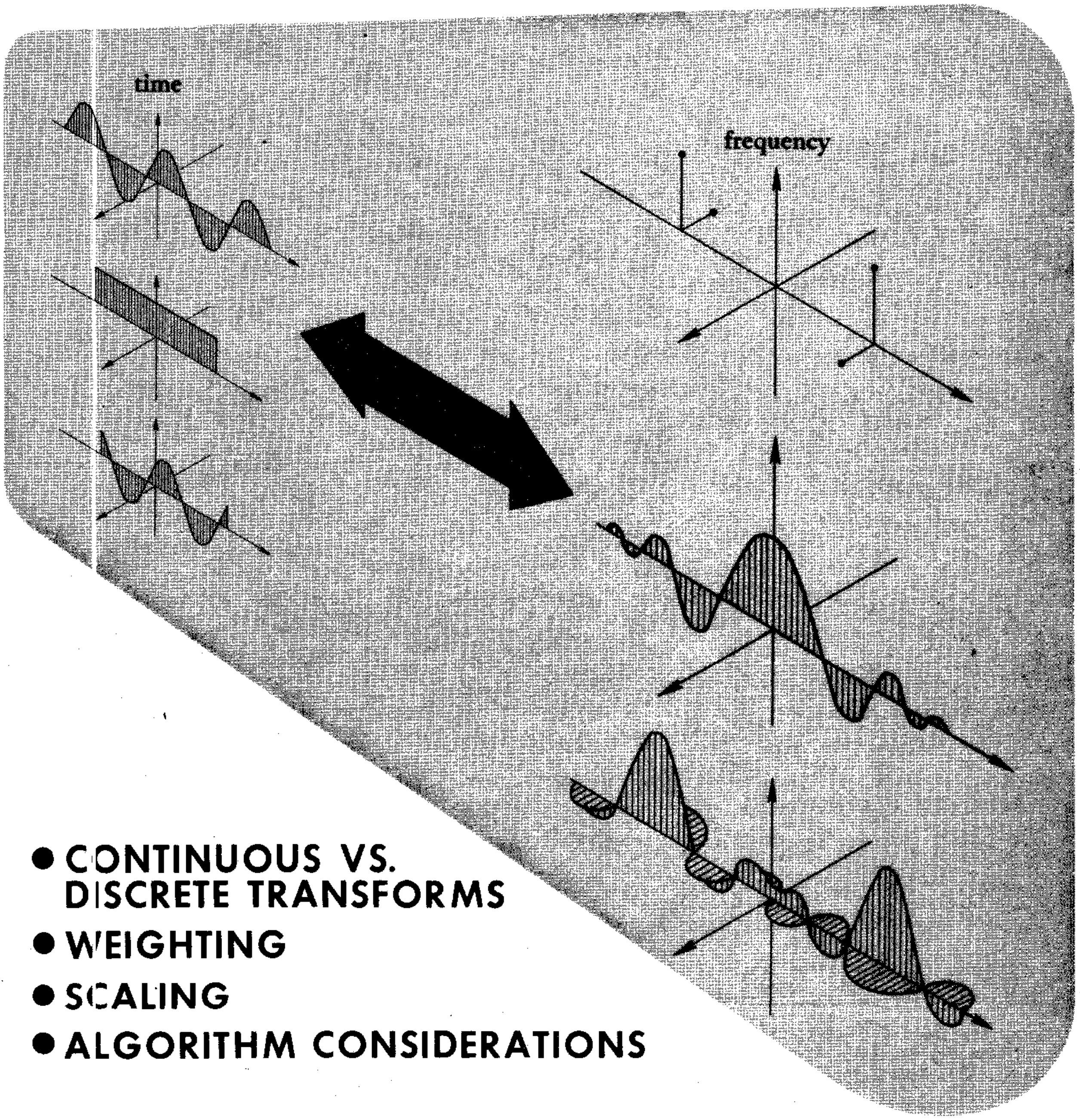
R Jeppesen

TECHNICAL PUBLICATION DSP-005 (8-81)

Trigonometric Transforms

a unique introduction to the FFT

by frederic j harris



- CONTINUOUS VS. DISCRETE TRANSFORMS
- WEIGHTING
- SCALING
- ALGORITHM CONSIDERATIONS

Scientific-Atlanta

Spectral Dynamics Division

United States: P.O. Box 23575 • San Diego, CA 92123-0575 • (619) 268-7100 • TWX 910-335-2022
Europe: Horton Manor • Stanwell Road • Horton, Slough, SL3 9PA, England • Telephone: Colnbrook 3211 • Telex: 849833

TRIGONOMETRIC TRANSFORMS

by
frederic j. harris
San Diego State University

SINE WAVES

To start our review of trigonometric transforms we will present a convenient and often illuminating representation of the sinusoid and of the co-sinusoid waveforms. We first recognize that there is a one-to-one correspondence between an ordered pair of real numbers, a point on a plane, and a vector leading to that point on the plane. We will reserve the option to use any of these interchangeably as the need arises to aid our understanding of the transforms.

Let us examine the point on the plane $R \bullet e^{+j\theta}$. As indicated in Figure 1, the point can be decomposed into components in each of the two coordinates, X and Y, where

$$X = R \cos \theta$$

$$Y = R \sin \theta.$$

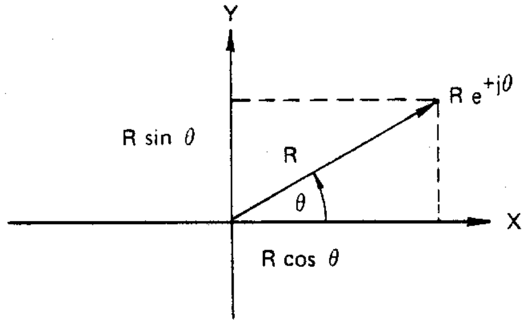


Figure 1. Mapping of the Point $R\ e^{+j\theta}$

We use the standard convention that positive angles are measured counterclockwise (CCW) with respect to the X axis, and θ is identified as an angle in radians if it appears as the argument of a trigonometric function.

We sometimes denote the vector (or point) in the form of an ordered pair of real numbers of the form

$$R\ e^{+j\theta} = [R \cos(\theta), R \sin(\theta)].$$

Recognizing that the order is important, we sometimes label the ordering and call the complex pair a complex number of the form

$$R\ e^{+j\theta} = R \cos(\theta) + j\ R \sin(\theta).$$

Note the "j" serves as a label, identifying the component associated with the second position of the ordered pair.

We also observe that $R\ e^{-j\theta}$ can be represented by

$$R\ e^{-j\theta} = R \cos(-\theta) + j\ R \sin(-\theta)$$

$$= R \cos(\theta) - j\ R \sin(\theta),$$

which is indicated in Figure 2.

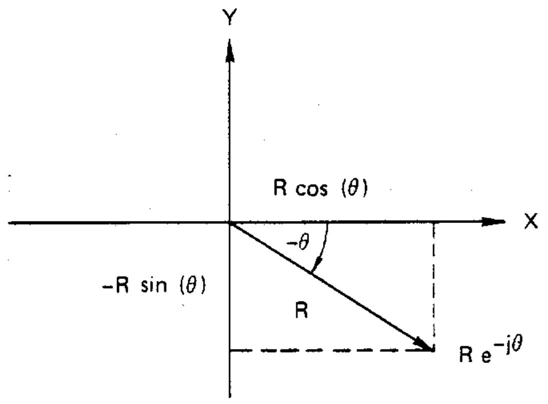


Figure 2. Mapping of the Point $R\ e^{-j\theta}$

It may happen that θ is an angle that varies linearly with time, that is, $\theta = \omega t$, so that at different points in time the point (or vector) identified by $R\ e^{j\omega t}$ will be found at different places on the plane. This is indicated in Figure 3.

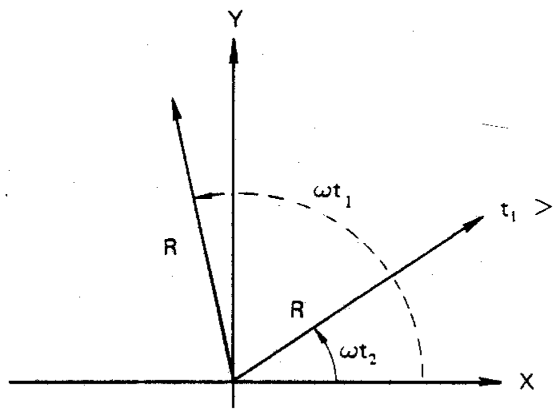


Figure 3. Mapping of the Point $R\ e^{j\omega t}$ for distinct values of t

The components into which we can decompose the point $R\ e^{j\omega t}$ are also time varying, that is,

$$R\ e^{j\omega t} = R \cos(\omega t) + j\ R \sin(\omega t)$$

Classically, a vector is not allowed to spin, so we call a spinning vector, a phasor. But for our discussion, we will use the rather descriptive name of rotating vector and we will talk about $R\ e^{j\omega t}$ as a vector of length R and spinning CCW at a fixed rate of ω radians per second. We observe the vectors $R\ e^{+j\omega t}$ and $R\ e^{-j\omega t}$ spin in opposite directions. By simple vector addition, we compute the sum and difference of these counter-rotating vectors as

$$R\ e^{j\omega t} + R\ e^{-j\omega t} = 2\ R \cos(\omega t)$$

$$R\ e^{j\omega t} - R\ e^{-j\omega t} = 2jR \sin(\omega t),$$

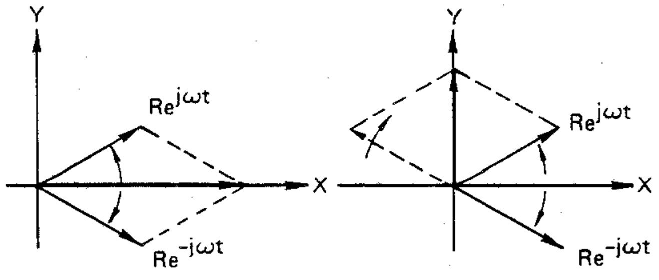


Figure 4. Sum and Difference of $R\ e^{j\omega t}$ and $R\ e^{-j\omega t}$

from which we have Eulers' form of the trigonometric functions

$$\cos(\omega t) = \frac{e^{j\omega t} + e^{-j\omega t}}{2}$$

$$\sin(\omega t) = \frac{e^{j\omega t} - e^{-j\omega t}}{2j}.$$

From this point, when we say cosine or sine function, we should visualize a pair of counter-rotating vectors. Thus A cos (ωt) is a pair of vectors, each of length A/2, one rotating CCW (from positive real to positive imaginary), the other rotating CW (from positive real to negative imaginary). A very convenient scoreboard to assist in this visualization is presented in Figure 5.

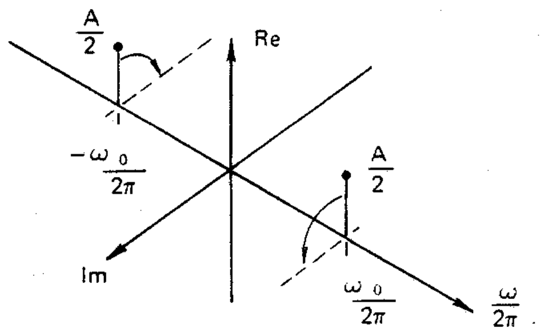


Figure 5. Scoreboard representation of A $\cos(\omega t)$

The position of the vectors on the $\omega/2\pi$ axis indicates the rate of spin. Thus the vector at $\omega_0/2\pi$ is a vector of length A/2, spinning in a CCW direction at a rate of ω_0 radians/second, and the vector at $-\omega_0/2\pi$ is the same length vector spinning in a CW direction at the same rate. The use of the $\omega/2\pi$ coordinate anticipates work we will be examining shortly. We also recognize that since the vectors are rotating, the scoreboard represents a photograph at a given instant of time. Figure 6 is a representation of A sin (ωt).

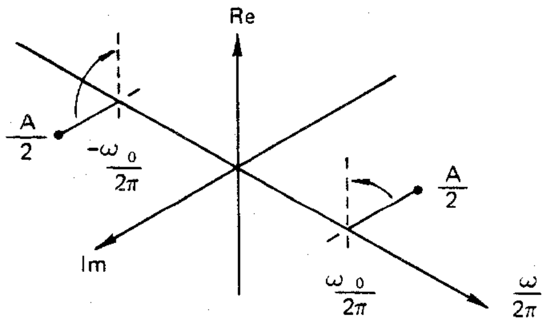


Figure 6. Scoreboard representation of A $\sin(\omega t)$

We note if we allow the sine vectors to rotate a quarter of a turn, and then photograph them, we will have the cosine vectors. Thus, the distinction between the cosine and the sine wave is our definition of the time origin and we see time delay as phase shift applied to a sine wave. See Figure 7.

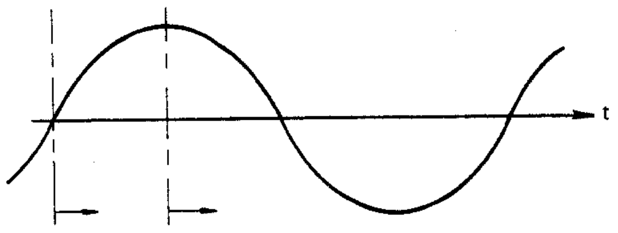


Figure 7. Sine and Cosine Wave Distinguished by Time Origin

Suppose we have a waveshape composed of both a sine and a cosine wave? Then it is of the form

$$A\cos(\omega t) + B\sin(\omega t),$$

which on our scoreboard has the representation shown in Figure 8.

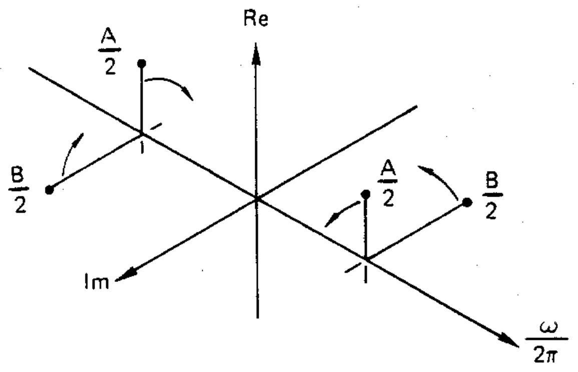


Figure 8. Scoreboard for $A\cos(\omega t) + B\sin(\omega t)$

But we observe that the ordered pair associated with the rate of change of angle ω can be written as

$$\left(\frac{A}{2}, \frac{B}{2}\right) \text{ or as } R\cos(\phi) + jR\sin(\phi) = Re^{j\phi},$$

$$\text{where } R = \sqrt{\left(\frac{A}{2}\right)^2 + \left(\frac{B}{2}\right)^2} \text{ and } \phi = \tan^{-1}\left(-\frac{A}{B}\right),$$

so that an alternate representation for $A\cos(\omega t) + B\sin(\omega t)$ in terms of the complex amplitude of the rotating vector is

$$Re^{j\phi}e^{j\omega t} + Re^{-j\phi}e^{-j\omega t}.$$

See Figure 9.

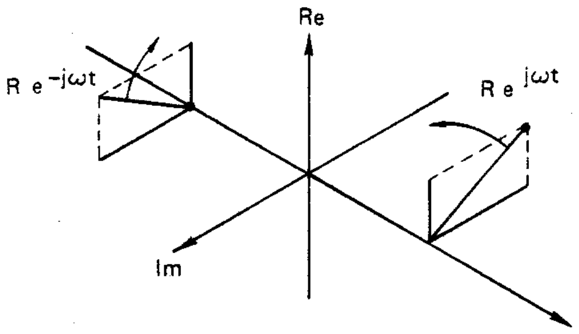


Figure 9. Scoreboard for $Re^{j\phi}e^{j\omega t} + Re^{-j\phi}e^{-j\omega t}$

EVENNESS AND ODDNESS

Another useful tool which we will be using to our advantage is the concept of the even part and the odd part of a function.

For a real function h(t) to be even, h(t) must satisfy h(t) = h(−t). That is, when we rotate the time axis about the origin, we should see the same function. See Figure 10 for examples of an even and a not-even function.

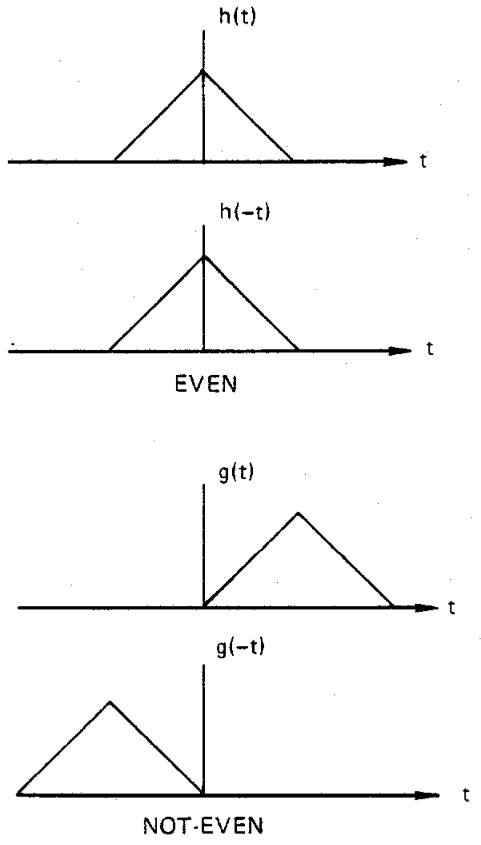


Figure 10. An Even and a Not-Even Function

For a real function h(t) to be odd, h(t) must satisfy $h(t) = -h(-t)$. That is, when we rotate the time axis about the origin we should see the negative of the original function. See Figure 11 for examples of an odd and a not-odd function.

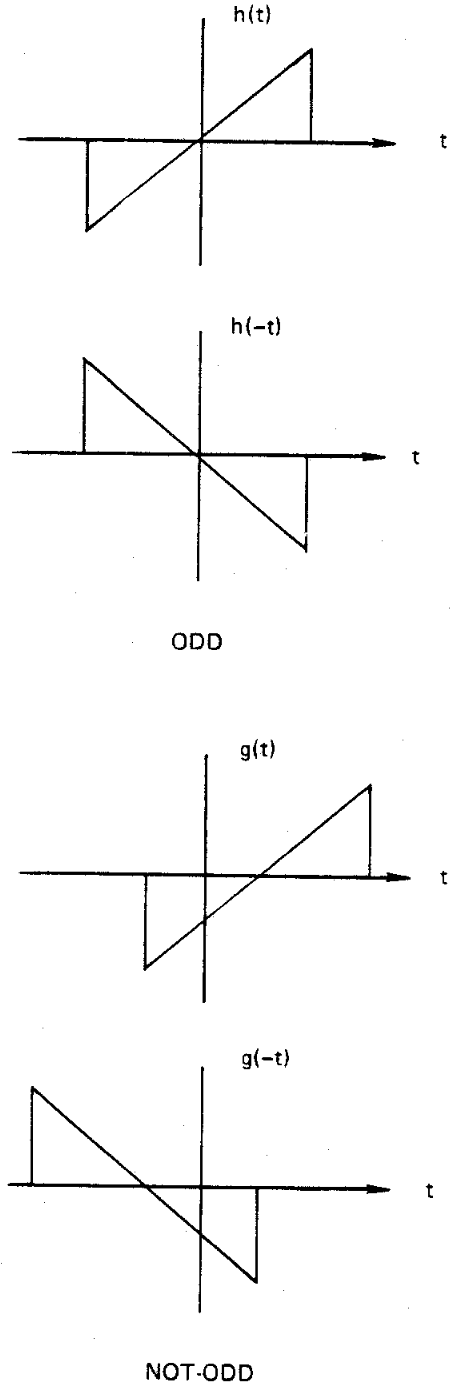


Figure 11. An Odd and a Not-Odd Function

Every function can be decomposed uniquely into its even and its odd parts.

From

$$h(x) = Ev(x) + Od(x),$$

$$h(-x) = Ev(-x) + Od(-x)$$

$$= Ev(x) - Od(x),$$

we derive by addition and by subtraction

$$Ev(x) = \frac{h(x) + h(-x)}{2},$$

$$Od(x) = \frac{h(x) - h(-x)}{2}.$$

See Figure 12 for an example of this decomposition.

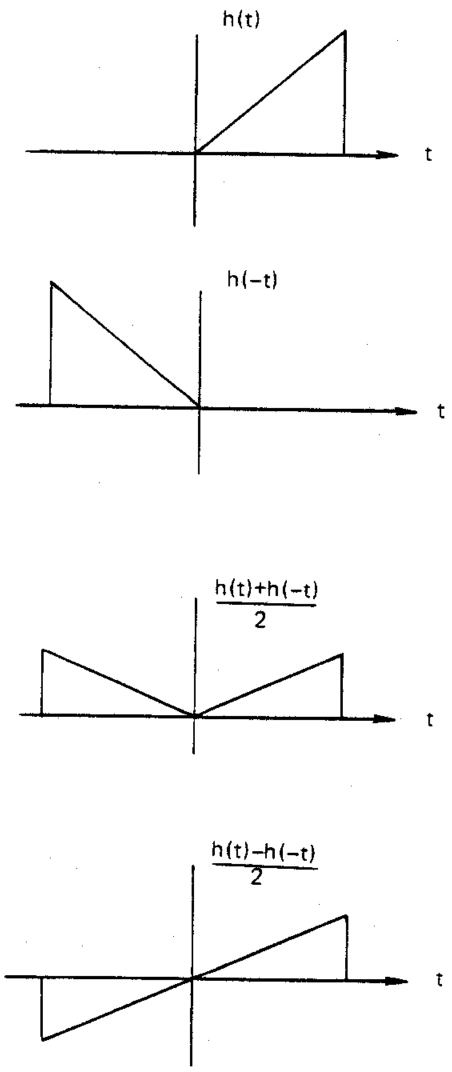


Figure 12. Decomposition to Odd and Even Parts of h(t)

We observe that if $h(t) = e^{j\omega t}$, then

$$Ev(t) = \frac{e^{j\omega t} + e^{-j\omega t}}{2} = \cos(\omega t),$$

$$Od(t) = \frac{e^{j\omega t} - e^{-j\omega t}}{2} = j\sin(\omega t),$$

or that the cosine and sine functions are the even and the odd parts of the complex sinusoid.

FOURIER TRANSFORM

We define the Fourier transform by a pair of integral operators of the form

Fourier transform $$H(\omega) = \int_{-\infty}^{+\infty} h(t)\, e^{-j\omega t}\, dt,$$

Inverse transform $$h(t) = \int_{-\infty}^{\infty} H(\omega)\, e^{+j\omega t}\, d\omega/2\pi.$$

We note that since the inverse transform is an integral with respect to $d\omega/2\pi$ (actually df; oh, that's why the scoreboard has units of $\omega/2\pi$!) another equivalent form of the transform is

$$H(f) = \int_{-\infty}^{+\infty} h(t)\, e^{-j\,2\pi f t}\, dt$$

$$h(t) = \int_{-\infty}^{+\infty} H(f)\, e^{+j\,2\pi f t}\, df.$$

We will use either definition according to our needs and whims. Notice that since the argument of the complex exponential must be dimensionless, the two components of the argument must have reciprocal units such as

t (sec.) – f (sec.$^{-1}$ or Hz)

λ (ft.) – κ (ft.$^{-1}$ or wave number).

In some applications, such as multi-dimensional transforms with input variables having the same units such as x, y, and z, the output variables are denoted ω_x, ω_y, and ω_z.

We note that there are a number of conditions which must be satisfied to assure existence of the transform. These conditions are fairly easy to visualize, and include finite power (square integrable), and finite variation (finite length curve in a finite interval). The introduction of generalized functions, such as impulses, relaxes the first condition. For most applications of engineering interest, a sufficient condition for existence is simply (the *barris* condition) "If you can draw a picture of it, its transform exists!" This is a sufficient but not necessary condition, because there are obviously functions we can not draw, such as those with non-finite support [e.g. $e^{-t}u(t)$], that do have transforms. Bear in mind, of course, that for any machine processing, all conditions for existence are satisfied.

Now let us examine one of the significant properties of the Fourier transform, that property related to even and odd parts of a function. We will use this information to realize considerable computational savings in addition to those realized by use of the fast Fourier transform.

SIGNIFICANCE OF ODDNESS AND EVENNESS

Let h(t) be decomposed into its even and odd parts

$$h(t) = E(t) + O(t),$$

then its Fourier transform is easily seen to be

$$\begin{aligned} H(j\omega) &= \int_{-\infty}^{+\infty} h(t)\, e^{-j\omega t}\, dt \\ &= \int_{-\infty}^{+\infty} [E(t) + O(t)]\,[\cos(\omega t) - j\sin(\omega t)]\, dt \\ &= \int_{-\infty}^{+\infty} [E(t)\cos(\omega t) - jO(t)\sin(\omega t) + \\ &\qquad O(t)\cos(\omega t) - jE(t)\sin(\omega t)]\, dt. \end{aligned}$$

But $O(t)\ \cos(\omega t)$ is an odd function,
$E(t)\ \sin(\omega t)$ is an odd function,

then the integrals

$$\int_{-\infty}^{+\infty} E(t) \sin(\omega t)\ dt = 0$$

$$\int_{-\infty}^{+\infty} O(t) \cos(\omega t)\ dt = 0$$

and $H(j\omega)$ =

$$\int_{-\infty}^{+\infty} E(t) \cos(\omega t)\ dt - j \int_{-\infty}^{+\infty} O(t) \sin(\omega t)\ dt.$$

It follows, then, that the transform of a real, even function is real and that the transform of a real, odd function is imaginary.

But there is more (much more)!

Observe

$$H_E(j\omega) = \int_{-\infty}^{+\infty} E(t)\ \cos(\omega t) dt$$

$$H_E(-j\omega) = \int_{-\infty}^{+\infty} E(t)\ \cos(-\omega t) dt$$

$$= \int_{-\infty}^{+\infty} E(t)\ \cos(\omega t) dt,$$

from which we see that $H_E(j\omega) = H_E(-j\omega)$ or $H_E(j\omega)$ is also even.

Also,

$$H_O(j\omega) = -j \int_{-\infty}^{+\infty} O(t)\ \sin(\omega t)\ dt$$

$$H_O(-j\omega) = -j \int_{-\infty}^{+\infty} O(t)\ \sin(-\omega t)\ dt$$

$$= +j \int_{-\infty}^{+\infty} O(t)\ \sin(\omega t)\ dt,$$

from which we see that $H_O(j\omega) = -H_O(-j\omega)$ or $H_O(j\omega)$ is also odd.

Since the transforms are linear, multiplication by a constant in one domain is equivalent to multiplying by the same constant in the other domain. If the constant is $j(\sqrt{-1})$, then when we move a real function into the imaginary plane, its transforms similarly move! These relationships are depicted quite graphically in Figure 13, after Bracewell (Reference 1, pg. 15).

Given $h(t)$, we can find $H(j\omega)$ by

$$H(j\omega) = \int_{-\infty}^{+\infty} h(t)\ e^{-j\omega t}\ dt.$$

Now, suppose we examine the even part of $h(t)$, $h_e(t)$, then the real part of

$$H(j\omega) = \int_{-\infty}^{+\infty} h_e(t)\ e^{-j\omega t}\ dt$$

$$H_e(j\omega) = \int_{-\infty}^{+\infty} h_e(t)\ \cos(\omega t)\ dt.$$

Again, our instructions: for each value ω, say $\omega = \omega_0$, take the product of $h_e(t)$ with $\cos(\omega_0 t)$, find the area under the product; that area is exactly equal to $H_e(j\omega_0)$.

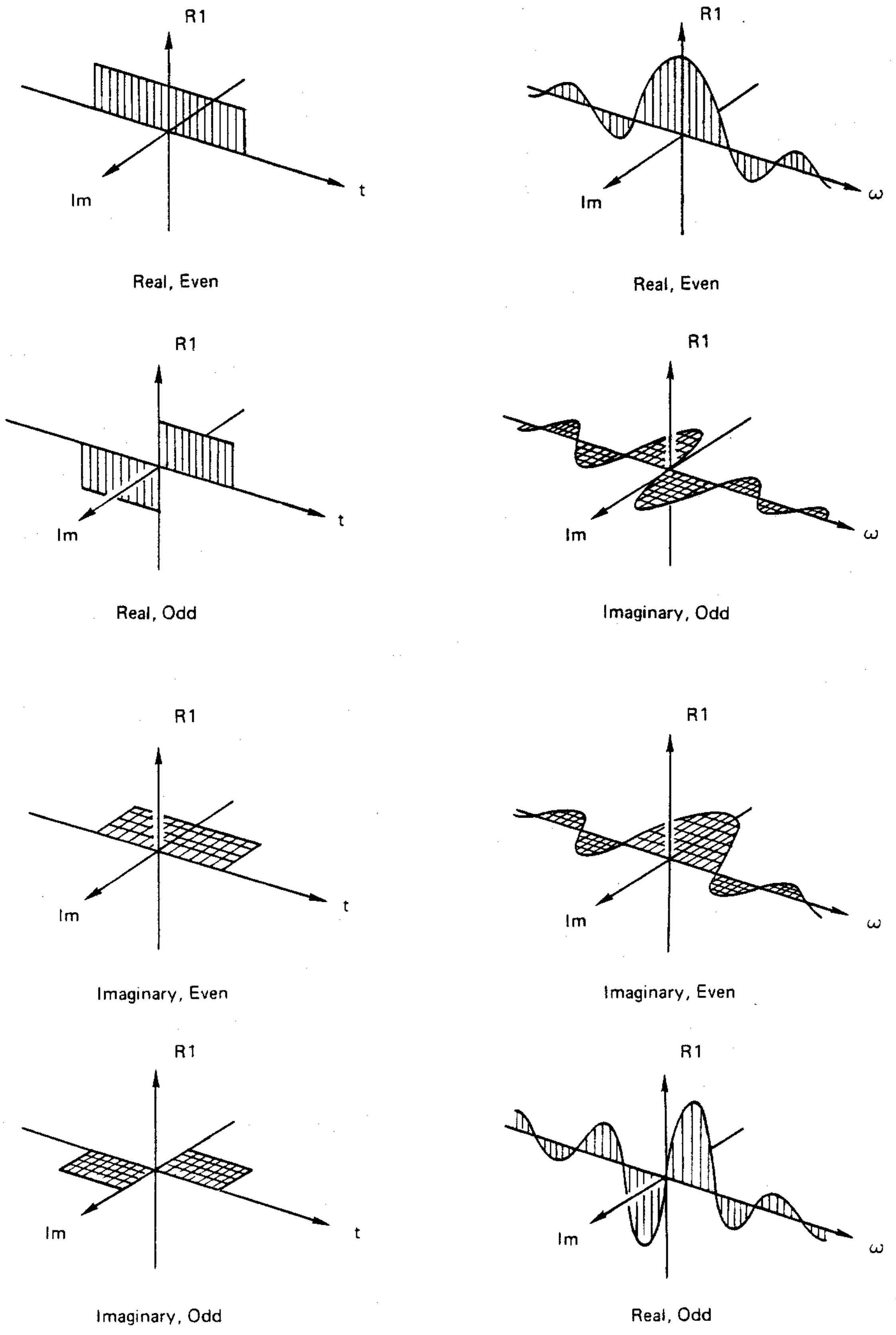
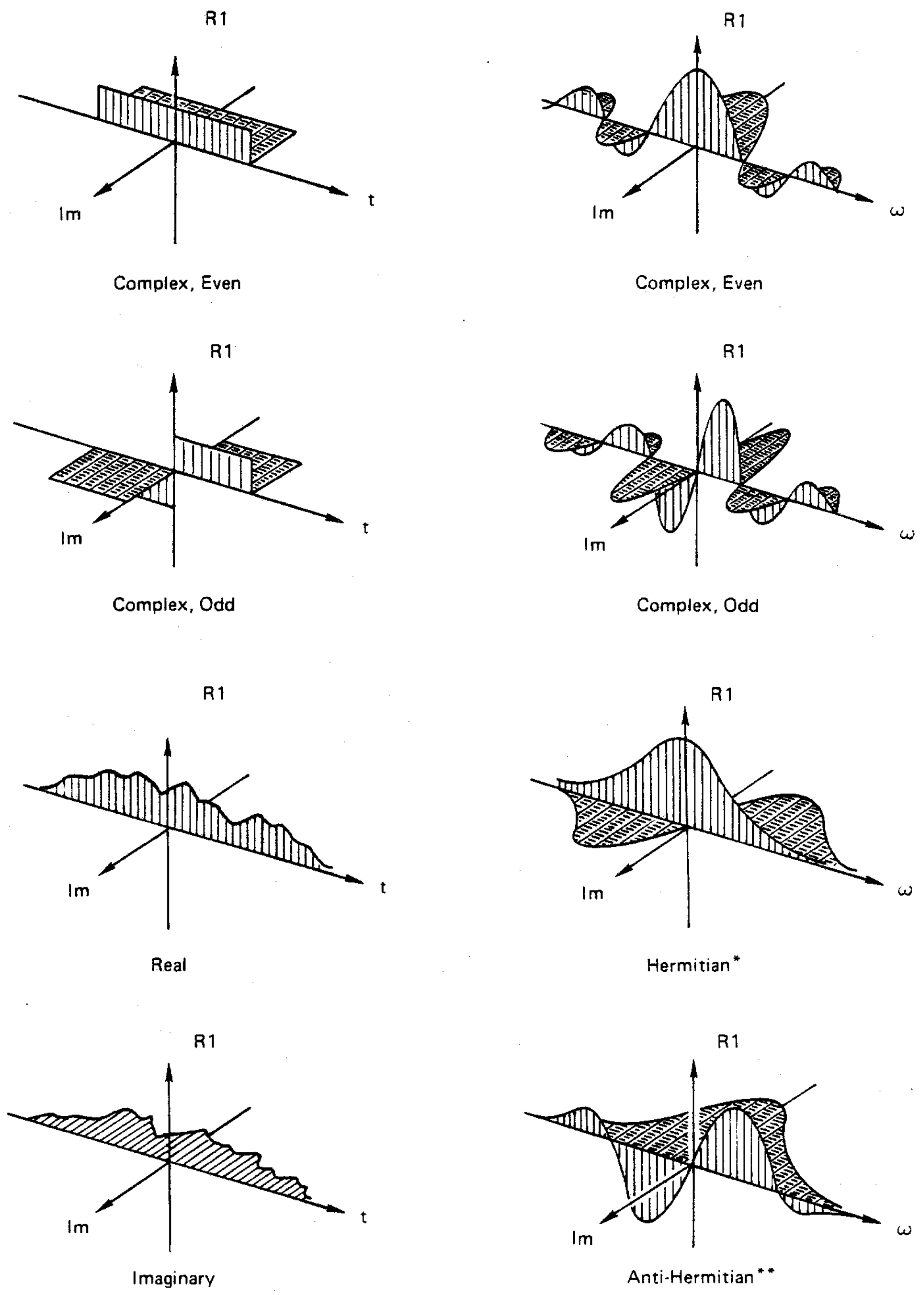


Figure 13a. Even, Odd Relationships of the Fourier Transform



*Hermitian	Real part EVEN
	Imaginary part ODD
**Anti-Hermitian	Real part ODD
	Imaginary part EVEN

Figure 13b. Even, Odd Relationships of the Fourier Transform

For example:

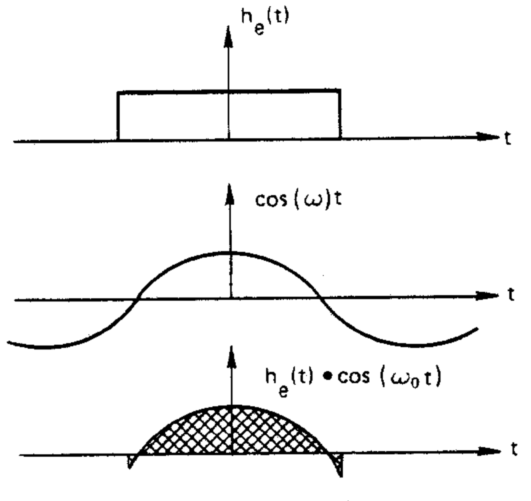


Figure 14a. Product Term, $h_e(t)\cos(\omega_0 t)$

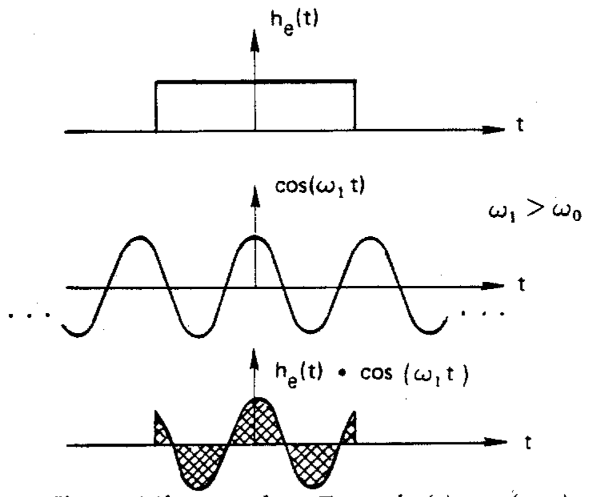


Figure 14b. Product Term $h_e(t)\cos(\omega_1 t)$

We can see that as ω increases, the product $h_e(t) \bullet \cos \omega t$ has vanishingly small area. (This is related to the convergence of the integral and hence to the existence of the transform.)

For real world functions, the area for large ω does go to zero; hence, the Fourier transform goes to zero for large ω.

Again, after Bracewell, we present in Figure 15 a graphical interpretation of an evolving (with ω) transform (Reference 1, page 20).

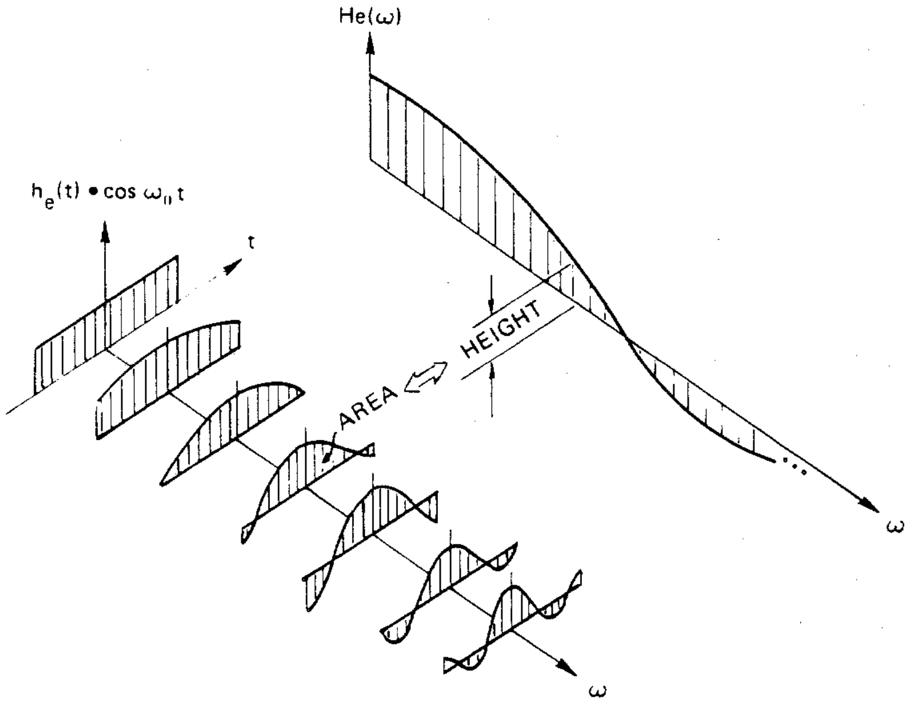


Figure 15. Area Under Product as a Function of ω

A similar sequence can be drawn for the odd part of the function; hence the imaginary part of the transform.

Let us examine the Fourier transform of a very simple function, the rectangle pulse. We use this function as an example because we will need this transform for later work.

Let h(t) be the time function indicated in Figure 16,

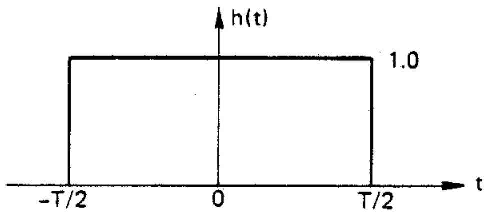


Figure 16. Rectangle Pulse

then the Fourier transform of h(t), H(ω), is seen to be

$$H(\omega) = \int_{-\infty}^{+\infty} h(t)\, e^{-j\omega t}\, dt$$

$$= \int_{-T/2}^{+T/2} h(t)\, e^{-j\omega t}\, dt$$

$$= \int_{-T/2}^{+T/2} 1.0\, e^{-j\omega t}\, dt$$

$$= -\frac{1}{j\omega} e^{-j\omega t} \Big]_{t=-T/2}^{t=T/2}$$

$$= \frac{e^{+j\omega T/2} - e^{-j\omega T/2}}{j\omega}.$$

Multiplying and dividing by T/2, we have

$$H(\omega) = T\, \frac{e^{+j\omega T/2} - e^{-j\omega T/2}}{2j \quad \omega T/2}$$

$$= T\, \frac{\sin(\omega T/2)}{\omega T/2}.$$

We note for later use that this function is called the sinc kernel. The kernel has zeros at each value of ω for which the numerator has a zero, except the values corresponding to zeros of the denominator. The numerator has zeros at those values of ω satisfying

$$\omega \frac{T}{2} = k\ \pi \qquad \text{for all integer } k,$$

$$\text{or } \omega = \frac{2\pi}{T} k$$

$$= \pm\frac{2\pi}{T},\ \pm 2\,\frac{2\pi}{T},\ \pm 3\,\frac{2\pi}{T},\ \ldots\ldots$$

At k = 0, both the numerator and the denominator have zeros which make the evaluation indeterminant. But we can resort to Taylor series or to L'Hospital's rule to determine that $\frac{\sin x}{x}$ evaluated at the origin is unity, so that the kernel evaluated at the origin is T, the average value of the pulse. A typical sinc kernel has the form shown in Figure 17.

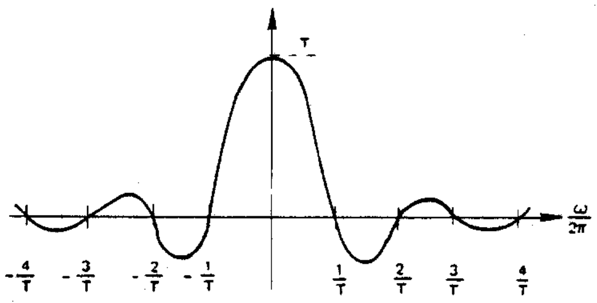


Figure 17. Transform of Rectangle Pulse, Sinc Kernel

FOURIER SERIES

If h(t) is periodic in T, that is, if h(t+T) = h(t) for all t, then at each point of continuity h(t) can be expressed by

$$h(t) = \sum_{-\infty}^{+\infty} c_n\, e^{+j\omega_n t},$$

$$\text{where } \omega_n = \frac{2\pi}{T} n,$$

$$\text{and where } c_n = \frac{1}{T} \int_{-T/2}^{+T/2} h(t)\, e^{-j\,\omega_n t}\, dt.$$

We note that c_n is the complex coefficient of our scoreboard which associates an ordered pair of real numbers with each value of $\omega_n/2\pi$. We also note that the above integral is of the form defining $H(\omega)$ for the Fourier transform. Thus for real $h(t)$, c_n also decomposes into real and imaginary components with the real part exhibiting even symmetry and the imaginary part exhibiting odd symmetry; that is,

$$\mathrm{Re}[c_n] = \mathrm{Re}[c_{-n}]$$

$$\mathrm{Im}[c_n] = -\mathrm{Im}[c_{-n}],$$

which is equivalent to

$$c_n = a_n + j\,b_n$$

$$c_{-n} = a_n - j\,b_n .$$

See Figure 18.

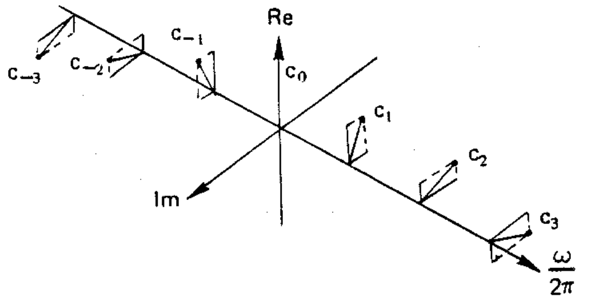


Figure 18. Fourier Series Coefficients on Frequency Axis

By simply reading the real and imaginary components of the complex coefficients, we have an alternate, but equivalent definition of the Fourier series;

$$h(t) = a_0 + \sum_{n=1}^{\infty} 2\,a_n \cos(\omega_n t) + \sum_{n=1}^{\infty} 2\,b_n \sin(\omega_n t),$$

where $a_n = \frac{1}{T}\int_{-T/2}^{+T/2} h(t)\cos(\omega_n t)\,dt$

$$b_n = \frac{1}{T}\int_{-T/2}^{+T/2} h(t)\sin(\omega_n t)\,dt.$$

Returning to the first definition of the Fourier series and applying it to the simple periodic square wave as shown in Figure 19,

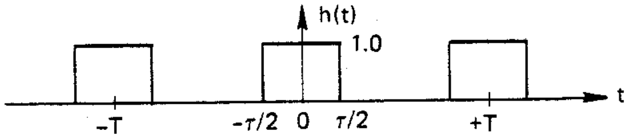


Figure 19. Periodic Rectangle Pulses

$$c_n = \frac{1}{T}\int_{-T/2}^{+T/2} h(t)\,e^{-j\omega_n t}\,dt$$

$$= \frac{1}{T}\int_{-\tau/2}^{+\tau/2} h(t)\,e^{-j\omega_n t}\,dt$$

$$= \frac{1}{T}\,\frac{e^{-j\omega_n t}}{-j\omega_n}\Bigg]_{t=-\tau/2}^{t=\tau/2}$$

$$= \frac{1}{T}\,\frac{e^{+j\omega_n\tau/2} - e^{-j\omega_n\tau/2}}{j\omega_n}.$$

Multiplying and dividing by $\tau/2$, we have

$$c_n = \frac{\tau}{T}\,\frac{e^{+j\omega_n\tau/2} - e^{-j\omega_n\tau/2}}{2j\omega_n\tau}$$

$$= \frac{\tau}{T}\,\frac{\sin(\omega_n\tau/2)}{j\omega_n\tau/2}$$

$$= (\tau/T)\,\frac{\sin[\pi(\tau/T)n]}{[\pi(\tau/T)n]}$$

For $(\tau/T) = 0.1$, the scoreboard representation of the Fourier series coefficients is presented in Figure 20.

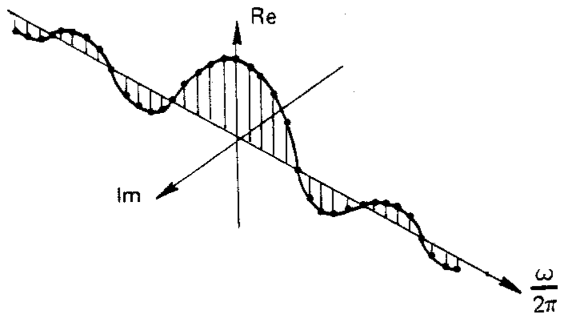


Figure 20. Fourier Series Coefficients for 10% Duty Cycle Square Wave of Figure 19

SAMPLING THEOREM

The two integrals defining entries for our scoreboard were the Fourier transform

$$F(\omega) = \int_{-\infty}^{+\infty} h(t)\, e^{-j\omega t}\, dt,$$

and the Fourier series coefficients

$$c_n = \frac{1}{T} \int_{-T/2}^{+T/2} h(t)\, e^{-j\omega_n t}\, dt.$$

We observed from the two examples that the application of the two integrals lead to functions and to sequences which bear a striking similarity. In fact, we will now concentrate on this similarity and construct the details of the sampling theorem.

Notice we can write the second integral (from above) in the form

$$Tc_n = \left. \int_{-T/2}^{+T/2} h(t)\, e^{-j\omega t}\, dt \right]_{\omega = \omega_n = \frac{2\pi}{T} n}.$$

In this form it is apparent that the Fourier series coefficients of a periodic wave are merely the uniformly spaced samples of the Fourier transform of the nonperiodic version of the same waveshape. We further note that for periodic replicates separated by intervals of T seconds, the corresponding sampling of the Fourier transform occurs at uniform spacing of multiples of 1/T. See Figure 21.

We can play this game from a number of starting places. We can have a nonperiodic function, h(t), and its Fourier transform H(f). We can then sample H(f) and recognize that the sequence of samples corresponds to the Fourier series of the periodically extended version of h(t). See Figure 22.

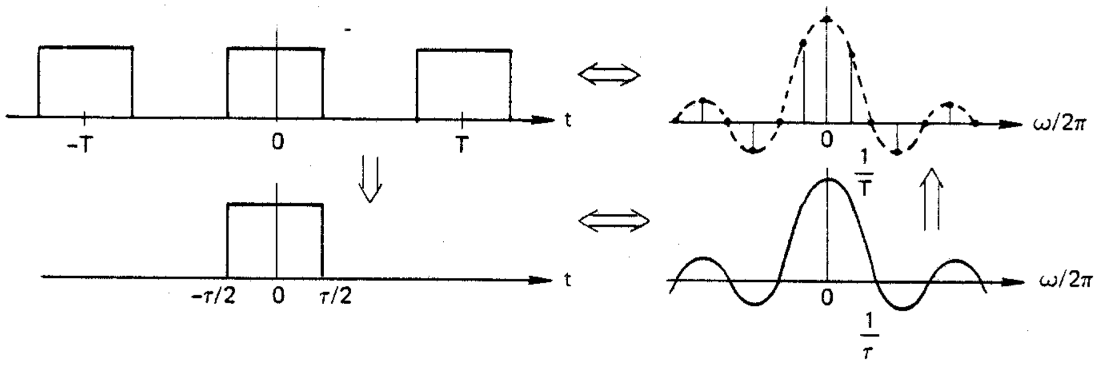


Figure 21. Relation Between Sample Spacing and Replicate Spacing

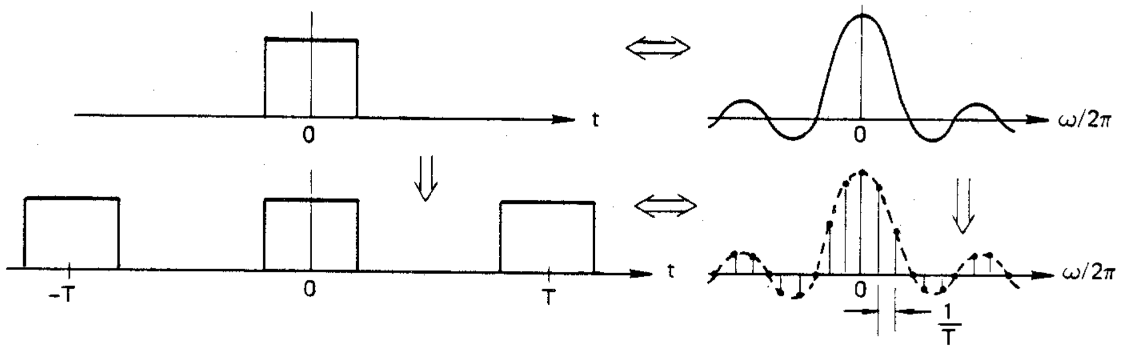


Figure 22. Relation Between Replicate Spacing and Sample Spacing

We observe that the two spacings are reciprocally related; that is, if we increase the spacing in one domain, we decrease it in the other domain.

Thus by taking samples in one domain which are arbitrarily close, the replicates in the other domain are arbitrarily far apart. Then, by increasing the spacing between samples in one domain, the replicates in the other domain move closer together. We may wish to control the sample rate so that the replicates do not overlap. This is accomplished by simply requiring the replicate spacing to exceed the width of the function being replicated; that is equivalent to requiring the sample spacing to be smaller than the reciprocal width of the replicating wave. Note, it may not be possible to avoid overlap because the function being replicated is not of finite width. The overlap between periodic replicates is known as aliasing, or foldback of the replicates.

Finally we note, given a Fourier transform pair, we can sample in either domain, and those samples will be the Fourier series coefficients of the periodic extension in the other domain. Also, the samples may be taken sufficiently close together to avoid unacceptable overlap of the periodic replicates. An example of this operation is shown in Figure 23.

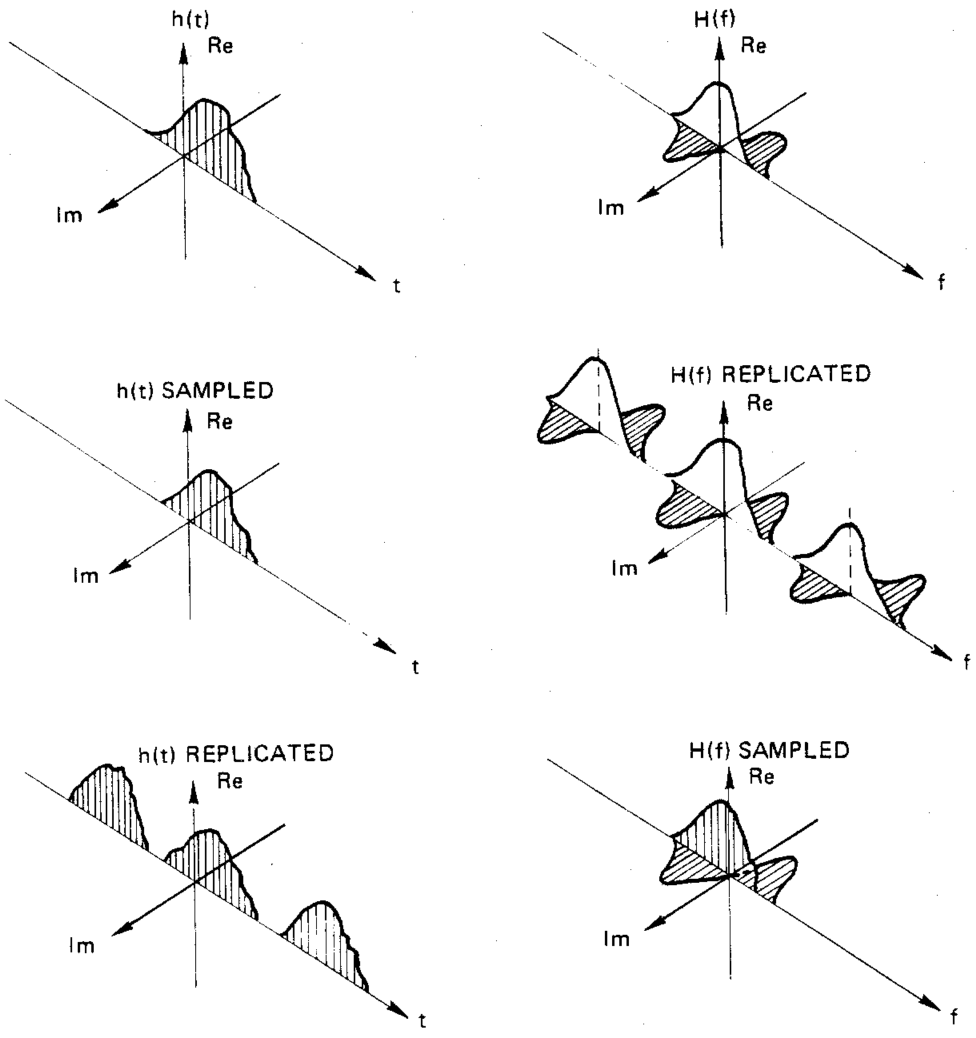


Figure 23. Sampling and Replicating in the Two Domains

We note that for every sequence of numbers (real or complex), there corresponds a periodic waveshape which can be constructed with that sequence being the Fourier series coefficients. Then, given a signal h(t), with transform $H(f)$, we can uniformly sample h(t) to obtain the sequence h(nT) which is the Fourier transform coefficients of the periodically extended version of $H(f)$, denoted $H_T(f)$, where $H_T(f)$ is

$$= \sum_{n=-\infty}^{+\infty} h(nT)\, e^{-j2\pi f(nT)}.$$

FINITE FOURIER TRANSFORM

We observed that the periodic and coordinate normalized Fourier transform associated with a set of time samples, h(nT), or simply h(n), could be written as $H_T(\theta)$, where

$$H_T(\theta) = \sum_{n=-\infty}^{+\infty} h(n)\, e^{-j\theta n} \qquad -\pi < \theta < \pi$$

Of course, for machine computation of the transform we must limit the number of samples h(n) to some manageable finite number. So we now define a finite Fourier transform by

$$H(f) = \int_{-T/2}^{+T/2} \cos(2\pi f_c t)\, e^{-j2\pi ft}\, dt$$

$$= \int_{-T/2}^{+T/2} \frac{e^{+j2\pi f_c t} + e^{-j2\pi f_c t}}{2}\, e^{-j2\pi ft}\, dt$$

$$= \int_{-T/2}^{+T/2} \frac{1}{2} e^{-j2\pi(f-f_c)t}\, dt + \int_{-T/2}^{+T/2} \frac{1}{2} e^{-j2\pi(f+f_c)t}\, dt$$

$$= \frac{1}{2} T \frac{\sin[2\pi(f-f_c)T/2]}{[2\pi(f-f_c)T/2]} + \frac{1}{2} T \frac{\sin[2\pi(f+f_c)T/2]}{[2\pi(f+f_c)T/2]},$$

where the window for this example is the unit rectangle of width T, for which the transform is

$$T \frac{\sin(2\pi fT/2)}{(2\pi fT/2)}.$$

We see that the original unwindowed transform (Fourier series) scoreboard was a set of real numbers located at $\pm f_C$. See Figure 25a. The transform of the window was of the form sin x /x, see Figure 25b, and the windowed transform was of the form two sin x /x, each translated to $\pm f_C$ and

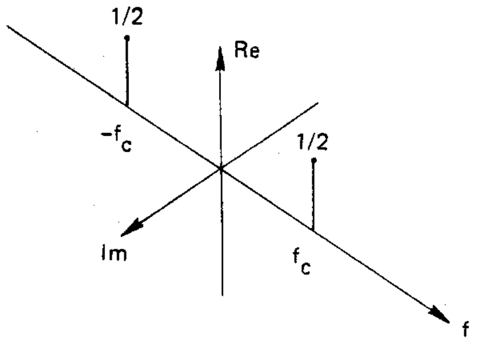


Figure 25a. Unwindowed Transform

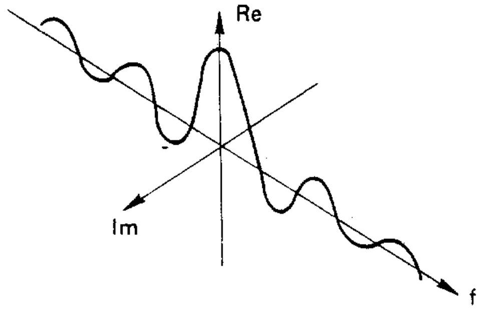


Figure 25b. Window

The Fourier series spectral scoreboard of this signal is presented in Figure 26.

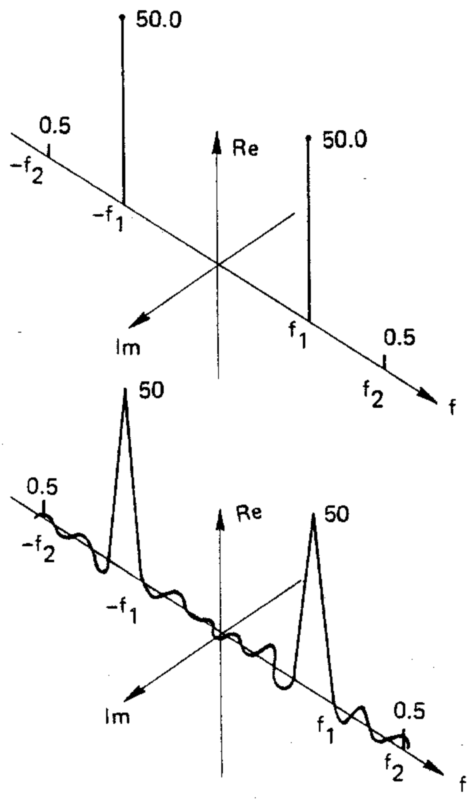


Figure 26. Fourier Spectra of Cosine and Windowed Cosine

When we examine a finite amount of the signal, say T seconds worth, through the rectangle window, the transform becomes the pair of sin x /x located at each spectral line location. Obviously, if f_2 resides in one of the side lobes of the window located at f_1, there is no hope of finding it! The solution, of course, is to select windows whose Fourier transform have low side-lobe levels. But this leads to an interesting conflict. It is easy to demonstrate that the rate at which the envelope of a Fourier transform can fall is related to discontinuities of the original function. If the n-th derivative of a function has a discontinuity, then the transform of that function falls like $\frac{1}{\omega^{(n+1)}}$. One approach to good window selection is to pick windows with discontinuities in the higher derivatives.

The rules of the game are:

1. Windows with smoother behavior in the time domain exhibit smoother behavior in the frequency domain.

2. Windows that are smoother in the time domain tend to have narrower time duration (i.e. smaller rms time width).

3. Windows that are narrower in the time domain tend to have wider bandwidths (i.e. greater rms bandwidth).

4. Convolving with wider spectral width windows will impair the ability to resolve close spectral components.

See Figure 27.

Specific examples and performance of classic windows will be presented after the details of the fast Fourier transform are examined.

DISCRETE FOURIER TRANSFORM

We have gone through a sequence of manipulations on a signal h(t) and its transform H(f). These include, windowing, so that a finite amount of data is processed, and sampling, so that a certain type of machine can perform the processing. These considerations lead us to the finite Fourier transform, which is of the form

$$H(\theta) = \sum_{n=0}^{N-1} h(n)\, e^{-j\theta n},$$

$$\text{or } H(f) = \sum_{n=0}^{N-1} h(nT)\, e^{-j2\pi f Tn}.$$

Notice that H(f) or H(θ) is still a continuous and periodic function. But for machine computation, we must select the values of θ or f for which the computation is to be performed. In fact we have to

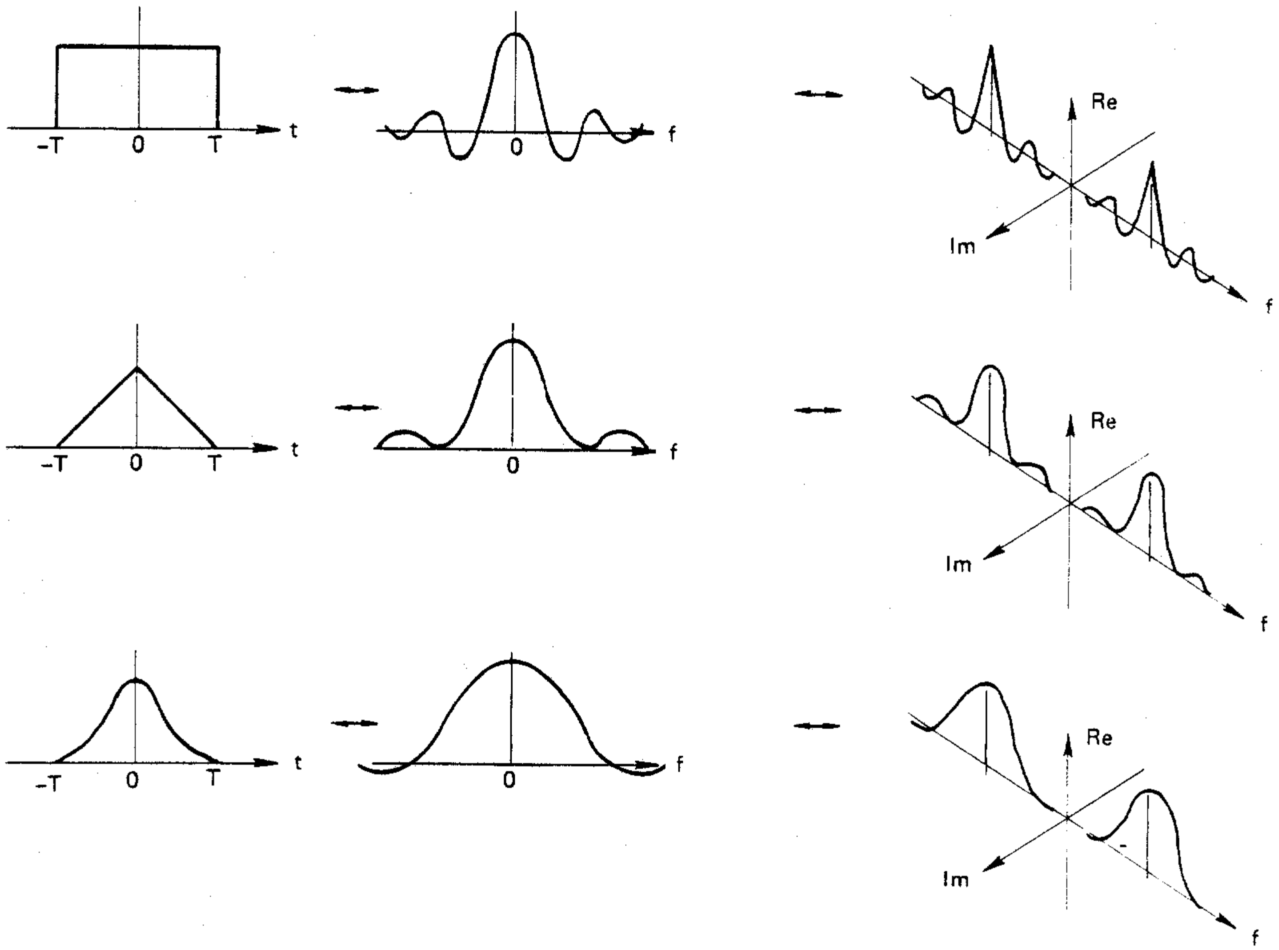


Figure 27. Sequence Showing Smoother Windows, Smoother But Wider Transforms Hence Increased Difficulty in Separating Spectral Lines

sample the continuous and periodic Fourier transform. A reasonable set of θ's or f's at which to sample the Fourier transform are N equally spaced samples over one period of the transform. Thus $\theta = \frac{2\pi}{N}k$, k = 0, 1, 2, . . . N−1, or $f = \frac{f_s}{N}$ k, same k's, so that the sampled finite Fourier transform has the form

$$H(\theta_k) = H\left(\frac{2\pi}{N}k\right)$$

$$= \sum_{n=0}^{N-1} h(nT)\, e^{-j\frac{2\pi}{N}k\ n}.$$

Now, denoting h(nT) by h(n) or h_n to suppress dependence upon time, but not upon the index n, and denoting $H(\theta_k)$ by H(k) or H_k to suppress dependence upon angle, but not upon the index k, we have the form

$$H(k) = \sum_{n=0}^{N-1} h(n)\, e^{-j\frac{2\pi}{N}k\ n}.$$

Before we sampled $H(\theta)$ to obtain $H(\theta_k)$, we had a vehicle to return from the periodic Fourier transform to the time samples; it was the integral transform of the Fourier series. We now address the question, given the N samples of $H(\theta)$, is it still possible to return to the original time samples? And the answer is yes! This is one reason for selecting N samples of the transform. Since we started with N samples of data, the transformation is invertible.

One way to arrive at the inverse transformation is through a simple matrix inverse of the original transformation. We note that for each value of $\theta = \theta_k = \frac{2\pi}{N} k$ we have one equation of the form

$$H_k = \sum_{n=0}^{N-1} h_n e^{-j\frac{2\pi}{N}nk} .$$

For the benefit of the typist, we define $W = e^{-j\frac{2\pi}{N}}$, so that the summation has the form

$$H_k = \sum_{n=0}^{N-1} h_n W^{nk} .$$

The collection of equations can be represented in compact form as a vector matrix equation of the form

$$\begin{bmatrix} H_0 \\ H_1 \\ H_2 \\ \vdots \\ \vdots \\ H_{N-1} \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & \cdots & W^{0(N-1)} \\ W^0 & W^1 & W^2 & \cdots & W^{1(N-1)} \\ W^0 & W^2 & W^4 & \cdots & W^{2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ W^0 & W^{(N-1)} & W^{2(N-1)} & \cdots & W^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ \vdots \\ h_{N-1} \end{bmatrix}$$

where $W = e^{-j\frac{2\pi}{N}}$.

This collection of equations represents a mapping from a point in N space to another point in N space. The mapping is represented compactly as $\overline{H} = \psi \overline{h}$. If we were to invert this mapping, we would be solving for the h_n's in terms of the H_k's. The inverse mapping is represented compactly as $\overline{h} = \psi^{-1} \overline{H}$. The ψ matrix has some very unusual properties; these include orthogonal rows and columns. From the matrix theory, we find the inverse is (within a multiplicative constant) equal to the original matrix, conjugated and transposed. But since the matrix is symmetric, all we need to do is conjugate and divide by a constant. The constant happens to be the determinant of the matrix.

Then,

$$\begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ \vdots \\ h_{N-1} \end{bmatrix} = \frac{1}{N} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W^{-1} & W^{-2} & \cdots & W^{-(N-1)} \\ 1 & W^{-2} & W^{-4} & \cdots & W^{-2(N-1)} \\ \vdots & & & & \vdots \\ \vdots & & & & \vdots \\ 1 & W^{-(N-1)} & W^{-2(N-1)} & \cdots & W^{-(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} H_0 \\ H_1 \\ H_2 \\ \vdots \\ \vdots \\ H_N \end{bmatrix}$$

It is rather simple to demonstrate that this is the correct inverse from which we arrive at the interesting expression

$$h_n = \frac{1}{N} \sum_{k=0}^{N-1} H_k e^{+j\frac{2\pi}{N}kn} .$$

We thus have the discrete Fourier transform pair which we compare now to the continuous Fourier transform pair;

$$\text{Discrete} \quad \begin{cases} H_k = \displaystyle\sum_{n=0}^{N-1} h_n e^{-j\frac{2\pi}{N}kn} \\ h_n = \dfrac{1}{N} \displaystyle\sum_{k=0}^{N-1} H_k e^{+j\frac{2\pi}{N}kn} \end{cases}$$

$$\text{Continuous} \quad \begin{cases} H(\omega) = \displaystyle\int h(t)\, e^{-j\omega t}\, dt \\ h(t) = \dfrac{1}{2\pi} \displaystyle\int H(\omega)\, e^{+j\omega t}\, d\omega \end{cases}$$

INTERESTING SIMILARITY!

We also note that except for the scale factor ($1/N$ or $1/2\pi$), the forward and the inverse transforms are identical, if we consider the conjugation as a trivial operation.

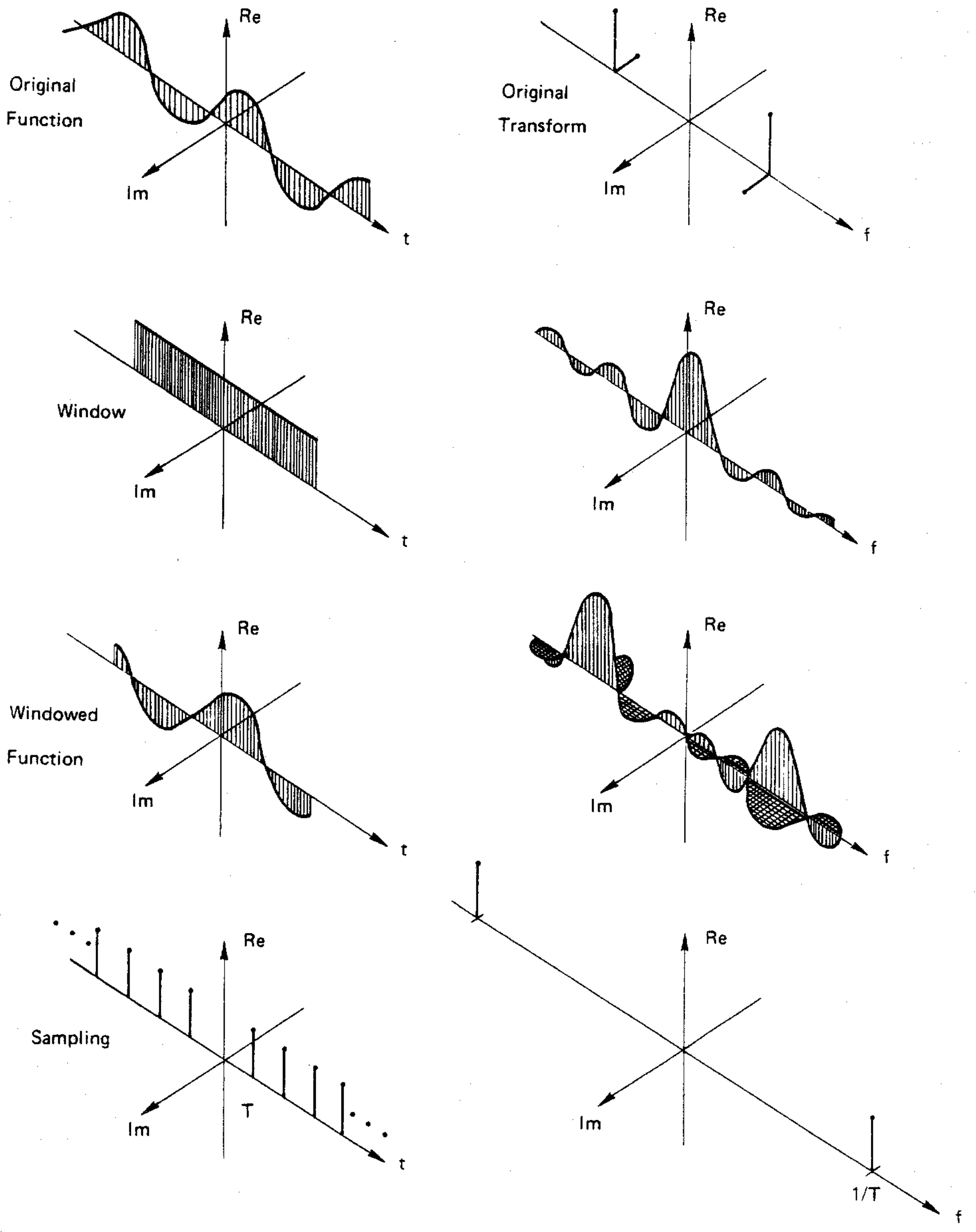


Figure 28a. Sequence of Operations to Obtain DFT

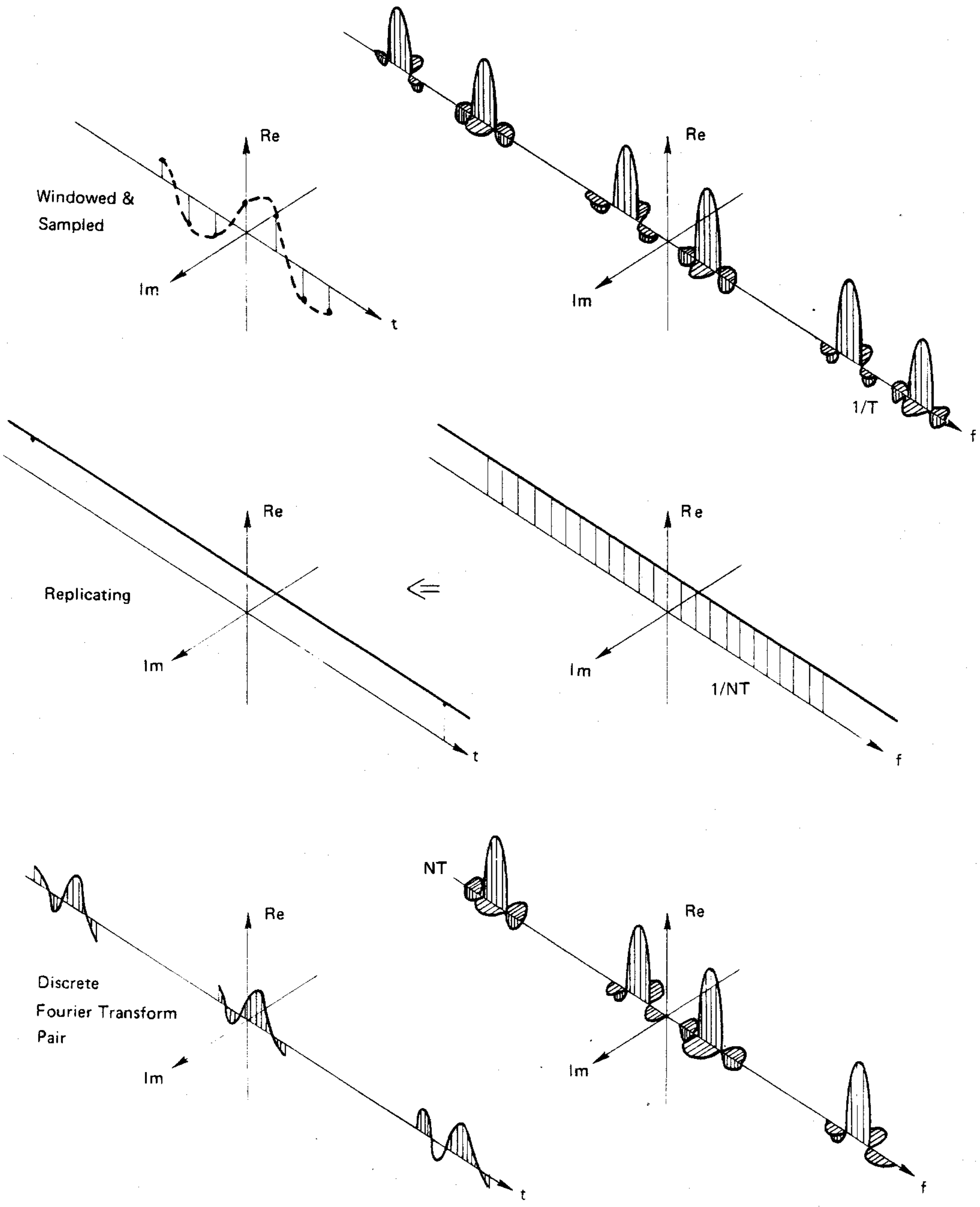


Figure 28b. Sequence of Operations to Obtain DFT

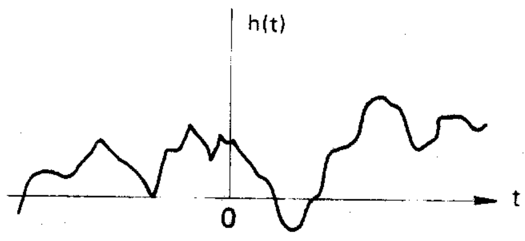
Notice the evolution of our development!

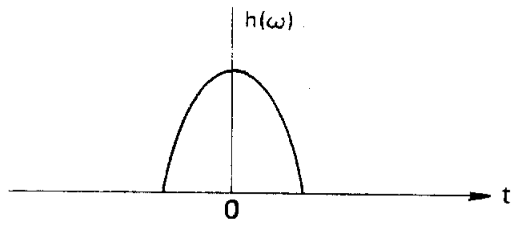
Fourier transform pair:

$$h(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} H(\omega)\, e^{+j\omega t}\, d\omega,$$

$$H(\omega) = \int_{-\infty}^{+\infty} h(t)\, e^{-j\omega t}\, dt,$$

which corresponds to two aperiodic continuous functions.



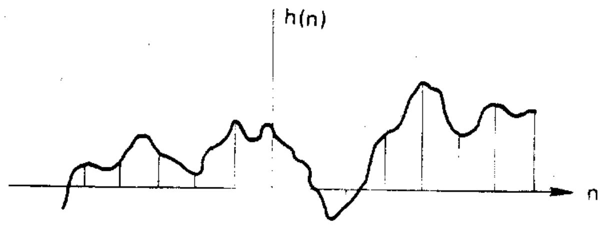


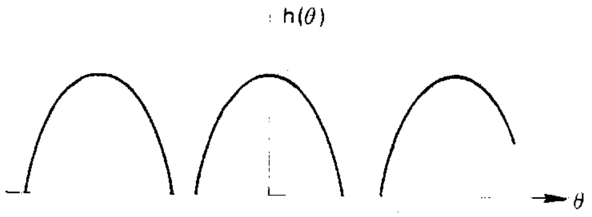
Fourier series pair:

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

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

which corresponds to one aperiodic discrete sequence and to one periodic continuous function.



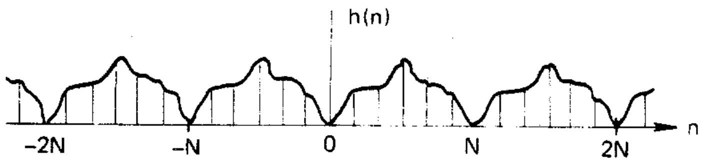


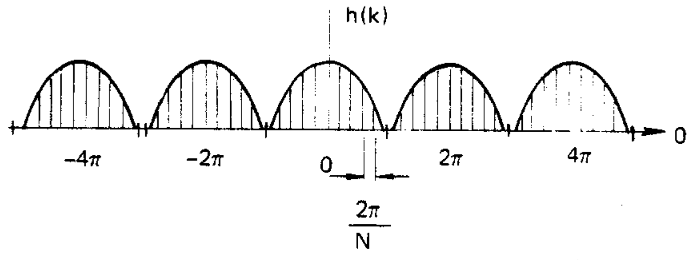
Discrete Fourier transform pair:

$$h(n) = \frac{1}{N} \sum_{k=0}^{N-1} H(k)\, e^{+j\frac{2\pi}{N}nk}$$

$$H(k) = \sum_{n=0}^{N-1} h(n)\, e^{-j\frac{2\pi}{N}nk},$$

which corresponds to two periodic discrete sequences!





We can determine the number of computations required for the discrete Fourier transform by examining the matrix description of the transform.

$$\begin{bmatrix} H_0 \\ H_1 \\ H_2 \\ H_3 \\ \vdots \\ \vdots \\ H_{N-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W^1 & W^2 & & W^{N-1} \\ 1 & W^2 & W^4 & & W^{2(N-1)} \\ 1 & W^3 & W^6 & & W^{3(N-1)} \\ \vdots & & & & \\ \vdots & & & & \\ 1 & W^{(N-1)} & W^{2(N-1)} & & W^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ \vdots \\ \vdots \\ h_{N-1} \end{bmatrix}$$

Each element of the output column vector (H_m) requires N complex multiplications and N complex additions. There are N elements in the output vector, hence we require N^2 complex additions and N^2 complex multiplications.

Recall that a complex multiply is of the form

$$(a,b) \bullet (c,d) = (ac\text{-}bd, ad+bc);$$

hence a complex multiply is actually 4 real multiplications and two real additions.

Similarly, a complex addition is of the form

$$(a,b) + (c,d) = (a+c, b+d);$$

hence a complex add is actually two real additions.

The total computation load is then N^2 complex multiplications and N^2 complex additions, or $4N^2$ real multiplications and $4N^2$ real additions.

Now, let us examine the discrete Fourier transform of the sample set h(n) = 1.0, n = 0, 1, . . . N-1. This is effectively the DFT of the sampled rectangle window.

$$H(\theta) = \sum_{n=0}^{N-1} h(n)\, e^{-j\theta n}.$$

$$= \sum_{n=0}^{N-1} e^{-j\theta n}$$

$$= 1 + e^{-j\theta n} + e^{-j2\theta n} + \ldots + e^{-j(N-1)\theta n}$$

$$= 1 + e^{-j\theta n} + \ldots + e^{-j(N-1)\theta n} + [e^{-jN\theta n} + \ldots] - [e^{-jN\theta n} + \ldots],$$

but $1 + e^{-j\theta n} + e^{-j2\theta n} + \ldots = \dfrac{1}{1 - e^{-j\theta}}$, the closed form of the geometric series.

Also $e^{-jN\theta n} + e^{-j(N+1)\theta n} + \ldots = \dfrac{e^{-jN\theta}}{1 - e^{-j\theta}}$

so that

$$H(\theta) = \frac{1 - e^{-jN\theta}}{1 - e^{-j\theta}}$$

$$= \left(\frac{e^{-j(N/2)\theta}}{e^{-j(1/2)\theta}}\right)\left(\frac{e^{+j(N/2)\theta} - e^{-j(N/2)\theta}}{e^{+j(1/2)\theta} - e^{-j(1/2)\theta}}\right).$$

$$= \left(e^{-j\left[\frac{N-1}{2}\right]\theta}\right)\left(\frac{\sin[(N/2)\theta]}{\sin[(1/2)\theta]}\right)$$

Note $e^{-j\left[\frac{N-1}{2}\right]\theta}$ is the phase-shift term of the transform, and the $\sin[(N/2)\theta]/\sin[(1/2)\theta]$ is the amplitude term. Let us examine the amplitude term. We note here that this term is called the Dirichlet kernel. The kernel has zeros at each value of θ for which the numerator has a zero, except at those values corresponding to zeros of the denominator. The numerator has zeros at those values of θ satisfying

$$\frac{N}{2}\theta = k\,\pi \qquad \text{for all integer } k,$$

$$\text{or} \quad \theta = \frac{2\pi}{N}k, \text{ or at } \omega = \left(\frac{2\pi}{T}\right)\left(\frac{1}{N}k\right),$$

$$\text{where} \quad \theta = \frac{\omega_s}{N}k.$$

The denominator has zeros at the values of θ satisfying

$$\frac{1}{2}\theta = k\pi,$$

$$\text{or} \quad \theta = 2\pi\ k.$$

Thus the Dirichlet kernel has zeros at equally spaced increments $\frac{2\pi}{N}$ apart, except at the origin and multiples of 2π away. If we recall our concept of the periodic function being defined on the unit circle, we see the points 2π apart are the same point. We can resort to Taylor series or to L'Hospital's rule to determine that $\sin(Nx)/\sin x$ evaluated at the origin is N, the number of points in the original sequence. The kernel has the form presented in Figure 29.

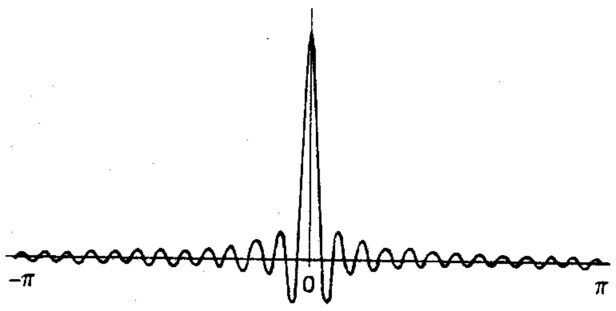


Figure 29. Dirichlet Kernel

We note that the kernel is the periodic replication of the sinc kernel.

Now we must sample $H(\theta)$ at $\theta = \frac{2\pi}{N}k$ to obtain the output of the discrete Fourier transform. But the sample points at $\frac{2\pi}{N}k$ coincide (except at the origin) with the zeros of the kernel. Thus the sample set H(k) is the collection of points shown in Figure 30. Note that linear interpolation between sample points does not properly reflect the function between these sample points.

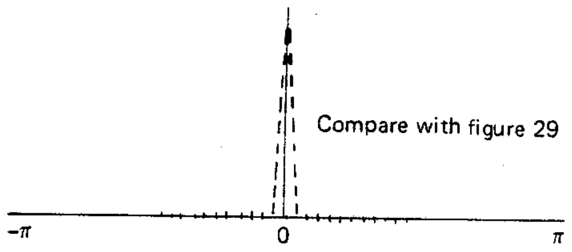


Figure 30. Discrete Fourier Transform Sample Points

Now consider the DFT of samples of the sinusoid defined by

$h(n) = \cos(n\omega_0 T) = \cos(n\theta_0)$ where $\theta_0 = \omega_0 T$.

Thus,

$$H(\theta) = \sum_{n=0}^{N-1} \cos(n\theta_0)\, e^{-jn\theta}$$

$$= \sum_{n=0}^{N-1} \frac{1}{2} e^{jn\theta_0}\, e^{-jn\theta} + \sum_{n=0}^{N-1} \frac{1}{2} e^{-jn\theta_0}\, e^{-jn\theta}$$

$$= \sum_{n=0}^{N-1} \frac{1}{2} e^{-jn(\theta-\theta_0)} + \sum_{n=0}^{N-1} \frac{1}{2} e^{-jn(\theta+\theta_0)}$$

$$= \frac{1}{2} e^{-j\left[\frac{N-1}{2}\right](\theta-\theta_0)} \left(\frac{\sin[\frac{N}{2}(\theta-\theta_0)]}{\sin[\frac{1}{2}(\theta-\theta_0)]}\right) +$$

$$\frac{1}{2} e^{-j\left[\frac{N-1}{2}\right](\theta-\theta_0)} \left(\frac{\sin[\frac{N}{2}(\theta+\theta_0)]}{\sin[\frac{1}{2}(\theta+\theta_0)]}\right),$$

which is a pair of kernels scaled by 1/2 and located at $\pm\theta_0$. See Figure 31.

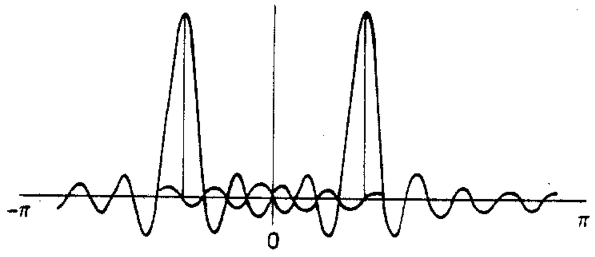


Figure 31. Transform of Windowed and Sampled Cosine

Now note, if θ_0 is any multiple of $\frac{2\pi}{N}$ which we call a DFT bin, then the sample set taken at $k\frac{2\pi}{N}$ coincides exactly with the zeros of the kernels. The sample set will be of the form shown in Figure 32. But if the angle θ_0 is not a multiple of $\frac{2\pi}{N}$, say midway between DFT bins, then the sample set will coincide with the local extrema of the kernel. The magnitude of the samples will have the form shown in Figure 33.

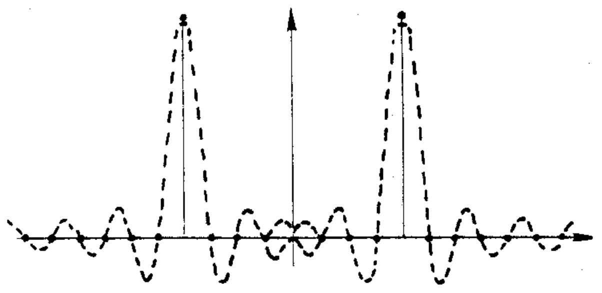


Figure 32. DFT of Sinusoid Harmonically Related to Sample Rate

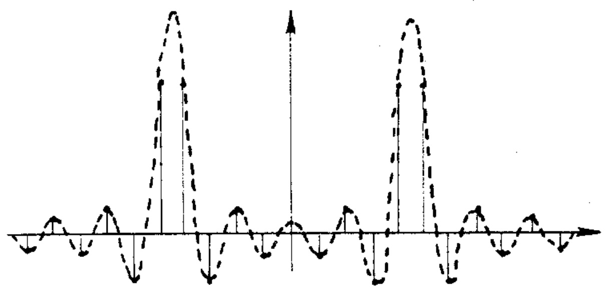


Figure 33. DFT of Sinusoid Not Harmonically Related to Sample Rate

We note that a small shift in the sinusoidal frequency relative to sample rate causes all the zero samples of the kernel to come up apparently out of nowhere. We recognize that the kernel always exhibits the side-lobe structure and for certain frequencies we are fortunate to sample the side lobes at their zero crossing. Often though, we are not so fortunate and the samples of the side lobes impart misleading information to the observer as to the actual spectra. So, what do we do? I'm glad you asked! We apply a different window to the data, one which has significantly lower side-lobe levels.

WINDOWS REVISITED

Windows are an often misunderstood and misapplied stepchild of spectral analysis. Many windows are used in machine processing, not because they are in some way superior, but rather because they are easy to visualize; or are simply generated; or someone thinks that they perform well; or "we've always done it this way". We will first examine classic windows and then demonstrate their performance in a spectral resolution application.

A word of caution, windows are even functions about their point of symmetry. If the point of symmetry is the origin, the evenness requires an odd number of points. The transform sequences, however, will have an even number of points. How do we resolve the discrepancy? Easy, we have to remember that the windows are periodically extended under the DFT sampling. Thus the two end points are the same point! The triangle wave shown below demonstrates that one end point belongs to the beginning of the waveshape and that the other end point is the beginning of the next replicate (Figure 34).

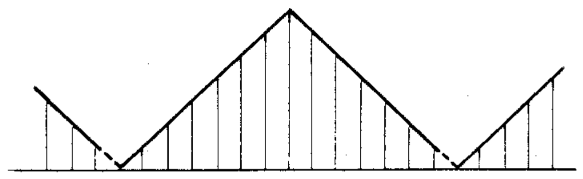


Figure 34. Periodic Extension of Window

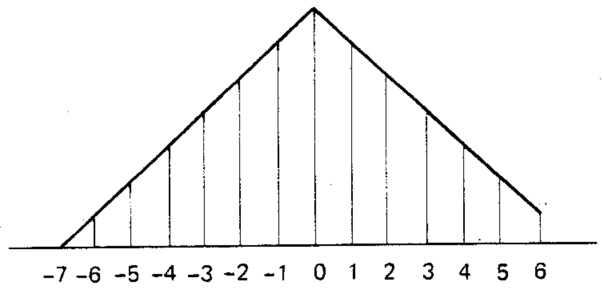


Figure 35. An Even Sequence, on the Circle +7 = –7

Thus the sequence presented in Figure 35 is an even sequence under the DFT. Since the sequence is periodic, it can be shifted so that the end point coincides with the origin. The shift, of course, only contributes linear phase shift and does not affect the magnitude response of the window transform. All of the descriptions of the windows will assume even sequences about the origin. When used as a window, the sequence may be shifted and one end point deleted. All of the windows now discussed will have N+1 points with N being even.

Rectangle Window: $W(n) = 1.0$ See Figure 36.

$$n = -N/2, \ldots, -1, 0, 1, \ldots, N/2$$

Triangle Window: (Fejer or Parzen) See Figure 37.

$$W(n) = 1.0 - \left|\frac{2n}{N}\right| \quad n = -N/2, \ldots, -1, 0, 1, \ldots, N/2$$

$\cos^2$ Window: (Hanning) See Figure 38.

$$W(n) = 0.5 + 0.5 \cos(n \frac{2\pi}{N})$$

$$n = -N/2, \ldots, -1, 0, 1, \ldots, N/2$$

Raised cos Window: (Hamming) See Figure 39.

$$W(n) = 0.54 + 0.46 \cos(n \frac{2\pi}{N})$$

$$n = -N/2, \ldots, -1, 0, 1, \ldots, N/2$$

Gaussian Window: See Figures 40a,b,c.

This window has an adjustable parameter, effectively the reciprocal standard deviation β.

$$W(n) = e^{-[(\frac{2n}{N})^2 \frac{\beta^2}{2}]}$$

$$n = -N/2, \ldots, -1, 0, 1, \ldots, N/2$$

Dolph-Tchebyschev Window: See Figures 41a,b,c.

This window has an adjustable parameter, the side-lobe level, where β is the number of decades that the side-lobe level is down relative to the main lobe. The window is most easily defined in terms of its frequency response through the Tchebyschev polynomials. The window values are obtained as an inverse FFT.

$$W(n) = \mathrm{FFT}^{-1}\left\{\frac{\cos\{N \cos^{-1}[\alpha \cos(2\pi \frac{n}{N})]\}}{\cosh[N \cosh^{-1}(\alpha)]}\right\},$$

$$\text{where } \alpha = \cosh[\frac{1}{N} \ln(10^\beta + \sqrt{10^{2\beta} - 1})],$$

$$\text{and } \cos^{-1}(x) = \pi/2 - \tan^{-1}[x/\sqrt{1 - x^2}] \quad |x| < 1.0$$

$$= \ln(x + \sqrt{x^2 - 1}) \quad |x| > 1.0.$$

Kaiser-Bessel Window: See Figures 42a,b,c.

This window has an adjustable parameter, the time-bandwidth product β.

$$W(n) = \frac{I_0[\beta \sqrt{1 - (2n/N)^2}]}{I_0(\beta)}$$

$$n = -N/2, \ldots, -1, 0, 1, \ldots, N/2,$$

$$\text{where } I_0(x) = 1 + \sum_{k=1}^{\infty} \left[\frac{[\frac{x}{2}]^k}{k!}\right]^2$$

We note in all the windows the trade-off between side-lobe level and main-lobe width. When we push down on the bumps, the main lobe widens. A good figure of merit for the width can be derived according to the following reasoning.

The main-lobe width of any window cannot be less than $2\pi/N$, the width of the Dirichlet kernel or rectangle window. Examining the various windows we see that each 20 dB drop in side-lobe level will cost approximately a single DFT bin width. The superior windows will exhibit main-lobe widths of between 3 and 4 DFT bins for side-lobe suppression exceeding 60 dB.

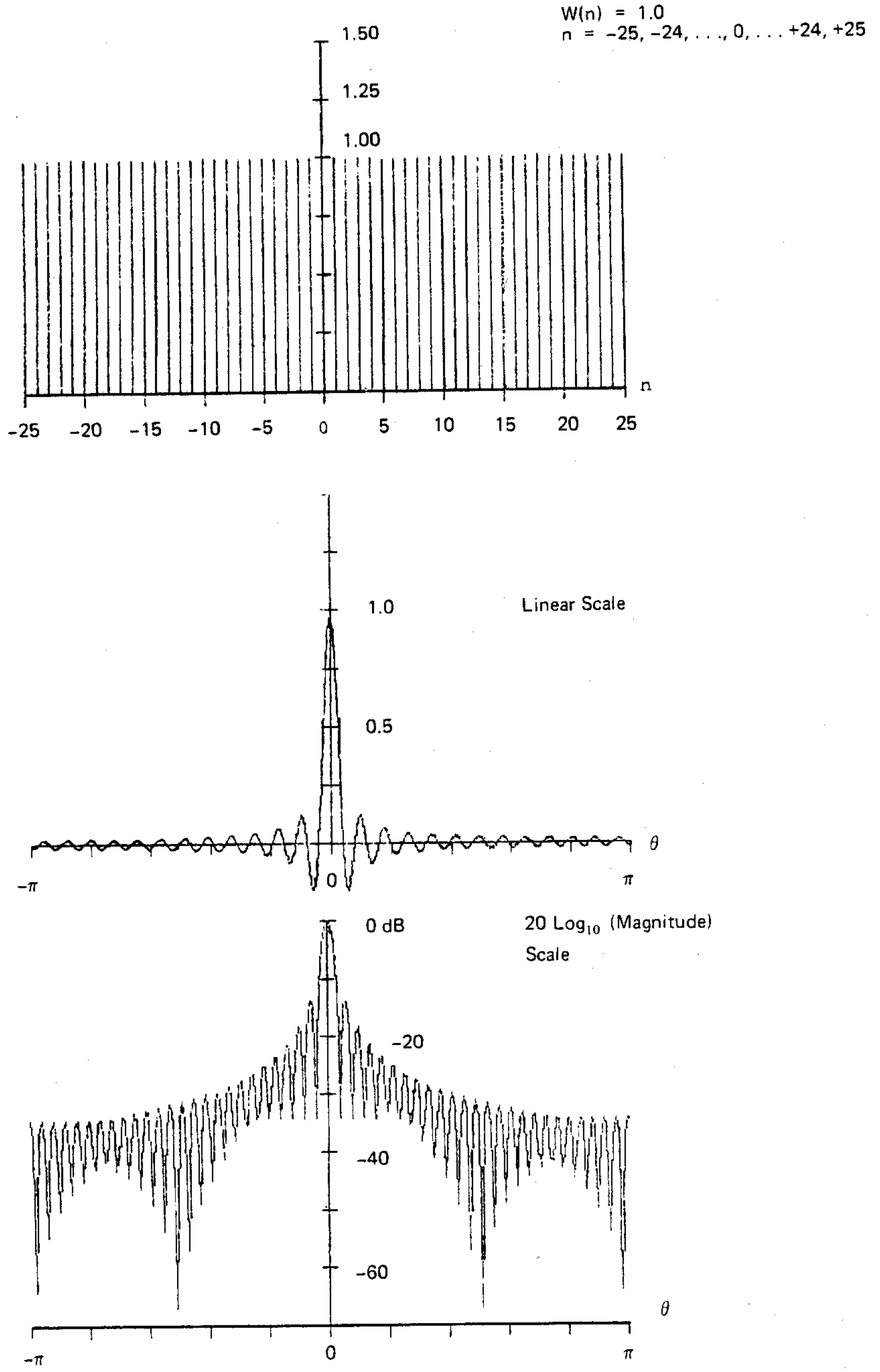


Figure 36. Rectangle Window and Transform

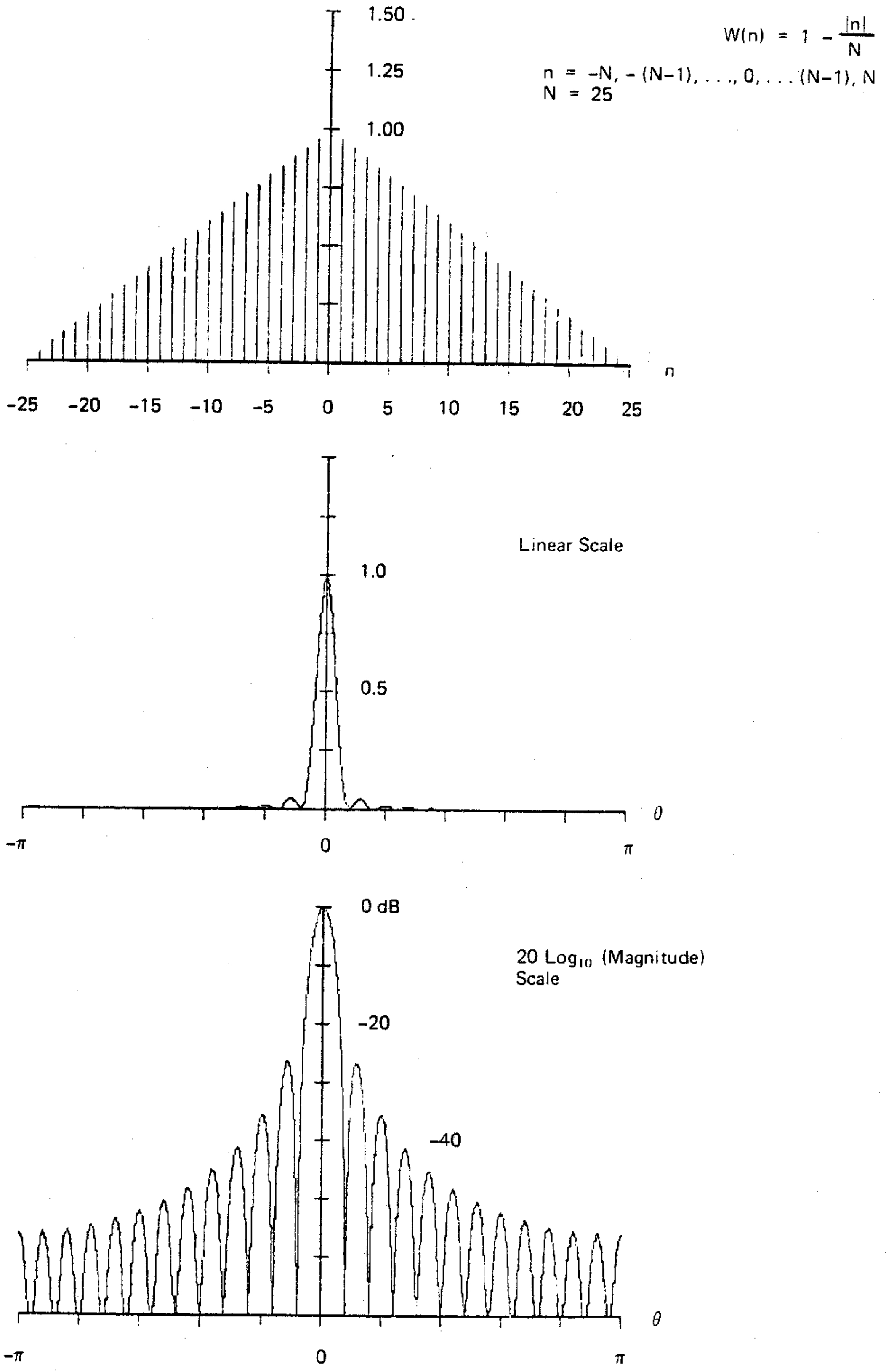


Figure 37. Triangle Window and Transform

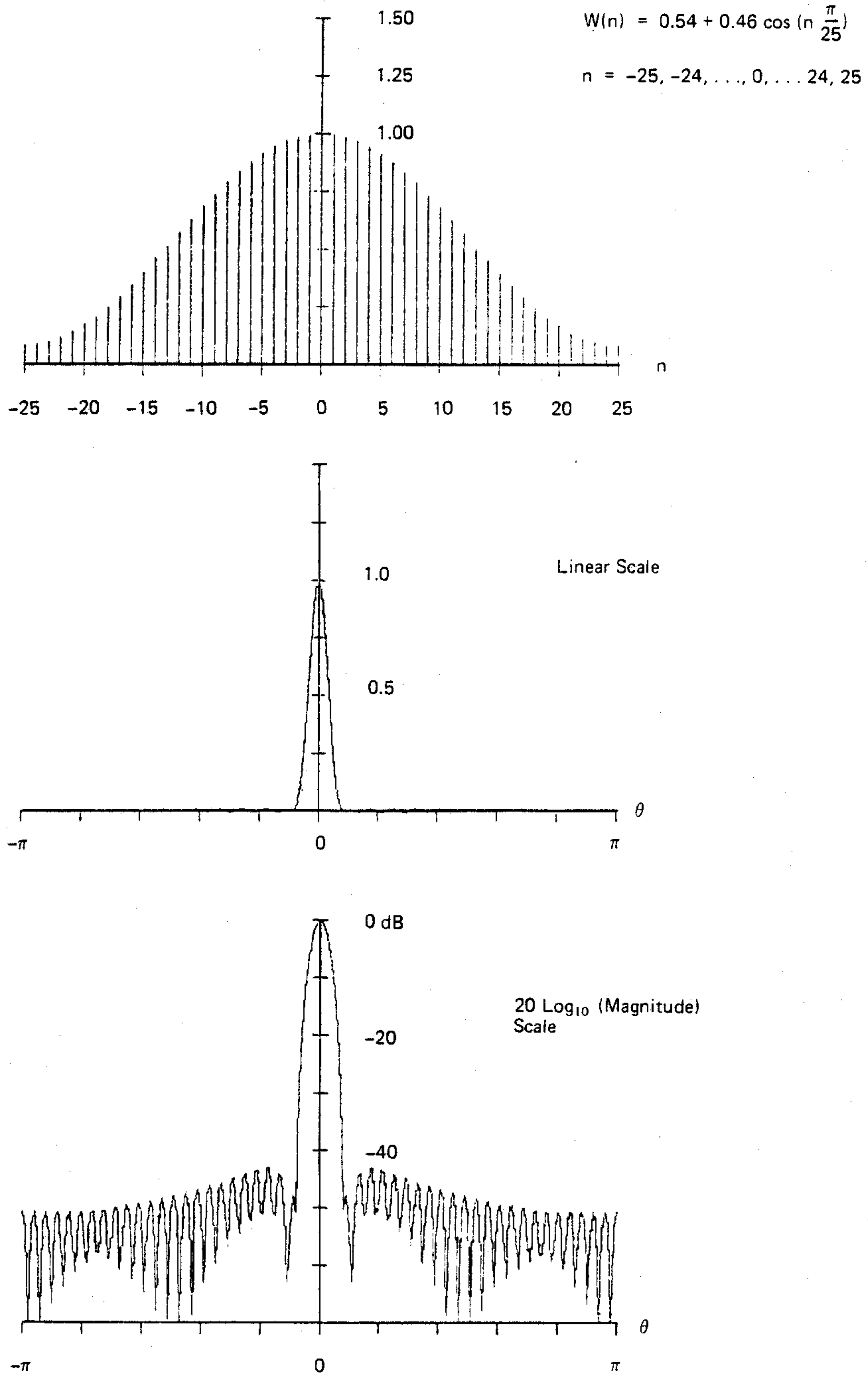


Figure 39. Raised Hamming Window and Transform

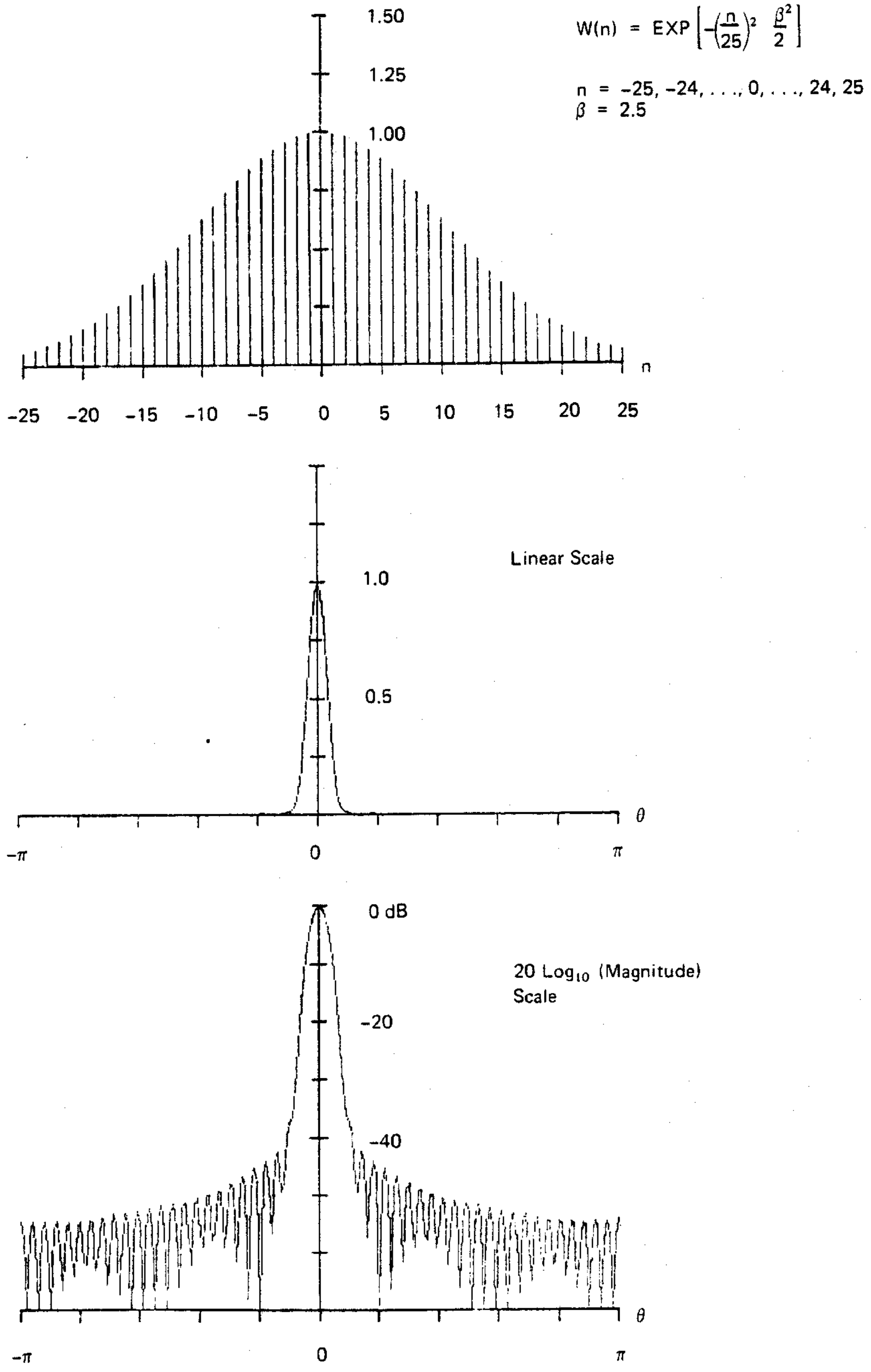


Figure 40a. Gaussian Window ($\beta = 2.5$) and Transform

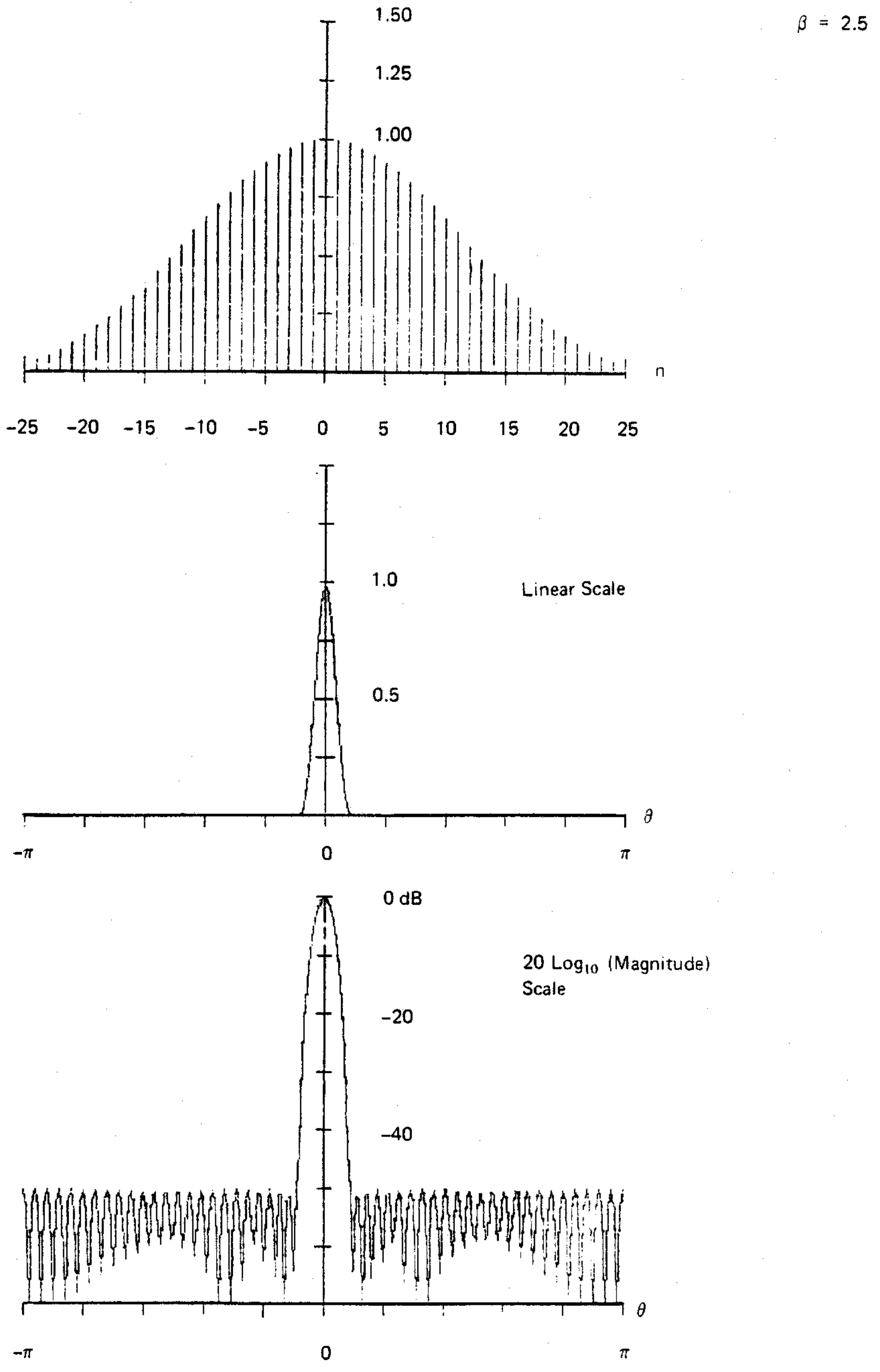


Figure 41a. Dolph-Tchebyshev Window (β = 2.5) and Transform

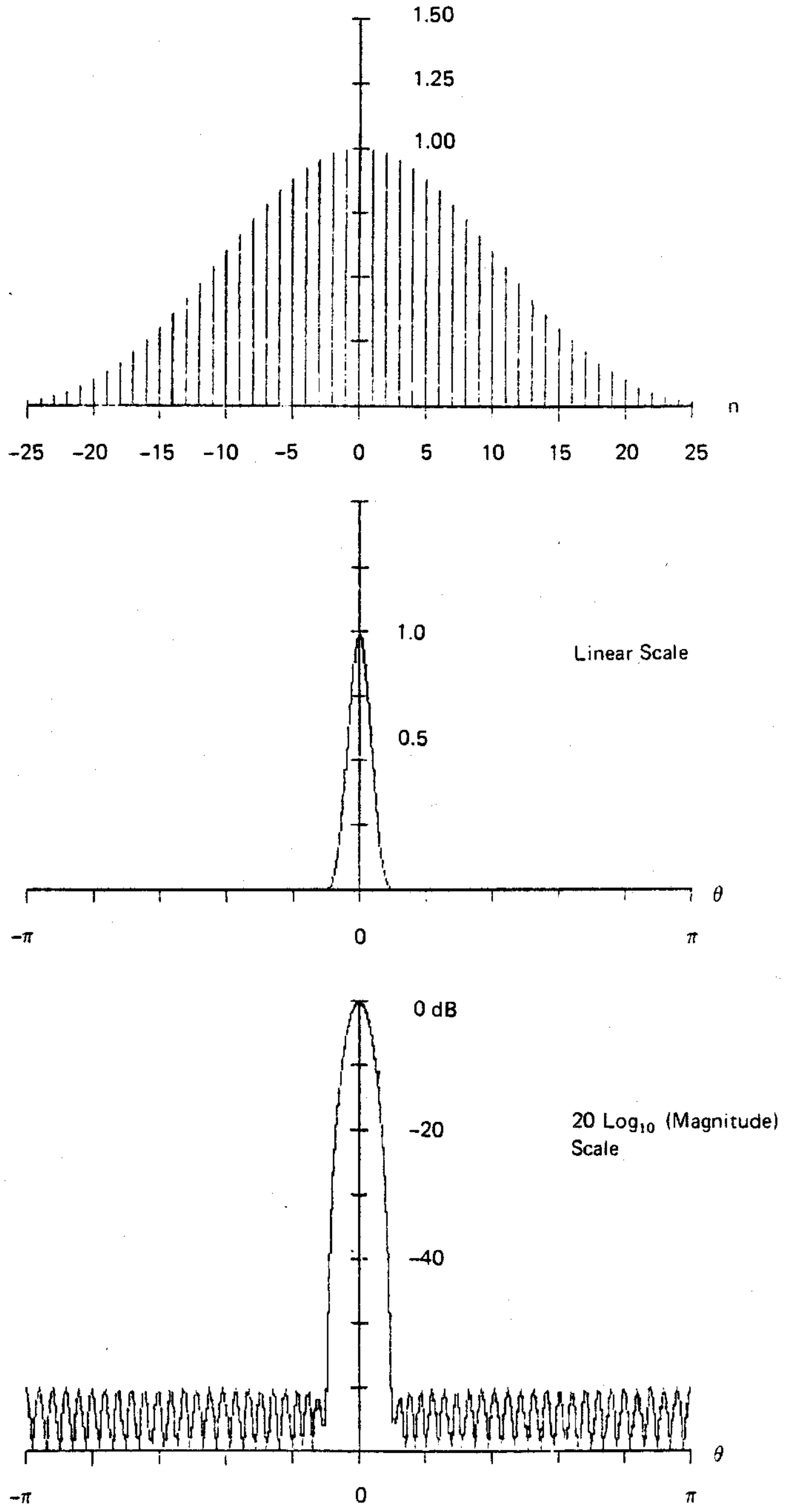


Figure 41c. Dolph-Tchebyshev Window (β = 3.0) and Transform

SPECTRAL RESOLUTION

We now describe a simple experiment. We sample two sinusoids of frequencies $10\frac{f_s}{N}$ and $16\frac{f_s}{N}$ and of amplitudes 1.0 and 0.01 respectively, where f_s is the sample frequency. The DFT of the signal is presented in Figure 43 and reflects our description of the signal. We now shift the frequency of the larger signal to $10.5\frac{f_s}{N}$ which is midway between two DFT bins. The DFT of this signal is presented in Figure 44 from which we observe the total loss of the smaller signal in the side lobes of the larger signal.

We now apply different windows so we can observe the effects on detectability and resolution. The first window is the triangle window, and as expected, the side lobes have fallen considerably, but not enough. The second signal is barely detectable (Figure 45).

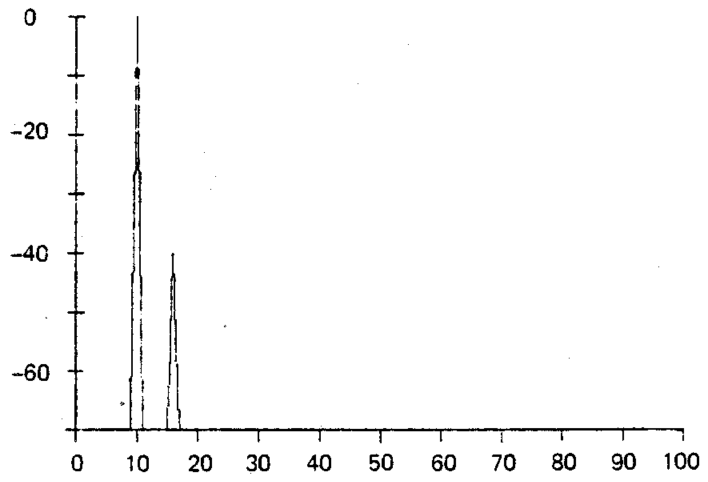
The Hanning window performs slightly better. The second signal is positively detected with about a 3 dB notch separating the two main lobes (Figure 46).

The Hamming window realizes a deeper notch between the lobes of nearly 20 dB, but now the side-lobe structure obscures the rest of the spectra (Figure 47).

The Dolph-Tchebyschev window easily separates the two signals. The side-lobe structure is disappointing. The reasoning of selecting a window with the narrowest main lobe for a given side-lobe level seemed to make great sense. It makes sense, but it doesn't perform well (Figures 48a, b, c).

The Gaussian window does fairly well. The two signals are easily separated for reciprocal standard deviations between 3.0 and 3.5 and the side-lobe structure is manageable. Here we can see the diminishing returns of depressing the side lobes at the expense of main-lobe width. During the change from 3.0 to 3.5 to further control side lobes, the main-lobe width has increased sufficiently to fill the notch between the two signals (Figures 49a, b, c).

The Kaiser-Bessel window does an outstanding job in separating the two signals and holding down the side-lobe levels. For time-bandwidth products exceeding 7.0, the side lobes are more than 60 dB down and the notch separating the two signals is 20 dB. The Kaiser-Bessel window achieves this superior performance because it is the window which, for a given time duration, maximizes the energy concentrated in the main lobe.



	FFT Bin	Ampl.
Signal 1.	10.0	1.0
Signal 2.	16.0	0.01

Figure 43. Rectangle Window

FAST FOURIER TRANSFORM

The discrete Fourier transform is a well defined trigonometric summation relating a pair of periodic sequences each of length N. The fast Fourier transform on the other hand is merely a collection of algorithms (or any one of the collection) which realizes the required computations with a significant reduction in the actual number of multiplications performed. The algorithms effectively factor the weighted summations into a sequence of shorter weighted summations. This is done by taking advantage of the periodicities and the symmetries of the periodic weighting factors and of the transform itself. We will first describe the factoring as an iterative process and we will then interpret the resultant factoring as a matrix factoring, which we can describe quite nicely with signal flow graphs.

The discrete Fourier transform is a collection of equations defined by

$$F(k) = \sum_{n=0}^{N-1} f(n)\, e^{-j\frac{2\pi}{N}nk} \quad \text{for } k = 0, 1, \ldots, N-1.$$

For the sake of the typist we define $W = e^{-j\frac{2\pi}{N}}$, and rewrite the definition of the summation by

$$F(k) = \sum_{n=0}^{N-1} f(n)\, W^{nk} \quad \text{for } k = 0, 1, \ldots, N-1.$$

It is convenient to recognize that W is a point on the unit circle, and that W^m is also a point on the circle with an angle "m" times the size of the angle associated with W; that is, $m\frac{2\pi}{N}$. See Figure 51.

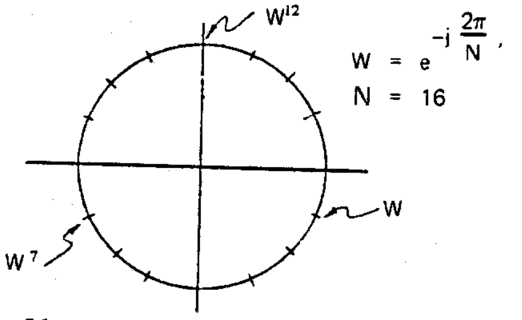


Figure 51. The Points W^m Identified on the Unit Circle (For N = 16)

We note that W is periodic in N terms, so that there are only N distinct points W^m. We can actually do better by recognizing that half of the points are negative of some other point on the circle; for example, $W^{10} = -W^2$.

Returning to the expression for F(k), we now proceed to factor the summation into two summations, each equal to half the original length.

$$F(k) = \sum_{n=0}^{N-1} f(n) W^{nk}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} \left| f(2n) W^{(2n)k} + f(2n+1) W^{(2n+1)k} \right|$$

$$= \sum_{n=0}^{\frac{N}{2}-1} f(2n) W^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} f(2n+1) W^{2nk}\, W^k.$$

But k is selected outside the summation, so we can factor the W^k term from the second summation.

$$\text{Thus } F(k) = \sum_{n=0}^{\frac{N}{2}-1} f(2n) W^{2nk} +$$

$$W^k \sum_{n=0}^{\frac{N}{2}-1} f(2n+1) W^{2nk},$$

$$\text{or } F(k) = A(k) + W^k B(k),$$

where A(k) is recognized as an N/2 point transform over the even indexed data points, and B(k) is an N/2 point transform over the odd indexed data points. Notice, stepping around the circle to pick up the complex coefficients, now steps across every other angle; that is,

$$W^{2m} = e^{-j\frac{2\pi}{N}2m} = e^{-j\frac{2\pi}{N/2}m}$$

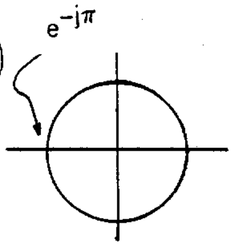
which is equivalent to saying that we need only N/2 angles for an N/2 point transform.

Let us also examine $W(k+\frac{N}{2})$, the transform point halfway through the list of output points. Substituting $k+\frac{N}{2}$ for k in $F(k) = A(k) + W^k B(k)$,

$$F(k+\tfrac{N}{2}) = A(k+\tfrac{N}{2}) + W^{(k+\frac{N}{2})} B(k+\tfrac{N}{2}).$$

But A(k) and B(k) are N/2 point transforms and periodic in N/2 points; thus, $A(k+\frac{N}{2}) = A(k)$ and $B(k+\frac{N}{2}) = B(k)$. Also,

$$W^{(k+\frac{N}{2})} = W^k W^{(\frac{N}{2})} = W^k e^{-j\frac{2\pi}{N}\left(\frac{N}{2}\right)} = W^k e^{-j\pi} = -W^k.$$



Thus $F(k+\frac{N}{2}) = A(k) - W^k B(k)$.

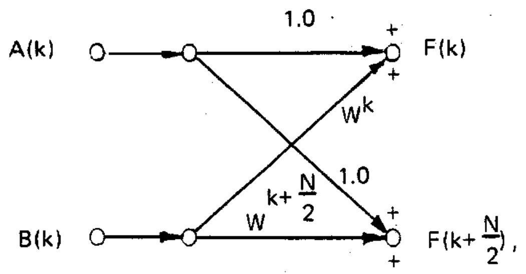
The pair of equations,

$$F(k) = A(k) + W^k B(k)$$

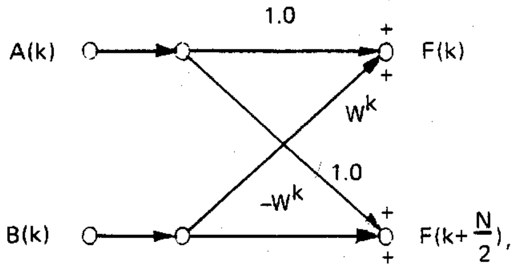
and $$F(k+\tfrac{N}{2}) = A(k) - W^k B(k),$$

is the fundamental operation of the FFT. In fact, it is the only required operation (besides calls to memory and to the trig tables).

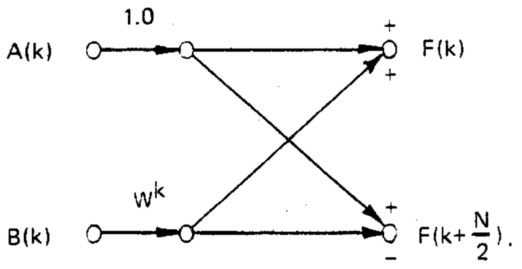
This operation is called a "butterfly" and has a signal flow representation of the form



which is equivalent to



which is also equivalent to



Now, we ask, "What has the rearrangement and factoring accomplished?" First, we have taken an N point transform which will normally require N^2 operations and replaced it with two N/2 point transforms and an additional overhead of N/2 complex multiplies and N complex additions (one multiply and two adds for each of the N/2 butterflies). The N/2 point transforms each require $(N/2)^2$ operations, so the total work load becomes

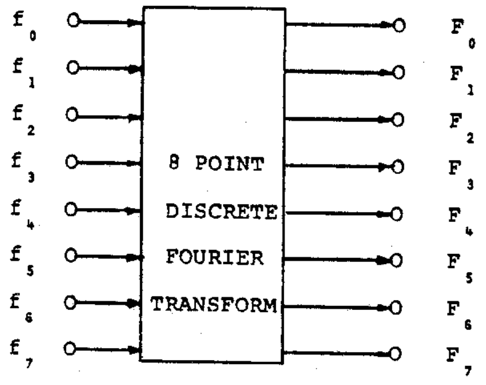
$$2\,(N/2)^2 + N/2 \text{ Mult} + N \text{ Add},$$

$$\text{or} \quad N^2/2 + N/2 \text{ Mult} + N \text{ Add}.$$

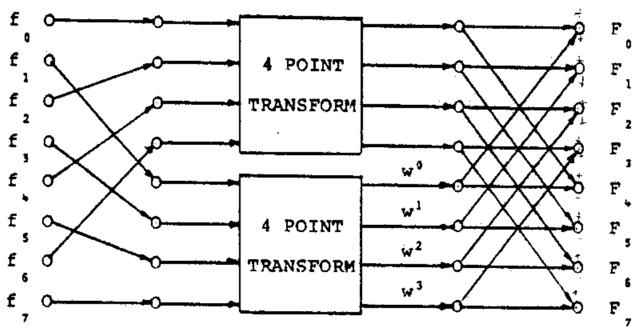
We observe that if N is large (say 1024) then $N^2/2$ is much greater than N/2, so the total work load is approximately $N^2/2$ which is one half of the work load of the direct application of the DFT summation.

Let's follow an 8 point FFT through this type of factoring. What we will do is construct an 8 point transform by two 4 point transforms. But why stop there? We can construct each 4 point transform as two 2 point transforms. We will then see that the 2 point transforms are simply butterflies, so we will be finished. In general, we construct a single N point transform as N/2 2 point transforms. We then use the two points to construct N/4 4 point transforms, and this continues till we have N/N N point transforms.

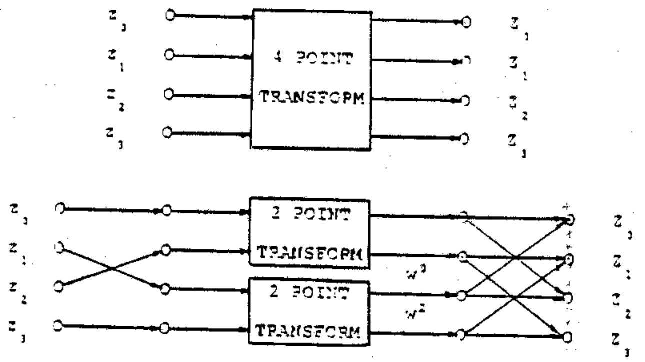
A processor that performs the computations for an 8 point discrete Fourier transform is depicted below. Eight points of data go in, and eight points of DFT come out.



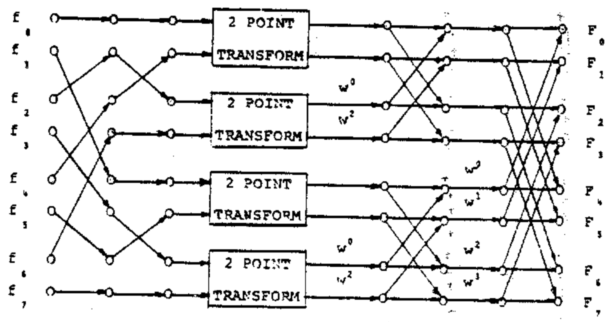
We look inside the processor and see that someone has built the 8 point transform as two 4 point transforms and then combined the outputs to realize the 8 point.



We now look inside the 4 point processor and see that the same someone has built the 4 point transform as two 2 point transforms and then combined the outputs to realize the 4 point.



The 8 point processor now has the form of



Finally we construct 2 point transforms. A 2 point transform is defined by

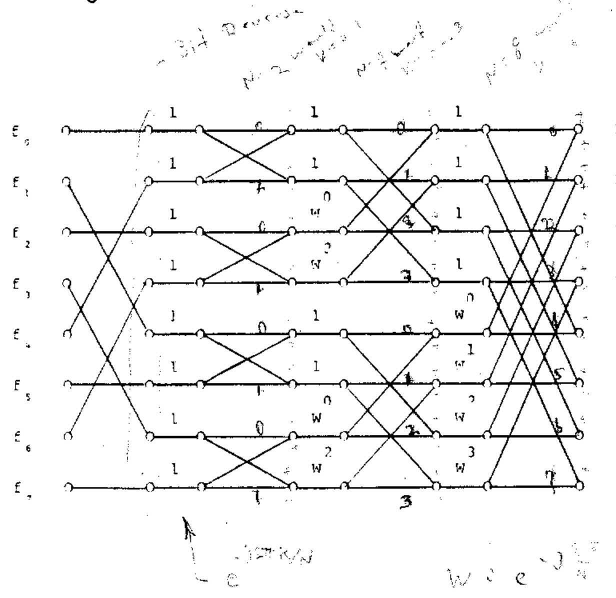
$$H(k) = \sum_{n=0}^{1} h(n)e^{-j\frac{2\pi}{2}nk},$$

$$\text{or } H(0) = h(0) + h(1),$$

$$H(1) = h(0) + e^{-j\pi}h(1) = h(0) - h(1).$$

But, of course, we recognize this sum and difference as a butterfly with a weighting factor of +1.0.

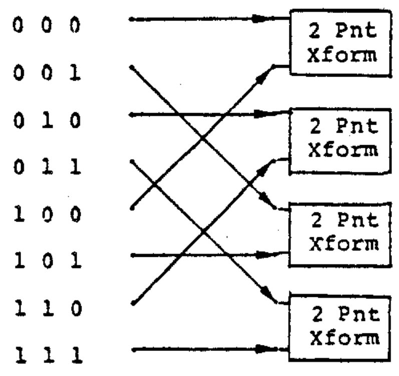
Thus the total 8 point transform with all of the rearrangements combined is of the form



REARRANGEMENT MATRIX

The rearrangement of the input or of the output of the in-place computation fast Fourier transform exhibits an interesting symmetry.

When the number of points being processed is 2^p, the rearrangement scheme is called binary reversal. If the address (as in an array) is written in binary form, the rearrangement sends the number in a given address to the address which is the binary number reversed.



We can now determine the total work count of the FFT. Each stage of the FFT will require the computation of N/2 butterflies. This represents N/2 complex multiplies (one for each butterfly) and N complex adds (two for each butterfly). Since each complex multiple is 4 real multiplies and 2 real adds, and a complex add is 2 real adds, we have a total of

$$\frac{N}{2}[4\,M + 2\,A] + N\,[2A],$$

or $N[2\,M + 3\,A]$ Real operations where M denotes a real multiply and A denotes a real add.

But there are a total of $\log_2 N$ stages for a total of

$$\log_2 N\,[N]\,[2M + 3A]$$

or $N\log_2 N\ [2M + 3A]$ real operations for an N point complex transform.

Actually for a more precise count, we recognize that the first 2 stages of the FFT contain no multiplies; hence the work count is

$$[\log_2 N - 2]\,[N]\,[2M + 3A] + [2]N[2A],$$

$$\text{or } N\left|\log_2 N\,[2M + 3A] - [4M + 2A]\right|.$$

For example, if N = 1024, the work load is 1024[16M + 28A], as opposed to direct DFT computation work load of $(1024)^2[4M + 4A]$.

RADIX 4 FFT COMPUTATIONS

Again, we start with the expression for an N point discrete Fourier transform

$$F_{(k)} = \sum_{n=0}^{N-1} f_{(n)}\, e^{-j\frac{2\pi}{N}nk}.$$

Factoring, we have

$$F_{(k)} = \sum_{n=0}^{\frac{N}{4}-1} f_{(4n)}\, e^{-j\frac{2\pi}{N}(4n)k}$$

$$+\sum_{n=0}^{\frac{N}{4}-1} f_{(4n+1)}\, e^{-j\frac{2\pi}{N}(4n+1)k}$$

$$+\sum_{n=0}^{\frac{N}{4}-1} f_{(4n+2)}\, e^{-j\frac{2\pi}{N}(4n+2)k}$$

$$+\sum_{n=0}^{\frac{N}{4}-1} f_{(4n+3)}\, e^{-j\frac{2\pi}{N}(4n+3)k}$$

$$= \sum_{n=0}^{\frac{N}{4}-1} \underbrace{f_{(4n)}\, e^{-j\frac{2\pi}{N/4}nk}}_{A_0(k)}$$

$$+ e^{-j\frac{2\pi}{N}k} \sum_{n=0}^{\frac{N}{4}-1} \underbrace{f_{(4n+1)}\, e^{-j\frac{2\pi}{N/4}nk}}_{A_1(k)}$$

$$+ e^{-j\frac{2\pi}{N}2k} \sum_{n=0}^{\frac{N}{4}-1} \underbrace{f_{(4n+2)}\, e^{-j\frac{2\pi}{N/4}nk}}_{A_2(k)}$$

$$+ e^{-j\frac{2\pi}{N}3k} \sum_{n=0}^{\frac{N}{4}-1} \underbrace{f_{(4n+3)}\, e^{-\frac{2\pi}{N/4}nk}}_{A_3(k)}$$

$$= A_0(k) + W^k A_1(k) + W^{2k} A_2(k) + W^{3k} A_3(k.$$

This expression is true for any k, and in particular is true for $[k+r\frac{N}{4}]$.

Then,

$$F(k+r\tfrac{N}{4}) = A_0(k+r\tfrac{N}{4}) + W^{(k+r\frac{N}{4})} A_1(k+r\tfrac{N}{4}) + W^{2(k+r\frac{N}{4})} A_2(k+r\tfrac{N}{4}) + W^{3(k+r\frac{N}{4})} A_3(k+r\tfrac{N}{4}).$$

But
$$A_0(k+r\tfrac{N}{4}) = A_0(k)$$
$$A_1(k+r\tfrac{N}{4}) = A_1(k)$$
$$A_2(k+r\tfrac{N}{4}) = A_2(k)$$
$$A_3(k+r\tfrac{N}{4}) = A_3(k)$$

These are output from an $\frac{N}{4}$ point transform, hence are periodic in $\frac{N}{4}$.

and
$$e^{-j\frac{2\pi}{N}r\frac{N}{4}} = e^{-j\frac{\pi}{2}r} = (-j)^r$$
$$e^{-j\frac{2\pi}{N}2r\frac{N}{4}} = e^{-j\frac{\pi}{2}2r} = (-1)^r$$
$$e^{-j\frac{2\pi}{N}3r\frac{N}{4}} = e^{-j\frac{\pi}{2}3r} = (+j)^r;$$

therefore,

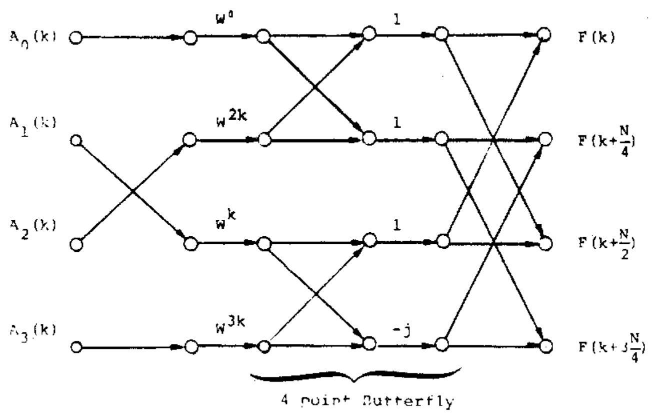
$$F(k) = A_0(k) + W^k A_1(k) + W^{2k} A_2(k) + W^{3k} A_3(k)$$

$$F(k+\frac{N}{4}) = A_0(k) - jW^k A_1(k) - W^{2k} A_2(k) + jW^{3k} A_3(k)$$

$$F(k+2\frac{N}{4}) = A_0(k) - W^k A_1(k) + W^{2k} A_2(k) - W^{3k} A_3(k)$$

$$F(k+3\frac{N}{4}) = A_0(k) + jW^k A_1(k) - W^{2k} A_2(k) - jW^{3k} A_3(k)$$

The basic 4 point butterfly is



Then to compute F(k), $F(k+\frac{N}{4})$, $F(k+2\frac{N}{4})$ and $F(k+3\frac{N}{4})$, we need to extract from the four N/4 point transforms the outputs $A_0(k)$, $A_1(k)$, $A_2(k)$ and $A_3(k)$, and then form the following set of computations

$$\left.\begin{aligned} \text{TEMP } 1 &= W^k A_1(k) \\ \text{TEMP } 2 &= W^{2k} A_2(k) \\ \text{TEMP } 3 &= W^{3k} A_3(k) \end{aligned}\right\} \text{3 complex multiplies,}$$

$$\left.\begin{aligned} \text{TEMP } 4 &= A_0(k) + \text{TEMP2} \\ \text{TEMP } 5 &= A_0(k) - \text{TEMP2} \\ \text{TEMP } 6 &= \text{TEMP1} + \text{TEMP3} \\ \text{TEMP } 7 &= \text{TEMP1} - \text{TEMP3} \end{aligned}\right\} \text{4 complex adds,}$$

and

$$\left.\begin{aligned} F(k) &= \text{TEMP4} + \text{TEMP6} \\ F(k+\frac{N}{2}) &= \text{TEMP4} - \text{TEMP6} \\ F(k+\frac{N}{4}) &= \text{TEMP5} - j\text{TEMP7} \\ F(k+3\frac{N}{4}) &= \text{TEMP5} + j\text{TEMP7} \end{aligned}\right\} \text{4 complex adds,}$$

for a total of 3 complex multiplies and 8 complex adds, or a total of 12 real multiplies and 22 real adds. Thus, each stage of the N point transform will require N/4 groups of 12 real multiplies and 22 real adds, or

$$\frac{N}{4}(12M + 22A),$$

or N(3M + 5.5A) real operations per stage.

But there is a total of $\frac{1}{2}\log_2 N$ stages in the transform, for a total of

$$\frac{1}{2}\log_2 N[N(3M + 5.5A)],$$

or $N \log_2 N[1.5M + 2.75A]$ real operations per transform.

Again for a more precise count, we recognize that the first stage of the FFT contains no multiplies; hence the count is

$$[\frac{1}{2}\log_2 N-1]\ N\ [3M + 5.5A] + N\ [4A],$$

$$\text{or} \quad N \log_2 N\ [1.5M + 2.75A] - N\ [3M + 1.5A],$$

or $N[\log_2 N(1.5M + 2.75A) - (3M + 1.5A)]$

real operations.

Now let us compare the operation count for the Radix 2 and for the Radix 4 transform. Further, let us assume the operation (cycle time) for a multiply is equivalent to 10 operations (cycle times) of an add. Also assume $N = 1024$, and $\log_2 N = 10$.

Radix 2.

$1024\ (\log_2 1024)\ (2M + 3A)$

$1024\ (10)\ (20 + 3)\ A$

$1024\ (230)\ A$

Radix 4.

$1024\ (\log_2 1024)\ (1.5M + 2.75A)$

$1024\ (10)\ (15 + 2.75)\ A$

$1024\ (177.5)\ A$

Or more precisely;

Radix 2.

$1024[10(20 + 3) - (40 + 2)]A$

$1024[230 - 42]A$

$1024[188]A$

Radix 4.

$1024[10\ (15 + 2.75) - (30 + 1.5)]A$

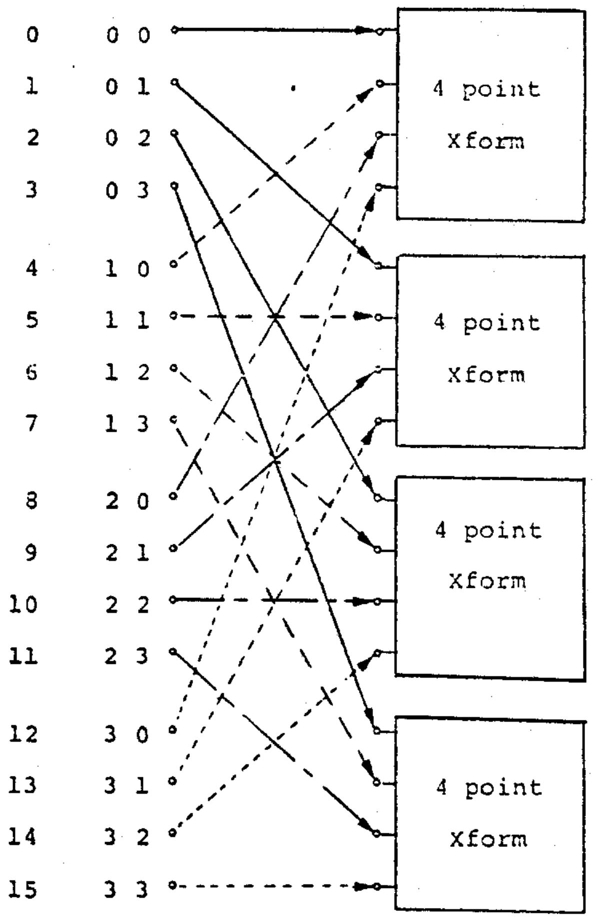
$1024[177.5 - 31.5]A$

$1024[146]A$

Thus we see that the Radix 4 transform can be performed in approximately 75% of the time in which a given machine performs a Radix 2 transform. Of course, the transform size must be a power of 4 to be performed as a Radix 4. If the size of the transform is not a power of 4, say 512 for instance, the economy of the Radix 4 can be used to perform two 256 point transforms and then a Radix 2 transform to combine the results for the final transform.

REARRANGEMENT MATRIX FOR RADIX 4 FFT COMPUTATION

When the number of points being processed is 4^p, the rearrangement scheme for the Radix 4 fast Fourier transform is called Mod 4 address reversal. If the address (as in the array) is written in base 4, the rearrangement sends the number in a given address to the address which is the base 4 number reversed. The reversals corresponding to a 16 point transform performed Radix 4 are



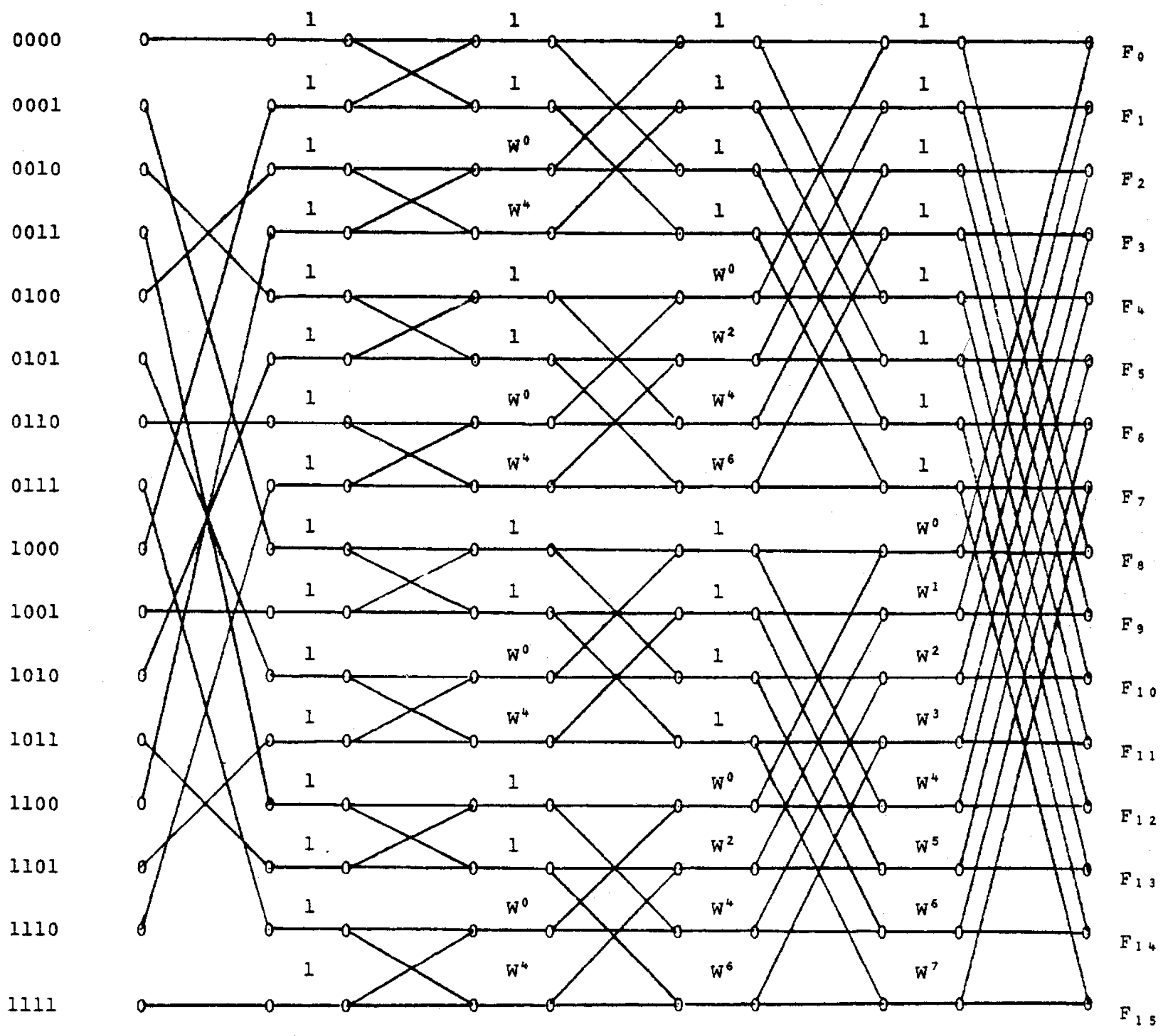


Figure 52. Radix 2 – 16 Point FFT

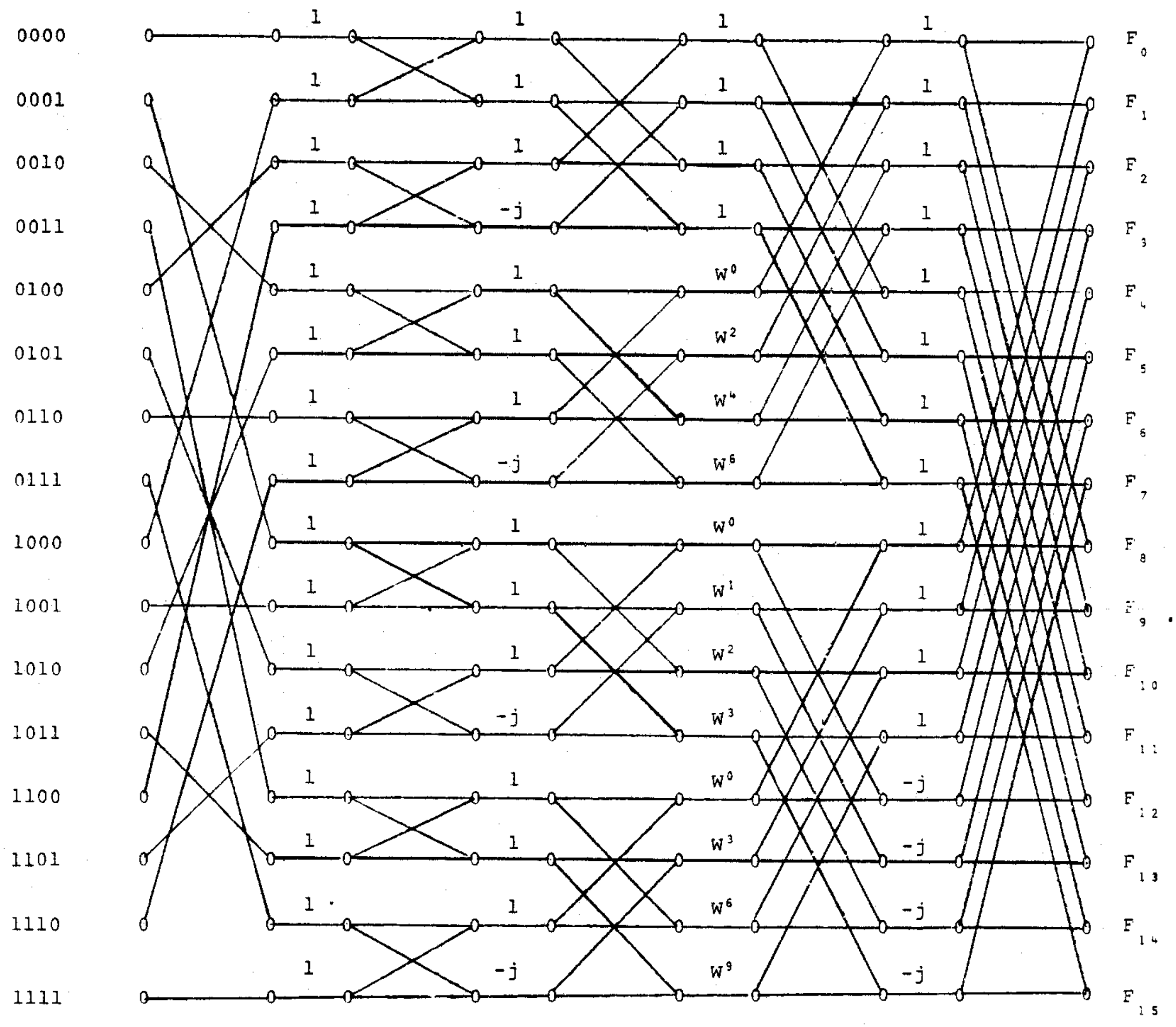
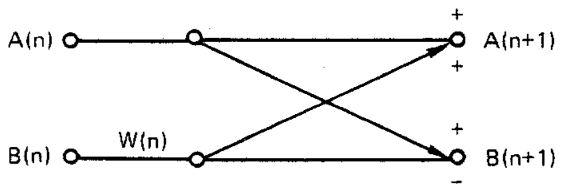


Figure 53. Radix 4 – 16 Point FFT

SCALING CONSIDERATIONS

The FFT as an algorithm processes an array of data by successive passes at the array. At each pass, the algorithm performs N/2 butterflies, each butterfly picking up two complex numbers and returning two complex numbers to the same addresses. The numbers returned to memory by the butterfly are larger than the numbers picked from memory by the butterfly. In fact, we will now demonstrate that the incoherent part of the data has an expected increase of $\sqrt{2}$ per stage, while the coherent part of the data has an expected increase by 2 per stage with a maximum increase per stage of $1 + \sqrt{2}$ or 2.414.

A typical butterfly is



At the (n+1)th pass of the data, the butterfly selects two data points A(n) and B(n) and returns to memory A(n+1) and B(n+1). Let us examine the squared magnitude of A(n+1) and of B(n+1)

$$A(n+1) = A(n) + W(n)\,B(n)$$
$$B(n+1) = A(n) - W(n)\,B(n)$$

$$|A(n+1)|^2 = |A(n)|^2 + |B(n)|^2 + A(n)\,W^*(n)\,B^*(n) + A^*(n)\,W(n)\,B(n)$$

$$|B(n+1)|^2 = |A(n)|^2 + |B(n)|^2 - A(n)\,W^*(n)\,B^*(n) - A^*(n)\,W(n)\,B(n)$$

Adding the pair $|A(n+1)|^2$ and $|B(n+1)|^2$, we have

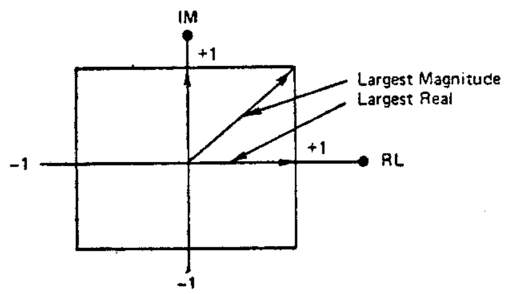
$$|A(n+1)|^2 + |B(n+1)|^2 = 2[\,|A(n)|^2 + |B(n)|^2\,].$$

For random data, the expected value of $|A(n+1)|^2$ is the same as the expected value of $|B(n+1)|^2$; so that,

$$|A(n+1)|^2 = 2\ |A(n)|^2,$$
$$\text{or } |A(n+1)| = \sqrt{2}\ |A(n)|.$$

MAXIMUM BUTTERFLY GAIN

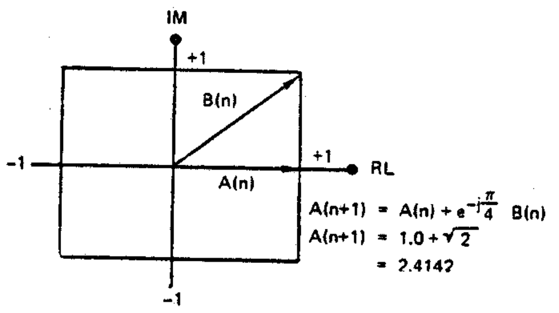
The square which represents the range of the complex numbers stored in memory of the FFT processor is



Both the real part and the imaginary part of the complex word is constrained to fall between +1 and −1, which constrains the magnitude to be between 0 and $\sqrt{2}$. The first pass of the FFT performs only complex addition and subtraction. Since the input to the butterfly is constrained to a maximum real and imaginary component of ±1, the output is constrained to a maximum component of ±2. Either a prescale by 1/2 is required prior to entering the butterfly or an extra bit must be available in memory to accommodate this potential gain of two. Automatic prescaling shifts out the least significant bit and represents significant processing noise, particularly in the case for which no scaling is necessary for the pass. Automatic prescaling is better then no scaling, but not much better, and should be avoided. A much more desirable form of scaling is data dependent scaling in which scaling (before or after the butterfly) is performed only if one or more butterflies in an FFT pass will (or does) overflow without the scaling.

The second pass of the FFT performs rotations through complex multiplications by W(n) of ±90°. Thus again, if the input components are constrained to ±1, the output components are constrained to ±2. Thus, the second stage of the FFT can at most exhibit a gain of two. The scaling routine must allow for this possible extra bit.

The third and subsequent passes of the FFT perform rotations through a larger set of angles which include the angle 45°. This represents the most stringent scaling task. A complex number with maximum magnitude ($\sqrt{2}$) can be rotated through 45° and added to a maximum real or imaginary component of +1 for a maximum output component of $1+\sqrt{2}$. This is demonstrated by


IM
+1
B(n)
-1
+1
RL
A(n)
-1
$A(n+1) = A(n) + e^{-j\frac{\pi}{4}} B(n)$
$A(n+1) = 1.0 + \sqrt{2}$
$= 2.4142$

Thus the maximum gain per stage is greater than 2.0 and is actually 2.4142.

Prescaling data before the butterfly can represent a maximum loss of two bits of significance, which is, of course, undesirable. Post scaling (up to two bits) after the butterfly upon detection of overflow will require multiple passes through the array to scale data already returned to the array by previous butterflies before the detected overflow or overflows. For very high speed processing, this is unacceptable.

DATA DEPENDENT SCALING

Both the highest speed operation and the best computational noise performance of the FFT are realized by an architecture which allows the up to two overflow bits from the butterfly to be returned to memory. This requires the 16 bits for data and 2 extra bits for overflow for memory of 18 bits per complex component (36 bits per complex word).

If any word in memory has an overflow bit set, all words are right shifted (the necessary one or two bits) as they are called from memory for the next pass of the FFT. Of course, no scaling is performed unless the overflow bit(s) have been set by the previous pass. Then, total number of right shifts, p, executed during the FFT is available at the output of the FFT as a scale factor of 2^p. This scale factor must be applied to the output of the FFT so that total processing gain is properly presented. Right shifting the entire array when required at each pass and accumulating the count of the number of shifts is known as block floating point processing.

SIMULTANEOUS N POINT (REAL) TRANSFORMS

Suppose we have two real sequences

$$f(n) \quad ; \quad n = 0, 1, 2, \ldots, N-1$$

$$g(n) \quad ; \quad n = 0, 1, 2, \ldots, N-1,$$

and we wish to obtain the transforms of both sequences. We have seen that the transform of the real sequence $f(n) + j0$, that is, $F(k) = RL[F(k)] + jIM[F(k)]$, exhibits the following symmetries;

$$\left.\begin{aligned} RL[F(k)] &= RL[F(N-k)] \\ &= RL[F(-k)] \end{aligned}\right\} \text{ i.e., RL part is even,}$$

$$\left.\begin{aligned} IM[F(k)] &= -IM[F(N-k)] \\ &= -IM[F(-k)] \end{aligned}\right\} \text{ i.e., IM part is odd.}$$

Then the transform of the "complex" sequence $f(n) + j\,g(n)$, (the first sequence loaded into the real array, and the second sequence loaded into the imaginary array) can be decomposed into the complex sequences F(k) and G(k) by virtue of the even and odd properties of their respective real and imaginary parts.

If we define the transform of the complex input sequence as

$$Z(k) = X(k) + jY(k),$$

where

$$\begin{aligned} Z(k) &= FFT[f(n) + jg(n)] \\ &= FFT[f(n)] + j\,FFT[g(n)] \\ &= F(k) + j\,G(k). \end{aligned}$$

$$\text{But } F(k) = RL[F(k)] + j\,IM[F(k)]$$

$$= EV[F(k)] + j\,OD[F(k)],$$

and $G(k) = RL[G(k)] + j\,IM[G(k)]$

$= EV[G(k)] + j\,OD[G(k)]$,

then

$$\begin{aligned} Z(k) &= \{RL[F(k)] + j\,IM[F(k)]\} + \\ &\quad j\,\{RL[G(k)] + j\,IM[G(k)]\} \\ &= \{RL[F(k)] - IM[G(k)]\} + \\ &\quad j\,\{IM[F(k)] + RL[G(k)]\} \\ &= \{\underbrace{EV[F(k)] - OD[G(k)]}_{X(k)}\} + \\ &\quad j\,\{\underbrace{OD[F(k)] + EV[G(k)]}_{Y(k)}\}, \end{aligned}$$

from which we obtain the following;

$RL[F(k)] = X(k) + X(-k)\,;\ k = 0, \ldots \frac{N}{2}-1$

$RL[F(-k)] = RL[F(k)]$;

$IM[F(k)] = Y(k) - Y(-k)\ ;\ k = 0, \ldots \frac{N}{2}-1$

$IM[F(-k)] = -IM[F(k)]$;

$RL[G(k)] = Y(k) + Y(-k)\ ;\ k = 0, \ldots \frac{N}{2}-1$

$RL[G(-k)] = RL[G(k)]$;

$IM[G(k)] = X(-k) - X(k)\ ;\ k = 0, \ldots \frac{N}{2}-1$

$IM[G(-k)] = -IM[G(k)]$.

Thus we see a method of performing two N point transforms in the time duration necessary to perform one N point complex transform plus the overhead to compute the odd and the even parts of the real and of the imaginary parts of the resultant transform.

DOUBLE LENGTH (2N) REAL TRANSFORMS

Suppose we have one real sequence

$f(n)\ ;\ n = 0, 1, 2, \ldots, 2N-1$

and we only have access to an N point (complex) transform. We can perform the double length transform by partitioning the sequence into two real sequences of length N, perform the simultaneous transforms outlined on the previous page and then combine the results using the following observations;

$$\begin{aligned} F(k) &= \sum_{n=0}^{2N-1} f(n)\, e^{-j\frac{2\pi}{2N}nk} \\ &= \sum_{n=0}^{N-1} f(2n)\, e^{-j\frac{2\pi}{2N}(2n)k} + \\ &\quad \sum_{n=0}^{N-1} f(2n+1)\, e^{-j\frac{2\pi}{2N}(2n+1)k} \\ &= \underbrace{\sum_{n=0}^{N-1} f(2n)\, e^{-j\frac{2\pi}{N}nk}}_{A(k)} + \\ &\quad e^{-j\frac{2\pi}{2N}k} \underbrace{\sum_{n=0}^{N-1} f(2n+1)\, e^{-j\frac{2\pi}{N}nk}}_{B(k)} \\ &= A(k) + e^{-j\frac{2\pi}{2N}k}\, B(k), \end{aligned}$$

where A(k) and B(k) are the transforms of the even indexed and of the odd indexed data points. If the even indexed data points are loaded into the real array and if the odd indexed data points are loaded into the imaginary array, then the decomposition of the previous section must be used to obtain A(k) and B(k). If we denote $X(k) + jY(k)$ as the transform of the sequence $f(2n) + j\,f(2n+1)$, we can obtain the desired transform F(k) from

$$\begin{aligned}F(k) &= A(k) + [C(k) - jS(k)]\ B(k) \\ &= [X(k)+X(-k)]+j[Y(k)-Y(-k)] \\ &\quad +[C(k)-jS(k)]\{[Y(k)+Y(-k)]-j[X(k)-X(-k)]\} \\ &= [X(k)+X(-k)]+C(k)\ [Y(k)+Y(-k)]-S(k) \\ &\quad [X(k)-X(-k)]+j\{[Y(k)-Y(-k)]-C(k) \\ &\quad [X(k)-X(-k)]-S(k)\ [Y(k)+Y(-k)]\}.\end{aligned}$$

In terms of storage, the original 2N data points are loaded into the 2N point array (N real, N imaginary). N complex numbers are generated by the transform and the linear sum above is used to generate the first N complex numbers of the 2N point transform F(k). The second set of N points are not computed because, one, there is no room for them without auxiliary storage, and two, they are available in terms of the conjugate images of the first N points;

$$\text{i.e.}\quad F(2N-k) = F(-k) = F^*(k),$$

$$\text{or}\quad RL[F(k)] = RL[F(2N-k)]$$

$$IM[F(k)] = -IM[F(2N-k)].$$

OTHER VARIANTS OF THE FAST FOURIER TRANSFORM

The algorithm we derived as the fast Fourier transform is characterized by the need to re-order the input data. We called this "decimation in time". The algorithm was also characterized by in-place computations. This is one of many possible implementations with these characteristics. There are algorithms which allow the data to be processed in natural order and still allow in - place computations. These algorithms will have to have the output rearranged, and the rearrangement is identical to the one we derived. Other forms of the algorithm allow input and output data to be available in natural order, but we then lose the in-place computation and the symmetries so implied.

Let us derive (rather rapidly) some of these other forms of the algorithm. Our derivation will rely upon two observations; first, the reordering matrix [R] is its own inverse, i.e. [R] [R] = [I], and second, the transform matrix [W] is its own transpose, i.e. $[W]^T = [W]$.

Now, starting with the transform in its factored form

$$\bar{F} = [[W]]\ \bar{f}$$

$$\bar{F} = [[A]]\ [[B]]\ [[C]]\ [R]\ \bar{f},$$

we identify this as a Type I algorithm.

Now, since [R] [R] = [I], we can interpose the identity matrix throughout the Type I algorithm.

$$\text{Then }\bar{F} = \underbrace{[R]\,[R]}_{[I]}\,[[A]]\,\underbrace{[R]\,[R]}_{[I]}\,[[B]]$$

$$\underbrace{[R]\,[R]}_{[I]}\,[[C]]\,[R]\ \bar{f}.$$

Now, regrouping an [R] as a pre and a post multiply to each of [[A]], [[B]], and [[C]],

$$\bar{F} = [R]\ \{[R]\,[[A]]\,[R]\}\ \ \{[R]\,[[B]]\,[R]\}$$
$$\{[R]\,[[C]]\,[R]\}\ \bar{f}.$$

We identify this as a Type II algorithm.

$$\text{Now with }\bar{F} = [[W]]\ \bar{f}$$

$$\text{or }\bar{F} = [[W]]^T\ \bar{f}$$

$$\bar{F} = \{\,[[A]]\,[[B]]\,[[C]]\,[R]\,\}^T\ \bar{f}$$

$$\bar{F} = [R]^T\,[[C]]^T\,[[B]]^T\,[[A]]^T\ \bar{f}$$

$$\text{But }[R]^T = [R],$$

$$\text{thus,}\quad \bar{F} = [R]\,[[C]]^T\,[[C]]^T\,[[A]]^T\ \bar{f}.$$

We identify this as a Type III algorithm.

Now returning to the Type II algorithm,

$$\begin{aligned}\bar{F} &= [[W]]\ \bar{f} \\ &= [[W]]^T\ \bar{f} \\ &= \{\,[R]\,([R]\,[[A]]\,[R])\ ([R]\,[[B]]\,[R]) \\ &\qquad ([R]\,[[C]]\,[R])\,\}^T\ \bar{f} \\ &= ([R]\,[[C]]\,[R])^T\ ([R]\,[[B]]\,[R])^T \\ &\qquad ([R]\,[[A]]\,[R])^T\,[R]^T\ \bar{f}.\end{aligned}$$

But $[R]^T = [R]$,

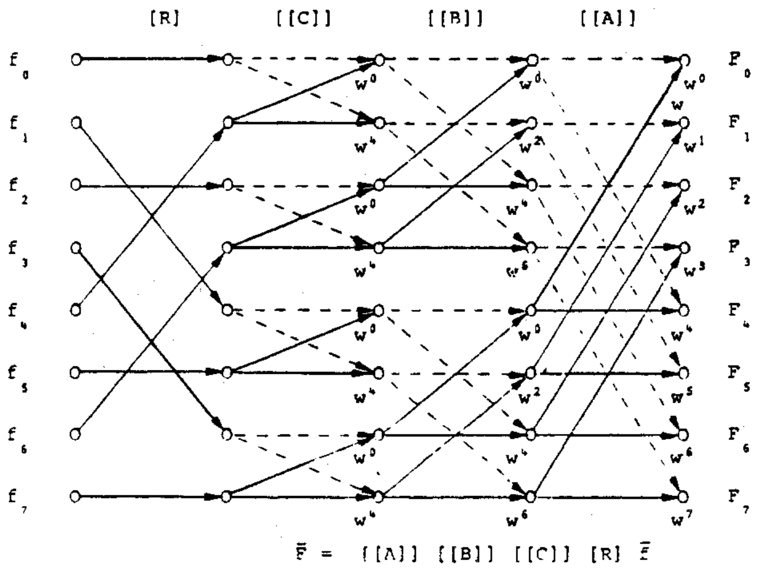
thus $\bar{F} = ([R]\,[[C]]\,[R])^T \; ([R]\,[[B]]\,[R])^T$
$([R]\,[[A]]\,[R])^T \; [R]\; \bar{f}$,

or $\bar{F} = ([R]\,[[C]]^T\,[R]) \; ([R]\,[[B]]^T\,[R])$
$([R]\,[[A]]^T\,[R]) \; [R]\; \bar{f}$.

We identify this as a Type IV algorithm.

We present the signal flow graphs of the TYPE I through the TYPE IV algorithms on the next pages. Other algorithms can be formed by omitting the [R] [R] products from any of the interposed positions used to construct the Type II algorithm. These forms would not reflect the in-place capabilities of the fast Fourier transform.

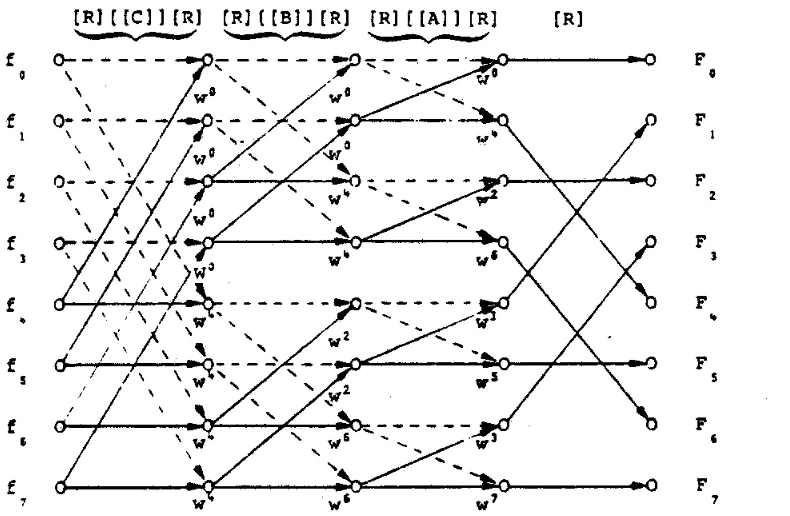
$$\underset{[[A]]}{\begin{bmatrix} 0 & & & & 0 & & & \\ & 0 & & & & 1 & & \\ & & 0 & & & & 2 & \\ & & & 0 & & & & 3 \\ 0 & & & & 4 & & & \\ & 0 & & & & 5 & & \\ & & 0 & & & & 6 & \\ & & & 0 & & & & 7 \end{bmatrix}} \underset{[[B]]}{\begin{bmatrix} 0 & & 0 & & & & & \\ & 0 & & 2 & & & & \\ 0 & & 4 & & & & & \\ & 0 & & 6 & & & & \\ & & & & 0 & & 0 & \\ & & & & & 0 & & 2 \\ & & & & 0 & & 4 & \\ & & & & & 0 & & 6 \end{bmatrix}} \underset{[[C]]}{\begin{bmatrix} 0 & 0 & & & & & & \\ 0 & 4 & & & & & & \\ & & 0 & 0 & & & & \\ & & 0 & 4 & & & & \\ & & & & 0 & 0 & & \\ & & & & 0 & 4 & & \\ & & & & & & 0 & 0 \\ & & & & & & 0 & 4 \end{bmatrix}} \underset{[R]}{\begin{bmatrix} 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \end{bmatrix}}$$



$\bar{F} = [[A]]\;[[B]]\;[[C]]\;[R]\;\bar{f}$

Figure 54. Type I Transform

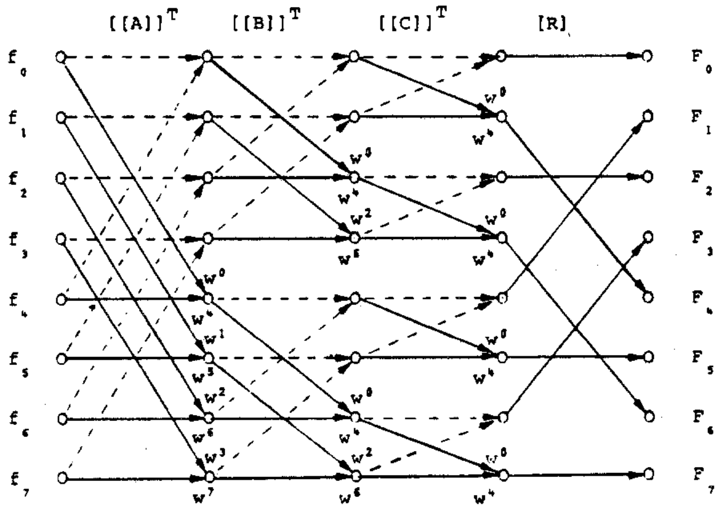
$$\underset{[R]}{\begin{bmatrix}1&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&1&\cdot&\cdot&\cdot\\\cdot&\cdot&1&\cdot&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&1&\cdot\\\cdot&1&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&1&\cdot&\cdot\\\cdot&\cdot&\cdot&1&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&1\end{bmatrix}}
\underset{([R][[A]][R])}{\left[\begin{bmatrix}0&0&&&&&&\\0&4&&&&&&\\&&0&2&&&&\\&&0&6&&&&\\&&&&0&1&&\\&&&&0&5&&\\&&&&&&0&3\\&&&&&&0&7\end{bmatrix}\right]}
\underset{([R][[B]][R])}{\left[\begin{bmatrix}0&&0&&&&&\\&0&&0&&&&\\0&&4&&&&&\\&0&&4&&&&\\&&&&0&&2&\\&&&&&0&&2\\&&&&0&&6&\\&&&&&0&&6\end{bmatrix}\right]}
\underset{([R][[C]][R])}{\left[\begin{bmatrix}0&&&&0&&&\\&0&&&&0&&\\&&0&&&&0&\\&&&0&&&&0\\0&&&&4&&&\\&0&&&&4&&\\&&0&&&&4&\\&&&0&&&&4\end{bmatrix}\right]}$$



$\bar{F}$ = [R] ([R][[C]][R]) ([R][[B]][R]) ([R][[A]][R]) $\bar{f}$

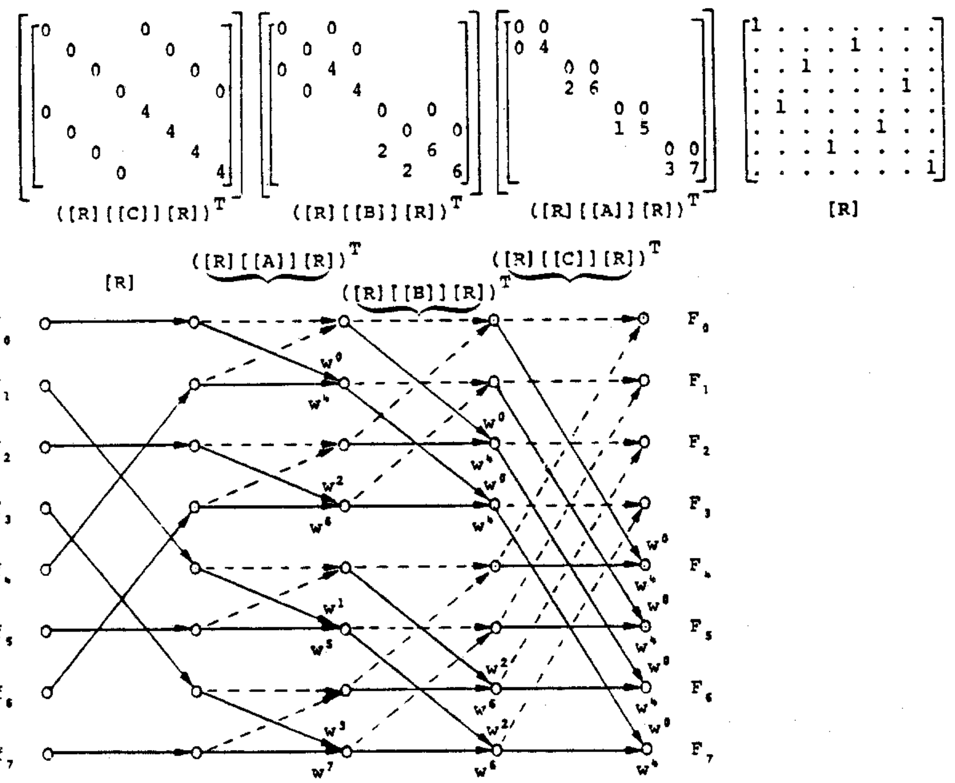
Figure 55. Type II Transform

$$\underset{[R]}{\begin{bmatrix}1&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&1&\cdot&\cdot&\cdot\\\cdot&\cdot&1&\cdot&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&1&\cdot\\\cdot&1&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&1&\cdot&\cdot\\\cdot&\cdot&\cdot&1&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&1\end{bmatrix}}
\underset{[[C]]^T}{\left[\begin{bmatrix}0&0&&&&&&\\0&4&&&&&&\\&&0&0&&&&\\&&0&4&&&&\\&&&&0&0&&\\&&&&0&4&&\\&&&&&&0&0\\&&&&&&0&4\end{bmatrix}\right]}
\underset{[[B]]^T}{\left[\begin{bmatrix}0&&0&&&&&\\&0&&0&&&&\\0&&4&&&&&\\&2&&6&&&&\\&&&&0&&0&\\&&&&&0&&0\\&&&&0&&4&\\&&&&&2&&6\end{bmatrix}\right]}
\underset{[[A]]^T}{\left[\begin{bmatrix}0&&&&0&&&\\&0&&&&0&&\\&&0&&&&0&\\&&&0&&&&0\\0&&&&4&&&\\&1&&&&5&&\\&&2&&&&6&\\&&&3&&&&7\end{bmatrix}\right]}$$



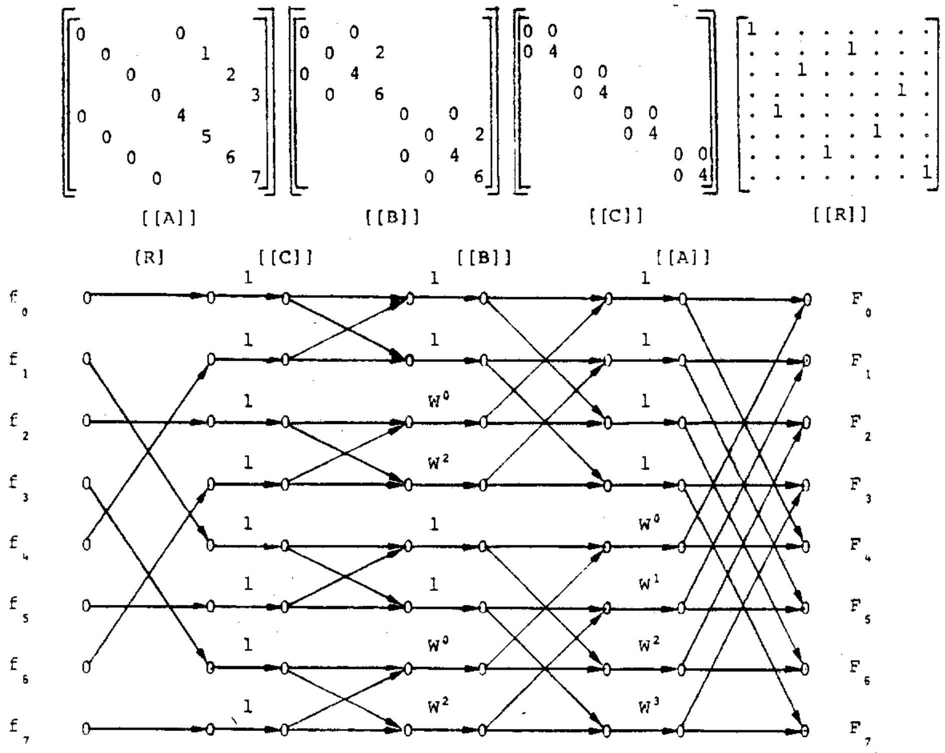
$\bar{F}$ = [R] $[[C]]^T$ $[[B]]^T$ $[[A]]^T$ $\bar{f}$

Figure 56. Type III Transform


([R][[C]][R])^T
([R][[B]][R])^T
([R][[A]][R])^T
[R]
f_0 f_1 f_2 f_3 f_4 f_5 f_6 f_7
F_0 F_1 F_2 F_3 F_4 F_5 F_6 F_7

$$\bar{F} = ([R][[C]][R])^T\ ([R][[B]][R])^T\ ([R][[A]][R])^T\ [R]\ \bar{f}$$

Figure 57. Type IV Transform


[[A]]
[[B]]
[[C]]
[[R]]
[R]
f_0 f_1 f_2 f_3 f_4 f_5 f_6 f_7
F_0 F_1 F_2 F_3 F_4 F_5 F_6 F_7

$$\bar{F} = [[A]]\ [[B]]\ [[C]]\ [R]\ \bar{f}$$

Figure 58. Type I Transform

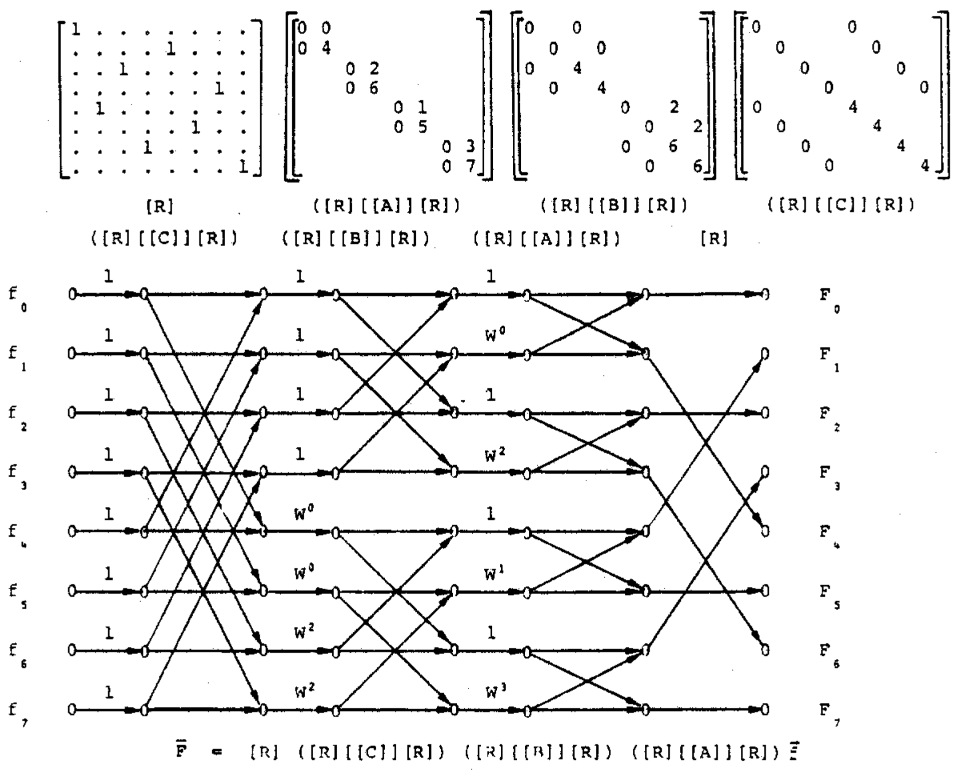


Figure 59. Type II Transform

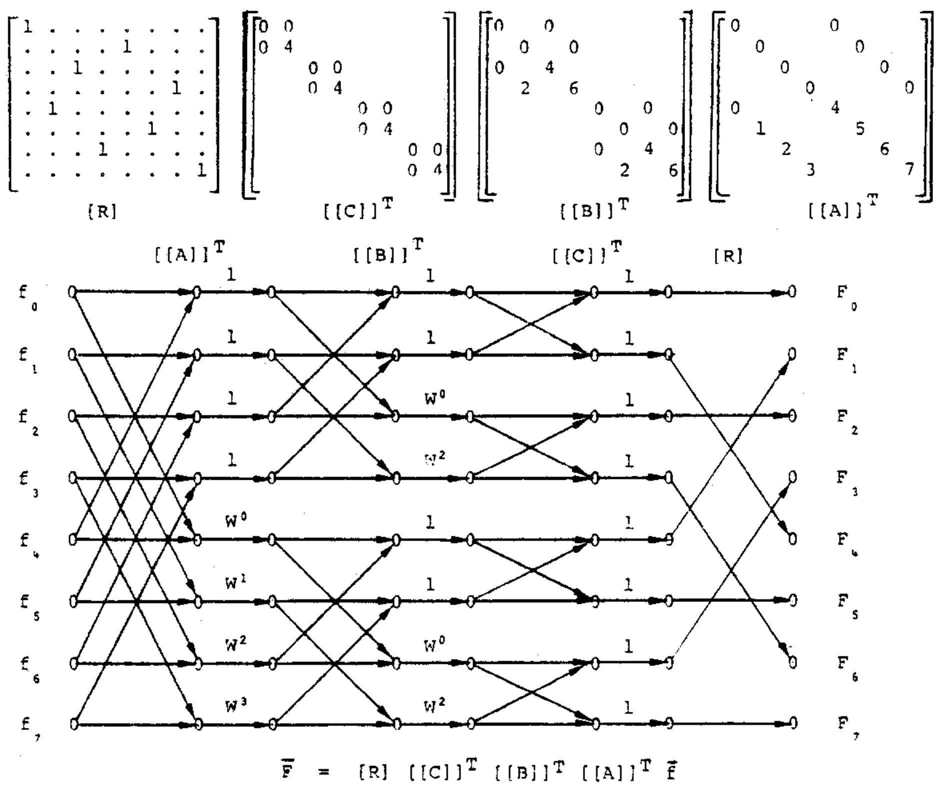


Figure 60. Type III Transform

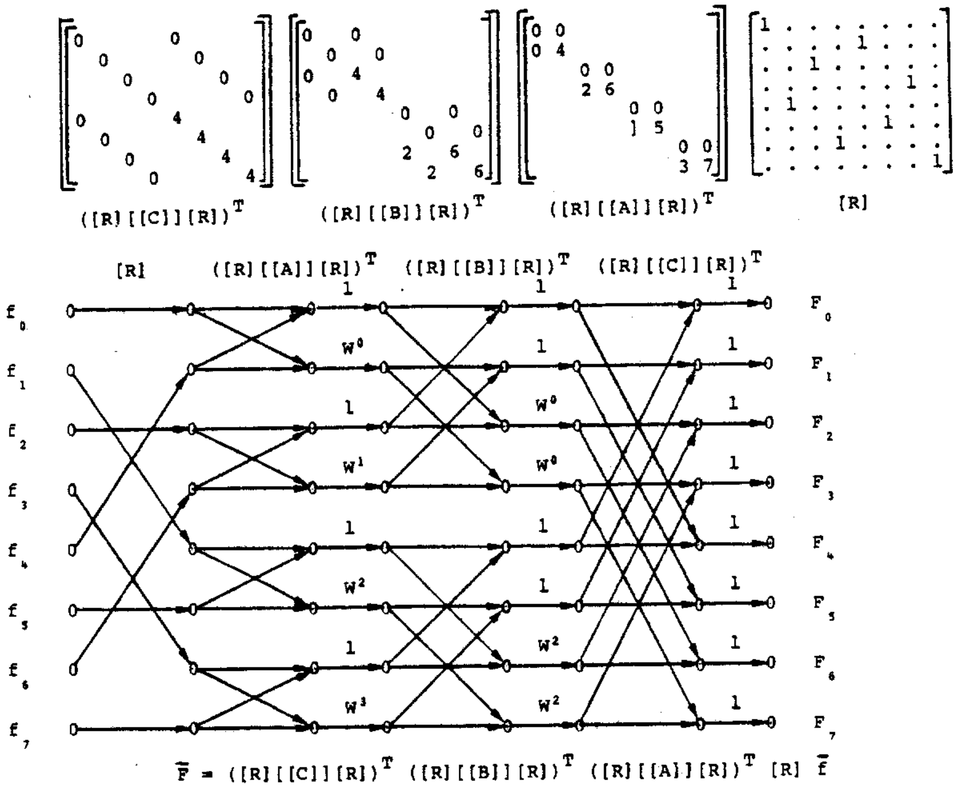

([R][[C]][R])^T
([R][[B]][R])^T
([R][[A]][R])^T
[R]
f_0 f_1 f_2 f_3 f_4 f_5 f_6 f_7
F_0 F_1 F_2 F_3 F_4 F_5 F_6 F_7
W^0 W^1 W^2 W^3
$\bar{F}$ = ([R][[C]][R])^T ([R][[B]][R])^T ([R][[A]][R])^T [R] $\bar{f}$

Figure 61. Type IV Transform

References:

1. Bracewell, Ron, *The Fourier Transform and Its Applications,* McGraw-Hill, Stanford University, 1965.
2. Gingrass, Don, *Time Series Windows for Improving Discrete Spectra Estimations,* NUC TECH Note 715, April 1972.
3. harris, frederic j., *Digital Signal Processing,* Classnotes, San Diego State University, 1973.

WINDOW		HIGHEST SIDE LOBE LEVEL (dB)	SIDE LOBE FALL OFF (dB/OCT)	COHERENT GAIN	EQUIV. NOISE BW (BINS)	3.0-dB BW (BINS)	SCALLOP LOSS (dB)	WORSE CASE PROCESS LOSS (dB)	6.0-dB BW (BINS)
RECTANGLE		−13	−6	1.00	1.00	0.89	3.92	3.92	1.21
TRIANGLE		−27	−12	0.50	1.33	1.28	1.82	3.07	1.78
$\cos^{\alpha}(X)$	$\alpha = 1.0$	−23	−12	0.64	1.23	1.20	2.10	3.01	1.65
	$\alpha = 2.0$	−32	−18	0.50	1.50	1.44	1.42	3.18	2.00
	$\alpha = 3.0$	−39	−24	0.42	1.73	1.66	1.08	3.47	2.32
	$\alpha = 4.0$	−47	−30	0.38	1.94	1.86	0.86	3.75	2.59
HAMMING		−43	−6	0.54	1.36	1.30	1.78	3.10	1.81
RIESZ		−21	−12	0.67	1.20	1.16	2.22	3.01	1.59
RIEMANN		−26	−12	0.59	1.30	1.26	1.89	3.03	1.74
DE LA VALLE-POUSSIN		−53	−24	0.38	1.92	1.82	0.90	3.72	2.55
TUKEY	$\alpha = 0.25$	−14	−18	0.88	1.10	1.01	2.96	3.39	1.38
	$\alpha = 0.50$	−15	−18	0.75	1.22	1.15	2.24	3.11	1.57
	$\alpha = 0.75$	−19	−18	0.63	1.36	1.31	1.73	3.07	1.80
BOHMAN		−46	−24	0.41	1.79	1.71	1.02	3.54	2.38
POISSON	$\alpha = 2.0$	−19	−6	0.44	1.30	1.21	2.09	3.23	1.69
	$\alpha = 3.0$	−24	−6	0.32	1.65	1.45	1.46	3.64	2.08
	$\alpha = 4.0$	−31	−6	0.25	2.08	1.75	1.03	4.21	2.58
HANNING-POISSON	$\alpha = 0.5$	−35	−18	0.43	1.61	1.54	1.26	3.33	2.14
	$\alpha = 1.0$	NONE	−18	0.38	1.73	1.64	1.11	3.50	2.30
	$\alpha = 2.0$	NONE	−18	0.29	2.02	1.87	0.87	3.94	2.65
CAUCHY	$\alpha = 3.0$	−31	−6	0.42	1.48	1.34	1.71	3.40	1.90
	$\alpha = 4.0$	−35	−6	0.33	1.76	1.50	1.36	3.83	2.20
	$\alpha = 5.0$	−30	−6	0.28	2.06	1.68	1.13	4.28	2.53
GAUSSIAN	$\alpha = 2.5$	−42	−6	0.51	1.39	1.33	1.69	3.14	1.86
	$\alpha = 3.0$	−55	−6	0.43	1.64	1.55	1.25	3.40	2.18
	$\alpha = 3.5$	−69	−6	0.37	1.90	1.79	0.94	3.73	2.52
DOLPH-TCHEBYSHEV	$\alpha = 2.5$	−50	0	0.53	1.39	1.33	1.70	3.12	1.85
	$\alpha = 3.0$	−60	0	0.48	1.51	1.44	1.44	3.23	2.01
	$\alpha = 3.5$	−70	0	0.45	1.62	1.55	1.25	3.35	2.17
	$\alpha = 4.0$	−80	0	0.42	1.73	1.65	1.10	3.48	2.31
KAISER-BESSEL	$\alpha = 2.0$	−46	−6	0.49	1.50	1.43	1.46	3.20	1.99
	$\alpha = 2.5$	−57	−6	0.44	1.65	1.57	1.20	3.38	2.20
	$\alpha = 3.0$	−69	−6	0.40	1.80	1.71	1.02	3.56	2.39
	$\alpha = 3.5$	−82	−6	0.37	1.93	1.83	0.89	3.74	2.57
BARCILON-TEMES	$\alpha = 3.0$	−53	−6	0.47	1.56	1.49	1.34	3.27	2.07
	$\alpha = 3.5$	−58	−6	0.43	1.67	1.59	1.18	3.40	2.23
	$\alpha = 4.0$	−68	−6	0.41	1.77	1.69	1.05	3.52	2.36

TABLE 1. WINDOWS AND FIGURES OF MERIT

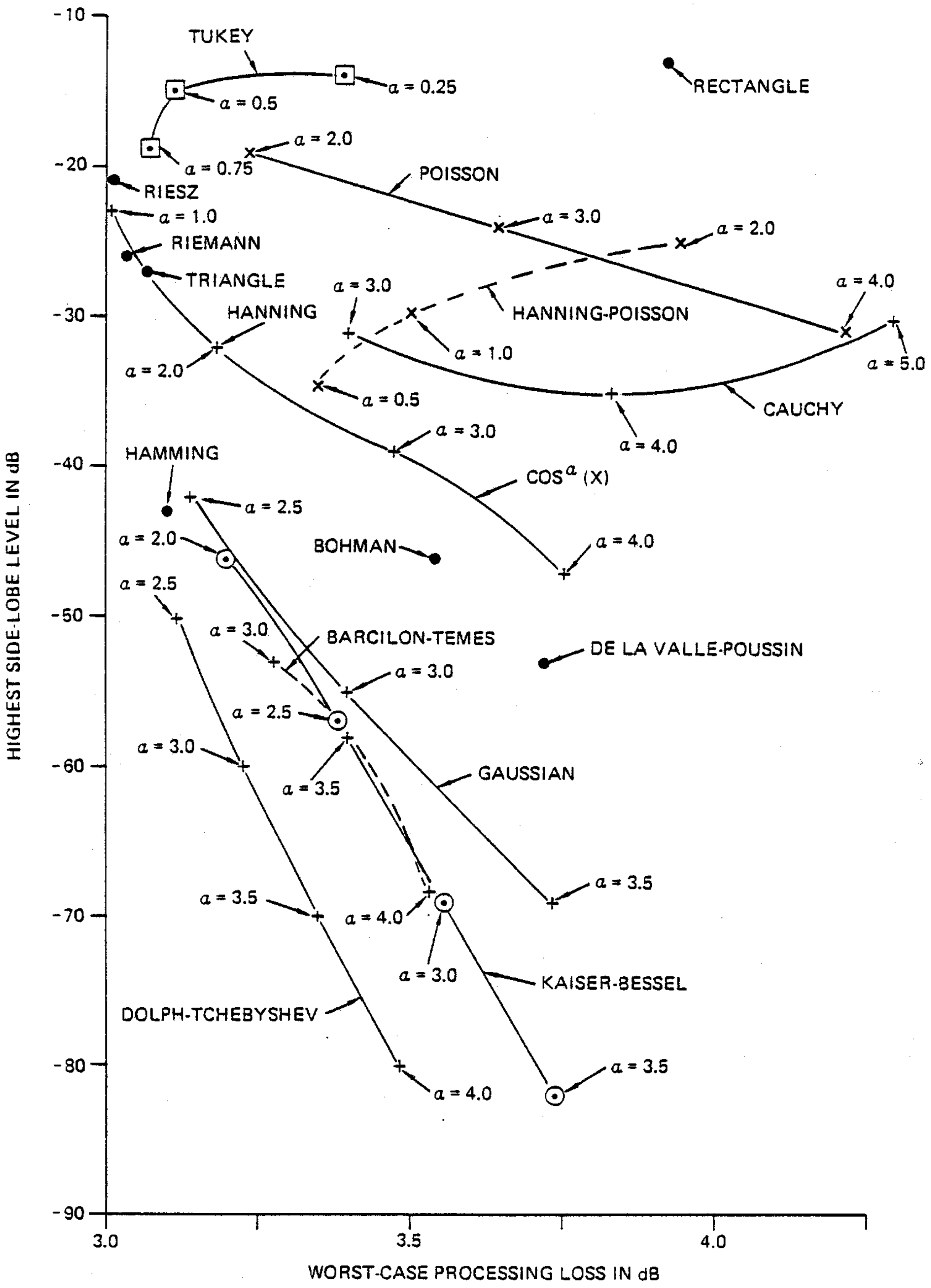


Figure 10. Comparison of Windows for Side-Lobe Level and Worst-Case Processing Loss

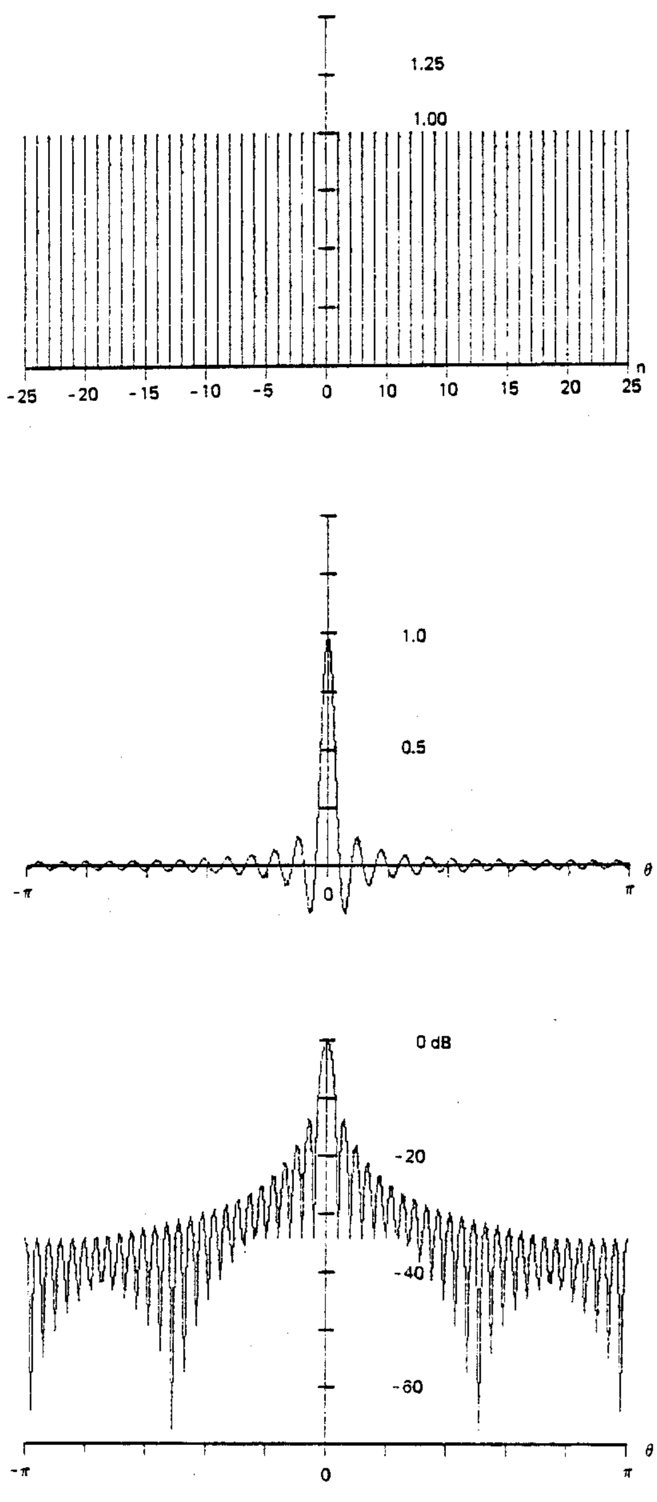


Figure 11. Rectangle Window, Fourier Transform, and Log-Magnitude of Transform

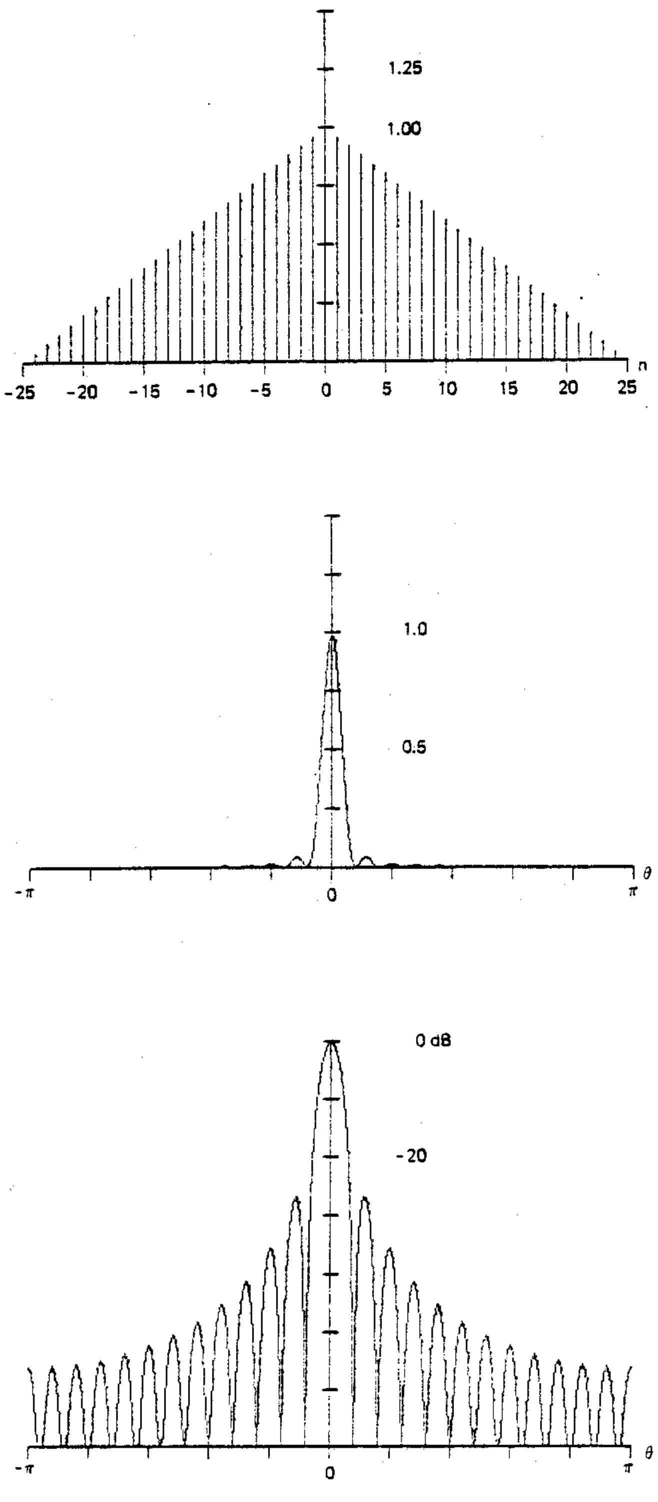


Figure 12. Triangle Window, Fourier Transform, and Log-Magnitude of Transform

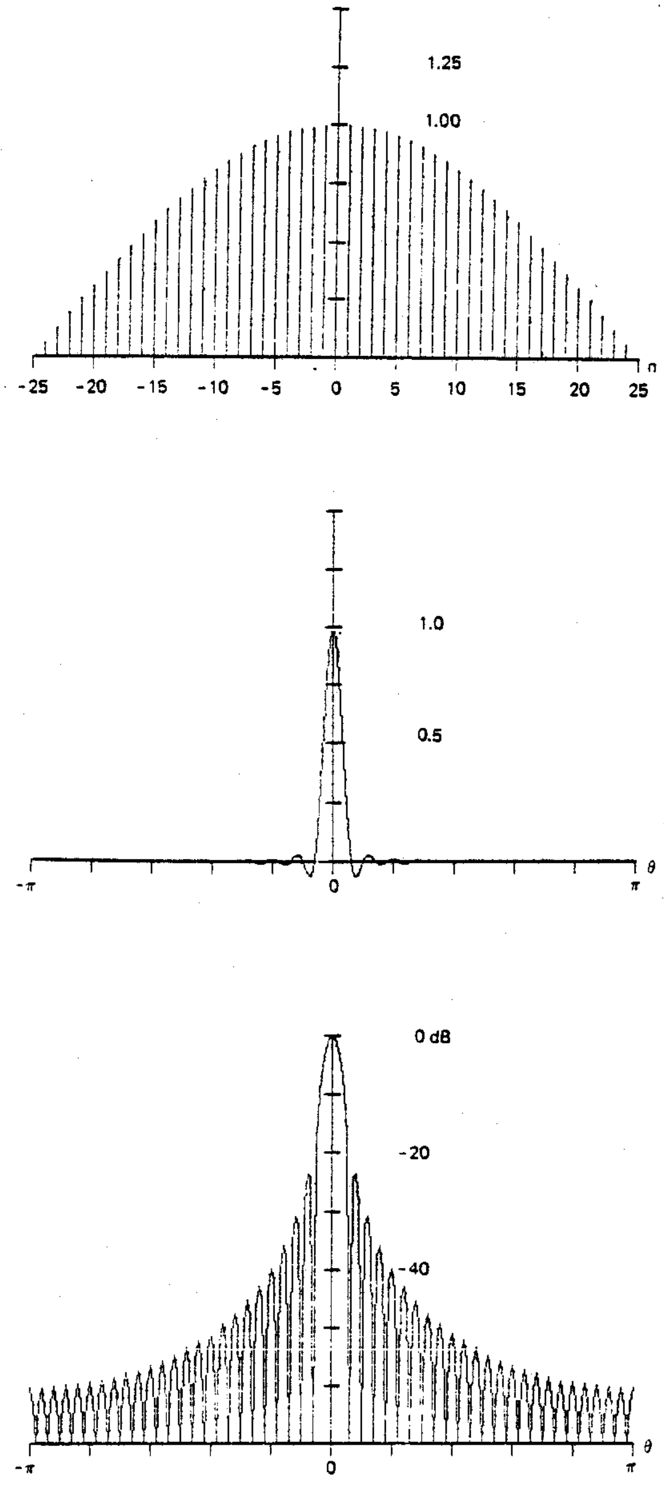


Figure 14. Cos (n π/N) Window, Fourier Transform, and Log-Magnitude of Transform

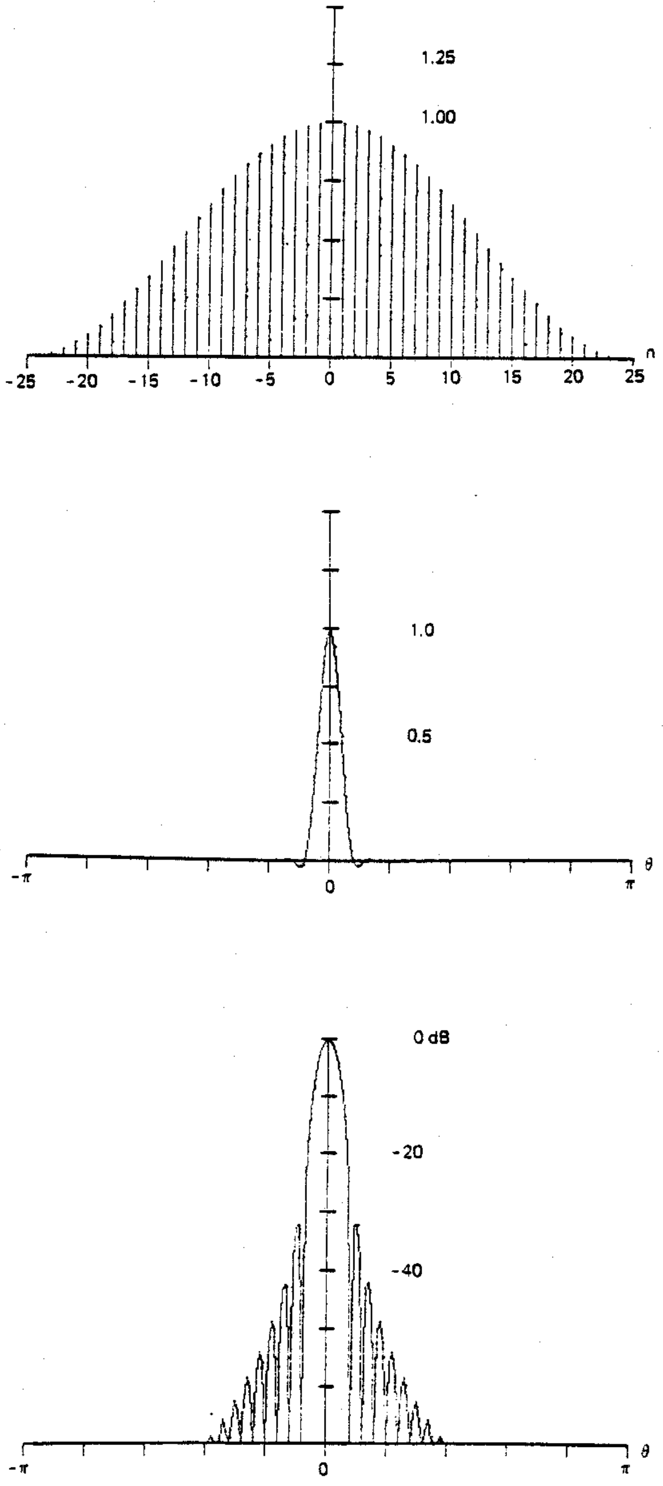


Figure 15. Cos^2 (n π/N) Window, Fourier Transform, and Log-Magnitude of Transform

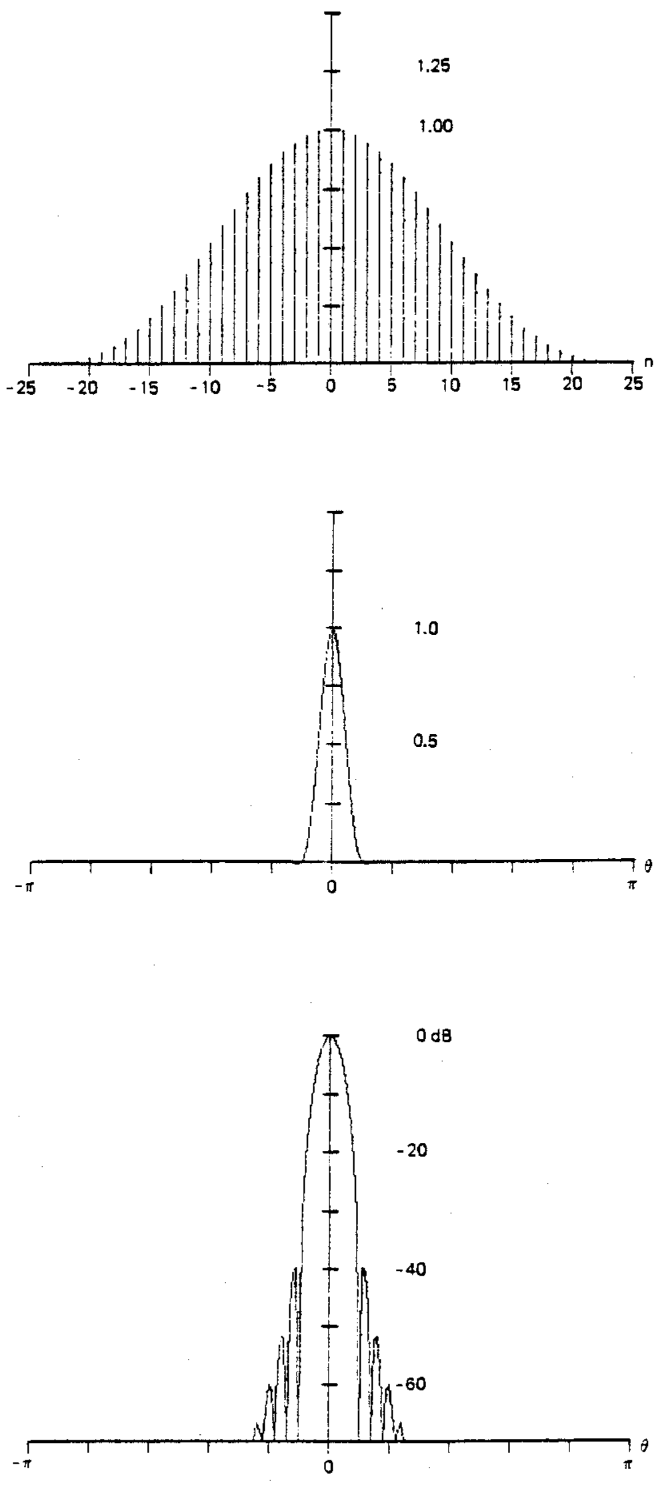


Figure 16. *$\cos^3$ ($n\ \pi/N$) Window, Fourier Transform, and Log-Magnitude of Transform*

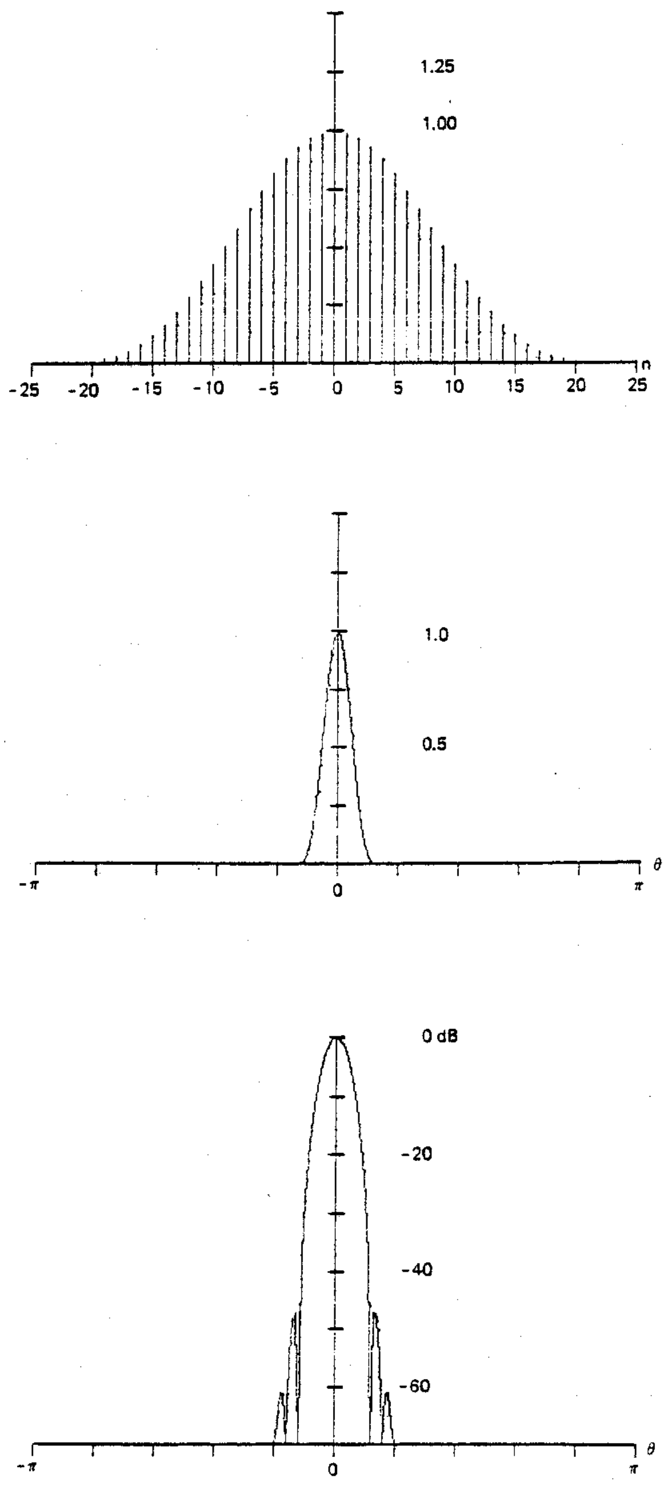


Figure 17. *$\cos^4$ ($n\ \pi/N$) Window, Fourier Transform, and Log-Magnitude of Transform*

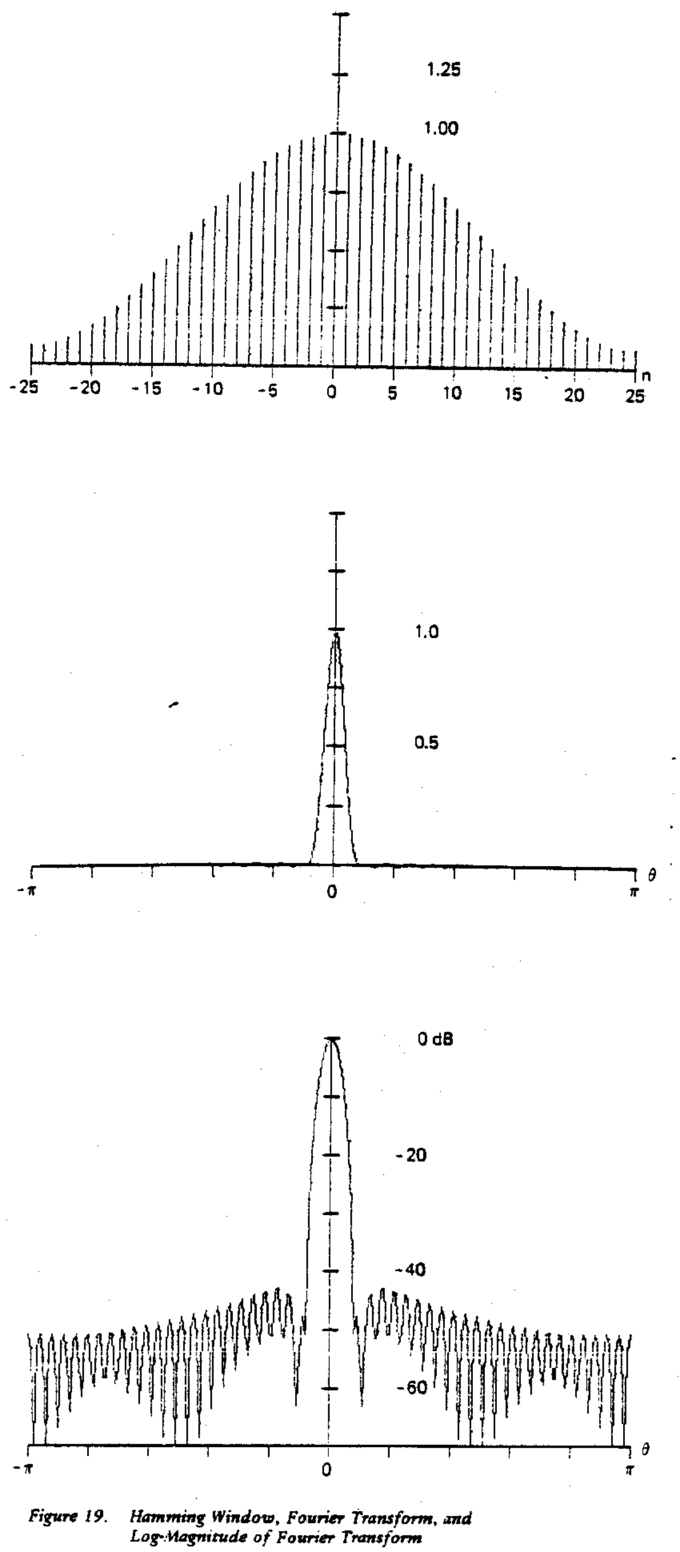


Figure 19. Hamming Window, Fourier Transform, and Log-Magnitude of Fourier Transform

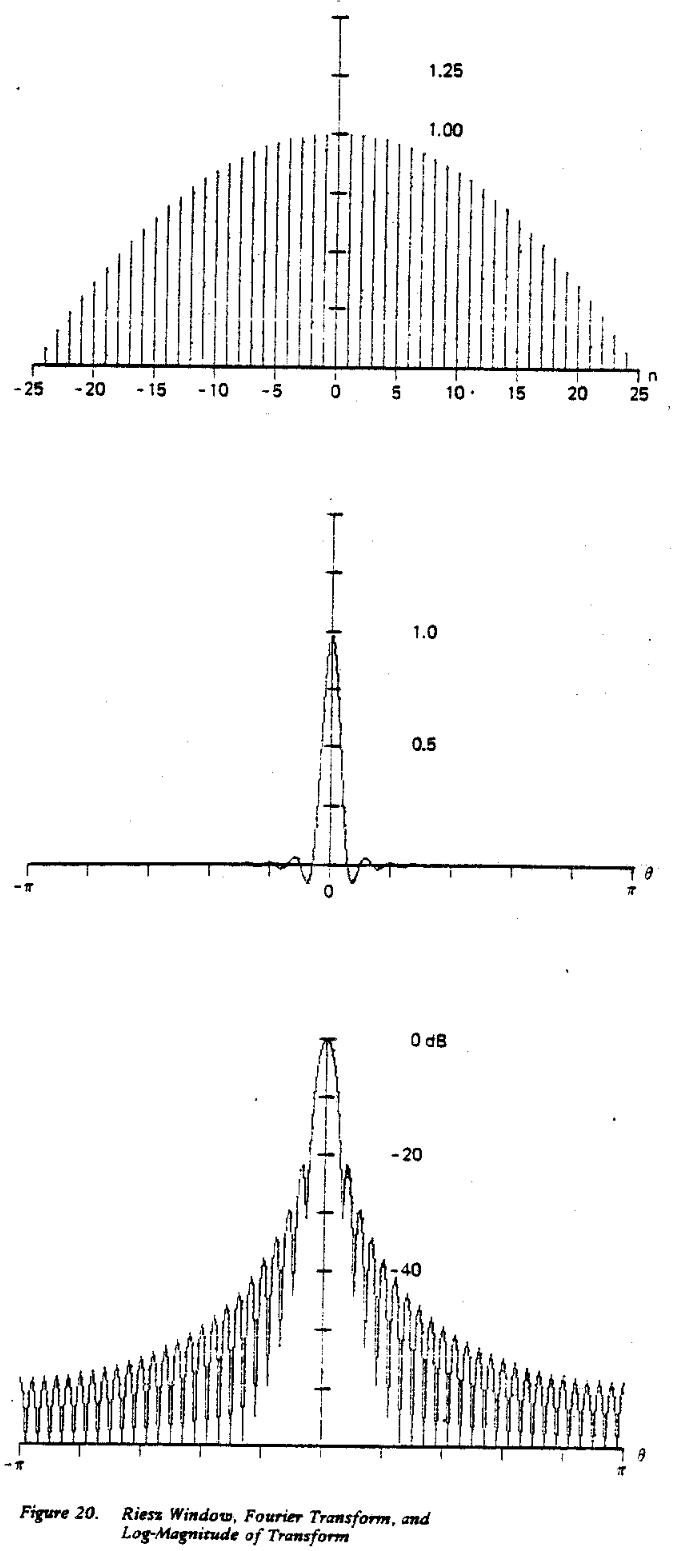


Figure 20. Riesz Window, Fourier Transform, and Log-Magnitude of Transform

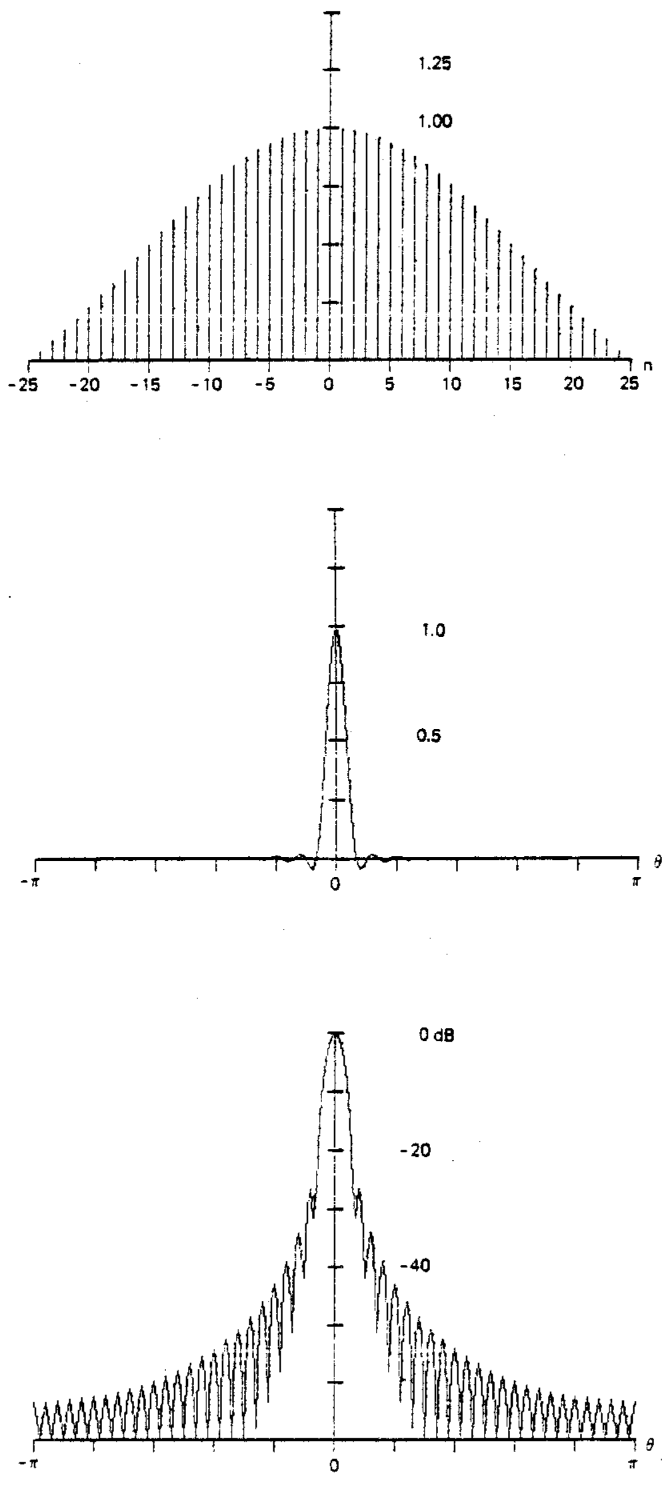


Figure 21. Riemann Window, Fourier Transform, and Log-Magnitude of Transform

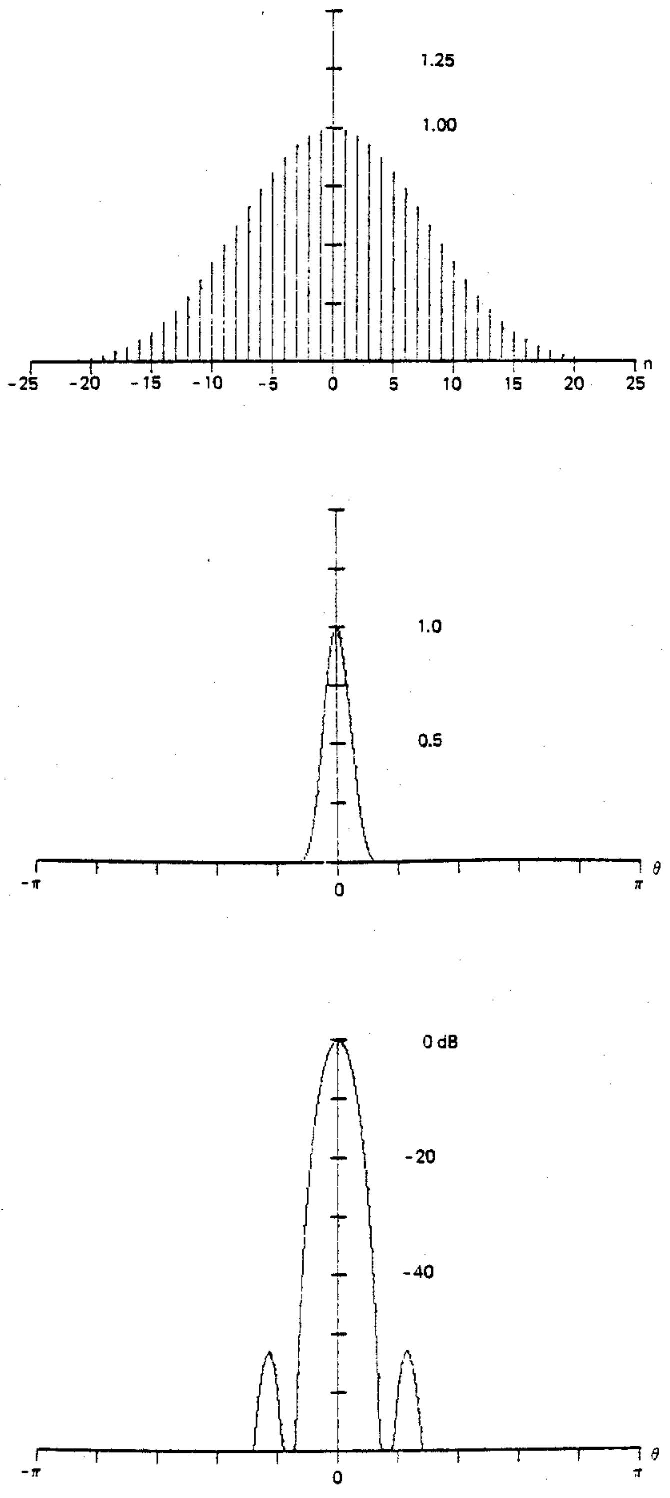


Figure 22. de la Valle'-Poussin Window, Fourier Transform and Log-Magnitude of Transform

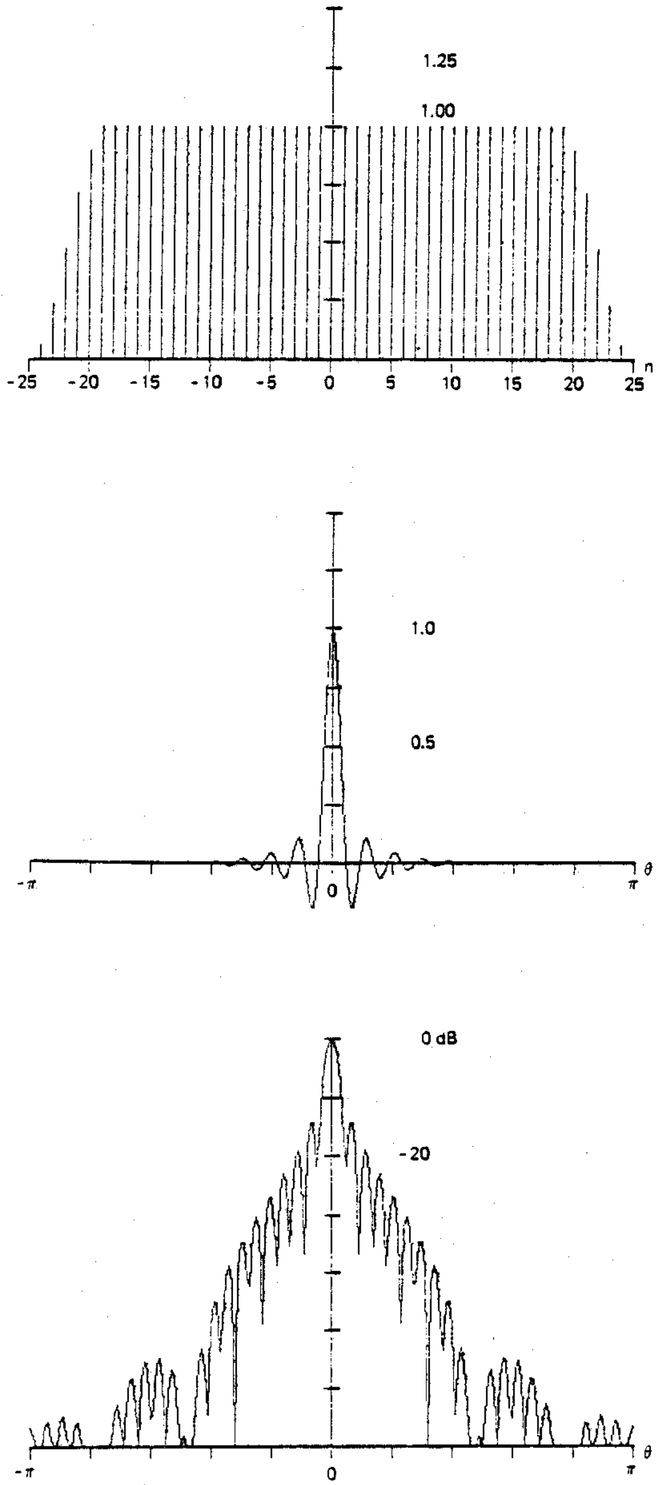


Figure 23. *25% Cosine Taper (Tukey) Window, Fourier Transform and Log-Magnitude of Transform*

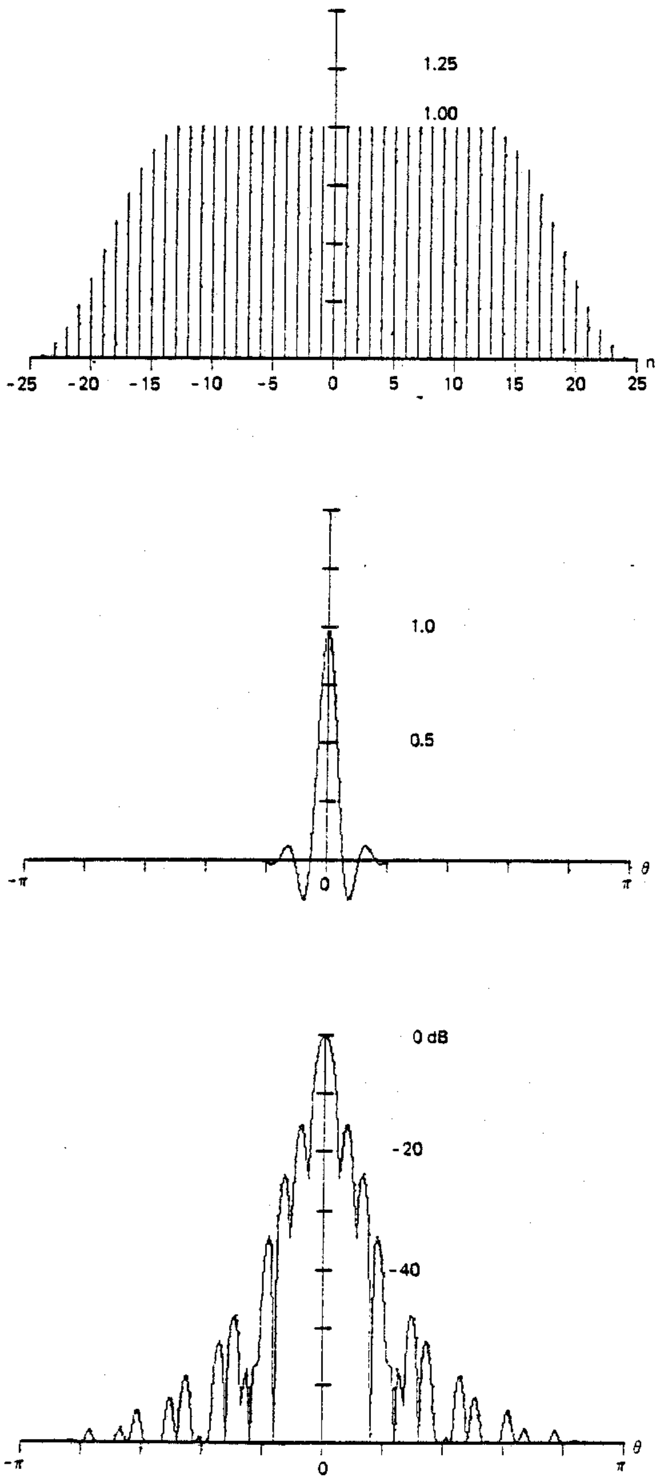


Figure 24. *50% Cosine Taper (Tukey) Window, Fourier Transform and Log-Magnitude of Transform*

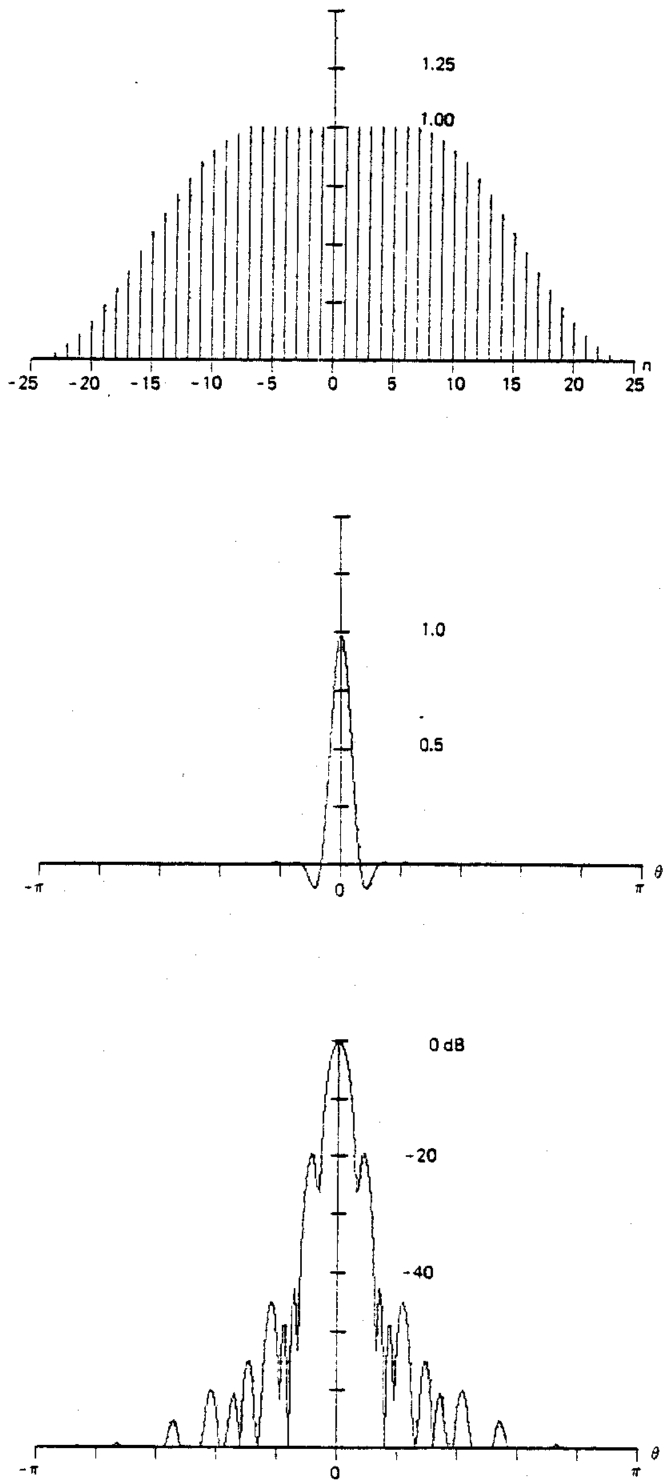


Figure 25. *75% Cosine Taper (Tukey) Window, Fourier Transform and Log-Magnitude of Transform*

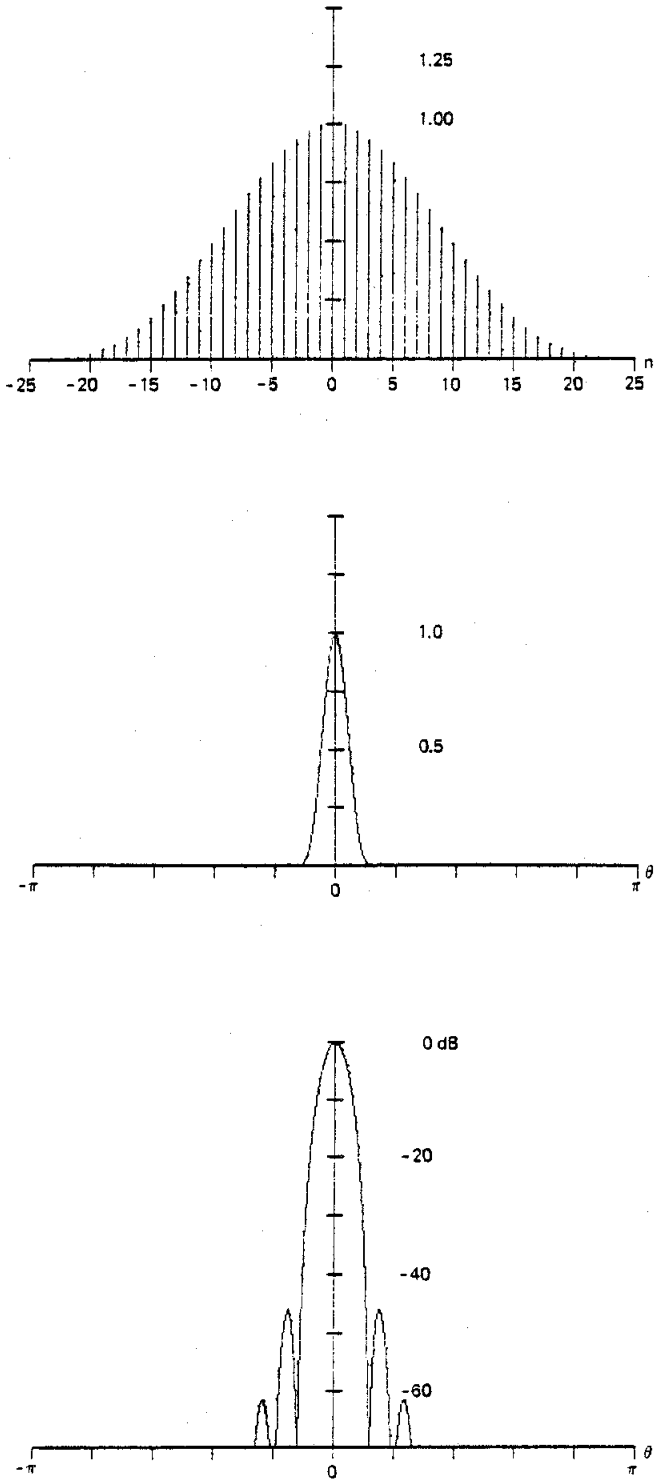


Figure 26. *Bobman Window, Fourier Transform, and Log-Magnitude of Transform*

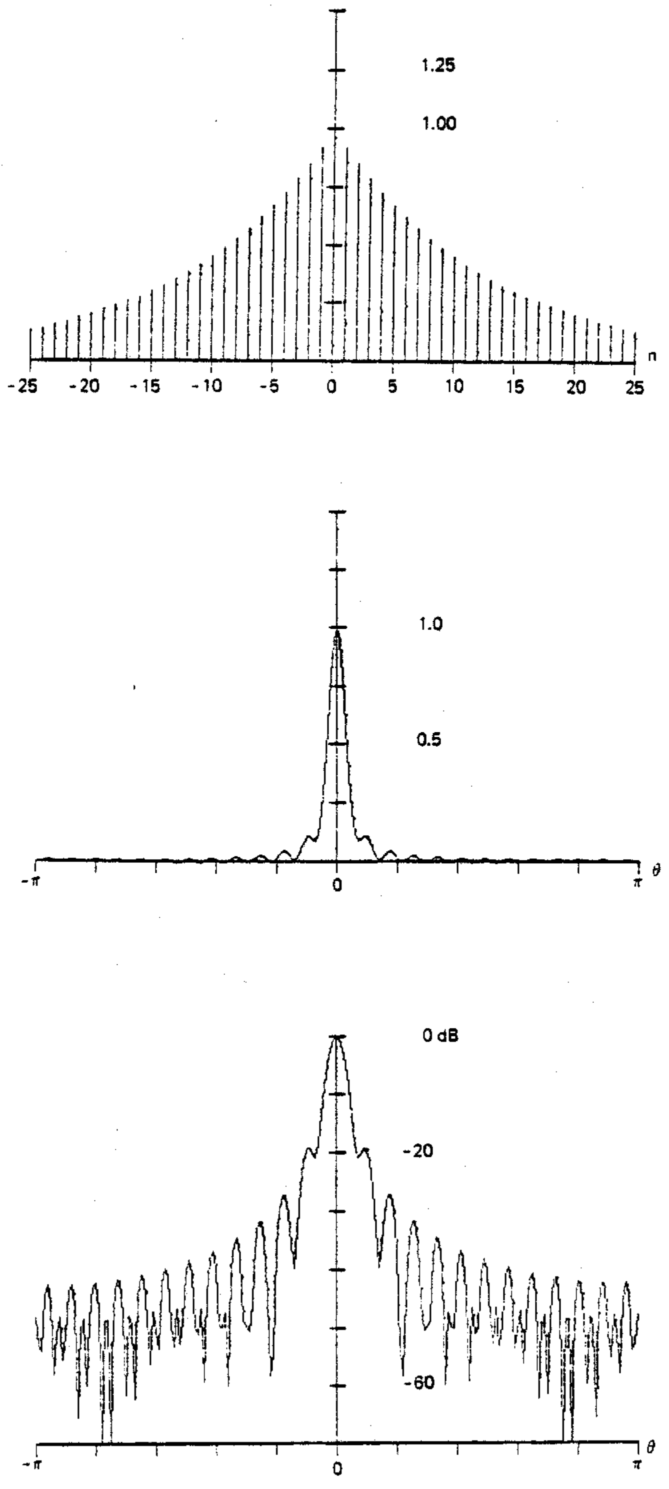


Figure 27. Poisson Window, Fourier Transform, and Log-Magnitude of Transform. (a = 2.0)

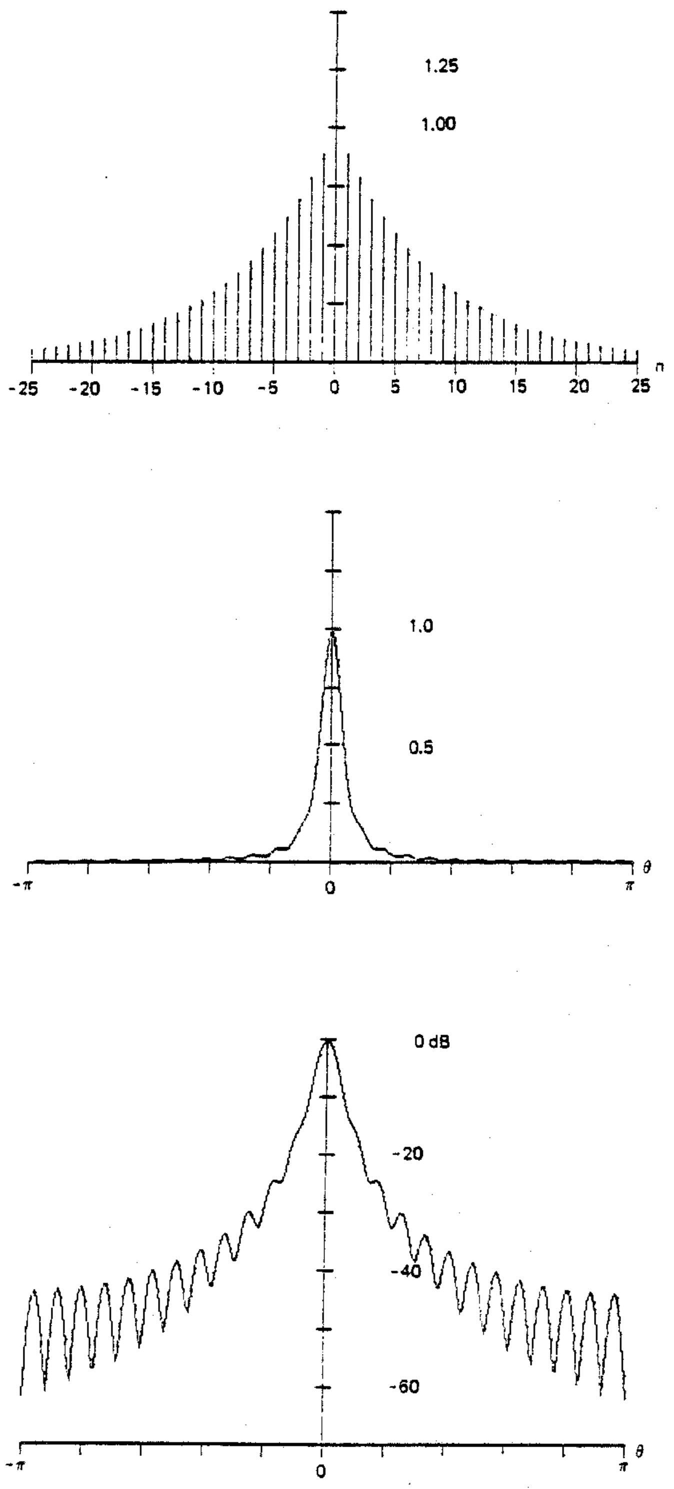


Figure 28. Poisson Window, Fourier Transform, and Log-Magnitude of Transform. (a = 3.0)

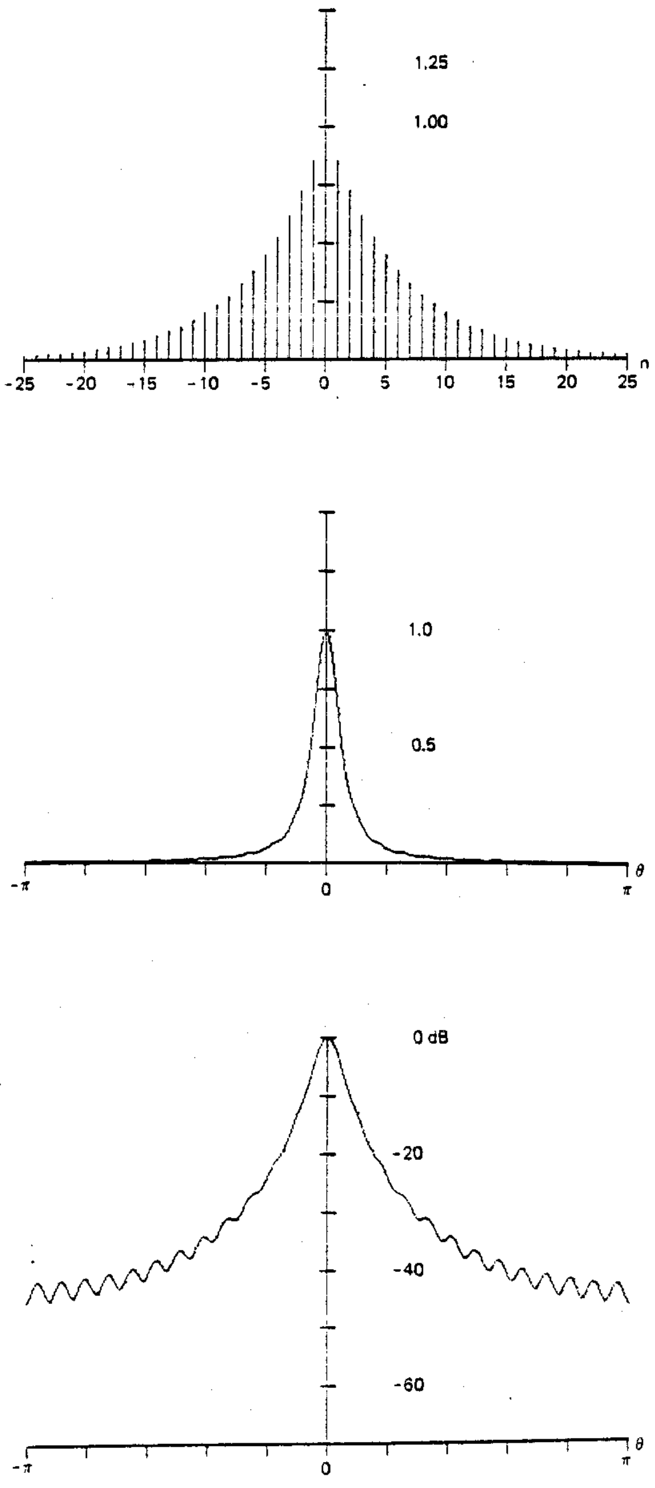


Figure 29. *Poisson Window, Fourier Transform, and Log-Magnitude of Transform. (α = 4.0)*

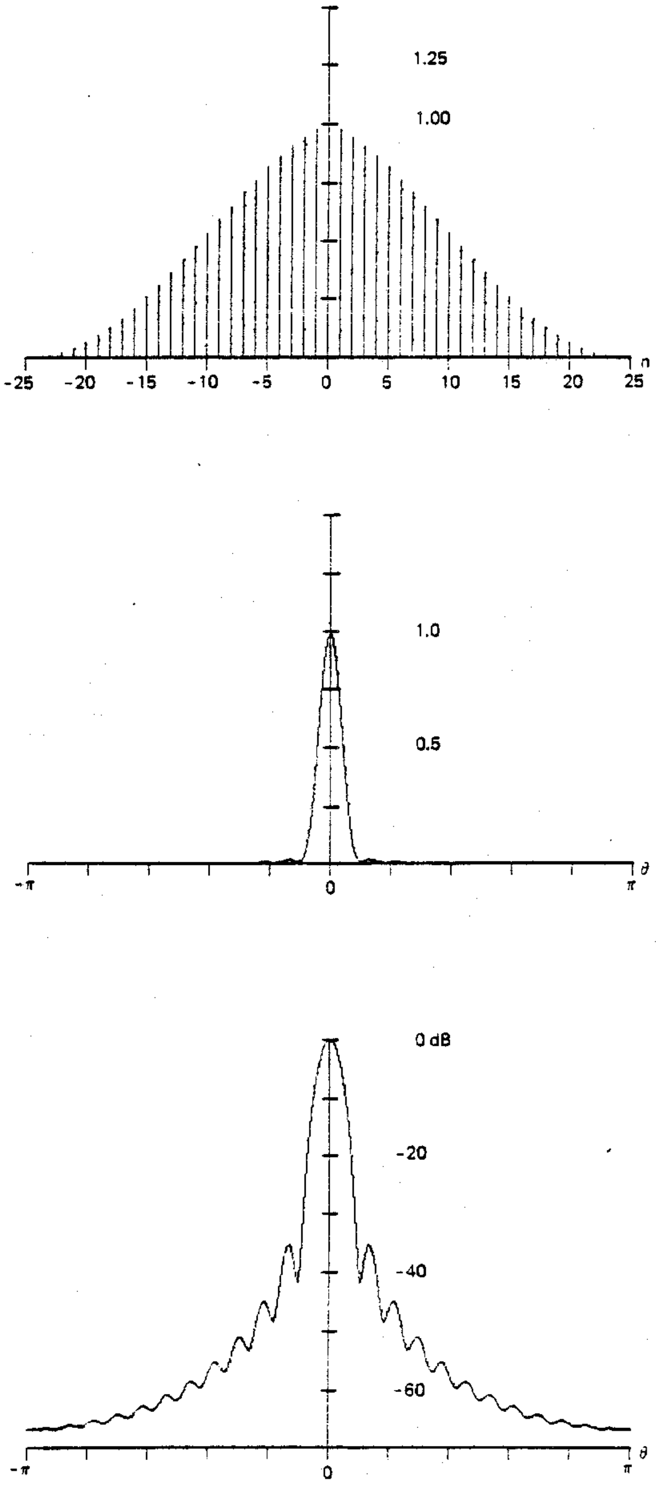


Figure 30. *Hanning-Poisson Window, Fourier Transform, and Log-Magnitude of Transform. (α = 0.5)*

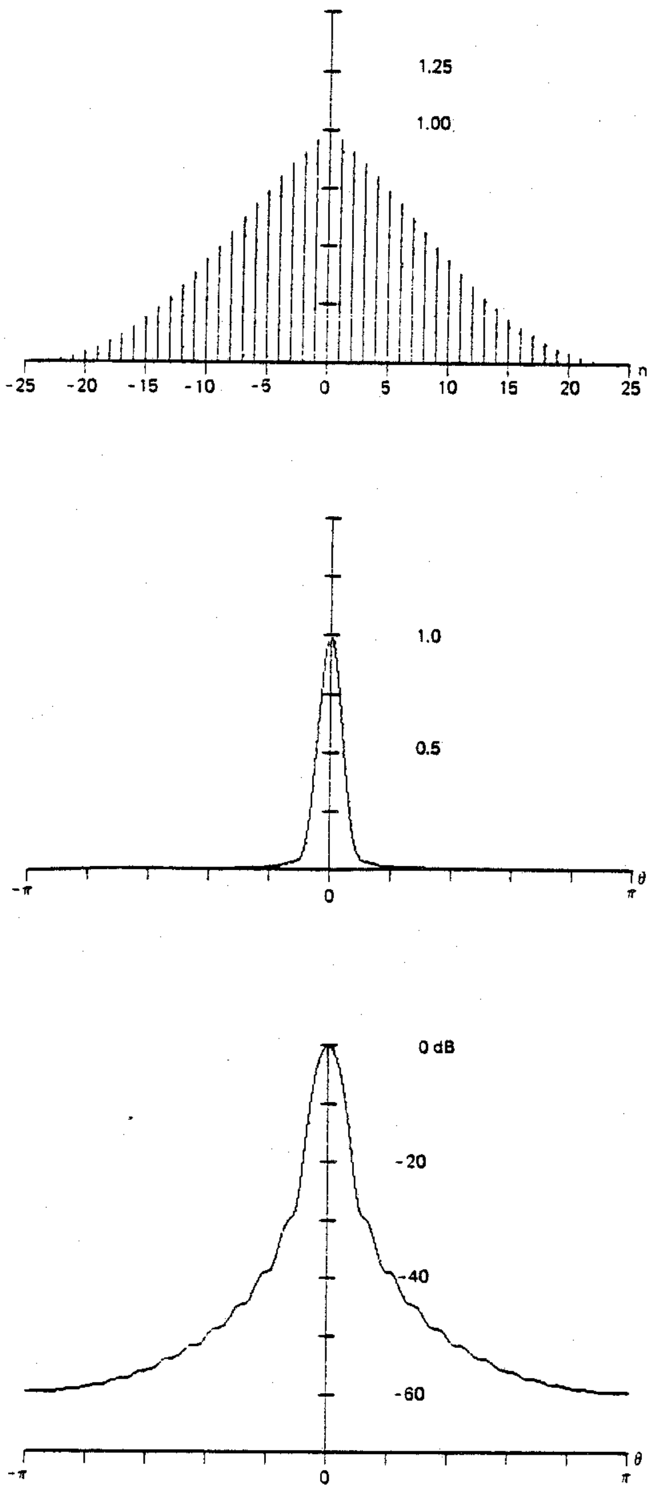


Figure 31. Hanning-Poisson Window, Fourier Transform, and Log-Magnitude of Transform. (α = 1.0)

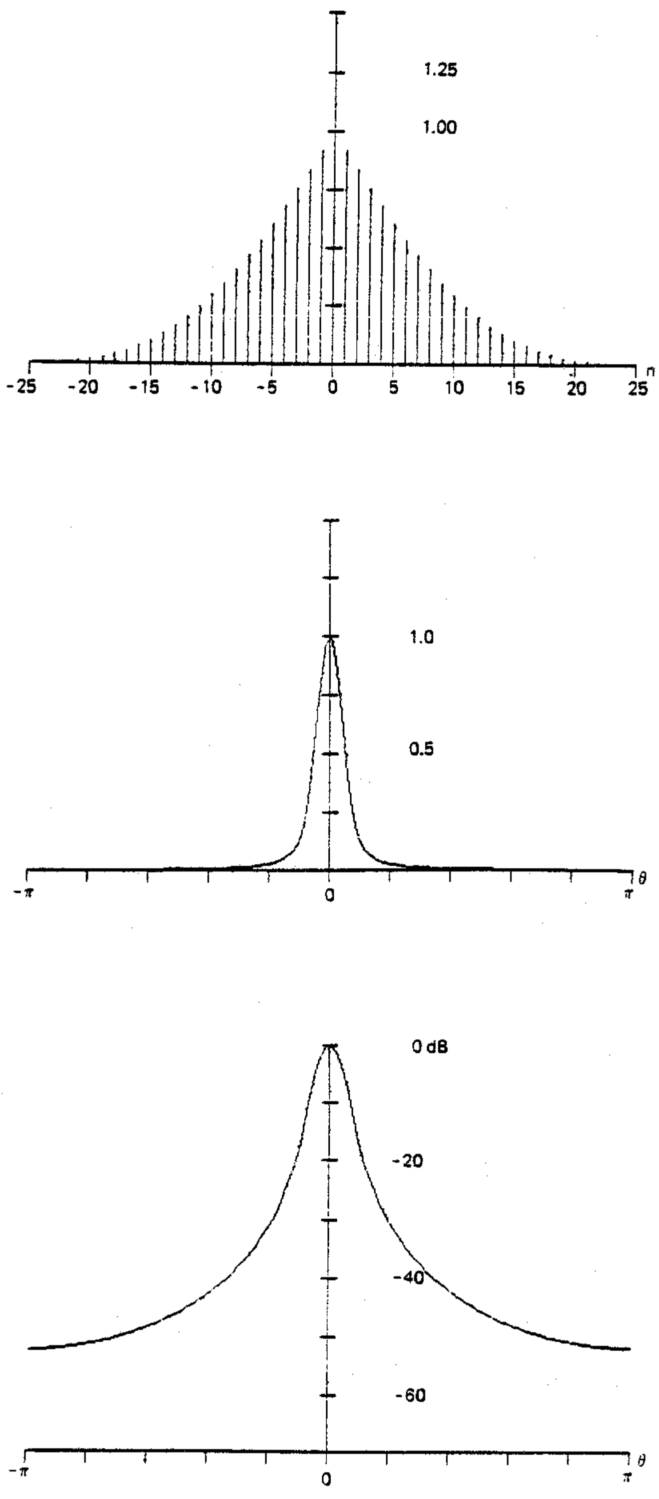


Figure 32. Hanning-Poisson Window, Fourier Transform, and Log-Magnitude of Transform. (α = 2.0)

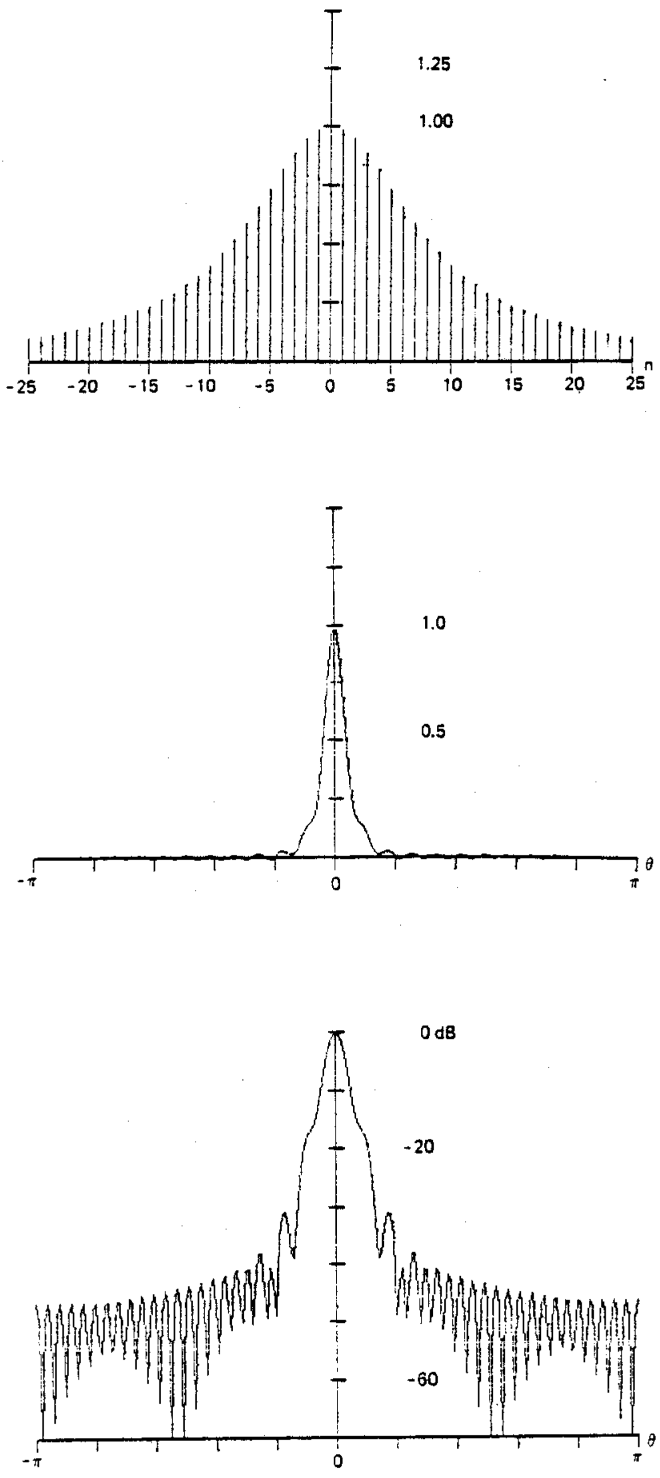


Figure 33. Caucby Window, Fourier Transform, and Log-Magnitude of Transform. (α = 3.0)

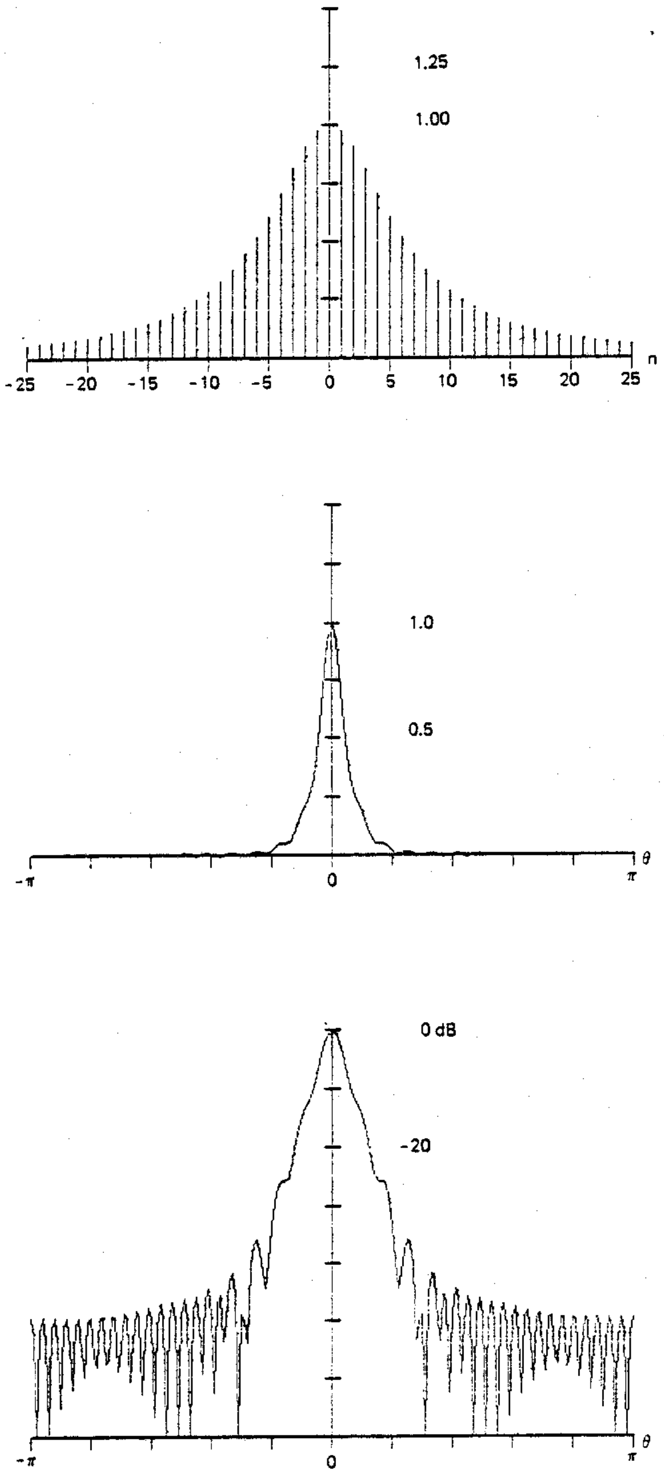


Figure 34. Caucby Window, Fourier Transform, and Log-Magnitude of Transform. (α = 4.0)

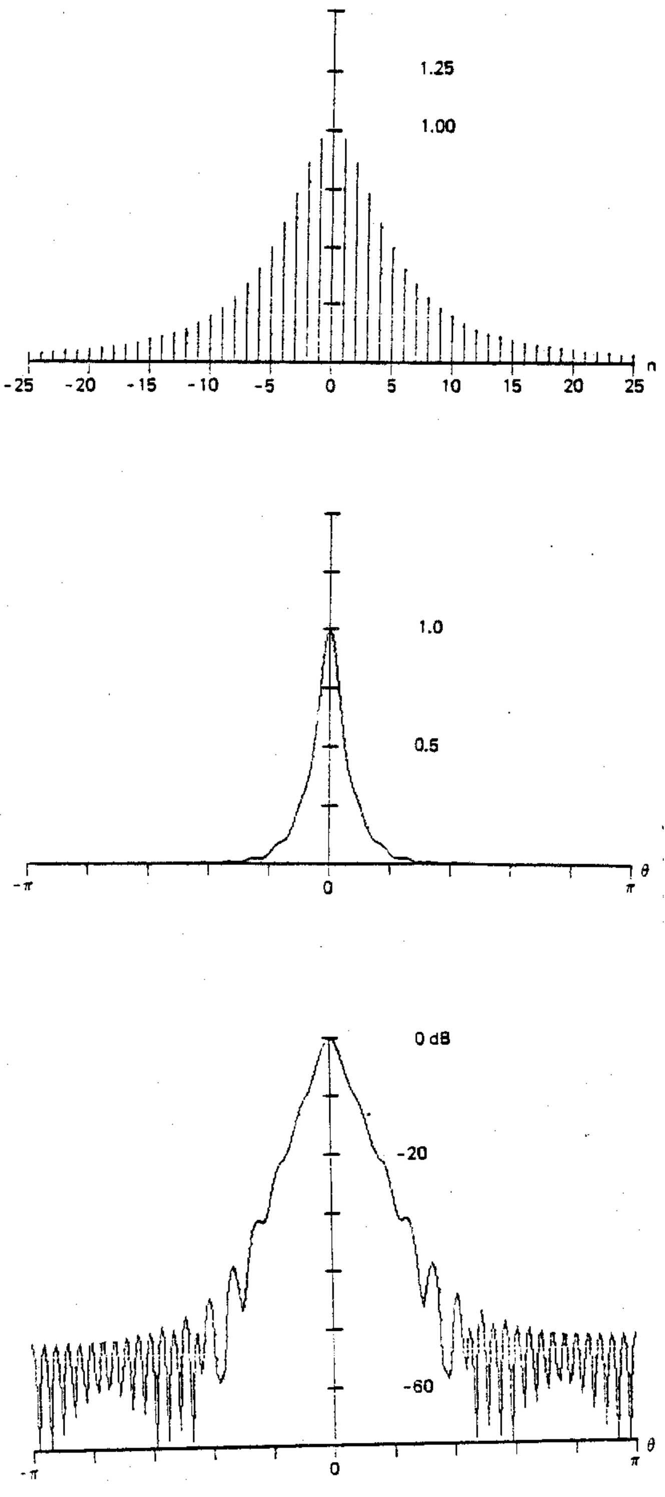


Figure 35. Cauchy Window, Fourier Transform, and Log-Magnitude of Transform. (α = 5.0)

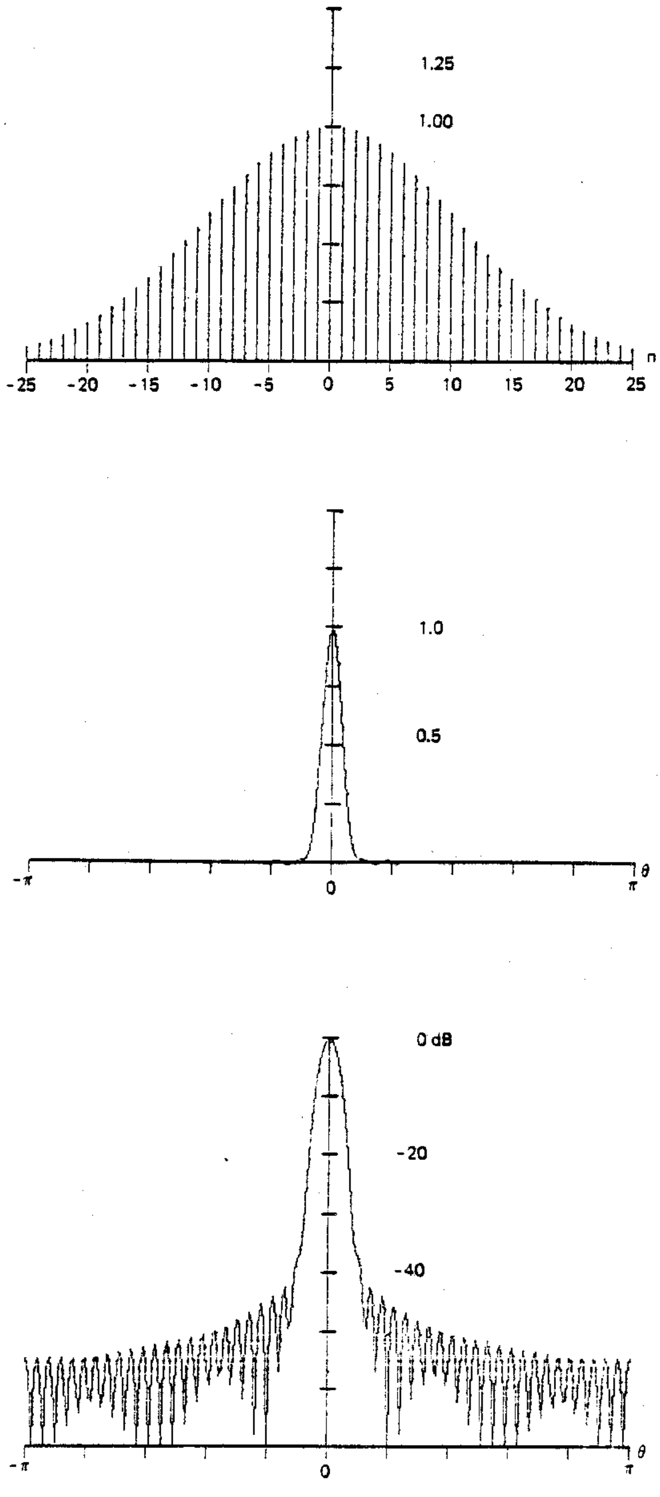


Figure 36. Gaussian Window, Fourier Transform, and Log-Magnitude of Transform. (α = 2.5)

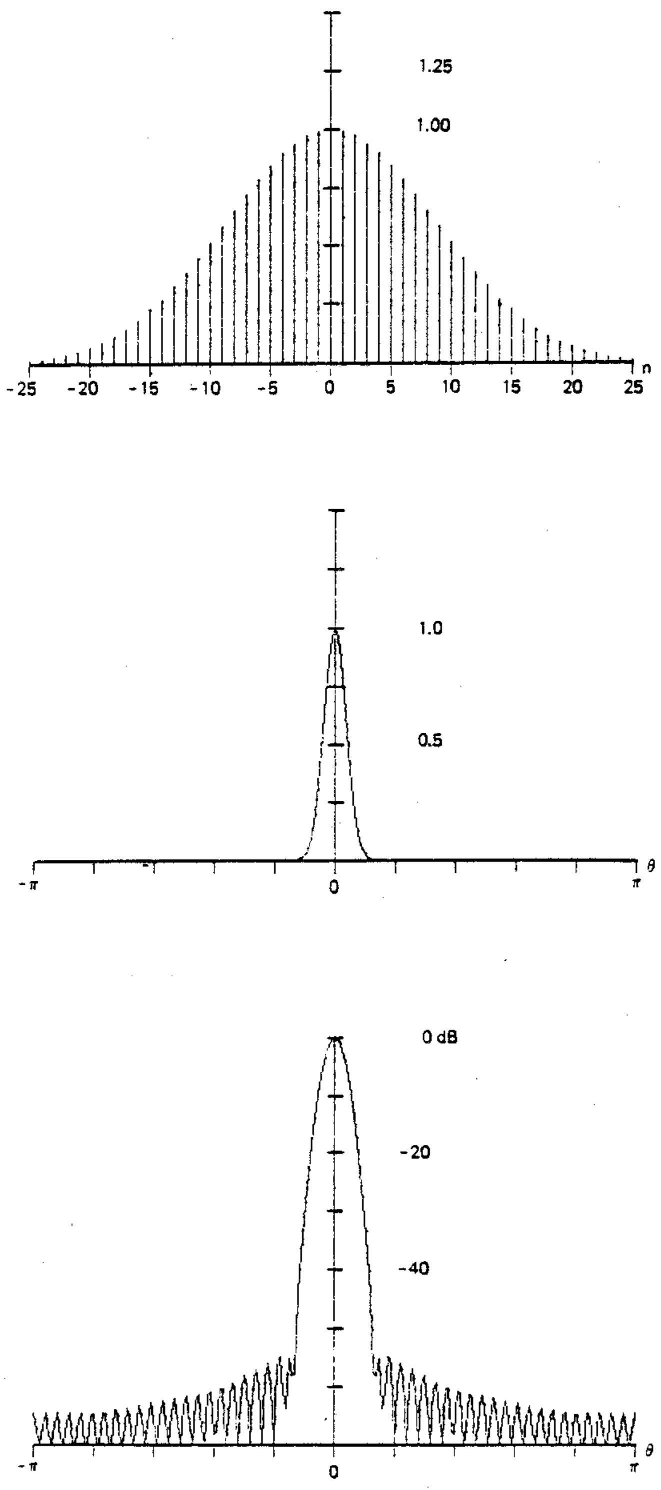


Figure 37. Gaussian Window, Fourier Transform, and Log-Magnitude of Transform. (α = 3.0)

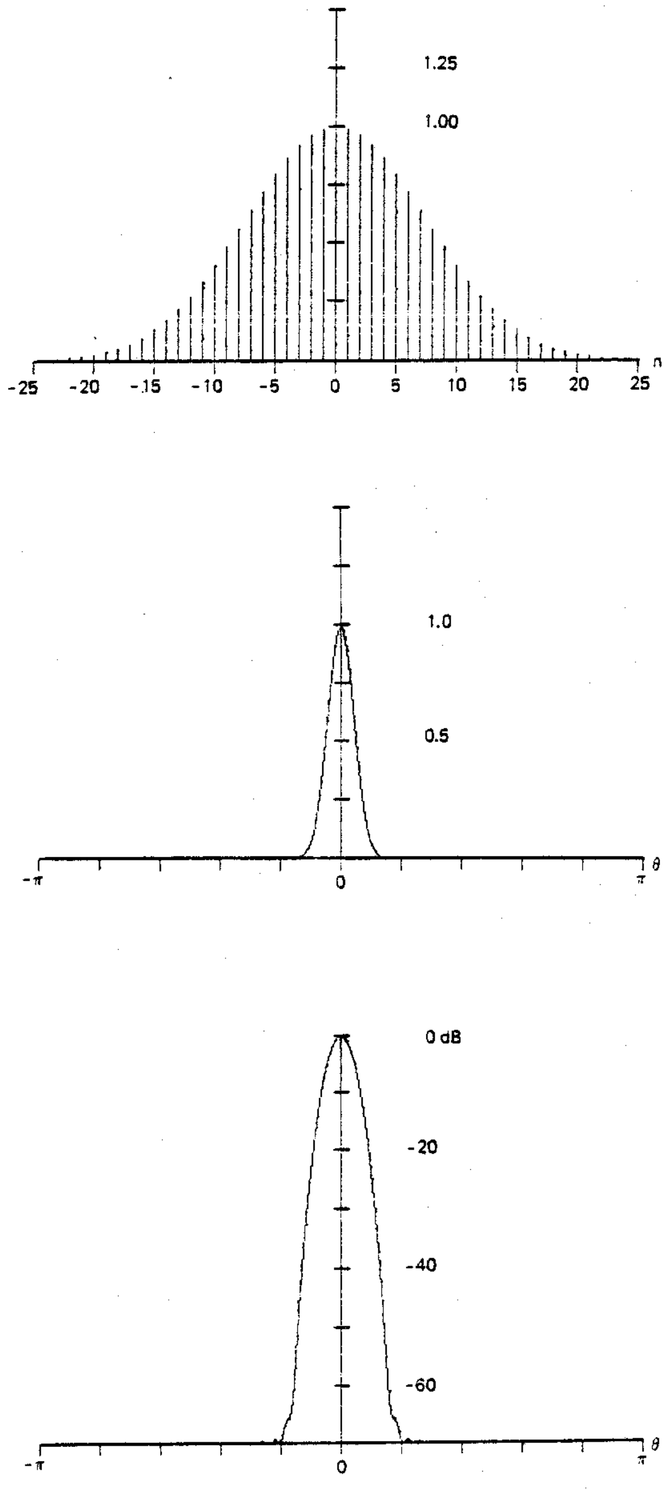


Figure 38. Gaussian Window, Fourier Transform, and Log-Magnitude of Transform. (α = 3.5)

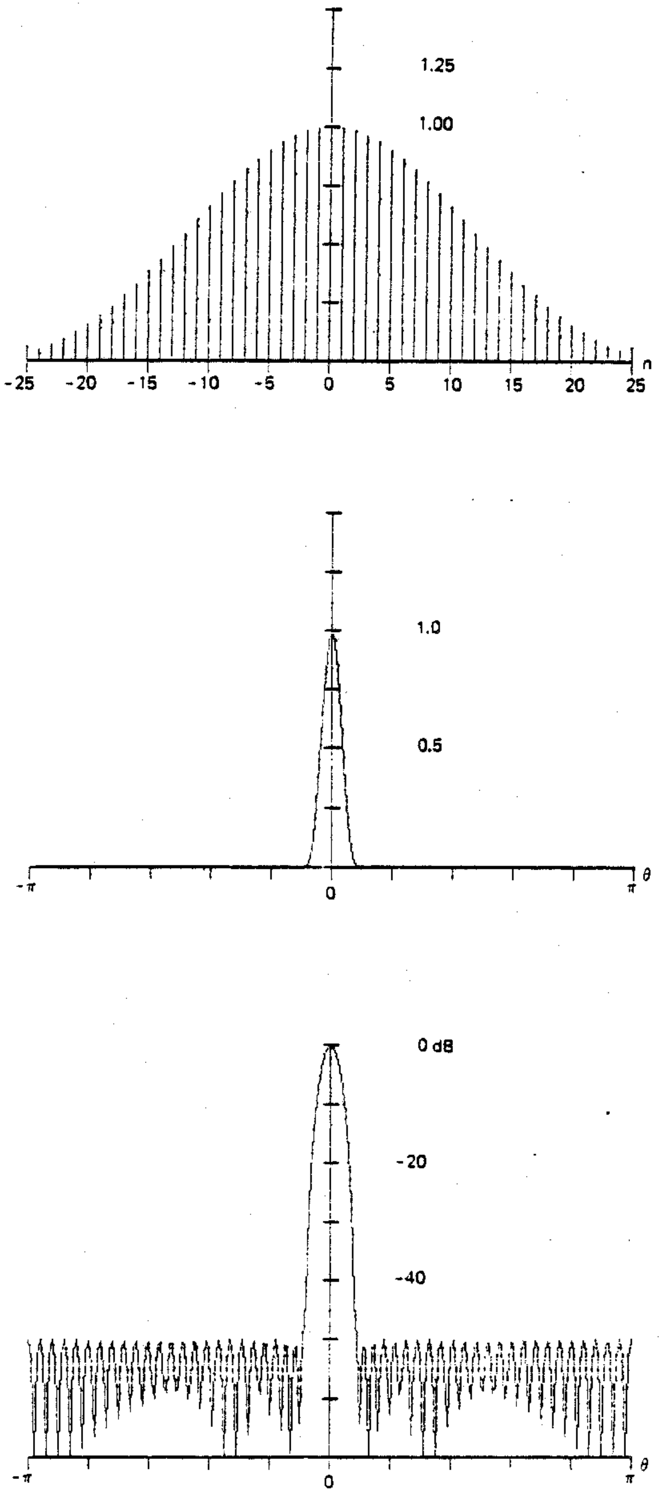


Figure 39. Dolph-Tchebyshev Window, Fourier Transform, and Log-Magnitude of Transform. (α = 2.5)

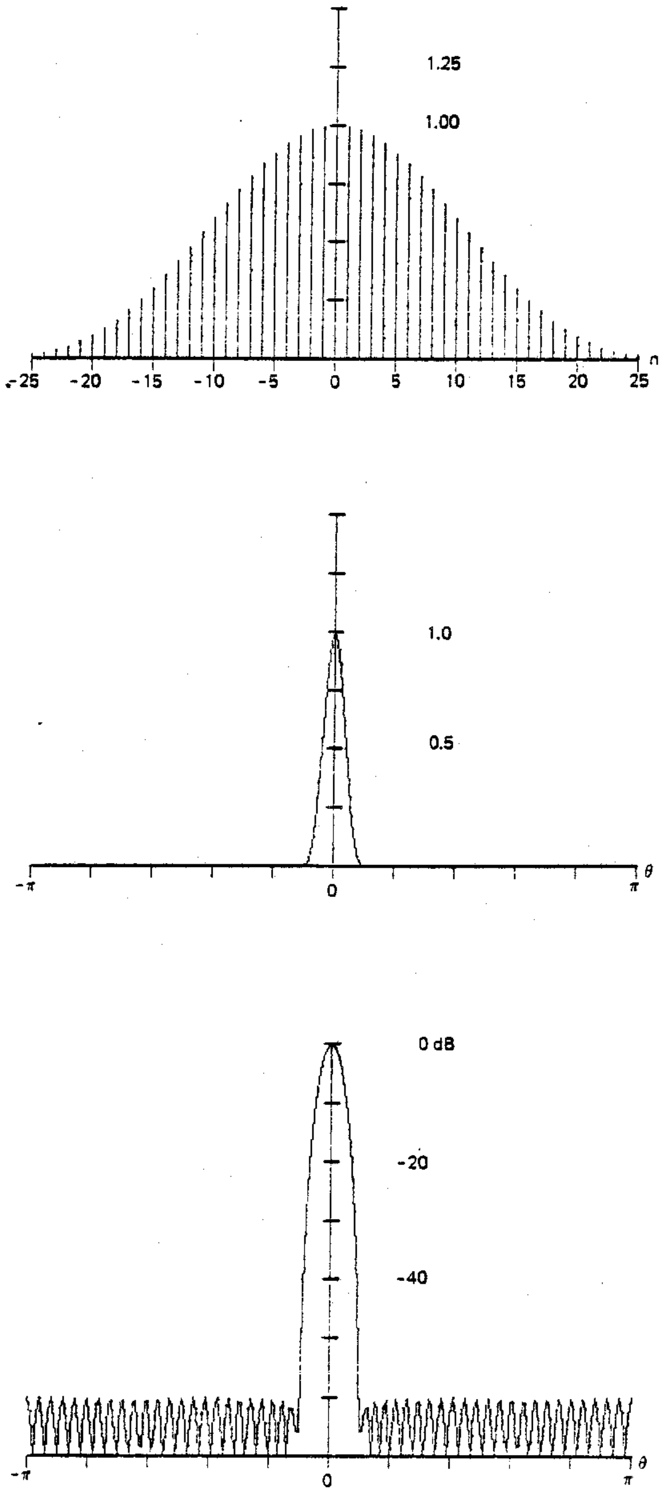


Figure 40. Dolph-Tchebyshev Window, Fourier Transform, and Log-Magnitude of Transform. (α = 3.0)

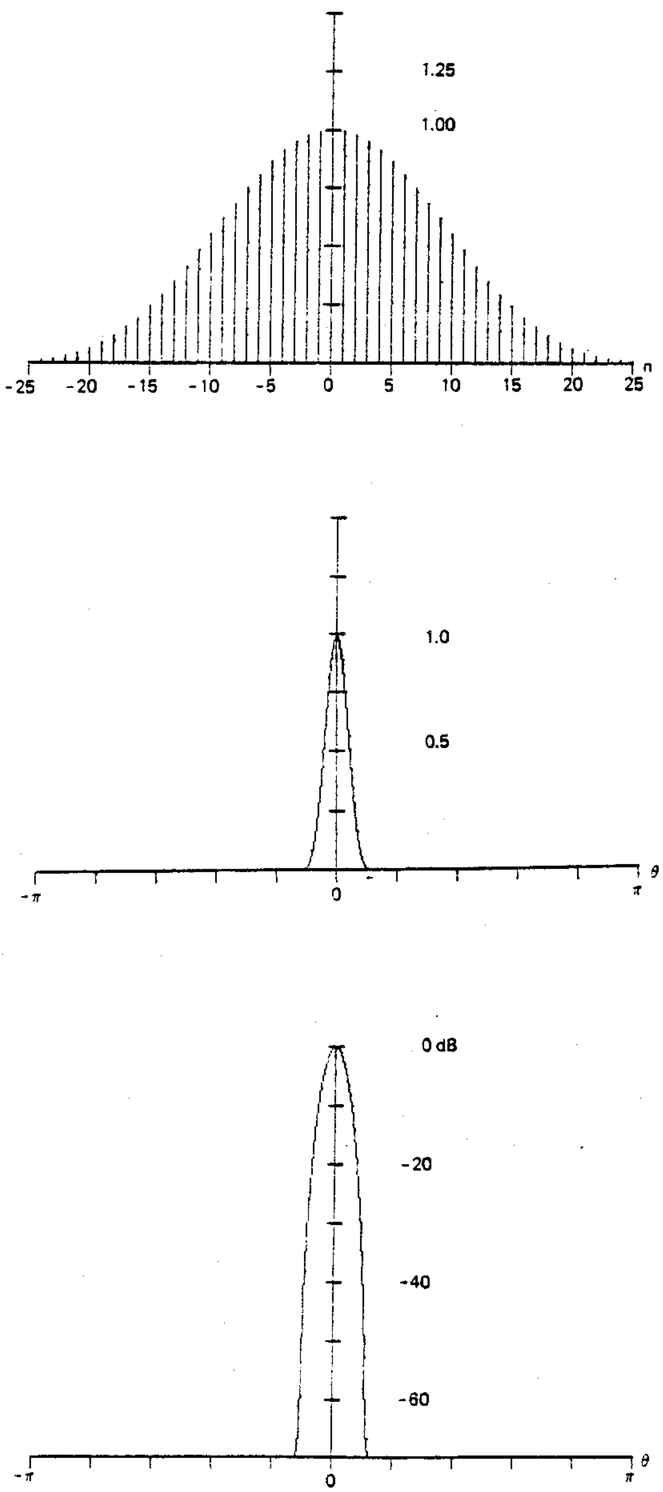


Figure 41. Dolph-Tchebyshev Window, Fourier Transform, and Log-Magnitude of Transform. (α = 3.5)

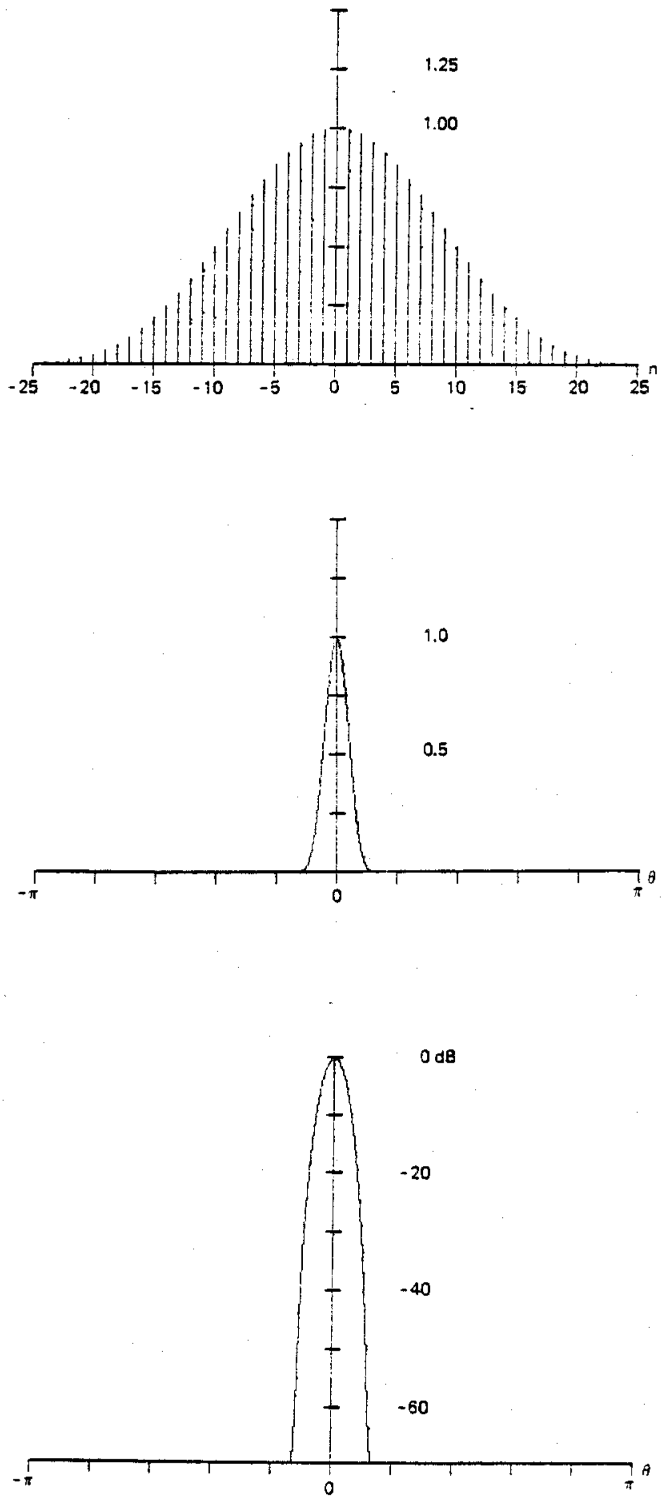


Figure 42. Dolph-Tchebyshev Window, Fourier Transform, and Log-Magnitude of Transform. (α = 4.0)

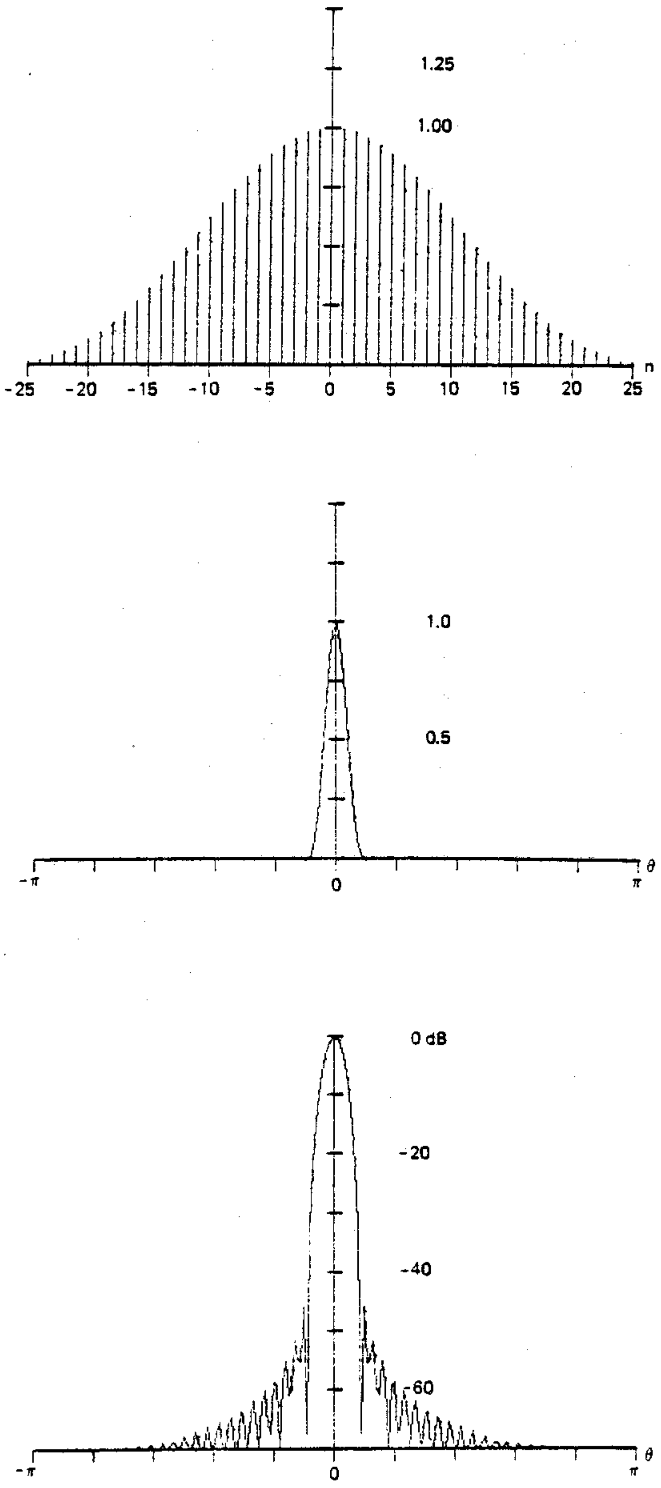


Figure 43. Kaiser-Bessel Window, Fourier Transform, and Log-Magnitude of Transform. (α = 2.0)

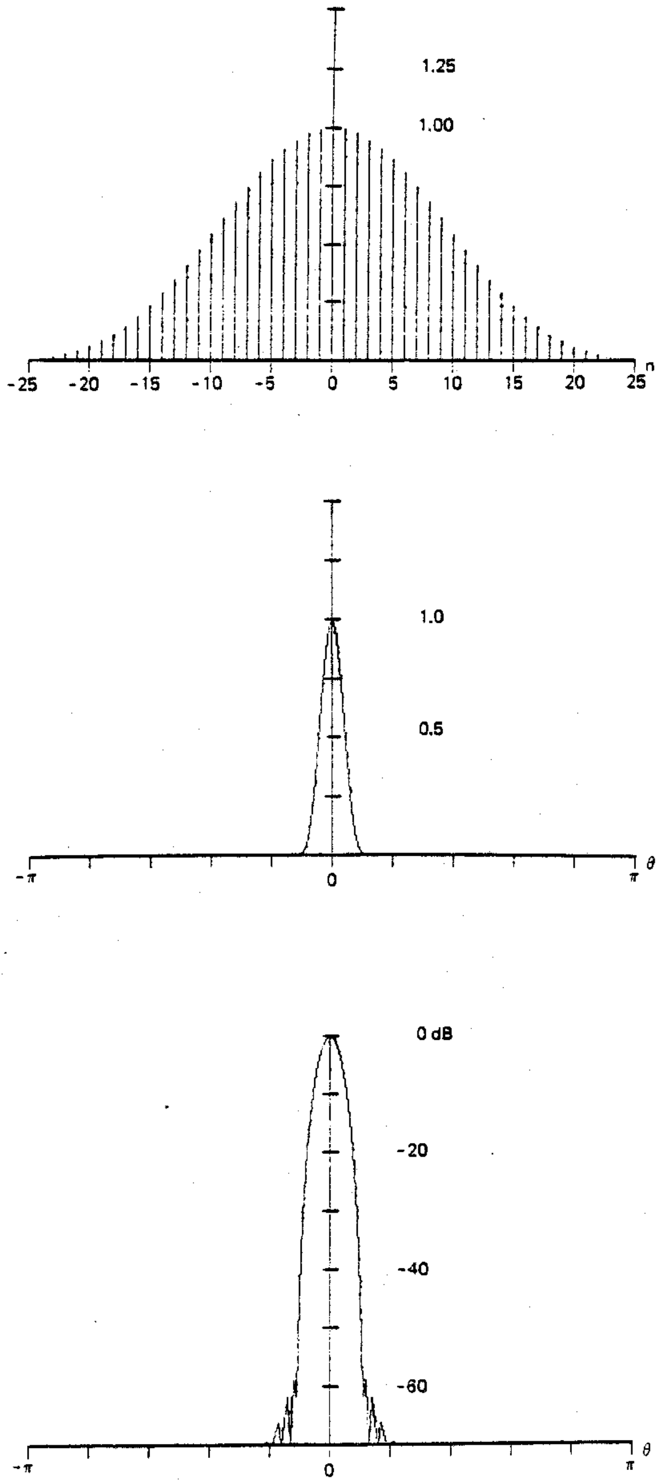


Figure 44. Kaiser-Bessel Window, Fourier Transform and Log-Magnitude of Transform. (α = 2.5)

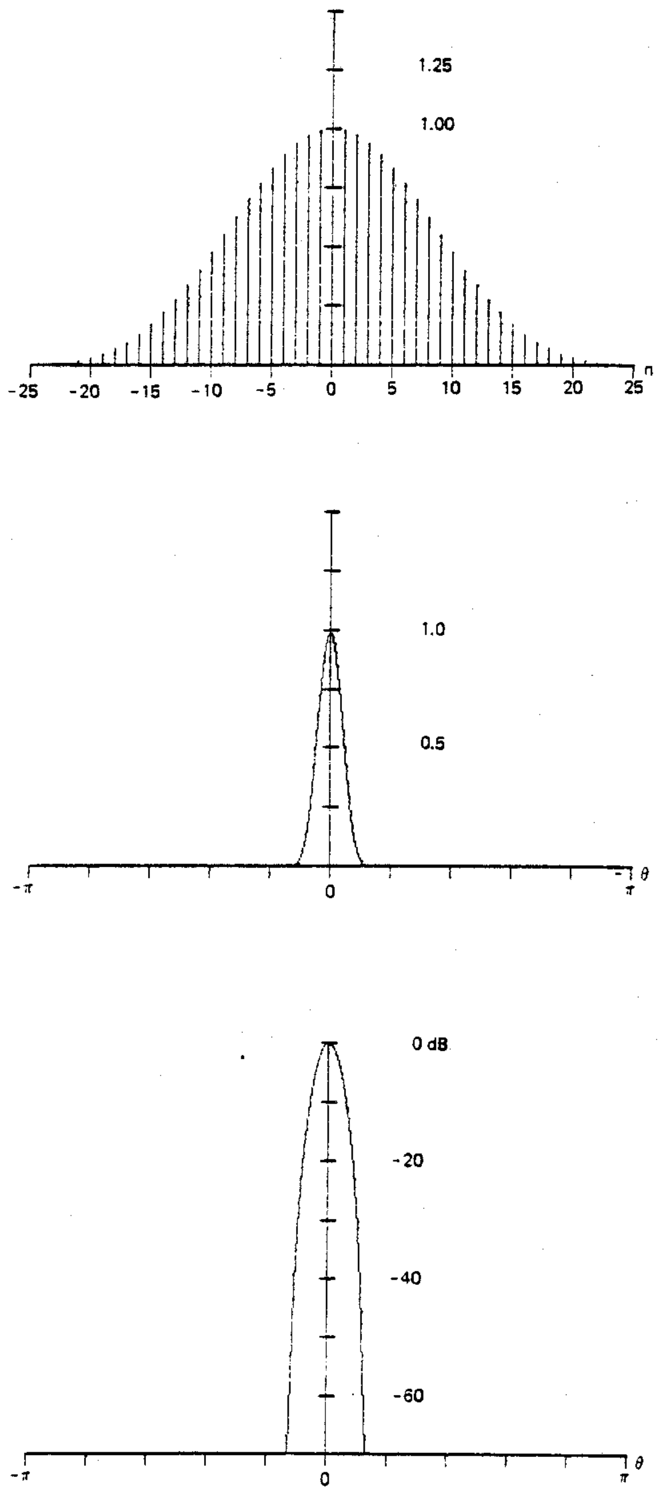


Figure 45. Kaiser-Bessel Window, Fourier Transform, and Log-Magnitude of Transform. (α = 3.0)

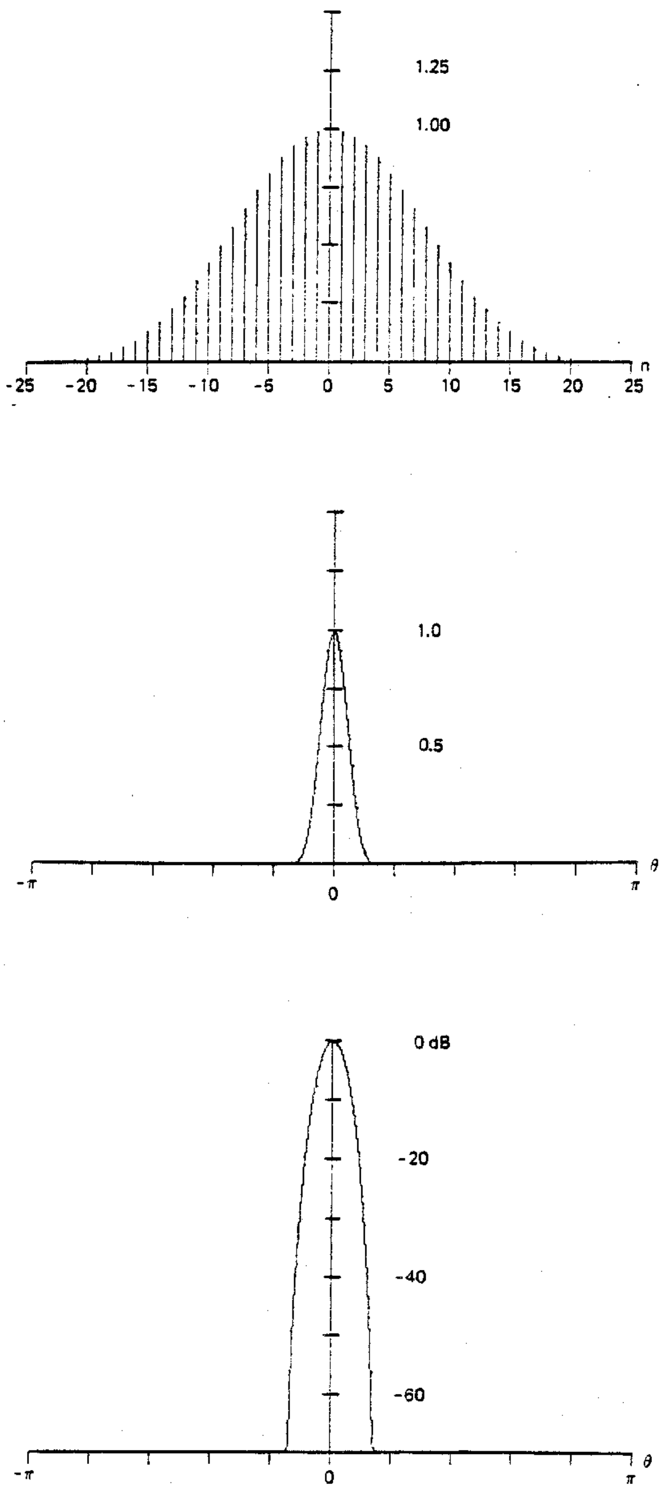


Figure 46. Kaiser-Bessel Window, Fourier Transform, and Log-Magnitude of Transform. (α = 3.5)

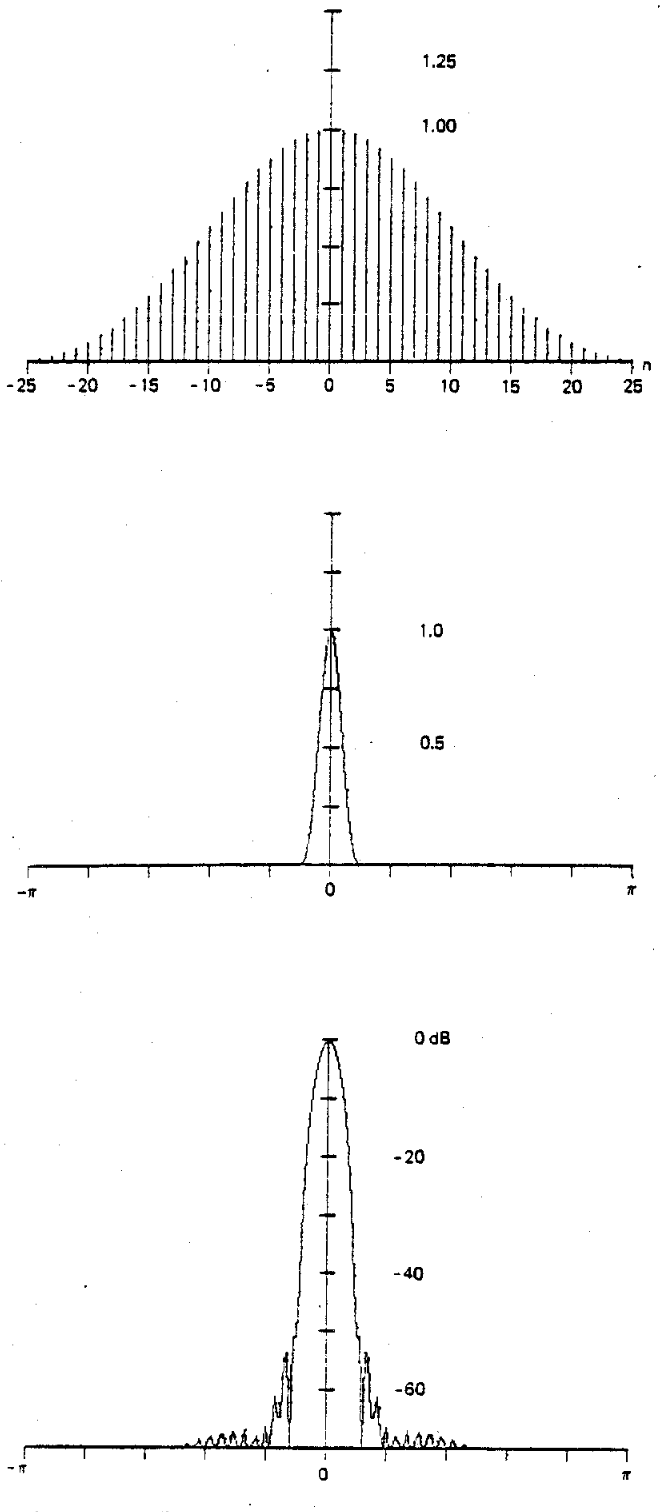


Figure 47. Barcilon-Temes Window, Fourier Transform, and Log-Magnitude of Transform. (a = 3.0)

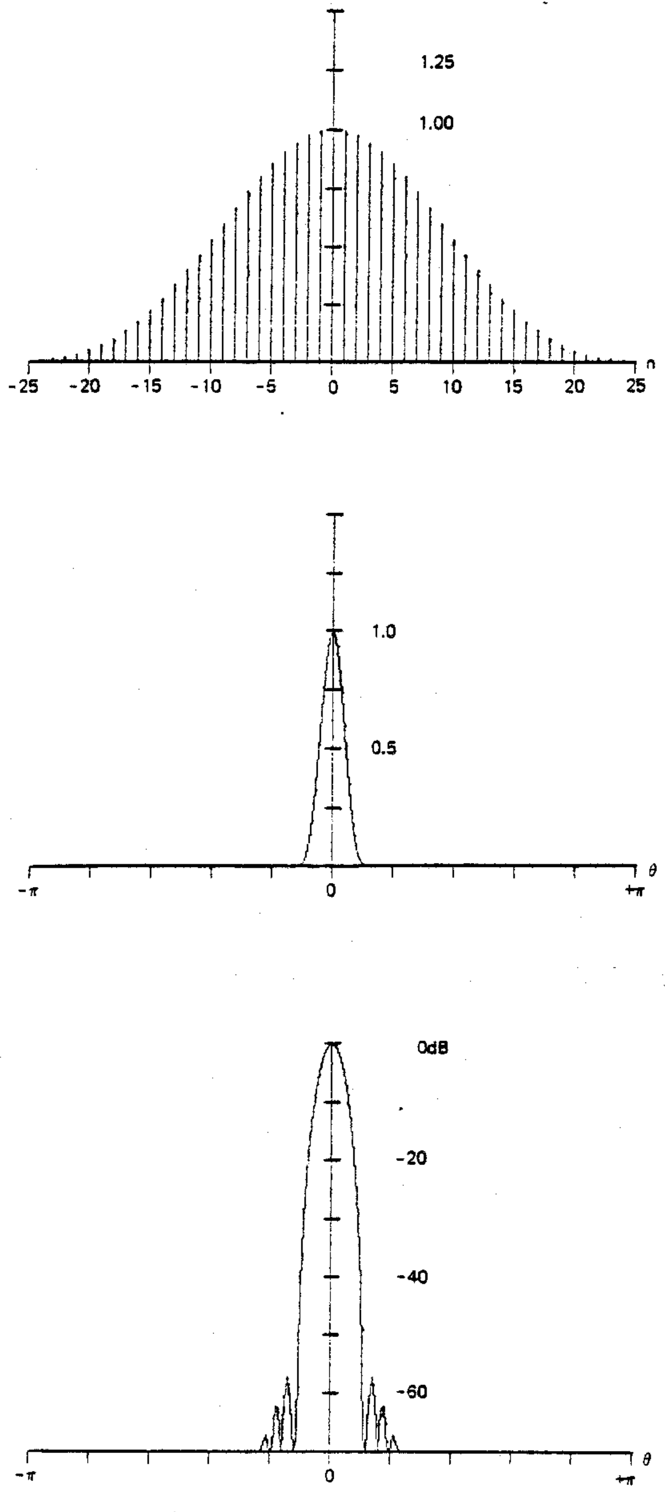


Figure 48. Barcilon-Temes Window, Fourier Transform, and Log-Magnitude of Transform. (a = 3.5)