Terminology in Digital Signal Processing

LAWRENCE R. RABINER, JAMES W. COOLEY, HOWARD D. HELMS, LELAND B. JACKSON, JAMES F. KAISER, CHARLES M. RADER, RONALD W. SCHAFER, KENNETH STEIGLITZ, and CLIFFORD J. WEINSTEIN

Abstract—The committee on Digital Signal Processing of the IEEE Group on Audio and Electroacoustics has undertaken the project of recommending terminology for use in papers and texts on digital signal processing. The reasons for this project are twofold. First, the meanings of many terms that are commonly used differ from one author to another. Second, there are many terms that one would like to have defined for which no standard term currently exists. It is the purpose of this paper to propose terminology which we feel is self-consistent, and which is in reasonably good agreement with current practices. An alphabetic index of terms is included at the end of the paper.

Introduction

As an aid to classifying the different types of terms to be defined, we have placed each term in one of the following groups:

1) Introductory Terms—General Definitions
2) Discrete Systems—Block Diagram Terminology
3) Relations Between Discrete and Continuous Signals
4) Theory and Design of Digital Filters
5) Finite Word Length Effects—A/D, D/A Conversion
6) Discrete Fourier Transforms and the FFT
7) Discrete Convolution and Spectrum Analysis.

In the above mentioned sections of this paper we will be discussing terminology related to the processing of one-dimensional signals. For convenience, we will assume that this dimension is time—although the definitions apply equally well to any single dimension.

I. Introductory Terms—General Definitions

1) In discussing waveform processing problems, the distinctions analog versus digital and continuous time versus discrete time are often made. Although they are often used interchangeably, different meanings should be attributed to the two sets of terms.

2) The term *analog* generally describes a waveform that is continuous in time (or any other appropriate independent variable) and that belongs to a class that can take on a continuous range of amplitude values. Examples of *analog waveforms* or *analog signals* are those derived from acoustic sources. Such signals are represented mathematically as functions of a continuous variable. The functions sin (ωt) and the step function $au_{-1}(t)$ are examples of common mathematical functions that could describe "analog signals." The use of the term "analog" in this context appears to stem from the field of analog computation, where a current or voltage waveform serves as a physical analog of some variable in a differential equation.

3) The term *continuous time* implies that only the independent variable necessarily takes on a continuous range of values. In theory the amplitude may, but need not, be restricted to a finite or countable infinite set of values (i.e., the amplitude may be quantized). Therefore, analog waveforms are continuous-time waveforms with continuous amplitude. In practice, however, "continuous-time waveforms" and "analog waveforms" are equivalent. Since most signal processing problems have nothing to do with analogs as such, the use of the term analog waveform is often ambiguous at the least and may in fact be misleading. Thus, the term continuous-time waveform is preferable.

4) *Discrete time* implies that time (the independent variable) is quantized. That is, discrete-time signals are defined only for discrete values of the independent variable. Such signals are represented mathematically as *sequences* of numbers. Those discrete-time signals that take on a continuum of values are referred to as *sampled-data* signals.

5) The term *digital* implies that both time and amplitude are quantized. Thus a *digital system* is one in which signals are represented as sequences of numbers which take on only a finite set of values. Thus one uses *digital* when discussing actual physical realizations (as hardware or programs) of discrete-time signal processing systems, whereas the term *discrete time* is a better modifier when considering mathematical abstractions of such systems in which the effects of amplitude quantization are ignored. A *digital signal* or *digital waveform* is a sequence produced, for example, by digital circuitry or by an analog-to-digital converter which is sampling a continuous-time waveform. In digital signal processing these terms are commonly shortened to *signal* or *waveform*. Sometimes the term signal is restricted to being a desirable component of a sequence instead of being used interchangeably with waveform. *Noise* is either defined as a) an undesirable component of a sequence, or b) a sequence of random variables.

6) (*Digital*) *simulation* is the exact or approximate representation of a given system (discrete or contin-

Manuscript received August 1, 1972.
L. R. Rabiner, J. F. Kaiser, and R. W. Schafer are with Bell Telephone Laboratories, Inc., Murray Hill, N. J. 07974.
J. W. Cooley is with the IBM T. J. Watson Research Center, Yorktown Heights, N. Y. 10598.
H. D. Helms is with Bell Telephone Laboratories, Inc., Whippany, N. J. 07981.
L. B. Jackson is with Rockland Systems Corporation, West Nyack, N. Y. 10994.
C. M. Rader and C. J. Weinstein are with M.I.T. Lincoln Laboratory, Lexington, Mass. 02173. (Operated with support from the U. S. Air Force.)
K. Steiglitz is with Princeton University, Princeton, N. J. 08540.

uous) called the source system, by a (digital) system called the *object system.*

7) *Next-state simulation* is a method of digital simulation whereby the values of the digital system signals are represented by nodes in a block diagram representation. Usually, there is a close correlation between blocks in the object system and elements of the source system. The method entails ordering the calculations in the digital system so that all the inputs to each block at a given sample time are computed before the output is computed.

8) A *real-time process* is one for which, on the average, the computing associated with each sampling interval can be completed in a time less than or equal to the sampling interval. A program running in 100 times real time requires 100 times as long to process the same number of samples; i.e., it is 100 times too slow for real time operation. A program ten times as fast as it needs to be could be said to run in 1/10 real time. Obviously, the extra speed can only be used if other computing can be done in the interstices, or if the complete sequences to be processed have been stored beforehand.

9) *Throughput rate* is the total rate at which digital information is processed by a discrete-time system, measured in bits per second or samples per second. In a multiplexed system, where several signals are processed, we may refer to the *throughput rate per signal*, measured in bits per second per signal or samples per second per signal. Thus, a multiplexed system which processes 10 signals, each at 1000 bits/s, has a throughput rate of 10 000 bits/s, and a throughput rate/signal of 1000 bits/s/signal.

10) A *multirate system* is a discrete-time system in which there are signals sampled at different intervals which are usually integer multiples of some basic or fundamental interval.

II. Discrete Systems

1) The *z transform* plays a role in discrete-time system theory analogous to that of the Laplace transform in continuous-time system theory. Two view points regarding the z transform are common. One is based on what may be termed the one-sided z transform, which is defined as

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

regardless of the value of $x(n)$ for $n<0$.[1] One application of the *one-sided z transform* is in the solution of linear difference equations with constant coefficients. Solutions are obtained for the interval $0 \leq n < \infty$ subject to prescribed initial conditions. These solutions are obtained with the aid of the equations

$$y(n) \equiv x(n-m), \qquad m > 0 \tag{2}$$

$$Y(z) = z^{-m}\left[X(z) + \sum_{i=1}^{m} x(-i)z^{i}\right]. \tag{3}$$

In (3), $X(z)$ appears multiplied by z^{-m}. This result for $m=1$ accounts for the fact that z^{-1} is often termed the *unit delay operator*, since a delay of the sequence by one sample is equivalent to multiplication of the z transform by z^{-1}. (Similarly, z is often called the *unit advance* operator.)

2) In many cases, sequences are defined over both positive and negative values of n. In such cases, a somewhat more general point of view is called for. In general, the z transform is written as

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n}. \tag{4}$$

It should be noted that a common usage is to call (1) simply the z transform, and (4) the *two-sided z transform.* Since (4) is most general, it would seem preferable to refer to (4) as the z transform, and the special case, (1), as the *one-sided z transform.*

3) It is possible to think of the z transform as simply a formal series whose properties can be tabulated, and which never need be summed. However, it is generally preferable to realize that if certain convergence conditions are met, both (1) and (4) are Laurent series in the complex variable z. As such, all the properties of the Laurent series apply. For example, if the series in (1) converges, it must converge in a region $|z| > R_+$. If the series of (4) converges, it must converge in an annular region $R_+ < |z| < R_-$, where R_+ may be zero and R_- may be infinity. The coefficients of a Laurent series are determined by an integral relationship. In the context of the z transform, this relation is

$$x(n) = \frac{1}{2\pi j}\oint_C X(z)z^{n-1}\,dz, \tag{5}$$

where C is a closed contour inside the region of convergence of the power series and enclosing the origin. Equation (5) is referred to as the *inverse z transform.*

4) In the region of convergence of the series, both (1) and (4) represent analytic functions of the complex variable z. These functions can often be extended by analytic continuation everywhere except at certain singular points (poles). Since these singularities of the z transform are characteristic of the particular sequence, it is common to plot their locations in the z *plane*, (i.e., the complex plane determined by the real and imaginary parts of z). It should be noted that it is often convenient, because of the special functional form which characterizes exponential sequences, to plot singularities in the z^{-1} *plane.* Furthermore, some authors define the z transform as

$$\tilde{X}(z) = \sum_{n=-\infty}^{\infty} x(n)z^{n}. \tag{6}$$

[1] The notation $x(n)$ is used rather than x_n or $x(nT)$ to denote a sequence because of the ease of handling complicated indices, e.g., $x(N-1/2)$.

Clearly, (6) is related to our definition by

$$\tilde{X}(z) = X(z^{-1}). \tag{7}$$

If either (6) or the z^{-1} plane is encountered, it is a simple matter to replace z with z^{-1} in order to relate the z-transform definitions and to note that the region inside the unit circle of the z plane corresponds to the region outside the unit circle of the z^{-1} plane.

5) A *discrete-time impulse at* $k=k_0$ is a discrete-time signal $x(k)$ such that $x(k)=0$ unless $k=k_0$, in which case $x(k)=1$. This is an analogy with an *impulse* at time t_0 in the continuous-time case, where $x(t)=\delta(t-t_0)$, the Dirac delta function. The response of a digital filter to a *discrete-time impulse* at $k=0$ is called its *impulse response*, or sometimes, the *unit sample response*. Other terms generally used for *digital impulse* are *unit sample*, *unit pulse*, or simply *impulse*.

6) A *sample value* is the value or number associated with one member of a sequence that represents a discrete-time signal. This term is generally used regardless of whether or not the value represents a sample of a continuous-time signal.

7) A *discrete-time linear time-invariant system* or a *discrete-time linear filter* is characterized by its impulse-response sequence $h(n)$.

8) *Discrete-time convolution* is the operation on a signal or sequence $g(n)$ by the impulse response sequence $h(n)$ to yield a digital signal (or sequence) $f(n)$; the operation is defined by the expression

$$f(n) = \sum_{k=-\infty}^{\infty} h(k)g(n-k). \tag{8}$$

This expression is the discrete-time counterpart of the convolution integral for continuous-time systems.

9) An alternate characterization of a discrete-time linear system is the z transform of $h(n)$:

$$H(z) = \sum_{n=-\infty}^{\infty} h(n)z^{-n}. \tag{9}$$

The complex function $H(z)$ is called the *system function* or transfer function. The values taken by $H(z)$ when evaluated on the unit circle in the z plane give the *frequency response*. Each point on the unit circle, characterized by its angle only, corresponds to a particular frequency.[2] Several different units of frequency are in common use. Some authors express frequency in the conventional units of Hz, kHz, etc. Others use a system of rad/s. Still other authors use a normalized frequency with each frequency expressed as a fraction of the sampling frequency (f/f_s) or half the sampling frequency $(f/(f_s/2))$. Finally, some authors express the normalized frequency in rad/sample. The following table relates the units of frequency to the corresponding angle in the z-plane ($T=1/f_s$ is the sampling period).

Unit of Frequency	z-Plane Substitution	Range of Frequency Around Unit Circle
f in Hz	$z=e^{j2\pi fT}$	$0\leq f\leq f_s$
ω in rad/s	$z=e^{j\omega T}$	$0\leq\omega\leq 2\pi f_s$
$\hat{f}=(f/f_s)$, a fraction of the sampling frequency	$z=e^{j2\pi\hat{f}}$	$0\leq\hat{f}\leq 1$
$\tilde{f}=(f/0.5f_s)$, a fraction of half the sampling frequency	$z=e^{j\pi\tilde{f}}$	$0\leq\tilde{f}\leq 2$
θ in rad/sample	$z=e^{j\theta}$	$0\leq\theta\leq 2\pi$

When the frequency response is expressed in polar form, its magnitude as a function of frequency is called the *amplitude response*, and its angle as a function of frequency is called the *phase response*.

10) One class of linear time-invariant discrete-time filters is characterized by system functions of the form

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1+\sum_{k=1}^{N} a_k z^{-k}}. \tag{10}$$

Such filters have a *recursive realization* in the form of the difference equation

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k), \tag{11}$$

where y is the output sequence and x is the input sequence. The system function is generally written in terms of powers of z^{-1} as in (10) because it places in evidence the form of the difference equation given in (11), i.e., in terms of delays. $H(z)$ can also be written as

$$H(z) = \frac{z^{N-M}\sum_{k=0}^{M} b_k z^{M-k}}{z^N+\sum_{k=1}^{N} a_k z^{N-k}} \tag{12}$$

where it is assumed that the zeros of the numerator [zeros of $H(z)$] are distinct from the zeros of the denominator [poles of $H(z)$]. This form places in evidence the fact that regardless of the relative values of M and N, $H(z)$ has the same number of poles as zeros. In most cases—especially digital filters derived from analog designs—M will be less than or equal to N, and there will be at least $N-M$ zeros at $z=0$. Systems of this type are called Nth-order systems. When $M>N$, the order of the system is no longer unambiguous. Here, N gives the order as the term is used to characterize the mathematical properties of the difference equations. M gives the order used to characterize the complexity of a realization of the system. There is no general agreement as to which of M or N best characterizes the system when $M>N$.

[2] As will be clear from the discussion in Section III, each point on the unit circle, in fact, corresponds to an infinite set of uniformly spaced frequencies that are indistinguishable in a digital system.

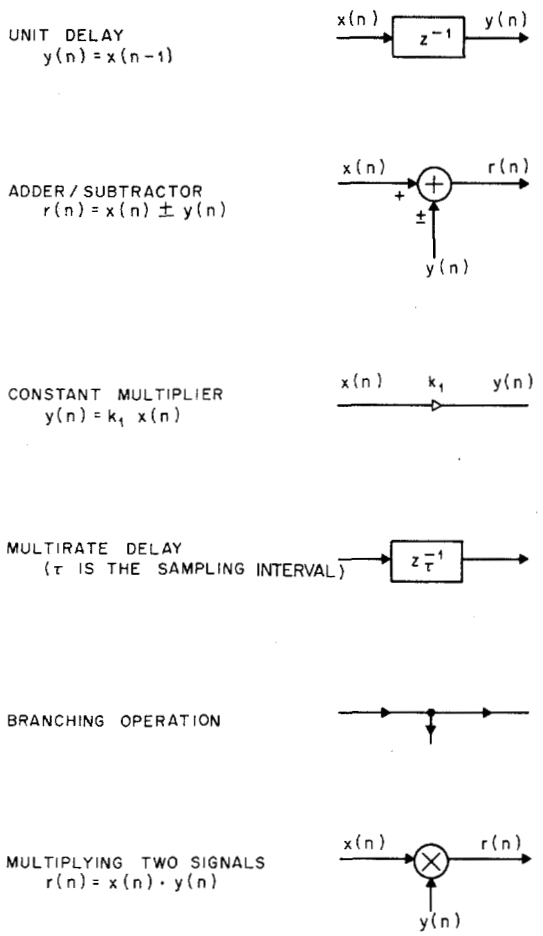

Fig. 1. Recommended terminology for use in block diagrams of digital systems.

11) Since the purpose of a *block diagram* is to graphically depict the way in which a particular system is realized, the terminology shown in Fig. 1 is recommended.

III. Relations Between Discrete and Continuous Signals

1) If a sequence arises as the result of periodic sampling of a continuous-time signal $x_c(t)$, i.e., $x(n)=x_c(nT)$ where T is the sampling period, then $X(z)$, the z transform of $x(n)$, is related to $X_c(s)$, the Laplace transform of $x_c(t)$ by the relationship

$$X(z)\big|_{z=e^{sT}} = \sum_{n=-\infty}^{\infty} x_c(nT)e^{-snT} = \frac{1}{T}\sum_{k=-\infty}^{\infty} X_c\left(s+j\frac{2\pi}{T}k\right). \tag{13}$$

This implies that the s plane and the z plane are related. It should be emphasized that (13) makes clear the fact that the s plane is *not* mapped into the z plane in a one-to-one manner. The actual relationship of $X(z)$ to $X_c(s)$ is depicted in Fig. 2, which shows the s plane divided into an infinite number of horizontal strips of

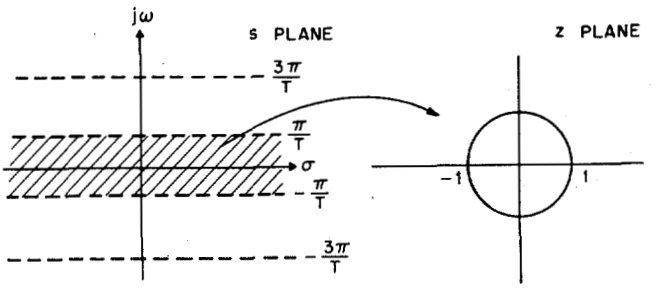

Fig. 2. The mapping of the s plane to the z plane implied by sampling a continuous-time signal.

width $2\pi/T$, each of which maps into the entire z plane. The contributions from each strip are added to produce $X(z)$.

2) The j axis of the s plane corresponds to the unit circle of the z plane. For this reason, the point $z=1$ is often casually referred to as the *DC point* since it corresponds to the point $s=0$ of the s plane. The z transform evaluated on the *unit circle* ($|z|=1$) is of particular interest in digital filtering of sampled signals. For example, there are some sequences for which the z transform does not converge (does not exist) except on the unit circle, e.g., the ideal lowpass digital filter and the ideal digital differentiator for band-limited waveforms. The z transform evaluated on the unit circle is called the *Fourier transform* of the sequence. This definition is consistent with the classical terminology for continuous-time systems. Evaluating (4) and (5) of Section II on the unit circle, yields

$$X(e^{j\theta}) = \sum_{n=-\infty}^{\infty} x(n)e^{-jn\theta} \tag{14}$$

$$x(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\theta})e^{jn\theta}\,d\theta. \tag{15}$$

This pair of equations is a Fourier transform pair for the sequence $x(n)$. Alternatively, any of the frequency units of the table in Section II can be used in place of θ.

3) If a continuous-time waveform, $x_c(t)$, is band-limited to a frequency f_0, i.e., $X_c(j2\pi f)$, the Fourier transform of $x_c(t)$, is zero for $|f|>f_0$, then $x_c(t)$ can be recovered exactly from its samples, i.e., $x_c(nT)$, $-\infty<n<\infty$ if $T<1/2f_0$. This is clear from (13) where we can see that

$$X(e^{j2\pi fT}) = \frac{1}{T}X_c(j2\pi f), \qquad -\frac{1}{2T}<f<\frac{1}{2T} \text{ if } T<1/2f_0.$$

$f_N=2f_0$ is called the *Nyquist rate*, and is the lowest rate at which $x_c(t)$ can be sampled, and still be recovered.[3] $T_N=1/f_N=1/2f_0$ is often called the *Nyquist interval*. Additional terms which are often used are *sampling frequency* or *sampling rate* for f_s, the rate at which $x_c(t)$ is actually sampled in a particular system; and $T=1/f_s$

[3] This discussion ignores the possible reduction in sampling rate which can be obtained for bandpass signals.

is called the *sampling interval*. The highest frequency present in $x_c(t)$, (defined above) f_0, is called the *Nyquist frequency*. The Nyquist frequency is sometimes called the *folding frequency*. It is recommended that the term Nyquist frequency be avoided because of the general confusion with the term Nyquist rate. Furthermore, we recommend that the term folding frequency refer to half of the actual sampling frequency (see Fig. 3).

4) The relationship (13) between the Fourier transform of a sequence of samples $x_c(nT)$ and the Fourier transform of the continuous time signal $x_c(t)$ is depicted in Fig. 4. Part (a) of this figure depicts a band-limited Fourier transform $X_c(j2\pi f)$. In Fig. 4(b) and (c) the sampling rate is greater than or equal to the Nyquist rate and we note that the form of $X_c(j2\pi f)$ is preserved to within a constant multiplier $1/T$ in the frequency range $-1/2T<f\leq 1/2T$. However, in Fig. 4(d) the signal $x_c(t)$ is *undersampled*, i.e., sampled at a rate below the Nyquist rate. In this case the Fourier transform of the sequence obtained by sampling is not equal to $X_c(j2\pi f)/T$ due to the fact that some of the other terms in (13) such as $X_c(j2\pi f-j(2\pi/T))$ are nonzero in the frequency range $-1/2T\leq f\leq 1/2T$. One way of viewing this is to say that a set of frequencies in $X_c(j2\pi f)$ is indistinguishable from a different set of frequencies in $X_c(j2\pi f-j2\pi/T)$. These frequencies are called *aliases* of one another and the process of confounding frequencies as in Fig. 4(d) is called *aliasing*.

5) Suppose we have a sampled waveform $x(n)$ with z transform $X(z)$. We define a new sampled waveform $y(n)$ using one of every M samples as the samples of the new waveform, i.e., $y(n)=x(Mn)$, with M any positive integer. Clearly, this process is equivalent to sampling at a lower rate, and it is to be expected that aliasing may occur. When the aliasing occurs due to "sampling" a discrete-time signal it is called *digital aliasing*. It is readily shown that $Y(z)$ can be written in terms of $X(z)$ as

$$Y(z)=\frac{1}{M}\sum_{l=0}^{M-1}X(z^{1/M}e^{-j(2\pi/M)l}). \tag{16}$$

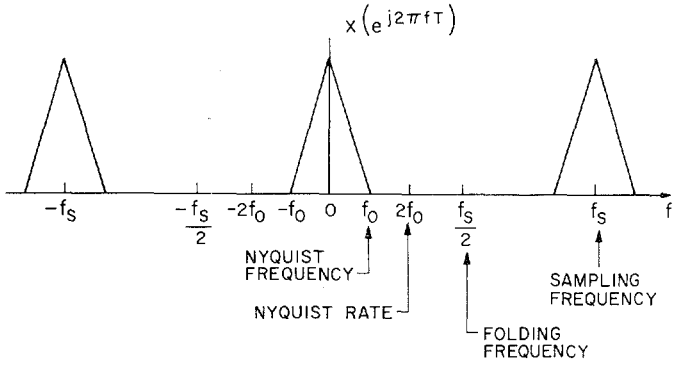

Fig. 3. Labeling of terminology concerned with frequencies related to the sampling process.

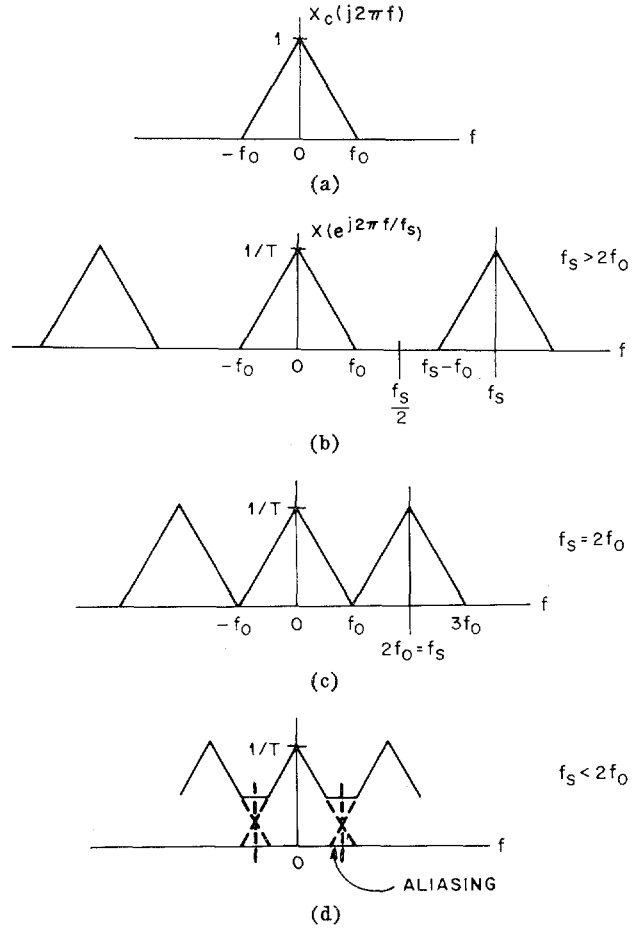

Fig. 4. An example of the effects of various sampling frequencies on the frequency response of the digital signal.

IV. Theory and Design of Digital Filters

1) *Discrete filters* may be divided into two classes on the basis of whether the signal values can take on a continuum of values (sampled-data filters) or a finite set of values (digital filter). Thus we have the following.

a) A *sampled-data filter* is a computational process or algorithm by which a sampled-data signal acting as an input is transformed into a second sampled-data signal termed the output. The sampled-data signal is considered only at a set of points (usually equally spaced in time or space as the independent variable); at these points the signal can take on a continuum of values.

b) A *digital filter* is a computational process or algorithm by which a digital signal or sequence of numbers (acting as input) is transformed into a second sequence of numbers termed the output digital signal. The numbers are limited to a finite precision. The algorithm may be implemented in software as a computer subroutine for a general-purpose machine or in hardware as a special-purpose computer. The term digital filter is then applied to the specific routine in execution or to the hardware.

2) Further complexity of filtering action may be obtained by switching. Thus, a *switched filter* is one in

which the input and output are simultaneously switched in a definite pattern among a group of input and output ports. The filter being switched may be either of the continuous or discrete types. Examples of switched filters are commutating filters or n-path filters.

3) A *multiplexed filter* is a restricted form of a switched filter; commonly a single discrete filter which by means of a switching action is made to perform the function of several discrete filters virtually simultaneously. The multiplexing is most commonly done in a time-division manner whereby the input to the discrete filter is sequentially switched from a number of input signals and the filter output sequentially switched in synchronism to a corresponding set of output signal lines. Thus a single filter may be made to do the work of many filters by this time division multiplexing.

4) A *recursive filter* is a discrete-time filter which is realized via a recursion relation, i.e., the output samples of the filter are explicitly determined as a weighted sum of past output samples as well as past and/or present input samples. For example, $y(n)=b_0x(n)+b_1x(n-1)+b_2x(n-2)-a_1y(n-1)-a_2y(n-2)$.

5) A *nonrecursive filter* is a discrete-time filter for which the output samples of the filter are explicitly determined as a weighted sum of past and present input samples only. For example, $y(n)=b_0x(n)+b_1x(n-1)+b_2x(n-2)$.

6) A *finite impulse response* (FIR) filter is a filter whose impulse response $h(n)$ is zero outside some finite limits, i.e., $h(n)=0$, for $n>N_1$ and $n<N_2$ with $N_1\geq N_2$.

7) An *infinite impulse response* (IIR) filter is a filter for which either $N_1=\infty$ or $N_2=-\infty$ or both, in 6). Thus the duration of the filter's impulse response is infinite.

8) It should be noted that the poles of FIR filters are restricted to $z=0$ or $z=\infty$, whereas there are no such restrictions on the positions of either the poles or zeros of IIR filters.

9) The terms recursive and nonrecursive are recommended as descriptions for how a filter is *realized* and not whether or not the filter impulse response is of finite duration. (Although IIR filters are generally realized recursively, and FIR filters are generally realized nonrecursively, IIR filters can be realized nonrecursively and FIR filters can be realized recursively.)

10) A *transversal filter* is a filter (either continuous or discrete) in which the output signal is generated by summing a series of delayed versions of the input signal weighted by a set of weights termed the *tap gains*. If the signal delays are accomplished by a tapped delay line then the filter is termed a *tapped delay line* filter.

11) A *comb filter* is a filter comprised of the sum or difference of input and output of a delay of M units and unit gain yielding a transfer characteristic $H(z)=1\pm z^{-M}$ (see Fig. 5); this filter has M zeros of transmission equally spaced on the unit circle in the z plane thus giving rise to a frequency characteristic having M equal peaks and M real frequency zeros.

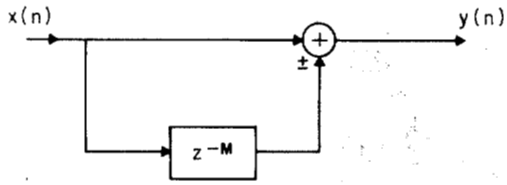

Fig. 5. Block diagram representation of comb filter.

12) A *frequency-sampling filter* is an FIR filter which is designed by varying one or more of its DFT[4] coefficients (called *frequency samples*) to minimize some aspects of the filter's frequency response. For example, the DFT coefficients of a frequency-sampling lowpass filter are 1.0 in the passband, 0.0 in the stopband, and variable in the transition band. One design criterion would be to choose the variable coefficients to minimize peak stopband ripple.

13) An *extraripple filter* (also called *maximal ripple filter*) is an FIR filter whose frequency response is equiripple in both the passband and stopband, and whose frequency response contains the maximum possible number of ripples.[5] There is no general agreement as to the appropriateness of this term, and as such, no recommendation as to its usage is made.

14) An *equiripple (optimal) filter* is an FIR filter which is the unique best approximation in the minimax sense to some specified frequency response characteristic over any closed subset of the frequency interval. For the lowpass filter case the optimal filter may be an extraripple filter, an equiripple filter with one less than the maximum possible number of ripples, or a filter with the maximum possible number of ripples all except one of which are of equal amplitude.

15) A *frequency-sampling realization* is a means of realization of an FIR filter of duration N samples as a cascade of a comb filter and a parallel bank of N complex pole resonators. The filter output is obtained as a weighted sum of the outputs of each of the parallel branches; the multiplier on the kth branch being the kth DFT coefficient of the filter impulse response.

16) A *Kalman filter* (*discrete time*) is a linear, but possibly time-varying discrete-time filter with the property that it provides a least mean-square error estimate of a (possibly vector-valued) discrete-time signal based on noisy observations. The statistical description of the problem is such that the Kalman filter has a recursive implementation, using a linear combination of new observations and old estimates. The filter design may be based on a more general error criterion, using a nonquadratic loss function. Its essential features are that its design is based on a statistical criterion in the time domain, and that it is, in general, time varying. If the filter is further restricted to be time invariant it becomes the *Wiener filter*.

[4] See Section VI-1 for a definition of DFT.
[5] See Section IV-29 for a definition of ripple.

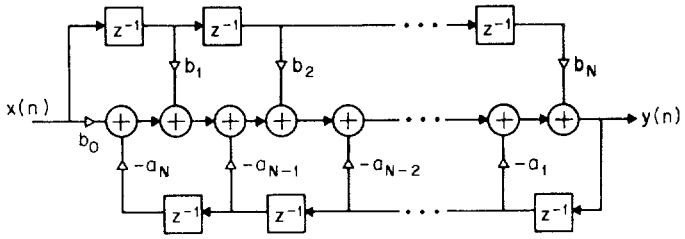

Fig. 6. Block diagram representation of direct form 1 for an Nth-order system.

17) The forms for realizing digital filters include the following:

a) *Direct form 1* (shown in Fig. 6) where

$$H(z) = \frac{\sum_{i=0}^{N} b_i z^{-1}}{\sum_{i=0}^{N} a_i z^{-1}}, \qquad a_0 = 1. \tag{17}$$

For convenience in showing the realization, the order of the numerator and denominator are set to be the same. Direct form 1 uses separate delays for both the numerator polynomial and the denominator polynomial. In certain cases, e.g., floating-point additions, the results may depend on the exact ordering in which the additions are performed.

b) *Direct form 2* is shown in Fig. 7. Direct form 2 has been called the *canonic form* because it has the minimum number of multiplier, adder, and delay elements, but since other configurations also have this property, this terminology is not recommended.

c) *Cascade canonic form* (or *series form*), which is shown in Fig. 8, where

$$H(z) = b_0 \prod_{i=1}^{K} H_i(z) \tag{18}$$

and $H_i(z)$ is either a *second-order section*, i.e.,

$$H_i(z) = \frac{1 + b_{1i}z^{-1} + b_{2i}z^{-2}}{1 + a_{1i}z^{-1} + a_{2i}z^{-2}}, \tag{19}$$

or a *first-order section*, i.e.,

$$H_i(z) = \frac{1 + b_{1i}z^{-1}}{1 + a_{1i}z^{-1}}, \tag{20}$$

and b_0 is implicitly defined in (17), where K is the integer part of $(N+1)/2$.

d) *Parallel canonic form*, which is shown in Fig. 9, where

$$H(z) = C + \sum_{i=1}^{K} H_i(z) \tag{21}$$

where $H_i(z)$ is either a *second-order section*, i.e.,

$$H_i(z) = \frac{b_{0i} + b_{1i}z^{-1}}{1 + a_{1i}z^{-1} + a_{2i}z^{-2}}, \tag{22}$$

or a *first-order section*, i.e.,

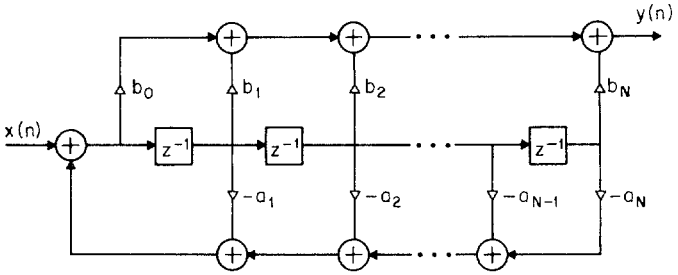

Fig. 7. Block diagram representation of direct form 2 for an Nth-order system.

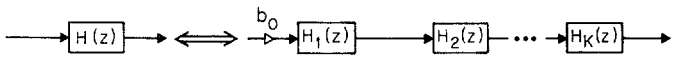

Fig. 8. Block diagram representation of the cascade form.

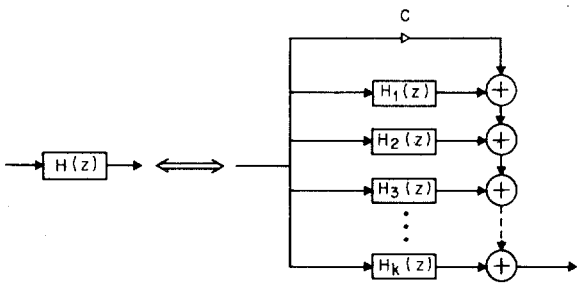

Fig. 9. Block diagram representation of the parallel form.

$$H_i(z) = \frac{b_{0i}}{1 + a_{1i}z^{-1}}, \tag{23}$$

and K = integer part of $(N+1)/2$ and C is proportional to b_N as defined in (17).

18) The individual second- and first-order sections of the cascade and parallel forms are generally realized in one of the direct forms.

19) *Transpose configurations* for all of the above forms can be obtained by reversing the directions of all signal flow (i.e., by reversing the directions of all arrows) and by interchanging all branch nodes and summing junctions. The resulting circuits have the same transfer functions but different roundoff noise and overflow properties.

20) When the transfer function of a high-order filter is decomposed into a cascade connection of lower order filter sections by distributing the pole and zero factors among the lower order sections, then *pairing* refers to the associating of a specific zero factor with a specific pole factor to form an elemental or individual section. *Ordering* refers to the sequence or order in which the individual sections are connected in cascade to form the composite higher order filter. Varying the pairing and ordering can dramatically change the noise properties and dynamic range of both discrete and continuous filters. As an example, if

$$H(z) = \frac{\prod_{j=1}^{l} N_j(z)}{\prod_{j=1}^{m} D_j(z)} \tag{24}$$

and $l=m=5$ then a possible detailed realization would be

$$H(z) = \frac{N_1(z)}{D_2(z)} \times \frac{N_3(z)}{D_1(z)} \times \frac{N_2(z)}{D_5(z)} \times \frac{N_4(z)}{D_3(z)} \times \frac{N_5(z)}{D_4(z)} \quad (25)$$

where the pairing is N_1 with D_2, N_3 with D_1, N_2 with D_5, N_4 with D_3, and N_5 with D_4. The implied ordering is N_1/D_2 first, followed by N_3/D_1, N_2/D_5, N_4/D_3, and finally N_5/D_4.

21) Two important properties of digital filters are *stability* and *causality*. The definition of *stability* most often used in digital filtering is as follows: a system is stable if every bounded (finite) input produces a bounded (i.e., finite) output. For linear time-invariant digital filters, a necessary and sufficient condition for stability is

$$\sum_{n=-\infty}^{\infty} |h(n)| < \infty. \quad (26)$$

22) A system is said to be *causal* if the output for $n=n_0$ is dependent only on values of the input for $n \leq n_0$. For linear time-invariant digital filters, this implies that the unit sample response sequence (i.e., the impulse response) is zero for $n<0$. For the case of most interest, i.e., causal linear time-invariant filters with rational transfer functions, stability implies that all the poles of $H(z)$ must be inside the unit circle in the z plane.

23) The *gain of a discrete filter* is the steady-state ratio of the peak magnitude (or any other consistent measure like root-mean-square, for example) of the output to the peak magnitude (or other consistent measure) of the input signal to the discrete filter. The usual input signals are either periodic sequences, e.g., sine waves, or pseudo-random sequences.

24) The *frequency-scale factor* is the factor by which all the poles and zeros of a normalized filter (cutoff frequency of 1 rad/s) must be multiplied to yield the actual filter pole and zero values, i.e., the ratio of the unnormalized to the normalized frequency scale of a filter.

25) The *filter bandwidth* is the width, in units of frequency, between the two points that define the edges of the passband of the frequency characteristics of a filter. The frequency points are usually defined as those values of frequency at which the attenuation or loss is a specified amount and beyond which the essential filter characteristic changes from pass (small attenuation) to stop (larger attenuation).

26) A commonly specified frequency point is the 3-dB or half-power point. For the elliptic and Chebyshev filters the frequency points are the highest and lowest frequencies at which the filter attenuation satisfies the equiripple passband attenuation limits. For other filters the frequency points may be defined in terms of the asymptotic intersections of the passband and stopband logarithmic asymptotes. Some examples of typical filter characteristics are shown.

27) Typical magnitude-square characteristics for several of the standard forms of continuous-time filters are given below using the following terminology.

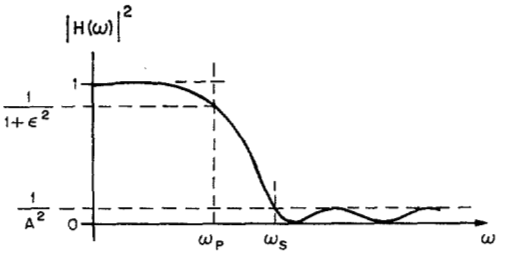

Fig. 10. An example of the magnitude-squared characteristics of a typical filter.

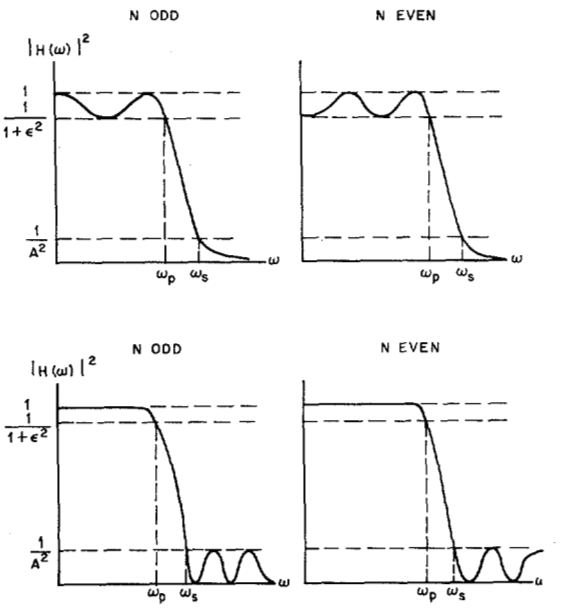

Fig. 11. The magnitude-squared characteristics for even and odd order Chebyshev types I (top) and II (bottom) filters.

$|H(\omega)|^2 \triangleq$ Magnitude-squared characteristic (frequency in rad/s).
ω_p Passband edge frequency.
ω_s Stopband edge frequency.

A typical response is shown in Fig. 10.

a) *Butterworth filter*: Maximally flat magnitude at $\omega=0$

$$|H(\omega)|^2 = \frac{1}{1+(\omega/\omega_p)^{2N}}. \quad (27)$$

b) *Chebyshev filters*:

Type I—Equiripple passband, monotone stopband:

$$|H(\omega)|^2 = \frac{1}{1+\epsilon^2 C_N^2(\omega)}. \quad (28)$$

Type II—Equiripple stopband, passband maximally flat at $\omega=0$:

$$|H(\omega)|^2 = \frac{1}{1+\epsilon^2\left[\frac{C_N(\omega_s)}{C_N(\omega_s/\omega)}\right]^2} \quad (29)$$

where the $C_N(\omega)$ are the Chebyshev polynomials. Fig. 11

shows the response for the two types of Chebyshev filters for N odd and even.

c) *Elliptic (Cauer) filters*—equiripple passband and stopband:

$$|H(\omega)|^2 = \frac{1}{1 + \epsilon^2 \psi_N^2(\omega)} \tag{30}$$

where the $\psi_N(\omega)$ is a rational Chebyshev function involving elliptic functions. Fig. 12 shows the response of elliptic filters for N both odd and even.

28) The term *transition band* is used to describe an interval of frequencies where a filter characteristic changes from one kind of behavior to another, one example being the transition band from a pass to a stop characteristic. The *transition ratio* is a relative measure of the passband width to the sum of the widths of the passband and the adjacent transition band(s). It can also be defined for a single-transition band-passband pair provided the width of the passband is defined. For the filter shown in Fig. 13 the transition ratios are defined as

$$\text{transition ratio} = \frac{\omega_c - \omega_{l2}}{\omega_c - \omega_l} \text{ (lower region)} \tag{31}$$

$$\text{transition ratio} = \frac{\omega_{u1} - \omega_c}{\omega_u - \omega_c} \text{ (upper region)} \tag{32}$$

where ω_c may be defined as either the arithmetic mean of the band-edge frequencies, i.e.,

$$\omega_c = \frac{\omega_{u1} + \omega_{l2}}{2} \tag{33}$$

or as the geometric mean of these same two frequencies, i.e.,

$$\omega_c = \sqrt{\omega_{u1}\omega_{l2}}. \tag{34}$$

Thus the transition ratio is bounded on the upper side by unity. Transition ratios near unity imply sharp cutoff filters.

29) The nature of a filter's response characteristic that approximates a desired characteristic by being alternatively greater than and less than the desired response as the independent variable is increased is called the *ripple*. The ripple may be expressed as the ratio of the maximum to the minimum of the response in a specified range, e.g., the passband of a filter. In this case, the ripple is usually expressed in percent or in decibels by taking 20 $\log_{10}$ of the ratio. Alternatively, the ripple may be expressed relative to some specified level of response such as plus or minus a fixed number of units. For example, consider the magnitude response shown in Fig. 14, where *passband ripple* $= 2.268/2.160 = 1.05$ which implies a $(2.268)/\sqrt{2.268 \times 2.160} = 1.0247$ or ± 2.47 percent variation about the geometric mean; thus passband ripple expressed in dB $(=)$ 20 $\log_{10}$ (1.05) $= 0.424$ dB overall or ± 0.212 dB ripple about the geometric mean.

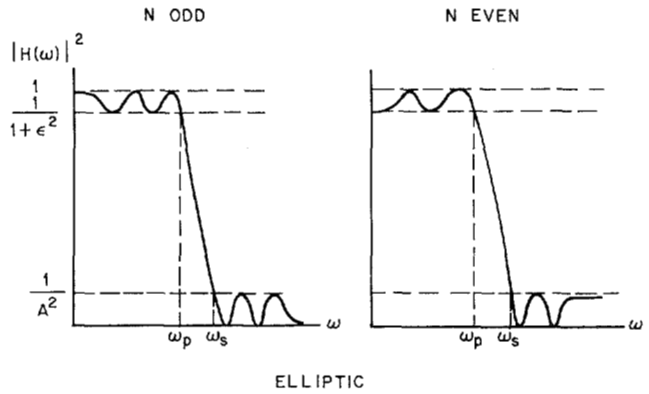

Fig. 12. The magnitude-squared characteristics for even and odd order elliptic filters.

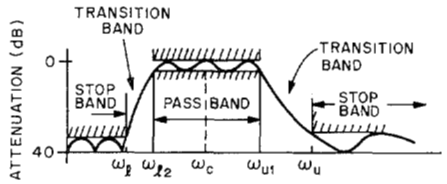

Fig. 13. The attenuation characteristics of a typical bandpass filter showing passband, stopbands, and transition bands.

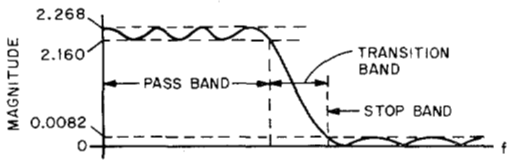

Fig. 14. The magnitude characteristic of a typical filter showing its ripple characteristics.

30) The passband ripple is also termed the *in-band ripple*. The terms *stopband ripple* and *out-of-band ripple* have also been used when the out-of-band frequency response has the characteristic of a ripple; numerically this has been used to express the ratio of the minimum out-of-band attenuation to the mean in-band attenuation. We recommend that this ratio be termed *minimum stopband attenuation* and that the terms stopband ripple and out-of-band ripple not be used except qualitatively. In the example, minimum stopband attenuation $= 0.0082(=) -41.7$ dB and the relative minimum stopband attenuation $= (0.0082)/\sqrt{2.268 \times 2.160} = 0.003705$ $(=) -48.6$ dB.

Methods for Designing Digital Filters

31) An important class of techniques for designing infinite impulse response filters to be realized recursively is based on a transformation of a continuous-time filter. This class consists of at least three techniques.

a) *Impulse invariance* (also called the standard z

transformation or standard z) is a technique in which the impulse response of the derived digital filter is identical to the sampled impulse response of a continuous-time filter. If the continuous-time filter has a transfer function[6]

$$H_c(s) = \frac{\sum_{k=0}^{M} d_k s^k}{\sum_{k=0}^{N} c_k s^k} = \sum_{k=1}^{N} \frac{A_k}{s + \alpha_k}, \qquad M < N \quad (35)$$

then the requirement that

$$h(n) = h_c(t)\,|_{t=nT}, \qquad 0 \leq n \leq \infty \quad (36)$$

implies that $H(z)$ is obtained from the partial fraction expansion of $H_c(s)$ by the substitution

$$\frac{1}{s + \alpha_k} \rightarrow \frac{1}{1 - e^{-\alpha_k T} z^{-1}}. \quad (37)$$

It can be shown that

$$H(z)\,|_{z=e^{sT}} = \frac{1}{T} \sum_{k=-\infty}^{\infty} H_c\left(s + j\frac{2\pi}{T}k\right). \quad (38)$$

Thus, impulse invariance is only satisfactory when $H_c(s)$ is band limited. If as in most instances, $H_c(s)$ is not sufficiently band limited, $H(z)$ is an aliased version of $H_c(s)$. Therefore, this technique is primarily used for narrowband filter designs or else the transformation is applied to the cascade combination of a *guard filter* and $H_c(s)$.

Another important point is clear from (38). Due to the $1/T$ multiplier, digital filters derived by impulse invariance have a gain approximately $1/T$ that of the continuous-time filter. This is generally compensated by multiplying each factor in the partial fraction expansion by T, so that the digital filter will have approximately the same gain as the continuous-time filter from which it was derived.

b) *Bilinear transformation* (also called the bilinear z transform, the bilinear z transformation or z form) is a technique used to circumvent the aliasing problem of the impulse invariant technique. This approach uses the algebraic transformation

$$s = \frac{2}{T} \frac{1 - z^{-1}}{1 + z^{-1}}, \quad (39)$$

to derive the system function of the digital filter as

$$H(z) = H_c(s)\,|_{s=(2/T)(1-z^{-1})/(1+z^{-1})}. \quad (40)$$

This transformation has the effect of mapping the entire s plane into the z plane in such a way that the left-half s plane maps into the inside of the unit circle and the right-half s plane maps to the outside of the unit circle. This results in a nonlinear warping of the frequency scale according to the relation

$$\frac{\omega_c T}{2} = \tan\frac{\omega_d T}{2} \quad (41)$$

where ω_c is the continuous-time frequency variable and ω_d is the discrete-time frequency variable. Because of this warping of the frequency scale, this design technique is most useful in obtaining digital designs of filters whose frequency response can be divided into a number of pass and stop bands in which the response is essentially constant. Generally it is necessary to take appropriate account of the warping of the frequency scale.

c) *Matched z transform* (also called the matched z transformation, mapping poles and zeros, or matched z) is a technique based on mapping the poles and zeros of the continuous-time filter by the substitution

$$s - s_i \rightarrow 1 - e^{s_i T} z^{-1}. \quad (42)$$

This means that the poles of $H(z)$ will be identical to those obtained by impulse invariant method, however the zeros will not correspond.

32) In the context of designing a discrete-time system and especially a digital filter, an *optimization technique* is a procedure for minimizing a prescribed performance function based on design requirements. An example is the design of a discrete-time filter to have the minimum mean-square deviation from a desired frequency-domain characteristic. An *iterative optimization technique* is a procedure for generating successive approximations converging (hopefully) to an optimum. This is opposed to an *analytical design technique*, which yields a closed form solution, such as the Chebyshev design for a lowpass filter.

V. Finite Word Length Effects—A/D, D/A Conversion

1) A *digital-to-analog (D/A) converter* is a device which operates on a digital input signal $s(nT)$ to produce a continuous-time output signal $s(t)$ ideally defined by

$$s(t) = \sum_n s(nT)h(t - nT) \quad (43)$$

where $h(t)$ characterizes the particular converter. For example, $h(t)$ is a square pulse of duration T for a zero order hold D/A converter. The D/A converter is usually followed by a linear time-invariant low-pass continuous-time filter called a *postfilter*. The combination of D/A converter and postfilter is called a *reconstruction device* or *reconstruction filter*.

2) An *analog-to-digital (A/D) converter* is a device which operates on a continuous-time waveform to pro-

[6] This formulation assumes that all poles are distinct. Appropriate modifications can be made to deal with multiple order poles.

duce a digital output consisting of a sequence of numbers each of which approximates a corresponding sample of the input waveform. Expressing the numerical equivalent of each sample by a finite number of bits (instead of the infinite number required to completely specify each sample) is the *quantizing* inherent in the conversion process. The error produced by quantizing is called *quantizing noise* or *A/D conversion noise.*

Representation of Numbers

3) Various systems are used to represent the numbers in a digital filter. In *fixed-point number* representation, the position of the binary (or decimal) point is assumed fixed. The bits to the right of the (fixed) binary (or decimal) point represent the fraction part of the number and the bits to the left represent the integer part. For example, the binary number 011.001 has the value $0\times2^2+1\times2^1+1\times2^0+0\times2^{-1}+0\times2^{-2}+1\times2^{-3}$.

4) A *floating-point number* is formed by two fixed point numbers, the *mantissa*[7] and the *exponent.* The floating-point number is equal to the product of the mantissa with the quantity resulting when a given *base* is raised to the power denoted by the exponent. The base is the same for every floating-point number in the digital filter. Consequently, the numerical value of an entry in a specified position in the mantissa is determined by the exponent. The mantissa is generally normalized to be as large as possible but less than some number (e.g., 1.0). For example, 0.1×10^2 is legitimate, whereas 0.01×10^3 and 10.0×10^0 are usually considered to be illegitimate floating-point decimal representations of the number 10. The most commonly used base is two (*binary representation*). The base 16 (*hexadecimal representation*) is used in some general purpose computers. The base 8 is called *octal representation.*

5) The representation of *block floating-point numbers* is determined by examining all numbers in a block (i.e., array). The largest number is represented as an ordinary floating-point number with a normalized mantissa. The remaining numbers in the block use the exponent associated with this largest number. This use of a single exponent for the whole block saves memory. This type of arithmetic is popular in realizations of the fast Fourier transform.

Representation of Negative Numbers

6) The discussion so far has dealt with the representation of nonnegative numbers. There are three common systems used for representing signed numbers. The representation of positive numbers is the same in these three systems. The first, and most familiar, is *sign and magnitude*, i.e., the magnitude (which is, of course, positive) is represented as a binary number and the sign is represented by an additional binary digit in the leading position which, if 0 corresponds to a + and if 1 corresponds to a − (or vice versa). Thus, for example, in sign and magnitude **0.0011** represents 3/16 and **1.0011** represents −3/16. Two related representations of signed numbers are ones complement and twos complement. In each of these systems a positive number is represented as in sign and magnitude. For *twos-complement* representation the negative of a particular positive number is obtained by complementing all the bits and adding one unit in the position of the least significant bit. For example, −(**0.0110**) would be represented in twos complement as (1.1001)+(0.0001)=1.1010. A carry out of the sign bit is neglected in the addition, so that −(**0.0000**) = (1.1111)+(0.0001) = 0.0000. For *ones-complement* representation the negative of a given positive number is obtained simply by complementing all the bits.

7) The choice of representation for negative numbers in a particular system is based almost entirely on hardware considerations. With ones-complement and twos-complement numbers, subtraction can be performed conveniently with an adder. For example, in twos complement, the difference $A-B$ is formed by simply adding to A the twos complement of B.

Finite Word Length Effects

8) Even though the input to a digital filter is represented with finite word length (e.g. through A/D conversion), the result of processing will naturally lead to values requiring additional bits fo. their representation. For example, a b-bit data sample multiplied by a b-bit coefficient results in a product which is $2b$ bits long. If in a recursive realization of a filter we do not quantize the result of arithmetic operations, the number of bits required will increase indefinitely, since after the first iteration $2b$ bits are required, after the second iteration $3b$ bits are required, etc. Two common methods are used to eliminate the lower order bits resulting from arithmetic operations in a digital filter.

a) *Truncation* is accomplished by discarding all bits (or digits) less significant than the least significant bit (or digit) which is retained.

b) *Rounding* of a number to b bits, when the number is initially specified to more than b bits, is accomplished by choosing the rounded result as the b-bit number closest to the original unrounded quantity. When the unrounded quantity lies equidistant between two adjacent b-bit numbers, a random choice ought to be made as to which of these numbers to round to. For example, **0.01011** rounded to three bits would be **0.011**; but **0.01010** rounded to three bits can be chosen as either **0.011** or **0.010**, and the choice should be random. In many situations, however, one can choose to always round up in this midway situation with negligible effect on the accuracy of the computation.

9) *Roundoff error* (or *roundoff noise*) or *truncation error* (or *truncation noise*) is caused by rounding off or truncating the products formed within the digital filter.

[7] The term mantissa as defined here is unfortunately not the same as the term mantissa commonly used in logarithm tables. The definition presented here is dictated by its extensive occurrence in the literature.

The roundoff or truncation error is sometimes well modeled as a random process. On the other hand, if the data sequence to a recursive realization of a digital filter consists of constants (e.g., zero) or some other periodically repeating samples, the roundoff or truncation error is periodic and causes a *deadband effect* or *limit cycle* in the filter output. A common type of limit cycle is a *zero-input limit cycle* where the output of a digital filter remains periodic and nonzero, after the input has been set to zero. *Dither* is a sequence of numbers that is added to the input to a recursive digital filter to ameliorate the deadband effect. Even though dither increases the mean-square error in the output, it can disrupt the pattern of roundoff errors causing the deadband effect, thereby permitting the output to return to zero.

10) *Overflow* occurs when a digital filter computes a number that is too large to be represented in the arithmetic used in that filter. If no compensation is made for the overflow then large errors in the filter output can result either in the form of transients or of *overflow oscillations*. A technique used to compensate in part for overflow is *saturation arithmetic* where a sum that is too large to be represented is set equal to the largest representable number in the filter.

11) *Dynamic range* is the ratio between the largest and smallest signals which can be represented in the filter with a given fidelity criterion. Unfortunately the fidelity criterion is often vague or unspecified.

12) The roundoff or truncation noise introduced within a digital filter produces a resultant noise at the output of the digital filter. A *signal-to-noise ratio* can be defined in this context, for example, as the ratio of the ideal mean-squared output signal (filter output in the absence of any rounding) to the mean-squared output noise due to rounding or truncation. Expressing this ratio in bits, as $(1/2)\log_2$ of the ratio, gives an approximate indication of the number of accurate bits in the filter output. Different definitions of signal-to-noise ratio may be appropriate in different contexts.

13) Another effect of finite word length is *coefficient quantization error* (or *parameter quantization error*), which occurs when the coefficients of a digital filter, initially specified with unlimited accuracy, are quantized by rounding or truncation. Coefficient quantization error appears as error in the digital filter's response (e.g., impulse response, transfer function, frequency response, etc.).

VI. Discrete Fourier Transforms and the FFT

1) For a sequence of N numbers, possibly complex, the *discrete Fourier transform* (*DFT*) is another sequence of exactly N numbers which are the values of the z transform of the original finite sequence for N values of z, specifically

$$z = e^{j(2\pi/N)k}, \qquad k = 0, 1, \cdots, N-1. \tag{44}$$

Discrete Fourier transformation is the operation of computing, or otherwise forming, the discrete Fourier transform of the sequence.

2) From the definition, the sequence $\{f(0), f(1), \cdots, f(N-1)\}$ has the DFT[8] $\{F(0), F(1), \cdots, F(N-1)\}$

$$F(k) = \sum_{n=0}^{N-1} f(n)e^{-j(2\pi/N)nk}. \tag{45}$$

3) It is possible to recover the original sequence from its DFT by the operation

$$f(n) = \frac{1}{N}\sum_{k=0}^{N-1} F(k)e^{j(2\pi/N)nk}, \tag{46}$$

giving a sequence of N samples $\{f(0), f(1), \cdots, f(N-1)\}$ as the inverse discrete Fourier transform of $\{F(0), F(1), \cdots, F(N-1)\}$. The operation is called *inverse discrete Fourier transformation* or IDFT, and is remarkably similar in form to the discrete Fourier transformation.

4) Some authors have defined the DFT in related but different ways, involving $e^{j(2\pi/N)nk}$, or a multiplicative factor of $1/N$ or $1/\sqrt{N}$. By considering the expressions for the DFT and IDFT it is evident that the constants $1/N$ or $1/\sqrt{N}$ and the possible use of $e^{j(2\pi/N)nk}$ in other definitions of the DFT can easily be compensated for in the other definitions of the IDFT.

5) Suppose for an N-point sequence we are interested in computing its DFT, and suppose N is a composite integer

$$N = r_1 \times r_2 \times \cdots \times r_\mu \tag{47}$$

where the r_i are a set of factors of N, not necessarily prime factors. Of the various algorithms for computing such a DFT, some require a number of operations *proportional* to $N\sum_{i=1}^{\mu} r_i$ (since the word proportional allows considerable latitude, it is not necessary to be too specific about the meaning of "operation"); such algorithms are called *fast Fourier transforms* (FFT).[9] An important special case is when

$$r_1 = r_2 = \cdots = r_\mu = 2$$

so that

$$\sum_{k=1}^{\mu} r_k \to 2\log_2 N.$$

For fast Fourier transforms in this case, the proportionality is to $N\log_2 N$.

6) A subclass of FFT algorithms is known which use high speed convolution techniques to compute the DFT of a sequence through a formula in which it is expressed as a convolution. Examples of such algorithms are the chirp z transform and the prime algorithm.

7) In order to classify different FFT algorithms and

[8] For convenience, the notation of $F(k)$ is used to denote DFT coefficients rather than $F(e^{j(2\pi/N)k})$.

[9] The word transform instead of algorithm is embedded ineradicably in the literature.

to relate them to one another it is useful to consider the following procedure applied to a sequence $f(n)$ of length N where N has a factorization $P\times Q\times R$. The reader should generalize to more complicated factorizations. Let us replace the index n by the triplet (n_0, n_1, n_2) where

$$(n_0, n_1, n_2) = n = n_0 + Rn_1 + QR\, n_2 \tag{48}$$

and

$$0 \le n_0 < R$$
$$0 \le n_1 < Q$$
$$0 \le n_2 < P.$$

Similarly, we replace the index k by the triplet (k_0, k_1, k_2) where

$$(k_0, k_1, k_2) = k = k_0 + Pk_1 + PQ\, k_2 \tag{49}$$

and

$$0 \le k_0 < P$$
$$0 \le k_1 < Q$$
$$0 \le k_2 < R.$$

Then (45) can be manipulated into the form

$$F(k_0, k_1, k_2) = \sum_{n_0=0}^{R-1}\left(\left\{\sum_{n_1=0}^{Q-1}\left(\left\{\sum_{n_2=0}^{P-1} f(n_0, n_1, n_2)\right.\right.\right.$$
$$\left.\left.\left.\cdot e^{-j(2\pi/P)n_2k_0}\right\}\phi_A\right)e^{-j(2\pi/Q)n_1k_1}\right\}\phi_B\Bigg)e^{-j(2\pi/R)n_0k_2}. \tag{50}$$

Here ϕ_A and ϕ_B are of unit magnitude and have arguments dependent on the indices. Equation (50), if followed as a recipe, suggests a way of computing an N-point DFT as a collection of smaller DFT's. There are: QR DFT's of P-point sequences; PR DFT's of Q-point sequences; and PQ DFT's of R-point sequences.

The only other operations called for in (50) are the multiplications by ϕ_A, ϕ_B. These have been called *twiddle factors*, *phase factors*, and *rotation factors* by various authors.

8) To save multiplications (50) is commonly modified in one of two ways. The first way is to combine the factors ϕ_A and $e^{-j(2\pi/P)n_2k_0}$ inside the sum over n_2 and the factors ϕ_B and $e^{-j(2\pi/Q)n_1k_1}$ inside the sum over n_1. The algorithm so produced has been called a *decimation-in-frequency* algorithm or a *Sande–Tukey* algorithm.

9) A second way to save multiplications is to combine the factors ϕ_A and $e^{-j(2\pi/Q)n_1k_1}$ inside the sum over n_1 and the factors ϕ_B and $e^{-j(2\pi/R)n_0k_2}$ inside the sum over n_0. The resulting form of the algorithm is called a *Cooley–Tukey* or *decimation-in-time* algorithm.

10) An algorithm with the twiddle factors explicitly present, as separate multiplications, is neither Cooley–Tukey nor Sande–Tukey.

11) There is a third way to save multiplications, which works only when the factors P, Q, R are relatively prime. By permuting the data sequence $f(n)$ and accepting a permuted transform sequence, a formula like (50) can be derived in which $\phi_A = \phi_B = 1$. This algorithm is called the *prime factor algorithm*.

12) Any of the basic forms can be programmed so that each summation is computed and the result stored in the memory formerly occupied by its input data as soon as that input data is no longer needed. In the case of (50), the amount of extra storage over the N cells needed for the sequence itself is only the greater of P, Q, or R cells.

A programmed version of an algorithm which takes advantage of this possibility is called *in place*. Unfortunately, when storage use is minimized in this way, either the input sequence or the output sequence must be unusually ordered. An example of this unusual order might be that

$$F(k_0 + Pk_1 + PQk_2) \tag{51}$$

ends up in storage location $k_2+Rk_1+QRk_0$. The unusual ordering may also take place in the use of weights in the algorithm. All such effects are called "digit reversal." If N is a power of two, and $P=Q=R=2$, the effect is called *bit reversal*.

13) If the factors of N are equal, say to r, the algorithm is called a *base-r* or *radix-r* algorithm and, if the factors are different, it is called a *mixed-radix* algorithm.

14) A common notation is to let W or W_N represent the reciprocal of the Nth principle root of unity,

$$W = W_N = \exp\left(-j\frac{2\pi}{N}\right). \tag{52}$$

Some authors have used

$$W = \exp\left(j\frac{2\pi}{N}\right). \tag{53}$$

15) For the base-2 Cooley–Tukey form of the FFT algorithm, the most fundamental operation is of the form

$$X = A + W^kB$$
$$Y = A - W^kB. \tag{54}$$

The Sande–Tukey form can be obtained by solving for A and B in terms of X and Y,

$$A = 0.5(X + Y)$$
$$B = 0.5(X - Y)W^{-k}. \tag{55}$$

This gives an elementary operation of an inverse transform. The Sande–Tukey form of the forward transform (DFT) is obtained by eliminating the 1/2 and replacing W^{-k} by W^k. From the appearance of the system flow graph for these operations, each operation is called a *butterfly*.

16) For diagramming the flow of processing data for the FFT, a flow graph notation is used. The flow graph for a trivial eight-point FFT appears as follows:

$$F(k) = \sum_{n=0}^{7} f(n)W^{nk}, \qquad k = 0, 1, \cdots, 7 \tag{56}$$

where

$$W = \exp\left(-j\frac{\pi}{4}\right). \tag{57}$$

The flow graph is shown in Fig. 15.

17) Each node represents a variable and the arrows terminating at that node originate at the nodes whose variables contribute to the value of the variable at that node. The contributions are additive, and the weight of each contribution, if other than unity, is indicated by the constant written close to the arrowhead of the contribution. Each node is assigned a pair of indices n, L. Variables at nodes in row n replace each other as they are computed and are stored in the cell with index n. All nodes in a column L are computed on iteration L. In this form of the algorithm, the exponent of W on the line entering node $(n; L)$ is $n \cdot 2^{3-L}$ (mod 8). It can also be seen that for each pair of operands, the second W is $W^4 = -1$ times the first and that the "butterfly" operation is indeed given by (54).

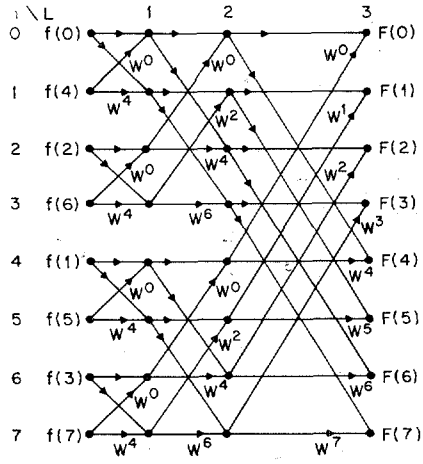

Fig. 15. The flow graph for an eight-point FFT.

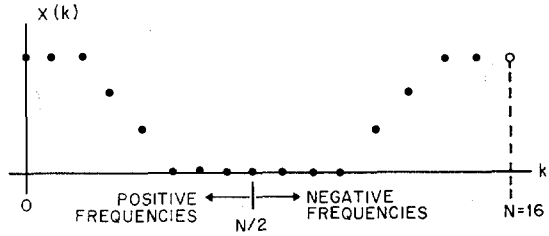

Fig. 16. The locations (in frequency) of the DFT points for a 16-point transform.

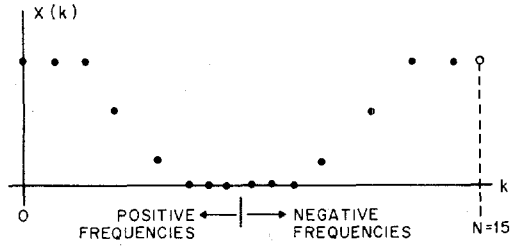

Fig. 17. The locations (in frequency) of the DFT points for a 15-point transform.

VII. Discrete Convolution and Spectrum Analysis

1) The Fourier transform (i.e., the z transform evaluated for $z = e^{j2\pi fT}$) of a sampled waveform is periodic in frequency, i.e., $X(e^{j2\pi T(f+f_s)}) = X(e^{j2\pi Tf})$ where $f_s = 1/T$ = sampling frequency. For this reason it is convenient to represent a *negative frequency* as the equivalent *positive frequency* below the sampling frequency, i.e., $f = -f_s/20$ is equivalent to $f = -f_s/20 + f_s = 19/20\, f_s$. In this manner one need only describe the spectrum in the positive frequency range of $0 \leq f < f_s$. For the DFT this convention also holds. Thus a typical spectrum of a 16-point DFT is shown in Fig. 16. For the above example the number of DFT points was even (16). The first DFT point, $X(0)$, corresponds to the Fourier transform evaluated at 0 frequency. The $(N/2+1)$st DFT point, $X(N/2)$, corresponds to the Fourier transform evaluated at half the sampling frequency. The index k corresponds to a frequency $f = k/TN$ in $X(e^{j2\pi Tf})$.

2) If the number of points in the DFT were odd, say $N = 15$, a typical spectrum would be as shown in Fig. 17. In this case there is no DFT point which corresponds to evaluating the Fourier transform at half the sampling frequency.

3) The *discrete convolution* of two sequences can be computed from the inverse discrete Fourier transform of the product of the discrete Fourier transforms of the two sequences. Thus, if $X(k)$ and $Y(k)$ are the discrete Fourier transforms of $x(n)$ and $y(n)$, the inverse discrete Fourier transform of the product of these discrete Fourier transforms produces a *periodic discrete convolution*, also called a *cyclic or circular discrete convolution* or simply a cyclic convolution.

4) The cyclic convolution can be written algebraically as

$$\sum_{m=0}^{N-1} x((n-m))y((m)) \tag{58}$$

where $((m))$ means the index is taken modulo N. Equation (58) can be written as

$$\sum_{m=0}^{n} x((n-m))y((m)) + \sum_{m=n+1}^{N-1} x((n-m))y((m)), \tag{59}$$

or, alternately,

$$\sum_{m=0}^{n} x((m))y((n-m)) + \sum_{m=n+1}^{N-1} x((m))y((n-m+N)). \tag{60}$$

The simultaneous presence of both summations is generally undesirable for computing ordinary convolutions. For example, if the first summation in (60) is chosen to represent the desired convolution, then the second summation represents an error term. By augmenting both sequences with zeros so that they have the same length N, which is at least as great as one less than the sum of

the lengths of the two sequences, cyclic convolution can be made to yield the same result as ordinary convolution.

5) This use of the fast Fourier transform to compute discrete convolutions is sometimes called *fast convolution* or *FFT convolution.* This technique can be easily adapted for computing acyclic (i.e., nonperiodic) correlation functions. In this form, it is called *fast correlation or FFT correlation.*

6) If one of the two sequences is much shorter than the other, the longer sequence can be *sectioned* into pieces whose discrete convolutions can be computed separately. These discrete convolutions can be combined to produce the discrete convolution of the whole sequence. (*Sectioning* is used because it reduces the required amounts of computation and memory.) One of these sectioning techniques (*overlap-save* or *select-save*) involves computing the inverse DFT of the product of the DFT's of a) N samples of the input sequence, and b) the shorter sequence augmented with a sufficient number of zeros so that its sequence contains N samples. (Usually N is at least twice as large as the length of the shorter sequence.) Some of the members of the sequence resulting from the inverse DFT are members of the sequence formed by the desired acyclic (i.e., nonperiodic) convolution. (This number of members equals one more than the number of zeros originally augmenting the shorter sequence.) The longer original sequence is advanced by this number of members. Iterating this process gives the whole convolution. The *overlap-add* technique for sectioning uses a similar technique but additionally requires adding shifted sequences of partial convolutions.

7) A *window* is a finite sequence, each element of which multiplies a corresponding element of the main sequence. (This is called *windowing.*) The sequence of products formed by this element-by-element multiplication is often more useful than the main sequence. The Fourier transform of a typical window (sometimes called the *spectral window*) consists of a *mainlobe,* which usually contains a large percentage of the energy in the window, and *sidelobes* which contain the remaining energy in the window.

8) Windows can be used in estimating power spectra. In the *direct method,* the power spectrum is estimated by computing the square of the absolute value of the DFT of the windowed sequence. The DFT of the windowed sequence is the convolution of the DFT's of the window and the original sequence. This convolution smoothes the input power spectrum, consequently values of the power spectrum at frequencies separated by less than the width of the mainlobe of the spectral window cannot be *resolved.* In addition to this limit on *resolution,* the estimate of the power spectrum may contain significant *leakage,* i.e., erroneous contributions from components of the power spectrum at frequencies possibly distant from the frequency of interest because of the nonzero energy in the spectral window sidelobes.

9) Windows are useful in determining the coefficients of a finite impulse response digital filter. In this case, the original sequence consists of samples of the impulse response corresponding to a transfer function which is approximated by the Fourier transform of the sequence of pairwise products; the product sequence is used as the coefficients of the finite impulse response digital filter.

10) Windows are used also in the *indirect method* of computing a power spectrum. In this method, the sequence consisting of samples of the autocorrelation function is multiplied by the window. The DFT of the resulting sequence is an estimate of the power spectrum.

11) Windows can be used also in estimating cross spectra where the estimates are obtained by multiplying the products of the DFT's of two or more distinct sequences.

12) The determination of a finite impulse response described by an ordinary convolution is called *deconvolution* or *FIR identification.*

Acknowledgment

The authors would like to acknowledge the comments and criticisms of this paper provided by G. D. Bergland, C. H. Coker, D. L. Favin, B. Gold, O. Hermann, S. Lerman, A. V. Oppenheim, H. O. Pollak, and H. F. Silverman.

Alphabetic Index of Terms