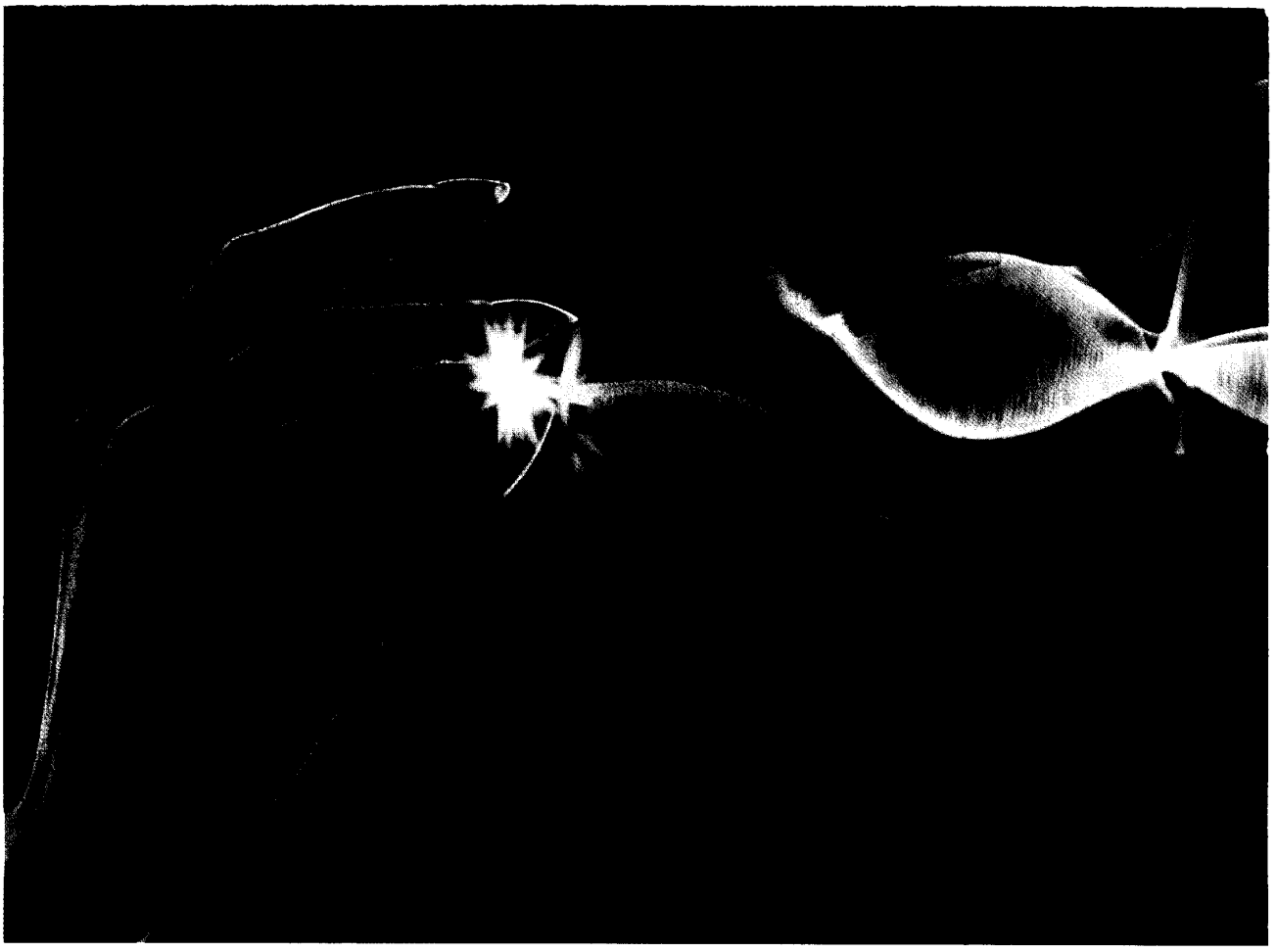


FPG International/Frank Rich

A Plain Man's Guide to the FFT

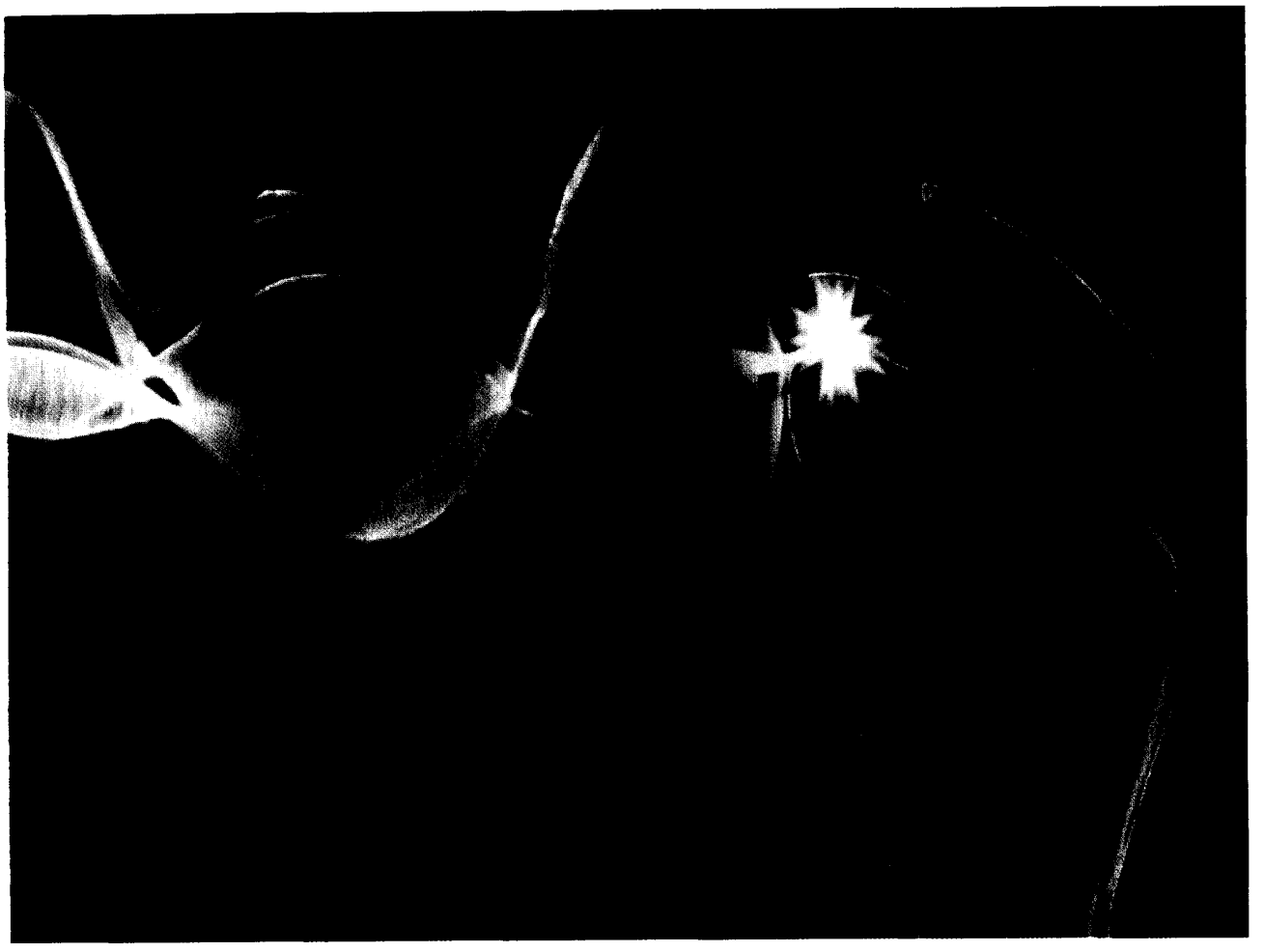
BY PETER KRANIAUSKAS

The familiar acronym *FFT* describes a set of algorithms that permit a **F**ast evaluation of the discrete **F**ourier **T**ransform. Replacing the original sequence by a sum of shorter ones requires smaller transforms, thereby dramatically reducing the number of computations. Within a few years of the FFT's introduction [1], most of what could be said about it had been said, much of it in two special issues of *IEEE Transactions* [2,3], distilled for wider appeal in *IEEE Spectrum* [4].

Though engineers and scientists readily accept the basic principles and benefits of the FFT, even the prescribed implementation rules, they often find prevailing mathematical justifications difficult to swallow.

This article provides a visual interpretation of the Cooley-Tukey family of FFT algorithms [1] in terms of the shifting properties of the Fourier transform, which equate shifts in one domain to proportional rotations in the alternative domain. This approach was proposed in an appendix to Chapter 6 of

1053-5888/94/$4.00©1994IEEE



[5], focusing on the radix-2 decimation-in-time algorithm. It is revised here and extended to other members of the Cooley-Tukey family, to cover decimations in time and in frequency and a choice of radices.

We first introduce the discrete Fourier transform, its shifting properties and the 3-dimensional graphics needed for complex functions.

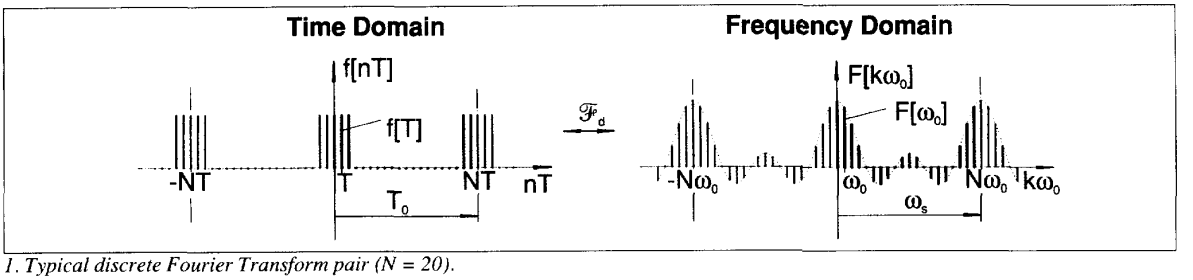
Discrete Fourier Transform

The discrete Fourier transform is the link between the time representation $f[nT]$ and the frequency representation $F[k\omega_0]$ of a signal that is discrete and periodic in both the time variable nT and the frequency variable $k\omega_0$. The relationship is expressed succinctly by the Fourier pair

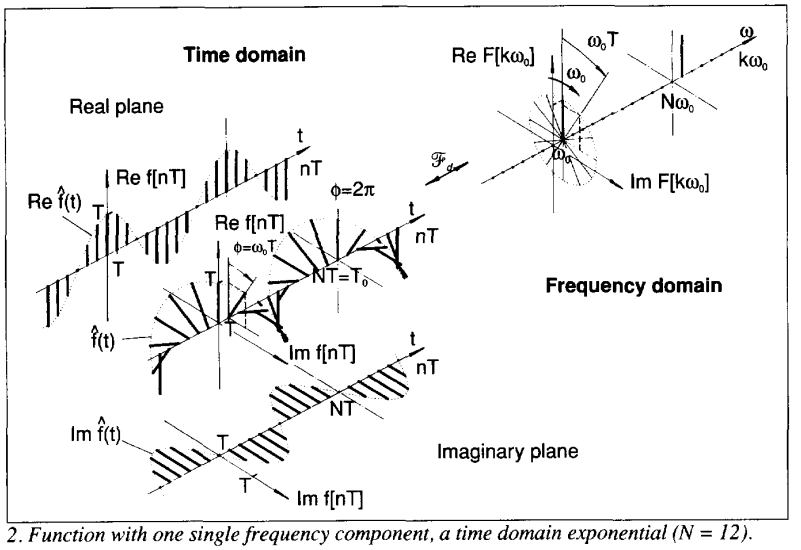
$$f[nT] \overset{\mathcal{F}_d}{\leftrightarrow} F[k\omega_0]$$

where $\mathcal{F}_d$ represents the discrete Fourier transform operator, and can be interpreted graphically, as illustrated for a typical pair in Fig. 1.

The operations involved in the forward and inverse transformations are expressed as finite sums of the form



1. Typical discrete Fourier Transform pair (N = 20).



2. Function with one single frequency component, a time domain exponential (N = 12).

$$F[k\omega_0] = \frac{1}{NT}\sum_{n=0}^{N-1} f[nT]\, e^{-jk\omega_0 nT} \qquad k = 0, 1, 2, ..., N-1 \tag{1}$$

$$f[nT] = T\sum_{k=0}^{N-1} F[k\omega_0]\, e^{jk\omega_0 nT} \qquad n = 0, 1, 2, ..., N-1 \tag{2}$$

The parameters T and $NT = T0$ represent the temporal sampling interval and temporal period, and these are inversely related to similar frequency domain parameters, namely the sampling interval $\omega_0 = 2\pi/T_0$, called the fundamental frequency, and the period $\omega_s = 2\pi/T = N\omega_0$, called the sampling frequency.

Transform as sum of exponentials

The above expressions effectively interpret each discrete component from one domain as a complex exponential in the alternative domain. The amplitude of the exponential is proportional to that of the component, while its period is inversely related to the component's distance from the origin. For instance, in Eq. 2, a frequency domain sample $F[l\omega_0]$, having frequency $\omega = l\omega_0$, becomes a time domain exponential $TF[l\omega_0]\, e^{jl\omega_0 nT}$. The transform is the sum of the contributions from the N components of one period.

These complex exponential functions can be interpreted as discrete versions of related continuous complex exponentials $e^{j\varphi}$, whose real and imaginary parts are given by the Euler equation

$$e^{j\varphi} = \cos\varphi + j\sin\varphi \tag{3}$$

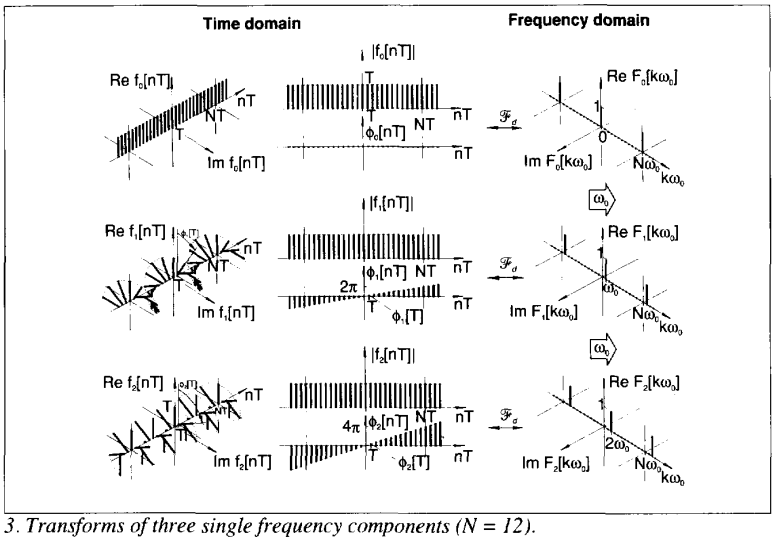
In Eqs. 1 and 2, the argument φ is a product of the frequency and time variables, $\varphi = \pm k\,\omega_0 nT$, one of which, nT in Eq. 1 and $k\omega_0$ in Eq. 2, is a fixed parameter of the exponential. We illustrate this with some simple cases.

Single frequency component

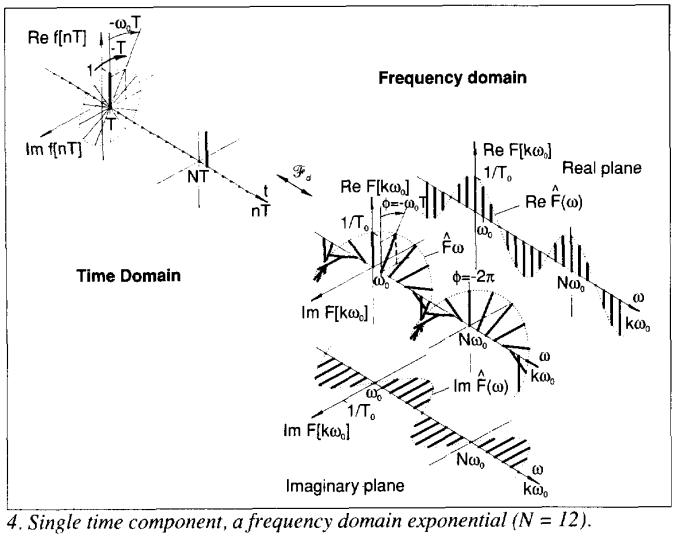
Figure 2 shows a function whose frequency domain contains one single component $F[\omega_0] = 1$, repeated periodically at intervals $\omega_s = N\omega_0$. This component is real-valued, of unit magnitude, and, since $k = 1$, is located at the fundamental frequency $\omega = \omega_0$. From Eq. 2, the time domain representation of this frequency component is the exponential $f[nT] = Te^{j\omega_0 nT}$.

To show all real and imaginary parts in context, we use three-dimensional graphics. If we think of the component $F[\omega_0]$ as a vector, rotating in its complex plane with angular velocity $+\omega_0$ and positive angles measured from the real axis towards the imaginary axis, then its succesive positions describe the angles $\varphi = \omega_0 nT$ of a circle in the frequency domain. Stringing out those successive vectors along the time axis, generates a right-handed helix representing the complex exponential function.

The dotted line linking the discrete vectors represents their continuous-time envelope $\hat{f}(t) = T\, e^{j\omega_0 t}$. Its projections on the vertical and horizontal planes, shown separately in the time domain of Fig. 2, give the real and imaginary parts $\mathrm{Re}\ \hat{f}(t) = T\cos\omega_0 t$ and $\mathrm{Im}\ \hat{f}(t) = T\sin\omega_0 t$. Slicing the enve-



3. Transforms of three single frequency components (N = 12).



4. Single time component, a frequency domain exponential (N = 12).

lope and its projections with parallel planes, spaced at sampling intervals T, gives the discrete function $f[nT]$ and its real and imaginary parts, Re $f[nT] = T \cos \omega_0 nT$, and Im $f[nT] = T \sin \omega_0 nT$, shown bold in Fig. 2.

Note that one full revolution $\varphi = 2\pi$ of the fundamental component $F[\omega_0]$ generates one full period $t = T_0$ of the helix and its projections. If the frequency of the component is increased, the representative vector rotates faster, thus describing a tighter helix in the time domain. This is shown in the progression of Fig. 3, where a single frequency component is located at $\omega = 0$, $\omega = \omega_0$, and $\omega = 2\omega_0$, respectively, with periodic repetitions at the sampling frequency, $\omega_s = N\omega_0$. The complex function of the time domain is shown both in three dimensions and in the more common magnitude-and-angle form.

Single time component

The above examples have dual counterparts in functions consisting of single, periodically repeated, time components. One such component, $f[T] = 1$, of unit magnitude located at the time $t = T$ is shown in Fig. 4. By Eq. 1, its discrete Fourier transform has the form $F[k\omega_0] = 1/T_0\, e^{-jk\omega_0 T}$, where $1/T$ is a scaling factor and the complex exponential is of the form of Eq. 3, and $\varphi = -k\omega_0 T$ is now a linear function of the frequency $k\omega_0$, with rate $-T$.

In Fig.4, this effect is visualised by imagining $f[T]$ as a vector rotating with angular velocity $-T$, to describe a circle in the time domain and a left-handed helix in the frequency domain. The envelope is a continuous function $\hat{F}(\omega) = 1/T_0\, e^{-j\omega T}$, with real and imaginary parts Re $\hat{F}(\omega) = 1/T_0 \cos\omega T$ and Im $\hat{F}(\omega) = -1/T_0 \sin\omega T$. Sliced with planes at multiples of the fundamental frequency, ω_0, they represent the discrete samples of $F[k\omega_0]$.

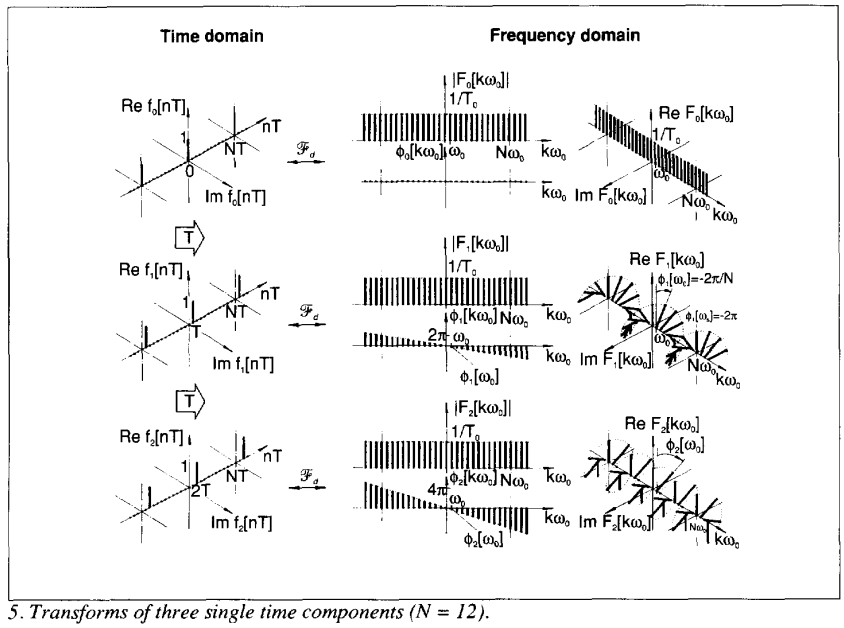
A component located at a higher multiple of T would describe a tighter helix. A progression with $t = 0$, $t = T$ and $t = 2T$ is illustrated in Fig. 5, where the complex function of the frequency domain is shown both in three dimensions, and in a magnitude-and-phase representation.

Shifting properties

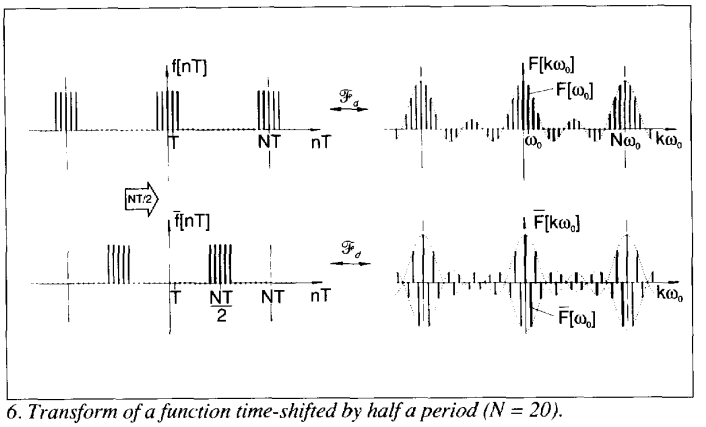
In Fig. 5, we can interpret the functions $f_1[nT]$ and $f_2[nT]$ as time-shifted versions of $f_0[nT]$. Clearly, a shift of one time interval, T, does not change the magnitude of the frequency representation, but adds the argument $\varphi = -k\omega_0 T$ of the exponential $e^{-jk\omega_0 T}$ to the existing phase.

When an arbitrary function $f[nT] \leftrightarrow F[k\omega_0]$ is shifted in time, all the exponentials associated with individual time components have the same amount of phase added, and this addition extends to the complete function, $F[k\omega_0]$. This is an effect of the time-shifting property of the discrete Fourier transform, expressed analytically as a frequency domain multiplication,

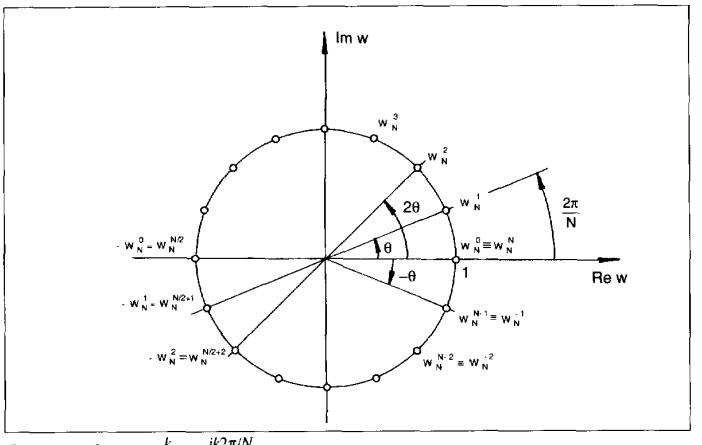
$$f[nT - mT] \overset{\mathcal{F}_d}{\leftrightarrow} e^{-jk\omega_0 mT} F[k\omega_0] \tag{4}$$



5. Transforms of three single time components (N = 12).



6. Transform of a function time-shifted by half a period (N = 20).



7. Roots of unity $w_n^k = e^{jk2\pi/N}$.

To quantify this relationship, we observe in Fig. 5 that one full backward turn $\varphi_1[\omega_s] = -2\pi$ of the sampling frequency component $F[\omega_s]$ corresponds to a rotation $\varphi_1[\omega_0] = -2\pi/N$ of the fundamental component $F[\omega_0]$, and shifts the time function $f[nT]$ by one sampling interval, T. Conversely, one full turn of the fundamental $F[\omega_0]$ would correspond to N turns of $F[\omega_s]$ and shift $f[nT]$ by one time period, $T_0 = NT$. An arbitrary time shift, mT, involves m/N turns of the fundamental and m turns of $F[\omega_s]$.

By analogy, a shift of the frequency representation $F[k\omega_0]$ is governed by the frequency-shifting property of the discrete Fourier transform,

$$e^{jl\omega_0 nT} f[nT] \overset{\mathcal{F}_d}{\leftrightarrow} F[k\omega_0 - l\omega_0] \tag{5}$$

The sequence of Fig. 3 is an example involving a single frequency component.

Shifting a function by half its period is relevant to radix-2 algorithms. Take for instance the function $f[nT] \leftrightarrow F[k\omega_0]$ of Fig. 1 and rotate the fundamental frequency component $F[\omega_0]$ half a turn and all other components a proportional number of half turns, as shown in Fig. 6. This operation is translated to the time domain as a shift of $f[nT]$ by half its period, $T_0 = NT$.

Simplified notation

To simplify expressions, it is customary to normalize the time-sampling interval to $T = 1$, giving a temporal period $T_0 = N$, sampling frequency $\omega_s = 2\pi$ and fundamental frequency $\omega_0 = 2\pi/N$. Next, the exponential core, $e^{j\omega_0 T}$, is given the notation $e^{j\omega_0 T} = e^{j2\pi/N} = w_N$. It represents the principal N^{th} root of unity (Fig. 7), whose N powers $w_N^0, W_N^1, w_N^2, \ldots, W_N^{N-1}$ form the set of N distinct roots of unity. Any higher power, as well as negative powers, coincide with one of the distinct roots. For even N, the relationship $w_N^{k+N/2} = -w_n^k$ holds, and will be used later to simplify the radix-2 decimation-in-time algorithm.

Finally, if ω_0 is dropped from the notation $F[k\omega_0]$, Eqs. 1 and 2 can be written more compactly as the pair

$$f[n] = \sum_{k=0}^{N-1} F[k] w_N^{nk} \overset{\mathcal{F}_N}{\leftrightarrow} F[k] = \frac{1}{N} \sum_{n=0}^{N-1} f[n] w_N^{-nk} \tag{6}$$

where the subscript N of the Fourier operator specifies the length of the transform. We use this notation in the remaining graphics, highlighting samples included in the sums by bolder lines.

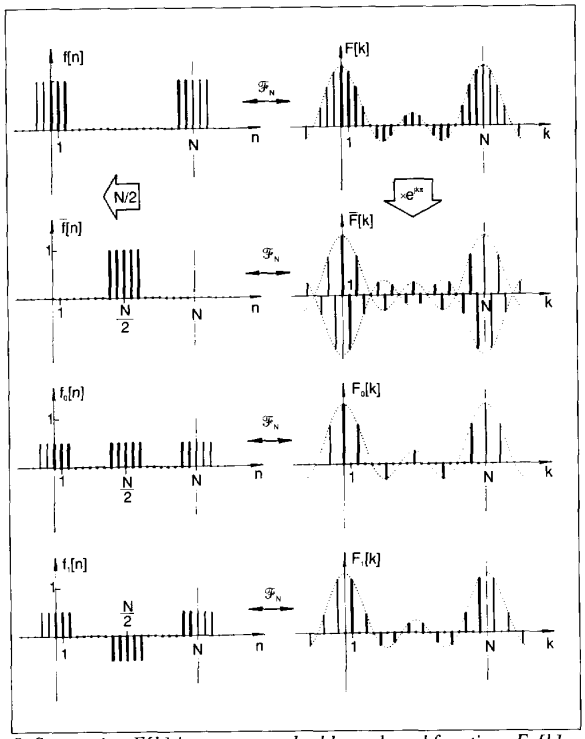
The economics of FFT algorithms

To compute one value of Eq. 6 involves the addition of N complex terms, each containing a multiplication. A complete N-point transform thus involves a number of complex operations proportional to N^2. Halving the size N reduces the number of operations to one quarter.

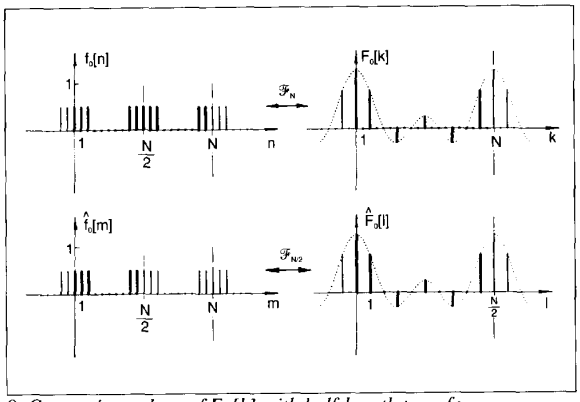
The FFT algorithms exploit this quadratic law. If N is composite, being the product of smaller factors, such as $N = N_1 N_2$, one domain of the original sequence is the sum of N_1 interleaved sequences, each of length N_2. Such separation is called *decimation* in the FFT literature, and the factor N_1 determines the radix of the algorithm. We start with an easy case where the radix is 2 and decimation takes place in frequency.

Radix-2 Decimation-in-Frequency

Given a discrete function $f[n] \leftrightarrow F[k]$ whose length N is an even number, we interpret the frequency representation $F[k]$ as a sum of two sequences, $F[k] = F_0[k] + F_1[k]$, where $F_0[k]$ extracts the even-numbered and $F_1[k]$ the odd-num-



8. Separating $F[k]$ into even- and odd-numbered functions $F_0[k]$ and $F_1[k]$.

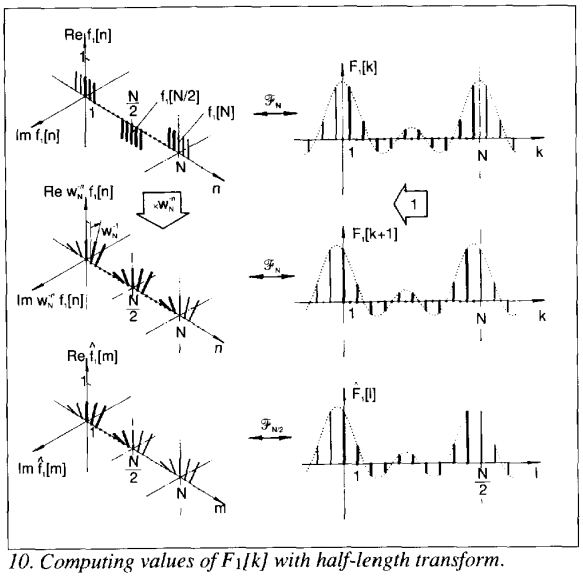


9. Computing values of $F_0[k]$ with half-length transform.

bered samples of $F[k]$, suitably interleaved with zeros, as shown in the frequency domain of the lower half of Fig.8.

To obtain the corresponding time representations, we first form an auxiliary function $\bar{f}[n] \leftrightarrow \bar{F}[k]$, by inverting all odd-numbered samples of $F[k]$ (second row, Fig.8). But this represents a half-period time-shift of $f[n]$, as shown earlier in Fig. 6, so that $\bar{f}[n] = f[n \pm N/2]$, where the $\pm$ sign shows indifference towards left-shifting or right-shifting a periodic function by half a period.

The function $F_0[k]$ is now seen as the semi-sum of $F[k]$ and $\bar{F}[k]$, while $F_1[k]$ is their semi-difference. More to the point, the same sums performed in the time domain yield their transforms $f_0[n]$ and $f_1[n]$ (Fig. 8).



10. Computing values of $F_1[k]$ with half-length transform.

Even-numbered and odd-numbered samples

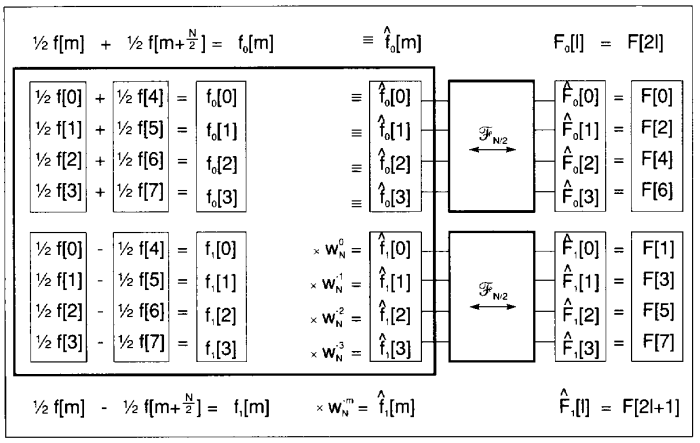
Since the odd-numbered samples of the sequence $F_0[k]$ are zero-valued and the period of $f_0[n]$ is effectively of length $N/2$, we can obtain the even-numbered values of the original function $F[k]$ by taking the half-length discrete Fourier transform $\mathcal{F}_{N/2}$ of the equivalent half-length function $\hat{f}_0[m] \leftrightarrow \hat{F}_0[l]$, shown in Fig. 9, where the change of variables reflects the change of fundamental frequency and effective time period.

If only the same could be said of $F_1[k]$, the full N-point discrete Fourier transform of the original function $f[n]$ could be obtained from two $N/2$-point transforms, thereby halving the overall number of computations. But the non-zero samples of $F_1[k]$ fall half-way between the sampling points of the half-length transform $\mathcal{F}_{N/2}$, while the effective period of $f_1[n]$ remains at full length, N.

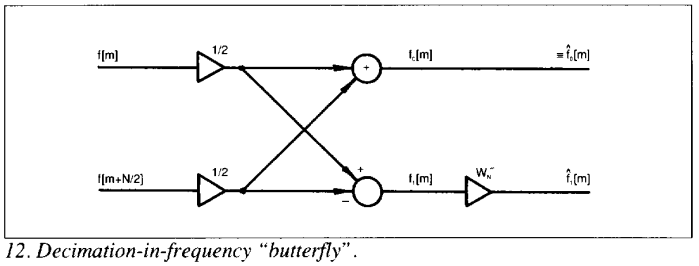
However, we can still use the half-length transform, if we first shift the whole function $F_1[k]$, including its envelope, one frequency-sampling interval to the left. By the frequency-shifting property, Eq. 5, this involves rotating the time samples $f_1[n]$, in proportion to their position n, as shown in the time domain of Fig. 10. We find that the component $f_1[N]$ takes one full turn, while $f_1[N/2]$ takes a half turn, with the result that the function $f_1[n]$ with period N, becomes the function $\hat{f}_1[m]$ whose period is half that length. The half-length transform of $\hat{f}_1[m]$ now provides the odd-numbered samples of the original transform, $F(k)$.

Decimation stage

We showed earlier that the sum $f_0[n] + f_1[n] = f[n] \leftrightarrow F_0[k] + F_1[k] = F[k]$ reconstructs both domains of the original function (Fig.8), and this brings together our interpretation of the radix-2 decimation-in-frequency algorithm. The even-numbered samples of $F[k]$ come



11. Algebraic operations involved in one decimation stage (N = 8).



12. Decimation-in-frequency "butterfly".

from $F_0[2l] = \hat{F}_0[l]$, while its odd-numbered samples are taken from $F_1[2l+1] = \hat{F}_1[l]$.

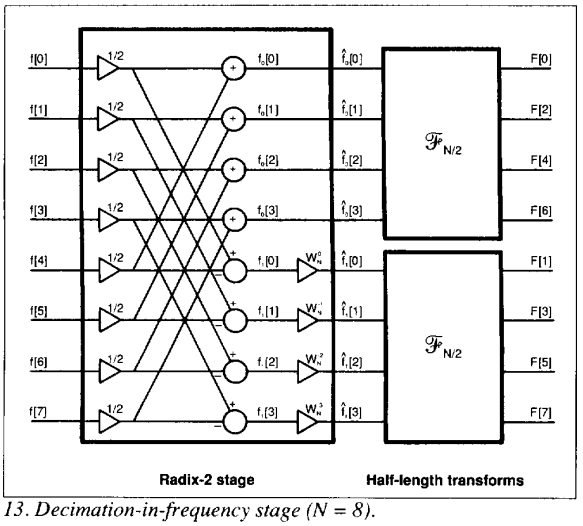
The algebraic operations involved in one decimation stage are shown in full in Fig. 11, for the case N = 8. All the intermediate sequences, shown boxed, are of length $N/2$. It is customary to combine the semi-sum and semi-difference of one related pair of samples into a diagram called 'butterfly' in the FFT literature (Fig. 12), where the associated multiplier w_N^{-m} is called their 'twiddle factor.' Using such representation for each of the $N/2$ pairs, a full decimation stage can be expressed succinctly, as shown in the diagram of Fig. 13. Note that the outputs $\hat{f}_0[n]$ and $\hat{f}_1[n]$ (n = 0, 1,..., $N/2$ -1) from the decimation stage can be interpreted as half-length time sequences whose discrete Fourier transforms, however obtained, provide the even and odd samples, respectively, of $F[k]$. One could think of the decimation stage as merely conditioning the time sequence $f[n]$ for a later transformation.

Radix-2 algorithm

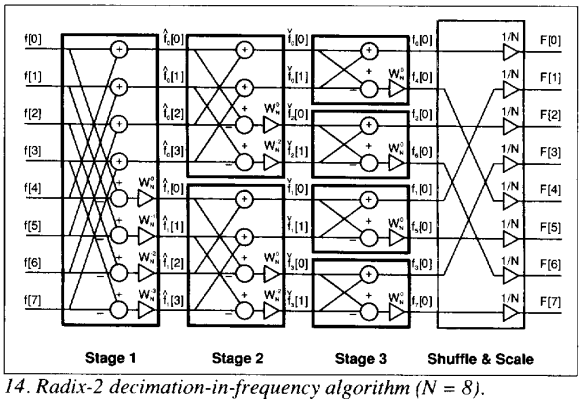
When the composite number N is a power of 2, $N = 2^r$, and all intermediate output sequences, of the form $\hat{f}_i[m]$, are of a length divisible by 2. The decimation process can then be recursively applied to subsequent stages, each with its own even and odd samples, leading to the well known radix-2 decimation-in-frequency algorithm. The particular case $N = 2^3 = 8$, which has $r = 3$ stages, is illustrated in Fig. 14.

As a consequence of repeatedly interleaving the even and odd samples of each decimation stage, the output samples $F[k]$ emerge in an order that is not the natural order 0, 1,..., N-1 of the input. But the disorder is rigidly structured and easily compensated by so-called 'bit-reversal' algorithms found in the literature.

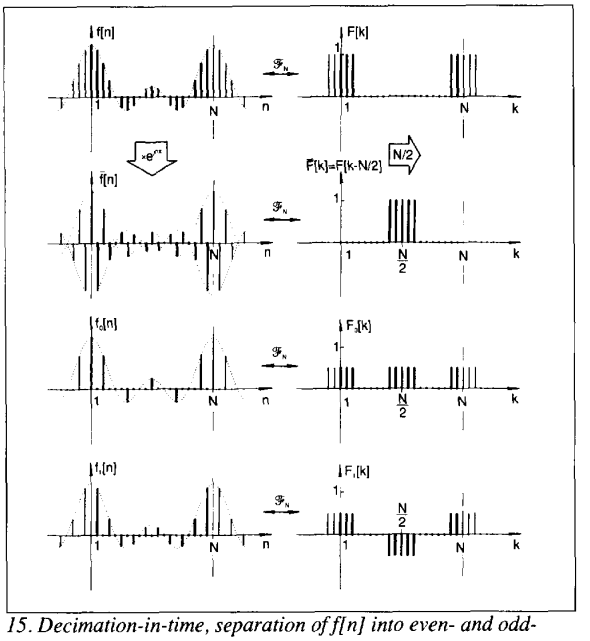
The scaling factors halving the inputs need not appear at every decimation stage. They represent each stage's contribution to the overall scaling factor $1/N$ of Eq. 6 and can be collected, to be applied only once, either to the input sequence $f[n]$ or the output sequence $F[k]$ of the algorithm. In Fig. 14, the overall scaling factors $1/N$ are introduced when re-ordering the outputs $F[k]$. Having thus justified the FFT procedure, it is nevertheless remarkable that a few sample rearrangements, seemingly postponing the transformation, acting together should achieve that transformation.



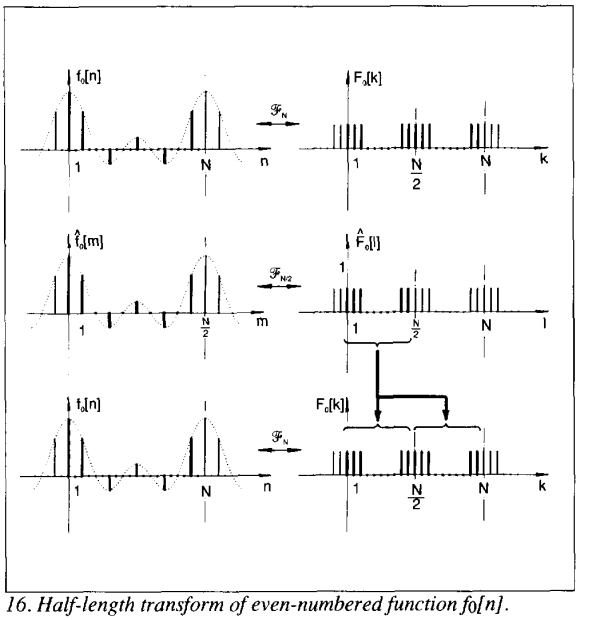
13. Decimation-in-frequency stage (N = 8).



14. Radix-2 decimation-in-frequency algorithm (N = 8).



15. Decimation-in-time, separation of $f[n]$ into even- and odd-numbered functions $f_0[n]$ and $f_1[n]$.



16. Half-length transform of even-numbered function $f_0[n]$.

Radix-2 Decimation-in-Time

Symmetries of the Fourier transform make it almost inevitable that the dual counterparts of the arguments put forward in the preceding section should lead to the decimation-in-time algorithm. To stress transform dualities, we use a new function $f[n] \leftrightarrow F[k]$ (top row of Fig. 15), obtained by exchanging the domains of the corresponding function of Fig.8. For time domain decimation we again form an auxiliary function $\bar{f}[n] \leftrightarrow \bar{F}[k]$, this time inverting the odd samples of $f[n]$, or, equivalently, multiplying by factors $e^{jn\pi}$ (second row of Fig. 15). From the frequency-shifting property, Eq. 5, this represents a half-period shift of $F[k]$.

Appropriate semi-sums and semi-differences yield the functions $f_0[n] \leftrightarrow F_0[k]$ and $f_1[n] \leftrightarrow F_1[k]$ of Fig. 15, which contain, respectively, the even- and odd-numbered samples of $f[n]$. Interleaved in time, the latter reconstruct both domains of the original function.

Half-length transforms

All non-zero samples of $f_0[n]$ are even-numbered and the effective period of $F_0[k]$ is of length $N/2$ (top row of Fig. 16), so that this function is well conditioned for half-length transformation (middle row of Fig. 16). The resulting sequence $\hat{F}_0[l]$, of length $N/2$, only needs duplicating (lower half of Fig. 16), to give the contribution of $f_0[n]$ to the original N-point transform, $F[k]$.

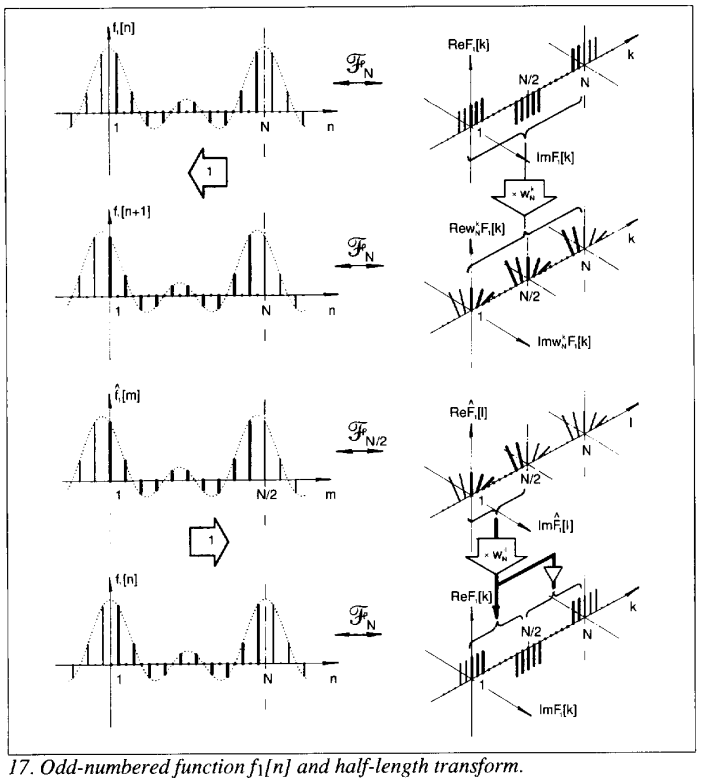
The odd-numbered function $f[n]$ is ill-conditioned and must first be nudged one time-sampling interval to the left (upper half of Fig. 17). In terms of the time-shifting property, Eq. 4, this operation involves rotating its frequency components $F_1[k]$ by multiples k of the angle $\varphi_N = 2\pi/N$; that is, multiplying each component by the k^{th} power of $w_N = e^{j2\pi/N}$.

After taking the half-length transform, the results can be restored to their original positions by back-shifting the time domain one sampling interval to the right, as indicated in the time domain of the lower half of Fig. 17. In the frequency domain, this would involve duplicating the $N/2$-point sequence $\hat{F}_1[l]$, thus assembling one full N-point period, and then undoing the earlier rotations by multiplying by corresponding negative powers of w_N. But, recalling the property, $w_N^{k+N/2} = -w_N^k$, half the above multiplications are redundant, as the replication scheme shown in the lower half of Fig. 17 yields the full contribution of $f_1[n]$ to the original transform $F[k]$.

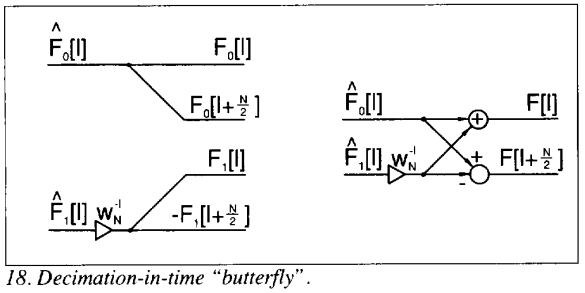
Decimation stage and radix-2 algorithm

To build up the decimation-in-time algorithm, we need to change our strategy. Thus we now start with the final decimation stage and back-propagate the arguments towards the input stage. Figures 15 to 17 justify the use of half-length transforms, but for actual implementation we only require the final replication schemes, highlighted in the lower right of Figs. 16 & 17. They represent the two halves of the well known decimation-in-time 'butterfly,' as shown in Fig. 18, where the multiplier w_N^{-l} represents the applicable 'twiddle factor.'

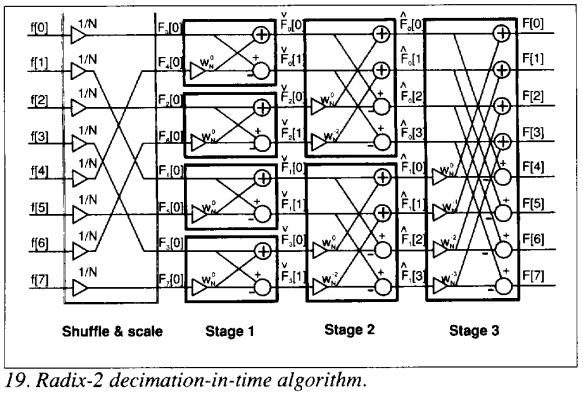
A complete radix-2 decimation-in-time algorithm ($N = 8$) is shown in Fig. 19. The right-hand box encloses the final decimation stage, whose inputs are two half-length sequences $\hat{F}_0[l]$ and $\hat{F}_1[l]$. But these are the outputs of the preceding decimation stage, which had similar quarter-length sequences as the inputs. The recursive process eventually leads to the



17. Odd-numbered function $f_1[n]$ and half-length transform.



18. Decimation-in-time "butterfly".



19. Radix-2 decimation-in-time algorithm.

first stage, whose inputs are unit-length sequences representing individual samples of $f[n]$.

Due to repeated separation into even- and odd-numbered samples, the function $f[n]$ must be shuffled before the first stage, as shown on the left side of Fig. 19. Also, since each separation involved semi-sums and semi-differences of samples, either the input or the output functions must be scaled by $1/N$.

Other Radices

The algorithms examined so far assume that the composite length N contains the factor 2. But the arguments are easily extended to other factors that are not necessarily multiples of 2. For brevity, we merely outline the essential concepts underlying radix-3 and radix-4 algorithms when decimation is in frequency. Other cases can be developed by analogy.

Recall the cases of radix-2 decimation, where we extracted even-numbered and odd-numbered samples by adding and subtracting an auxiliary function $\bar{f}[n] \leftrightarrow \bar{F}[k]$, where subtraction represents addition of the negated function. Negation means scaling by the real factor $a = -1 = e^{j\pi}$. Interpreted as a complex number operation, it is equivalent to rotating both domains of the function, each in its own complex plane, by the same angle $\theta = \pi$.

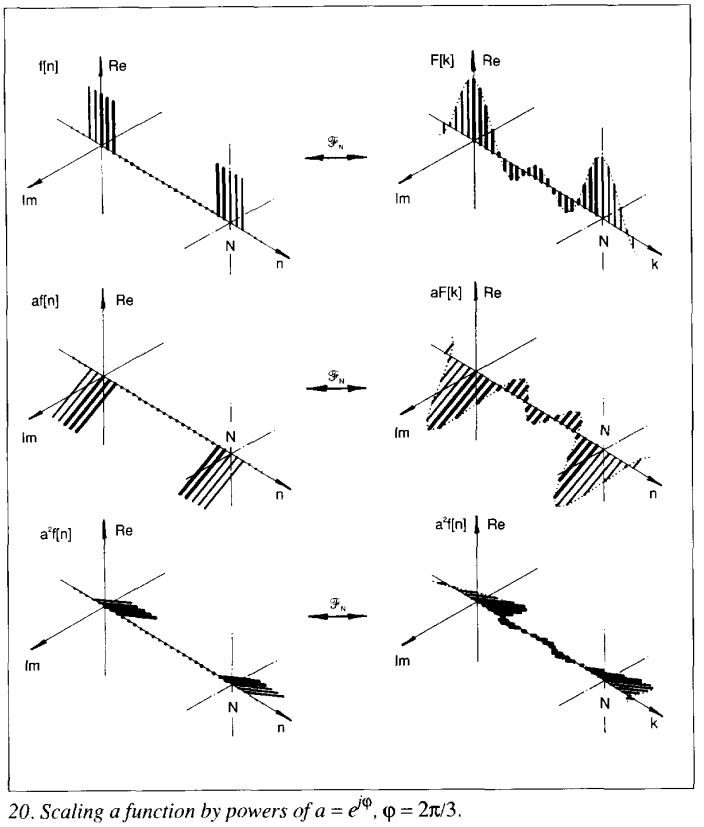
Sequences with other radices can be similarly extracted with the aid of complex scaling factors of the form $a = e^{j\theta}$, of unit magnitude and argument $\theta = 2\pi/radix$. Since the Fourier transform is a linear operator, such scaling merely rotates both domains of a function by the same angle, θ. An example relevant to radix-3 decimation is given in Fig. 20, where the scaling factors are powers of $a = e^{j2\pi/3}$.

Radix-3 decimation stage

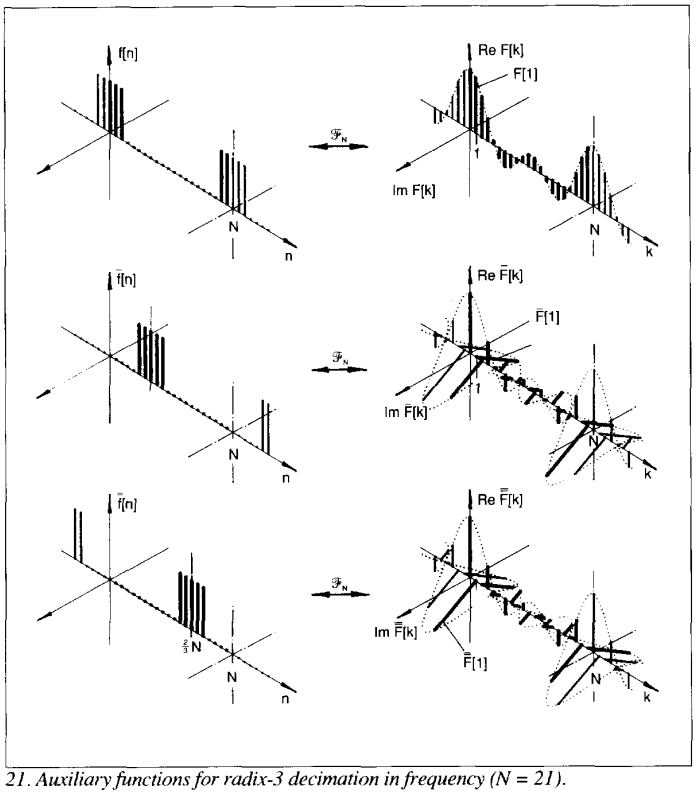
To frequency-decimate a function $f[n] \leftrightarrow F[k]$, whose period N is divisible by 3, we need to separate the original function $F[k]$ into three functions $F_0[k]$, $F_1[k]$, $F_2[k]$, each of effective length $N/3$, containing the samples (0, 3, 6, ..., N-3), (1, 4, 7, ..., N-2) and (2, 5, 8, ..., N-1), respectively (look ahead to the right half of Fig. 22).

To visualize the corresponding time domain representations, we proceed as in the radix-2 case, but using two auxiliary functions, $\bar{F}[k]$ and $\bar{\bar{F}}$, obtained by shifting the original time representation $f[n]$ by $N/3$ and $2N/3$ samples, respectively (Fig. 21). This is equivalent to rotating the fundamental frequency component $F[1]$ by angles $\theta = -2\pi/3$ and $\theta = -4\pi/3 = +2\pi/3$, respectively, with proportional rotations of all other components of $F[k]$.

The function $F_0[k]$ is now simply the sum $F_0[k] = (F[k] + \bar{F}[k] + \bar{\bar{F}}[k])/3$ (top row of Fig. 22), with a corresponding summation in the time domain. To obtain



20. Scaling a function by powers of $a = e^{j\varphi}$, $\varphi = 2\pi/3$.



21. Auxiliary functions for radix-3 decimation in frequency ($N = 21$).

$F_1[k]$ and its transform $f_1[n]$, we first rotate both domains of the function $\bar{f}[n] \leftrightarrow \bar{F}[k]$ by the angle $\theta = 2\pi/3$, as shown earlier in Fig. 20, and both domains of $\bar{\bar{f}}[n] \leftrightarrow \bar{\bar{F}}[k]$ by $\theta = -2\pi/3$, to then add the frequency domains as $F_1[k] = (F[k] + e^{j2\pi/3}\,\bar{F}[k] + e^{-j2\pi/3}\,\bar{\bar{F}}[k])/3$, and similarly the time domains, with the results shown in the middle row of Fig. 22.

Similar counter-rotations of $\bar{f}[n] \leftrightarrow \bar{F}[k]$ and $\bar{\bar{f}}[n] \leftrightarrow \bar{\bar{F}}[k]$, and sums of the form $F_2[k] = (F[k] + e^{-j2\pi/3}\,\bar{F}[k] + e^{j2\pi/3}\bar{\bar{F}}[k])/3$, lead to the remaining function $f_2[n] \leftrightarrow F_2[k]$ (bottom row of Fig. 22). The above operations resemble phasor additions in 3-phase electric networks. It is clear from the figure that, interleaved by addition, the functions thus obtained reconstruct the original function as $F_0[k]+F_1[k]+F_2[k] = F[k]$, with consistent results for the time domain.

The function $f_0[n] \leftrightarrow F_0[k]$ is well conditioned for $N/3$-point Fourier transformation, while $F_1[k]$ and $F_2[k]$ must first be nudged one and two samples, respectively, to the left, with equivalent rotations of their time domains, as shown in Fig. 23.

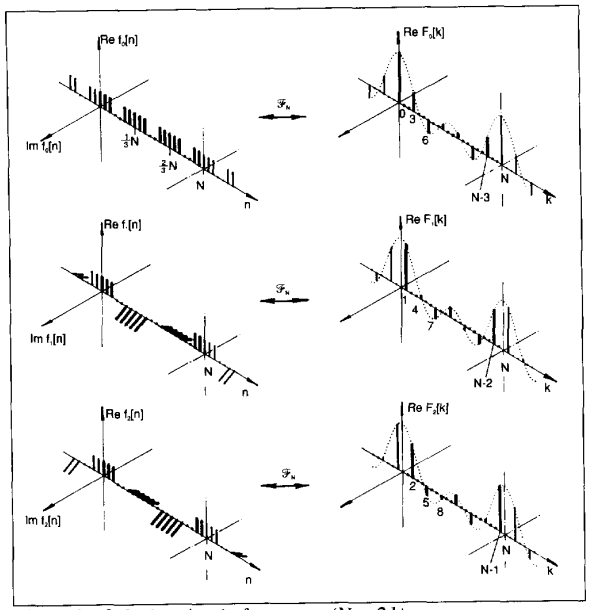
The process leading to the results of Fig. 23 provides the basis for the radix-3 decimation-in-frequency algorithm of Fig. 24. The procedure is essentially the same as for radix-2, and is left to the reader. The extended 'butterfly' processes the input samples in sets of 3, whereby a typical input set $\{f[m], f[m+N/3], f[m+2N/3]\}$ gives rise to an intermediate output set $\{\hat{f}_0[m], \hat{f}_1[m], \hat{f}_2[m]\}$.

Radix-4 decimation stage

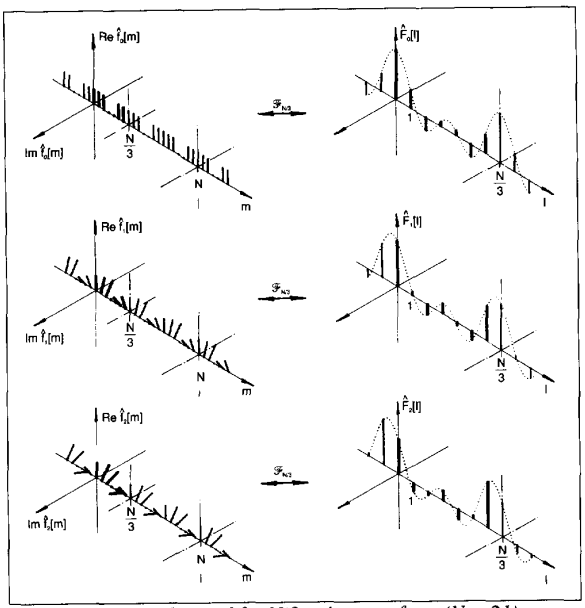
By similar arguments, when N is divisible by 4, we form three auxiliary functions $F[k]$, $\bar{F}[k]$ and $\bar{\bar{\bar{F}}}[k]$, which result from shifting $f[n]$ by $N/4$, $N/2$ and $3N/4$ samples, respectively, as shown in Fig.25. Their purpose is to extract four interleaving sequences from $F[k]$, whose indices 0, 1, 2, and 3 identify their first non-zero values, as shown in Fig. 26.

With appropriate conditioning, these lead to four $N/4$-point Fourier transforms, each requiring 1/16th the number of complex operations of the original N-point transform, and give rise to the algorithm of Fig. 27.

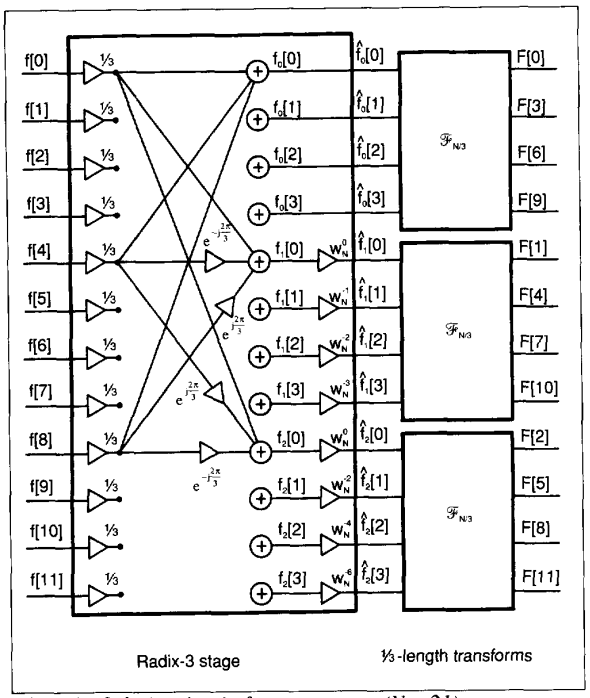
The potential savings per stage are a factor of 4, comparable to those of an equivalent pair of radix-2 stages. The multipliers within the extended butterfly only take the values 1, j, -1, or $-j$, attainable by simple exchanges involving real and imaginary parts, so that multiplication can be avoided. The subtle attraction of radix-4 decimation is that its larger size implies a smaller number



22. Radix-3 decimation in frequency (N = 21).



23. Functions conditioned for N/3 point transform (N = 21).



24. Radix-3 decimation-in-frequency stage (N = 21).

of decimation stages, hence a reduced number of multiplications involving 'twiddle factors'.

Conclusions

This article has provided basic and consistent interpretations of the Cooley-Tukey family of FFT algorithms. The developments rely entirely on linearity and the shifting properties of the discrete Fourier transform.

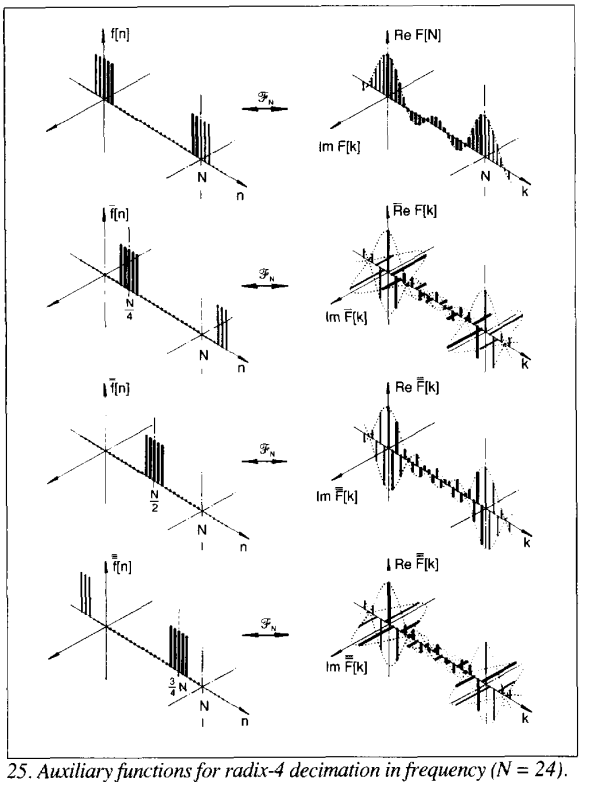
These algorithms apply to sequences whose length, N, is composite, such that $N = N_1 N_2$. The first step is to separate one domain of the original sequence into N_1 sequences, each containing N_2 equally spaced samples from the original sequence, suitably padded with zeros. This process is called 'decimation', with N_1 the radix. We provided a sequence of examples of decimation-in-frequency, and one of decimation-in-time.

In principle, the next step would involve short transforms of what are, in essence, sequences of length N_2. However, all but one of these sequences are ill-conditioned for such treatment, and need to be shifted in the decimated domain by either 1, 2, or up to $N_1 - 1$ samples, to move their first non-zero sample to the origin. We showed that such shifting winds up the complex functions of the transformed domain, thereby creating N_1 identical cycles of length N/N_1 within one original period of length N. All N_1 functions are thus made suitable for N_2-point transformation.

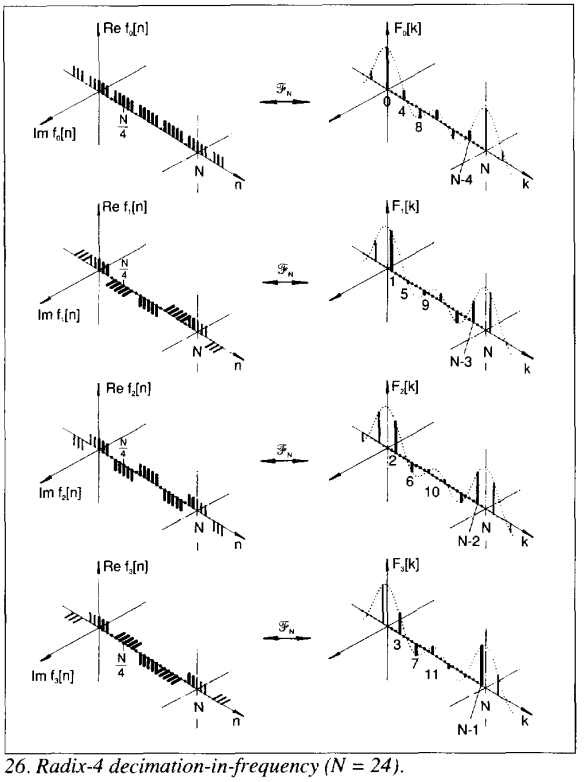
The above concepts need only be translated into algebraic operations or algorithmic flow charts. Depending on which domain is chosen for decimation, the algorithm is started from either the input stage or the output stage, to be extended forward or backward, with progressively shorter sequences, until the opposite end is reached.

To simplify the graphics we used very simple real even-symmetric functions for illustration. But the concepts are equally valid for arbitrary complex functions, where rotations in the complex plane represent additions to existing arguments.

The most popular FFT algorithms are those whose length, N is a power r of the radix 2, that is, $N = 2^r$. Common implementations usually have a number r of radix-2 stages, although the use of radix-4 and radix-8 stages can sometimes be advantageous. Combining different radices widens the choice for the length N. Readers wishing to explore implementation aspects of various FFT algorithms, and to compare their relative advan-

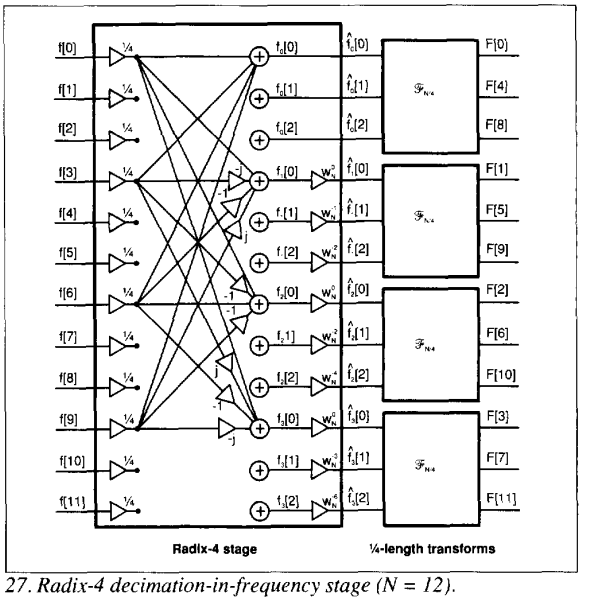


25. Auxiliary functions for radix-4 decimation in frequency (N = 24).



26. Radix-4 decimation-in-frequency (N = 24).

tages, are referred to Chapter 9 of [6], which also places the Cooley-Tukey family of algorithms into a wider FFT context.



27. Radix-4 decimation-in-frequency stage (N = 12).

The arguments presented in this article were developed with the aid of 3-dimensional graphics, a natural way to visualise complex functions. Concepts that may appear intricate in algebraic notation can be grasped at a glance. This approach is developed much further in [5], where it is used to derive the discrete Fourier transform, placing it in the wider context of the continuous Fourier transform and the Laplace and z-transforms, to then present their applications to continuous and discrete signals and systems.

Peter Kraniauskas was born in Lithuania in 1939, received an Engineering degree from the University of Buenos Aires, Argentina in 1962, and a Ph.D. degree from the University of Newcastle-upon-Tyne, UK, in 1974. His industrial experience extends over 20 years at General Motors, Xerox Research, and Racal Electronics. He is an engineering consultant based at Southampton, UK, and gives courses on signals and systems to professionals in high-tech industry and research establishments.

References

1. Cooley, J. W. and Tukey, J. W. "An algorithm for the machine computation of complex Fourier series," *Mathematics of Computation*, **19,** pp. 297-301, April 1965.

2. Special Issue on Fast Fourier Transforms and Applications to Digital Filtering and Spectral Analysis, *IEEE Trans. Audio Electroacoust.*, **AU-15,** No. 2, June 1967.

3. Special Issue on Fast Fourier Transforms, *IEEE Trans. Audio Electroacoust.*, **AU-17,** No. 2, June 1969.

4. Bergland, G. D., "A guided tour of the fast Fourier Transform," *IEEE Spectrum*, **6,** pp. 41-52, July 1969.

5. Kraniauskas, P., *Transforms in Signals and Systems,* Wokingham: Addison-Wesley, 1992.

6. Oppenheim, A. V. and Schafer, R. W., *Discrete-time Signal Processing,* Englewood Cliffs NJ: Prentice-Hall International, 1989.