described in Sec-

results in Exam-

ϕ) with $a = 0.8$,

CHAPTER

8

Efficient Computation of the DFT: Fast Fourier Transform Algorithms

As we have observed in the preceding chapter, the discrete fourier transform (DFT) plays an important role in many applications of digital signal processing, including linear filtering, correlation analysis, and spectrum analysis. A major reason for its importance is the existence of efficient algorithms for computing the DFT.

The main topic of this chapter is the description of computationally efficient algorithms for evaluating the DFT. Two different approaches are described. One is a divide-and-conquer approach in which a DFT of size N, where N is a composite number, is reduced to the computation of smaller DFTs from which the larger DFT is computed. In particular, we present important computational algorithms, called fast Fourier transform (FFT) algorithms, for computing the DFT when the size N is a power of 2 and when it is a power of 4.

The second approach is based on the formulation of the DFT as a linear filtering operation on the data. This approach leads to two algorithms, the Goertzel algorithm and the chirp-z transform algorithm, for computing the DFT via linear filtering of the data sequence.

8.1 Efficient Computation of the DFT: FFT Algorithms

In this section we present several methods for computing the DFT efficiently. In view of the importance of the DFT in various digital signal processing applications, such as linear filtering, correlation analysis, and spectrum analysis, its efficient computation is a topic that has received considerable attention by many mathematicians, engineers, and applied scientists.

Basically, the computational problem for the DFT is to compute the sequence $\{X(k)\}$ of N complex-valued numbers given another sequence of data $\{x(n)\}$ of length

N, according to the formula

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \qquad 0 \le k \le N-1 \tag{8.1.1}$$

where

$$W_N = e^{-j2\pi/N} \tag{8.1.2}$$

In general, the data sequence $x(n)$ is also assumed to be complex valued.

Similarly, the IDFT becomes

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-nk}, \qquad 0 \le n \le N-1 \tag{8.1.3}$$

Since the DFT and IDFT involve basically the same type of computations, our discussion of efficient computational algorithms for the DFT applies as well to the efficient computation of the IDFT.

We observe that for each value of k, direct computation of $X(k)$ involves N complex multiplications ($4N$ real multiplications) and $N-1$ complex additions ($4N-2$ real additions). Consequently, to compute all N values of the DFT requires N^2 complex multiplications and $N^2 - N$ complex additions.

Direct computation of the DFT is basically inefficient, primarily because it does not exploit the symmetry and periodicity properties of the phase factor W_N. In particular, these two properties are:

$$\text{Symmetry property:} \qquad W_N^{k+N/2} = -W_N^k \tag{8.1.4}$$

$$\text{Periodicity property:} \qquad W_N^{k+N} = W_N^k \tag{8.1.5}$$

The computationally efficient algorithms described in this section, known collectively as fast Fourier transform (FFT) algorithms, exploit these two basic properties of the phase factor.

8.1.1 Direct Computation of the DFT

For a complex-valued sequence $x(n)$ of N points, the DFT may be expressed as

$$X_R(k) = \sum_{n=0}^{N-1} \left[x_R(n) \cos \frac{2\pi kn}{N} + x_I(n) \sin \frac{2\pi kn}{N} \right] \tag{8.1.6}$$

$$X_I(k) = -\sum_{n=0}^{N-1} \left[x_R(n) \sin \frac{2\pi kn}{N} - x_I(n) \cos \frac{2\pi kn}{N} \right] \tag{8.1.7}$$

The direct computation of (8.1.6) and (8.1.7) requires:

1. $2N^2$ evaluations of trigonometric functions.
2. $4N^2$ real multiplications.
3. $4N(N-1)$ real additions.
4. A number of indexing and addressing operations.

These operations are typical of DFT computational algorithms. The operations in items 2 and 3 result in the DFT values $X_R(k)$ and $X_I(k)$. The indexing and addressing operations are necessary to fetch the data $x(n)$, $0 \le n \le N-1$, and the phase factors and to store the results. The variety of DFT algorithms optimize each of these computational processes in a different way.

8.1.2 Divide-and-Conquer Approach to Computation of the DFT

The development of computationally efficient algorithms for the DFT is made possible if we adopt a divide-and-conquer approach. This approach is based on the decomposition of an N-point DFT into successively smaller DFTs. This basic approach leads to a family of computationally efficient algorithms known collectively as FFT algorithms.

To illustrate the basic notions, let us consider the computation of an N-point DFT, where N can be factored as a product of two integers, that is,

$$N = LM \tag{8.1.8}$$

The assumption that N is not a prime number is not restrictive, since we can pad any sequence with zeros to ensure a factorization of the form (8.1.8).

Now the sequence $x(n)$, $0 \le n \le N-1$, can be stored either in a one-dimensional array indexed by n or as a two-dimensional array indexed by l and m, where $0 \le l \le L-1$ and $0 \le m \le M-1$ as illustrated in Fig. 8.1.1. Note that l is the row index and m is the column index. Thus, the sequence $x(n)$ can be stored in a rectangular

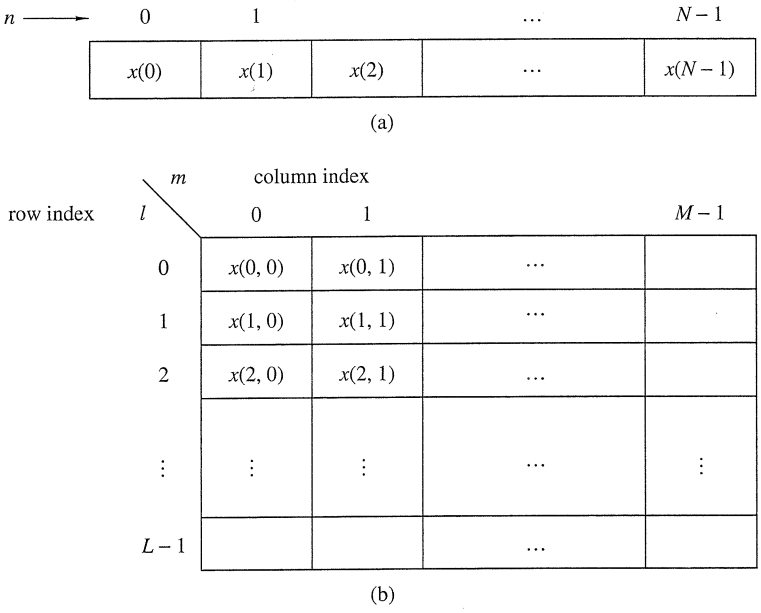

Figure 8.1.1 Two dimensional data array for storing the sequence $x(n)$, $0 \le n \le N-1$.

array in a variety of ways, each of which depends on the mapping of index n to the indexes (l, m).

For example, suppose that we select the mapping

$$n = Ml + m \tag{8.1.9}$$

This leads to an arrangement in which the first row consists of the first M elements of $x(n)$, the second row consists of the next M elements of $x(n)$, and so on, as illustrated in Fig. 8.1.2(a). On the other hand, the mapping

$$n = l + mL \tag{8.1.10}$$

stores the first L elements of $x(n)$ in the first column, the next L elements in the second column, and so on, as illustrated in Fig. 8.1.2(b).

Row-wise $n = Ml + m$

l \ m	0	1	2	...	$M-1$
0	$x(0)$	$x(1)$	$x(2)$	...	$x(M-1)$
1	$x(M)$	$x(M+1)$	$x(M+2)$	...	$x(2M-1)$
2	$x(2M)$	$x(2M+1)$	$x(2M+2)$	...	$x(3M-1)$
	⋮	⋮	⋮	⋯	⋮
$L-1$	$x((L-1)M)$	$x((L-1)M+1)$	$x((L-1)M+2)$	...	$x(LM-1)$

(a)

Column-wise $n = l + mL$

l \ m	0	1	2	...	$M-1$
0	$x(0)$	$x(L)$	$x(2L)$	...	$x((M-1)L)$
1	$x(1)$	$x(L+1)$	$x(2L+1)$	⋯	$x((M-1)L+1)$
2	$x(2)$	$x(L+2)$	$x(2L+2)$	...	$x((M-1)L+2)$
	⋮	⋮	⋮	⋯	⋮
$L-1$	$x(L-1)$	$x(2L-1)$	$x(3L-1)$	⋯	$x(LM-1)$

(b)

Figure 8.1.2 Two arrangements for the data arrays.

A similar arrangement can be used to store the computed DFT values. In particular, the mapping is from the index k to a pair of indices (p, q), where $0 \le p \le L-1$ and $0 \le q \le M-1$. If we select the mapping

$$k = Mp + q \tag{8.1.11}$$

the DFT is stored on a row-wise basis, where the first row contains the first M elements of the DFT $X(k)$, the second row contains the next set of M elements, and so on. On the other hand, the mapping

$$k = qL + p \tag{8.1.12}$$

results in a column-wise storage of $X(k)$, where the first L elements are stored in the first column, the second set of L elements are stored in the second column, and so on.

Now suppose that $x(n)$ is mapped into the rectangular array $x(l, m)$ and $X(k)$ is mapped into a corresponding rectangular array $X(p, q)$. Then the DFT can be expressed as a double sum over the elements of the rectangular array multiplied by the corresponding phase factors. To be specific, let us adopt a column-wise mapping for $x(n)$ given by (8.1.10) and the row-wise mapping for the DFT given by (8.1.11). Then

$$X(p, q) = \sum_{m=0}^{M-1} \sum_{l=0}^{L-1} x(l, m) W_N^{(Mp+q)(mL+l)} \tag{8.1.13}$$

But

$$W_N^{(Mp+q)(mL+l)} = W_N^{MLmp} W_N^{mLq} W_N^{Mpl} W_N^{lq} \tag{8.1.14}$$

However, $W_N^{Nmp} = 1$, $W_N^{mqL} = W_{N/L}^{mq} = W_M^{mq}$, and $W_N^{Mpl} = W_{N/M}^{pl} = W_L^{pl}$.

With these simplications, (8.1.13) can be expressed as

$$X(p, q) = \sum_{l=0}^{L-1} \left\{ W_N^{lq} \left[\sum_{m=0}^{M-1} x(l, m) W_M^{mq} \right] \right\} W_L^{lp} \tag{8.1.15}$$

The expression in (8.1.15) involves the computation of DFTs of length M and length L. To elaborate, let us subdivide the computation into three steps:

1. First, we compute the M-point DFTs

$$F(l, q) \equiv \sum_{m=0}^{M-1} x(l, m) W_M^{mq}, \qquad 0 \le q \le M-1 \tag{8.1.16}$$

for each of the rows $l = 0, 1, \ldots, L-1$.

2. Second, we compute a new rectangular array $G(l, q)$ defined as

$$G(l, q) = W_N^{lq} F(l, q), \qquad \begin{matrix} 0 \le l \le L-1 \\ 0 \le q \le M-1 \end{matrix} \tag{8.1.17}$$

3. Finally, we compute the L-point DFTs

$$X(p,q) = \sum_{l=0}^{L-1} G(l,q) W_L^{lp} \tag{8.1.18}$$

for each column $q = 0, 1, \ldots, M-1$, of the array $G(l,q)$.

On the surface it may appear that the computational procedure outlined above is more complex than the direct computation of the DFT. However, let us evaluate the computational complexity of (8.1.15). The first step involves the computation of L DFTs, each of M points. Hence this step requires LM^2 complex multiplications and $LM(M-1)$ complex additions. The second step requires LM complex multiplications. Finally, the third step in the computation requires ML^2 complex multiplications and $ML(L-1)$ complex additions. Therefore, the computational complexity is

$$\begin{array}{ll} \text{Complex multiplications:} & N(M+L+1) \\ \text{Complex additions:} & N(M+L-2) \end{array} \tag{8.1.19}$$

where $N = ML$. Thus the number of multiplications has been reduced from N^2 to $N(M+L+1)$ and the number of additions has been reduced from $N(N-1)$ to $N(M+L-2)$.

For example, suppose that $N = 1000$ and we select $L = 2$ and $M = 500$. Then, instead of having to perform 10^6 complex multiplications via direct computation of the DFT, this approach leads to 503,000 complex multiplications. This represents a reduction by approximately a factor of 2. The number of additions is also reduced by about a factor of 2.

When N is a highly composite number, that is, N can be factored into a product of prime numbers of the form

$$N = r_1 r_2 \cdots r_v \tag{8.1.20}$$

then the decomposition above can be repeated $(v-1)$ more times. This procedure results in smaller DFTs, which, in turn, leads to a more efficient computational algorithm.

In effect, the first segmentation of the sequence $x(n)$ into a rectangular array of M columns with L elements in each column resulted in DFTs of sizes L and M. Further decomposition of the data in effect involves the segmentation of each row (or column) into smaller rectangular arrays which result in smaller DFTs. This procedure terminates when N is factored into its prime factors.

EXAMPLE 8.1.1

To illustrate this computational procedure, let us consider the computation of an $N = 15$ point DFT. Since $N = 5 \times 3 = 15$, we select $L = 5$ and $M = 3$. In other words, we store the 15-point sequence $x(n)$ column-wise as follows:

$$\begin{array}{lccc} \text{Row 1:} & x(0,0) = x(0) & x(0,1) = x(5) & x(0,2) = x(10) \\ \text{Row 2:} & x(1,0) = x(1) & x(1,1) = x(6) & x(1,2) = x(11) \\ \text{Row 3:} & x(2,0) = x(2) & x(2,1) = x(7) & x(2,2) = x(12) \\ \text{Row 4:} & x(3,0) = x(3) & x(3,1) = x(8) & x(3,2) = x(13) \\ \text{Row 5:} & x(4,0) = x(4) & x(4,1) = x(9) & x(4,2) = x(14) \end{array}$$

Now, we compute the three-point DFTs for each of the five rows. This leads to the following 5×3 array:

$$\begin{array}{lll} F(0,0) & F(0,1) & F(0,2) \\ F(1,0) & F(1,1) & F(1,2) \\ F(2,0) & F(2,1) & F(2,2) \\ F(3,0) & F(3,1) & F(3,2) \\ F(4,0) & F(4,1) & F(4,2) \end{array}$$

The next step is to multiply each of the terms $F(l, q)$ by the phase factors $W_N^{lq} = W_{15}^{lq}$, $0 \le l \le 4$ and $0 \le q \le 2$. This computation results in the 5×3 array:

Column 1	Column 2	Column 3
$G(0,0)$	$G(0,1)$	$G(0,2)$
$G(1,0)$	$G(1,1)$	$G(1,2)$
$G(2,0)$	$G(2,1)$	$G(2,2)$
$G(3,0)$	$G(3,1)$	$G(3,2)$
$G(4,0)$	$G(4,1)$	$G(4,2)$

The final step is to compute the five-point DFTs for each of the three columns. This computation yields the desired values of the DFT in the form

$$\begin{array}{lll} X(0,0) = X(0) & X(0,1) = X(1) & X(0,2) = X(2) \\ X(1,0) = X(3) & X(1,1) = X(4) & X(1,2) = X(5) \\ X(2,0) = X(6) & X(2,1) = X(7) & X(2,2) = X(8) \\ X(3,0) = X(9) & X(3,1) = X(10) & X(3,2) = X(11) \\ X(4,0) = X(12) & X(4,1) = X(13) & X(4,2) = X(14) \end{array}$$

Figure 8.1.3 illustrates the steps in the computation.

It is interesting to view the segmented data sequence and the resulting DFT in terms of one-dimensional arrays. When the input sequence $x(n)$ and the output DFT $X(k)$ in the two-dimensional arrays are read across from row 1 through row 5, we obtain the following sequences:

INPUT ARRAY

$$x(0)\ x(5)\ x(10)\ x(1)\ x(6)\ x(11)\ x(2)\ x(7)\ x(12)\ x(3)\ x(8)\ x(13)\ x(4)\ x(9)\ x(14)$$

OUTPUT ARRAY

$$X(0)\ X(1)\ X(2)\ X(3)\ X(4)\ X(5)\ X(6)\ X(7)\ X(8)\ X(9)\ X(10)\ X(11)\ X(12)\ X(13)\ X(14)$$

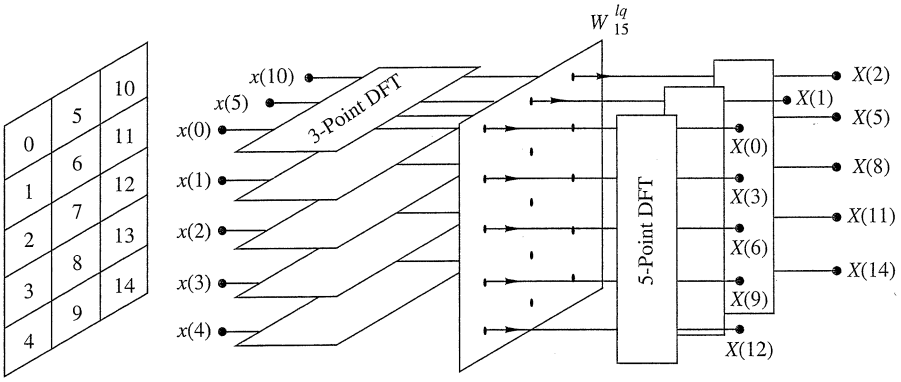

Figure 8.1.3 Computation of $N = 15$-point DFT by means of 3-point and 5-point DFTs.

We observe that the input data sequence is shuffled from the normal order in the computation of the DFT. On the other hand, the output sequence occurs in normal order. In this case the rearrangement of the input data array is due to the segmentation of the one-dimensional array into a rectangular array and the order in which the DFTs are computed. This shuffling of either the input data sequence or the output DFT sequence is a characteristic of most FFT algorithms.

To summarize, the algorithm that we have introduced involves the following computations:

Algorithm 1

1. Store the signal column-wise.
2. Compute the M-point DFT of each row.
3. Multiply the resulting array by the phase factors W_N^{lq}.
4. Compute the L-point DFT of each column
5. Read the resulting array row-wise.

An additional algorithm with a similar computational structure can be obtained if the input signal is stored row-wise and the resulting transformation is column-wise. In this case we select

$$\begin{aligned} n &= Ml + m \\ k &= qL + p \end{aligned} \tag{8.1.21}$$

This choice of indices leads to the formula for the DFT in the form

$$\begin{aligned} X(p,q) &= \sum_{m=0}^{M-1} \sum_{l=0}^{L-1} x(l,m) W_N^{pm} W_L^{pl} W_M^{qm} \\ &= \sum_{m=0}^{M-1} W_M^{mq} \left[\sum_{l=0}^{L-1} x(l,m) W_L^{lp} \right] W_N^{mp} \end{aligned} \tag{8.1.22}$$

Thus we obtain a second algorithm.

Algorithm 2

1. Store the signal row-wise.
2. Compute the L-point DFT at each column.
3. Multiply the resulting array by the factors W_N^{pm}.
4. Compute the M-point DFT of each row.
5. Read the resulting array column-wise.