A guided tour of the fast Fourier transform

The fast Fourier transform algorithm can reduce the time involved in finding a discrete Fourier transform from several minutes to less than a second, and also can lower the cost from several dollars to several cents

G. D. Bergland *Bell Telephone Laboratories, Inc.*

For some time the Fourier transform has served as a bridge between the time domain and the frequency domain. It is now possible to go back and forth between waveform and spectrum with enough speed and economy to create a whole new range of applications for this classic mathematical device. This article is intended as a primer on the fast Fourier transform, which has revolutionized the digital processing of waveforms. The reader's attention is especially directed to the IEEE Transactions on Audio and Electroacoustics for June 1969, a special issue devoted to the fast Fourier transform.

This article is written as an introduction to the fast Fourier transform. The need for an FFT primer is apparent from the barrage of questions asked by each new person entering the field. Eventually, most of these questions are answered when the person gains an understanding of some relatively simple concept that is taken for granted by all but the uninitiated. Here the basic concepts will be introduced by the use of specific examples. The discussion is centered around these questions:

1. What is the fast Fourier transform?
2. What can it do?
3. What are the pitfalls in using it?
4. How has it been implemented?

Representative references are cited for each topic covered so that the reader can conveniently interrupt this fast guided tour for a more detailed study.

What is the fast Fourier transform?

The Fourier transform has long been used for characterizing linear systems and for identifying the frequency components making up a continuous waveform. However, when the waveform is sampled, or the system is to be analyzed on a digital computer, it is the finite, discrete version of the Fourier transform (DFT) that must be understood and used. Although most of the properties of the continuous Fourier transform (CFT) are retained, several differences result from the constraint that the DFT must operate on sampled waveforms defined over finite intervals.

The fast Fourier transform (FFT) is simply an efficient method for computing the DFT. The FFT can be used in place of the continuous Fourier transform only to the extent that the DFT could before, but with a substantial reduction in computer time. Since most of the problems associated with the use of the fast Fourier transform actually stem from an incomplete or incorrect understanding of the DFT, a brief review of the DFT will first be given. The degree to which the DFT approximates the continuous Fourier transform will be discussed in more detail in the section on "pitfalls."

The discrete Fourier transform. The Fourier transform pair for continuous signals can be written in the form

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-i2\pi ft}\,dt \tag{1}$$

$$x(t) = \int_{-\infty}^{\infty} X(f)e^{i2\pi ft}\,df \tag{2}$$

for $-\infty < f < \infty$, $-\infty < t < \infty$, and $i = \sqrt{-1}$. The uppercase $X(f)$ represents the frequency-domain function; the lowercase $x(t)$ is the time-domain function.

The analogous discrete Fourier transform pair that applies to sampled versions of these functions can be written in the form

$$X(j) = \frac{1}{N}\sum_{k=0}^{N-1} x(k)e^{-i2\pi jk/N} \tag{3}$$

$$x(k) = \sum_{j=0}^{N-1} X(j)e^{i2\pi jk/N} \tag{4}$$

for $j = 0, 1, \cdots, N-1$; $k = 0, 1, \cdots, N-1$. Both

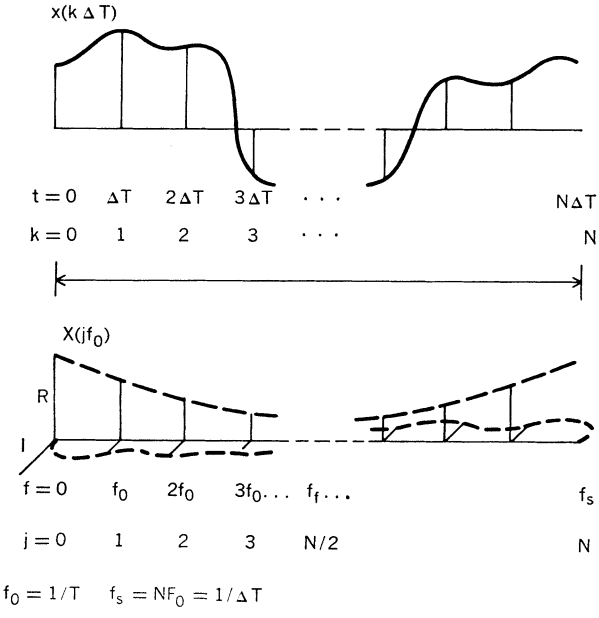
$X(j)$ and $x(k)$ are, in general, complex series. A derivation of the discrete Fourier transform from the continuous Fourier transform can be found in Refs. 12 and 23.

When the expression $e^{2\pi i/N}$ is replaced by the term W_N, the DFT transform pair takes the form

$$X(j) = \frac{1}{N}\sum_{k=0}^{N-1} x(k)W_N{}^{-jk} \tag{5}$$

$$x(k) = \sum_{j=0}^{N-1} X(j)W_N{}^{+jk} \tag{6}$$

FIGURE 1. A real signal and its complex discrete Fourier transform displayed in the FFT algorithm format.



An example of a real-valued time series and its associated DFT is shown in Fig. 1. The time series $x(k\Delta T)$ is assumed to be periodic in the time domain of period T seconds, and the set of Fourier coefficients $X(jf_0)$ is assumed to be periodic over the sample frequency f_s. Only one complete period of each function is shown.

The fundamental frequency f_0 and the sample period ΔT do not appear explicitly in Eqs. (5) and (6), but each j should still be interpreted as a harmonic number and each k still refers to a sample period number. That is, the true frequency is the product of j and f_0 and the true time is the product of k and ΔT.

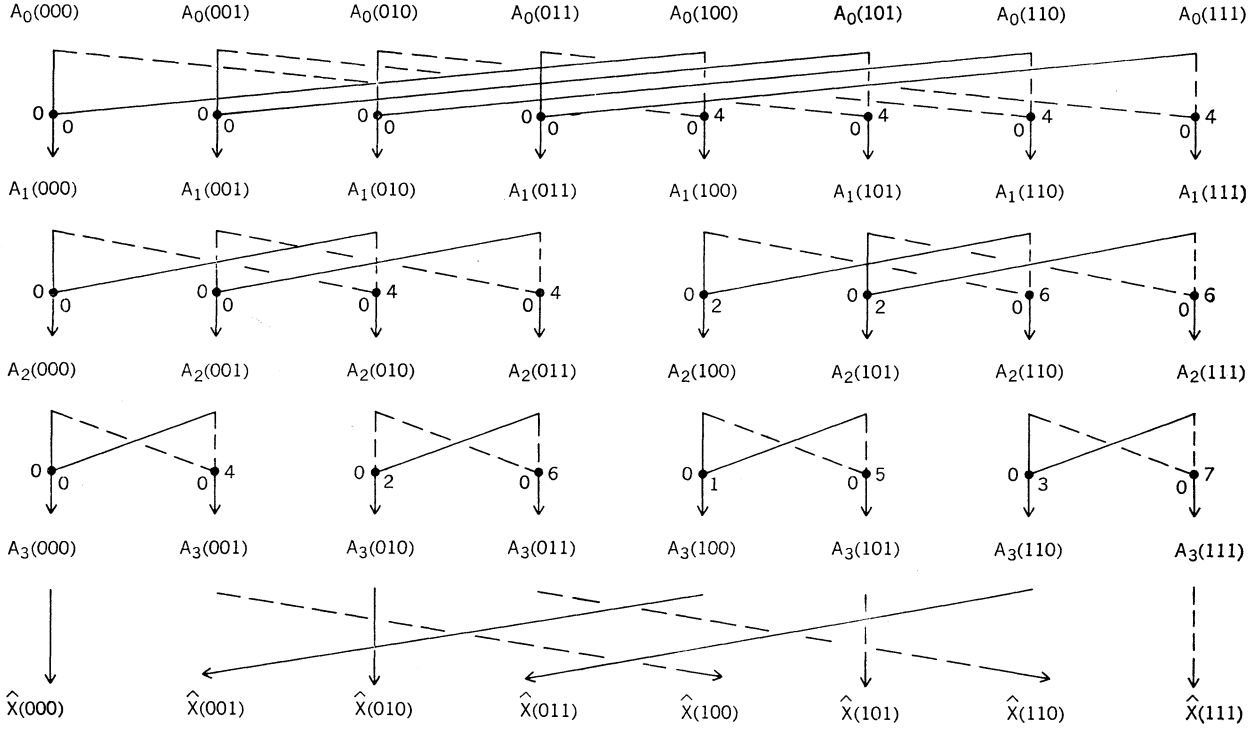
When the $x(k)$ series is real, the real part of $X(j)$ is symmetric about the folding frequency f_f (where $f_f = f_s/2$) and the imaginary part is antisymmetric. Since $X(j)$ has been interpreted as being periodic, these symmetries are equivalent to saying that the real part of $X(j)$ is an even function, and that the imaginary part of $X(j)$ is an odd function. This also means that the Fourier coefficients between $N/2$ and $N - 1$ can be viewed as the "negative frequency" harmonics between $-N/2$ and -1. Likewise, the last half of the time series can be interpreted as negative time (that is, as occurring before $t = 0$).

Derivation of the Cooley–Tukey FFT algorithm. A derivation of the Cooley–Tukey FFT algorithm[8] for evaluating Eq. (6) is given in this section for the example of $N = 8$. This derivation is also appropriate to the forward transform, since Eq. (5) can be rewritten in the form[9]

$$X(j) = \frac{1}{N}\left[\sum_{k=0}^{N-1} x(k)^* W_N{}^{jk}\right]^* \tag{7}$$

where the asterisk refers to the complex conjugate operation. Alternatively, the FFT algorithm used for comput-

FIGURE 2. A flow diagram of the Cooley–Tukey FFT algorithm for performing an eight-point transform.



ing Eq. (6) can be altered by redefining W_N to be exp $(-2\pi i/N)$ and dividing each result by N.

Using Cooley's notation,[8] the FFT algorithm involves evaluating the expression

$$\hat{X}(j) = \sum_{k=0}^{N-1} A(k)W^{jk} \tag{8}$$

where $j = 0, 1, \ldots, N-1$, and $W = \exp(2\pi i/N)$. Note that $\hat{X}$ and A can be interpreted as X^* and x^*/N, respectively, if the forward transform is being computed, and can be interpreted as x and X, respectively, if the inverse transform is being computed.

When N is equal to 8, it is convenient to represent both j and k as binary numbers; that is, for

$$j = 0, 1, \cdots, 7 \qquad k = 0, 1, \cdots, 7$$

we can write

$$j = j_2 4 + j_1 2 + j_0 \qquad k = k_2 4 + k_1 2 + k_0 \tag{9}$$

where j_0, j_1, j_2, k_0, k_1, and k_2 can take on values of 0 and 1 only. Using this representation of j and k, Eq. (8) becomes

$$\hat{X}(j_2,j_1,j_0) = \sum_{k_0=0}^{1}\sum_{k_1=0}^{1}\sum_{k_2=0}^{1} A(k_2,k_1,k_0)W^{(j_2 4+j_1 2+j_0)(k_2 4+k_1 2+k_0)} \tag{10}$$

Noting that $W^{m+n} = W^m \cdot W^n$, we have

$$W^{(j_2 4+j_1 2+j_0)(k_2 4+k_1 2+k_0)} = W^{(j_2 4+j_1 2+j_0)k_2 4}W^{(j_2 4+j_1 2+j_0)k_1 2}W^{(j_2 4+j_1 2+j_0)k_0} \tag{11}$$

If we look at these terms individually, it is clear that they can be written in the form

$$W^{(j_2 4+j_1 2+j_0)k_2 4} = [W^{8(j_2 2+j_1)k_2}]W^{j_0 k_2 4} \tag{12}$$

$$W^{(j_2 4+j_1 2+j_0)k_1 2} = [W^{8j_2 k_1}]W^{(j_1 2+j_0)k_1 2} \tag{13}$$

$$W^{(j_2 4+j_1 2+j_0)k_0} = W^{(j_2 4+j_1 2+j_0)k_0} \tag{14}$$

Note, however, that

$$W^8 = [e^{2\pi i/8}]^8 = e^{2\pi i} = 1 \tag{15}$$

Thus, the bracketed portions of Eqs. (12) and (13) can be replaced by a one. This means that Eq. (10) can be written in the form

$$\hat{X}(j_2,j_1,j_0) = \sum_{k_0=0}^{1}\sum_{k_1=0}^{1}\sum_{k_2=0}^{1} A(k_2,k_1,k_0)W^{j_0 k_2 4}W^{(j_1 2+j_0)k_1 2}W^{(j_2 4+j_1 2+j_0)k_0} \tag{16}$$

where $A_1(j_0,k_1,k_0)$, $A_2(j_0,j_1,k_0)$, $A_3(j_0,j_1,j_2)$.

In this form it is convenient to perform each of the summations separately and to label the intermediate results. Note that each set consists of only eight terms and that only the latest set needs to be saved. Thus the equations can be rewritten in the form

$$A_1(j_0,k_1,k_0) = \sum_{k_2=0}^{1} A(k_2,k_1,k_0)W^{j_0 k_2 4} \tag{17}$$

$$A_2(j_0,j_1,k_0) = \sum_{k_1=0}^{1} A_1(j_0,k_1,k_0)W^{(j_1 2+j_0)k_1 2} \tag{18}$$

$$A_3(i_0,j_1,j_2) = \sum_{k_0=0}^{1} A_2(j_0,j_1,k_0)W^{(j_2 4+j_1 2+j_0)k_0} \tag{19}$$

$$\hat{X}(j_2,j_1,j_0) = A_3(j_0,j_1,j_2) \tag{20}$$

The terms contributing to each sum are shown in Fig. 2. Each small number refers to a power of W applied along the adjacent path. The last operation shown in Fig. 2 is the reordering. This is due to the bit reversal in the arguments of Eq. (20).

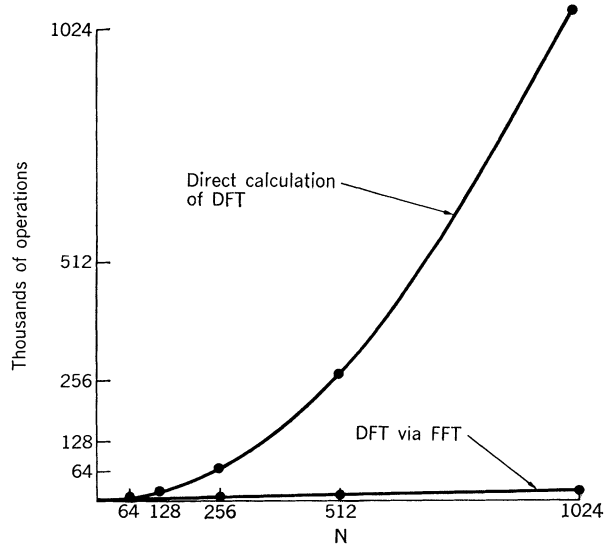
This set of recursive equations represents the original Cooley–Tukey formulation of the fast Fourier transform algorithm for $N = 8$. Although a direct evaluation of Eq. (8) for $N = 8$ would require nearly 64 complex multiply-and-add operations, the FFT equations show 48 operations. By noting that the first multiplication in each summation is actually a multiplication by $+1$, this number becomes only 24. By further noting that $W^0 = -W^4$, $W^1 = -W^5$, etc., the number of multiplications can be reduced to 12. These reductions carry on to the more general case of $N = 2^m$, reducing the computation from nearly N^2 operations to $(N/2)\log_2 N$ complex multiplications, $(N/2)\log_2 N$ complex additions, and $(N/2)\log_2 N$ subtractions. For $N = 1024$, this represents a computational reduction of more than 200 to 1. This difference is represented graphically in Fig. 3.

What can it do?

The operations usually associated with the FFT are: (1) computing a spectrogram (a display of the short-term power spectrum as a function of time); (2) the convolution of two time series to perform digital filtering; and (3) the correlation of two time series. Although all of these operations can be performed without the FFT, its computational savings have significantly increased the interest in performing these operations digitally.

Spectrograms. The diagram in Fig. 4 represents a method of obtaining estimates of the power spectrum of a time signal through the use of the fast Fourier transform.

FIGURE 3. The number of operations required for computing the discrete Fourier transform using the FFT algorithm compared with the number of operations required for direct calculation of the discrete Fourier transform.



In this case the square of the magnitude of the set of complex Fourier coefficients (that is, the periodogram) is used to estimate the power spectrum of the original signal.

A snapshot of the spectrum of the signal can always be computed from the last T seconds of data. By taking a series of these snapshots, estimates of the power spectrum can be displayed as a function of time as shown in Fig. 5. When the audio range is displayed, this is usually called a sound spectrogram or sonogram.

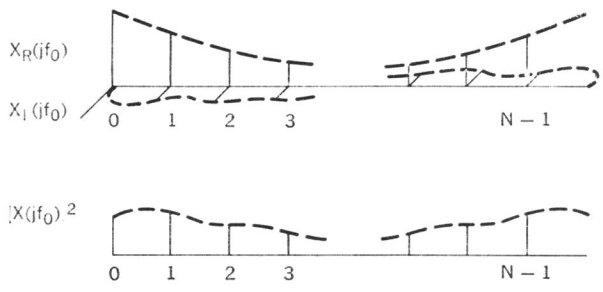
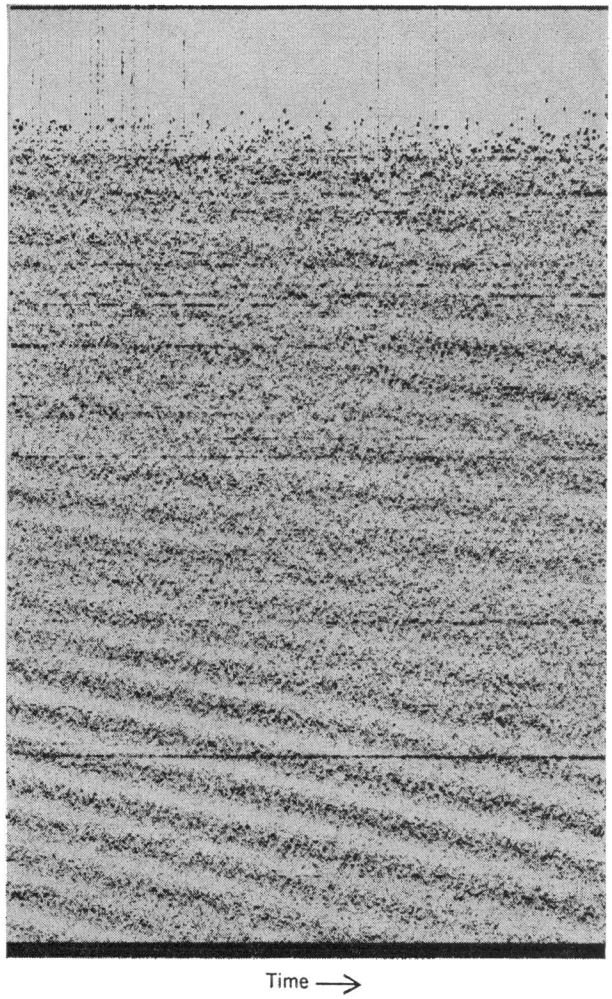


FIGURE 4. The power spectrum of a real function computed by taking the sum of the squares of the real and imaginary components of the discrete Fourier transform Fourier coefficients.

FIGURE 5. A spectrogram made by using the fast Fourier transform algorithm.



In some cases it is of interest to go one step further. When the spectrum of a signal contains a periodic component, this spectrum can be compressed by taking the logarithm, and then the fast Fourier transform can be taken again. The result is called a cepstrum (pronounced "kepstrum").[13,19] An example of using the cepstrum to determine the pitch period of a speaker is described in Ref. 14. For a more complete discussion of short-term spectrum and cepstrum analysis see Refs. 10–21.

Digital filtering. In a linear system, one is frequently confronted with the problem of either (1) determining the output, given the input and the impulse response, or (2) finding the impulse response, given the input and the output. Both of these problems can be approached rather easily in the frequency domain.

In Fig. 6, the output $c(t)$ is formed by convolving the input $g(t)$ with the impulse response of the system $h(\tau)$. For sampled functions this convolution takes the form

$$c(k) = \frac{1}{N}\sum_{\tau=0}^{N-1} g(\tau)h(k-\tau) \tag{21}$$

This equation represents the linear system in Fig. 6, as long as $h(\tau)$ is assumed to be zero for $\tau < 0$. If both $g(k)$ and $h(k)$ consist of N consecutive nonzero samples, the series $c(k)$ can consist of $2N-1$ nonzero terms.

Since the FFT algorithm gives us a fast way of getting to the frequency domain, it is interesting to consider

$$C(j) = G(j)\cdot H(j) \tag{22}$$

where $C(j)$ is the DFT of $c(k)$, $G(j)$ is the DFT of $g(k)$, and $H(j)$ is the DFT of $h(k)$. By Eq. (6), this is equivalent to

$$c(k) = \sum_{j=0}^{N-1} [G(j)\cdot H(j)]W^{jk} \tag{23}$$

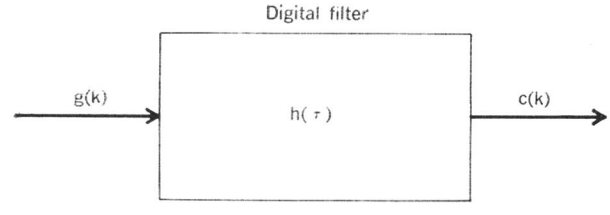
By Eq. (5), this can be written as

$$c(k) = \sum_{j=0}^{N-1}\left(\frac{1}{N}\sum_{\tau=0}^{N-1} g(\tau)W^{-j\tau}\right)\left(\frac{1}{N}\sum_{\hat{\tau}=0}^{N-1} h(\hat{\tau})W^{-j\hat{\tau}}\right)W^{jk} \tag{24}$$

Since all of the sums are finite, this can be rewritten as

$$c(k) = \frac{1}{N}\sum_{\tau=0}^{N-1}\sum_{\hat{\tau}=0}^{N-1} g(\tau)h(\hat{\tau})\left[\frac{1}{N}\sum_{j=0}^{N-1} W^{+j(k-\tau-\hat{\tau})}\right] \tag{25}$$

FIGURE 6. The response of a linear system to a driving function g(k) expressed as the convolution of the input signal with the impulse response of the system.



$$c(k) = \frac{1}{N}\sum_{\tau=0}^{N-1} g(\tau)\,h(k-\tau)$$

$$= g * h$$

This can be simplified through the use of the orthogonality relationship

$$\sum_{j=0}^{N-1} W_N^{nj} W_N^{-mj} = N \qquad \text{if } n = m \bmod N$$

$$= 0 \qquad \text{otherwise} \qquad (26)$$

Thus Eq. (25) is equal to zero unless $\hat{\tau} = k - \tau$, for which we have

$$c(k) = \frac{1}{N} \sum_{\tau=0}^{N-1} g(\tau) h(k - \tau) \qquad (27)$$

This equation is identical to the desired Eq. (21) but the requirement that $h(\tau)$ be zero for $\tau < 0$ is not met. In representing $g(k)$ and $h(k)$ by their Fourier coefficients the assumption is made that they are periodic functions. A method of sidestepping this problem is described later.

Correlation. By considering the equation

$$C(j) = G(j) \cdot H^*(j) \qquad (28)$$

and following a development similar to that of the preceding section, we obtain the result

$$c(k) = \frac{1}{N} \sum_{\tau=0}^{N-1} g(\tau) h(\tau - k) \qquad (29)$$

This is the form of the cross-correlation function of $g(k)$ and $h(k)$. When $h(k) = g(k)$ we obtain the autocorrelation function. The problem again is that both $g(k)$ and $h(k)$ were assumed to be periodic in finding their Fourier coefficients. This problem is discussed further in the next section.

From Eqs. (29) and (27), it is clear that convolution is simply the process of correlating one time series with another time series that has been reversed in time.

What are the pitfalls?

The three problems most often encountered in using the discrete Fourier transform appear to be aliasing, leakage, and the picket-fence effect. Also of interest are the problems associated with the blind use of Eqs. (22) and (28) to perform convolutions and correlations. An ever-present problem, which is not discussed here, concerns finding the statistical reliability of an individual power spectral estimate when the signal being analyzed is noise-like. A discussion of this last problem can be found in Refs. 11, 16, 17, 19, and 21.

Relating the DFT to the CFT. Most of the time, engineers are interested in the discrete Fourier transform only because it approximates the continuous Fourier transform. Most of the problems in using the DFT are caused by a misunderstanding of what this approximation involves.

In Fig. 7, the results of the DFT are treated as a corrupted estimate of the CFT. The example considers a cosine-wave input, but the results can be extended to any function expressed as a sum of sine and cosine waves.

Line (a) of Fig. 7 represents the input signal $s(t)$ and its continuous Fourier transform $S(f)$. Since $s(t)$ is shown to be a real cosine waveform, the continuous Fourier transform consists of two impulse functions that are symmetric about zero frequency.

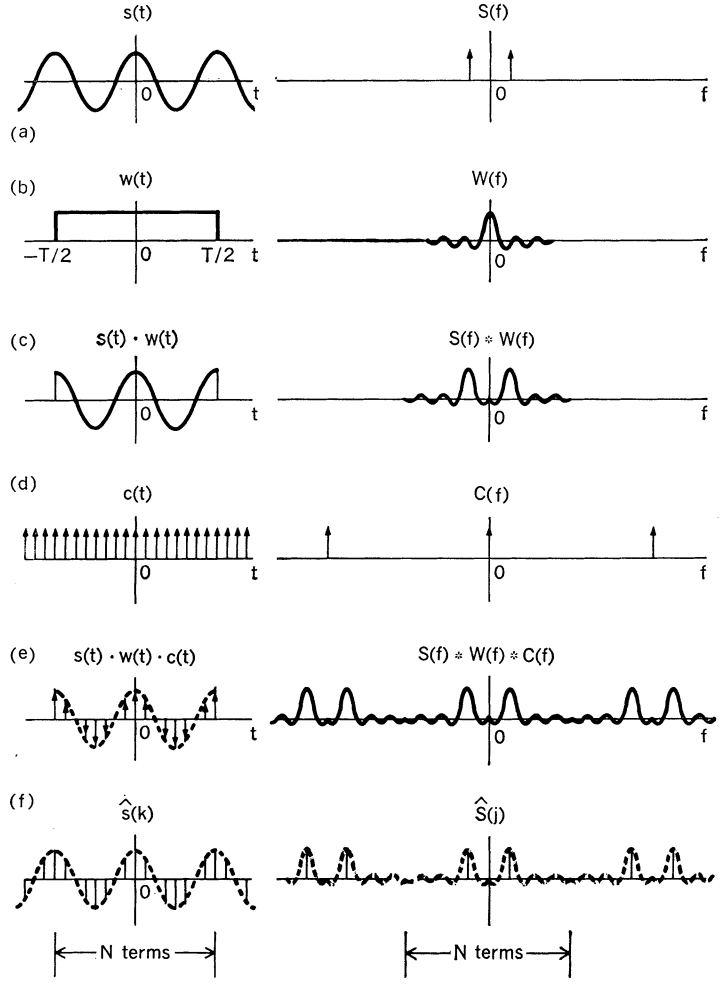
The finite portion of $s(t)$ to be analyzed is viewed through the unity amplitude data window $w(t)$. This rectangular data window has a continuous Fourier transform, which is in the form of a $(\sin x)/x$ function. [When this function takes the form $(\sin \pi x)/\pi x$, it is referred to as the sinc x function.[2]] The portion of $s(t)$ that will be analyzed is represented as the product of $s(t)$ and $w(t)$ in line (c) of Fig. 7. The corresponding convolution in the frequency domain results in a blurring of $S(f)$ into two $(\sin x)/x$-shaped pulses. Thus our estimate of $S(f)$ is already corrupted considerably.

The sampling of $s(t)$ is performed by multiplying by $c(t)$. (In Ref. 12 this infinite train of impulses is called a Dirac comb.) The resulting frequency-domain function is shown in line (e).

The continuous frequency-domain function shown in line (e) can also be made discrete if the time function is treated as one period of a periodic function. This assumption forces both the time-domain and frequency-domain functions to be infinite in extent, periodic, and discrete, as shown in line (f).

The discrete Fourier transform is simply a reversible mapping of N terms of $\hat{s}(k)$ into N terms of $\hat{S}(j)$. In this example, the N terms of $\hat{s}(k)$ and $\hat{S}(j)$ approximate $s(t)$ and $S(f)$ extremely well. This is an unusual case, however, in that the frequency-domain function of line (e) of Fig. 7 was sampled at exactly the peaks and zeros. The problems

FIGURE 7. The Fourier coefficients of the discrete Fourier transform viewed as a corrupted estimate of the continuous Fourier transform.



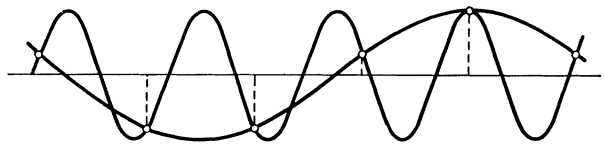


FIGURE 8. An example of a high frequency "impersonating" a low frequency.

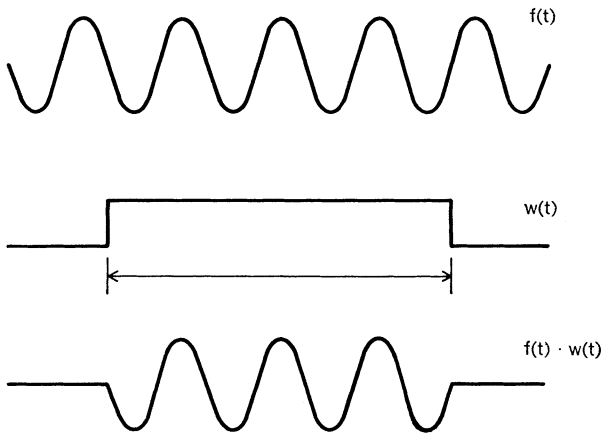
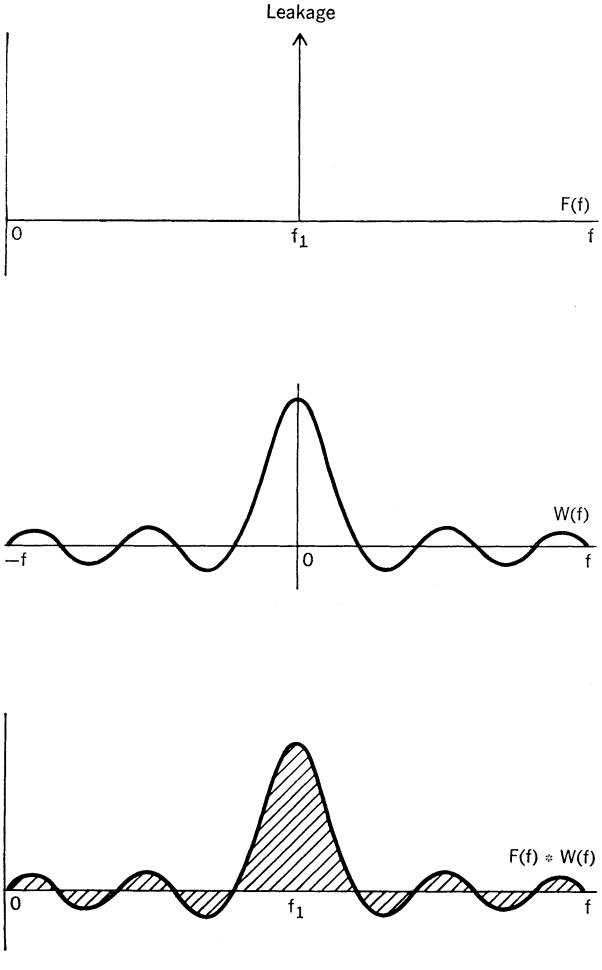


FIGURE 9. The rectangular data window implied when a finite record of data is analyzed.

FIGURE 10. The leakage of energy from one discrete frequency into adjacent frequencies resulting from the analysis of a finite record.



of leakage, aliasing, and picket-fence effects are associated with variations from this ideal condition.

Since both $\hat{s}(k)$ and $\hat{S}(j)$ are periodic, with a period N, any set of N adjacent terms can be used to characterize either set. The magnitude of $\hat{S}(j)$ is the same for samples of $\hat{s}(k)$ that are symmetric about the origin as it is for the DFT of a set of samples starting at the origin. The change in the phase of $\hat{S}(j)$ for different time origins can be determined by application of the DFT shifting theorem.[9]

Aliasing. The term "aliasing" refers to the fact that high-frequency components of a time function can impersonate low frequencies if the sampling rate is too low. This is demonstrated in Fig. 8 by showing a relatively high frequency and a relatively low frequency that share identical sample points. This uncertainty can be removed by demanding that the sampling rate be high enough for the highest frequency present to be sampled at least twice during each cycle.

In the diagram of Fig. 1, the sampling frequency f_s is shown to be $1/\Delta T$ samples per second. The folding frequency (or Nyquist frequency) f_f is shown to be equal to $f_s/2$. In this example, an input signal $x(t)$ will be represented correctly if its highest frequency component is less than the folding frequency. A frequency component 5 Hz higher than the folding frequency will impersonate a frequency 5 Hz lower than the folding frequency. Thus any components in $x(t)$ that are higher than the folding frequency are aliased (or folded) into the frequency interval below the folding frequency. In Fig. 7, aliasing occurs when $S(f)$ extends over a wider range of frequency than one period of $\hat{S}(j)$.

The cure to this problem involves sampling the signal at a rate at least twice as high as the highest frequency present. If the signal is known to be restricted to a certain band, the sampling rate can be picked accordingly. If the signal has been passed through a low-pass filter, a sampling rate can be chosen so that the components above the Nyquist frequency are negligible. Aliasing is discussed in more detail in Refs. 1–3, 10, and 12.

Leakage. The problem of leakage is inherent in the Fourier analysis of any finite record of data. The record has been formed by looking at the actual signal for a period of T seconds and by neglecting everything that happened before or after this period. As shown in Figs. 7 and 9, this is equivalent to multiplying the signal by a rectangular data window.

Had the continuous Fourier transform of the pure cosine wave in Fig. 9 been found, its contribution would have been limited to only one point on the frequency axis. (This is represented by the impulse at frequency f_1 in Fig. 10.) The multiplication by the data window in the time domain, however, is equivalent to performing a convolution in the frequency domain. Thus this impulse function is convolved with the Fourier transform of the square data window, resulting in a function with an amplitude of the $(\sin x)/x$ form centered about f_1.

This function is not localized on the frequency axis and in fact has a series of spurious peaks called sidelobes. The objective is usually to localize the contribution of a given frequency by reducing the amount of "leakage" through these sidelobes.

The usual approach consists of applying a data window to the time series, which has lower sidelobes in the frequency domain than the rectangular data window. This is

analogous to weighing the outputs of a linear antenna array to reduce the sidelobe levels of the antenna pattern.

A host of different data windows have been discussed in the literature; see Refs. 11, 12, 19, and 21. An example of Tukey's "interim" data window is shown in Fig. 11. In this window, a raised cosine wave is applied to the first and last 10 percent of the data and a weight of unity is applied in between. This window was suggested with reservations in Refs. 11 and 19. Since only 20 percent of the terms in the series are given a weight other than unity, the computation required to apply this window in the time domain is relatively small. Another window that can be applied conveniently is the Hanning window, described by Blackman and Tukey.[12] This window is a cosine bell on a pedestal, but it can be applied by convolving the DFT coefficients with the weights $-\frac{1}{4}$, $\frac{1}{2}$, and $-\frac{1}{4}$.[11]

When the computation required is not the overriding consideration, one can find a number of windows that give rise to more rapidly decreasing sidelobes than the cosine tapers described previously. An application of the Parzen window (which has a shape of $1 - |t|$, for $-1 \leq t \leq 1$) is described in Ref. 21, and an application of the Dolph–Chebyshev function is described in Ref. 31. The spectral window arising from an application of the Dolph–Chebyshev function has a principal lobe width that is as small as possible for a given sidelobe ratio and a given number of terms.[31]

Whatever the form of the data window, it should be applied only to the actual data and not to any artificial data generated by filling out a record with zeros.

Picket-fence effect. In Fig. 12, an analogy is drawn between the output of the FFT algorithm and a bank of bandpass filters. Although each Fourier coefficient would ideally act as a filter having a rectangular response in the frequency domain, the amplitude response is in fact of the form shown in Fig. 10 because of the multiplication of the input time series by the T-second data window. In Fig. 12, the main lobes of the resulting set of spectral windows have been plotted to represent the output of the FFT. The sidelobes are not shown here. The width of each main lobe is inversely proportional to the original record length T.

At the frequencies computed, these main lobes appear to be N independent filters; that is, a unity-amplitude complex exponential (of the form $e^{i\omega t}$) with a frequency that is an integral multiple of $1/T$, would result in a response of unity at the appropriate harmonic frequency and zero at all of the other harmonics.

The picket-fence effect becomes evident when the signal being analyzed is not one of these discrete orthogonal frequencies. A signal between the third and fourth harmonics, for example, would be seen by both the third- and fourth-harmonic spectral windows but at a value lower than one. In the worst case (exactly halfway between the computed harmonics) the amplitude of the signal is reduced to 0.637 in both of the spectral windows. When this result is squared, the apparent peak power of the signal is only 0.406. Thus the power spectrum seen by this "set of filters" has a ripple that varies by a factor of 2.5 to 1. One seems to be viewing the true spectrum through a picket fence.

A cure to this problem involves performing complex interpolation on the complex Fourier coefficients. This can be accomplished by the use of an interpolation function[29] or through modification of the DFT. The latter approach will be described here.

By extending the record analyzed with a set of samples that are identically zero, one can make the redundant FFT algorithm compute a set of Fourier coefficients with terms lying between the original harmonics. Since the width of the spectral window associated with each coefficient is related solely to the reciprocal of the true record length (T), the width of these new spectral windows remains unchanged. As shown in Fig. 13, this means that the spectral windows associated with this new set of Fourier coefficients overlap considerably.

If the original time series is represented by $g(k)$ for $k = 0, 1, \cdots, N - 1$, then the series analyzed in this example can be represented by $\hat{g}(k)$, where

$$\hat{g}(k) = g(k) \qquad \text{for } 0 \leq k < N$$
$$\hat{g}(k) = 0 \qquad \text{for } N \leq k < 2N \qquad (30)$$

The additional Fourier coefficients that result are interleaved with the original set.

As shown in Fig. 13, the ripple in the power spectrum has been reduced from approximately 60 percent to 20 percent. The ripple can be made larger or smaller than 20 percent by the use of less or more than N additional zeros.

In practice the picket-fence problem is not as great as this discussion implies. In many cases the signal being processed will not be a pure sinusoid but will be broad enough to fill several of the original spectral windows. Moreover, the use of any data window other than the rectangular (or boxcar) data window discussed here,

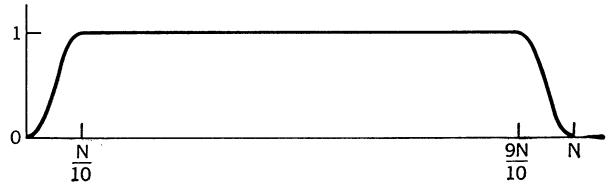
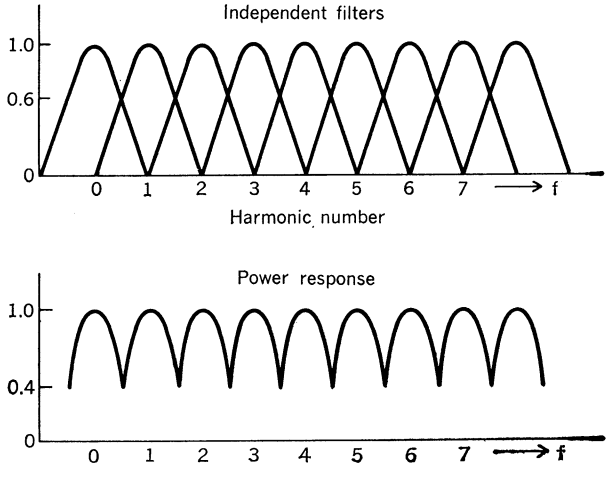


FIGURE 11. An extended cosine-bell data window.

FIGURE 12. The response of the discrete Fourier transform Fourier coefficients viewed as a set of bandpass filters (picket-fence effect).



usually tends to reduce the picket-fence effect by widening the main lobe of each spectral window.

Convolution and correlation. The blind use of Eqs. (22) and (28) to perform correlation and convolution is often inconvenient and usually incorrect. An example of incorrect usage is discussed first.

Given a function $g(t)$, one is frequently interested in convolving this function with another function $h(t)$. As shown in Fig. 14, this involves reversing (or flipping) $h(t)$, and sliding it by $g(t)$. The convolution of these functions is formed by computing and integrating the product $g(\tau)\cdot h(t-\tau)$ as a function of the relative displacement t. As indicated by Fig. 14, both functions are considered to be identically zero outside of their domain of definition. Where $g(t)$ and $h(t)$ are finite and sampled, this corresponds to the sliding, multiplying, and summing operations of Eq. (21).

When convolutions are computed with the aid of Eq. (22) and the fast Fourier transform, both of the functions are treated as being periodic. The corresponding forms of $g(\tau)$ and $h(t-\tau)$ are shown in Fig. 15. The result is a cyclical convolution, which is entirely different from the noncyclical convolution described previously.

Fortunately, this problem is easily sidestepped by simply defining and convolving $\hat{g}(\tau)$ and $\hat{h}(t-\tau)$ as shown in Fig. 16; see Refs. 7, 10, 23, 24, 28, and 31. In this case

$$\hat{g}(k) = g(k) \qquad 0 \leq k < N$$
$$\hat{g}(k) = 0 \qquad N \leq k < 2N \tag{31}$$

Thus the appropriate procedure is to

(1) Form $\hat{g}(k)$ and $\hat{h}(k)$.
(2) Find $\hat{G}(j)$ and $\hat{H}(j)$ via the FFT.
(3) Compute $\hat{C}(j) = \hat{G}(j)\cdot\hat{H}(j)$.
(4) Find $\hat{c}(k) = F^{-1}[\hat{C}(j)]$ via the FFT.

The series $\hat{c}(k)$ represents the $2N$-term series resulting from the convolution of the two N-term series $g(k)$ and $h(k)$. This technique is expressed in general terms in the section on "select-saving" in Ref. 24, and it is appropriate to correlation as well as convolution.

When one of the series convolved or correlated is much longer than the other, transforming the entire record of both functions is inconvenient and unnecessary.

If $h(k)$ is an N-term series and $g(k)$ is much longer, the procedure shown in Fig. 17 can be used. The $h(k)$ series can be thought of as the impulse response of a filter that is acting on the series $g(k)$. As shown in Fig. 14, the "present" output of the digital filter is merely a weighted sum of the last N samples it has seen. In the example of Fig. 17, this means that the $2N$-term series $h(k)$ convolved with one of the $2N$-term segments of $g(k)$ would result in N lags (or displacements), for which the correct N-term history is not available, and N lags for which the correct N-term history is available. Thus the first N lags computed are meaningless and the last N lags are correct.

In Fig. 17, overlapping sets of $2N$-term segments of $g(k)$ are shown. When convolved with $h(k)$, each of these segments contributes N correct terms to the final convolution. When all of these sets of N terms are pieced together, the result is the desired convolution of $g(k)$ and $h(k)$. Thus the length of the required FFTs is determined by the length of the short series rather than the long one. The length of the long series doesn't need to be specified.

Having exactly half of the values of $\hat{h}(\tau)$ be zero is a specific example of the "select-saving" method[24] and is made a requirement only to limit the discussion.

The segmenting technique used in convolving a short series with a long series can also be applied to the problem

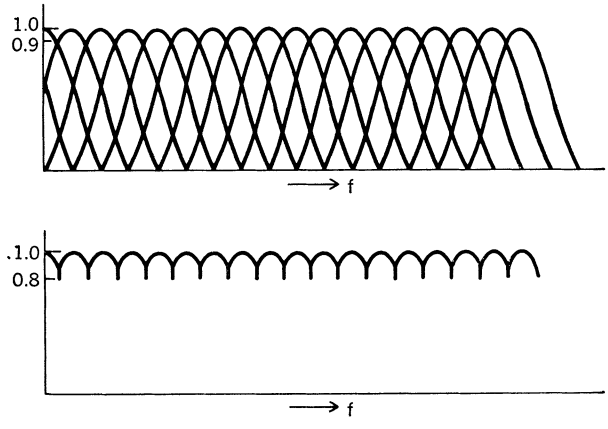


FIGURE 13. The reduction of the picket-fence effect, brought about by computing redundant overlapping sets of Fourier coefficients.

FIGURE 14. The procedure for performing noncyclical convolution on two finite signals.

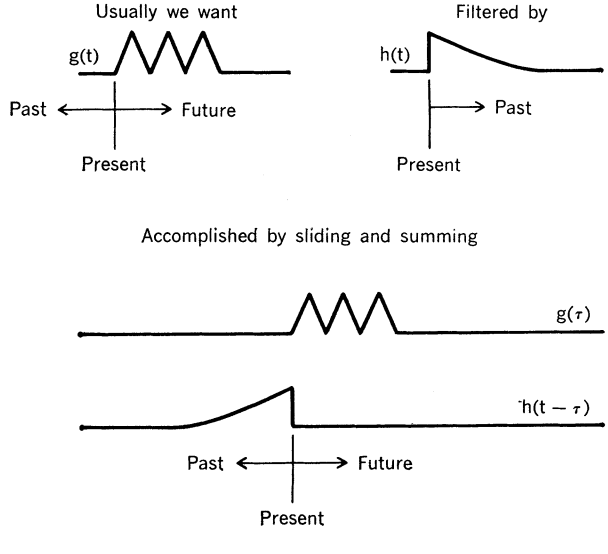
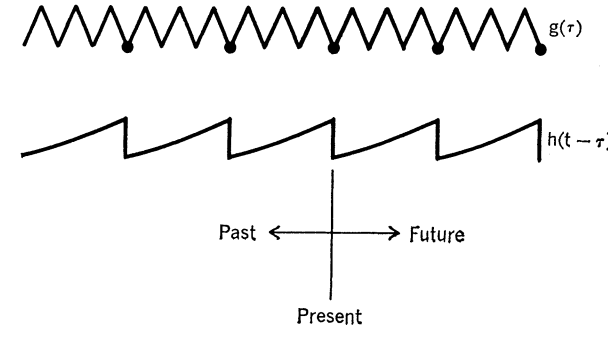


FIGURE 15. The cyclical convolution of two finite signals analogous to that performed by the FFT algorithm.



of computing N lags of the autocorrelation function of an M-term series when $N \ll M$. Figure 18 shows a function $g(t)$, which can be sampled to form an M-term time series. To compute N lags of the autocorrelation function of $g(t)$, this series is broken into overlapping segments of at least $2N$ terms. For each segment, the functions $\hat{g}(t)$ and $\hat{\hat{g}}(t)$ can be formed and sampled such that

$$\hat{g}(k) = g(k) \qquad 0 \leq k < 2N \tag{32}$$

$$\hat{\hat{g}}(k) = g(k) \qquad 0 \leq k < N$$

$$= 0 \qquad N \leq k < 2N \tag{33}$$

An appropriate procedure is to

(1) Form $\hat{G}(j)$ and $\hat{\hat{G}}(j)$ via the FFT.

(2) Compute $\hat{C}(j) = \hat{G}(j) \cdot \hat{\hat{G}}^*(j)$.

(3) Find $\hat{c}(k) = F^{-1}[\hat{C}(j)]$ via the FFT.

The first half of the $c(k)$ series represents the contribution of the first half of the $2N$-term segment to the first N lags of the autocorrelation function of $g(k)$. When the contributions from all of the segments are added together, the result is the first N lags of the autocorrelation function of all M terms of $g(k)$. This technique, called "overlap-adding," is described in detail by Helms.[24]

The choice of $2N$ zeros is made to limit the present discussion and can be changed. The optimum choice with respect to computational effort is discussed in Refs. 22 and 24. It is also a simple matter to compute any set of N lags and to apply this same technique to computing cross-correlation functions.

Frequency-domain filter design. Given the capability for convenient and efficient digital filtering, one possesses considerable degree of freedom in specifying the form of the filter. It becomes very tempting to make use of the ideal low-pass, high-pass, and bandpass filters that are so difficult to come by in the real world. This should not be done, however, without exercising great caution.

When a digital filter is specified in the frequency domain, this is equivalent to multiplying the Fourier coefficients by a window. This multiplication in the frequency domain is equivalent to performing a convolution in the time domain. Thus, the constraints discussed previously still apply but are not as readily apparent. For this discussion, the constraint is that at least half of the impulse response implied by the frequency domain filter be identically zero.

When an ideal filter, such as $H(f)$ in Fig. 19, is specified in the frequency domain, this implies a $(\sin x)/x$ impulse response in the time domain. In this example of an ideal low-pass filter, $h(t)$ does not go to zero, meaning that the tails of the $(\sin x)/x$ impulse response fold back into the T-second region. The result is aliasing in the time domain caused by undersampling the filter function in the frequency domain. This is directly analogous to the aliasing in the frequency domain caused by undersampling in the time domain.

The cure to this problem is always to choose a filter with an impulse response that dies out or can be truncated so that at least half of its terms are essentially zero. Two procedures based on truncating the impulse response are described in detail by Helms[30,31]; see also Refs. 32 and 33. In specifying a filter in the frequency domain, remember that a filter with a real impulse response implies a set of Fourier coefficients whose real part is an even function and whose imaginary part is an odd function. In the fast Fourier transform format, this implies that the real part be symmetric about the folding frequency [the (N/2)th harmonic] and the imaginary part be

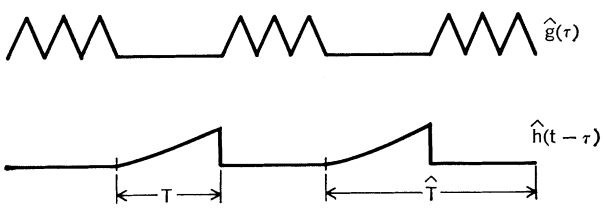


FIGURE 16. Noncyclical convolution of two finite signals analogous to that performed by the FFT algorithm.

FIGURE 17. A method for convolving a finite impulse response with an infinite time function by performing a series of fast Fourier transforms.

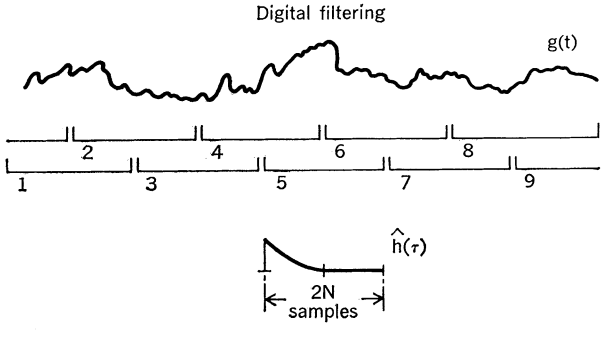
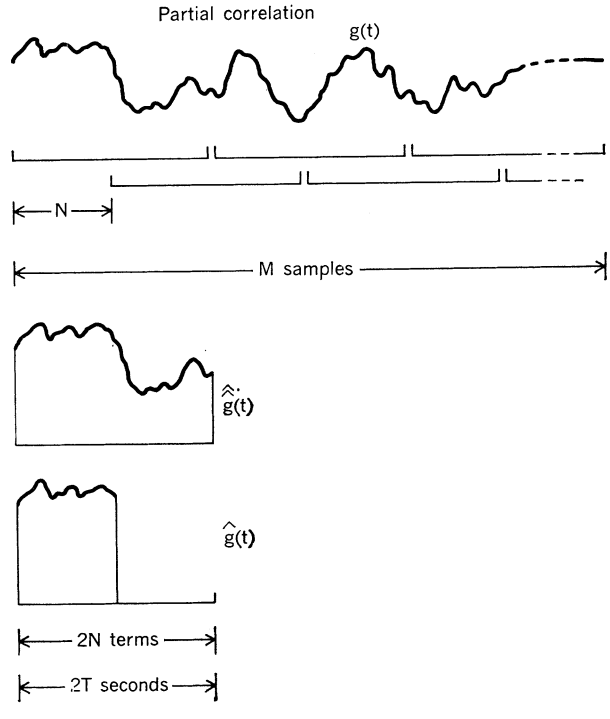


FIGURE 18. A method of using the fast Fourier transform algorithm to compute N lags of the autocorrelation function of an M-term series.



antisymmetric, as shown in the example of Fig. 1.

Applications. An example of a radar signal processing system is shown in the top half of Fig. 20. A chirped radar pulse, $s(t)$, is shown entering a matched dechirping filter followed by a Taylor weighting network.[62] In practice the input signal is corrupted by noise, the atmosphere, the equipment, etc., and the problem is to determine the effect this has on system performance.[61] In Fig. 20 this degradation is expressed in the form of a weighting function applied to $s(t)$ before it enters the matched filter. In this example the weighting is of the form $1 - x(t)$.

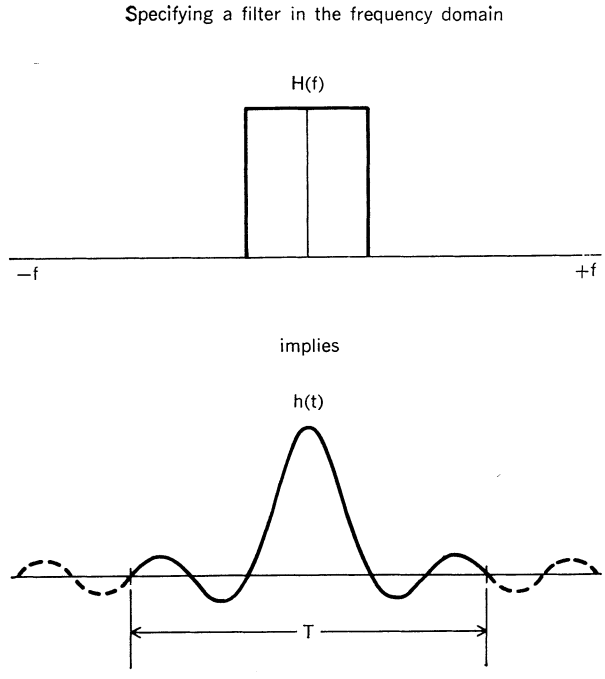
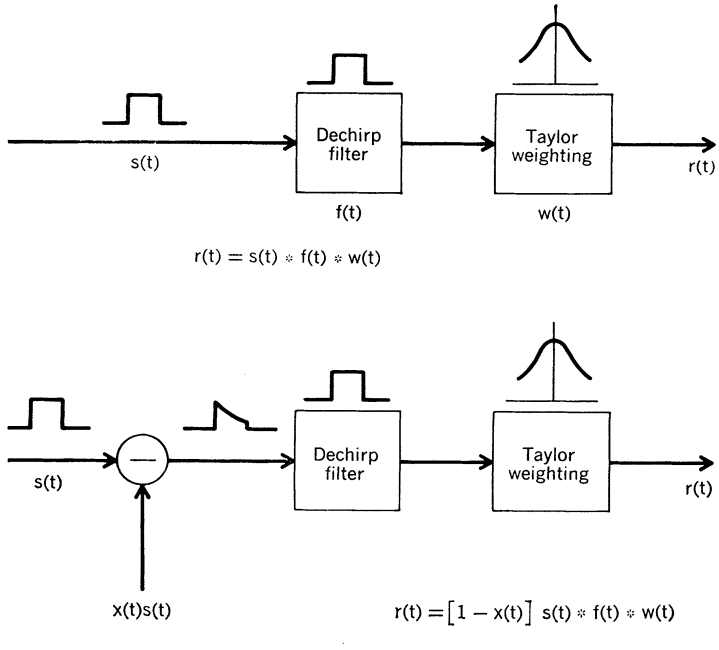


FIGURE 19. The (sin x)/x impulse response of a filter implied when a rectangular frequency response is specified.

FIGURE 20. Block diagrams illustrating the use of the FFT for simulating a radar signal processing system.



For our discussion, the matched filter can be viewed as finding the autocorrelation function of the chirped signal, and the Taylor weighting can be viewed as applying a weighting function in the frequency domain. The autocorrelation can be thought of as convolving the waveform $s(t)$ with itself reversed in time—that is, with $f(t)$. The multiplication by the Taylor weighting coefficients in the frequency domain corresponds to convolving the result of the first convolution with the impulse response of the Taylor weighting network $w(t)$. Thus the response of the system, $r(t)$, is actually the result of performing two convolutions. If $s(t)$, $f(t)$, and $w(t)$ can all be represented by N-term series, they each should be augmented with $3N$ zeros to assure that the final convolution is noncyclical.

The application of the fast Fourier transform to radar and sonar ranging systems can be viewed in terms of performing a correlation. Since correlation can be viewed as one function searching to find itself in another, it is clear how echo-ranging systems using a chirped or even a random-noise signal can operate. Since these correlations can now be done by means of the FFT, many of these operations can be performed digitally in real time.

How has it been implemented?

Software. The number of variations of the FFT algorithm appears to be directly proportional to the number of people using it.[34–53] Most of these algorithms are based on either the Cooley–Tukey or the Sande–Tukey algorithm,[11] but are formulated to exploit properties of the series being analyzed or properties of the computer being used.

Equations (17) through (20) represent the Cooley–Tukey formulation of the FFT algorithm. By separating the components of j instead of the components of k in Eq. (11), the Sande–Tukey equations would have resulted. In either case, the recursive equations can be written in the form of a two-point transform followed by a rereferencing (or twiddling[9]) operation. The resulting counterparts of Eqs. (17)–(20) are as follows:

$$\hat{A}_1(j_0,k_1,k_0) = \left\{\sum_{k_2=0}^{1} A(k_2,k_1,k_0)W_2^{j_0k_2}\right\} W_4^{j_0k_1} \quad (34)$$

$$\hat{A}_2(j_0,j_1,k_0) = \left\{\sum_{k_1=0}^{1} \hat{A}_1(j_0,k_1,k_0)W_2^{j_1k_1}\right\} W_8^{(j_12+j_0)k_0} \quad (35)$$

$$\hat{A}_3(j_0,j_1,j_2) = \left\{\sum_{k_0=0}^{1} \hat{A}_2(j_0,j_1,k_0)W_2^{j_2k_0}\right\} \quad (36)$$

$$\hat{X}(j_2,j_1,j_0) = \hat{A}_3(j_0,j_1,j_2) \quad (37)$$

The extension of the FFT algorithm from radix-two algorithms to arbitrary-radix algorithms is accomplished by representing the j and k variables of Eq. (8) in a mixed radix number system.[34,51] For the example of $N = r_1r_2r_3$, they take the form

$$\begin{aligned} j &= j_2(r_1r_2) + j_1(r_1) + j_0 \\ k &= k_2(r_2r_3) + k_1(r_3) + k_0 \end{aligned} \quad (38)$$

where $j_0, k_2 = 0, 1, \cdots, r_1 - 1$; $j_1, k_1 = 0, 1, \cdots, r_2 - 1$; and $j_2, k_0 = 0, 1, \cdots, r_3 - 1$. The resulting recursive equations can functionally be separated into r_1 point transforms, r_2 point transforms, r_3 point transforms, and rereferencing operations.

Since four-point transforms can be performed with only additions and subtractions, an algorithm with a large number of factors that are four requires less computation than a radix-two algorithm.[44] In much the same manner, a further reduction can be made by forcing the algorithm into a form where a large number of eight-point transforms can be done very efficiently.[35] The number of multiplications required by the resulting algorithms is reduced by 30 percent and 40 percent, respectively, compared with radix-two algorithms.

Several variants of the FFT algorithm have been motivated by characteristics of the series being transformed. When the series is real, the expected two-to-one reduction in computation and storage can be obtained by either putting the N-term real record in the form of an artificial $N/2$-term complex record,[7] or by restructuring the FFT algorithm.[36]

An algorithm that allows the value of N to take on any value (including a prime number) has recently been proposed.[25,38] For any value of N, the discrete Fourier transform can be written

$$X(j) = \frac{1}{N}\sum_{k=0}^{N-1} x(k)W^{-jk} \tag{39}$$

$$X(j) = \frac{1}{N}\sum_{k=0}^{N-1} x(k)W^{-jk+[(k^2-k^2+j^2-j^2)/2]} \tag{40}$$

$$X(j) = \frac{W^{-j^2/2}}{N}\left\{\sum_{k=0}^{N-1}[W^{-k^2/2}x(k)]W^{(j-k)^2/2}\right\} \tag{41}$$

Note that Eq. (41) is in the form of a convolution; i.e.,

$$X(j) = \frac{W^{-j^2/2}}{N}\sum_{k=0}^{N-1} g(k)h(j-k) \tag{42}$$

where $g(k) = W^{-k^2/2}x(k)$, and $h(j-k) = W^{(j-k)^2/2}$. In performing this convolution, both of the series can be augmented with zeros until they contain a highly factorable number of terms $\hat{N}$. These extended series can be transformed by a conventional $\hat{N}$-term FFT algorithm to obtain the noncyclical convolution required by Eq. (42). The Fourier coefficients corresponding to the original N-term DFT of Eq. (39) are obtained by multiplying the results of this convolution by the unit amplitude complex exponential $W^{-j^2/2}$, where $W = \exp(2\pi i/N)$.

This algorithm should be very useful when employed with a hardware FFT processor that otherwise would be restricted to values of N that are powers of two.

A variety of different FFT programs for performing one-dimensional and multidimensional fast Fourier transforms have been made available by Cooley,[41,42] Sande,[48] Singleton,[49-52] and Brenner,[39] who have programmed most of the options described here plus several others as well.

Hardware. The advent of the FFT algorithm reduced the time required for performing a 2^{10}-point discrete Fourier transform from several minutes to less than a second. The advent of special-purpose digital hardware further reduced this time to tens of milliseconds.[54-60]

The cost of finding a 2^{10}-point transform used to be measured in dollars. The FFT algorithm reduced this cost to a few cents. Special-purpose hardware has reduced it further to hundredths of a cent.

Thus in the past five years, both the cost and execution time of computing a discrete Fourier transform have been reduced by nearly four orders of magnitude. Therefore, we can now build a relatively inexpensive special-purpose processor that is able to meet the real-time constraints of a wide variety of signal-processing problems.

A survey of FFT processors and their characteristics is reported in Ref. 55. Most of these processors can be classified in one of the four families of FFT processor machine organizations discussed in Ref. 56. These four families represent varying degrees of parallelism, performance, and cost.

Most of the hardware implementations built to date have relied on either a radix-two or a radix-four algorithm. The regularity of these algorithms and the resulting simplification in control have tended to discourage the use of the more general arbitrary-radix algorithms. In addition, most users of real-time FFT hardware seem to have adapted quickly to thinking in powers of two.

The author would like to thank D. E. Wilson for the loan of the title; R. A. Kaenel and W. W. Lang for their encouragement; and B. P. Bogert, W. T. Hartwell, H. D. Helms, C. M. Rader, and P. T. Rux for suggesting several improvements, which were incorporated in this article.

REFERENCES

Introduction to Fourier analysis

1. Arsac, J., *Fourier Transforms.* Englewood Cliffs, N.J.: Prentice-Hall, 1966.
2. Bracewell, R., *The Fourier Transform and Its Applications.* New York: McGraw-Hill, 1965.
3. Papoulis, A., *The Fourier Integral and Its Applications.* New York: McGraw-Hill, 1962.

Historical development of the fast Fourier transform

4. Cooley, J. W., Lewis, P. A. W., and Welch, P. D., "Historical notes on the fast Fourier transform," *IEEE Trans. Audio and Electroacoustics*, vol. AU-15, pp. 76–79, June 1967.

Introduction to the fast Fourier transform

5. Brigham, E. O., and Morrow, R. E., "The fast Fourier transform," *IEEE Spectrum*, vol. 4, pp. 63–70, Dec. 1967.
6. Cochran, W. T., *et al.*, "What is the fast Fourier transform?" *IEEE Trans. Audio and Electroacoustics*, vol. AU-15, pp. 45–55, June 1967.
7. Cooley, J. W., Lewis, P. A. W., and Welch, P. D., "The fast Fourier transform algorithm and its applications," IBM Research Paper RC-1743, Feb. 1967.
8. Cooley, J. W., and Tukey, J. W., "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, pp. 297–301, Apr. 1965.
9. Gentleman, W. M., and Sande, G., "Fast Fourier transforms—for fun and profit," *1966 Fall Joint Computer Conf., AFIPS Proc.*, vol. 29. Washington, D.C.: Spartan Books.
10. Gold, B., and Rader, C. M., *Digital Processing of Signals.* New York: McGraw-Hill, 1969.

Spectrum and cepstrum analysis

11. Bingham, C., Godfrey, M. D., and Tukey, J. W., "Modern techniques of power spectrum estimation," *IEEE Trans. Audio and Electroacoustics*, vol. AU-15, pp. 56–66, June 1967.
12. Blackman, R. B., and Tukey, J. W., *The Measurement of Power Spectra.* New York: Dover, 1958.
13. Bogert, B. P., Healy, M. J. and Tukey, J. W., "The frequency analysis of time series for echoes: cepstrum, pseudoautocovariance, cross-cepstrum and saphe-cracking," *Time Series Analysis*, Murray Rosenblatt, ed. New York: Wiley, 1963, pp. 201–243.
14. Noll, A. Michael, "Short-time spectrum and 'cepstrum' techniques for vocal-pitch detection," *J. Acoust. Soc. Am.*, vol. 36, pp. 296–302, 1964.
15. Oppenheim, A. V., Schafer, R. W., and Stockham, T. G., Jr., "Nonlinear filtering of multiplied and convolved signals," *Proc. IEEE*, vol. 56, pp. 1264–1291, Aug. 1968. (Reprinted in *IEEE Trans. Audio and Electroacoustics*, vol. AU-16, pp. 437–465, Sept. 1968.)
16. Parzen, E., "Statistical spectral analysis (single channel case) in 1968," Tech. Rep. 11, ONR Contract Nonr-225(80)(NR-042-234), Stanford University, Dept. of Statistics, Stanford, Calif., June 10, 1968.
17. Richards, P. I., "Computing reliable power spectra," *IEEE Spectrum*, vol. 4, pp. 83–90, Jan. 1967.

18. Tukey, J. W., "An introduction to the measurement of spectra," in *Probability and Statistics*, Ulf Grenander, ed. New York: Wiley, 1959, pp. 300–330.

19. Tukey, J. W., "An introduction to the calculations of numerical spectrum analysis," in *Spectral Analysis of Time Series*, Bernard Harris, ed. New York: Wiley, 1967, pp. 25–46.

20. Welch, P. D., "A direct digital method of power spectrum estimation," *IBM J. Res. Develop.*, vol. 5, pp. 141–156, 1961.

21. Welch, P. D., "The use of the fast Fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms," *IEEE Trans. Audio and Electroacoustics*, vol. AU-15, pp. 70–73, June 1967.

Use of the fast Fourier transform in convolution, correlation, digital filtering, etc.

22. Cooley, J. W., "Applications of the fast Fourier transform method," *Proc. IBM Scientific Computing Symp. on Digital Simulation of Continuous Systems*, Thomas J. Watson Research Center, Yorktown Heights, N.Y., 1966.

23. Cooley, J. W., Lewis, P. A. W., and Welch, P. D., "Application of the fast Fourier transform to computation of Fourier integrals, Fourier series, and convolution integrals," *IEEE Trans. Audio and Electroacoustics*, vol. AU-15, pp. 79–84, June 1967.

24. Helms, H. D., "Fast Fourier transform method of computing difference equations and simulating filters," *IEEE Trans. Audio and Electroacoustics*, vol. AU-15, pp. 85–90, June 1967.

25. Rabiner, L. R., Schafer, R. W., and Rader, C. M., "The chirp *Z*-transform algorithm and its applications," *Bell System Tech. J.*, vol. 48, pp. 1249–1292, May–June 1969.

26. Sande, G., "On an alternative method of calculating covariance functions," unpublished technical note, Princeton University, Princeton, N.J., 1965.

27. Singleton, R. C., "Algorithm 345, an Algol convolution procedure based on the fast Fourier transform," *Commun. Assoc. Comput. Mach.*, vol. 12, Mar. 1969.

28. Stockham, T. G., "High-speed convolution and correlation," *1966 Spring Joint Computer Conf., AFIPS Proc.*, vol. 28. Washington, D.C.: Spartan Books, pp. 229–233.

The picket-fence effect

29. Hartwell. W. T., "An alternate approach to the use of discrete Fourier transforms," to be published.

Frequency-domain filter design

30. Helms, H. D., "Fast Fourier transform method for computing difference equations and simulating filters," *IEEE Trans. Audio and Electroacoustics*, vol. AU-15, pp. 85–90, June 1967.

31. Helms, H. D., "Nonrecursive digital filters: design methods for achieving specifications on frequency response," *IEEE Trans. Audio and Electroacoustics*, vol. AU-16, pp. 336–342, Sept. 1968.

32. Kaiser, J. F., "Digital filters," in *System Analysis by Digital Computer*, F. F. Kuo and J. F. Kaiser, eds. New York: Wiley, 1966.

33. Otnes, R. K., "An elementary design procedure for digital filters," *IEEE Trans. Audio and Electroacoustics*, vol. AU-16, pp. 336–342, Sept. 1968.

Algorithms and software

34. Bergland, G. D., "The fast Fourier transform recursive equations for arbitrary length records," *Math. Comput.*, vol. 21, pp. 236–238, Apr. 1967.

35. Bergland, G. D., "A fast Fourier transform algorithm using base 8 iterations," *Math. Comput.*, vol. 22, pp. 275–279, Apr. 1968.

36. Bergland, G. D., "A fast Fourier transform algorithm for real-valued series," *Commun. Assoc. Comput. Mach.*, vol. 11, pp. 703–710, Oct. 1968.

37. Bergland, G. D., and Wilson, D. E., "An FFT algorithm for a global, highly parallel processor," *IEEE Trans. Audio and Electroacoustics*, vol. AU-17, June 1969.

38. Bluestein, L. I., "A linear filtering approach to the computation of the discrete Fourier transform," *1968 NEREM Record*, pp. 218–219.

39. Brenner, N. M., "Three Fortran programs that perform the Cooley–Tukey Fourier transform," Tech. Note 1967-2, Lincoln Laboratory, M.I.T., Lexington, Mass., July 1967.

40. Cooley, J. W., and Tukey, J. W., "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, pp. 297–301, Apr. 1965.

41. Cooley, J. W., "Harmonic analysis complex Fourier series," SHARE Doc. 3425, Feb. 7, 1966.

42. Cooley, J. W., "Complex finite Fourier transform subroutine," SHARE Doc. 3465, Sept. 8, 1966.

43. Danielson, G. C., and Lanczos, C., "Some improvements in practical Fourier analysis and their application to X-ray scattering from liquids," *J. Franklin Inst.*, vol. 233, pp. 365–380, 435–452.

44. Gentleman, W. M., and Sande, G., "Fast Fourier transforms—for fun and profit," *1966 Fall Joint Computer Conf., AFIPS Proc.*, vol. 29. Washington, D.C.: Spartan Books.

45. Good, I. J., "The interaction algorithm and practical Fourier series," *J. Roy. Statist. Soc.*, vol. 20, series B, pp. 361–372, 1958; addendum, vol. 22, pp. 372–375, 1960.

46. Pease, M. C., "An adaption of the fast Fourier transform for parallel processing," *J. Assoc. Comput. Mach.*, vol. 15, pp. 252–264, Apr. 1968.

47. Rader, C. M., "Discrete Fourier transforms when the number of data samples is prime," *Proc. IEEE*, vol. 56, pp. 1107–1108, June 1968.

48. Sande, G., "Arbitrary radix one-dimensional fast Fourier transform subroutines," University of Chicago, Ill., 1968.

49. Singleton, R. C., "On computing the fast Fourier transform," *Commun. Assoc. Comput. Mach.*, vol. 10, pp. 647–654, Oct. 1967.

50. Singleton, R. C., "Algorithm 338, Algol procedures for the fast Fourier transform," *Commun. Assoc. Comput. Mach.*, vol. 11, pp. 647–654, Nov. 1968.

51. Singleton, R. C., "Algorithm 339, an Algol procedure for the fast Fourier transform with arbitrary factors," *Commun. Assoc. Comput. Mach.*, vol. 11, pp. 776–779, Nov. 1968.

52. Singleton, R. C., "Algorithm 345, an Algol convolution procedure based on the fast Fourier transform," *Commun. Assoc. Comput. Mach.*, vol. 12, Mar. 1969.

53. Yavne, R., "An economical method for calculating the discrete Fourier transform," *1968 Fall Joint Computer Conf., IFIPS Proc.*, vol. 33. Washington, D.C.: Spartan Books, pp. 115–125.

Fast Fourier transform hardware

54. Bergland, G. D., and Hale, H. W., "Digital real-time spectral analysis," *IEEE Trans. Electronic Computers*, vol. EC-16, pp. 180–185, Apr. 1967.

55. Bergland, G. D., "Fast Fourier transform hardware implementations—a survey," *IEEE Trans. Audio and Electroacoustics*, vol. AU-17, June 1969.

56. Bergland, G. D., "Fast Fourier transform hardware implementations—an overview," *IEEE Trans. Audio and Electroacoustics*, vol. AU-17, June 1969.

57. McCullough, R. B., "A real-time digital spectrum analyzer," Stanford Electronics Laboratories Sci. Rept. 23, Stanford University, Calif., Nov. 1967.

58. Pease, M. C., III, and Goldberg, J., "Feasibility study of a special-purpose digital computer for on-line Fourier analysis," Order No. 989, Advanced Research Projects Agency, Washington, D.C., May 1967.

59. Shively, R. R., "A digital processor to generate spectra in real time," *IEEE Trans. Computers*, vol. C-17, pp. 485–491, May 1968.

60. Smith, R. A., "A fast Fourier transform processor," Bell Telephone Laboratories, Inc., Whippany, N.J., 1967.

Other

61. Gilbert, S. M., Private communication, Bell Telephone Laboratories, Inc., Whippany, N.J.

62. Klauder, J. R., Price, A. C., Darlington, S., and Albersheim, W. J., "The theory and design of chirp radars," *Bell System Tech. J.*, vol. 39, pp. 745–808, July 1960.

G. D. Bergland (M) received the B.S., M.S., and Ph.D. degrees from Iowa State University in 1962, 1964, and 1966, respectively. While at Iowa State he was a teaching assistant in the Electrical Engineering Department and a research assistant in the Engineering Experiment Station. From 1964 to 1966 he held a National Science Foundation traineeship. In 1966 he joined the Digital Systems Laboratory at Bell Telephone Laboratories, Inc., Whippany, N.J., where he conducted research in the area of special-purpose computer organizations. Since 1968 he has been the supervisor of the Computer Systems Studies Group, which is involved in the study and evaluation of real-time applications of special-purpose data-processing techniques. He is a member of Sigma Xi, Phi Kappa Phi, Eta Kappa Nu, and Tau Beta Pi.