COMPUTATION OF THE DISCRETE FOURIER TRANSFORM

Consequently, computation of the N-point DFT corresponds to the computation of N samples of the Fourier transform at N equally spaced frequencies $\omega_k = 2\pi k/N$, i.e., at N points on the unit circle in the z-plane. In this chapter, we discuss several methods for computing values of the DFT. The major focus of the chapter is a particularly efficient class of algorithms for the digital computation of the N-point DFT. Collectively, these efficient algorithms are called *fast Fourier transform* (FFT) algorithms.

the DFT of a finite-length sequence of length N is

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{kn}, \qquad k = 0, 1, \ldots, N-1, \tag{9.1}$$

where $W_N = e^{-j(2\pi/N)}$. The inverse discrete Fourier transform is given by

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]W_N^{-kn}, \qquad n = 0, 1, \ldots, N-1. \tag{9.2}$$

To compute all N values therefore requires a total of N^2 complex multiplications and $N(N-1)$ complex additions.

Most approaches to improving the efficiency of the computation of the DFT exploit the symmetry and periodicity properties of W_N^{kn}; specifically,

1. $W_N^{k[N-n]} = W_N^{-kn} = (W_N^{kn})^*$ (complex conjugate symmetry);
2. $W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n}$ (periodicity in n and k).

FFT algorithms are based on the fundamental principle of decomposing the computation of the discrete Fourier transform of a sequence of length N into successively smaller discrete Fourier transforms. The manner in which this principle is implemented leads to a variety of different algorithms, all with comparable improvements in computational speed.

DECIMATION-IN-TIME FFT ALGORITHMS

In computing the DFT, dramatic efficiency results from decomposing the computation into successively smaller DFT computations. In this process, we exploit both the symmetry and the periodicity of the complex exponential $W_N^{kn} = e^{-j(2\pi/N)kn}$. Algorithms in which the decomposition is based on decomposing the sequence $x[n]$ into successively smaller subsequences are called *decimation-in-time algorithms*.

With $X[k]$ given by

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{nk}, \qquad k = 0, 1, \ldots, N-1, \tag{9.10}$$

and separating $x[n]$ into its even- and odd-numbered points, we obtain

$$X[k] = \sum_{n \text{ even}} x[n]W_N^{nk} + \sum_{n \text{ odd}} x[n]W_N^{nk}, \tag{9.11}$$

or, with the substitution of variables $n = 2r$ for n even and $n = 2r + 1$ for n odd,

$$X[k] = \sum_{r=0}^{(N/2)-1} x[2r]W_N^{2rk} + \sum_{r=0}^{(N/2)-1} x[2r+1]W_N^{(2r+1)k} \tag{9.12}$$

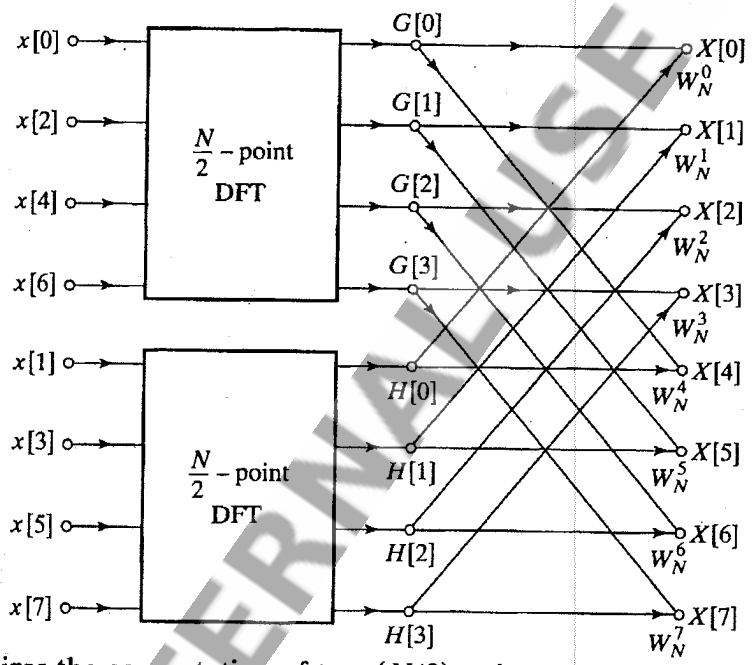
$$= \sum_{r=0}^{(N/2)-1} x[2r](W_N^2)^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1](W_N^2)^{rk}.$$

$$W_N^2 = e^{-2j(2\pi/N)} = e^{-j2\pi/(N/2)} = W_{N/2}. \tag{9.13}$$

$$X[k] = \sum_{r=0}^{(N/2)-1} x[2r]W_{N/2}^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1]W_{N/2}^{rk} \tag{9.14}$$

$$= G[k] + W_N^k H[k], \qquad k = 0, 1, \ldots, N-1.$$

Although the index k ranges over N values, $k = 0, 1, \ldots, N-1$, each of the sums must be computed only for k between 0 and $(N/2)-1$, since $G[k]$ and $H[k]$ are each periodic in k with period $N/2$.



Eq. (9.14) requires the computation of two $(N/2)$-point DFTs, which in turn requires $2(N/2)^2$ complex multiplications and approximately $2(N/2)^2$ complex additions if we do the $(N/2)$-point DFTs by the direct method. Then the two $(N/2)$-point DFTs must be combined, requiring N complex multiplications, corresponding to multiplying the second sum by W_N^k, and N complex additions, corresponding to adding the product obtained to the first sum. Consequently, the computation of Eq. (9.14) for all values of k requires at most $N + 2(N/2)^2$ or $N + N^2/2$ complex multiplications and complex additions. It is easy to verify that for $N > 2$, the total $N + N^2/2$ will be less than N^2.

$$G[k] = \sum_{r=0}^{(N/2)-1} g[r]W_{N/2}^{rk} = \sum_{\ell=0}^{(N/4)-1} g[2\ell]W_{N/2}^{2\ell k} + \sum_{\ell=0}^{(N/4)-1} g[2\ell+1]W_{N/2}^{(2\ell+1)k}, \tag{9.15}$$

or

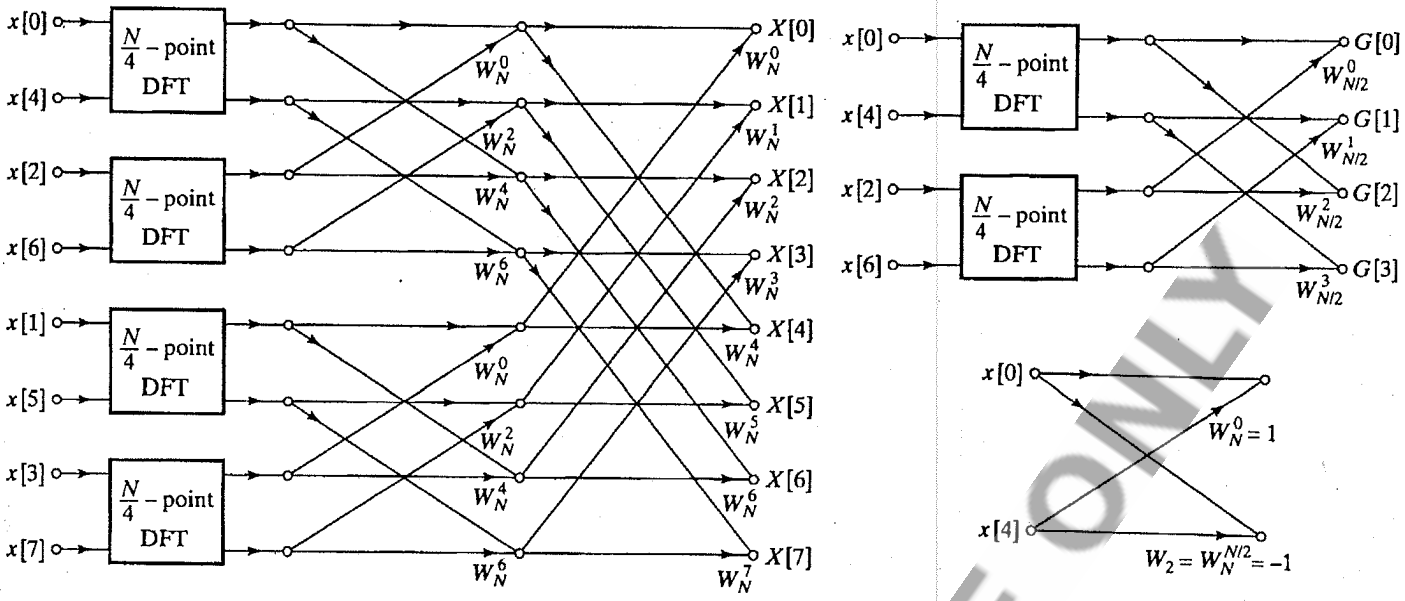
$$G[k] = \sum_{\ell=0}^{(N/4)-1} g[2\ell]W_{N/4}^{\ell k} + W_{N/2}^k \sum_{\ell=0}^{(N/4)-1} g[2\ell+1]W_{N/4}^{\ell k}. \tag{9.16}$$

If $N/2$ is even,

Similarly, $H[k]$ would be represented as

$$H[k] = \sum_{\ell=0}^{(N/4)-1} h[2\ell]W_{N/4}^{\ell k} + W_{N/2}^k \sum_{\ell=0}^{(N/4)-1} h[2\ell+1]W_{N/4}^{\ell k}. \tag{9.17}$$

$$W_{N/2} = W_N^2.$$



Because of the shape of the flow graph, this elementary computation is called a *butterfly*. Since

$$W_N^{N/2} = e^{-j(2\pi/N)N/2} = e^{-j\pi} = -1, \tag{9.18}$$

the factor $W_N^{r+N/2}$ can be written as

$$W_N^{r+N/2} = W_N^{N/2} W_N^r = -W_N^r. \tag{9.19}$$

When the $(N/2)$-point transforms are decomposed into $(N/4)$-point transforms, the factor of $(N/2)^2$ is replaced by $N/2 + 2(N/4)^2$, so the overall computation then requires $N + N + 4(N/4)^2$ complex multiplications and additions. If $N = 2^\nu$, this can be done at most $\nu = \log_2 N$ times, so that after carrying out this decomposition as many times as possible, the number of complex multiplications and additions is equal to $N\nu = N\log_2 N$.

For example, if $N = 2^{10} = 1024$, then $N^2 = 2^{20} = 1{,}048{,}576$, and $N\log_2 N = 10{,}240$, a reduction of more than two orders of magnitude!

DECIMATION-IN-FREQUENCY FFT ALGORITHMS

The decimation-in-time FFT algorithms are all based on structuring the DFT computation by forming smaller and smaller subsequences of the input sequence $x[n]$. Alternatively, we can consider dividing the output sequence $X[k]$ into smaller and smaller subsequences in the same manner. FFT algorithms based on this procedure are commonly called *decimation-in-frequency* algorithms.

Since

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \qquad k = 0, 1, \ldots, N-1, \tag{9.23}$$

the even-numbered frequency samples are

$$X[2r] = \sum_{n=0}^{N-1} x[n] W_N^{n(2r)}, \qquad r = 0, 1, \ldots, (N/2)-1, \tag{9.24}$$

which can be expressed as

$$X[2r] = \sum_{n=0}^{(N/2)-1} x[n]W_N^{2nr} + \sum_{n=N/2}^{N-1} x[n]W_N^{2nr}. \tag{9.25}$$

With a substitution of variables in the second summation

$$X[2r] = \sum_{n=0}^{(N/2)-1} x[n]W_N^{2nr} + \sum_{n=0}^{(N/2)-1} x[n+(N/2)]W_N^{2r[n+(N/2)]}. \tag{9.26}$$

$$W_N^{2r[n+(N/2)]} = W_N^{2rn}W_N^{rN} = W_N^{2rn}, \tag{9.27}$$

$$X[2r] = \sum_{n=0}^{(N/2)-1} (x[n] + x[n+(N/2)])W_{N/2}^{rn}, \qquad r = 0, 1, \ldots, (N/2)-1. \tag{9.28}$$

We can now consider obtaining the odd-numbered frequency points, given by

$$X[2r+1] = \sum_{n=0}^{N-1} x[n]W_N^{n(2r+1)}, \qquad r = 0, 1, \ldots, (N/2)-1. \tag{9.29}$$

$$X[2r+1] = \sum_{n=0}^{(N/2)-1} x[n]W_N^{n(2r+1)} + \sum_{n=N/2}^{N-1} x[n]W_N^{n(2r+1)}. \tag{9.30}$$

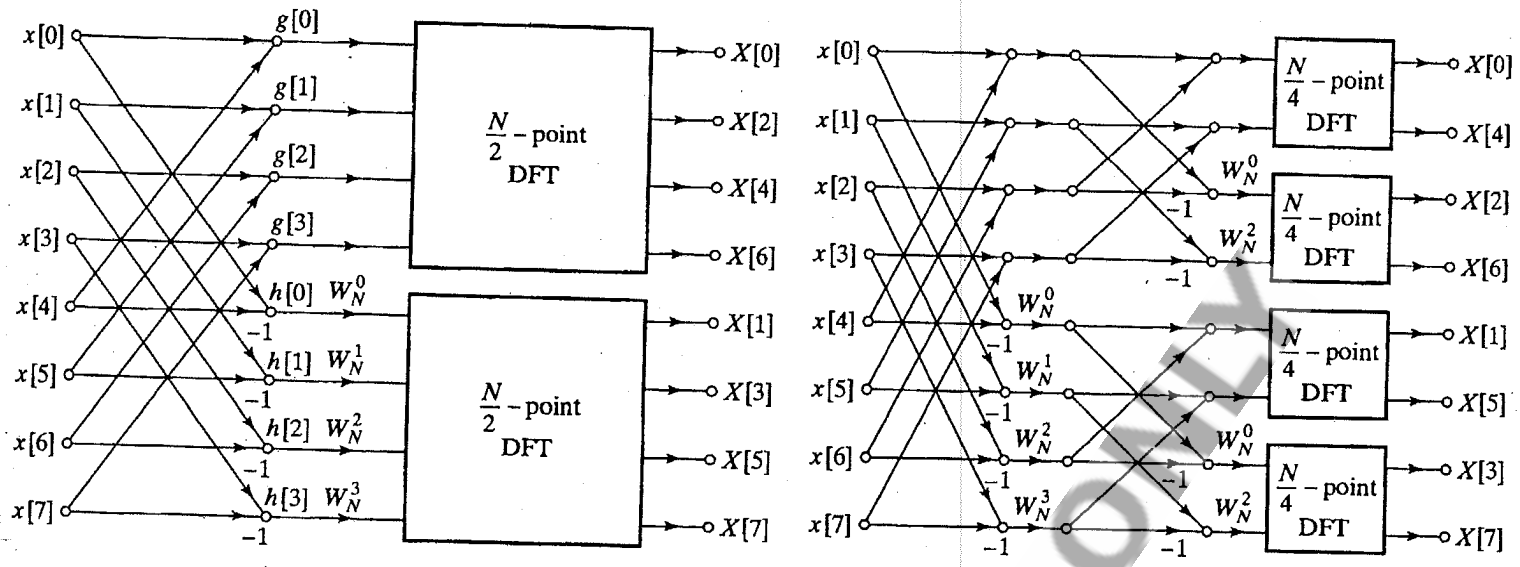
$$\begin{aligned}\sum_{n=N/2}^{N-1} x[n]W_N^{n(2r+1)} &= \sum_{n=0}^{(N/2)-1} x[n+(N/2)]W_N^{[n+(N/2)](2r+1)} \\ &= W_N^{(N/2)(2r+1)} \sum_{n=0}^{(N/2)-1} x[n+(N/2)]W_N^{n(2r+1)} \\ &= -\sum_{n=0}^{(N/2)-1} x[n+(N/2)]W_N^{n(2r+1)},\end{aligned} \tag{9.31}$$

where we have used the fact that $W_N^{(N/2)2r} = 1$ and $W_N^{(N/2)} = -1$.

$$X[2r+1] = \sum_{n=0}^{(N/2)-1} (x[n] - x[n+(N/2)])W_N^{n(2r+1)}, \tag{9.32}$$

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

$$X[2r+1] = \sum_{n=0}^{(N/2)-1} (x[n] - x[n+(N/2)])W_N^n W_{N/2}^{nr}, \qquad r = 0, 1, \ldots, (N/2)-1. \tag{9.33}$$



Thus, the total number of computations is the same for the decimation-in-frequency and the decimation-in-time algorithms.

Algorithms for More General Values of N

Although the special case of N a power of 2 leads to algorithms that have a simple structure, this is not the only restriction on N that can lead to a reduction in computation in the DFT. Indeed, in many cases it is desirable to evaluate the DFT efficiently for other values of N, and the same principles that were applied in the power of 2 decimation-in-time and decimation-in-frequency algorithms can be employed when N is a composite integer, i.e., the product of two or more integer factors. For example, if $N = RQ$, it is possible to express an N-point DFT as either the sum of R Q-point DFTs or as the sum of Q R-point DFTs and thereby obtain reductions in the number of computations. If N has many factors, the process can be repeated for each of the factors. Algorithms for general composite N involve more complicated indexing than the power of 2 case.

The Chirp Transform Algorithm

Another algorithm based on expressing the DFT as a convolution is referred to as the chirp transform algorithm (CTA).

To derive the CTA, we let $x[n]$ denote an N-point sequence and $X(e^{j\omega})$ its Fourier transform. We consider the evaluation of M samples of $X(e^{j\omega})$ that are equally spaced in angle on the unit circle

$$\omega_k = \omega_0 + k\Delta\omega, \qquad k = 0, 1, \ldots, M-1, \tag{9.36}$$

where the starting frequency ω_0 and the frequency increment $\Delta\omega$ can be chosen arbitrarily.

$$X(e^{j\omega_k}) = \sum_{n=0}^{N-1} x[n]e^{-j\omega_k n}, \qquad k = 0, 1, \ldots, M-1, \tag{9.37}$$

or, with W defined as

$$W = e^{-j\Delta\omega} \tag{9.38}$$

$$X(e^{j\omega_k}) = \sum_{n=0}^{N-1} x[n]e^{-j\omega_0 n}W^{nk}. \tag{9.39}$$

To express $X(e^{j\omega_k})$ as a convolution, we use the identity

$$nk = \tfrac{1}{2}[n^2 + k^2 - (k-n)^2] \tag{9.40}$$

$$X(e^{j\omega_k}) = \sum_{n=0}^{N-1} x[n]e^{-j\omega_0 n}W^{n^2/2}W^{k^2/2}W^{-(k-n)^2/2}. \tag{9.41}$$

Letting

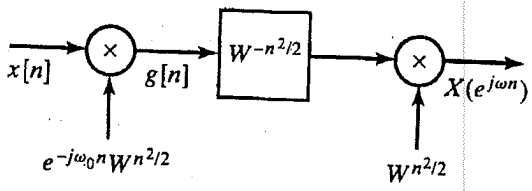
$$g[n] = x[n]e^{-j\omega_0 n}W^{n^2/2}, \tag{9.42}$$

we can then write

$$X(e^{j\omega_k}) = W^{k^2/2}\left(\sum_{n=0}^{N-1} g[n]W^{-(k-n)^2/2}\right), \qquad k = 0, 1, \ldots, M-1. \tag{9.43}$$

$$X(e^{j\omega_n}) = W^{n^2/2}\left(\sum_{k=0}^{N-1} g[k]W^{-(n-k)^2/2}\right), \qquad n = 0, 1, \ldots, M-1. \tag{9.44}$$

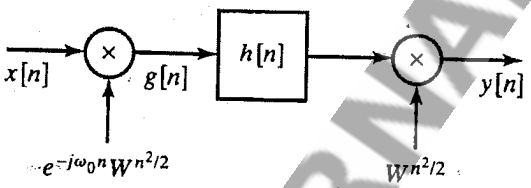
The sequence $W^{-n^2/2}$ can be thought of as a complex exponential sequence with linearly increasing frequency $n\Delta w$. In radar systems, such signals are called chirp



Since $g[n]$ is of finite duration, only a finite portion of the sequence $W^{-n^2/2}$ is used in obtaining $g[n] * W^{-n^2/2}$ over the interval $n = 0, 1, \ldots, M-1$, specifically, that portion from $n = -(N-1)$ to $n = M-1$. Let us define

$$h[n] = \begin{cases} W^{-n^2/2}, & -(N-1) \le n \le M-1, \\ 0, & \text{otherwise.} \end{cases} \tag{9.45}$$

$$g[n] * W^{-n^2/2} = g[n] * h[n], \qquad n = 0, 1, \ldots, M-1. \tag{9.46}$$



$$X(e^{j\omega_n}) = y[n], \qquad n = 0, 1, \ldots, M-1.$$

Evaluation of frequency samples using the procedure has a number of potential advantages. In general, we do not require $N = M$ as in the FFT algorithms, and neither N nor M need be composite numbers. In fact, they may be prime numbers if desired. Furthermore, the parameter ω_0 is arbitrary. This increased flexibility over the FFT does not preclude efficient computation, since the convolution can be implemented efficiently using an FFT algorithm