1}}, \quad |z| > |K|.$$

If $x(n)$ is the unit sample, then $X(z) = 1$, and therefore

$$Y(z) = \frac{1}{1 - Kz^{-1}} .$$

Then the unit-sample response is $y(n)$, where

$$y(n) = \frac{1}{2\pi j} \oint_C \frac{1}{1 - Kz^{-1}} z^{n-1}\, dz$$

$$= \frac{1}{2\pi j} \oint_C \frac{z^n}{z - K}\, dz$$

$$= K^n.$$

2.6 Discrete-Time Convolution

As mentioned earlier, the unit-sample response can be used to determine the response of a linear shift-invariant system to any linear sequence. This is accomplished by a concept analogous to the convolution integral of continuous systems known as the discrete convolution. This concept can be approached from different points of view. One approach [5] considers the response of a system with unit-sample response $h(n - k)$, at a point $n - k$ sample units after a corresponding sample input of magnitude $x(k)$ at k. This impulse $x(k)$ then makes a contribution $y_k(n)$ to the total output of the system at n, where

$$y_k(n) = x(k)\, h(n - k).$$

Considering the sample to be one element of a sequence $x(n)$, the response of the system will be the sum of all the contributions $y_k(n)$. Thus

$$y(n) = \sum_{k=-\infty}^{n} x(k)h(n - k).$$

Since the impulsive response for a realizable system can be considered to be zero for all negative arguments, the upper limit of the summation can be extended to infinity without any effect on the summation; thus

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n - k),$$

which is the convolution sum. Hence $y(n)$ can be described as the convolution of the sequences $x(n)$ and $h(n)$, which is denoted as $y(n) = x(n) * h(n)$. In another approach [11] the system output $y(n)$ is taken as the sum of the system transformations $T[\ldots]$ of each of the input samples, that is,

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)T[\delta(n - k)],$$

and since $h(n)$ is the unit-sample response of a linear shift-invariant system, then

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n - k)$$

as before. By substitution of variables this summation can be alternatively expressed as

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n - k) = h(n) * x(n).$$

Thus the order of convolution is insignificant.

From the convolution sum it can be seen that the output of a linear shift-invariant system corresponding to any linear sequence can be determined from a knowledge of the unit-sample response of the system.

2.7 System Function

It is shown in Refs. 5, 11, and 15 that the z transform of the system output can be expressed as the product of the z transforms of the input sequence and the unit-sample response sequence:

$$Y(z) = X(z)H(z).$$

This can be shown by substituting the convolution sum for $y(n)$ in the defining expression for $Y(z)$, the z transform of $y(n)$, and manipulating the resulting expression into the product of z transform sum expressions for $X(z)$ and $H(z)$. The function $H(z)$, the z transform of the unit sample response, is by definition the *system function.* Although it was also referred to by Barker [18] as the *pulse transfer function.* Thus, if the system function is known, the output sequence will be the inverse z transform of the product of the system function and the z transform of the input sequence.

For a system described by linear constant-coefficient difference equations, such as given earlier in the form

$$\sum_{k=0}^{N} a_k y(n - k) = \sum_{r=0}^{M} b_r x(n - r),$$

the system function can be shown to be a ratio of polynomials in z^{-1}. This can be seen by taking the z transform of each term of the preceding equation:

$$\sum_{k=0}^{N} a_k z[y(n - k)] = \sum_{r=0}^{M} b_r z[x(n - r)]$$

or

$$Y(z)\sum_{k=0}^{N} a_k z^{-k} = X(z)\sum_{r=0}^{M} b_r z^{-r}.$$

Finally,

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{r=0}^{M} b_r z^{-r}}{\sum_{k=0}^{N} a_k z^{-k}}.$$

As stated earlier, the values of z that make $H(z)$ go to zero are the zeros of $H(z)$, and the values of z that result in an infinite $H(z)$ are the poles of $H(z)$. Just as in the case of the complex transfer function of continuous systems, the system function and thus the behavior of a discrete system are completely specified to within a multiplicative constant by the location of the poles and zeros in the complex z plane.

2.8 Stability and Causality

Any practical discrete system must possess two important properties. Such a system must be stable as well as causal.

A linear discrete system is considered to be stable if to all bounded inputs there always correspond bounded outputs [15]. From arguments in Refs. 5 and 15 it can be shown that requiring a bounded output in response to a bounded input leads to a condition on the unit-sample response that the sum of the magnitudes of its samples be bounded, that is,

$$\sum_{k=-\infty}^{\infty} |h(k)| < \infty.$$

Using this result along with the definition of $H(z)$, the preceding stability requirement is equivalent to the condition that $H(z)$ be analytic for $|z| \geq 1$. This requires that a stable linear shift-invariant discrete system, with system function $H(z)$, have no poles which lie outside the unit circle of the z plane. It is shown in Ref. 11 that if the region of convergence of the system function $H(z)$ contains the unit circle, the corresponding discrete system is stable.

In the case of continuous systems it is not generally convenient to determine the stability of a system by locating its poles and zeros. Likewise the same situation applies in determining the stability of discrete systems. As in the case of continuous systems, other methods which do not require the determination of pole and zero locations have been developed. Such methods are described in Refs. 5 and 15. These include discrete system variations of the Routh-Hurwitz criterion and root locus methods commonly used for stability determination in continuous systems.

A causal system is one for which the output response does not precede the application of the input sequence. This property must apply of course to any discrete system realizable in practice. By definition, for a linear shift-invariant system to be causal its unit-sample response must be zero for $n < 0$. It is shown in Ref. 11 that this will be the case only if the region of convergence of the system function $H(z)$ includes $z = \infty$.

3. RELATION BETWEEN DISCRETE AND CONTINUOUS SYSTEMS

3.1 Fourier Transforms of Continuous and Discrete Signals

From the theory of linear time-invariant continuous systems it is well known that the Fourier transform [19] is a useful tool in the decomposition of a signal into its frequency components. The Fourier transform, expressible as

$$X(j\omega) = F[x(t)] = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}\,dt,$$

gives the amplitude of the signal as a continuous function of frequency. The Fourier transform can be alternately represented as $X(\omega)$ or, since $\omega = 2\pi f$, $X(f)$. This transform is invertible and thus, from the continuous frequency spectrum, the function in the time domain can be recovered as

$$x(t) = F^{-1}[X(j\omega)] = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(j\omega)e^{j\omega t}\,d\omega.$$

Fourier transforms can be represented as sets of transform pairs of time functions and their corresponding Fourier transform or spectrum.

The Fourier transform of an infinite sequence of discrete samples can be represented [6] as

$$X(e^{j\theta}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\theta n},$$

with its inverse transform being

$$x(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\theta})e^{j\theta n}\,d\theta,$$

where $\theta = \omega T$ is the angular frequency on the unit circle with respect to the sampling frequency $1/T$. This Fourier transform is a continuous function of θ although $x(n)$ is discrete.

It can be shown [11] that the preceding transform pair, for discrete signals, aids in sinusoidal signal analysis and is related to the z transform. Similar to continuous systems,

the steady-state response to a sinusoidal input is sinusoidal, of the same frequency as the input, with amplitude and phase modified as a function of the particular system characteristics. Signals can be represented in terms of sinusoids or complex exponentials, thus simplifying system analysis. With the input $x(n) = e^{j\theta n}$ to a system of unit sample response $h(n)$, by the convolution sum the output response $y(n)$ is

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)e^{-j\theta(k-n)}$$

$$= e^{j\theta n} \sum_{k=-\infty}^{\infty} h(k)e^{-j\theta k}.$$

If

$$H(e^{j\theta}) = \sum_{k=-\infty}^{\infty} h(k)e^{-j\theta k},$$

then

$$y(n) = H(e^{j\theta})e^{j\theta n}.$$

$H(e^{j\theta})$ is the *frequency response* of the system. It can be seen from its defining equation to be the Fourier transform of the unit-sample response. From the equation for $y(n)$ the output response is of angular frequency θ, with the magnitude and phase of $H(e^{j\theta})$ determining the output response to a complex exponential input. It can be seen that the frequency response is the z transform of a sequence evaluated for $z = e^{j\theta}$. Thus the frequency response, or Fourier transform, of a sequence is its z transform evaluated on the unit circle.

Two important extensions of the Fourier transform are the *convolution theorem* and *frequency convolution theorem*, proofs of which appear in Ref. 20. The convolution theorem gives a Fourier-transform pair relation between the convolution of time functions and the product of their Fourier transforms, that is,

$$F[x(t) * h(t)] = X(j\omega)H(j\omega).$$

The frequency convolution theorem is analogous and gives a Fourier-transform pair relation between the product of time functions and the convolution of their Fourier transforms. Simply stated, convolution in the time domain is equivalent to multiplication in the frequency domain, and multiplication in the time domain is equivalent to convolution in the frequency domain.

3.2 Laplace and z Transform Relations

The Fourier transform for continuous functions is a generalization of the Laplace transform, being the Laplace transform evaluated on the imaginary axis of the complex

s plane. Likewise the Fourier transform for discrete signals is the z transform evaluated on the unit circle of the complex z plane.

Consider a sequence $x(n)$ derived from sampling with period T a continuous function $x_c(t)$, so that $x(n) = x_c(nT)$. There is a relationship between $X(z)$, the z transform of $x(n)$, and $F_c(s)$, the Laplace transform of $x_c(t)$, which is derived in Ref. 5 as well as in Ref. 15 and was discovered originally by Poisson. This relationship, which implies a mapping between the s plane and z plane, is

$$X(z)|_{z=e^{sT}} = \sum_{n=-\infty}^{\infty} x_c(nT)e^{-snT} = \frac{1}{T}\sum_{n=-\infty}^{\infty} F_c\left(s + j\frac{2\pi}{T}n\right).$$

This mapping from the s plane to the z plane is not one to one. The mapping between the two planes is shown in Fig. 3.1, taken from Ref. 6. From $z = e^{st}$ it follows that strips of width $2\pi/T$ in the s plane map onto the entire z plane [11]. The left half of each strip in the s plane maps onto the interior of the unit circle, and the right half of each strip maps onto the exterior of the unit circle. Each segment of the imaginary axis in the s plane maps onto the unit circle.

3.3 Sampling of Continuous Time Signals

Most signals considered for processing originate in a continuous-time form. To process these signals by means of the discrete systems and related algorithms discussed here, it is necessary to represent them in the discrete-signal form of the sequences discussed earlier. These sequences are obtained by periodic sampling of the continuous-time signal. Because of the necessarily finite speed and data-storage capabilities of practical systems it is desired to keep the signal sample rate to a minimum.

The sampling of a continuous signal $x(t)$ by impulse sampling is presented in Ref. 5 as well as in Ref. 20. If $\delta(t)$ is the unity impulse function of value unity at $t = 0$ and value zero everywhere else, then $\delta(t - nT)$ is zero everywhere but unity at $t = nT$. Let $\Delta(t)$ represent an impulse train which consists of an infinite set of unity impulses separated in time by an interval T. Then $\Delta(t)$ can be represented mathematically as

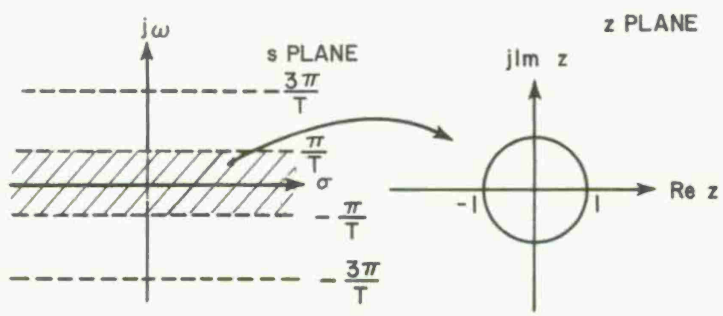


Fig. 3.1—The mapping of the s plane to the z plane implied by sampling a continuous-time signal. (From Ref. 6 by permission.)

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

The sampling process can then be described as a modulation of $\Delta(t)$ by the continuous signal $x(t)$; therefore

$$x(n) = x(t)\Delta(t).$$

From the definition of $\Delta(t)$ and the fact that the only values of $x(t)$ of interest are those at $t = nT$, $x(n)$ is more precisely represented as

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

Since $x(n)$ is formed from the product of $x(t)$ and $\Delta(t)$, by the frequency convolution theorem the spectrum (Fourier transform) of $x(n)$ is the convolution of the Fourier transforms of $x(t)$ and $\Delta(t)$. From Ref. 20 the spectrum of $x(n)$ is found to be the spectrum of $x(t)$ infinitely repeated at intervals $1/T$ for both positive and negative frequencies. If for example the spectrum of $x(t)$ is as indicated in Fig. 3.2a, the spectrum of $x(n)$ is as shown in Fig. 3.2b.

If as indicated in Fig. 3.2a, the spectrum of the continuous-time signal is *band limited*, that is, zero outside the region $|f| < f_c$, the original signal can be reconstructed from $x(n)$ exactly by a low-pass filter which passes, without alteration, only signal frequency components in the interval $|f| < f_c$. Several factors should be noted from the previous discussion. First, if the spectrum of $x(t)$ is not strictly limited and has frequency components such that the periodic spectrum of $x(n)$ overlap, there will be a distortion in the spectrum of the recovered signal. Second, even if the frequency components are band

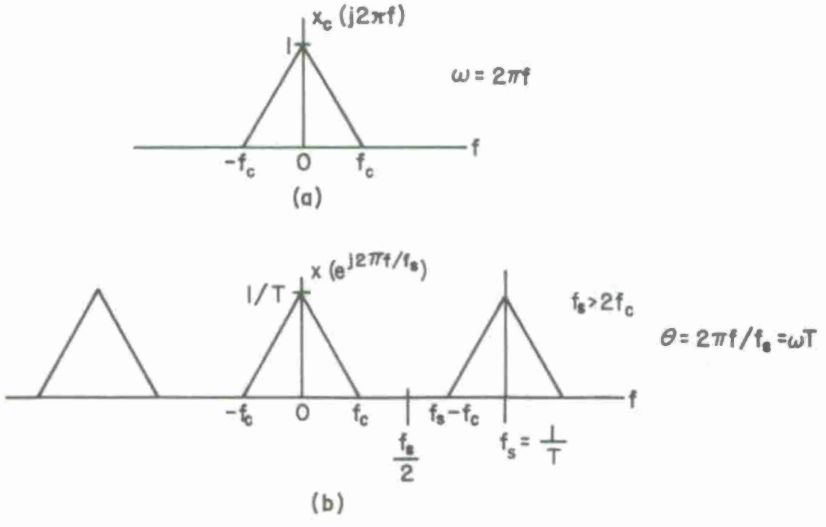


Fig. 3.2—The spectrum of a continuous-time signal and the spectrum of the digital signal resulting from sampling. (From Ref. 6 by permission.)

limited to some $|f| < f_c$, there will also be a distortion in the sampled signal if T does not satisfy the inequality $f_c < 1/2T$. This distortion, which results from a sampling rate $1/T$ that is not high enough (T too large) in relation to the largest frequency component in the signal $x(t)$, is referred to as *aliasing*. The term aliasing is due to the manner in which higher frequencies masquerade as lower frequencies due to the spectrum overlap. A simple way to envision aliasing is to consider a signal with sinusoidal components including frequencies that exceed $1/2T$, half the sampling rate. Samples of components of frequency beyond $1/2T$ can upon sampling appear as samples from lower frequency components. Thus the only way to avoid aliasing is to insure that the sampling rate is at least the *Nyquist rate*, that is, twice the frequency of the highest component in the signal.

These ideas with respect to sampling were first manifested in communication theory in the form of the *sampling theorem* [21]. This theorem, proven in Refs. 5, 11, and 20 as well as 21, states that if a signal $x(t)$ is band limited with spectrum zero for $|f| > f_c$ and if $T = 1/2f_c$, then $x(t)$ can be unambiguously reconstructed from its samples

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

and the recovered signal will be

$$x(t) = \sum_{n=-\infty}^{\infty} x(nT) \frac{\sin 2\pi f_c(t - nT)}{2\pi f_c(t - nT)} .$$

3.4 Equivalence of Analog and Digital Signal Processing

The equivalence of signal processing in analog and digital realizations provides for application of the wealth of available knowledge and techniques developed for analog designs to digital implementations with the inherent advantages of the latter. The equivalence between time-invariant, continuous and discrete systems was addressed by Steiglitz [7] and Gibbs [22]. A specific isomorphism between the analog and digital signal spaces was shown to exist. Although the natural correspondence provided by the instantaneous sampling of continuous signals would be a match of e^{st} with z, this mapping is not one to one. An isomorphic mapping is, however, provided by the bilinear transformation

$$s = \frac{z - 1}{z + 1} \quad \text{and} \quad z = \frac{1 + s}{1 - s} .$$

This specific isomorphism results in a matching of continuous signals with rational transforms in s with discrete signals with rational transforms in z as well as a match between time-invariant realizable continuous transforms and time-invariant realizable discrete transforms.

Stieglitz considered optimization problems for both continuous and discrete signals using a least-integral-square-error criterion. His analysis applied to both deterministic and random signals under the assumption of the isomorphism between continuous and discrete signals and the existence of a class of continuous filters providing minimization of some function. The resulting theorems state an equivalence to discrete filters in the sense that a solution in the continuous case confirms the existence of a solution in the discrete case.

4. DIGITAL NETWORKS

4.1 Digital Network Elements

With reference to the earlier discussion on system functions, linear shift-invariant systems to be discussed here can be represented by system functions of the form

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

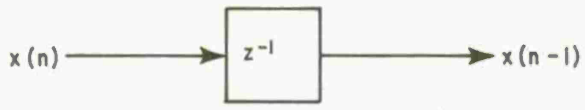
The output sequence of such systems can then be represented as

$$y(n) = -\sum_{k=1}^{N} a_k y(n - k) + \sum_{k=0}^{M} b_k x(n - k).$$

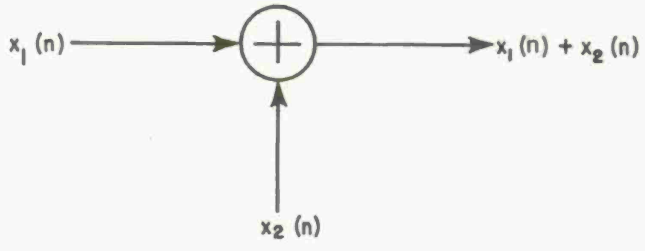
These systems can be realized by a direct application of the preceding difference equation. Thus the delayed inputs and outputs are obtained, multiplied by coefficients, and summed as indicated in the equation. To carry out these operations in a block-diagram representation or practical implementation requires the definition and use of certain arithmetic network elements [6] as shown in Fig. 4.1. Figure 4.1a is the diagrammatic symbol for the unit-delay operator z^{-1}. Addition is indicated as shown in Fig. 4.1b, where the two inputs, $x_1(n)$ and $x_2(n)$, are summed to form $x_1(n) + x_2(n)$. Multiplication by a constant is represented as shown in Fig. 4.1c, where $x(n)$ is multiplied by a to form $ax(n)$. The element indicated in Fig. 4.1d realizes the branching operation, with input $x(n)$ branching out to various points of a network as necessary. As an example of the use of the above network elements, consider the block diagram representing the difference equation $y(n) = -a_1 y(n - 1) - a_2 y(n - 2) + b_0 x(n) + b_1 x(n - 1)$, as shown in Fig. 4.2. These network elements will normally be implemented in the binary arithmetic system. In that case the network elements will be formed from basic binary logic gates and flipflops.

4.2 Representation of Digital Networks by Signal Flow Graphs

A digital network can be represented by a connection of directed branches which interconnect at nodes and are known as linear signal flow graphs. The details of the

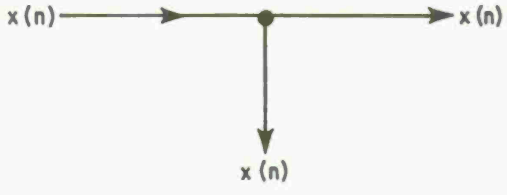


(a) Unit Delay



(b) Adder

(c) Constant Multiplier



(d) Branch

Fig. 4.1—Digital network elements

theory and applications of linear signal flow graphs are presented in Refs. 23, 24, and 25. The application of signal flow graphs to digital networks is discussed in Ref. 11. The graphs can be used to represent z transform relationships and as such have been used to provide a general representation of digital networks by matrices as described in Ref. 26.

For this presentation it will suffice to limit discussion to the representation of digital networks by signal flow graphs. Each branch in a digital network represents a network element that can be replaced by a directed branch along with an indication of the transmittance function between branch input and output, with the absence of such a function indicating unity transmittance. With respect to the nodes there are source nodes representing the network inputs, sink nodes representing network outputs, summation nodes, with multiple inputs and a single output, representing the addition of all entering branches, and branch nodes, with a single input and multiple outputs, indicating the branching out of the entering branch. As an example of the application of linear signal flow graphs to digital networks, Fig. 4.3 is a representation of the digital network in

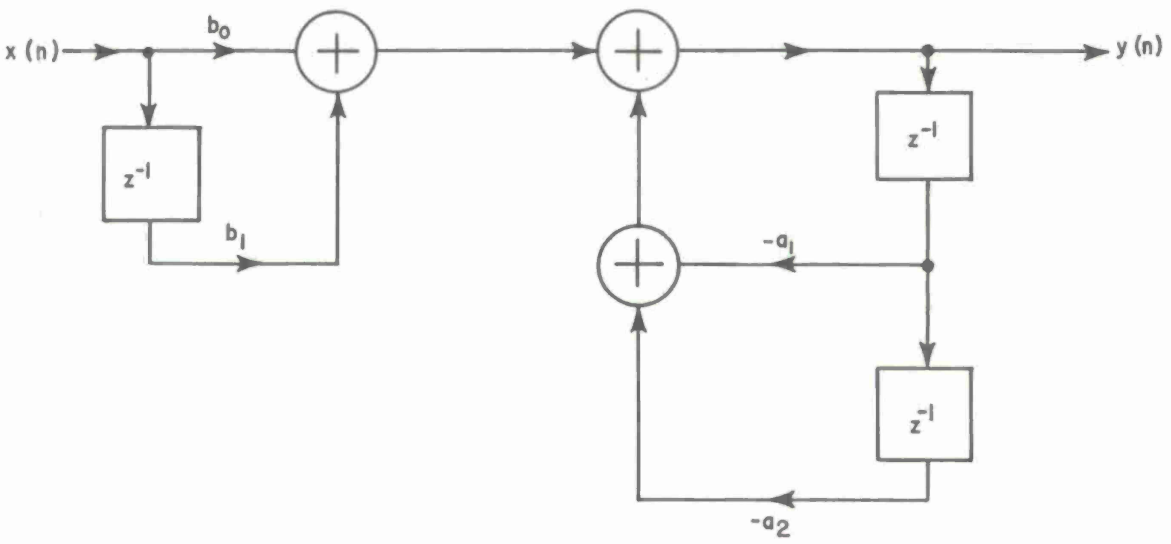


Fig. 4.2—Digital network realization of $y(n) = -a_1 y(n-1) - a_2 y(n-2) + b_0 x(n) + b_1 x(n-1)$

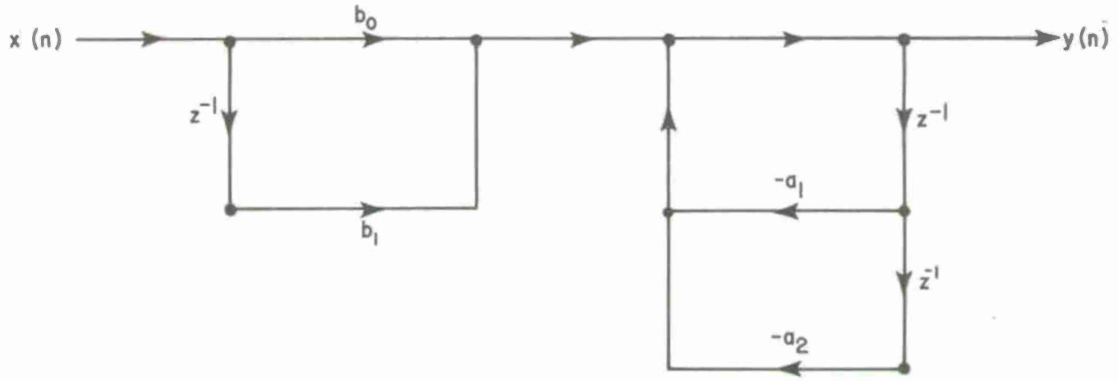


Fig. 4.3—Digital signal flow graph of the network of Fig. 4.2

Fig. 4.2 in signal-flow-graph notation. Mason's rule [23], a method of evaluating the transfer function of a network from its signal flow graph, can be applied to digital signal flow graphs in z transform notation to determine system functions.

4.3 IIR Network Structures

Many equivalent digital networks can be used to realize a particular system function. Networks with both poles and zeros, that is IIR networks, will be discussed here. As discussed previously, many such networks can be represented in the form of a rational system function:

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

In subsection 4.1 the realization of a digital network from its linear difference equation was demonstrated. This method can be generalized for any integral values of M and N. The transmittance of the branches is determined by the coefficients of the difference equation or system function. The canonical forms of $H(z)$ as discussed in Ref. 27 will be presented here.

The form shown in Fig. 4.4 and referred to as direct form I is a direct realization from the coefficients and values of M and N appearing in the system function. For ease of representation it will be assumed for this discussion that $M = N$. By separating direct form I into two networks of all poles and all zeroes and reversing their order, Oppenheim and Schafer [11] derive the direct form II with minimum number of multiplier, adder, and delay elements, as shown in Fig. 4.5. Kaiser [28], has recommended that direct forms not be used in high-order systems due to the accuracy required in order to avoid severe errors in performance.

The cascade canonic form is obtained by factoring the numerator and denominator of $H(z)$ and forming a product of ratios of second-order polynomials. Thus

$$H(z) = b_0 \prod_{n=1}^{m} \frac{1 + \beta_{1i} z^{-1} + \beta_{2i} z^{-2}}{1 + \alpha_{1i} z^{-1} + \alpha_{2i} z^{-2}} ,$$

where m is the integer part of $(N + 1)/2$. If N is odd, that is, if there are an odd number of poles and zeros, then α_{2i} and β_{2i} for some i will be 0. Thus the system function can

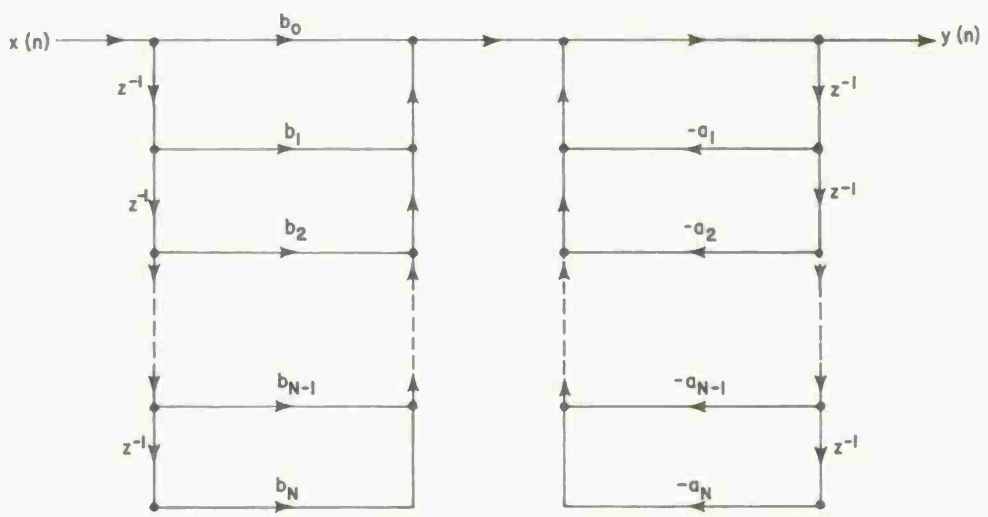


Fig. 4.4—Direct-form-I of realization of $H(z)$

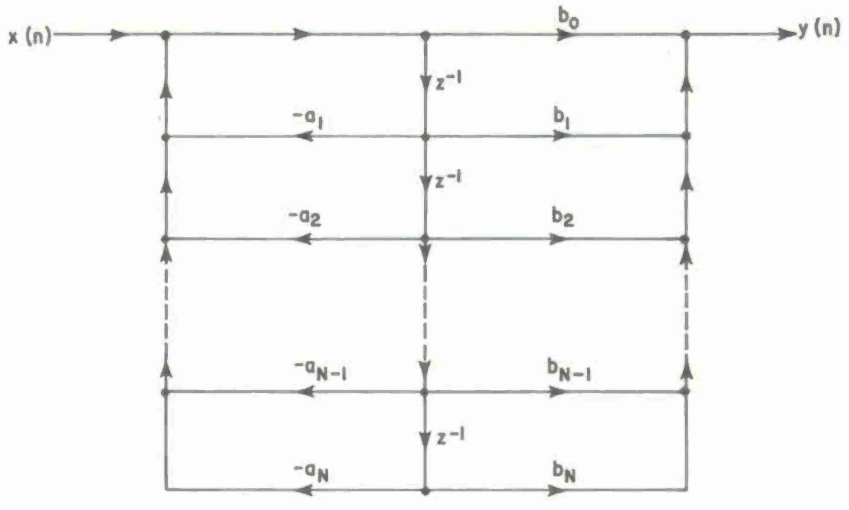


Fig. 4.5—Direct-form-*II* of realization of $H(z)$

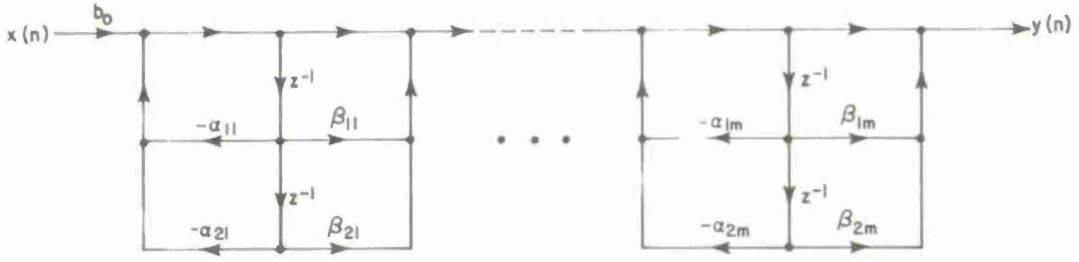


Fig. 4.6—The cascade-form realization of $H(z)$

be realized by a cascade of generalized second-order sections, as shown in Fig. 4.6. Each second-order section is in direct form *II*. Networks using these sections can be equivalently formed by any ordering of the poles and zeros of the sections. Although the resulting networks are equivalent for infinite precision representation and arithmetic considerations, the performance of practical implementations will vary due to quantization effects that will be discussed later.

The parallel canonic form results from a partial-fraction expansion of the rational form of $H(z)$. If it is assumed again that $M = N$ and m is the integer part of $(N + 1)/2$, then

$$H(z) = \gamma_0 + \sum_{i=1}^{m} \frac{\gamma_{0i} + \gamma_{1i} z^{-1}}{1 + \alpha_{1i} z^{-1} + \alpha_{2i} z^{-2}} ,$$

where $\gamma_0 = b_N / a_N$.

If N is odd, some γ_{1i} and α_{2i} will be zero. Thus $H(z)$ can be realized by a parallel combination of general second-order forms, as shown in Fig. 4.7. Again, each second-order section is realized in direct form *II*.

4.4 FIR Network Structures

The terms FIR and IIR refer to the characteristics of the response of a digital system rather than the realizations which would be referred to as recursive or nonrecursive [29]. Recursive realizations have outputs which are a function of past outputs as well as past and present inputs; nonrecursive realizations have outputs which are a function of past and present inputs only [30]. Both FIR as well as IIR systems can be realized by means of either recursive or nonrecursive algorithms [31].

A nonrecursive realization of an FIR system can be implemented by means of the direct convolution sum

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n - k),$$

where $h(k) = b_k$, or by setting all denominator coefficients a_k in the general expression for $H(z)$ equal to 0 [32]. The resulting direct-form realization is shown in Fig. 4.8.

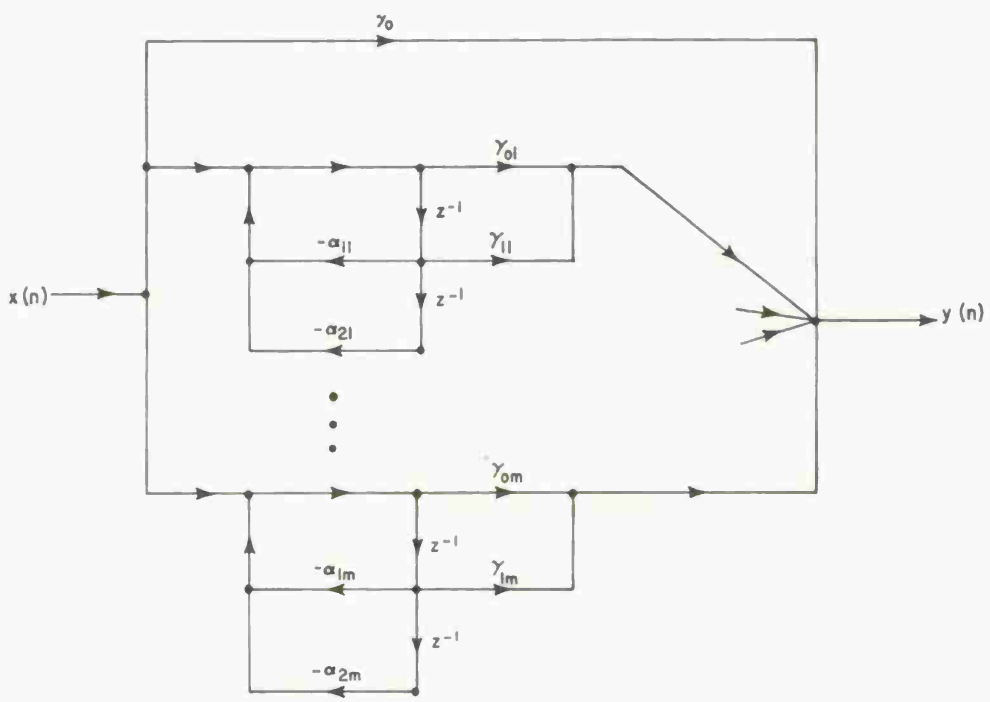


Fig. 4.7—The parallel-form realization of $H(z)$

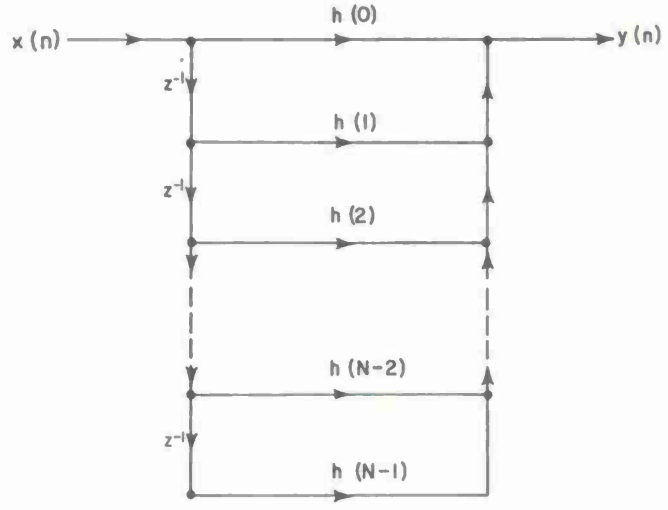


Fig. 4.8—Direct form of an FIR system

An alternative form is presented in Ref. 11 based on the system function which can be written as

$$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n}$$

for an FIR system. The $H(z)$ can be expressed as a product of second-order factors,

$$H(z) = \prod_{k=1}^{M} (\beta_{0k} + \beta_{1k} z^{-1} + \beta_{2k} z^{-2}),$$

where M is the largest integer in $N/2$ and, if N is even, β_{2k} will be 0 for some k. The corresponding network is then a cascade of general second-order sections, as shown in Fig. 4.9.

The frequency sampling technique [30], which will be discussed later in connection with the design of FIR filters, leads to a structure which is an example of an FIR system realized by a recursive algorithm. In this case the system function can be expressed in the form

$$H(z) = (1 - z^{-N}) \frac{1}{N} \sum_{k=0}^{N-1} \frac{H_k}{1 - z^{-1} e^{j(2\pi/N)k}}.$$

This is a cascade of an FIR-system function $1 - z^{-N}$ with zeros at $e^{j(2\pi/N)k}$, described as a comb filter, and of an IIR system. The IIR system is the parallel combination of N single-pole filters with poles at $z_k = e^{j(2\pi/N)k}$ and is described as a resonator. Each of

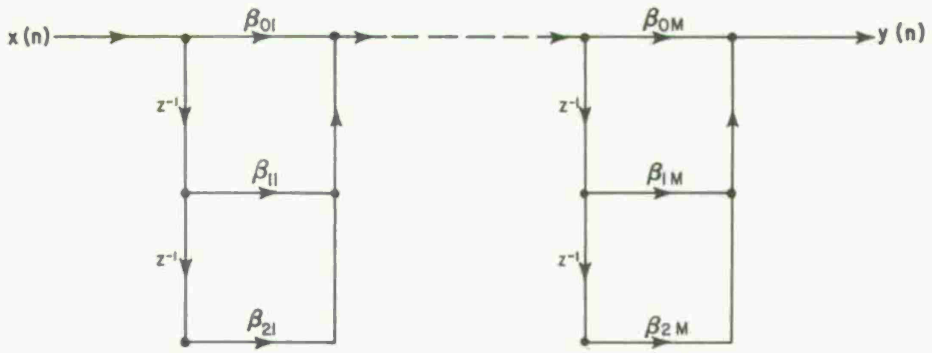


Fig. 4.9—Cascade form of an FIR system

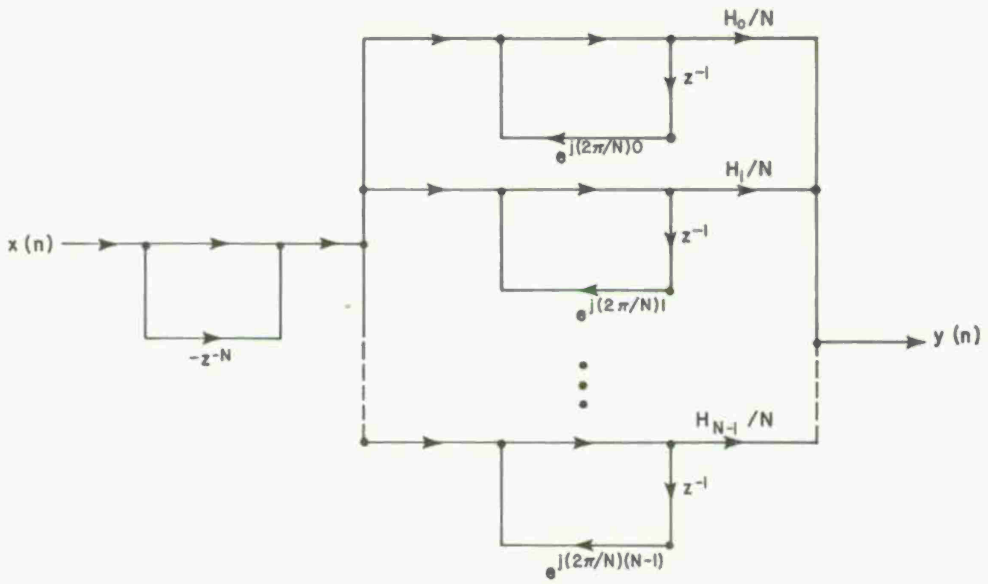


Fig. 4.10—Frequency-sampling realization of an FIR system

the resonator poles cancel a zero of the comb filter and its conjugate. The resonator used to cancel the kth zero is referred to as the kth elemental filter. The outputs of the elemental filters are weighted by the H_k and are summed to form the system output. The H_k represent "samples" of the desired frequency response equally spaced around the unit circle. From Ref. 33, the structure of such an FIR system is as shown in Fig. 4.10.

4.5 Transpose of a Digital Network

Mason and Zimmerman [23], in a discussion of linear flow graphs, present a concept of "reversal" of a flow graph. With reference to Mason's formula for the transmission of a multiloop graph, the reversal of the directions of all branches in a graph

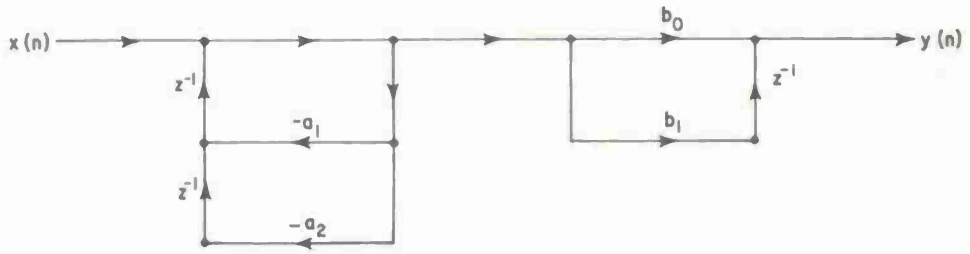


Fig. 4.11—Transpose form of the signal flow graph of Fig. 4.3

along with interchange of network input and output results in a new graph of identical transmission. Alternate proofs are presented in Refs. 25 and 34. As applied to digital networks, a *transpose* network of identical system function can be obtained from a known network by reversing the direction of all branches and interchanging input and output, with all branch transmittances remaining fixed.

Thus, for each of the digital network structures presented here, a transpose structure can be obtained. As an example, the signal flow graph in Fig. 4.11 represents the transpose of the flow graph of Fig. 4.3. Some networks are their own transpose. Although a digital network and its transpose would have identical system functions for infinite precision, in practical implementations one form will generally be more desirable due to errors caused by finite quantization effects [34].

4.6 Other Canonic Realizations of Digital Networks

As mentioned many realizations for an arbitrary digital system function are possible, but each has different characteristics with respect to quantization effects. It is therefore desirable to have a number of realizations of a given system function available in order to choose the one with the best performance.

In addition to the basic structures presented previously, a number of additional network forms have been developed recently. These developments have been based on a method presented by Mitra and Sherwood [35]. Their method uses continued-fraction expansion of a digital transfer function expressed as a real rational function in z in the form

$$G(z) = \frac{a_n z^n + a_{n-1} z^{n-1} + \cdots a_1 z + a_0}{b_n z^n + b_{n-1} z^{n-1} + \cdots b_1 z + b_0} .$$

Different expansions of $G(z)$ result in four canonic realization forms, each resembling a ladder. The realizability of each form depends on the existence of the associated continued-fraction expansion, which can be readily determined.

As an example of one form of such a realization development consider $G(z)$ for nonzero a_n, b_n, and b_0 such that $G(z)$ has the resulting continued fraction expansion

$$G(z) = A_0 + \cfrac{1}{B_1 z + \cfrac{1}{A_1 + \cfrac{1}{B_2 z + \cfrac{1}{\cdots \\ \cdots \\ B_n z + \cfrac{1}{A_n}}}}}$$

To implement this function, subnetworks of the form

$$G_1(z) = \frac{1}{Bz + T(z)}$$

and

$$G_2(z) = \frac{1}{A + T(z)}$$

are required with A and B real. The realizations of these subnetworks are shown in Fig. 4.12 for $G_1(z)$ and Fig. 4.13 for $G_2(z)$. To apply these subnetworks, $G(z)$ is written as

$$G(z) = A_0 + \frac{1}{B_1 z + T_1(z)},$$

where

$$T_1(z) = \cfrac{1}{A_1 + \cfrac{1}{B_2 z + \cfrac{1}{A_2 + \cfrac{1}{\cdots \\ \cdots \\ B_n z + \cfrac{1}{A_n}}}}}.$$

The second term of $G(z)$ is in the form of $G_1(z)$ and can be realized as shown in Fig. 4.12. $T_1(z)$ is next written in the form of $G_2(z)$ and realized accordingly. The process is continued until all terms of the expansion are exhausted. A realization of the form of Fig. 4.14 results.

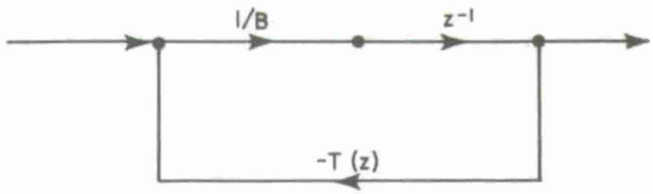


Fig. 4.12—Realization of $G_1(z) = \dfrac{1}{Bz + T(z)}$.
(From Ref. 35 by permission.)

Fig. 4.13—Realizations of $G_2(z) = \dfrac{1}{A + T(z)}$.
(From Ref. 35 by permission.)

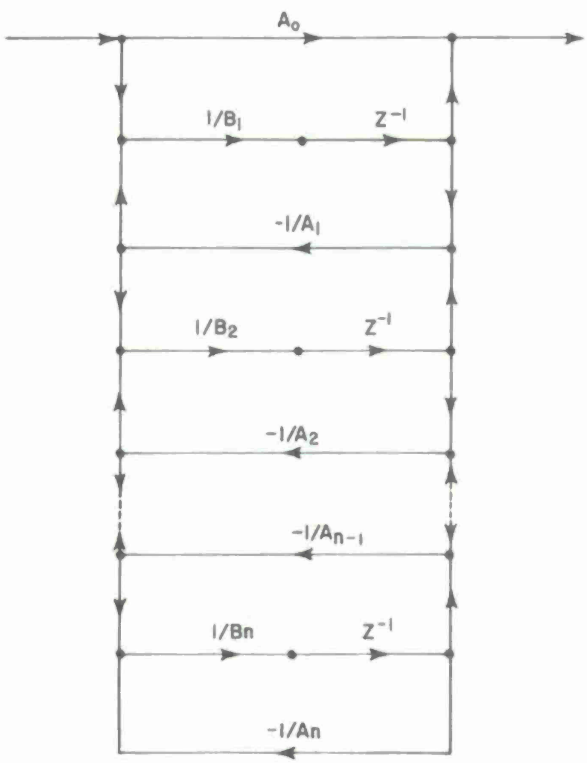


Fig. 4.14—Continued-fraction realization.
(From Ref. 35 by permission.)

In Ref. 36 Mitra and Sagar present three additional network structures derived by continued-fraction expansion. A realization of an arbitrary system function in the form of a cascade of digital two-pairs and using continued-fraction expansions is presented in Refs. 37 and 38. Hwang [39] presents formal realization procedures which use repeated divisions and order reductions or continued-fraction expansions. Including the forms discussed previously, Hwang obtains 14 basic canonical forms.

5. DIGITAL FILTER THEORY AND DESIGN

5.1 General Filter Properties

A filter is generally a system designed to shape the frequency spectrum of a given signal in some desired fashion. The type of filters considered here are within the class of linear time-invariant systems. For continuous filters the signals at all points in the system are continuous. For a digital filter the signals considered are represented only at quantized amplitudes and discrete time intervals and all operations within the system use finite precision arithmetic. Both analog and digital filters are most often specified in the frequency domain. Therefore a frequency response characteristic, or variation of the magnitude of the filter attenuation with frequency as independent variable, is specified as the design goal. A filter is generally categorized in terms of its relative frequency-passband behavior as lowpass, highpass, bandpass, or band elimination. Analog filters are specified in terms of analog frequency or cycles per second (hertz) whereas digital filters are more suitably specified in terms of phase angle on the unit circle, with 2π representing the sampling frequency (f_s) and π representing the folding frequency ($f_s/2$). Translation from analog to digital frequency or vice versa is readily accomplished.

Within the past half century a wealth of knowledge has developed with respect to continuous filter design. Such information can be found for example in Refs. 40, 41, and 42. To make full use of this knowledge, an important class of digital filter designs are based on translations of a known continuous filter design to a digital filter design. In the design of continuous filters it is well known that many "ideal" designs are not practically realizable. The resulting approximation problem also applies in the case of digital filters and is identical to that of the continuous case in the sense that if solvable in the one case it is solvable in the other [7,22]. In light of these factors a general filter specification is presented in the form of an approximate magnitude-squared characteristic with tolerance regions as shown in Fig. 5.1 [12]. A lowpass characteristic is indicated as an example, but the terminology is applicable to filters of other frequency-selectivity classes. Thus the *passband* is the frequency region in which the magnitude squared of the frequency response is between $1/(1 + \epsilon^2)$ and unity. The *stopband* is that region of magnitude-squared frequency response between zero and $1/A^2$. The oscillatory variation of a filter's response characteristic with increasing frequency within the above tolerance regions is referred to as *ripple*. It can be seen from Fig. 5.1 that ω_p is the upper frequency limit of the passband and ω_s is the lower frequency limit of the stopband. The region between the passband and stopband is the transition band. The width of the transition band is $\omega_s - \omega_p$, and the minimization of this band is often a desired design goal, since it determines the *sharpness* of the filter response characteristic. As an example of the extension of this terminology the frequency characteristic of a bandpass filter will have a finite passband with a transition band and stopband pair, above and below the passband. The Butterworth, Chebyshev, and elliptic filters, whose

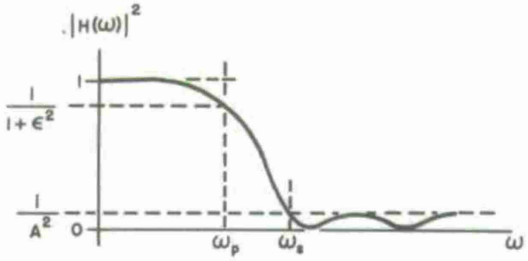


Fig. 5.1—Example of the magnitude-squared characteristic of a typical filter

squared-magnitude characteristics appear in Fig. 5.2, are those continuous filter forms commonly applied to digital filter designs. To apply one of these continuous filter characteristics, it is generally required to obtain the transfer function in terms of the specifications desired.

5.2 Design of IIR Filters

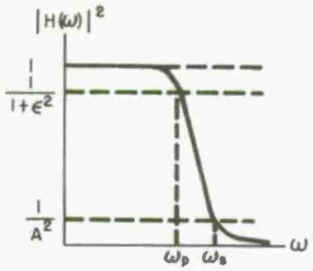
The basic form of the system function as presented earlier for an IIR digital filter is a ratio of polynomials in z^{-1}. The coefficients a_i and b_i of this basic form determine the number and location of its z-plane poles and zeros, and thus the frequency response, of the filter. It follows then that the design of such filters involves the determination of these coefficients so as to satisfy a desired filter specification. The coefficients could be so determined directly from the filter specifications [32]. The common approach however is to determine the system function coefficients indirectly by finding a suitable continuous filter with system function $H_c(s)$ and performing a translation to a discrete system function $H(z)$. Some of the more common techniques for performing such translations will be presented here.

5.2.1 *Impulse-Invariance Technique*

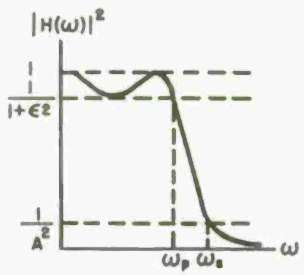
In translating a continuous filter design, of desired specifications, into a digital filter design, an *impulse-invariant* approach can be taken. This involves deriving a digital filter with unit-sample response equivalent to the sampled inpulse response of the given continuous filter. This technique is described in Refs. 30 and 43, appearing in the latter as the standard z-transform method.

It is assumed that the continuous filter used has a transfer function of the form

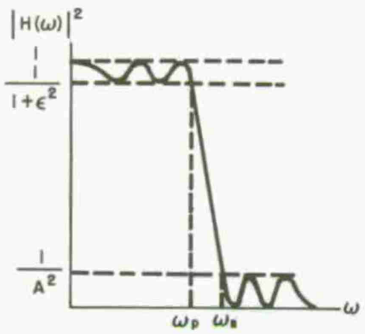
$$H_c(s) = \frac{\sum_{k=0}^{M} d_k s^k}{\sum_{k=0}^{N} c_k s^k}, \quad M < N,$$



(a) Butterworth



(b) Chebyshev



(c) Elliptic

Fig. 5.2—Magnitude-squared characteristics of the standard forms of continuous-time filters

with distinct poles such that by partial-fraction expansion

$$H_c(s) = \sum_{k=1}^{N} \frac{A_k}{s + \alpha_k}, \quad M < N.$$

With multiple-order poles this partial-fraction expansion must be appropriately modified. When the impulse-invariant restraint is imposed, then

$$h(n) = h_c(nT),$$

which implies the translation

$$\frac{1}{s + \alpha_k} \rightarrow \frac{1}{1 - \epsilon^{-\alpha_k} Tz^{-1}}$$

from each pole term in $H_c(s)$ to the corresponding term in $H(z)$. Thus for an appropriate continuous filter system function a partial-fraction expansion is performed, the term-by-term translation is implemented, and a rational system function in z^{-1} is obtained which can be realized by one of the structures described previously. One problem in this design technique is caused by spectrum folding, which causes the frequency response of the digital filter to differ from that of the continuous filter for $H_c(s)$ not bandlimited. $H_c(s)$ will not be bandlimited when it is a rational function [43]. Thus the impulse-invariant technique must be limited to narrowband applications, or a bandwidth-limiting guard filter $G(s)$ must be used in cascade with the transformation then applied to $H_c(s)\,G(s)$.

5.2.2 *Bilinear Transformation Technique*

To overcome the folding problem of the impulse-invariant design method, an s-plane-to-z-plane transformation is required that maps the entire imaginary axis in the s plane onto the unit circle in the z plane in a one-to-one fashion. A transformation that accomplishes this mapping is the bilinear transform

$$s = \frac{2}{T}\,\frac{1 - z^{-1}}{1 + z^{-1}}\,,$$

whose properties are discussed in Ref. 22.

The bilinear transformation design technique is discussed in Refs. 28, 30, and 44. The transform is used by substituting for s the preceding bilinear relation in the system function $H_c(s)$ of a given continuous filter of desired frequency-response characteristic. Thus

$$H(z) = H_c(s)\Big|_{s = \frac{2}{T}\,\frac{1 - z^{-1}}{1 + z^{-1}}}\,,$$

and the digital filter is realized using the resulting coefficients of $H(z)$. This transformation results in a nonlinear warping of the frequency relationship between the continuous frequency ω_c and the digital frequency ω_d, described by

$$\frac{\omega_c T}{2} = \tan\,\frac{\omega_d T}{2}\,.$$

To compensate for this frequency distortion, it is necessary to prewarp the continuous filter design such that the critical frequencies will be shifted to required values in the resulting digital filter design, as demonstrated in Ref. 44. Practical applications of this design technique appear in Refs. 11 and 12.

5.2.3. *Other IIR Design Methods*

Various other IIR design methods are available. Rader and Gold [30] present a method of obtaining a design from a digital squared-magnitude function. The function

$|H(z)|^2$ is obtained and, using complex plane transformations, the z-plane poles and thus the final design are obtained. Kaiser [32] discusses the Boxer-Thaler method, which involves substitution of tabulated [45] z-form expressions for each power of s^{-1} in a desired-continuous-filter-system function expressed in powers of s^{-1} instead of s. Oppenheim and Schafer [11] discuss a design based on numerical solution of the differential equation describing a continuous filter. This method leads to a mapping from the s plane to the z plane, requires high sampling rates well beyond twice the Nyquist frequency, and is suitable only for lowpass filters.

5.2.4. *Computer Methods*

One approach to digital filter design, mentioned previously, involves a direct approach whereby the filter coefficients are determined by some computation procedure directly from the desired filter characteristics [32]. Such techniques would involve some form of iterative approach to an optimized or minimum-error design based on some approximation criteria. A few of the methods presented in the literature will be discussed here.

Steiglitz [46] proposed a method for IIR filter design with arbitrary specification of system-function magnitude. The cascade structure is assumed, and the Fletcher-Powell [47] optimization procedure is used to determine the filter coefficients based on a square-error minimization in the frequency domain. The filter stability is maintained and phase is minimized by constraining poles and zeros respectively to the interior of the unit circle.

Optimization techniques such as used in Ref. 46 result in coefficients of continuous resolution. Suk and Mitra [48] propose a random search technique that operates for integer-valued functions. The technique is applied to the design of digital filters with finite word length. The basic step in the random-search optimization scheme is the search for a new optimum design vector X^1 from a previous point X by $X^1 = X + \Delta X$, where ΔX is generated randomly according to a prescribed probability-density function.

5.3 Design of FIR Filters

The use of FIR implementations to achieve desired filter characteristics has certain advantages over IIR counterparts. FIR filters can provide accurate approximations to arbitrary frequency characteristics as well as exactly linear phase. Additionally, FIR filters have stability and quantization-effect properties that are superior to those of IIR filters [29]. Various design techniques have been developed for IIR filters. The windowing technique is the most widely used of these.

5.3.1. *Windowing Design Technique*

The design of FIR filters using windows is described by Kaiser [43]. A desired continuous-frequency-response characteristic $H(\omega)$ can be expanded in a Fourier series. The resulting coefficients are then the coefficients of the impulse response $h(n)$ of the filter. In general the impulse response $h(n)$ will be infinite. To obtain a finite response, it is necessary to truncate the terms of the Fourier series. If, as will be the general case,

the Fourier series does not converge rapidly so as to make the truncation error negligible, the coefficients of the unit-sample response must be modified. The unit-sample response can be truncated by multiplying $h(n)$ by a windowing function $w(n)$. Since multiplication in the time domain is convolution in the frequency domain, the resulting frequency response characteristic will be the convolution of the Fourier transforms of $h(n)$ and $w(n)$. Due to the Gibbs phenomenon of Fourier series, a ripple with fixed percentage overshoot at approximated discontinuities appears in the resulting frequency characteristic.

To reduce the truncation error as well as the effects of the Gibbs phenomenon, a series of window functions have been developed. These windows are time-limited even functions and are tapered smoothly to zero at either end. These window functions generally have reduced sidelobes in their Fourier transforms, with energy concentrated in the main lobe. Some suitable window functions include the Hamming window [49],

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi}{N-1} n\right) \quad 0 \leqslant n \leqslant N-1,$$

which has 99.96 percent of its energy in its main lobe, and a family of optimum windows proposed by Kaiser [43]. The Kaiser window provides, by a parameter adjustment, for a tradeoff between peak sidelobe ripple and main-lobe width.

5.3.2 *Frequency-Sampling Technique*

The frequency-sampling technique [29] uses the structure of comb filters cascaded with a parallel bank of complex resonators, which was discussed in section 4. The desired continuous-frequency-response characteristic is sampled at N equispaced frequencies, where N is the number of samples in the filter impulse response. These samples are set equal to the coefficients of the Fourier transform of the filter impulse response and are used in forming the weighting factors in the filter realization.

5.3.3 *Computer Optimization Methods*

Various techniques have been developed that synthesize optimized nonrecursive filters of equiripple frequency characteristics. As proposed by Herrmann and Schuessler [50], a set of nonlinear equations can be found in which the unknown quantities are the unit-sample response coefficients and the frequencies at which extrema of the approximation error occur. The system of equations is formed from constraints on the equiripple frequency characteristic. Hofstetter, Oppenheim, and Siegel [52] present a design algorithm to produce equiripple designs by use of the Lagrange interpolation formula to obtain a polynomial that goes through the allowable ripple values at the frequencies of the extrema of preassigned value. Optimized designs of frequency-sampling filters can be obtained by an algorithmic iterative optimization procedure such as developed by Rabiner, Gold, and McGonegal [33].

Helms [51] discusses the determination of coefficients for digital filters with equiripple or minimax error. The simplex method of linear programming is used to determine the digital filter coefficients that minimize the maximum error in the complex response

for nonrecursive filters. By using integer techniques, a nonrecursive digital filter design with quantized coefficients can be obtained.

5.4 Spectral Transformations

A general transformation for the translation of a lowpass digital filter, designed by some available technique, to a highpass, bandpass, band-elimination, or other lowpass filter is developed by Constantinides [53]. This transformation development is based on the rotation of the frequency characteristic as represented on a cylinder normal to the unit circle. The transformation is easily implemented by the mapping $z^{-1} \rightarrow g(z^{-1})$ in a lowpass-digital-filter system function, with the function $g(z^{-1})$ being given in Table 5.1. Thus lowpass, highpass, bandpass, and band-elimination filters can each be synthesized by starting with a lowpass-digital-filter prototype.

5.5 Time-Domain Design Techniques

Historically the design methods for digital filters have generally been limited to frequency-domain techniques. These techniques mostly involve determination of system-function coefficients based on frequency-response characteristics of continuous-time filter theory. Iterative optimization techniques leading to coefficients for approximations of arbitrary frequency-domain specifications have also been developed. The development of design techniques in the time domain, which has been somewhat limited, involves the determination of a system function $G(z)$ to produce a unit-sample response $g(n)$ as some form of best approximation of a desired unit-sample response $h(n)$. An exact trivial solution can always be found in the case of FIR filters, which will not be discussed here any further.

For IIR filters the time-domain design problem involves finding all a_i and b_i such that the system function $G(z)$ of the design filter is some "best" approximation to the z transform of $h(n)$, that is,

$$\frac{\sum_{i=0}^{M} b_i z^{-i}}{1 + \sum_{i=1}^{N} a_i z^{-i}} \approx \sum_{i=0}^{K-1} h_i z^{-i}.$$

The solution of this problem is considered by Burrus and Parks [54]. The most general form of their solution involves the determination of the filter coefficients from the matrix equation

$$\mathbf{h} + \mathbf{e} = \mathbf{A}^{-1} \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix},$$

where $\mathbf{h}$ is the $K \times 1$ vector of desired unit-sample response coefficients, $\mathbf{e}$ is the $K \times 1$ vector of errors between the design and target unit sample response $(g_i - h_i)$, $\mathbf{A}$ is the $K \times K$ lower triangular matrix,

Table 5.1
Spectral Transformations from a Lowpass-Digital-Filter Prototype.
(From Ref. 53 by permission.)

Filter Type	Transformation	Associated Design Formulas	Comments
Lowpass	$\dfrac{z^{-1} - \alpha}{1 - \alpha z^{-1}}$	$\alpha = \dfrac{\sin\left(\dfrac{\beta - \omega_c}{2}\right)T}{\sin\left(\dfrac{\beta + \omega_c}{2}\right)T}$	β is the cutoff frequency of the prototype filter
Highpass	$-\left(\dfrac{z^{-1} + \alpha}{1 + \alpha z^{-1}}\right)$	$\alpha = -\dfrac{\cos\left(\dfrac{\beta - \omega_c}{2}\right)T}{\cos\left(\dfrac{\beta + \omega_c}{2}\right)T}$	ω_c is the cutoff frequency of the design filter
Bandpass	$-\left(\dfrac{z^{-2} - \dfrac{2\alpha k}{k+1} z^{-1} + \dfrac{k-1}{k+1}}{\dfrac{k-1}{k+1} z^{-2} - \dfrac{2\alpha k}{k+1} z^{-1} + 1}\right)$	$\alpha = \dfrac{\cos\left(\dfrac{\omega_2 + \omega_1}{2}\right)T}{\cos\left(\dfrac{\omega_2 - \omega_1}{2}\right)T}$ $k = \cot\left(\dfrac{\omega_2 - \omega_1}{2}\right)T \tan\dfrac{\beta T}{2}$	ω_2 and ω_1 are the upper and lower cutoff frequencies of the design filter
Band elimination	$\left(\dfrac{z^{-2} - \dfrac{2\alpha}{1+k} z^{-1} + \dfrac{1-k}{1+k}}{\dfrac{1-k}{1+k} z^{-2} - \dfrac{2\alpha}{1+k} z^{-1} + 1}\right)$	$\alpha = \dfrac{\cos\left(\dfrac{\omega_2 + \omega_1}{2}\right)T}{\cos\left(\dfrac{\omega_2 - \omega_1}{2}\right)T}$ $k = \tan\left(\dfrac{\omega_2 - \omega_1}{2}\right)T \tan\dfrac{\beta T}{2}$	

$$\mathbf{A} = \begin{bmatrix} a_0 & 0 & 0 & \cdots & 0 \\ a_i & a_0 & 0 & \cdots & 0 \\ & & \cdots & & \\ a_n & & \cdots & & 0 \\ 0 & & \cdots & & 0 \\ 0 & & \cdots & & a_0 \end{bmatrix},$$

b is the $(M + 1) \times 1$ vector of b_i coefficients, and **0** is the $(K - M - 1) \times 1$ zero vector. Solutions that provide a $g(n)$ that is exactly equal to $h(n)$ are possible under certain conditions, such as when $K = M + N + 1$. For exact solutions, **e** will be zero. Solutions under various conditions of approximation can be obtained. These conditions include an exact equivalence of $g(n)$ and $h(n)$ at certain sample points such as the first $M + 1$ or any $M + 1$. A $g(n)$ to provide minimization of some function of **e** can also be obtained.

There is a unit-sample response that corresponds to a particular frequency response and phase characteristic of a digital filter. Thus a digital filter with arbitrary frequency and phase characteristics can be obtained by a time-domain design that approximates, in some form, a unit-sample response corresponding to the desired characteristics. Time-domain designs can deal only with an overall frequency specification including both amplitude and phase. Such a design technique is proposed by Brophy and Salazar [55]. Due to the nonlinear programming and related problems involved in the determination of filter coefficients for a frequency-domain design of a desired IIR digital filter, it is suggested that a time-domain design may be more natural. Because of the advantages in determining initial values for the a_i and b_i in the time domain, it might be advantageous to first implement a time-domain procedure followed by a frequency-domain procedure resulting in a desired overall frequency characteristic. Approximation techniques similar to those of Ref. 54 are discussed in Ref. 55; in addition, results of various examples applying error minimization are presented.

A time-domain design technique proposed in Ref. 55 requires the determination of a target time sequence. The approximating filter is then forced by one of several techniques to have its unit-sample response approximate (in a least-squares sense) the target sequence. Iterative routines are then employed to find values of the a_i and b_i coefficients such that a locally optimum solution is obtained.

6. THE DISCRETE FOURIER TRANSFORM

6.1 Relation to the Continuous Fourier Transform

The continuous form of the Fourier transform was presented earlier. The analysis and synthesis forms of the transform, in relationship to the problems of system frequency response and design, are a mode of transformation between the time and frequency domain. To be able to use this powerful transform in conjunction with the computation advantages of the electronic digital computer requires a form of the Fourier transform that operates with a finite set of discrete data. The discrete Fourier transform (DFT) is the desired form, which can be derived analytically by taking the continuous Fourier transform of a periodic, truncated, and discrete representation of the original continuous function [20]. The resulting DFT is

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad 0 \leqslant k \leqslant N - 1,$$

where $W_N = e^{-j(2\pi/N)}$.

The development of the DFT can be seen from Fig. 6.1. At each step a time function and corresponding frequency domain representation are presented, with multiplication in one domain corresponding to convolution in the other. The time function of Fig. 6.1a is sampled by the pulse train $\Delta_0(t)$ of Fig. 6.1b at intervals of T. This produces the sampled version of Fig. 6.1c, whose periodic spectrum will in general be distorted unless band-limiting and sample-rate constraints are satisfied. The rectangular window (Fig. 6.1d) truncates the discrete representation of the signal to a finite set of N samples. Due to the Gibbs phenomenon the finite duration of the window causes the

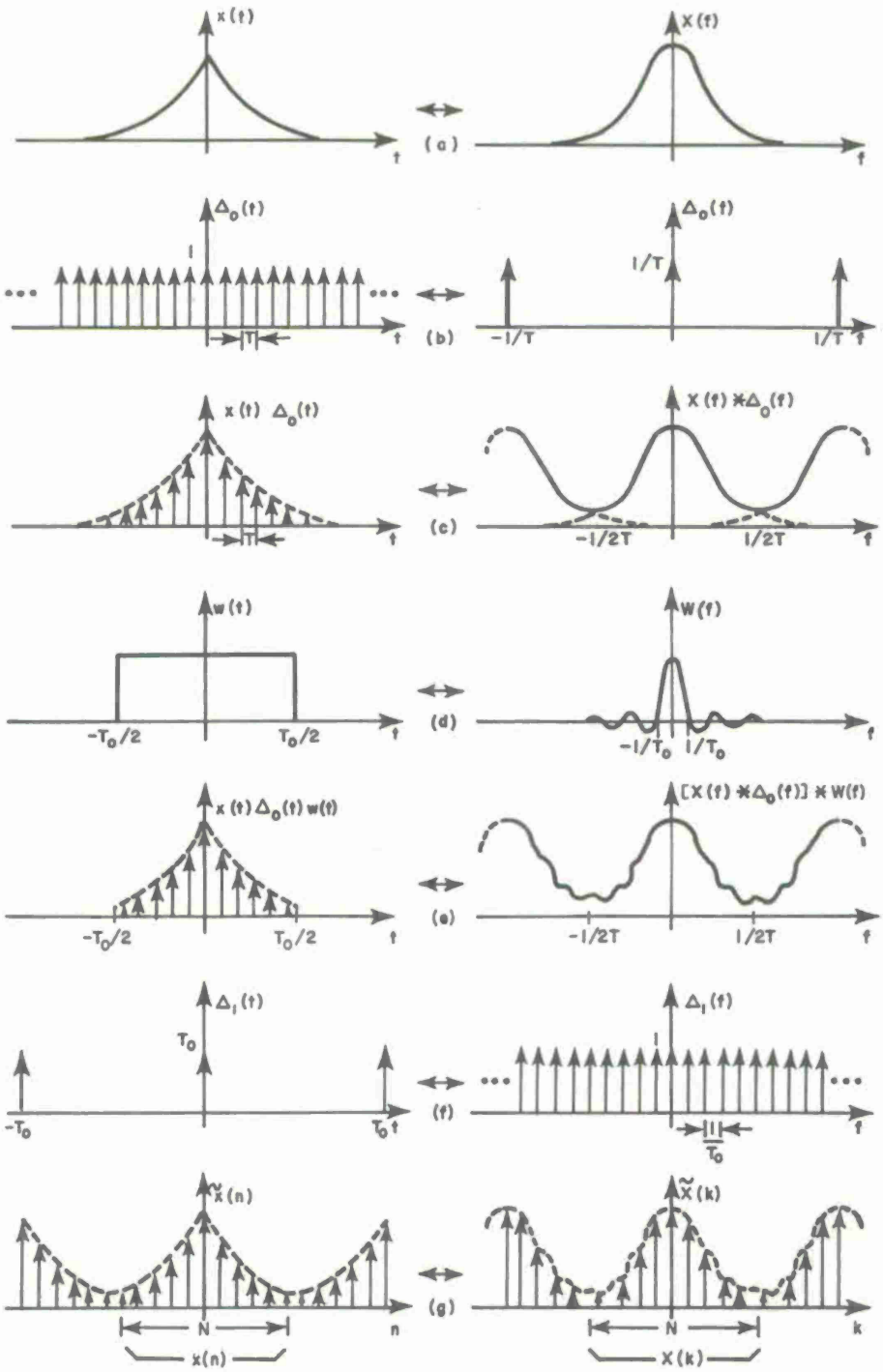


Fig. 6.1—The development of the discrete Fourier transform.
(From Ref. 20 by permission.)

spectrum of the finite discrete representation of the signal to be distorted (Fig. 6.1e). To represent the signal spectrum in discrete form it must be sampled by the pulse train $\Delta_1(f)$, at frequency sample interval $1/T_0$. Convolving the time-domain representation $\Delta_1(t)$ of the frequency sampling function $\Delta_1(f)$ with the discrete representation of the signal results in a periodic function $\tilde{x}(n)$ of the N time samples (Fig. 6.1g) with corresponding periodic DFT $\tilde{X}(k)$. The DFT relationships operate with the representation of the signal $x(n)$ and spectrum $X(k)$ limited to the set of N distinct samples as indicated.

An alternative interpretation of the DFT is related to the representation of the frequency response of a discrete system as the values of the z transform on the unit circle. Thus the DFT is a sequence of samples equally spaced in angle, on the unit circle, of the z transform [11]. Since $x(n)$ is interpreted as equal to 0 outside the range $0 \leqslant n \leqslant N-1$, then

$$X(z) = \sum_{n=0}^{N-1} x(n) z^{-n}.$$

With $z = e^{j(2\pi/N)k} = W_N^{-k}$ indicating the kth sample on the unit circle, the DFT relation results.

6.2 Inverse Discrete Fourier Transform

The DFT is an invertible transform. Thus an original sequence can be recovered by means of the inverse discrete Fourier transform (IDFT):

$$x(n) = \frac{1}{N}\left[\sum_{k=0}^{N-1} X(k) W_N^{-kn}\right], \quad 0 \leqslant n \leqslant N-1.$$

This can be shown by inserting the DFT relation into the relation for the IDFT [56]:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1}\left[\sum_{r=0}^{N-1} x(r) W_N^{kr}\right] W_n^{-kn}.$$

Reversing the order of the summations,

$$x(n) = \frac{1}{N}\sum_{r=0}^{N-1} x(r) \sum_{k=0}^{N-1} W_N^{k(r-n)}.$$

The right side of this equation is $x(n)$ by the orthogonality relationship:

$$\sum_{k=0}^{N-1} W_N^{k(r-n)} = N \text{ if } r = n$$

$$= 0 \text{ otherwise.}$$

The expressions for the DFT and IDFT differ only in the sign of the exponent of W_N and the scale factor of $1/N$ present in the expression for $X(k)$. Thus the form of the IDFT can be expressed as

$$x(n) = \frac{1}{N}\left[\sum_{n=0}^{N-1} X^*(k) W_N^{kn}\right]^*$$

where the superscript asterisk indicates complex conjugation [57]. Thus any computation algorithm applicable to the DFT can be used to compute the IDFT.

6.3 Properties of the DFT

A knowledge and understanding of the fundamental properties of the DFT is important for its proper and efficient use. This is particularly true when applying an extremely efficient algorithmic form of the DFT to be discussed later. Several of the important properties will be discussed here, and the presentation of elementary properties and their proofs in the literature will be cited.

6.3.1 *Periodicity*

The function W_N^{kn} is periodic of period N; therefore

$$W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n}.$$

Thus the DFT, $X(k)$, and its IDFT, $x(n)$, are also periodic of period N with the relationships

$$x(n) = x(iN + n), \quad i = 0, \pm 1, \pm 2, \ldots,$$

$$X(k) = X(iN + k), \quad i = 0, \pm 1, \pm 2, \ldots.$$

$$x(-n) = x(N - n)$$

$$X(-k) = X(N - k).$$

In light of the above relationships, the DFT and IDFT can be considered to be defined on a circle of circumference N at discrete points $0, 1, 2, \ldots, N - 1$. Thus the transforms are uniquely defined for a single traversal of the circumference, with the periodicity relations applicable to multiple as well as clockwise and counterclockwise traversals.

6.3.2 *Other Useful Properties*

With some alteration for discrete representation, many of the properties of the continuous Fourier transform are applicable in the case of the DFT. These properties are presented and proved in the literature with respect to linearity, symmetry, even and odd

sequences, complex-conjugate sequences [58,59], sine and cosine transforms [60], etc. One of the most useful of these properties is the linearity relation that the DFT of a complex weighted sum of sequences is the identically weighted sum of the DFT's of the sequences. That is, if the DFT of sequences $x(n)$ and $y(n)$ are $X(k)$ and $Y(k)$ respectively, as denoted by

$$x(n) \leftrightarrow X(k)$$

and

$$y(n) \leftrightarrow Y(k),$$

and a and b are some complex numbers, then

$$ax(n) + by(n) \leftrightarrow aX(k) + bY(k).$$

Another important property involves the relationship between a sequence and its DFT when they are circularly shifted along their time (n) or frequency (k) axis. Such a shift in the time sequence causes a phase change in the DFT of the sequence, and a shift in the frequency axis results in a phase change in the corresponding time sequence. This is denoted by

$$x(n - m) \leftrightarrow W_N^{km} X(k)$$

and

$$W_N^{-in} x(n) \leftrightarrow X(k - i).$$

The elementary properties of the DFT have been used for ease of evaluation of the DFT. For example it can be shown [58] that the DFT of two real functions can be performed simultaneously. Let $x_1(n)$ and $x_2(n)$ be real with $x_1(n) \leftrightarrow X_1(k)$ and $x_2(n) \leftrightarrow X_2(k)$. Forming $x(n) = x_1(n) + jx_2(n)$ with $x(n) \leftrightarrow X(k)$ and applying the linearity property, $X(k) = X_1(k) + jX_2(k)$. The desired DFT's are then

$$X_1(k) = \frac{X(k) + X^*(N - k)}{2}$$

and

$$X_2(k) = \frac{X(k) - X^*(N - k)}{2j}.$$

Similar procedures can lead to the evaluation of a $2N$-point DFT of real data by means of an N-point DFT [60].

Another important property of the DFT which will be discussed later is that the circular convolution of two sequences is the IDFT of the product of the DFT's of each of the sequences.

6.4 Computation of the DFT

Considering the DFT as presented, the direct application of the expression can be represented as a set of N equations of the form

$$X(k) = x(0)W_N^{k\cdot 0} + x(1)W_N^{k\cdot 1} + x(2)W_N^{k\cdot 2} + \cdots + x(N-1)W_N^{k\cdot(N-1)},$$

where $k = 0, 1, 2, \ldots, N-1$. As defined earlier, $W_N = e^{-j(2\pi/N)}$ which is of course complex, and each of the weighting factors of the general form W_N^{kn} can be represented in terms of its real and imaginary components as $\cos(2\pi/N)kn - j \sin(2\pi/N)kn$. Since any term $x(n)$ will in general be complex, each of the N terms of the above equation for $X(k)$ involves a multiplication of two complex factors and evaluation of the sum involves $N-1$ complex additions. Evaluation of the DFT involves N such equations and thus requires N^2 complex multiplications and $N(N-1)$ complex additions. The efficiency of computer operations involves the processing time, storage requirements, and number of accesses to that storage. The number of machine arithmetic operations is then a measure of efficiency of a computation procedure. Direct DFT computation requires $4N^2$ real multiplications and $N(4N-2)$ real additions [11]. The amount of computation time and the related complexity is thus proportional to N^2, which is quite large for large values of N. In the literature an operation with respect to the DFT refers to a complex multiplication and addition. Thus direct DFT computation requires approximately N^2 operations. The desire for application of the DFT as a computational tool for analysis in a wide range of scientific endeavors has greatly increased. With typical requirements for DFT's of $N = 2^{10}$ and greater, the number of real arithmetic operations required are such that the computation cost restricts the full potential of DFT applications. An extremely efficient algorithm that overcomes this restriction is discussed in the following section.

7. THE FAST FOURIER TRANSFORM

The desire to use the DFT in a wide range of applications, even prior to the computer age, led to techniques that reduced the number of required arithmetic operations. An interesting history of the development of such techniques is presented in Ref. 61. These techniques are based on the computational economy derived from the symmetry and periodicity of the sine and cosine functions. The development of these techniques dates back to 1903 in the work of Runge [62]. Danielson and Lanczos [63] generalized Runge's work in 1942 to an efficient computation scheme for N equal to an integer power of 2. Another line of development, based on analysis and design of experiments, including the work of Yates [64] and Good [65], led to other efficient techniques.

The most generalized of efficient DFT computation techniques was disclosed by Cooley and Tukey [9] in 1965. When N is an integer power of 2, the Cooley-Tukey method is similar to earlier methods. The Cooley-Tukey method is more general, however, since it can be used when N is not an integer power of 2 but is a highly composite number. Therefore, if N has m factors, such as $N = n_1 n_2 \ldots n_m$, a number of operations proportional to $N(n_1 + n_2 + \ldots + n_m)$ are required as opposed to N^2 operations for direct evaluation. In general, if $N > 4$, then $n_1 + n_2 + \ldots + n_m < N$.

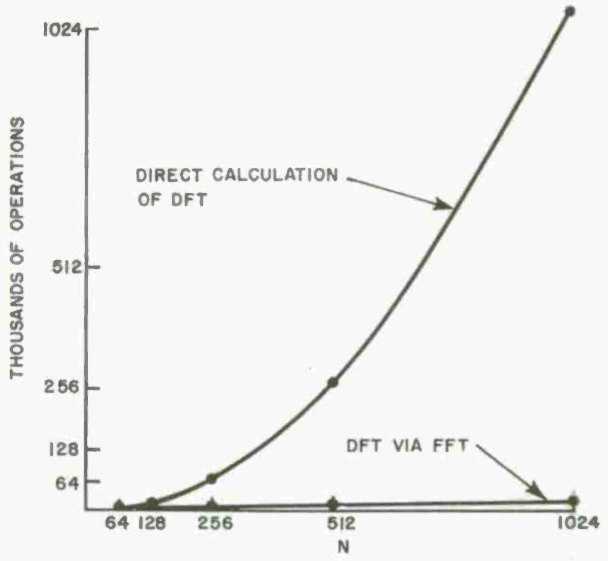


Fig. 7.1—Comparison of the number of operations versus N for direct and FFT computations of the DFT

The disclosure of a highly efficient algorithm for the computation of the DFT [9] led to the development of several other algorithms with similar savings in computation. The class of such algorithms is collectively known as the FFT or fast Fourier transform. The computational efficiency of the FFT is demonstrated in Fig. 7.1, which is a plot of number of operations versus N for the direct computation of the DFT and for the FFT computation [57]. This high efficiency is derived by breaking down an N-point DFT into a set of smaller transforms. There are various forms of the FFT algorithms as well as alternate derivations of each. Several of these forms will be discussed here.

7.1 Algorithms for $N = 2^m$

7.1.1 *Decimation in Time*

The original Cooley-Tukey form of the FFT corresponds to the decimation-in-time algorithm. A modified form attributed to Sande [59] is decimation in frequency, otherwise known as the Sande-Tukey method. The derivation of the two forms is presented in Refs. 59, and 56, with somewhat more mathematical detail in Ref. 59.

If a sequence $x(n)$ is considered to be composed of N samples, or points, of two sequences $x(2n)$ and $x(2n + 1)$, $n = 0, 1, 2, \ldots, (N/2) - 1$, that is sequences of $N/2$ even points and of $N/2$ odd points, then the DFT of $x(n)$ can be represented in terms of two DFT's of $N/2$ points each. Thus [56,11] the DFT of $x(n)$ is

$$X(k) = \sum_{n=0}^{(N/2)-1} \left[x(2n) W_N^{2nk} + x(2n+1) W_N^{(2n+1)k} \right], \quad k = 0, 1, 2, \ldots, N-1$$

or, since $W_N^2 = W_{N/2}$,

$$X(k) = \sum_{n=0}^{(N/2)-1} x(2n) W_{N/2}^{nk} + W_N^k \sum_{n=0}^{(N/2)-1} x(2n+1) W_{N/2}^{nk}, \quad k = 0, 1, 2, \ldots, N-1$$

If the DFT's of $x(2n)$ and $x(2n+1)$ are $A(k)$ and $B(k)$, where $k = 0, 1, 2, \ldots, (N/2)-1$, then

$$X(k) = A(k) + W_N^k B(k), \quad k = 0, 1, 2, \ldots, \frac{N}{2} - 1$$

Since the DFT is periodic, as was discussed earlier, the values of $A(k)$ and $B(k)$ for $k < N/2$ repeat for $k > N/2$. Therefore

$$X\left(k + \frac{N}{2}\right) = A(k) + W_N^{k+(N/2)} B(k), \quad k = 0, 1, 2, \ldots, \frac{N}{2} - 1,$$

and, since $W_N^{N/2} = e^{-j(2\pi/N)(N/2)} = -1$,

$$X\left(k + \frac{N}{2}\right) = A(k) - W_N^k B(k), \quad k = 0, 1, 2, \ldots, \frac{N}{2} - 1.$$

Thus the DFT of a sequence $x(n)$ of N samples can be found from the DFT's of two sequences of $N/2$ samples each. The decimation-in-time algorithm performs DFT's on smaller and smaller subsequences of the input sequence.

To illustrate the application of the preceding results, the signal-flow-graph representation will be used. In Fig. 7.2 two sequences of even and odd samples are each presented to an $N/2$-point DFT. The results of these DFT's are combined as indicated in Fig. 7.2 by equations for $X(k)$ and $X[k + (N/2)]$. This reduction process can be continued by next presenting four sequences, each of every fourth sample, to four $N/4$-point DFT's and combining as above. If $N = 2^m$, then m reductions can be made until N one-point DFT's are required. The signal flow graph of the decimation-in-time algorithm results, as indicated for $N = 8$ in Fig. 7.3. Each vertical column of nodes represents an iteration of the algorithm, there being a total of $m = \log_2 N$ iterations required for $N = 2^m$.

Several factors are to be noted in this development. Those branches with unity transmittance as well as those with $W_N^k = \pm 1$ require no multiplications. The final form of the graph requires that the input order be scrambled. This order corresponds to normal sequential order with the ordering argument represented in binary form with normal binary weighting order reversed. This is often referred to as *bit reversed order.* From the signal flow graph it can be seen that, for this particular ordering of $x(n)$, at

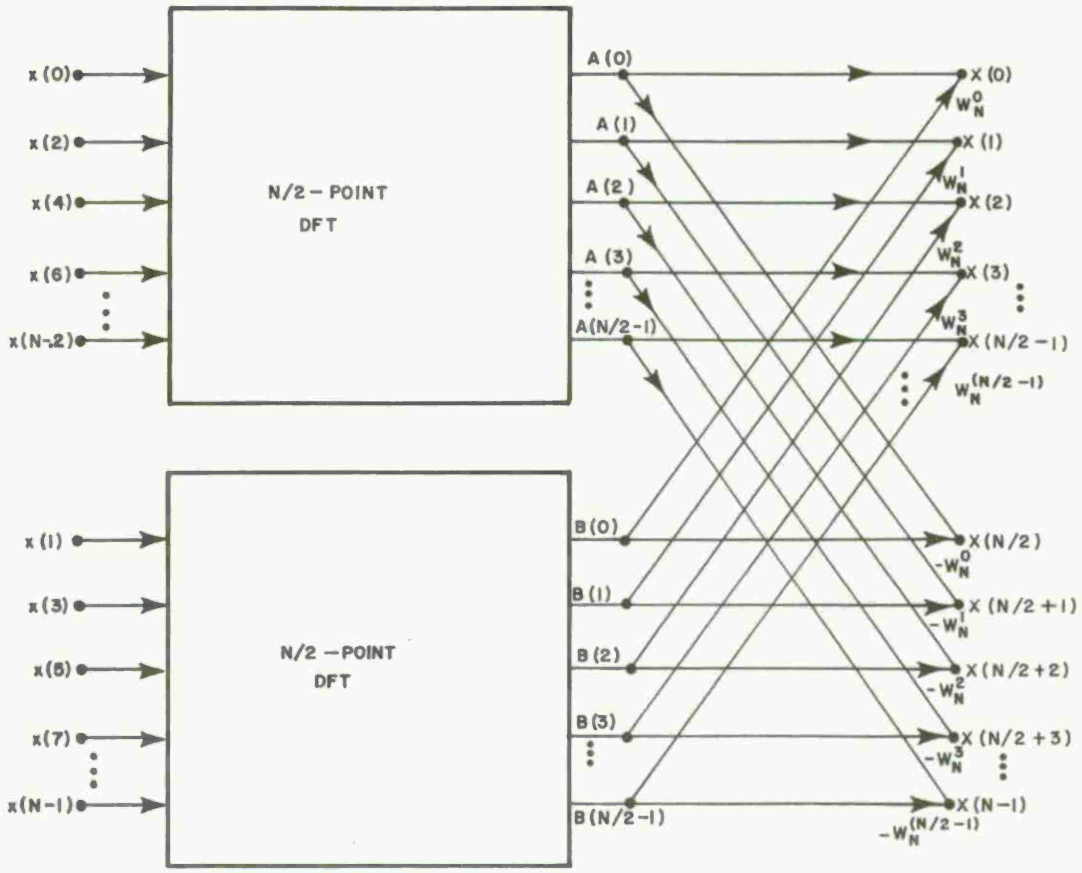


Fig. 7.2—Signal flow graph illustrating reduction of an N-point DFT to two $N/2$-point DFT's using the decimation-in-time algorithm

each iteration of the transform a pair of nodes in the ith iteration affects only the corresponding pair in the $(i + 1)$th iteration. The interval between those nodes increases by a factor of 2 in each iteration. This distinct pairing of nodes between iterations permits the algorithm to be essentially computed "in place," with the results from each iteration replacing or being written in memory over the results of the previous iteration. Thus the algorithm requires the implementation of a basic computation pair of the form

$$X_{i+1}(p) = X_i(p) + W_N^r X_i(q)$$

and

$$X_{i+1}(q) = X_i(p) - W_N^r X_i(q)$$

for node pair (p, q) in the $(i + 1)$th iteration as computed from the corresponding node pair from the ith iteration [11]. Multiplication by the weighting factor need be performed only once for each pair and used in the sum and difference relation. With such reductions in computations for $N = 2^m$, the resulting algorithm requires a number of operations proportional to $N \log_2 N$.

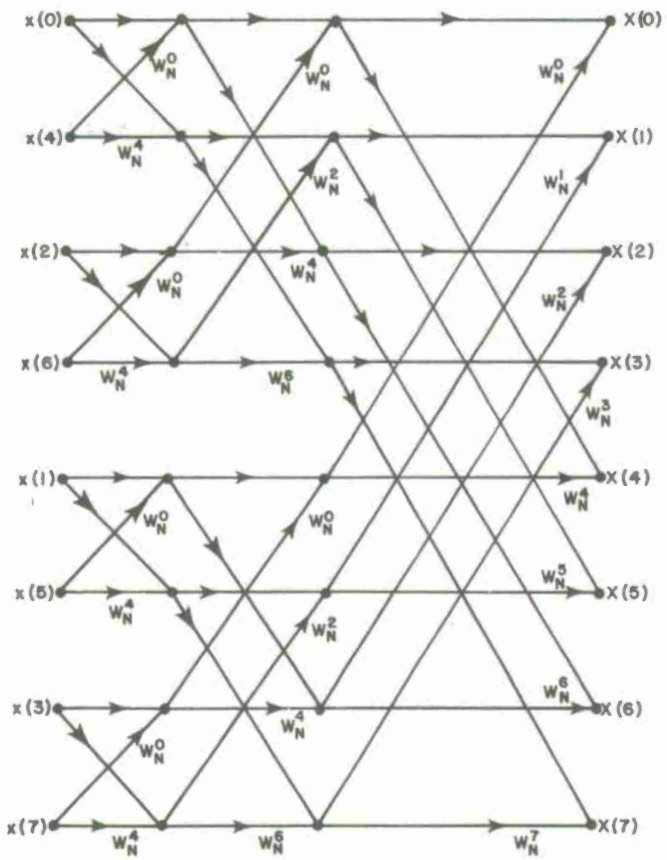


Fig. 7.3—Completely reduced signal flow graph for the decimation-in-time algorithm (for $N = 8$)

The signal flow graph for decimation in time can be modified by properly interchanging the sets of horizontally adjacent nodes to provide for transformation with inputs in naturally ordered sequence as shown for $N = 8$, in Fig. 7.4. In this case the spectral samples or outputs are in scrambled order, and the exponents of W_N, which can be computed or stored in memory, are used in natural order. This natural-ordered input form of the decimation-in-time algorithm corresponds to the original form of the Cooley-Tukey algorithm [9]. Another form of the decimation-in-time algorithm which provides for inputs and outputs in normal-ordered form [56] is attributed to Stockham. In this case exponents are used in normal order but computation can no longer be performed in place.

7.1.2 *Decimation in Frequency*

The decimation-in-frequency algorithm reverses the roles of $x(n)$ and $X(k)$ and thus accomplishes computation efficiency by performing DFT's for smaller and smaller subsequences of $X(k)$. Let a sequence $x(n)$ be composed of N samples of two shorter

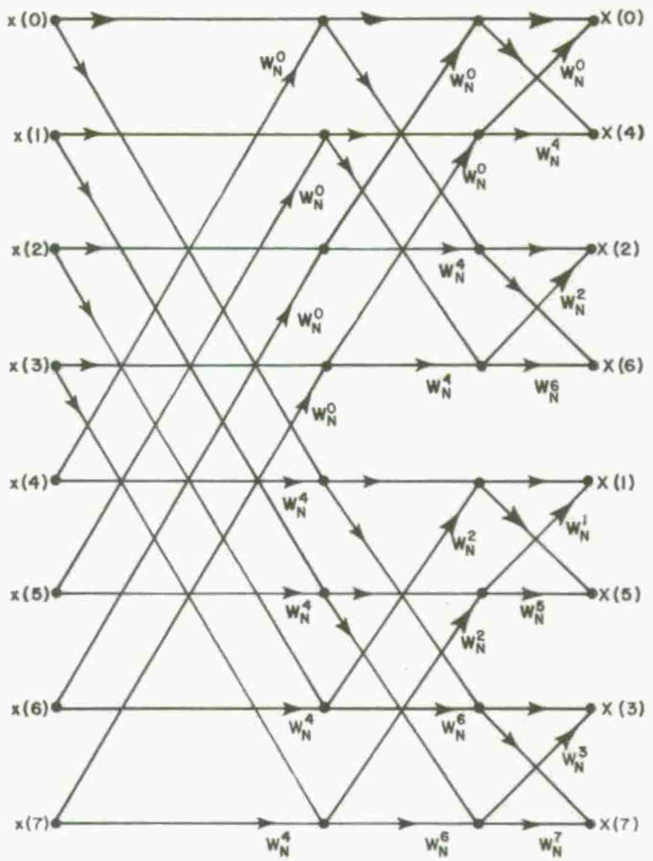


Fig. 7.4—Rearrangement of the signal flow graph of Fig. 7.3 for computation with naturally ordered time samples

sequences of $x(n)$ and $x[n + (N/2)]$, $n = 0, 1, 2, \ldots, (N/2) - 1$, of the first and last $N/2$ samples each. Thus [56,11]

$$X(k) = \sum_{n=0}^{(N/2)-1} \left\{ x(n) W_N^{nk} + x\left(n + \frac{N}{2}\right) W_N^{[n+(N/2)]k} \right\}$$

$$= \sum_{n=0}^{(N/2)-1} \left[x(n) + W_N^{(N/2)k}\, x\left(n + \frac{N}{2}\right) \right] W_N^{nk}.$$

when the frequency sequence is decimated, $X(2k)$ and $X(2k + 1)$, $k = 0, 1, 2, \ldots, (N/2) - 1$, consisting of the even and odd frequency points, are formed. From the preceding equation for $X(k)$, since $(W_N^{N/2})^{2k} = 1$,

$$X(2k) = \sum_{n=0}^{(N/2)-1} \left[x(n) + x\left(n + \frac{N}{2}\right) \right] W_{N/2}^{nk}$$

and

$$X(2k + 1) = \sum_{n=0}^{(N/2)-1} \left[x(n) + W_N^{N/2} x\left(n + \frac{N}{2}\right) \right] W_N^{n(2k+1)}$$

$$= \sum_{n=0}^{(N/2)-1} \left[x(n) - x\left(n + \frac{N}{2}\right) \right] W_N^n W_{N/2}^{nk} .$$

Thus $X(2k)$ is the $N/2$-point DFT of the sum of the first and last $N/2$ time samples, and $X(2k + 1)$ is the $N/2$-point DFT of the difference between the first and last $N/2$ time samples multiplied by W_N^n. The signal flow graph of Fig. 7.5 shows the reduction of the N-point DFT to two $N/2$-point DFT's of the functions as described in the above equations. Each $N/2$-point DFT can be replaced by two $N/4$-point DFT's, then by four $N/8$-point DFT's, and so on. The completely reduced form for $N = 8$ is shown in Fig. 7.6 and like the decimation-in-time algorithm requires a number of operations proportional to $N \log_2 N$. For this form of the decimation-in-frequency algorithm the time samples and exponents of W_N are used in natural order, producing frequency samples in bit reversed order. Because the computation at a pair of nodes in an iteration depends on only a unique pair of nodes in the previous iteration, computation can be performed in place. From Ref. 11 the basic computation pair of the decimation in frequency algorithm is

$$X_{i+1}(p) = X_i(p) + X_i(q)$$

and

$$X_{i+1}(q) = [X_i(p) - X_i(q)] W_N^r .$$

It is also shown in Ref. 11 that the form of these equations can be derived from the computation pair for decimation in time and that there is a transpose relation between decimation-in-frequency and decimation-in-time signal flow graphs.

The decimation-in-frequency signal flow graph can be rearranged to provide naturally ordered frequency samples but with time samples and coefficients used in bit reversed order. A form with time and frequency samples in natural order can be found, but computation is not in place.

7.1.3 *Other Formulations*

A matrix development of the FFT algorithm when N is a power of 2 is presented by Brigham and Morrow [66] and gives a different view of the efficiency in the FFT computation. The DFT can be represented as a matrix relationship

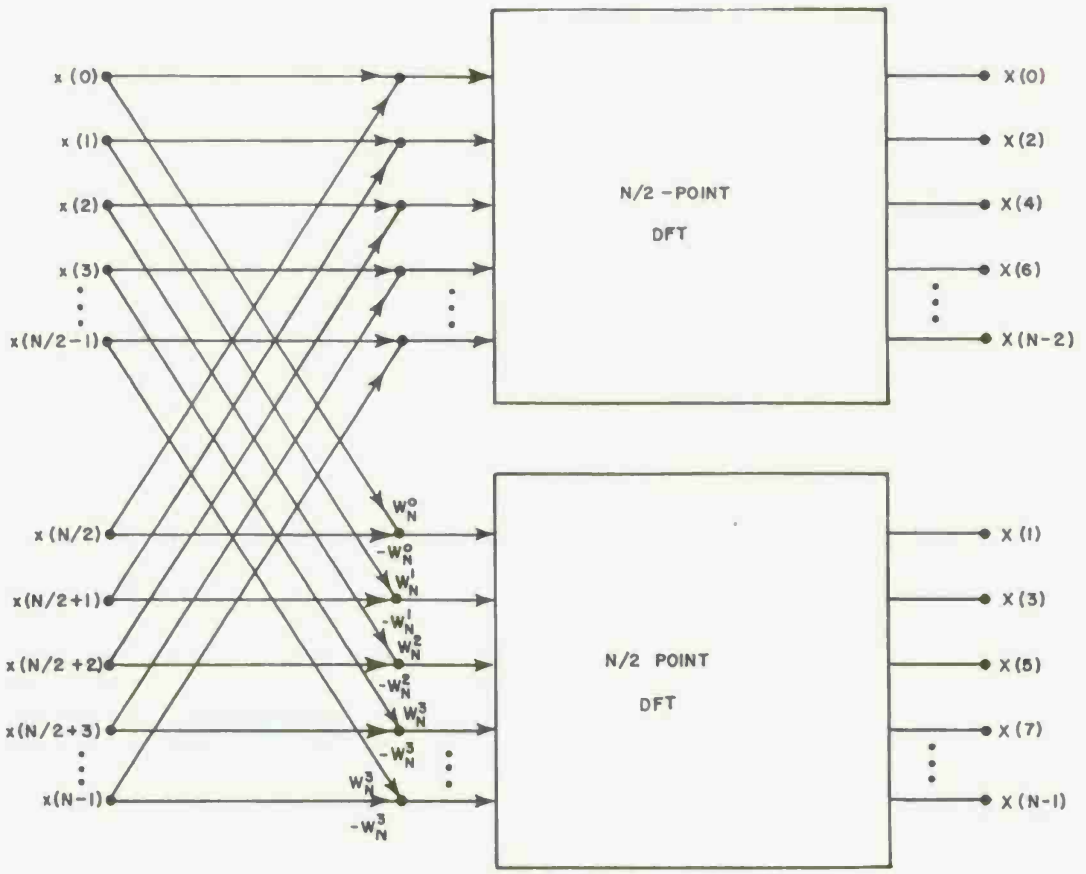


Fig. 7.5—Signal flow graph illustrating reduction of an N-point DFT to two $N/2$-point DFT's using the decimation-in-frequency algorithm

$$[X(k)] = [W_N^{nk}]\,[x(n)]\,.$$

Using matrix factoring, interchanging of rows, and the unity value of many of the terms in $[W_N^{nk}]$, a computationally more efficient form of the DFT results. For $N = 4$ the matrix equations are

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W_N^0 & 0 & 0 \\ 1 & W_N^2 & 0 & 0 \\ 0 & 0 & 1 & W_N^1 \\ 0 & 0 & 1 & W_N^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W_N^0 & 0 \\ 0 & 1 & 0 & W_N^0 \\ 1 & 0 & W_N^2 & 0 \\ 0 & 1 & 0 & W_N^2 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix},$$

which can be shown to require $Nm/2 = 4$ complex multiplications and $Nm = 8$ complex additions compared to $N^2 = 16$ complex multiplications and additions.

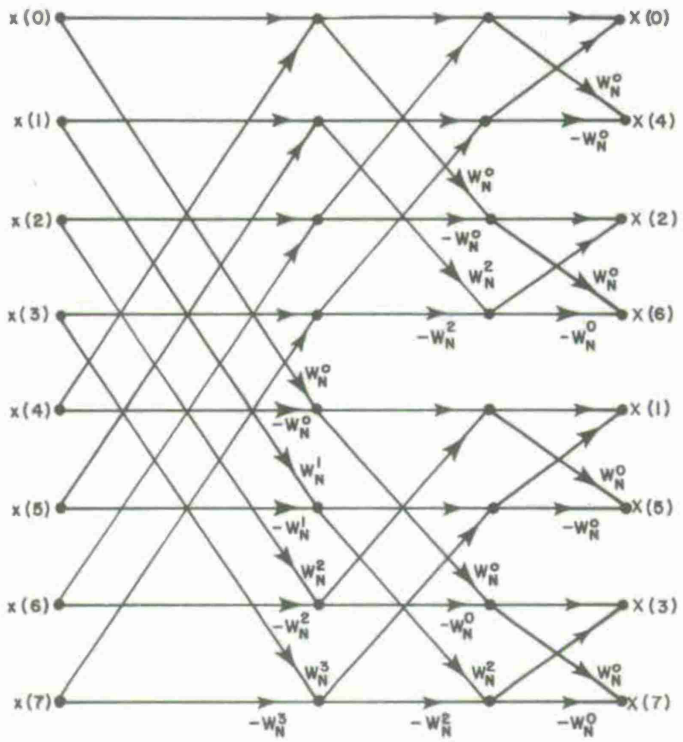


Fig. 7.6—Completely reduced signal flow graph for the decimation-in-frequency algorithm (for $N = 8$)

A theoretical formulation of the FFT algorithm is presented by Cooley and Tukey [9]. This involves representation of the DFT with k and n expressed in binary form with respect to summation representation and exponent representation. The efficiency in computation derives from simplifications due to periodicity in the powers of W_N following separation of the components of n for decimation in time or of k for decimation in frequency. Thus, if k and n are represented in binary form as

$$k = k_{m-1}2^{m-1} + \cdots + k_1 2 + k_0$$

and

$$n = n_{m-1}2^{m-1} + \cdots + n_1 2 + n_0$$

with k_i and n_i being binary components of value 0 or 1, the DFT can be expressed as

$$X(k_{m-1}, \ldots, k_0) = \sum_{n_0} \sum_{n_1} \cdots \sum_{n_{m-1}} x(n_{m-1}, \ldots, n_0) W^{k(n_{m-1}2^{m-1} + \cdots + n_0)}$$

This sum can be computed as a series of m successive arrays $x_1, x_2, \ldots, x_m$ based on the initial time samples in array x. Using simplifications due to the periodicity of powers of W_N and separating components of n for decimation in time, the resulting arrays are

$$x_i(k_0, \ldots, k_{i-1}, n_{m-i-1}, \ldots, n_0)$$

$$= \sum_{n_{m-i}=0}^{1} x_{i-1}(k_0, \ldots, k_{i-2}, n_{m-i}, \ldots, n_0) W_N^{(k_{i-1}2^{i-1}+\cdots+k_0)n_{m-i}2^{m-i}},$$

$i = 1, 2, \ldots, m$, where x_i is the array resulting from the ith iteration, the mth array being the resulting DFT sums. For decimation in frequency the components of k are separated [20]. Also, the algorithm can be derived for a base 4 by expressing k and n in quaternary form.

7.2 Techniques for Highly Composite N

The techniques discussed up to this point apply only when N is an integer power of 2. Obviously such a limitation would restrict the practical application of the algorithm. The great power of the Cooley-Tukey algorithm and other variations generated subsequent to their disclosure [9] is in its more general applicability when N is highly composite, that is, $N = r_1 r_2 \ldots r_m$. The methods of derivation described previously for decimation in time and decimation in frequency can be extended to this more general case of N being highly composite [56]. If N has a prime factor p, then in developing the decimation in time or frequency algorithm, p subsequences $X(pk + i)$ are formed, each having N/p-point DFT's. This procedure can be extended with further simplifications in the DFT's if N has other prime factors.

With respect to the theoretical FFT development discussed earlier, the successive arrays computed in each iteration of the algorithm can be modified for $N = r_1 r_2 \ldots r_m$ by expressing k and n in a mixed radix representation [67]:

$$k = k_{m-1}(r_1 r_2 \ldots r_{m-1}) + k_{m-2}(r_1 r_2 \ldots r_{m-2}) + \cdots + k_1 r_1 + k_0$$

and

$$n = n_{m-1}(r_2 r_3 \ldots r_m) + n_{m-2}(r_3 r_4 \ldots r_m) + \cdots + n_1 r_m + n_0,$$

where

$$k_{i-1} = 0, 1, 2, \ldots, r_i - 1, \quad 1 \leqslant i \leqslant m$$

and

$$n_i = 0, 1, 2, \ldots, r_{m-i} - 1, \quad 0 \leqslant i \leqslant m - 1.$$

The recursive equations for the successive iterations presented previously are accordingly modified such that they can be separated into a set of simpler transforms of $r_1, r_2, \ldots, r_m$ points. Thus the computation to obtain x_m requires a number of operations proportional to $N(r_1 + r_2 + \cdots + r_m)$.

7.3 Techniques for N a Prime Number

Up to this point, FFT techniques have been presented for N a power of 2 and N highly composite. Techniques have been developed to include the case of N a prime. The chirp z-transform (CZT) algorithm, due to Bluestein [68,69], permits the computation of the DFT to be performed by means of the FFT for any value of N including primes. In the expression for the DFT,

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk},$$

Bluestein [69] uses the substitution

$$nk = \frac{N^2 + n^2 + k^2 - (N + n - k)^2}{2}$$

or, since W_N to any integer multiple of N is unity,

$$nk = \frac{n^2 + k^2 - (k - n)^2}{2},$$

which results in

$$X(k) = W_N^{k^2/2} \sum_{n=0}^{N-1} x(n) W_N^{n^2/2} W_N^{-(k-n)^2/2}, \quad k = 0, 1, \ldots, N - 1.$$

From this expression it can be seen that if

$$y(n) = x(n) W_N^{n^2/2},$$

then

$$X(k) = W_N^{k^2/2} \left[y(n) * W_N^{-(n^2/2)} \right].$$

Thus the DFT is related to a discrete convolution. As will be shown in the next section, such a convolution can be computed by two DFT's, a multiplication of sequences, and an IDFT. Each of the three transforms can be performed using N'-point-FFT power-of-2 algorithms, where $N' = 2^m \geqslant 2N - 1$. Thus $X(k)$ can be computed in a number of operations essentially proportional to $N' \log_2 N'$.

The CZT is actually used more generally to evaluate the z transform of an N-point time sequence at any M equiangular points on a spiral contour of the z plane [70]. When the spiral contour is specifically the unit circle and $N = M$, the CZT is equivalent to the DFT. Details of the computation are provided in Ref. 70. A technique for the specific case of N a prime is disclosed by Rader [71]. This technique uses the properties of primitive roots to represent the DFT, when N is a prime, in terms of a discrete convolution which can be computed by power-of-2, or highly composite, FFT techniques.

8. DISCRETE CONVOLUTION AND CORRELATION

8.1 Relation to Convolution and Correlation Integrals

The convolution integral [19], generally expressed as

$$y(t) = \int_{-\infty}^{\infty} x(t)h(t - \tau)d\tau,$$

represents the response $y(t)$ of a system with impulse response $h(t)$ to an input stimulus $x(t)$ [72]. The value of the system output at time t is thus the area under the product of $x(t)$ and the mirror image of $h(\tau)$ about the $\tau = 0$ axis, shifted by t. For practical systems, which possess the properties of stability and causality, the integration limits will be $0 \leqslant \tau \leqslant T$, where the system response is 0 prior to $\tau = 0$ and 0, or negligible beyond $\tau = T$. The convolution integral can be thought of as representing the resultant response of a system to an input represented as a continuum of unit impulses with amplitude weighting $x(\tau)$. A similar line of reasoning, applied here earlier in the discrete case with the unit-sample sequence, led to the development of the discrete convolution or convolution sum,

$$y(n) = \sum_{m=0}^{N-1} x(m)h(n - m),$$

where the summation limits are compatible with application to practical systems. In general this relationship describes a linear discrete convolution. The nth value in the output sequence is determined as the sum of the products of each input sample, with the corresponding samples of the time-reversed representation of $h(m)$ shifted by n units of time. In both the convolution integral and convolution sum, either of the convolved functions can be selected for displacement. Thus convolution is a commutative operation, and the order of convolution in the preceding expressions can be reversed.

The utility of the convolution integral and its discrete representation, by which it can be computed by means of modern digital computer techniques, is in its ability to permit determination of the response to a general class of input signals with only a knowledge of the impulse, or unit-sample response, of the system. Thus convolution has broad applicability in various fields of engineering and science. A limitation to its usefulness in its direct form is that in general the number of operations required to compute the convolution sum is N^2.

The correlation integral [19] is generally represented as

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t + \tau)d\tau,$$

differing from convolution in that there is no time reversal, or folding about $\tau = 0$. The correlation sum for practical systems is

$$y(n) = \sum_{m=0}^{N-1} x(m)h(n + m).$$

Correlation is often referred to as the lagged-product operation. The relationship between discrete convolution and correlation is evident from their defining expressions. The difference is that there is no time reversal in correlation. The correlation operation is not commutative [58]. If the sequence to be shifted is an even function, convolution and correlation are identical. Because of the similarities, further discussion will center on convolution, and any differences with respect to correlation will be noted.

8.2 Application of the Convolution Theorem

The convolution theorem of Fourier analysis, as presented earlier, establishes a Fourier-transform-pair relationship between convolution in either the time or frequency domain and multiplication in the other domain. Thus in the continuous case the convolution integral can be evaluated from the inverse Fourier transform of the product of the Fourier transforms of $x(t)$ and $h(t)$ [72]. By direct substitution of the expressions for $x(n)$ and $h(n)$, in terms of IDFT's, into the convolution sum [58], or substitution of the product of the expressions for the DFTs of $x(n)$ and $h(n)$ into the expression for an IDFT [56], it can be shown that the convolution theorem applies in the discrete case and can be expressed [58] as

$$\sum_{m=0}^{N-1} x(m)h(n - m) \leftrightarrow X(k)H(k).$$

However, since the DFT is periodic, as discussed earlier, the multiplication of DFT's applies to convolution of periodic functions only [73]. To permit application of the convolution theorem in the discrete case it is necessary that the discrete time functions be made periodic [20]. The indices in the convolution sum are evaluated modulo N, and convolution of periodic functions is considered as a circular convolution in that the samples shifted out of one end of a period are shifted back into the other end. With respect to discrete correlation a Fourier-transform-pair relationship similar to that of the discrete convolution theorem exists and can be expressed [58] as

$$\sum_{m=0}^{N-1} x(m)h(n + m) \leftrightarrow X^*(k)H(k)$$

where the superscript asterisk indicates complex conjugation.

To visualize circular convolution, let each periodic time sequence be represented around a surface of a cylinder with a circumference of one period or N sample points. Let one cylinder be placed within the other, with each point in the convolution being computed by the sum of the products of all corresponding points on each cylinder following a unit circular shift of one cylinder relative to the other [11]. The circular shift, by which sample values are shifted from one end of a period into the other end, causes the convolution result of one period to interfere with the result of the following period [20]. To perform a linear convolution by means of a circular convolution, the sequences to be convolved must be suitably modified by inclusion of sufficient zero-valued samples to isolate each period. This modification requires that two sequences to be convolved of P and of Q samples be appended with $Q - 1$ and $P - 1$ zero-valued samples respectively so as to be each represented by a period of $N = P + Q - 1$ samples [20,73]. This ensures that there will be no overlap in the resulting convolution of period N which approximates the continuous convolution. Thus aperiodic or linear convolution in the discrete case can be performed by use of the DFT. To improve the efficiency of the convolution computation, the number of samples required to define each function in its periodic representation can be minimized by initially shifting the nonzero samples defining a function leftward to the origin and making a suitable correction in the resulting convolution [73]. For computation of discrete correlation, improved efficiency is attained if the sequence to be shifted in the correlation is initially shifted to the extreme right of the periodic interval and the other sequence is shifted to the extreme left, with suitable correction following the computation [20]. In convolving an infinitely long sequence with a finite sequence of duration Q, the resulting sequence contains a periodic interference error in the first $Q - 1$ sample points [73]. This is also often referred to as the *end effect* [20].

Since the convolution theorem is applicable to the discrete case, the convolution sum can be evaluated by two DFT's, a multiplication of the two resulting DFT sequences, and an IDFT. Since it was shown here earlier that an IDFT can be performed by a DFT, three DFT's are required. The desired application of discrete convolution to continuous functions results in some error. This is due to periodic-interference, or wrap-around, error and to evaluation of the convolution integral by the convolution sum which is effectively equivalent to approximation by the trapezoidal rule for numerical integration [72].

To demonstrate some of the concepts discussed, an example of a convolution [20] is presented in Fig. 8.1. Figure 8.1a indicates the continuous convolution of the aperiodic time functions $x(t)$ and $h(t)$. In Fig. 8.1b these functions are sampled and made periodic with period $N < P + Q - 1$, so that the resulting discrete convolution displays periodic overlap error. Figure 8.1c shows the discrete convolution with $N = P + Q - 1$ so that no overlap error results. If $N > P + Q - 1$, a correct aperiodic discrete convolution as in Fig. 8.1c would appear in each period, but there would be redundant zero samples following the nonzero samples of the convolution. As the sample interval is made smaller, the resulting discrete convolution would more closely approach the continuous convolution within a constant scaling factor.

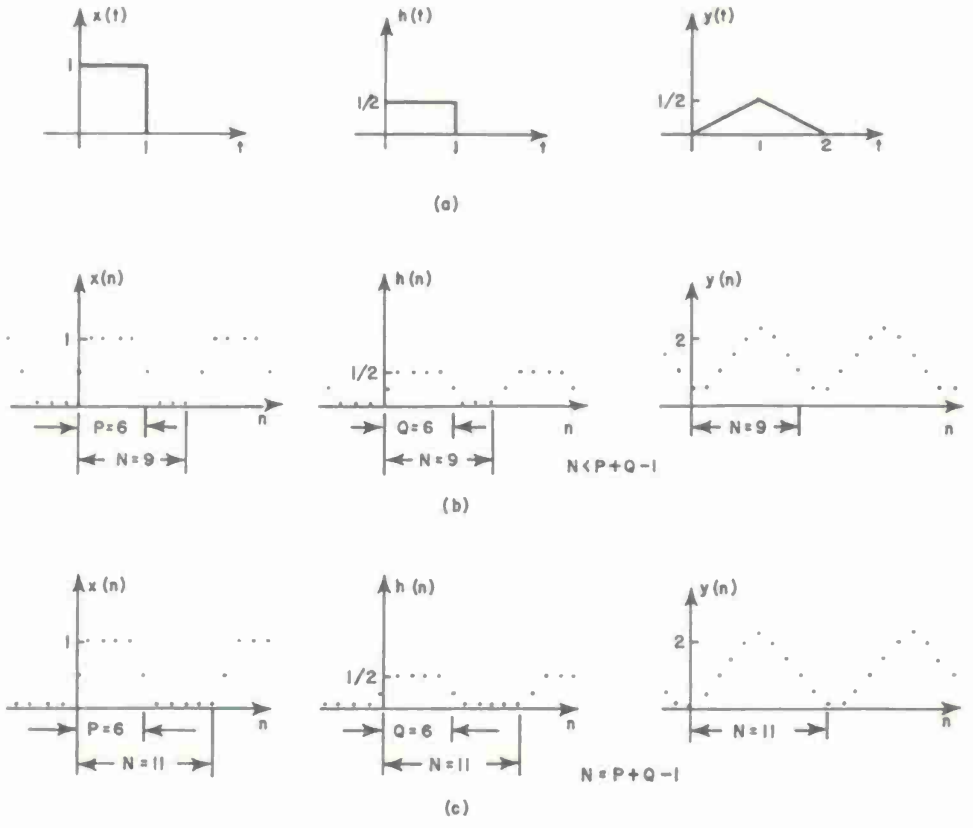


Fig. 8.1—Examples of continuous and discrete convolution.
(From Ref. 20 by permission.)

8.3 FFT Convolution and Correlation

It would appear that performing a discrete convolution or correlation by three DFT's is taking a long approach, since direct computation requires the evaluation of only a sum of N products for each sample point of the result. This would be a long approach if the DFT could be evaluated only directly. However, following disclosure of the basic FFT algorithm by Cooley and Tukey [9] as described here earlier, Stockham [74] proposed the concept of high-speed convolution and correlation whereby the required DFT's are evaluated by means of the FFT algorithm. Since the FFT for N an integer power of 2 requires a number of operations proportional to $N \log_2 N$, high-speed circular convolution of two sequences of length N requires a number of operations proportional to $3N \log_2 N$ plus N multiplications. Similar savings are possible for highly composite values of N. Based on his experimentation, Stockham [74] finds that the accuracy of the high-speed FFT procedure is as good or better than that of direct computation of discrete convolution by sums of products.

If the sequences considered for convolution are real, as is often the case, the properties of the DFT can be used to effect two real N-point transforms by means of a single complex N-point FFT [60]. The basis for this savings is the representation of the two real-valued sequences as the real and imaginary parts of the complex sequence to be transformed. Thus for real-valued sequences the number of operations and the resulting

computation time are approximately halved. Stockham's results [74] show high-speed convolution to be faster than direct evaluation for approximately $N \geqslant 28$, with an estimated computation-speed improvement of 80 times for $N = 10^{12}$.

The FFT is strictly a highly efficient algorithm for computing the DFT. Thus the previous discussion with respect to the DFT applies directly to high-speed convolution and correlation by means of the FFT. The details of the application of the FFT to computation of discrete convolution and correlation are presented in Refs. 20, 73, 74, and 75. To perform convolution of the sampled function $x(nT)$ of aperture P and starting point a with the sampled function $h(nT)$ of aperture Q and starting point b, where T is the sampling interval, it is first necessary, as in the case of direct evaluation, to shift the sequences to the origin to obtain computation efficiency by reduction of the effective periodic interval. Thus

$$x(n) = x(nT + a) \qquad n = 0, 1, \ldots, P - 1$$

$$h(n) = h(nT + b) \qquad n = 0, 1, \ldots, Q - 1.$$

It is also necessary to augment the shifted sequences with sufficient zero-valued samples to eliminate overlap effects. As described previously, this requires that each augmented sequence be at least of length $N = P + Q - 1$ with redundant zero-valued samples appearing in the resulting convolution sequence for values of N beyond the minimum. Assuming a radix 2 FFT algorithm, it is necessary that N also be some integer power of 2. Thus the value of N should be chosen such that

$$N = 2^m \geqslant P + Q - 1.$$

Then the augmented portions of the sequences are described as

$$x(n) = 0, \qquad n = P, P + 1, \ldots, N - 1,$$

and

$$h(n) = 0, \qquad n = Q, Q + 1, \ldots, N - 1.$$

The N-point DFT's of $x(n)$ and $h(n)$, $X(k)$ and $H(k)$, are computed by means of the FFT, and then the product of corresponding samples, $X(k)H(k)$, is evaluated. The IDFT of this product is found by computing the complex conjugate of the DFT of $(1/N)[X(k)H(k)]^*$ to give the desired convolution sequence. The FFT computation of discrete correlation, as described in Ref. 20, differs from convolution in two respects. First, the sequence to be shifted in the correlation is shifted to the upper end of the periodic interval N, with the zero-valued samples placed at the lower end of the interval. Second, the frequency-domain product is performed using the complex conjugate of the unshifted sequence, or kernel, of the correlation.

8.4 Sectioning

The discussion on convolution and correlation to this point has been limited to sequences that are finite in extent. In many practical applications the sequence $h(n)$

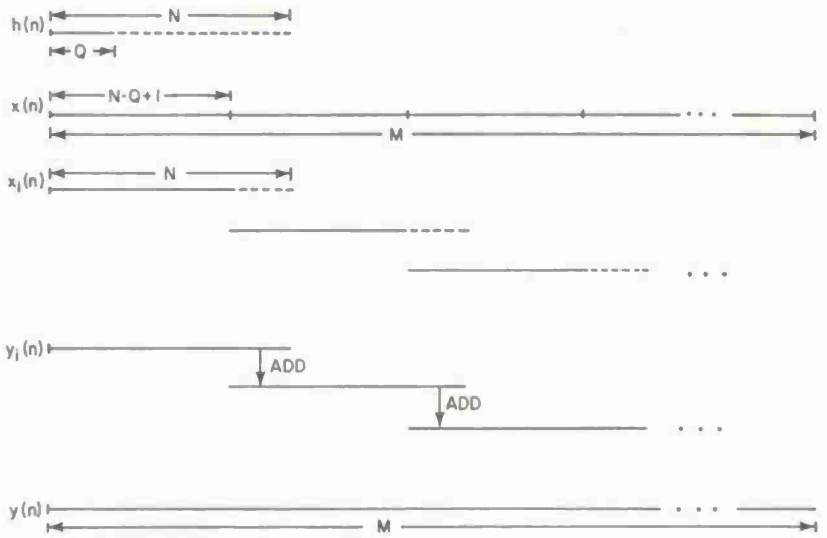


Fig. 8.2—Overlap-add method of sectioning

representing the unit-sample response of a system will be finite or can be suitably approximated by a finite sequence. The input sequence $x(n)$ however could have an extent that is infinite or beyond the capacity of available memory. Even if it is possible to compute the convolution, or correlation, for some situation where $P \gg Q$, there would be considerable delay in the generation of the results, since the resulting convolution could not be computed until all input samples were available. One of the most significant aspects of the high-speed convolution and correlation disclosure of Stockham [74] is the proposal to overcome these problems by sectioning the input data so that a series of smaller procedures can be performed. The methods that have been developed for sectioning make FFT convolution and correlation viable procedures in practical applications with extended input signals.

8.4.1 *Overlap-Add Sectioning*

The method of sectioning first proposed by Stockham [74] and described further in Refs. 73 and 75 is the *overlap-add* method. The following description of this method is aided by reference to the diagram of Fig. 8.2. It is assumed that the unit-sample response $h(n)$ of aperture Q is to be convolved with the M-point sequence $x(n)$ which is in general a portion of a longer or possibly infinite sequence, with $M \gg Q$. In this method the sequence $h(n)$ is considered as augmented with $N - Q$ zeros to form a periodic function of period N. The sequence $x(n)$ is partitioned into sections of $N - Q + 1$ samples. By appending $Q - 1$ zero samples to each of these sections, periodic sequences $x_i(n)$ of period N are formed. Then

$$x(n) = \sum_{i=0}^{K-1} x_i(n),$$

where K is the number of sections required. The purpose of adding $Q - 1$ zeros to each section is to prevent overlap error, as discussed previously. From the representation of $x(n)$ as a sum of the overlapping sequences $x_i(n)$, it can be seen that the convolution of

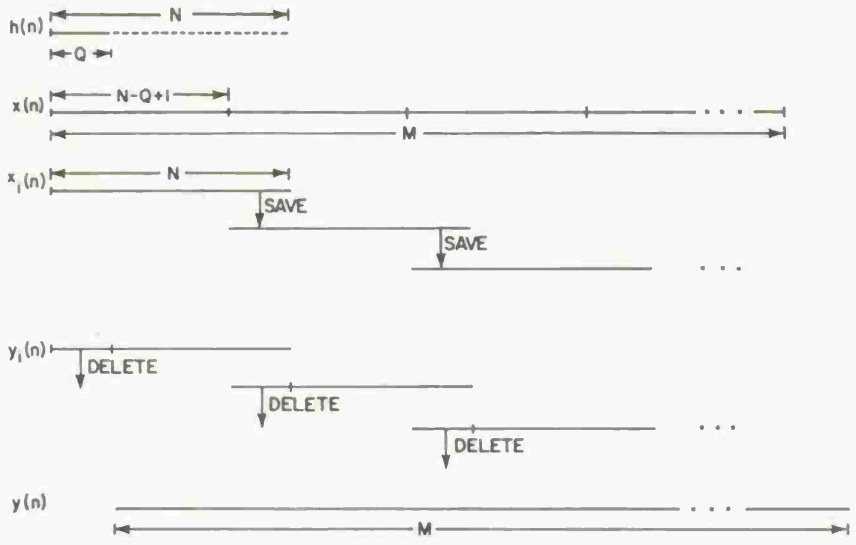


Fig. 8.3—Select-save method of sectioning

$x(n)$ and $h(n)$ can be performed as a summation of the shorter convolutions $y_i(n)$ of each $x_i(n)$ and $h(n)$. In performing these convolutions by means of a radix-2 algorithm, it is necessary that N be an integer power of 2. The overlap-add operation thus consists of shifting forward $N - Q + 1$ samples to the next $x_i(n)$, starting with $i = 0$, performing the convolution of $x_i(n)$ and $h(n)$, and adding each result $y_i(n)$ through $i = K - 1$ to an accumulation $y(n)$ which will represent the overall convolution.

8.4.2 *Select-Save Sectioning*

The *select-save* method of sectioning was proposed by Helms [75] and is discussed further by Stockham [73], Brigham [20], and others as the *overlap-save* method. The description of this method that follows is aided by reference to Fig. 8.3. In this method both the augmentation of $h(n)$ with zero-valued samples and the determination of the value of N are performed in the same manner as in the overlap-add method. The effectively periodic sequences $x_i(n)$ of period N are formed from $x(n)$. The sequence $x_0(n)$ consists of the first N samples of $x(n)$. The remaining sequences $x_i(n)$, $i \geq 1$, are formed from the last $Q - 1$ samples of $x_{i-1}(n)$ followed by the succeeding segment of $N - Q + 1$ new samples. Thus, if the number of total samples M of $x(n)$ is finite, the number of sequences $x_i(n)$ will be K, the same as in the overlap-add method. The FFT convolution of $h(n)$ and each sequence $x_i(n)$ is computed as described previously. In each resulting convolution sequence $y_i(n)$, of period N, the first $Q - 1$ samples are invalid. The correct overall convolution $y(n)$ is formed by appending the succeeding sequences of $N - Q + 1$ valid sample points from each succeeding $y_i(n)$. Because of the end effect the first $Q - 1$ values of $y(n)$ are undefined. The method of forming $y(n)$ from the $y_i(n)$ can be shown as follows:

$$
\begin{aligned}
y(n) &\text{ undefined}, && n = 0, 1, \ldots, Q-2,\\
y(n) &= y_0(n), && n = Q-1, Q, \ldots, N-1,\\
y(n+N) &= y_1(n+Q-1), && n = 0, 1, \ldots, N-Q,\\
y[n+2N-(Q-1)] &= y_2(n+Q-1), && n = 0, 1, \ldots, N-Q,\\
&\ldots && \ldots\\
y[n+iN-(i-1)(Q-1)] &= y_i(n+Q-1), && n = 0, 1, \ldots, N-Q.
\end{aligned}
$$

8.4.3 *Determination of N*

With respect to either sectioning method, if it is assumed that $M \gg N - Q + 1$, that is, $K \gg 1$, then the computation time of $H(k)$ can be ignored since it need be computed only once. Thus the number of N-point FFT's required is $2M/(N - Q + 1)$. Larger values of N reduce the total number of FFT's but increase the number of sample points involved in each FFT. It is desired to find the value of N that optimizes the overall computation time. This has been done numerically by Helms [75] for various ranges of Q, and the results are shown in Table 8.1. If N becomes too large in relation to available high-speed memory, Stockham [73,74] suggests that the unit-sample response $h(n)$ be split into "packets" which are considered individually, as separate unit-sample responses, with the results added together after suitable shifting. It is determined experimentally that, to avoid packeting, N must be limited to about 1/8 of the total memory not used for the program.

Table 8.1
Optimum Values of N for High-Speed Convolution

Q	N_{opt}	$\log_2 N_{opt}$
< 11	32	5
11 - 17	64	6
18 - 29	128	7
30 - 52	256	8
53 - 94	512	9
95 - 171	1,024	10
172 - 310	2,048	11
310 - 575	4,096	12
575 - 1050	8,192	13
1050 - 2000	16,384	14
2000 - 3800	32,768	15
3800 - 7400	65,536	16
> 7400	131,072	17

If the time sequences involved in the computation are real, the economy of the method of performing two N-point real FFT's by one complex N-point FFT, as described earlier, can be applied to reduce the total computation time of sectioning by approximately half. This is accomplished by combining pairs of successive sequences $x_i(n)$ and $x_{i+1}(n)$ as the real and imaginary parts of the input to a complex FFT. The desired DFT's $X_i(k)$ and $X_{i+1}(k)$ are found from the resulting complex transform as described previously.

8.5 Applications of High-Speed Convolution and Correlation

The concepts of high-speed convolution and correlation are applicable to the solution of computation problems in various areas of digital signal processing and discrete system representation. These include digital filtering, spectral estimation, and the computation of DFT's.

As discussed earlier, the convolution sum permits the determination of the output of a discrete system from a knowledge of the unit-sample response of the system. Thus a discrete system may be implemented by means of a discrete convolution using a unit-sample response determined by experimentation or by mathematical analysis based on the desired performance of the system. The unit-sample response must of course be nonzero only in a finite interval or must be approximated as such. Such systems designed specifically to possess desired system-frequency-response characteristics, were classified here earlier as digital filters. Digital filter designs are generally realized using finite-difference equations, the computations for which are much faster than those required for high-speed convolution. Stockham [74] suggests the high-speed convolution technique to accomplish filter characteristics beyond the capability of simple difference-equation techniques.

A discussion of digital filter design using high-speed convolution is presented by Stockham [73] and Helms [75]. The required unit-sample response is determined from the desired frequency response by transforming, truncating, applying window functions to reduce Gibbs phenomenon effects, inverse transforming, and testing for comparison with desired characteristics. Helms [75] and Gold and Jordan [31] discuss the realization of difference equations, in the form of discrete convolutions, computable by high-speed techniques.

Power spectra can be estimated directly as the square of the magnitude of the DFT of a windowed time sequence. By an indirect method, spectra can be estimated as the DFT of the autocorrelation sequence multiplied by a suitable window function. Rader [76] proposes the use of high-speed FFT correlation, using a sectioning technique, to compute the autocorrelation. The DFT for values of N not suitable for the FFT can be computed using high-speed convolution based on the representation of the DFT as a discrete convolution by the CZT algorithm described previously in the section on the FFT.

9. QUANTIZATION EFFECTS

In the implementation of continuous systems the value of components required to exactly meet a desired specification must in general be approximated. This is due

primarily to the inherent variations in practical component manufacturing processes. Additionally the dynamic range of signal inputs is limited by large-signal-performance and power limitations of circuit components. In the implementation of digital signal processing systems there are also problems of practical representation of values required for exact implementation of theory. These problems are collectively referred to as quantization effects and result in digital output signals that differ from those described by theory. These effects include the following:

- Input quantization—Continuous-signal sample values are approximated by the nearest level described by a finite-length binary number.

- Coefficient quantization—The coefficients required to implement a discrete algorithm are represented in a finite binary form with values generally different from that necessary to satisfy the requirements of a particular design specification.

- Roundoff error—The results of arithmetic operations in digital systems must be represented within registers of fixed length.

- Dynamic-range limitations—Due to the finite representation of numbers in digital systems, the magnitude of the input sequence must be constrained to limit the values at each point in the system and thus avoid distortion due to overflow.

These effects are inherent in digital systems and cannot be eliminated but can only be reduced by using longer register lengths or choosing among implementations that result in smaller quantization effects. The factors involved in the generation of errors due to quantization have been analyzed in the literature, which will be reviewed here. Due to the nature of quantization effects, many of the analyses in the literature involve the development of statistical and worst-case models of the generation of quantization effects. Of course such models have value only in relationship to proven agreement between predicted results and those observed in actual implementations. All analyses to be reviewed here are within such agreement.

9.1 Number Representations

Numerical values are represented within binary digital machines in one of several forms. These forms are of register lengths that are a compromise between precision of representation and data storage and processing efficiency. Floating-point representation is most commonly available in large-scale general-purpose computers. Minicomputers and special-purpose devices are generally limited to fixed-point representation. The details of fixed-point and floating-point number representation and arithmetic operation in binary machines are described in such sources as Refs. 77 and 78.

9.1.1 *Fixed-Point Number Representation*

In applications of digital signal processing all fixed-point numbers are generally represented as fractions, with an additional sign bit to the left of the binary point [79]. In this form the product of any two numbers is less than 1; thus no overflow can occur. Overflow can however result from the addition of such numbers and the values of these

numbers must be constrained to limit the magnitude of sums to less than 1. There are three forms of general representation of signed fixed-point numbers: sign and magnitude, 2's complement, and 1's complement. The representation of positive numbers is the same in all three forms, a 0 sign bit followed by b_1 magnitude bits, where b_1 is the number of bits required to precisely represent the magnitude M.

In the sign and magnitude representation, negative numbers are represented with a 1 sign bit followed by b_1 magnitude bits. In the 2's complement representation negative numbers are represented as $2.0 - M$, and in the 1's complement representation negative numbers are represented as $2.0 - M + 2^{-b_1}$, which includes the sign bit. The b_1-bit precise magnitude must be reduced to b bits. This quantization is accomplished by roundoff or truncation. In truncation the bits beyond the bth position are deleted. In rounding, the number is represented by the closest quantized value of b bits. This results in errors relative to the unquantized representation. These errors, which depend on the particular number representation in the case of truncation, are given by Oppenheim and Schafer [11] as

Truncation:

positive numbers and

2's complement negative numbers: $-2^{-b} < \epsilon \leqslant 0$,

sign and magnitude and

1's complement negative numbers: $0 \leqslant \epsilon < 2^{-b}$,

Rounding: $-\frac{1}{2}2^{-b} < \epsilon \leqslant \frac{1}{2}2^{-b}$.

If two fixed-point numbers with $b + 1$ bits are added, the result will have $b + 1$ bits, assuming no overflow. However, if these quantities are multiplied, the product will in general have more than $b + 1$ bits and will require truncation or rounding with generation of errors as just indicated.

9.1.2 *Floating-Point Number Representation*

A number x can be represented as a floating-point number in the form $(\text{sgn})2^c M$, where c is the characteristic or an integer exponent that is the smallest integer exceeding $\log_2 |x|$ and M is thus a fraction between 1/2 and 1 called the mantissa. The error in a quantized floating-point representation $Q(x)$ is relative to the quantized value x. Thus $Q(x) = x(1 + \epsilon)$, where, for rounding in the case of a b-bit mantissa, $-2^{-b} < \epsilon \leqslant 2^{-b}$. Due to the normalization required in floating-point addition, both addition and multiplication introduce quantization error; thus

$$Q(x_1 + x_2) = (x_1 + x_2)(1 + \epsilon)$$

and

$$Q(x_1x_2) = (x_1x_2)(1 + \delta).$$

The truncation of the mantissa for the representation of x results in a relative error ϵ, given by Ref. 11 as

1's complement and

sign and magnitude: $-2^{-b+1} < \epsilon \leqslant 0,$

2's complement: $-2^{-b+1} < \epsilon \leqslant 0, \quad x > 0,$

$0 \leqslant \epsilon < 2^{-b+1}, \quad x < 0.$

Since floating-point numbers have an exponent, a longer total register length than b bits is actually required. Further discussion of quantization effects will, as in the literature, be limited to roundoff. The results of roundoff analysis can be extended to truncation, which generates greater quantization errors.

9.1.3 *Block Floating-Point Representation*

To combine, to some extent, the dynamic range advantages of floating-point arithmetic with the increased accuracy and simplicity of fixed-point arithmetic, Oppenheim [80] proposed block floating-point arithmetic. In block floating-point arithmetic the input samples and the outputs of the delay registers are jointly normalized prior to the fixed-point multiplications and additions of the particular algorithmic process. To compensate for the normalization, the output is correspondingly scaled, producing a fixed-point result.

9.2 Input Quantization Effects

In sampling a continuous signal, each sample must be represented within the finite binary word size of the quantizing unit, which will generally be equal to that used within the processing element. Thus each sample is represented as the nearest one of a finite set of quantization levels separated by 2^{-b} when there are b bits in the binary representation. The error in the representation of any sample is uniformly distributed between $-(1/2)2^{-b}$ and $(1/2)2^{-b}$ with zero mean and variance $\sigma_e^2 = 2^{-2b}/12$. For floating-point representation the quantization error will depend on the statistics of the input signal. If $x(n)$ is a stationary random process with variance σ_x^2, then $\sigma_e^2 = \sigma_\epsilon^2\sigma_x^2$, assuming that $x(n)$ and ϵ are uncorrelated [81]. Bennett [82] and Widrow [83] show that if the signal variation is large in comparison to the 2^{-b} steps and fairly rapid in relation to the sample interval, the quantization noise can be treated as uncorrelated with the signal and as white noise. Thus the effects of input quantization can be represented by a noise source at the input to the digital processing system. The output due to noise is added to that due to the noiseless input to form a statistical representation of the total output. Using the convolution sum, Gold and Rader [84] show that in the steady state the output variance σ_0^2 resulting from an input that is zero for $n < 0$, with variance σ_i^2, is

$$\sigma_0^2 = \sigma_i^2 \sum_{n=0}^{\infty} h^2(n),$$

where $h(n)$ is the unit-sample response of the system. The identity

$$\sum_{n=0}^{\infty} h^2(n) = \frac{1}{2\pi j} \oint H(z) H\left(\frac{1}{Z}\right) z^{-1}\, dz$$

permits summation or contour integration to be used to compute the variance of the output noise due to input quantization. Algebraic methods developed to compute the contour integration will be discussed later.

9.3 Coefficient Quantization

9.3.1 *Fixed-Point Implementation*

In implementing a digital system design it is generally necessary to quantize the coefficients of the system function $H(z)$. The resulting system then differs from the desired design in response characteristic and, for the case of poles of $H(z)$ at or near the unit circle, can be unstable. The sensitivity of pole locations to coefficient quantization was first investigated by Kaiser [28]. He determined that for tightly clustered poles and a large sampling rate the required coefficient accuracy increases approximately linearly with the order of the filter. By approximate analysis assuming simple poles, he developed a lower bound on the accuracy required to guarantee stability. This analysis can be extended to multiple poles and bandpass or highpass filters. Kaiser concluded that under the assumptions of his analysis the problem of coefficient accuracy is most severe for realizations in the direct form that use the denominator polynomial of $H(z)$ in unfactored form. In factored form the pole-position sensitivity to coefficient accuracy is decreased; thus realization in cascade or parallel combinations of low-order forms should be used, especially in complex filters with steep transitions between passbands and stopbands. Kaiser, as well as Knowles and Edwards [85], found the parallel form less susceptible to coefficient quantization than the cascade form.

Rader and Gold [86] considered the effect of coefficient quantization on the pole locations of realizations of first- and second-order filters. For the first-order filter

$$y(n) = Ky(n - 1) + x(n),$$

the error in pole position is the error in quantization of the single coefficient. For the second-order filter

$$y(n) = Ky(n - 1) - Ly(n - 2) + x(n)$$

with complex conjugate poles at $re^{\pm j\theta}$, the errors in r and θ are approximately

$$\Delta r \approx \frac{\Delta L}{2r}$$

and

$$\Delta\theta \approx \frac{\Delta L}{2r \tan\theta} - \frac{\Delta k}{2r \sin\theta},$$

where the error in θ can be quite large for θ near 0. Rader and Gold propose a coupled form of second-order equation

$$y_1(n) = Ky_1(n-1) - Ly_2(n-1) + Ax(n)$$

and

$$y_2(n) = Ly_1(n-1) + Ky_2(n-1) + Bx(n)$$

with $K = r\cos\theta$ and $L = r\sin\theta$, which requires more computation but whose poles are less sensitive to coefficient quantization, with

$$\Delta r \approx \Delta K \cos\theta + \Delta L \sin\theta$$

and

$$\Delta\theta \approx \Delta L \frac{\cos\theta}{r} - \Delta K \frac{\sin\theta}{r}.$$

Knowles and Olcayto [87] represent coefficient quantization as a stray transfer function in parallel with the ideal transfer function and, making statistical assumptions, evaluate the expected mean-square difference between the frequency responses of the actual and ideal filters. Their analysis is carried out for the direct, parallel, and cascade forms. They conclude that even one extra bit significantly improves realization accuracy and that the direct form is more sensitive to coefficient accuracy than the parallel or cascade form, with less degeneration for the parallel form than for the cascade form.

To evaluate the quantization effects in different realizations of a given transfer function, Mitra and Sherwood [88] propose a technique which relates pole (or zero) displacement to small changes in multiplier coefficients. The pole displacements are expressed in a vector equation as the dot product of a sensitivity vector, derived on the basis of the ideal pole positions, and a vector of coefficient variations. Thus the same sensitivity vector can be used for various sets of coefficient changes. Outside of direct realizations multiplier and coefficient values do not have a direct relationship; thus it is necessary to determine a matrix which represents a dot-product relation between coefficient and multiplier quantization errors. The number of bits required for each multiplier to maintain all poles within some prescribed limit can be determined for truncation or rounding from the sensitivity vectors and the vector relating coefficient and multiplier errors. This analysis provides individual bit requirements for each multiplier in special-purpose hardware or provides a check on pole and zero displacements in general-purpose computer implementations with fixed register length.

The available pole locations for the direct and coupled second-order filter forms can be plotted on a grid as shown in Figs. 9.1 and 9.2 [81]. It can be seen from Fig. 9.1

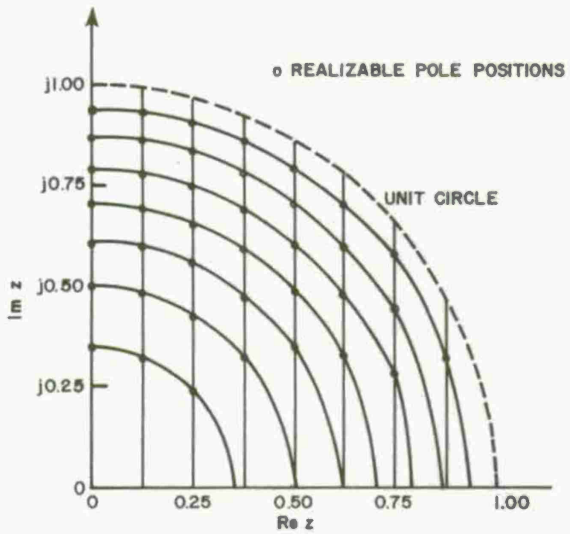


Fig. 9.1—Grid of allowable pole positions for the direct second-order filter form

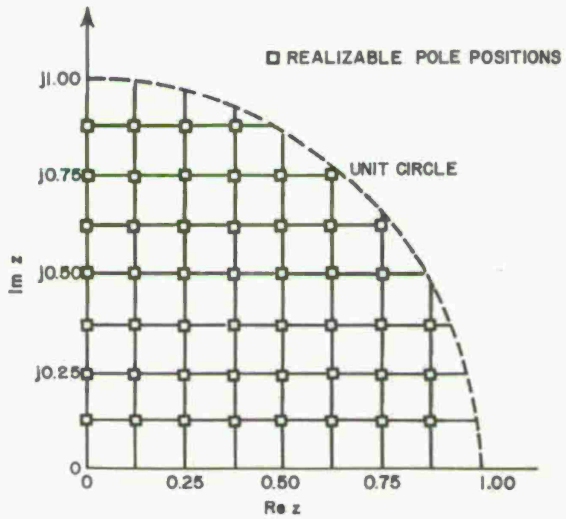


Fig. 9.2—Grid of allowable pole positions for the coupled second-order filter form

that the availability of allowable pole locations for the direct form increases as r and θ approach 1 and $\pi/2$ respectively. For the coupled form, as shown in Fig. 9.2, the allowable poles are located uniformly in the z plane. It can be seen that the coefficient sensitivity for a given design varies with the structure used for realization. The structure should be chosen that provides the greatest availability of quantized poles near the pole location required for the desired design. The variation of coefficient sensitivity with structure is analyzed by Crochiere [26], who by computer analysis compares 13 digital network structures. For the comparison an eighth-order elliptic bandpass filter is used as the design goal for each structure. A required word length, defined on a statistical basis, is determined for each structure. The variation in this word length is 3 to 1 for the structures analyzed.

Given a particular structure, a filter design can be optimally obtained with respect to coefficient accuracy by techniques that search over the grid of allowable pole positions corresponding to the particular structure, as proposed by Avenhaus and Schussler [89] and Avenhaus [90].

9.3.2 *Floating-Point Implementation*

The effect of coefficient quantization in floating-point digital filters was analyzed by Weinstein [81] with respect to pole sensitivity in the direct and coupled forms of second-order filters. For the direct form, if the roundoff errors in the coefficients are of the form $\Delta a_1 = \epsilon_1 a_1$ and $\Delta a_2 = \epsilon_2 a_2$, where $|\epsilon_1| < 2^{-b}$ and $|\epsilon_2| < 2^{-b}$, the error in pole position is approximately,

$$\Delta r \approx \epsilon_2 \left(\frac{r}{2}\right)$$

and

$$\Delta\theta \approx \epsilon_2 \left(\frac{r}{2 \tan\theta}\right) - \frac{\epsilon_1}{\tan\theta} .$$

These are similar to the fixed-point case, with comparable allowable pole spacing in the z plane, except that the density is greater for θ near $\pi/2$. For the coupled form filter the pole sensitivity is not significantly different from the fixed-point case, but for small values of the coefficients a_1 or a_2 or for θ near 0 or $\pi/2$ the grid of allowable poles is much finer.

Coefficient quantization in the floating-point case was also analyzed by Kaneko and Liu [91]. They show that the error, which is the difference in filter outputs due to finite and infinite precision realizations, consists of two uncorrelated components, one due to arithmetic roundoff and the other due to coefficient roundoff. For the direct, parallel, and cascade forms expressions for the mean-square value of this error are derived. These are proportional to $2^{-2b}/3$, the variance of the relative floating-point roundoff error. The analysis indicates that the error due to coefficient quantization is greatest for the direct form and slightly greater for the cascade form than for the parallel form. In this analysis Kaneko and Liu also develop lower bounds for mantissa length necessary to insure stability.

9.4 Dynamic-Range Limitations

As discussed earlier, in fixed-point realizations the input magnitude must be constrained to prevent overflow and distortion. For floating-point implementations the available dynamic range is generally assumed large enough to permit representation of any value generated in the system; thus the dynamic range problem is usually ignored. The fixed-point dynamic-range problem was analyzed by Jackson [34]. He derived one form of this constraint from the expression for the output $y_i(n)$ at the ith system node in terms of the convolution sum

$$y_i(n) = \sum_{k=0}^{\infty} h_i(k)x(n - k),$$

where $x(n)$ is the input sequence and $h_i(n)$ is the unit-sample response at the ith node. If $|x(n)| < x_{max}$, then

$$|y_i(n)| \leqslant x_{max} \sum_{k=0}^{\infty} |h_i(k)|.$$

Since in the present discussion each fixed-point value represents a signed fraction, to prevent overflow

$$|y_i(n)| < 1.$$

Thus the samples $x(n)$ must be constrained so that

$$|x(n)| < \frac{1}{\sum_{k=0}^{\infty} |h_i(k)|}.$$

Since the summation over $h_i(k)$ is generally difficult to evaluate, and since the preceding constraint is somewhat pessimistic for certain classes of signals, Jackson derived less general conditions using L_p norms. The L_p norm of a function $A(\omega)$ is defined as,

$$\|A\|_p = \left[\frac{1}{\omega_s} \int_0^{\omega_s} |A(\omega)|^p \, d\omega\right]^{1/p},$$

where ω_s is the radian sampling frequency $2\pi/T$. For continuous functions the L_p norms for $p = 1$, 2, and ∞ are the mean absolute value, rms magnitude, and maximum magnitude respectively over a period of the function. The Fourier transforms of $y_i(n)$, $x(n)$, and $h_i(n)$ can be represented as $Y_i(\omega)$, $X(\omega)$, and $H_i(\omega)$, respectively. Using the convolution theorem and the Schwarz inequality, Jackson shows that

$$|y_i(n)| \leqslant \|H_i\|_p \|X\|_q, \frac{1}{p} + \frac{1}{q} = 1.$$

Since we must have $|y_i(n)| < 1$, the input must be constrained so that

$$||X||_q < \frac{1}{||H_i||_p}.$$

Thus the input can be constrained in terms of either the rms or peak magnitude of $H_i(\omega)$. Jackson extended his analysis to the case of random inputs, determining similar conditions involving the variance of the output, the power spectral density of the input, and $||H_i||_p$.

9.5 Roundoff Errors

9.5.1 *Fixed-Point Arithmetic*

Each source of roundoff error for fixed-point arithmetic can be treated statistically, using linear system noise theory, in a manner similar to that of input quantization errors. To formulate a statistical model of roundoff noise it is necessary to make certain assumptions as considered by Knowles and Edwards [85] and others. It is assumed that the errors in each roundoff process are uncorrelated with the signal, are uncorrelated from sample to sample, and uncorrelated with other error sources. Thus the roundoff error is represented as an additive white-noise input source to the system. For rounding, the noise source has zero mean and variance $2^{-2b}/12$. Truncation involves some signal dependency and thus does not satisfy these assumptions.

Knowles and Edwards [85], using the white-noise model, analyzed fixed-point roundoff errors for direct, parallel, and cascade forms. Each noise source was represented as the input to a system $H_p(z)$, which represents the system function between the pth roundoff noise source and the system output. They formulated bounds on the mean-square measure of the system noise as formed by the sum of the outputs of the $H_p(z)$ as

$$y^2 \leqslant \sum_{p=1}^{N} |H_p^*(j\omega_m)|^2 \phi_{pp}^*(0),$$

where N is the number of error sources, $|H_p^*(j\omega_m)|$ is the maximum gain of $H_p(z)$, and $\phi_{pp}^*(0)$ is the maximum of the autocorrelation function of the pth error source. Their analysis indicates mean-square error in the system output due to roundoff is greatest in the direct form as compared to the cascade or parallel form.

Rader and Gold [86] used a similar model to analyze the error due to roundoff in several first- and second-order filters that can be used to represent higher order systems in cascade or parallel form. They demonstrated that different network structures for the same system function have different output noise characteristics, since, in general, roundoff noise can be generated at different points in the system. As described further in Refs. 12 and 30, each of the noise sources passes through different combinations of high-gain poles and low-gain zeros. Also different network-structure realizations of the same system function have different numbers of multipliers [26,39]. The method of determination of the output due to each noise source $\sigma_{e_i}^2$ is identical to that described earlier for input quantization error. Thus the total output noise is

$$\sigma_e^2 = \sum_{i=1}^{N} \sigma_{e_i}^2 \sum_{n=0}^{\infty} h^2(n).$$

The input quantization noise can also be included as a noise source at the input and used in determining total output noise.

As described earlier in the discussion on input quantization, the identity

$$\sum_{n=0}^{\infty} h^2(n) = \frac{1}{2\pi j} \oint H(z) H\left(\frac{1}{z}\right) z^{-1}\, dz$$

permits the output error variance due to rounding to be computed by use of either the summation or contour integral forms. The summation form is practical for only the simplest first- or second-order systems. The contour integral form for high-order $H(z)$ is also difficult to evaluate. Closed-form solutions have been tabulated by Jury [15], and a recursive formula for evaluation has been proposed by Åström, Jury, and Agniel [92]. A simple method of evaluation has been proposed recently by Mitra, Hirano, and Sakaguchi [93]. This method involves partial-fraction expansion of the noise transfer functions and thus replaces the contour integral by a sum of simpler integrals. As there are three possible forms in the partial-fraction expansion, there are only nine possible simpler integrals which are evaluated and tabulated, with four being of zero value.

Using the concepts of fixed-point roundoff error analysis described to this point in conjunction with the input dynamic range constraint, Weinstein [81] develops output noise-to-signal variance ratios for both white and narrowband signals applied to first- and second-order filters. As an example of such analysis, consider the fixed-point first-order filter of Fig. 9.3. For this filter, $h(n) = a^n$ and

$$\sum_{n=0}^{\infty} h^2(n) = \frac{1}{1 - a^2}.$$

The noise due to multiplication roundoff enters the system with variance $\sigma_{e_1}^2 = 2^{-2b}/12$ at the same point as the input. The steady-state output-noise variance is then

$$\sigma_e^2 = \sigma_{e_1}^2 \sum_{n=0}^{\infty} h^2(n) = \left(\frac{2^{-2b}}{12}\right) \frac{1}{1 - a^2}.$$

Imposing dynamic-range constraint to prevent overflow,

$$|x(n)| < 1 - a.$$

Assuming $x(n)$ is white and uniformly distributed over this range, $\sigma_x^2 = (1 - a)^2/3$ and the output signal variance is thus

$$\sigma_y^2 = \sigma_x^2 \sum_{n=0}^{\infty} h^2(n) = \frac{1 - a}{3(1 + a)}.$$

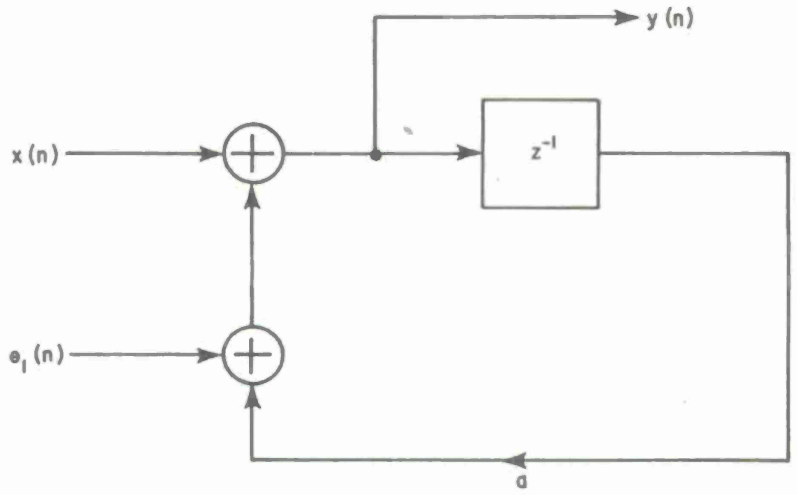


Fig. 9.3—First-order fixed-point filter with roundoff noise

If $\delta = 1 - a$, the noise-to-signal ratio is

$$\frac{\sigma_e^2}{\sigma_y^2} = \frac{1}{4}\frac{2^{-2b}}{\delta^2},$$

which is inversely proportional to δ^2, the square of the distance of the pole from the unit circle. For a sinusoidal input at low frequencies and small δ, the noise-to-signal ratio is inversely proportional to δ. The difference is attributable to increased gain for the low-frequency sinusoid as opposed to the white input. Similar analysis by Weinstein for second-order filters indicates the coupled form [86] to be superior to direct and canonic [84] forms with respect to noise-to-signal ratios at low frequencies.

Utilizing the various dynamic-range limitation concepts developed in Ref. 34, Jackson [94] analyzes the roundoff noise outputs from two transpose configurations, both for the cascade and parallel forms of a digital filter using a fixed-point noise model and limited dynamic range. Additional multipliers for scaling to satisfy dynamic-range constraints are included in the configurations. The analysis compares the different forms for various conditions of constraint on input and transfer function spectra on the basis of the variance or peak magnitude of the output noise due to roundoff. The results indicate little difference in the choice of parallel forms. For the cascade case, some advantages of one form over another are indicated. The analysis also indicates the variation in output noise measure for cascade forms with both the ordering of second-order sections and the pairing of poles and zeros. Good sequential orderings of sections are indicated by the variation in the *peaking* or ratio of peak-to-rms values of section transfer functions. The numerator and denominator factors corresponding to zeros and poles respectively should be paired to minimize the peak value of transfer functions of resulting individual sections.

Up to this point the discussion has been limited to IIR filters. The analysis of quantization in FIR digital filters has been considered by Chan and Rabiner [95,96]. In Ref. 95, relations for determining roundoff noise for statistical noise models are presented. Scaling methods for FIR filters are discussed and compared with IIR scaling procedures. The various quantization effects for the direct form are discussed in Ref. 96 for FIR filters, with concentration on coefficient quantization. Statistical bounds on the error in frequency response due to coefficient quantization are developed. The direct form is shown to be attractive due to low input quantization (A/D) and roundoff noise and minimal number of multipliers.

9.5.2 *Floating-Point Arithmetic*

From the earlier discussion on floating-point arithmetic as applied in digital filters, its error characteristics as compared to the fixed-point case have some differences that must be taken into account in any roundoff error analysis. First, the error in the representation depends on the magnitude of the quantity represented; second, roundoff error occurs during addition as well as during multiplication. An analysis of the accumulation of errors due to roundoff in floating-point digital filters, was presented by Sandberg [97]. Sandberg's analysis is nonstatistical and results in a deterministic bound based on the worst possible accumulation of errors. He uses a flow graph indicating the ordering of operations, and thus the accumulation of error, in the computations described by the linear difference equation used to implement a digital filter. For $e(n) = y(n) - w(n)$, which is the difference between the computed output and the ideal output at the nth iteration along with $K \geqslant N$, where N is the order of the filter, and

$$\langle e \rangle_K = \left(\frac{1}{K+1} \sum_{n=0}^{K} |e(n)|^2 \right)^{1/2},$$

which is the rms value of $e(n)$, the bound is expressed as

$$\langle e \rangle_K \leqslant c \langle y \rangle_K + f(K).$$

The function $f(K)$ and the constant c depend on the filter coefficients a_i and b_i, represented as machine numbers, on the order in which the products in the difference equation are summed, and on the number of bits in the mantissa. As $K \to \infty$, $f(K) \to 0$; thus c is an upper bound on an asymptotic output error-to-signal ratio.

Taking a more general statistical approach to floating-point roundoff error, Liu and Kaneko [98,99] derive expressions for the power spectral density $\Phi_{ee}(z)$ of the error sequence $e(n)$ for direct, parallel, and cascade forms for both roundoff and truncation. In their analysis they, like Sandberg, analyze the combination of errors using a flow graph of the difference equation. The random variables corresponding to the relative errors are indicated on the flow graph for an Lth-order filter in Fig. 9.4. The mean-square error is computed from $\Phi_{ee}(z)$ by a contour integration about $|z| = 1$. The results in all cases are proportional to 2^{-2b}, where b is the mantissa length, and for each realization form the error power spectral density for truncation is equal to that for rounding plus an additional term. Using the results of their analysis, Liu and Kaneko

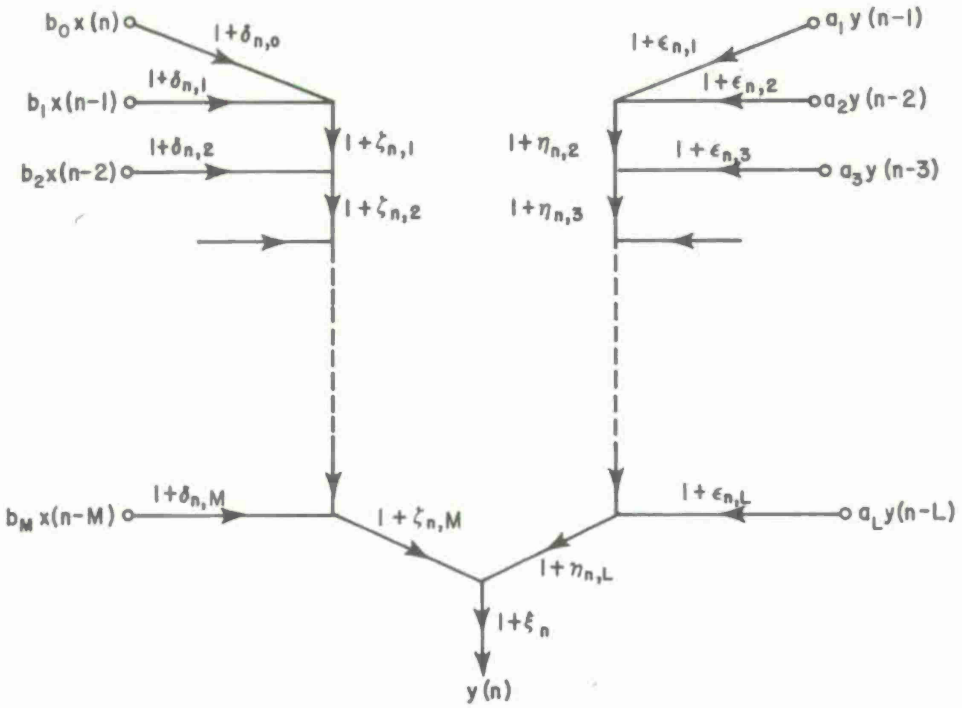


Fig. 9.4—Flow graph for a floating-point filter indicating roundoff errors. (From Ref. 99 by permission.)

derive an upper bound on the output error-to-signal ratio which for a second-order-filter example is found to be tighter than that of Sandberg [97].

Using the statistical method established by Kaneko and Liu in their analysis of roundoff error in floating-point digital filters, Weinstein [81] derives expressions for output noise-to-signal ratio. For the first-order single-pole filter with a white input signal, the noise-to-signal ratio is found to be inversely proportional to δ, the distance of the pole from the unit circle. Thus the noise-to-signal ratio increases as the pole moves toward the unit circle but at a lower rate than that found for the fixed-point filter. For a sinusoidal input the results are identical. For second-order filters with white signal input and poles near the unit circle, the direct and canonic [84] forms yield similar results for high gain and are inversely proportional to $\delta \sin^2 \theta$, where $z = re^{\pm j\theta}$ are the pole locations, and thus yield large noise-to-signal ratios at low resonant frequencies. For the coupled form with white signal input the noise-to-signal ratio is inversely proportional to δ only; thus it has improved noise-to-signal ratio at low resonant frequencies.

A simplified approach to the floating-point roundoff noise analysis of Weinstein is presented by Oppenheim and Weinstein in Ref. 79. Floating-point roundoff errors are represented as additive white-noise sources that enter the system following error-generating arithmetic operations in a manner similar to the fixed-point case. It is assumed that for the small errors considered the roundoff noise in a signal following an arithmetic operation is proportional to the signal that would result if there were no roundoff noise. As an example, consider the first-order filter of Fig. 9.5 with $h(n) = a^n$. The noise inputs are then

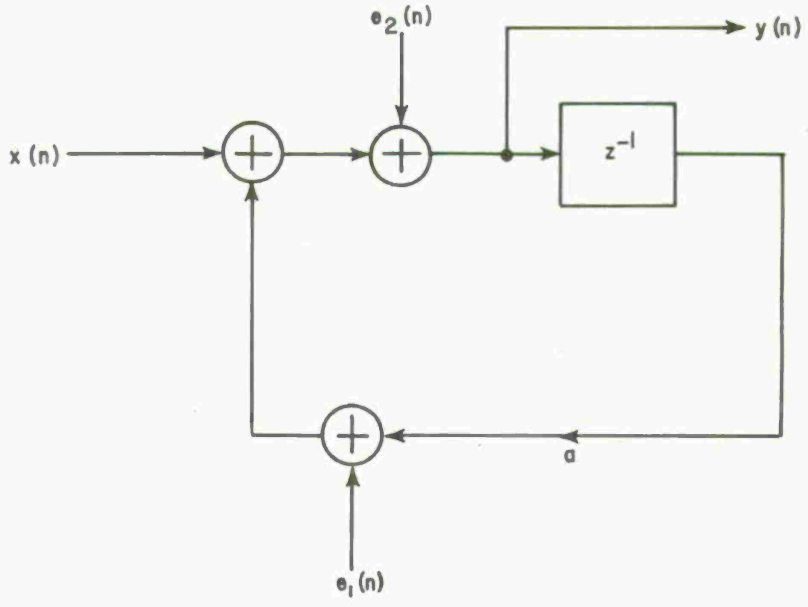


Fig. 9.5—First-order floating-point filter with roundoff noise

$$e_1(n) = ay(n-1)\epsilon_1$$

and

$$e_2(n) = y(n)\epsilon_2,$$

where ϵ_1 and ϵ_2 are the random variables describing the relative error in floating-point multiplication and addition respectively. It is assumed that ϵ_1 and ϵ_2 are uncorrelated from sample to sample, independent of each other and of the signal, and are uniformly distributed between -2^{-b} and 2^{-b} and thus have equal variance $\sigma_\epsilon^2 = 2^{-2b}/3$. If $x(n)$ is a zero-mean white-noise input with variance σ_x^2, and if linear system noise theory is used with

$$\sum_{n=0}^{\infty} h^2(n) = \frac{1}{1-a^2}$$

then

$$\sigma_y^2 = \sigma_x^2 \frac{1}{1-a^2},$$

$$\sigma_{e_1}^2 = a^2\sigma_\epsilon^2\sigma_x^2 \frac{1}{1-a^2},$$

and

$$\sigma_{e_2}^2 = \sigma_\epsilon^2\sigma_x^2 \frac{1}{1-a^2}.$$

Since $e_1(n)$ and $e_2(n)$ are independent, the output noise variance is

$$\sigma_e^2 = (\sigma_{e_1}^2 + \sigma_{e_2}^2)\,\frac{1}{1-a^2} = \sigma_\epsilon^2 \sigma_x^2\,\frac{1+a^2}{(1-a^2)^2}\ .$$

The noise-to-signal ratio is then

$$\frac{\sigma_e^2}{\sigma_y^2} = \sigma_\epsilon^2\,\frac{1+a^2}{1-a^2}\ ,$$

which for the high-gain case, with $a = 1 - \delta$ near unity, becomes

$$\frac{\sigma_e^2}{\sigma_y^2} = \frac{\sigma_\epsilon^2}{\delta}\ .$$

A comparison of fixed-point and floating-point filters for the first- and second-order cases is presented by Weinstein and Oppenheim [100]. The results of the comparison indicate that for a mantissa equal in length to a fixed-point word, floating-point leads to a lower noise-to-signal ratio. The increase in noise-to-signal ratio with increasing filter gain is greater in both cases for fixed-point. If the bits used for the characteristic in floating-point are considered, the noise-to-signal ratio is smaller for floating-point only for high gain.

9.5.3 *Block-Floating-Point Arithmetic*

Analysis of digital filters using block-floating-point arithmetic is presented by Oppenheim [80] for first- and second-order filters. The comparison of fixed-point, floating-point, and block-floating-point is on the basis of the output noise-to-signal ratio when the input is white noise with uniform amplitude distribution. The analysis does not account for the additional bits required for the characteristic in floating-point or block-floating-point. The results indicate that the noise-to-signal ratio for block-floating-point is generally between that of fixed-point and floating-point for higher gain and greater than either for low gain. The increase in noise-to-signal ratio with increasing gain is approximately the same for floating-point and block-floating-point.

9.6 Limit Cycles

Under certain conditions recursive digital filters can possess a type of instability, known as limit cycles, that cannot be described by normal linear system analysis. One type of limit cycle, known as the deadband effect, is due to rounding of multiplication operations and occurs for constant input, although most analyses of the associated effects assume zero input. The other form of limit cycle, known as overflow oscillations, is due to adder overflow and is usually of large amplitude and highly undesirable.

9.6.1 *Deadband Effect*

The limit cycles that occur from multiplier rounding were first noted by Blackman [101], who referred to the amplitude intervals within which these limit cycles occur as *deadbands*. Blackman analyzed only first-order limit-cycle effects, which lead to constant outputs or dc limit cycles. These first-order limit cycles are due to rounding of products such as in the difference equation

$$y(n) = x(n) - \alpha y(n-1),$$

where for the largest integer $k \leqslant 0.5/(1 - |\alpha|)$ a limit cycle in the range $[-k, k]$ can occur in which the product $\alpha y(n-1)$ is rounded to $\pm y(n-1)$; thus an effective pole on the unit circle occurs. The integers k are the maximum amplitudes of a limit cycle in units of the quantization steps 2^{-b} corresponding to roundoff to b bits [11]. For α negative the limit cycle has a constant magnitude and sign, and for α positive the sign alternates. Jackson [102] analyzed these effects for second-order digital filters described by the difference equation

$$y(n) = x(n) - \beta_1 y(n-1) - \beta_2 y(n-2)$$

with complex-conjugate poles. Due to rounding of the multiplication $\beta_2 y(n-2)$ for the largest integer k satisfying $k \leqslant 0.5/(1 - \beta_2)$, $0 < \beta_2 < 1$, there results an effective complex-conjugate pole pair on the unit circle and sinusoidal limit cycles in the range $[-k, k]$. The frequency of the oscillation is determined by β_1. First-order or dc limit cycles can also occur in digital filters of arbitrary order due to real effective poles. All deadband subregions for second-order filters are plotted in Fig. 9.6 and labeled with the k values. No limit cycles occur in the crosshatched region. An upper bound on the rms value of limit cycles is developed by Sandberg and Kaiser [103]. Blackman [101] proposed the use of *dither*, a small noise presented at the input of a system, to overcome the deadband effects.

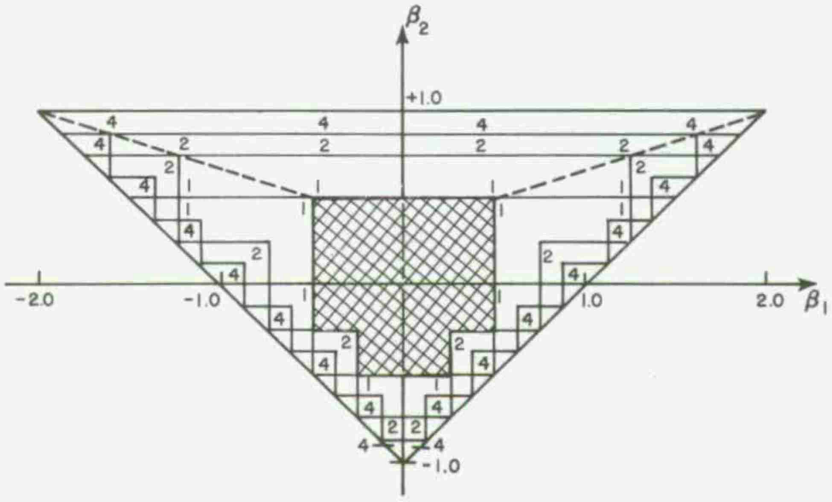


Fig. 9.6 — Deadband subregions for a second-order filter.
(From Ref. 102 by permission.)

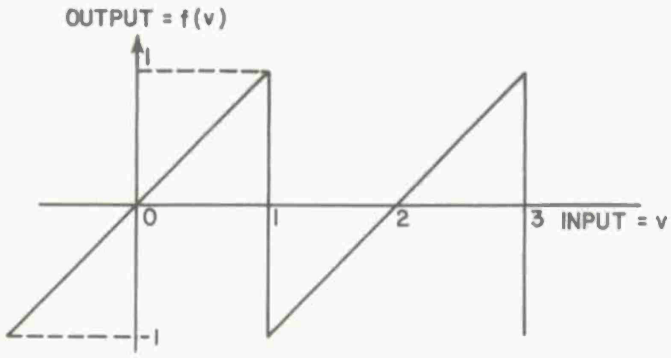


Fig. 9.7 — Instantaneous transfer function for an accumulator with overflow. (Reprinted with permission from *The Bell System Technical Journal*, Copyright 1969, The American Telephone and Telegraph Company.)

The existence of deadband-type limit cycles, including those of large amplitude, in floating-point digital filters was confirmed by Kaneko [104]. It was previously assumed that limit cycles in floating-point filters did not exist or were of negligibly small amplitude. Kaneko analyzes floating-point limit cycles and determines conditions for their existence.

9.6.2 *Overflow Oscillations*

Another type of self-sustained oscillation can occur in digital filters due to overflow in 1's complement and 2's complement addition operations and has been analyzed by Ebert, Mazo, and Taylor [105]. This is due to an instantaneous accumulator transfer function with overflow, as shown in Fig. 9.7. For the second-order filter described by

$$y(n) - ay(n-1) - by(n-2) = 0$$

overflow oscillations can occur when $|a| + |b| \geqslant 1$. These oscillations can be prevented by restricting the values of a and b, thus limiting design capability. The best solution is to use saturation arithmetic, indicated by the transfer function of Fig. 9.8, which limits the results of an addition to a maximum magnitude of 1 and is shown to always lead to stable behavior.

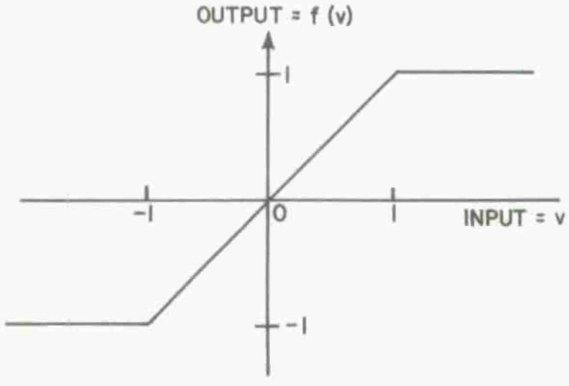


Fig. 9.8 — Instantaneous transfer function for saturation arithmetic. (Reprinted with permission from *The Bell System Technical Journal*, Copyright 1969, The American Telephone and Telegraph Company.)

9.7 FFT Quantization Effects

To complete the present discussion, the analyses that have been reported in the literature with respect to quantization errors in the computation of the FFT will be considered for both fixed- and floating-point implementations. Many of the quantization-effect concepts presented earlier for digital filters are applicable to the FFT. This includes number representations, statistical modeling of parameter quantization errors as noise sources, and the application of linear system noise theory to determine system output due to such sources.

The analyses to be considered generally involve N-point radix-2 FFT algorithms of the decimation-in-time or decimation-in-frequency form. As discussed earlier, the FFT computation is computed in $m = \log_2 N$ iterations on an array of N complex samples. The initial or 0th array represents the N points of the time sequence, and the mth array is the N points of the transform sequence. In general either the input or output sequence is in bit reversed order. The array corresponding to the $(i + 1)$th iteration is computed from the values in the ith iteration. Each element in the $(i + 1)$th array is determined from two elements in the ith array by a computation referred to as a *butterfly* due to its signal-flow-graph representation. There are $N/2$ butterfly computations required in each of the m iterations.

9.7.1 *Fixed-Point Implementations*

Using the decimation-in-time butterfly computations, Welch [106] analyzes the errors in the FFT for fixed-point b-bit-plus-sign arithmetic. He shows that the magnitudes of the complex numbers increase in an rms sense by the factor $\sqrt{2}$ in each iteration and that the maximum modulus is nondecreasing. Scaling to prevent overflow can be accomplished in several ways. If the magnitudes in the initial sequence are less than 1/2 proper scaling can be accomplished by shifting right one bit in each iteration or by checking the magnitudes in one iteration and shifting right one bit if necessary in the next iteration. Another scaling procedure used by Welch in his analysis involves scaling the initial real and imaginary components to a maximum unity magnitude and shifting the entire sequence right one bit whenever an overflow occurs in an array. This includes new results as well as those yet to be processed. In effect this is essentially a block-floating-point implementation. Computing an upper bound on the error due to rounding and rescaling involves the assumption of a rescaling being required for each iteration. This however corresponds to the method of shifting right one bit in each iteration. For real and imaginary parts represented as a sign bit plus b magnitude bits, the roundoff error variance is $\sigma_1^2 = 2^{-2b}/12 = \Delta^2$. When a shift occurs, a bit is lost. If the bit is 0, there is no error; if it is 1 there is an error of $\pm 2^{-b}$ depending on the sign of the number. The variance of this error is $\sigma_2^2 = 2^{-2b}/2 = 6\Delta^2$. Letting K equal the average modulus squared of the initial array, the resulting bound of the ratio of the rms error to rms result for large m is

$$\frac{\text{rms (error)}}{\text{rms (result)}} \approx \frac{2^{(m+3)/2}\Delta}{\sqrt{\frac{K}{2}}} = 4\Delta\sqrt{\frac{2N}{K}}\,.$$

Thus the upper bound increases as $\sqrt{N}$ or 1/2 bit per stage. The lower bound is

$$\frac{\text{rms (error)}}{\text{rms (result)}} \approx (m - 2.5)^{1/2}\,(0.3)2^{-b}$$

and increases as $(1/2)\log_2 N$. Experimentation by Welch indicates a 4/3 factor for the upper bound as well as a higher bound for truncation in place of rounding.

9.7.2 *Floating-Point Implementations*

The roundoff error in floating-point implementations of the FFT was first considered by Gentleman and Sande [59]. They compute worst-case bounds on noise-to-signal ratio for both the FFT and DFT. This ratio is represented by the ratio of the Euclidean norm of the output error sequence to that of the output signal. The Euclidean norm is the square root of the sum of the squares of the sequence values. The bound on noise-to-signal ratio for the FFT is derived for N highly composite. For $N = 2^m$ the bound is $1.06m2^{(m/2)-b+3}$, where b is the number of bits in the mantissa. For the direct or DFT computation the bound is $1.06\ 2^{2m-b-(3/2)}$. The noise-to-signal ratio for the FFT is then $m/2^{(3/2)(m-1)}$ times that for the direct computation by the DFT. This factor is less than 1 for $m > 1$ and decreases rapidly with increasing m.

The effects of arithmetic roundoff in floating-point implementations of the FFT were first analyzed statistically by Weinstein [107]. His results are generally valid for both decimation-in-time and decimation-in-frequency algorithms and involve an analysis of the basic butterfly equations. The floating-point noise models discussed earlier for digital filters are used, and the input is represented as white noise. The noise enters the signal flow graph representation of the FFT at the various points where arithmetic error is introduced and passes through the system in the same manner as signal. Due to the regular repeatibility of the computations from array to array, the signal propagation to the ith array can be described by

$$E[|X_i(p)|^2] = 2^i E[|X_0(p)|^2], \qquad i = 0, 1, \ldots, N-1,$$

and the noise variance at the output due to roundoff noise injected at the ith iteration is

$$E[|e_m(p)|^2] = 2^{m-i-1} E[|e_i(p)|^2], \qquad i = 0, 1, \ldots, N-1,$$

where X_i refers to the ith signal value, e_i to the ith noise value, and E to the expected value. The resulting output noise-to-signal ratio is found to be

$$\frac{\sigma_e^2}{\sigma_{X_m}^2} = 2\sigma_\epsilon^2 m,$$

where σ_ϵ^2 is the variance $2^{-2b}/3$ of the relative floating-point arithmetic error. The linear dependence on $m = \log_2 N$ should be noted. Considering a decimation-in-time algorithm, the butterfly computations involving powers of W_N equal to 1 or j are taken into account and lead to

$$\frac{\sigma_e^2}{\sigma_{X_m}^2} = 2\sigma_e^2\left[m - \frac{3}{2} + \left(\frac{1}{2}\right)^{m-1}\right].$$

For m moderately large the results are essentially the same. Weinstein proposes a tree-like summation of the products in the DFT that involves more memory and indexing than cummulative DFT summing but results in an accuracy essentially that of the FFT.

A more general statistical analysis is performed by Kaneko and Liu [108]. Their approach uses the statistical error models and methods of their earlier floating-point error analysis for digital filters [91, 98, 99]. Consideration of the butterfly equations for the decimation-in-frequency form of the FFT is the basis of the analysis. Kaneko and Liu take into consideration the absence of multiplication roundoff error for i equal to $m - 1$ and $m - 2$, where all the weighting coefficients are ± 1 or $\pm j$. For $i \leqslant m - 3$ such weighting coefficients are not taken into account, making the results somewhat pessimistic. Expressions are derived for the mean-square error $E[|e(p)|^2]$ due to roundoff as well as truncation for the pth Fourier coefficient $X_m(p)$ of the resulting FFT sequence. The mean-square error due to roundoff is proportional to $2^{-2b}/3$, where b is the number of bits in the mantissa and is a function of p and the Fourier coefficients that would result from an errorless computation. For truncation arithmetic the mean-square error is equal to that due to rounding plus an additional term dependent on p and proportional to 2^{-2b} and on the magnitude squared of the errorless result for $X_m(p)$. The total relative mean-square error is found to be bounded as

$$m\,\frac{2^{-2b}}{3} \leqslant \frac{\sum_{p=0}^{N-1} E[|e(p)|^2]}{\sum_{p=0}^{N-1} |X_m(p)|^2} < 3m\,\frac{2^{-2b}}{3}$$

for rounding and

$$m^2 2^{-2b} + m\,\frac{2^{-2b}}{3} \leqslant \frac{\sum_{p=0}^{N-1} E[|e(p)|^2]}{\sum_{p=0}^{N-1} |X_m(p)|^2} < 9m^2 2^{-2b} + 3m\,\frac{2^{-2b}}{3}$$

for truncation. The autocorrelation function corresponding to a random input sequence is used to express the corresponding mean-square error. For the case of white-noise input data the noise-to-signal ratio is found to be $2(m - 1)2^{-2b}/3$, which is similar to the result of Weinstein [107].

Kaneko and Liu also consider the effects of quantization of the input data and the weighting coefficients. The input quantization is considered as an error term in the representation of the input data, similar to the error due to the roundoff of a computation. The quantization of the input data to a mantissa of b' bits results in an additional

term in the mean-square error of

$$\frac{2^{-2b'}}{3}\, 2^{-m} \sum_{k=0}^{N-1} |X_m(k)|^2$$

for rounding and 4 times this quantity for truncation. Quantization errors in the weighting coefficients can be treated in a manner similar to roundoff errors and results in a modification in the expression for the mean-square error for roundoff as well as for truncation. Although the same weighting coefficients are used at various points in the computation, independence of the error is assumed.

The roundoff error analyses of Kaneko and Liu [108] and Weinstein [107] have been extended by Chan and Jury [109] to multidimensional FFT's as well as to generalized discrete transforms. These generalized discrete transforms include the BIFORE transform (BT) [110] and complex BIFORE transform (CBT), [111]. In their analysis Chan and Jury modify the error-to-signal ratio derived by Kaneko and Liu to account for all nonroundoff multiplications by weighting coefficients equal to ± 1 and $\pm j$. The results coincide with that derived by Weinstein for decimation in time, indicating an equivalence in error performance with decimation in frequency. The results of the error analysis for the one-dimensional FFT are extended to one-dimensional generalized transforms in order to derive expressions for the mean-square error and the noise-to-signal ratio for white-noise input. A noise-to-signal ratio analysis for the two-dimensional FFT is performed and extended to derive noise-to-signal expressions for L-dimensional FFT's and generalized transforms.

10. ACKNOWLEDGMENTS

The author expresses his gratitude to NRL which, through its Edison Memorial Fellowship, continues to support his doctoral research efforts. He also acknowledges the encouragement of Mr. R. E. Ellis, Head of the Special Projects Organization, NRL, and the guidance and advice of Prof. N. Kyriakopoulos of the George Washington University, School of Engineering and Applied Science. A special "thank you" goes to Mrs. Anita Latham who typed and otherwise labored with the manuscript.

11. REFERENCES

1. E. Polak and E. Wong, *Notes For A First Course On Linear Systems*, Van Nostrand Reinhold Company, New York, 1970.
2. C.A. Desoer, *Notes For A Second Course On Linear Systems*, Van Nostrand Reinhold Company, New York 1970.
3. C.T. Chen, *Introduction to Linear System Theory*, Holt, Rinehart, and Winston, New York, 1970.
4. M.E. Van Valkenburg, *Introduction to Modern Network Synthesis*, John Wiley, New York, 1967.

5. J.R. Ragazzini and G.F. Franklin, *Sampled Data Control Systems*, McGraw-Hill, New York, 1958.

6. L.R. Rabiner, J.W. Cooley, H.D. Helms, L.B. Jackson, J.F. Kaiser, C.M. Rader, R.W. Schafer, K. Steiglitz, and C.J. Weinstein, "Terminology in Digital Signal Processing," *I.E.E.E. Trans. Audio Electroacoust.* **AU-20**, 322-337 (Dec. 1972).

7. K. Steiglitz, "The Equivalence of Digital and Analog Signal Processing," *Inform. Contr.* 8, 455-467 (1965).

8. H. Schmid, "Electronic Analog/Digital Conversions," Van Nostrand Reinhold Company, New York, 1970.

9. J.W. Cooley and J.W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics of Computation*, **19**, 297-301 (Apr. 1965).

10. A.V. Oppenheim and D.H. Johnson, "Discrete Representation of Signals," *Proceedings of the I.E.E.E.*, Vol. 60, No. 6, pp. 681-691, June 1972.

11. A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, N.J., 1975.

12. B. Gold and C.M. Rader, "Digital Processing of Signals," McGraw-Hill Book Co., Inc., New York, 1968.

13. P.S. Laplace, *Ouevres Completes*, 1779.

14. DeMoivre, *Miscellanes Analytica de Seriebus et Quatratoris*, London, 1730.

15. E.I. Jury, *Theory and Application of the z-Transform Method*, Wiley, New York, 1964.

16. R.V. Churchill, *Complex Variables and Applications*, McGraw-Hill., New York, 1964.

17. L.M. Leibowitz, "Comparative Analysis of the use of Dynamic and Static Shift Registers in Digital Signal Processors," NRL Report 7482, Dec. 26, 1972.

18. R.H. Barker, "The Pulse Transfer Function and its Application to Sampling Servo Systems, *Proc. I.E.E.* **99**, pt. IV, monograph 43, July 15, 1952.

19. A. Papoulis, *The Fourier Integral and Its Applications*, McGraw-Hill, New York, 1962.

20. E.O. Brigham, *The Fast Fourier Transform*, Prentice-Hall, Englewood Cliffs, N.J., 1974.

21. B.M. Oliver, J.R. Pierce, and C.E. Shannon, "The Philosophy of PCM (Pulse Code Modulation)," *Proc. IRE* 36 (No. 11), 1324-1331 (Nov. 1948).

22. A.J. Gibbs, "The Design of Digital Filters," *Austrailian Telecommunications Research Journal* 4, 29-34 (May 1970).

23. S.J. Mason and H.J. Zimmermann, *Electronic Circuits, Signals, and Systems*, Wiley, New York, 1960.

24. Y. Chow and E. Cassignol, *Linear Signal-Flow Graphs and Applications*, Wiley, New York, 1962.

25. A. Fettweis, "A General Theorem for Signal-Flow Networks, with Applications," *Archiv fur Elektronik und Ubertragungstechnik* **25**, 557-561 (1971).

26. R.E. Crochiere, "Analysis and Synthesis of Digital Filter Structures," Ph.D. Thesis, M.I.T., Department of Electrical Engineering, 1974.

27. L.B. Jackson, J.F. Kaiser, and H.S. McDonald, "An Approach to The Implementation of Digital Filters," *IEEE Transactions on Audio and Electroacoustics* AU-16, 413-421, (Sept. 1968).

28. J.F. Kaiser, "Some Practical Considerations in the Realization of Linear Digital Filters," *Proc. 3rd. Annual Allerton Conf. on Circuit and System Theory*, pp. 621-633, 1965.

29. L.R. Rabiner, "Techniques for Designing Finite-Duration Impulse-Response Digital Filters," *IEEE Trans. Commun. Technol.* COM-19, 188-195 (Apr. 1971).

30. C.M. Rader and B. Gold, "Digital Filter Design Techniques in the Frequency Domain," *Proc. IEEE* 55, 149-171 (Feb. 1967).

31. B. Gold and K.L. Jordan, Jr., "A Note on Digital Filter Synthesis," *Proceedings IEEE* 56, 1717-1718 (Oct. 1968).

32. J.F. Kaiser, "Digital Filters," Chap 7 in *System Analysis by Digital Computers*, F.F. Kuo and J.F. Kaiser, editors, Wiley, New York, 1966.

33. L.R. Rabiner, B. Gold, and C.A. McGonegal, "An Approach to the Approximation Problem for Nonrecursive Digital Filters," *IEEE Trans. Audio and Electroacoustics* AU-18, 83-106 (June 1970).

34. L.B. Jackson, "On the Interaction of Roundoff Noise and Dynamic Range in Digital Filters," *Bell System Technical Journal*, 49, 159-184 (Feb. 1970).

35. S.K. Mitra and R.J. Sherwood, "Canonic Realizations of Digital Filters using the Continued Fraction Expansion," *IEEE Trans. Audio and Electroacoustics* AU-20, pp. 185-194 (Aug. 1972).

36. S.K. Mitra and A.D. Sagar, "Additional Canonic Realizations of Digital Filters Using the Continued-Fraction Expansion," *IEEE Transactions on Circuits and Systems* CAS-21, 135-136 (Jan. 1974).

37. S.K. Mitra and R.J. Sherwood, "Digital Ladder Networks," *IEEE Transactions on Audio and Electroacoustics* AU-21, 30-36 (Feb. 1973).

38. S.K. Mitra, D.C. Huey, and R.J. Sherwood, "New Methods of Digital Ladder Realization," *IEEE Transactions on Audio and Electroacoustics* AU-21, 485-491 (Dec. 1973).

39. S.Y. Hwang, "Realization of Canonical Digital Networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing* ASSP-22, 27-39 (Feb. 1974).

40. E.A. Guillemin, *Synthesis of Passive Networks*, Wiley, New York, 1957.

41. J.E. Storer, *Passive Network Synthesis*, McGraw-Hill, New York, 1957.

42. L. Weinberg, *Network Analysis and Synthesis*, McGraw-Hill, New York, 1962.

43. J.R. Kaiser, "Design Methods for Sampled Data Filters," Proceedings of the 1st Annual Allerton Conference on Circuit and System Theory, pp. 221-236, 1963.

44. R.M. Golden and J.F. Kaiser, "Design of Wideband Sampled-Data Filters," *Bell System Technical Journal* 43, 1533-1546 (July 1964).

45. J.E. Gibson, *Nonlinear Automatic Control*, McGraw-Hill, New York, 1963.

46. K. Steiglitz, "Computer-Aided Design of Recursive Digital Filters," *IEEE Transactions on Audio and Electroacoustics*, AU-18, 123-129 (June 1970).

47. R. Fletcher and M.J.D. Powell, "A Rapidly Convergent Descent Method for Minimization," *Comput. J.*, 6, 163-168 (Mar. 1963).

48. M. Suk and S.K. Mitra, "Computer-Aided Design of Digital Filters with Finite Word Lengths," *IEEE Transactions on Audio and Electroacoustics* AU-20, 356-362 (Dec. 1972).

49. R.B. Blackman and J.W. Tukey, "The Measurement of Power Spectra," Dover Publications, Inc., New York, 1958.

50. O. Herrmann and H.W. Schuessler, "On the Design of Selective Nonrecursive Digital Filters," *IEEE Arden House Workshop*, Harriman, N.Y., Jan. 1970.

51. H.D. Helms, "Digital Filters with Equiripple or Minimax Responses," *IEEE Transactions on Audio and Electroacoustics* AU-19, 87-94 (Mar. 1971).

52. E. Hofstetter, A. Oppenheim, and J. Siegel, "A New Technique for the Design of Nonrecursive Digital Filters," Proceedings of the 5th Annual Princeton Conference on Information Science Systems, pp. 64-72, 1971.

53. A.G. Constantinides, "Spectral Transformations for Digital Filters," *Proceedings IEE* 117, 1585-1590 (Aug. 1970).

54. C.S. Burrus and T.W. Parks, "Time Domain Design of Recursive Digital Filters," *IEEE Transactions on Audio and Electroacoustics* AU-18, 137-141 (June, 1970).

55. F. Brophy and A.C. Salazar, "Recursive Digital Filter Synthesis in the Time Domain," *IEEE Transactions on Acoustics, Speech, and Signal Processing* ASSP-22, 45-55 (Feb. 1974).

56. W.T. Cochran, et al., "What is the Fast Fourier Transform?" *IEEE Transactions on Audio and Electroacoustics* AU-15, 45-55 (June 1967).

57. G.D. Bergland, "A Guided Tour of the Fast Fourier Transform," *IEEE Spectrum* 6, 41-52, July 1969.

58. J.W. Cooley, P.A.W. Lewis, and P.D. Welch, "The Finite Fourier Transform," *IEEE Transactions on Audio and Electroacoustics* AU-17, 77-85 (June 1969).

59. W.M. Gentleman and G. Sande, "Fast Fourier Transforms—For Fun and Profit," 1966 Fall Joint Computer Conference, *AFIPS Proceedings*, Vol. 29, pp. 563-578, Spartan Books, Washington, D.C.

60. J.W. Cooley, P.A.W. Lewis, and P.D. Welch, "The Fast Fourier Transform Algorithm: Programming Considerations in the Calculation of Sine, Cosine, and LaPlace Transforms," *Journal of Sound and Vibration* 12, 315-337 (July 1970).

61. J.W. Cooley, P.A.W. Lewis, and P.D. Welch, "Historical Notes on the Fast Fourier Transform," *IEEE Transactions on Audio and Electroacoustics* AU-15, 76-79 (June 1967).

62. C. Runge, *Zeit. fur Math. und Physik* 48, 443 (1903).

63. G.C. Danielson and C. Lanczos, "Some Improvements in Practical Fourier Analysis and their Application to X-ray Scattering from Liquids," *Journal of the Franklin Institute* 233, 365-380 and 435-452, (May 1942).

64. F. Yates, "The Design and Analysis of Factorial Experiments," Commonwealth Agriculture Bureaux, Farnam Royal, Bucks, England, 1937.

65. I.J. Good, "The Interaction Algorithm and Practical Fourier Analysis," *Journal of the Royal Statistics Society, ser. B*, 20, 361-372 (1958) *Addendum*, 22, 372-375 (1960).

66. E.O. Brigham and R.E. Morrow, "The Fast Fourier Transform," *IEEE Spectrum* 4 (No. 12), 63-70 (Dec. 1967).

67. G.D. Bergland, "The Fast Fourier Transform Recursive Equations For Arbitrary Length Records," *Math. Comput.* 21, 236-238 (Apr. 1967).

68. L.I. Bluestein, "A Linear Filtering Approach to the Computation of the Discrete Fourier Transform." 1968 NEREM Record, pp. 218-219.

69. L.I. Bluestein, "A Linear Filtering Approach to the Computation of Discrete Fourier Transform," *IEEE Transactions on Audio and Electroacoustics* AU-18, 451-455 (Dec. 1970).

70. L.R. Rabiner, R.W. Schafer, and C.M. Rader, "The Chirp Z-Transform Algorithm," *IEEE Transactions on Audio and Electroacoustics* AU 17, 86-92 (June 1969).

71. C.M. Rader, "Discrete Fourier Transforms When the Number of Data Samples is Prime," *Proceedings IEEE* 56, 1107-1108 (June 1968).

72. J.W. Cooley, P.A.W. Lewis, and P.D. Welch, "Application of the Fast Fourier Transform to Computation of Fourier Integrals, Fourier Series, and Convolution Integrals," *IEEE Transactions on Audio and Electroacoustics* AU-15, 79-84 (June 1967).

73. T.G. Stockham, Jr., "High-speed Convolution and Correlation with Applications to Digital Filtering," Chap. 7 in *Digital Processing of Signals*, B. Gold and C.M. Rader, McGraw-Hill, New York, 1969.

74. T.G. Stockham, Jr., "High-speed Convolution and Correlation," *AFIPS Proceedings*, 1966 Spring Joint Computer Conference, Vol. 28, pp. 229-233, Spartan, Washington, D.C., 1966.

75. H.D. Helms, "Fast Fourier Transform Method of Computing Difference Equations and Simulating Filters," *IEEE Transactions on Audio and Electroacoustics* AU-15, 85-90 (June 1967).

76. C.M. Rader, "An Improved Algorithm for High Speed Autocorrelation with Applications to Spectral Estimation," *IEEE Transactions on Audio and Electroacoustics* AU-18, 439-441 (Dec. 1970). I 374

77. I. Flores, "The Logic of Computer Arithmetic." Prentice-Hall, Englewood Cliffs, N.J., 1963.

78. A.M. Abd-alla and A.C. Meltzer, *Principles of Digital Computer Design*, Vol. 1, Prentice Hall, Englewood Cliffs, N.J., 1975.

79. A.V. Oppenheim and C.J. Weinstein, "Effects of Finite Register Length in Digital Filtering and the Fast Fourier Transform," *Proceedings IEEE* 60, 957-976 (Aug. 1972).

80. A.V. Oppenheim, "Realization of Digital Filters Using Block Floating-Point Arithmetic," *IEEE Transactions on Audio and Electroacoustics* AU-18, 130-136 (June 1970).

81. C.J. Weinstein, "Quantization Effects in Digital Filters," M.I.T. Lincoln Lab. Tech. Rept. 468, ASTIA Doc. DDC AD-706 862, Nov. 21, 1969.

82. W.R. Bennett, "Spectra of Quantized Signals," *Bell System Technical Journal* 27, 446-472 (July 1948).

83. B. Widrow, "Statistical Analysis of Amplitude-Quantized Sampled-Data Systems," *AIEE Transactions on Applications and Industry* 79 (II), 555-568 (Jan. 1961).

84. B. Gold and C.M. Rader, "Effects of Quantization Noise in Digital Filters," Proceeding of the Spring Joint Computer Conference, AFIPS Conference Proceedings, Vol. 28, pp. 213-219, 1966.

85. J.B. Knowles and R. Edwards, "Effect of a Finite-Word-Length Computer in a Sampled-Data Feedback System," *Proceedings IEE (London)* 112, 1197-1207 (June 1965).

86. C.M. Rader and B. Gold, "Effects of Parameter Quantization on the Poles of a Digital Filter," *Proceedings IEEE (Letters)* 55, 688-689 (May 1967).

87. J.B. Knowles, and E.M. Olcayto, "Coefficient Accuracy and Digital Filter Response," *IEEE Transactions on Circuit Theory* CT-15, 31-41, (Mar. 1968).

88. S.K. Mitra and R.J. Sherwood, "Estimation of Pole-Zero Displacements of a Digital Filter Due to Coefficient Quantization," *IEEE Transactions on Circuits and Systems* CAS-21, 116-124 (Jan. 1974).

89. E. Avenhaus, and W. Schussler, "On the Approximation Problem in the Design of Digital Filters with Limited Wordlength." *Archiv fur Elektronik und Ubertragungstechnik* 24, 571-572 (1970).

90. E. Avenhaus, "On the Design of Digital Filters with Coefficients of Limited Word Length," *IEEE Transactions on Audio and Electroacoustics* AU-20, 206-212 (Aug. 1972).

91. T. Kaneko and B. Liu, "Effect of Coefficient Rounding in Floating-Point Digital Filters," *IEEE Transactions on Aerospace and Electronic Systems* AES-7, 995-1003 (Sept. 1971).

92. K.J. Åström, E.I. Jury, and R.G. Agniel, "A Numerical Method for the Evaluation of Complex Integrals," *IEEE Transactions on Automatic Control (Short Papers)* AC-15, 468-471 (Aug. 1970).

I3535
→ Noise.

93. S.K. Mitra, K. Hirano, and H. Sakaguchi, "A Simple Method of Computing the Input Quantization and Multiplication Roundoff Errors in a Digital Filter," *IEEE Transactions on Acoustics, Speech and Signal Processing* ASSP-22, 326-329 (Oct. 1974).

94. L.B. Jackson, "Roundoff-Noise Analysis for Fixed-Point Digital Filters Realized in Cascade or Parallel Form," *IEEE Transactions on Audio and Electroacoustics* AU-18, 107-122 (June 1970).

95. D.S.K. Chan and L.R. Rabiner, "Theory of Roundoff Noise in Cascade Realizations of Finite Impulse Response Digital Filters," *Bell System Technical Journal* 52, 329-345 (Mar. 1973).

96. D.S.K. Chan and L.R. Rabiner, "Analysis of Quantization Errors in the Direct Form for Finite Impulse Response Digital Filters," *IEEE Transactions on Audio and Electroacoustics* AU-21, 354-366 (Aug. 1973).

97. I.W. Sandberg, "Floating-Point-Roundoff Accumulation in Digital-Filter Realizations," *Bell System Technical Journal* 46, 1775-1791 (Oct. 1967).

98. T. Kaneko and B. Liu, "Round-off Error of Floating-Point Digital Filters," Proceedings of the Sixth Annual Allerton Conference on Circuit and System Theory, pp. 219-227, Oct. 2-4, 1968.

99. B. Liu and T. Kaneko, "Error Analysis of Digital Filters Realized with Floating Point Arithmetic," *Proceedings IEEE* 57, 1735-1747 (Oct. 1969).

100. C. Weinstein and A.V. Oppenheim, "A Comparison of Roundoff Noise in Floating Point and Fixed Point Digital Filter Realizations," *Proceedings of the IEEE (Letters)* 57, 1181-1183 (June 1969).

101. R.B. Blackman, *Linear Data-Smoothing and Prediction in Theory and Practice*, Addison-Wesley, Reading, Mass., 1965, pp. 75-79.

102. L.B. Jackson, "An Analysis of Limit Cycles Due to Multiplication Rounding in Recursive Digital (Sub) Filters," Proceedings 7th Annual Allerton Conference on Circuit System Theory, pp. 69-78, 1969.

103. I.W. Sandberg and J.F. Kaiser, "A Bound on Limit Cycles in Fixed-Point Implementations of Digital Filters," *IEEE Transactions on Audio and Electroacoustics* AU-20, 110-112 (June 1972).

104. T. Kaneko, "Limit-Cycle Oscillations in Floating-Point Digital Filters," *IEEE Transactions on Audio and Electroacoustics* AU-21, 100-106 (Apr. 1973).

105. P.M. Ebert, J.E. Mazo, and M.G. Taylor, "Overflow Oscillations in Digital Filters," *Bell System Technical Journal* 48, 2999-3020 (Nov. 1969).

106. P.D. Welch, "A Fixed-Point Fast Fourier Transform Error Analysis," *IEEE Transactions on Audio and Electroacoustics* AU-17, 151-157 (June 1969).

107. C.J. Weinstein, "Roundoff Noise in Floating Point Fast Fourier Transform Computation," *IEEE Transactions on Audio and Electroacoustics* AU-17, 209-215 (Sept. 1969).

108. T. Kaneko and B. Liu, "Accumulation of Round-off Error in Fast Fourier Transforms," *Journal of the Association for Computing Machinery* 17, 637-654 (Oct. 1970).

109. O.W.C. Chan and E.I. Jury, "Roundoff Error in Multidimensional Generalized Discrete Transforms," *IEEE Transactions on Circuits and Systems* CAS-21, 100-108 (Jan. 1974).

110. N. Ahmed, K.R. Rao, and A.L. Abdussattar, "BIFORE or Hadamard Transform," *IEEE Transactions on Audio and Electroacoustics* AU-19, 225-234 (Sept. 1971).

111. N. Ahmed and K.R. Rao, "Complex BIFORE Transform," *International Journal of Systems Science* 2, 149-162 (Sept. 1971).

DEPARTMENT OF THE NAVY

NAVAL RESEARCH LABORATORY
Washington, D.C. 20375

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300

POSTAGE AND FEES PAID
DEPARTMENT OF THE NAVY
DoD-316

SUPERINTENDENT
U.S. NAVAL POSTGRADUATE SCHOOL
ATTN: TECHNICAL LIBRARY
MONTEREY, CA 93940

0212