Then $z(n) = \sum_{p=0}^{3} x(p)y(n-p)$

$$= x(0)y(n) + x(1)y(n-1) + x(2)y(n-2) + x(3)y(n-3)$$

$$= x(1)y(n-1) + x(3)y(n-3) = y(n-1) + y(n-3)$$

$$z(0) = y(-1) + y(-3) = 1, \qquad z(1) = y(0) + y(-2) = 0$$

$$z(2) = y(1) + y(-1) = 1, \qquad z(3) = y(2) + y(0) = 0$$

$$z(n) = [1, 0, 1, 0]$$

Now for aperiodic convolution we must extend each function $x(n)$ and $y(n)$ by adding zero values in order to get no overlap in the intervals outside the given values. Imagine that the sequences are of length $N/2$, instead of N. Then form the sequences:

$$\hat{x}(n) = \left[x(0), x(1), \ldots, x\left(\frac{N}{2} - 1\right), 0, \ldots, 0\right]$$

$$\hat{y}(n) = \left[y(0), y(1), \ldots, y\left(\frac{N}{2} - 1\right), 0, \ldots, 0\right]$$

which are of length N. To both $x(n)$ and $y(n)$ we simply append $N/2$ zeros. Then $\hat{z}(n) = \hat{x}(n) * \hat{y}(n)$ is a periodic sequence of length N. Also, DFT $[\hat{z}(n)]$ = DFT $[\hat{x}(n) * \hat{y}(n)] = \hat{X}(k)\hat{Y}(k)$, where $\hat{X}(k)$ and $\hat{Y}(k)$ are the DFTs of $\hat{x}(n)$, $\hat{y}(n)$. All these functions, again, are periodic with period N. The periodic nature of these functions follows from the DFT and IDFT mechanisms. But by appending these zeros, we get a "true" convolution; that is, when the time sequences are shifted in the convolution operation, we get none of the overlap that periodic replication imposes. Let:

$$x(n) = [0, 1, 0, 1], y(n) = [0, 0, 0, 1] \quad \text{and} \quad N = 8.$$

Then $\hat{x}(n) = [0, 1, 0, 1, 0, 0, 0, 0], \hat{y}(n) = [0, 0, 0, 1, 0, 0, 0, 0]$

and $\hat{z}(n) = \sum_{p=0}^{7} \hat{x}(p)\hat{y}(n-p)$

$$= \hat{x}(0)\hat{y}(n) + \hat{x}(1)\hat{y}(n-1) + \hat{x}(2)\hat{y}(n-2) + \hat{x}(3)\hat{y}(n-3) + \hat{x}(4)\hat{y}(n-4) + \hat{x}(5)\hat{y}(n-5) + \hat{x}(6)\hat{y}(n-6) + \hat{x}(7)\hat{y}(n-7)$$

$$= \hat{y}(n-1) + \hat{y}(n-3)$$

$$\hat{z}(0) = \hat{y}(-1) + \hat{y}(-3) = 0, \qquad \hat{z}(1) = \hat{y}(0) + \hat{y}(-2) = 0$$

$$\hat{z}(2) = \hat{y}(1) + \hat{y}(-1) = 0, \qquad \hat{z}(3) = \hat{y}(2) + \hat{y}(0) = 0$$

$$\hat{z}(4) = \hat{y}(3) + \hat{y}(1) = 1, \qquad \hat{z}(5) = \hat{y}(4) + \hat{y}(2) = 0$$

$$\hat{z}(6) = \hat{y}(5) + \hat{y}(3) = 1, \qquad \hat{z}(7) = \hat{y}(6) + \hat{y}(4) = 0$$

$$\hat{z}(n) = [0, 0, 0, 0, 1, 0, 1, 0]$$

Note how this is considerably different from $z(n)$.

We avoid convolution, of course, by doing multiplication of the DFTs. To obtain $z(n)$, get $Z(k) = X(k)Y(k)$ and invert.

$$X(k) = \sum_{n=0}^{3} x(n)W^{nk} = x(0) + x(1)W^{k} + x(2)W^{2k} + x(3)W^{3k}, \qquad W = -j$$

$$= (-j)^k + (-j)^{3k}$$

$$X(0) = 2, \quad X(1) = 0, \quad X(2) = -2, \quad X(3) = 0,$$

$$X(k) = [2, 0, -2, 0]$$

Also $Y(k) = y(3)W^{3k} = (-j)^{3k}$

$$Y(0) = 1, \quad Y(1) = j, \quad Y(2) = -1, \quad Y(3) = -j,$$

$$Y(k) = [1, j, -1, -j]$$

$$Z(k) = [2, 0, 2, 0] \quad \text{and} \quad z(n) = \tfrac{1}{4}\sum_{k=0}^{3} Z(k)W^{-nk}$$

$$z(n) = \tfrac{1}{4}[2 + 2W^{-2n}] \rightarrow z(0) = 1,$$

$$z(1) = 0, \quad z(2) = 1, \quad z(3) = 0$$

$$z(n) = [1, 0, 1, 0]$$

which checks with the previous result. Finally:

$$\hat{Z}(k) = \hat{X}(k)\hat{Y}(k)$$

$$\hat{X}(k) = \sum_{n=0}^{7} \hat{x}(n)W^{nk} = W^k + W^{3k} \quad \text{and} \quad W = e^{-j\pi/4}$$

since $N = 8$

$$\hat{X}(0) = 2, \quad \hat{X}(1) = -j\sqrt{2}, \quad \hat{X}(2) = 0, \quad \hat{X}(3) = -j\sqrt{2},$$

$$\hat{X}(4) = -2, \quad \hat{X}(5) = j\sqrt{2}, \quad \hat{X}(6) = 0, \quad \hat{X}(7) = j\sqrt{2}$$

Therefore $\hat{X}(k) = [2, -j\sqrt{2}, 0, -j\sqrt{2}, -2, j\sqrt{2}, 0, j\sqrt{2}]$

And $\hat{Y}(k) = W^{3k}$.

$$\hat{Y}(0) = 1, \quad \hat{Y}(1) = \frac{-1-j}{\sqrt{2}}, \quad \hat{Y}(2) = j,$$

$$\hat{Y}(3) = \frac{1-j}{\sqrt{2}}, \quad \hat{Y}(4) = -1,$$

$$\hat{Y}(5) = \frac{1+j}{\sqrt{2}}, \quad \hat{Y}(6) = -j, \quad \hat{Y}(7) = \frac{j-1}{\sqrt{2}}$$

Thus $$\hat{Y}(k) = \left[1, \frac{-1-j}{\sqrt{2}}, j, \frac{1-j}{\sqrt{2}}, -1, \frac{1+j}{\sqrt{2}}, -j, \frac{j-1}{\sqrt{2}}\right]$$

Therefore $$\hat{Y}(k)\hat{X}(k) = \hat{Z}(k) = [2, j-1, 0, -1-j, 2, j-1, 0, -1-j]$$

$$\hat{z}(n) = \frac{1}{8}\sum_{k=0}^{7} \hat{Z}(k) W^{-nk} = \frac{1}{8}[2 + (j-1)W^{-n} + (-1-j)W^{-3n} + 2W^{-4n} + (j-1)W^{-5n} + (-1-j)W^{-7n}]$$

and $$\hat{z}(0) = \frac{1}{8}[2 + (j-1) + (-1-j) + 2 + (j-1) + (-1-j)] = 0$$

$$\hat{z}(1) = \frac{1}{8}\left[2 + (j-1)\frac{(1+j)}{\sqrt{2}} + (-1-j)\frac{(-1+j)}{\sqrt{2}} - 2 + (j-1)\frac{(-1-j)}{\sqrt{2}} + \frac{(-1-j)}{\sqrt{2}}(1-j)\right] = 0$$

$$\hat{z}(2) = \frac{1}{8}[2 + (j-1)j + (-1-j)(-j) + 2 + (j-1)j + (-1-j)(-j)] = 0$$

$$\hat{z}(3) = \frac{1}{8}\left[2 + (j-1)\frac{(-1+j)}{\sqrt{2}} + (-1-j)\frac{(1+j)}{\sqrt{2}} + 2(-1) + (j-1)\frac{(1-j)}{\sqrt{2}} + (-1-j)\frac{(-1-j)}{\sqrt{2}}\right] = 0$$

$$\hat{z}(4) = \frac{1}{8}[2 + (j-1)(-1) + (-1-j)(-1) + 2(1) + (j-1)(-1) + (-1-j)(-1)] = 1$$

$$\hat{z}(5) = \frac{1}{8}\left[2 + (j-1)\frac{(-1-j)}{\sqrt{2}} + (-1-j)\frac{(1-j)}{\sqrt{2}} + 2(-1) + (j-1)\frac{(1+j)}{\sqrt{2}} + (-1-j)\frac{(j-1)}{\sqrt{2}}\right] = 0$$

$$\hat{z}(6) = \frac{1}{8}[2 + (j-1)(-j) + (-1-j)(j) + 2(1) + (j-1)(j) + (-1-j)j] = 1$$

$$\hat{z}(7) = \frac{1}{8}\left[2 + (j-1)\frac{(1-j)}{\sqrt{2}} + (-1-j)\frac{(-1-j)}{\sqrt{2}} + 2(-1) + (j-1)\frac{(j-1)}{\sqrt{2}} + (-1-j)\frac{(1+j)}{\sqrt{2}}\right] = 0$$

Therefore $\hat{z}(n) = [0, 0, 0, 0, 1, 0, 1, 0]$

which checks with the previous result.

Since most of the other properties are analogous to the continuous Fourier transform properties, we end our discussion here. Some other properties will be dealt with as problems at the end of the chapter. We turn now to consider the fast Fourier transform.

9-4 THE FAST FOURIER TRANSFORM

The fast Fourier transform (FFT) is an algorithm or a procedure with which the discrete Fourier transform can be computed using far fewer calculations. For many signal processing operations, computational requirements using the FFT can be reduced considerably. Although the FFT has had a long and interesting history (see Brigham, 1974, pp. 8–9), it was a paper by Cooley and Tukey (1965) that really put the FFT on the map. Since their famous work, which heralded a major technological breakthrough, there have been hundreds of papers written on the FFT. Fields as diverse as seismology, radar, astronomy, linear systems, optics, quantum physics, neurology, and communications have benefited from the FFT, mainly because the DFT requires N^2 complex multiply and add operations, whereas the FFT needs only aproximately $N/2 \log_2 N$ operations. For

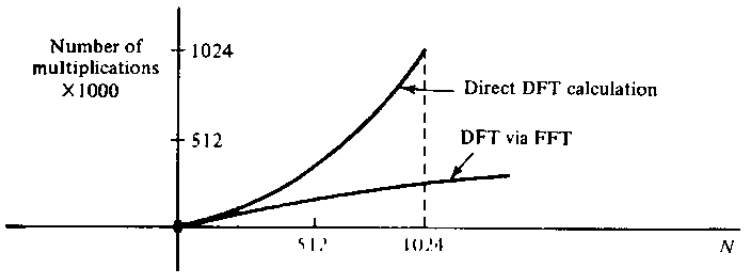


Figure 9-11 Computation for direct DFT and DFT via FFT

low values of N these numbers are not very different, but for large N the difference becomes dramatic. (See Figure 9-11.)

Numerous FFT algorithms and variations on these algorithms are presented in Brigham's (1974) book and, more recently, at a more advanced level, in Elliott and Rao's (1982) work, both containing very extensive bibliographies. Our approach to the FFT will be limited to an introduction. We will consider what are called *power-of-2 FFT algorithms* and focus on the two types: *Decimation-in-time* algorithms and *decimation-in-frequency* algorithms.

9-4-1 The Decimation-in-Time Algorithm

Assume that the number of points in the data sequence is a power of 2: $N = 2^r$ where r is a positive integer. The number of points N will be repeatedly divided by 2 in order to facilitate the algorithm. Algorithms for arbitrary N values have been developed and are even more efficient than power-of-2 algorithms; however, they are more complicated and will not be explored here.

Let us write $F(k)$ as follows:

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

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

even n odd n (9-30)

Then this expression can be written as:

$$F(k) = \sum_{n=0}^{N/2-1} f_1(n) W^{2kn} + \sum_{n=0}^{N/2-1} f_2(n) W^{k(2n+1)} \tag{9-31}$$

where $f_1(n) = f(2n)$ and $f_2(n) = f(2n+1)$ are, respectively, the even and odd components in $f(n)$, the original sequence to be transformed. Now let:

$$F_1(k) = \sum_{n=0}^{N/2-1} f_1(n) W^{2kn} \quad \text{and} \quad F_2(k) = \sum_{n=0}^{N/2-1} f_2(n) W^{2kn}$$

and we can write:

$$F(k) = F_1(k) + W^k F_2(k) \tag{9-30}$$

which expresses the original N point DFT as a summation of two $N/2$ point DFTs with the second multiplied by W^k. The DFT $F(k)$ has N points $F(0), F(1), \ldots, F(N-1)$. Imagine breaking this sequence of N points into two sets or sequences, the first being $F(0), F(1), \ldots, F(N/2-1)$ and the second being $F(N/2), F(N/2+1), \ldots, F(N-1)$. At this point in the development, the FFT advantage comes clearly to light. It can be shown that:

$$F\left(k + \frac{N}{2}\right) = F_1(k) - W^k F_2(k) \tag{9-31}$$

Comparing Equations 9-30 and 9-31, we observe that, except for a minus sign, their right-hand sides are equal. The same information used in Equation 9-30 to calculate $F(k)$ for $k = 0, \ldots, N/2-1$ can be used in Equation 9-31 to calculate $F(k)$ for $k = N/2, N/2+1, \ldots, N-1$, where the latter calculation is $F(k+N/2)$ for $k = 0, 1, \ldots, N/2-1$. Varying k only over the first half of its range $k = 0, 1, \ldots, N/2-1$, and using Equations 9-30 and 9-31, we can calculate all N points of $F(k)$.

EXAMPLE 9-10

Show how Equation 9-31 follows from Equation 9-30.

Solution. From Equation 9-30 we can write $F(k+N/2) = F_1(k+N/2) + W^{k+N/2} F_2(k+N/2)$.

But
$$F_1(k+N/2) = \sum_{n=0}^{N/2-1} f_1(n) W^{2n(k+N/2)} = \sum_{n=0}^{N/2-1} f_1(n) W^{2kn} W^{nN}$$

$$= \sum_{n=0}^{N/2-1} f_1(n) W^{2kn} = F_1(k)$$

since
$$W^{nN} = e^{-j(2\pi/N)(nN)} = e^{-j2n\pi} = 1$$

Likewise:

$$F_2\left(k + \frac{N}{2}\right) = F_2(k)$$

Also,

$$W^{k+N/2} = W^k W^{N/2} = W^k e^{-j\pi} = -W^k$$

Substituting, we get Equation 9-31.

Now the operations involved in Equations 9-30 and 9-31 can be expressed in a convenient signal flow representation called a "butterfly," which is illustrated in Figure 9-12. This butterfly arrangement can be expanded gradually backward to form a "cocoon" type lattice. Similarly, each of $F_1(k)$ and $F_2(k)$ can be treated as was $F(k)$. Since $F_1(k)$ and $F_2(k)$ are $N/2$ point sequences, we will break each of these $N/2$ point sequences up into two $N/4$ point sequences:

$$F_1(k) = F_3(k) + W^{2k} F_4(k), \qquad k = 0, 1, \ldots, \frac{N}{4} - 1 \tag{9-32}$$

$$F_1\left(k + \frac{N}{4}\right) = F_3(k) - W^{2k} F_4(k), \qquad k = 0, 1, \ldots, \frac{N}{4} - 1 \tag{9-33}$$

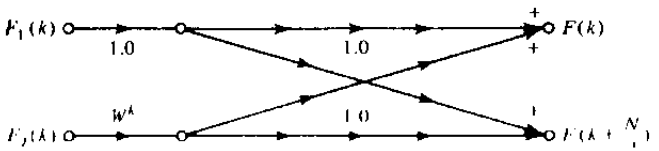


Figure 9-12 Butterfly computation scheme

Then the $N/4$-point sequences $F_3(k)$ and $F_4(k)$ can be split in two, and so on, until we have only 1-point sequences remaining. An 8-point FFT can be used to illustrate this procedure.

EXAMPLE 9-11

Construct an 8-point decimation-in-time (DIT) FFT butterfly signal flow representation.

Solution

$$F(k) = F_1(k) + W^k F_2(k), \qquad k = 0, 1, 2, 3$$

$$F\left(k + \frac{N}{2}\right) = F_1(k) - W^k F_2(k), \qquad k = 0, 1, 2, 3$$

$F(k)$ is an 8-point DFT. $F_1(k)$ and $F_2(k)$ are both 4-point DFTs, each of which can be separately decomposed:

$$F_1(k) = F_3(k) + W^{2k} F_4(k), \qquad k = 0, 1$$

$$F_1\left(k + \frac{N}{4}\right) = F_3(k) - W^{2k} F_4(k), \qquad k = 0, 1$$

and

$$F_2(k) = F_5(k) + W^{2k} F_6(k), \qquad k = 0, 1$$

$$F_2\left(k + \frac{N}{4}\right) = F_5(k) - W^{2k} F_6(k), \qquad k = 0, 1$$

Now $F_3(k)$, $F_4(k)$, $F_5(k)$, $F_6(k)$ are each 2-point DFTs that can be expressed as sums and differences of the original data.

$$F_3(k) = \sum_{n=0}^{N/4-1} f_3(n) W^{4kn}, \qquad \text{where } f_3(n) = f_1(2n) = f(4n)$$

$$F_4(k) = \sum_{n=0}^{N/4-1} f_4(n) W^{4kn}, \qquad \text{where } f_4(n) = f_1(2n+1) = f(4n+2)$$

Because F_3 and F_4 both relate to F_1, the time functions f_3 and f_4 both relate to f_1. Recall that $f_1(n) = f(2n)$. In a similar manner:

$$F_5(k) = \sum_{n=0}^{N/4-1} f_5(n) W^{4kn}, \qquad F_6(k) = \sum_{n=0}^{N/4-1} f_6(n) W^{4kn}$$

where $f_5(n) = f_2(2n) = f(4n+1)$

and $$f_6(n) = f_2(2n+1) = f(4n+3)$$

Now if we carry out the expressions for F_3, F_4, F_5, and F_6, we obtain:

$$F_3(k) = f_3(0)W^0 + f_3(1)W^{4k}, \qquad \text{for } k = 0, 1 \text{ and } W^4 = e^{j\pi} = -1$$

Also

$$F_4(k) = f_4(0)W^0 + f_4(1)W^{4k}$$

$$F_5(k) = f_5(0)W^0 + f_5(1)W^{4k}$$

$$F_6(k) = f_6(0)W^0 + f_6(1)W^{4k}$$

Thus

$$F_3(0) = f(0) + f(4), \qquad F_4(0) = f(2) + f(6)$$

$$F_3(1) = f(0) - f(4), \qquad F_4(1) = f(2) - f(6)$$

$$F_5(0) = f(1) + f(5), \qquad F_6(0) = f(3) + f(7)$$

$$F_5(1) = f(1) - f(5), \qquad F_6(1) = f(3) - f(7)$$

which can be viewed as four butterfly arrangements with gains of 1.0 and -1.0. The total 8-point FFT can be arranged as in Figure 9-13. Note that the F_1 and F_2 equations can take the form:

$$F_1(0) = F_3(0) + W^0 F_4(0)$$

$$F_1(1) = F_3(1) + W^2 F_4(1), \qquad \text{where } W^0 = 1 \text{ and } W^2 = e^{-j\pi/2} = -j$$

$$F_1(2) = F_3(0) - W^0 F_4(0)$$

$$F_1(3) = F_3(1) - W^2 F_4(1)$$

and

$$F_2(0) = F_5(0) + W^0 F_6(0)$$

$$F_2(1) = F_5(1) + W^2 F_6(1)$$

$$F_2(2) = F_5(0) - W^0 F_6(0)$$

$$F_2(3) = F_5(1) - W^2 F_6(1)$$

And, finally, the F equations take the form:

$$F(0) = F_1(0) + W^0 F_2(0) \qquad W^0 = 1$$

$$F(1) = F_1(1) + W^1 F_2(1) \qquad W^1 = 1/\sqrt{2} - j1/\sqrt{2}$$

$$F(2) = F_1(2) + W^2 F_2(2) \qquad W^2 = -j$$

$$F(3) = F_1(3) + W^3 F_2(3) \qquad W^3 = -1/\sqrt{2} - j1/\sqrt{2}$$

and

$$F(4) = F_1(0) - W^0 F_2(0)$$

$$F(5) = F_1(1) - W^1 F_2(1)$$

$$F(6) = F_1(2) - W^2 F_2(2)$$

$$F(7) = F_1(3) - W^3 F_2(3)$$

The FFT computations start from the given time samples and proceed through r stages, where $2^r = N$. In Example 9-11 with $N = 8$, we had three butterfly stages. A 16-point FFT would require four butterfly stages; a 256-point FFT would need eight butterfly stages, and so on. After completion of each stage of computation, the results can be stored in the registers that held the previous stage's information. The inputs to each butterfly stage are used only once, which saves a lot of space in the computer's memory.

There is an interesting symmetry that occurs in the arrangement of the input data in these decimation-in-time FFT algorithms. Consider the input order for the 8-point FFT of the previous example: $f(0)$, $f(4)$, $f(2)$, $f(6)$, $f(1)$, $f(5)$,

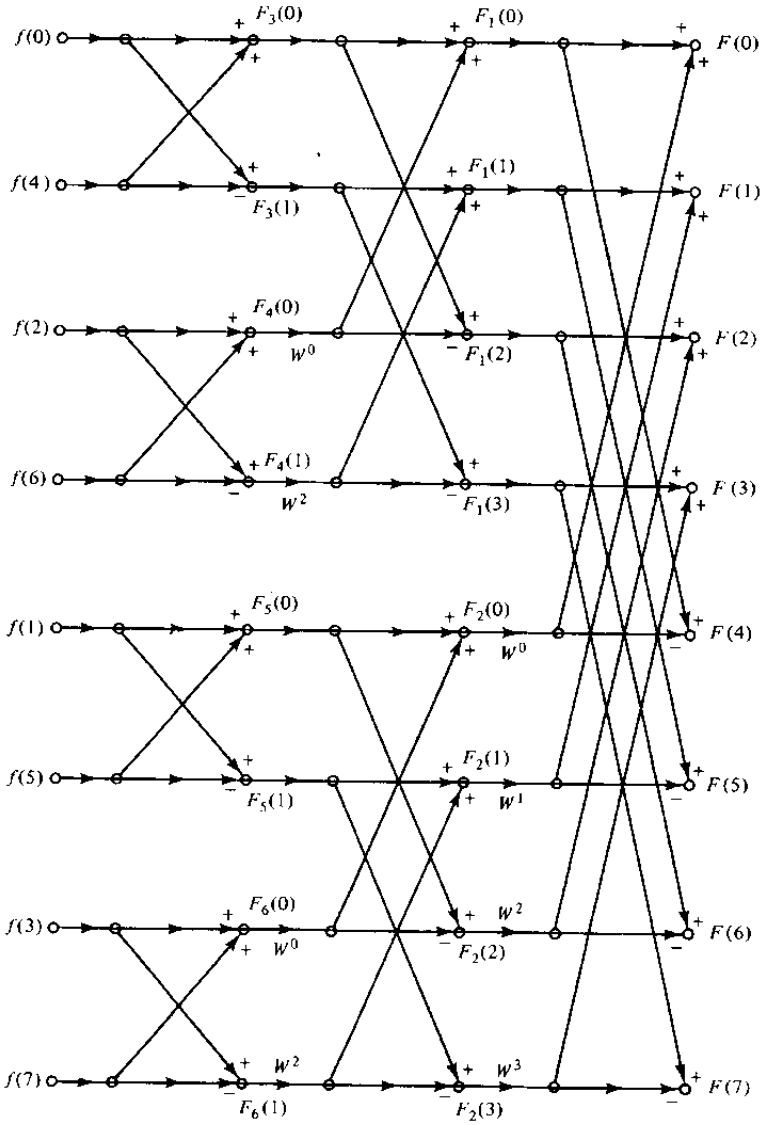


Figure 9-13 Butterfly signal flow graph for 8-point DIT FFT (all paths have unity gain unless otherwise indicated).

$f(3), f(7)$. This is the order in which the first butterfly stage calls for the data. It appears rather jumbled, the numbers in parentheses are not data values but locations of data in the data array. These locations are given addresses in computer memory. If we indicate these addresses in binary form, we can make the following correspondences:

$$f(0) \rightarrow 000$$
$$f(4) \rightarrow 100$$
$$f(2) \rightarrow 010$$
$$f(6) \rightarrow 110$$
$$f(1) \rightarrow 001$$
$$f(5) \rightarrow 101$$
$$f(3) \rightarrow 011$$
$$f(7) \rightarrow 111$$

In a typical signal processing situation, the data are gathered in a natural order, for example, $f(0), f(1), f(2), f(3), f(4), f(5), f(6), f(7)$. For this natural order we can make the following correspondences:

$$f(0) \rightarrow 000$$
$$f(1) \rightarrow 001$$
$$f(2) \rightarrow 010$$
$$f(3) \rightarrow 011$$
$$f(4) \rightarrow 100$$
$$f(5) \rightarrow 101$$
$$f(6) \rightarrow 110$$
$$f(7) \rightarrow 111$$

Now send the numbers in this data array reversed to the addresses indicated by the binary numbers. This technique of rearrangement is called **binary reversal.** After this reversal the data are in the proper order for the FFT processing.

EXAMPLE 9-12

Using the 8-point FFT for $f(n) = 1, n = 0, \ldots, 7$, determine F_3, F_4, F_5, F_6, then F_1, F_2 and, finally, $F(k)$ for $k = 0, \ldots, 7$.

Solution

$$F_3(0) = 2 \quad F_4(0) = 2 \quad F_5(0) = 2 \quad F_6(0) = 2$$
$$F_3(1) = 0 \quad F_4(1) = 0 \quad F_5(1) = 0 \quad F_6(1) = 0$$

then

$$F_1(0) = 4 \text{ and } F_2(0) = 4$$
$$F_1(1) = 0 \qquad F_2(1) = 0$$
$$F_1(2) = 0 \qquad F_2(2) = 0$$
$$F_1(3) = 0 \qquad F_2(3) = 0$$

and

$$F(0) = 8$$
$$F(1) = F(2) = F(3) = F(4) = F(5) = F(6) = F(7) = 0$$

The DFT pair $f(n) \leftrightarrow F(k)$ represents a mapping between N points in the discrete time domain and N points in the discrete frequency domain. But recall that in the DFT development, the actual functions $f(n)$ and $F(k)$ were two periodic discrete sequences. Tracing that development backward, we arrive at the aperiodic discrete time sequence $f(n)$ and the periodic continuous frequency function $F_T(\theta)$. Stepping back further still, through the duality property, we arrive at the periodic continuous time function $f_T(t)$ and the aperiodic discrete frequency function C_n. In light of this trace we can explain the results of Example 9-12 by viewing $f(n)$ as a periodically replicated and continuous $f_T(t)$ which

becomes a constant: $f_E(t) = 1$ for all t. Such a function has a frequency domain c_n as a single function c_0 with all $c_n = 0$ for $n \neq 0$.

EXAMPLE 9-13

Using the 8-point FFT for $f(n) = [1, 1, 1, 1, 0, 0, 0, 0]$, determine $F(k)$ for $k = 0, \ldots, 7$.

Solution

$$F_3(0) = 1 \qquad F_4(0) = 1 \qquad F_5(0) = 1 \qquad F_6(0) = 1$$

$$F_1(0) = 2 \qquad F_2(0) = 2$$

$$F_3(1) = 1 \qquad F_4(1) = 1 \qquad F_5(1) = 1 \qquad F_6(1) = 1$$

$$F_1(2) = 0 \qquad F_2(2) = 0 \qquad F_1(1) = 1 - j \qquad F_2(1) = 1 - j$$

$$W^0 = 1, \qquad W^1 = 1/\sqrt{2} - j1/\sqrt{2}, \qquad W^2 = -j,$$

$$W^3 = -1/\sqrt{2} - j1/\sqrt{2}, \qquad F_1(3) = 1 + j, \qquad F_2(3) = 1 + j$$

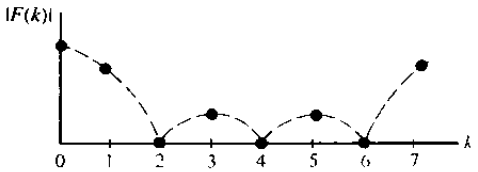
$$F(0) = 4, \qquad F(1) = 1 - j\left(\frac{2}{\sqrt{2}} + 1\right), \qquad F(2) = 0,$$

$$F(3) = 1 + j(1 - 2/\sqrt{2})$$

$$F(4) = 0, \qquad F(5) = 1 + j\left(\frac{2}{\sqrt{2}} - 1\right), \qquad F(6) = 0,$$

$$F(7) = 1 + j(1 + 2/\sqrt{2})$$

Plotting the magnitude of these terms, we get a plot similar to that in the following sketch.



This is the familiar Sin x/x pattern that we had with the Fourier series coefficients when the time function was a pulse. Interestingly, if we determined $F(k)$ for $f(n) = [0, 0, 1, 1, 1, 1, 0, 0]$, we would get the same magnitude plot as shown here, although the phase of the $F(k)$ terms would change. This is explained by the time shift property of the DFT.

9-4-2 The Inverse FFT

If we return to the previous example and imagine that the complex conjugate of the outputs are inputs to an FFT algorithm, we will obtain some interesting

results. We can write F^* as an array:

$$\begin{bmatrix} 4 \\ 1 + j(2/\sqrt{2} + 1) \\ 0 \\ 1 - j(1 - 2/\sqrt{2}) \\ 0 \\ 1 - j(2/\sqrt{2} - 1) \\ 0 \\ 1 - j(1 + 2/\sqrt{2}) \end{bmatrix}$$

Let these function as the $f(n)$ inputs to our 8-point FFT; that is, $f(0) = 4$, $f(1) = 1 + j(2/\sqrt{2} + 1)$, and so on. Tracing through the butterflies in the flow graph of Figure 9-13, we get as outputs the following array:

$$\begin{bmatrix} 8 \\ 8 \\ 8 \\ 8 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Notice that if we divide this array by $N = 8$, we have as outputs of the FFT the array that was previously the inputs; that is, $f(n) = [1, 1, 1, 1, 0, 0, 0, 0]$. In this way the FFT algorithm can function as an inverse FFT (IFFT) algorithm. To be totally general, we would have to take the complex conjugate of the output. But this is seldom necessary since $f(n)$ is typically a real sequence of data points.

To formalize this result, take the complex conjugate of both sides of Equation 9-25

$$f^*(n) = \left(\frac{1}{N}\sum_{k=0}^{N-1} F(k)W^{-kn}\right)^*$$

$$= \frac{1}{N}\sum_{k=0}^{N-1} F^*(k)W^{kn} \tag{9-34}$$

Now change the k to n and the n to k:

$$f^*(k) \qquad \frac{1}{N}\sum_{n=0}^{N-1} F^*(n)W^{kn} \tag{9-35}$$

Compare this summation to the DFT summation for $F(k)$ in Equation 9-24. Except for the N factor, they are the same if $F^*(n)$ replaces $f(n)$.

Then
$$f(k) = \left(\frac{1}{N}\sum_{n=0}^{N-1} F^*(n)W^{kn}\right)^* \tag{9-36}$$

Thus to use the FFT to calculate the IFFT, that is, to determine the time sequence using FFT algorithms:

1. Apply the complex conjugate of the transform array to the FFT input.
2. Multiply the FFT outputs by $1/N$.
3. Take the complex conjugate (if necessary) of the output array.

9-4-3 The Decimation-in-Frequency Algorithm

Let us write $F(k)$ as follows:

$$F(k) = \sum_{n=0}^{N/2-1} f(n)W^{nk} + \sum_{n=N/2}^{N-1} f(n)W^{nk}, \qquad k = 0, 1, \ldots, N-1 \tag{9-37}$$

If we change the index of summation in the second term, we can write:

$$F(k) = \sum_{n=0}^{N/2-1}\left[f(n)W^{nk} + f\left(n+\frac{N}{2}\right)W^{(n+N/2)k}\right]$$

but
$$W^{(N/2)k} = e^{-j\pi k} = (-1)^k \tag{9-38}$$

so we have:

$$F(k) = \sum_{n=0}^{N/2-1}\left(f(n) + f\left(n+\frac{N}{2}\right)\right)W^{nk}, \qquad \text{for } k = 0, 2, 4, 6, \ldots \tag{9-39}$$

$$F(k) = \sum_{n=0}^{N/2-1}\left(f(n) - f\left(n+\frac{N}{2}\right)\right)W^{nk}, \qquad \text{for } k = 1, 3, 5, 7, \ldots \tag{9-40}$$

It is from these equations that the "decimation-in-frequency" idea arises. We note that the frequency function $F(k)$ is broken up into even and odd components. In the "decimation-in-time" FFT considered earlier, recall that the time function $f(n)$ was broken up into even and odd

If we now make the definitions:

$$f_1(n) = f(n) + f\left(n+\frac{N}{2}\right) \tag{9-41}$$

$$f_2(n) = \left[f(n) - f\left(n+\frac{N}{2}\right)\right]W^n \tag{9-42}$$

then we can write:

$$F(2k) = \sum_{n=0}^{N/2-1} f_1(n)W^{2nk} \tag{9-43}$$

$$F(2k+1) = \sum_{n=0}^{N/2-1} f_2(n)W^{2nk} \tag{9-44}$$

both of which are good for $k = 0, 1, \ldots, N/2 - 1$. Then we can write each of these equations as two summations by dividing the interval $n = 0$ to $N/2 - 1$ into two intervals $n = 0$ to $N/4 - 1$ and $n = N/4$ to $N/2 - 1$. We demonstrate this procedure, as before, with an 8-point FFT.

EXAMPLE 9-14

Solution

$$F(2k) = \sum_{n=0}^{3} f_1(n)W^{2nk}, \qquad k = 0, 1, 2, 3$$

$$F(2k+1) = \sum_{n=0}^{3} f_2(n)W^{2nk}, \qquad k = 0, 1, 2, 3$$

Now write $F(2k) = \Sigma_{n=0}^{1} f_1(n)W^{2nk} + \Sigma_{n=2}^{3} f_1(n)W^{2nk}$

and
$$F(2k+1) = \sum_{n=0}^{1} f_2(n)W^{2nk} + \sum_{n=2}^{3} f_2(n)W^{2nk}$$

where
$$f_1(n) = f(n) + f(n+4), \qquad f_2(n) = [f(n) - f(n+4)]W^n$$

Now we can write the second summation in $F(2k)$, $F(2k+1)$ as summations from $n = 0$ to $n = 1$ so that:

$$F(2k) = \sum_{n=0}^{1} f_1(n)W^{2nk} + f_1(n+2)W^{2(n+2)k}, \qquad k = 0, 1, 2, 3$$

$$F(2k+1) = \sum_{n=0}^{1} f_2(n)W^{2nk} + f_2(n+2)W^{2(n+2)k}, \qquad k = 0, 1, 2, 3$$

Now define:

$$f_3(n) = f_1(n) + f_1(n+2)$$
$$f_4(n) = [f_1(n) - f_1(n+2)]W^{2n}$$

and
$$f_5(n) = f_2(n) + f_2(n+2)$$
$$f_6(n) = [f_2(n) - f_2(n+2)]W^{2n}$$

and we can write:

$$F(4k) = \sum_{n=0}^{1} f_3(n)W^{4kn} = f_3(0) + f_3(1)W^{4k}, \qquad k = 0, 1$$

$$F(4k+2) = \sum_{n=0}^{1} f_4(n)W^{4kn} = f_4(0) + f_4(1)W^{4k}, \qquad k = 0, 1$$

$$F(4k+1) = \sum_{n=0}^{1} f_5(n)W^{4kn} = f_5(0) + f_5(1)W^{4k}, \qquad k = 0, 1$$

$$F(4k+3) = \sum_{n=0}^{1} f_6(n)W^{4kn} = f_6(0) + f_6(1)W^{4k}, \qquad k = 0, 1$$

Thus
$$F(0) = f_3(0) + f_3(1)$$
$$F(4) = f_3(0) + f_3(1)W^4, \qquad W^4 = -1$$
$$= f_3(0) - f_3(1)$$
$$F(2) = f_4(0) + f_4(1)$$
$$F(6) = f_4(0) - f_4(1)$$
$$F(1) = f_5(0) + f_5(1)$$
$$F(5) = f_5(0) - f_5(1)$$
$$F(3) = f_6(0) + f_6(1)$$
$$F(7) = f_6(0) - f_6(1)$$

and
$$f_3(0) = f_1(0) + f_1(2), \qquad f_3(1) = f_1(1) + f_1(3)$$
$$f_4(0) = [f_1(0) - f_1(2)]W^0,$$
$$f_4(1) = [f_1(1) - f_1(3)]W^2, \qquad W^2$$
$$f_5(0) = f_2(0) + f_2(2), \qquad f_5(1) = f_2(1) + f_2(3)$$
$$f_6(0) = [f_2(0) - f_2(2)]W^0, \quad f_6(1) = [f_2(1) - f_2(3)]W^1$$

and
$$f_1(0) = f(0) + f(4)$$
$$f_1(1) = f(1) + f(5)$$
$$f_1(2) = f(2) + f(6)$$
$$f_1(3) = f(3) + f(7)$$

and
$$f_2(0) = [f(0) - f(4)]W^0$$
$$f_2(1) = [f(1) - f(5)]W$$
$$f_2(2) = [f(2) - f(6)]W^2$$
$$f_2(3) = [f(3) - f(7)]W^3$$

The total 8-point DIF FFT can be arranged as in Figure 9-14.

It is interesting to compare the DIT and DIF algorithms. The DIF algorithm has an advantage in that the data are input in their natural order, whereas the DIT algorithm requires that the data be put in the proper order for processing by performing a binary reversal. However, the DIF algorithm has the outputs in a jumbled order. These values can also be put in their natural order by binary reversal, if so desired.

EXAMPLE 9-15

Determine the DFT, then the DIF FFT, for the $f(n)$ used in Example 9-14. Compare the results.

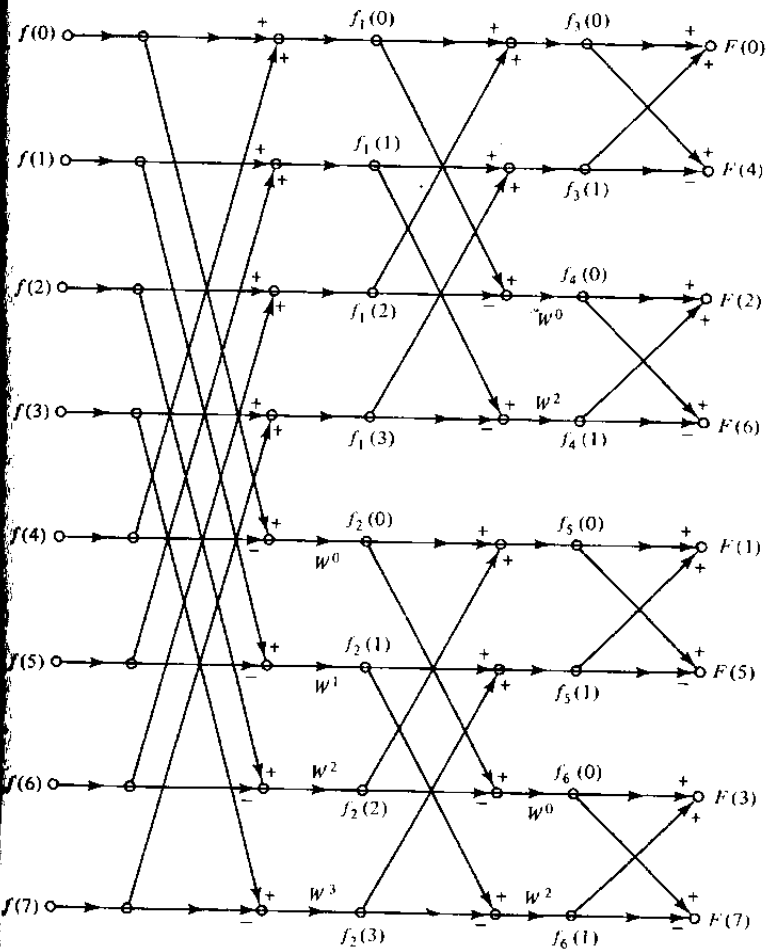


Figure 9-14 Butterfly signal flow graph for 8-point DIF FFT (all paths have unity gain unless otherwise indicated).

Solution

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

but
$$W^{kN} = e^{-j2\pi k} = 1.$$

Therefore the numerator = 0 and $F(k) = 0$. Except when? If $k = 0$, we get:

$$\Sigma \quad \sum_{n=0}^{N-1} W^0 \quad \sum_{n=0}^{N-1} (1)^n \quad N \quad 8$$

Therefore $F(k) \quad [8, 0, 0, 0, 0, 0, 0, 0]$

Now tracing through the DIF FFT algorithm, we obtain:

$$f_1(0) = 2 \quad f_2(0) = 0 \quad f_3(0) = 4 \quad f_5(0) = 0$$
$$f_1(1) = 2 \quad f_2(1) = 0 \quad f_3(1) = 4 \quad f_5(1) = 0$$
$$f_1(2) = 2 \quad f_2(2) = 0 \quad f_4(0) = 0 \quad f_6(0) = 0$$
$$f_1(3) = 2 \quad f_2(3) = 0 \quad f_4(1) = 0 \quad f_6(1) = 0$$

and finally, we have:

$$F(0) = 8$$
$$F(4) = F(2) = F(6) = F(1) = F(5) = F(3) = F(7) = 0$$

Both these approaches yield the same result we obtained with the DIT FFT. The number of computations involved in these approaches is roughly similar. This is verified by the plot in Figure 9-11, which shows that the tremendous advantage of either DIF or DIT FFT algorithms over the direct DFT calculation becomes apparent at larger values of N. The value of $N = 8$ is too small to make much of a difference.

9-5 APPLICATION OF THE FFT

The FFT is typically applied in the modern world of engineering in almost every situation where Fourier transforms are employed. Thus for FFT applications consider the discussion of applications from the previous chapter and use the FFT in place of the continuous time Fourier transform. However, recall that we did not always want to compute Fourier transforms. Often we used Fourier transforms in a theoretical sense to describe concepts involved in the communication theory or in the filter theory. In those discussions the FFT would not be needed. Yet, when it comes to real-world engineering practices, we frequently are required to compute transforms rather than use the transform theory to develop or demonstrate abstract concepts. A knowledge of the basics of the FFT, then, seems more and more essential for today's engineering work.

One place where the FFT is useful is in digital filter design. We know that the output of a digital filter is expressed by the convolution summation:

$$y(n) = \sum_{i=-\infty}^{\infty} x(i)h(n-i) \tag{9-45}$$

where $x(n)$ is the input and $h(n)$ is the unit pulse response. If $x(n)$ and $h(n)$ are finite data sequences, nonzero from $n = 0$ to $N - 1$, then we can write:

$$y(n) = \sum_{i=0}^{N-1} x(i)h(n-i) \tag{9-46}$$

To implement this digital filter as a software package, we could simply calculate $y(n)$ from Equation 9-46. The FFT, however, would permit us to save a lot of computer time by computing $X(k) = \text{FFT}[x(n)]$ and $H(k) = \text{FFT}[h(n)]$. Then

$Y(k) = X(k)H(k)$ for $k = 0, \ldots, N - 1$ and $y(n) = \text{IDFT}[Y(k)]$, where the inverse discrete Fourier transform can also be computed with the FFT algorithm.

There is another useful procedure here that will save more computer time. Often the input sequence will be rather long, requiring a delay in real-time system processing, because all the $x(n)$ sequence is needed before evaluation of $y(n)$ from Equation 9-46 can be initiated. To shorten this delay, we can decompose the $x(n)$ sequence into a number of shorter segments, each of which can be processed individually with $y(n)$ then becoming a summation of partial convolutions. This idea is based on superposition. Instead of having a single input $x(n)$, we have a sum of inputs. Let $x(n) = x_1(n) + x_2(n) + \cdots + x_m(n)$. We can start the processing involved in Equation 9-46 by using $x_1(n)$ which is the data record gathered first. As this processing proceeds, the second data record $x_2(n)$ can be obtained. Then use $x_2(n)$ in Equation 9-46 as $x_3(n)$ is being collected, and so on.

In addition to its use in fast convolution, the FFT is also useful for fast correlation. Since the correlation operation is structurally similar to convolution, the sequences involved are often segmented into shorter sequences with which partial correlations are performed and the partial results are combined into total correlation functions. Correlation and autocorrelation techniques can be used for system identification. For a system with input $x(n)$, output $y(n)$, and impulse response $h(n)$, we can of course compute $X(k)$, $Y(k)$, and $H(k)$ using the FFT. If we have the input and output available, but not the system description, that is, $h(n)$ or $H(k)$, then we can write:

$$Y(k) = H(k)X(k) \tag{9-47}$$

$$Y(k)X^*(k) = H(k)X(k)X^*(k) \tag{9-48}$$
$$= H(k)|X(k)|^2$$

or
$$H(k) = Y(k)X^*(k)/|X(k)|^2 \tag{9-49}$$

The numerator is the DFT of the correlation of the input and the output. The denominator is the DFT of the autocorrelation of the input. Both these DFTs can be computed with FFT algorithms. Then the inverse FFT can be used to determine $h(n)$, that is, to identify the system.

Finally, spectral analysis is a popular area of FFT employment. The basic idea is to transform a time function to get its Fourier spectrum. The time function is discretized and the DFT or FFT is used to obtain the spectrum. Most of this work presupposes an extensive background in probability and random variable theory and is beyond our present scope. Some general comments: To obtain spectral estimates from a finite set of measurements typically employs the power spectral density function which is just the Fourier transform of the autocorrelation function of the data time sequence. The data sequences, though, are typically random variables. One interesting problem in this area comes from the fact that the determination of the power spectral density requires complete knowledge of the autocorrelation function, a function of generally infinite extent. Since we only have a finite data sequence available, we must make some

estimates of the unavailable data. For instance, the autocorrelation function $r(n)$ is defined for all n, $-\infty < n < \infty$. However, in view of the finite nature of our data, we can only calculate $r(n)$ for $-k \leq n \leq< k$. The question is: How can we make reasonable estimates of the $r(n)$ values outside the range $-k \leq n \leq k$? A variety of spectral estimation techniques have been developed to deal with this problem. The interested reader is encouraged to consult the literature.

SUMMARY

In this chapter we have developed the discrete Fourier transform (DFT) and the inverse discrete Fourier transform (IDFT). We have shown the relationship of the discrete time Fourier transform to the Z transform. A number of examples illustrated this theory.

We considered the properties of the DFT, many of which were similar to the continuous Fourier transform properties. In particular, the convolution property deserved some extra attention due to the distinction that needed to be made between the periodic and aperiodic convolution.

Then we derived the fast Fourier transform (FFT) algorithms: The decimation-in-time (DIT) and the decimation-in-frequency (DIF) fast Fourier transforms. The similarities and the differences between these two approaches were indicated and the inverse FFT (IFFT) was developed. A few examples were worked to demonstrate the theory.

Finally, some FFT applications were discussed. These discussions were rather general in nature because many FFT applications are merely Fourier transform applications in which Fourier transforms are numerically calculated.

PROBLEMS

9-1. Let $f(n) = \text{Cos}\,(\pi/8\, n) + \text{Cos}\,(\pi/4\, n)$. Using the 8-point DIT FFT, determine the DFT of $f(n)$: $F(k)$. Use samples of the $f(n)$ from $n = 0, \ldots, 7$.

9-2. Prove Parseval's theorem for the DFT, that is:

$$\sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X(k)|^2$$

9-3. Prove the symmetry property for the DFT, that is:

$$\frac{1}{N} X(n) \leftrightarrow x(-k)$$

9-4. Find the discrete time Fourier transform for:
(a) $f(n) = u(n) - u(n-3) + \delta(n-4)$
(b) $f(n) = (\frac{1}{3})^n u(n)$
(c) $f(n) = (\frac{1}{3})^n \text{Cos}\, n\pi u(n)$
(d) $f(n) \quad (\frac{1}{3})^n u(n) \quad (\frac{1}{3})^n u(n \quad 1)$

9-5. Determine the Z transform for the functions in Problem 9.4. Then, using the Z transforms, find the discrete time Fourier transforms.

9-6. Determine the 8-point DFT for the sequences:
(a) $f(n) = [1, 0, 0, 1, 0, 0, 1, 0]$
(b) $f(n) = [1, 2, 3, 0, 0, 0, 0, 0]$
(c) $f(n) = [0, 0, 1, 1, 2, 2, 3, 3]$
(d) $f(n) = [10, 20, 10, 20, 10, 20, 10, 20]$

9-7. Determine the 8-point IDFT for:
(a) $F(k) = 1 - \text{Cos}\, \pi k/2$
(b) $F(k) = \text{Cos}\, \pi k + \text{Cos}\, \pi k/3$
(c) $F(k) = (k-1)(\frac{1}{2})^k$

9-8. Construct the butterfly signal flow graph for the 16-point DIF FFT and the 16-point DIT FFT.

9-9. Compute the 8-point FFT (DIF or DIT) for:
(a) $f(n) = (\frac{1}{2})^n u(n)$
(b) $f(n) = u(n+1) - u(n-1)$
(c) $f(n) = u(n)\, \text{Cos}\, n\pi/8$
(d) $f(n) = u(n) - u(n-2)(\frac{1}{2})^n$

9-10. Consider the function $x(n) = (\frac{1}{3})^n u(n)$. Let $N = 6$.
(a) Determine the even and odd parts of $x(n)$: $x_e(n)$ and $x_0(n)$.
(b) Show that $X_e(k) = \text{Re}[X(k)]$ and $X_0(k) = j\text{Im}[X(k)]$.

9-11. A certain time sequence $x(n)$ has a DFT:

$$X(k) = [1, 1-j, 0, 0, 0, 1+j, j, -1], \qquad N = 8$$

Determine $x(n)$ and also the IDFT of $X(k-4)$.

9-12. Let:

$$x(n) = [1, 0, 0, 1, j, -j, 1, 0]$$

Show that $x^*(n) \leftrightarrow X^*(-k)$.

9-13. Demonstrate Parseval's theorem for $x(n) \leftrightarrow X(k)$

where

$$x(n) = [8, 10, 4, 5], \qquad N = 4.$$

9-14. Using the symmetry property and knowing $x(n) \leftrightarrow X(k)$

where $x(n) = [\alpha_1, \alpha_2, \alpha_3, \alpha_4]$ and $X(k) = [\beta_1, \beta_2, \beta_3, \beta_4]$ and $N = 4$

Determine the DFT of $y(n) = [\beta_1, \beta_2, \beta_3, \beta_4]$.

9-15. Determine the "coordinate normalized Fourier transforms" $F_E(\theta)$ for the following time sequence:
(a) $f(n) = [1, 2, 3, 4, 3, 2, 1]$
(b) $f(n) = (\frac{1}{2})^n u(n) + 2^n u(-n)$
(c) $f(n) = \delta(n-5) + \delta(n+5) + 2\delta(n)$

9-16. Determine $f(n)$ using Equation 9-22 when $N = 4$ and
(a) $F(k) = [1, \quad 1, 0, 1]$
(b) $F(k) = [0, 0, 1, \quad 1]$
(c) $F(k) \quad [\quad 1, 0, 0, 2]$

9-17. Determine the DFT of the Hamming window that is described by the equation $w(n) = 0.54 - 0.46 \text{ Cos } [\pi(n + 0.5)/4]$. Let $N = 8$. Compare these results to the DFT for a corresponding rectangular window.

9-18. From Equation 9-25 we have:

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

and from Equation 9-36 we can write:

$$f(n) = \left(\frac{1}{N} \sum_{k=0}^{N-1} F^*(k) W^{kn} \right)^*$$

(a) Show that these two results are the same.
(b) Explain why the second formulation is more appropriate than the first for IFFT computations.

9-19. Let:

$$f(n) = [1, 1, 1, 1, 0, 0, 0, 0], \qquad N = 8$$

We know that $|F(\theta)| = \alpha_1 \text{ Cos } \alpha_2\theta \text{ Cos } \alpha_3\theta$. Determine α_1, α_2, and α_3.

9-20. Demonstrate the linearity property of the DFT by determining $F(k)$ when

$$f(n) = 8(\tfrac{1}{2})^n + 12(\tfrac{1}{4})^n, \qquad N = 4, n = 0, \ldots, 3$$

9-21. Let:

$$x(n) = [1, 1, 0, 1] \quad \text{and} \quad y(n) = [0, 0, 1, 1], \qquad N = 4$$

Determine $X(k)$, $Y(k)$, and show that $z(n) = x(n)\ y(n) = [0, 0, 0, 1] \cdot\cdot X(k)*Y(k)$.

Chapter 10

State Variable Theory

INTRODUCTION

This final chapter on state variables provides an integration of much of the material in the text. State variable equations are solved with time-domain and transform methods. Our knowledge of differential and difference equation solutions comes into play, and we use the Laplace and Z transforms as well. We will consider the application of state variable ideas to some control system problems. State variable theory is one of the most important areas of systems theory and in control theory we have one of the most important applications of state variable theory.

State variable representations of linear systems are time-domain descriptions. They have certain advantages over the familiar difference or differential equation representations studied in Chapter 2. The state variable representation offers a concise and precise notation that lends itself well to digital computer processing. Multiple-input–multiple-output (MIMO) systems are easily managed within the state variable framework. In addition, extensions to time-varying and nonlinear systems—although not considered in this text—can be readily effected through the state variable representations.

The everyday idea of state is well-known: "the state of the Union," "the state of the art," "the state of one's health," and so on. In this sense, the concept of state refers to the conditions or attributes or circumstances of a person or thing. Within the linear systems world the idea of *state* is much more precise. The **state** of a dynamic system is the smallest set of numbers called **state variables**—which if specified at some initial time t_0 or k_0 can be used to predict the system's behavior for all $t \cdot t_0$ or $k \cdot k_0$, provided that the input to the system