Ivan W. Selesnick et al. "The Discrete Fourier Transform"
The Transform and Data Compression Handbook
Ed. K. R. Rao et al.
Boca Raton, CRC Press LLC, 2001

Chapter 2

The Discrete Fourier Transform

Ivan W. Selesnick
Polytechnic University

Gerald Schuller
Bell Labs

2.1 Introduction

The discrete Fourier transform (DFT) is a fundamental transform in digital signal processing, with applications in frequency analysis, fast convolution, image processing, etc. Moreover, fast algorithms exist that make it possible to compute the DFT very efficiently. The algorithms for the efficient computation of the DFT are collectively called fast Fourier transforms (FFTs). The historic paper by Cooley and Tukey [15] made well known an FFT of complexity $N \log_2 N$, where N is the length of the data vector. A sequence of early papers [3, 11, 13, 14, 15] still serves as a good reference for the DFT and FFT. In addition to texts on digital signal processing, a number of books devote special attention to the DFT and FFT [4, 7, 10, 20, 28, 33, 36, 39, 48].

The importance of Fourier analysis in general is put forth very well by Leon Cohen [12]:

> . . . Bunsen and Kirchhoff, observed (around 1865) that light spectra can be used for recognition, detection, and classification of substances because they are unique to each substance.
>
> This idea, along with its extension to other waveforms and the invention of the tools needed to carry out spectral decomposition, certainly ranks as one of the most important discoveries in the history of mankind.

The kth DFT coefficient of a length N sequence $\{x(n)\}$ is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn}, \qquad k = 0, \ldots, N-1 \tag{2.1}$$

where

$$W_N = e^{-j2\pi/N} = \cos\left(\frac{2\pi}{N}\right) - j\ \sin\left(\frac{2\pi}{N}\right)$$

is the principal N-th root of unity. Because W_N^{nk} as a function of k has a period of N, the DFT coefficients $\{X(k)\}$ are periodic with period N when k is taken outside the range $k = 0, \ldots, N-1$. The original sequence $\{x(n)\}$ can be retrieved by the inverse discrete Fourier transform (IDFT)

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)\, W_N^{-kn}, \qquad n = 0, \ldots, N-1\ .$$

The inverse DFT can be verified by using a simple observation regarding the principal N-th root of unity W_N. Namely,

$$\sum_{n=0}^{N-1} W_N^{nk} = N \cdot \delta(k), \qquad k = 0, \ldots, N-1\ ,$$

where $\delta(k)$ is the Kronecker delta function defined as

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0\ . \end{cases}$$

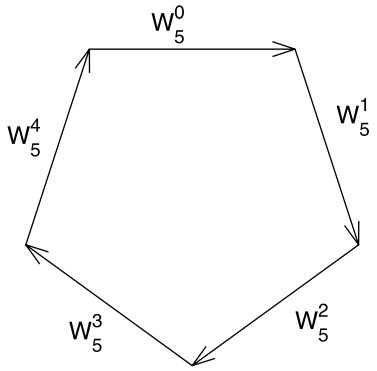
For example, with $N = 5$ and $k = 0$, the sum gives

$$1 + 1 + 1 + 1 + 1 = 5\ .$$

For $k = 1$, the sum gives

$$1 + W_5 + W_5^2 + W_5^3 + W_5^4 = 0$$

which can be graphically illustrated as:



The sums can also be visualized by looking at the illustration of the DFT matrix in Fig. 2.1. Because W_N^{nk} as a function of k is periodic with period N, we can write

$$\sum_{n=0}^{N-1} W_N^{nk} = N \cdot \delta\left(\langle k \rangle_N\right)$$

where $\langle k \rangle_N$ denotes the remainder when k is divided by N, i.e., $\langle k \rangle_N$ is k modulo N.

To verify the inversion formula, we can substitute the DFT into the expression for the IDFT:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} \left(\sum_{l=0}^{N-1} x(l)\, W_N^{kl} \right) W_N^{-kn} \,, \tag{2.2}$$

$$= \frac{1}{N} \sum_{l=0}^{N-1} x(l) \sum_{k=0}^{N-1} W_N^{k(n-l)} \,, \tag{2.3}$$

$$= \frac{1}{N} \sum_{l=0}^{N-1} x(l)\, N\, \delta\left(\langle n-l \rangle_N\right) \,, \tag{2.4}$$

$$= x(n) \,. \tag{2.5}$$

2.2 The DFT Matrix

The DFT of a length N sequence $\{x(n)\}$ can be represented as a matrix-vector product. For example, a length 5 DFT can be represented as

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 & W^4 \\ 1 & W^2 & W^4 & W^6 & W^8 \\ 1 & W^3 & W^6 & W^9 & W^{12} \\ 1 & W^4 & W^8 & W^{12} & W^{16} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \end{bmatrix}$$

where $W = W_5$, or as

$$\mathbf{X} = \mathbf{F}_N \cdot \mathbf{x} \,,$$

where $\mathbf{F}_N$ is the $N \times N$ DFT matrix whose elements are given by

$$(\mathbf{F}_N)_{l,m} = W_N^{lm} \qquad 0 \le l, m \le N-1 \,.$$

As the IDFT and DFT formulae are very similar, the IDFT represented as a matrix is closely related to $\mathbf{F}_N$,

$$\mathbf{F}_N^{-1} = \frac{1}{N} \mathbf{F}_N^*$$

where $\mathbf{F}_N^*$ represents the complex conjugate of $\mathbf{F}_N$.

It is very useful to illustrate the entries of the matrix $\mathbf{F}_N$ as in Fig. 2.1, where each complex value is shown as a vector. In Fig. 2.1, it can be seen that in the kth row of the matrix the elements consist of a vector rotating clockwise with a constant increment of $2\pi k/N$. In the first row $k = 0$ and the vector rotates in increments of 0. In the second row $k = 1$ and the vector rotates in increments of $2\pi/N$.

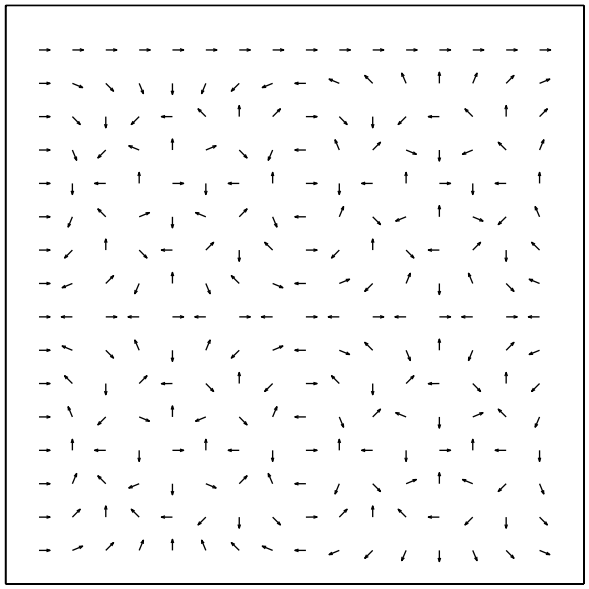


FIGURE 2.1
The 16-point DFT matrix.

2.3 An Example

The DFT is especially useful for efficiently representing signals that are comprised of a few frequency components. For example, the length 2048 signal shown in Fig. 2.2 is an electrocardiogram (ECG) recording from a dog[1]. The DFT of this real signal, shown in Fig. 2.2, is greatest at specific frequencies corresponding to the fundamental frequency and its harmonics. Clearly, the signal $\{x(n)\}$ can be represented well even when many of the small DFT $\{X(k)\}$ coefficients are set to zero. By discarding, or coarsely quantizing, the DFT coefficients that are small in absolute value, one obtains a

[1]The dog ECG data is available from the Signal Processing Information Base (SPIB) at URL `http://spib.rice.edu/`.

more efficient representation of $\{x(n)\}$. Fig. 2.3 illustrates the DFT coefficients when the 409 coefficients that are largest in absolute value are kept, and the remaining 1639 DFT coefficients are set to zero. Fig. 2.3 also shows the signal reconstructed from this truncated DFT. It can be seen that the reconstructed signal is a fairly accurate depiction of the original signal $\{x(n)\}$. For signals that are made up primarily of a few strong frequency components, the DFT is even more suitable for compression purposes.

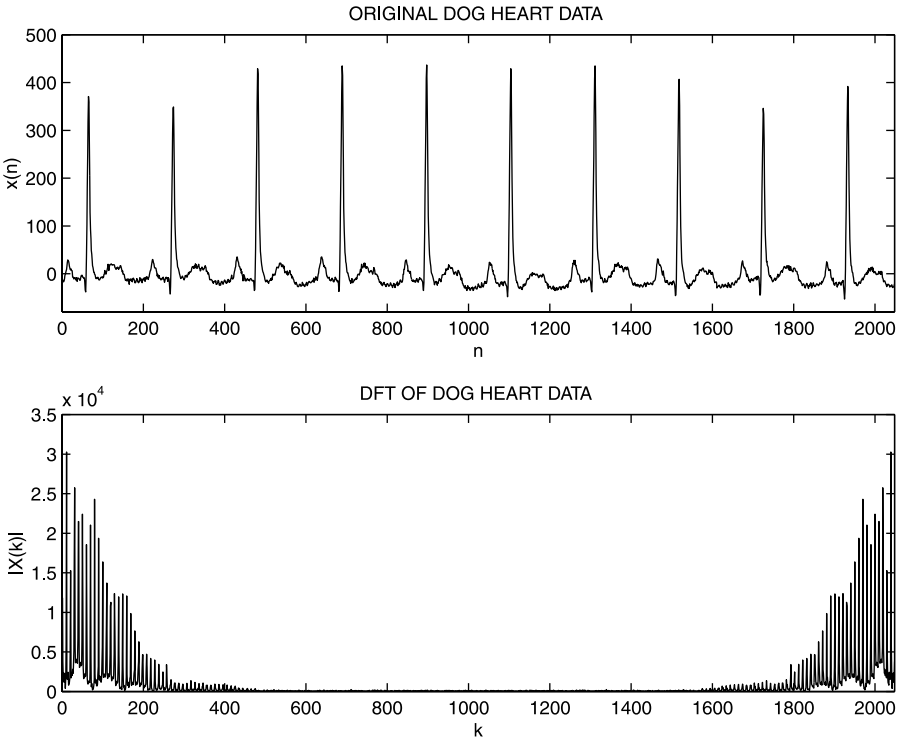


FIGURE 2.2
2048 samples recorded of a dog heart and its DFT coefficients. The magnitudes of the DFT coefficients are shown (see property 1 in Section 2.5.1).

2.4 DFT Frequency Analysis

To formalize the type of frequency analysis accomplished by the DFT, it is useful to view each DFT value $\{X(k)\}$ as the output of a length N FIR filter $h_k(n)$. The

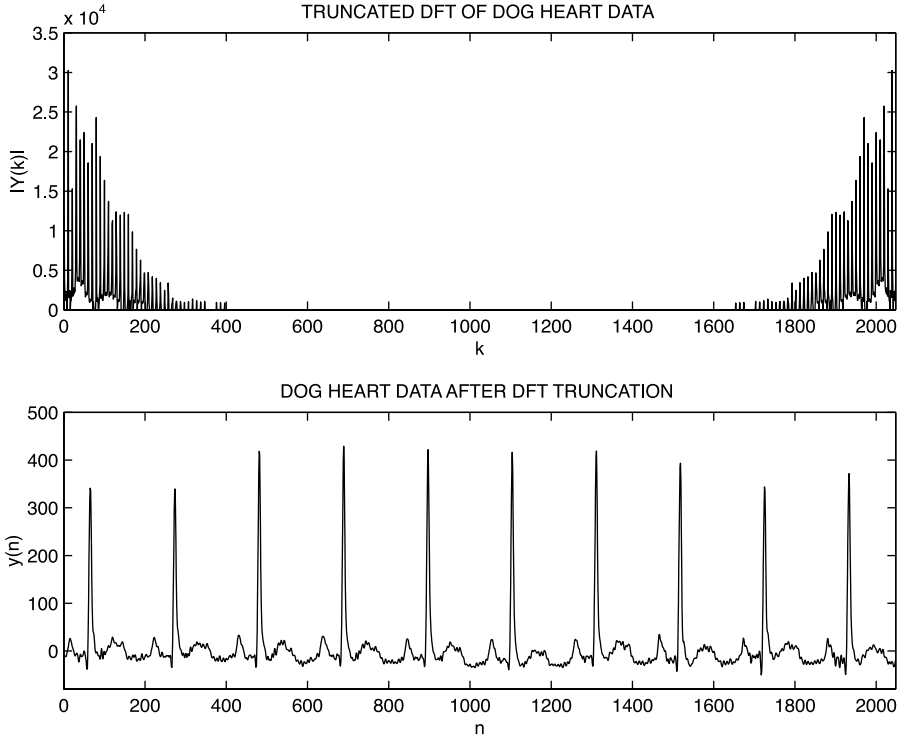


FIGURE 2.3
The truncated DFT coefficients and the time signal reconstructed from the truncated DFT.

output of the filter is given by the convolution sum

$$y_k(l) = \sum_{n=0}^{l} x(n)\, h_k(l-n)\ .$$

When the output $y_k(l)$ is evaluated at time $l = N-1$, one has

$$y_k(N-1) = \sum_{n=0}^{N-1} x(n)\, h_k(N-1-n)\ .$$

If the filter coefficients $h_k(n)$ are defined as

$$h_k(n) = \begin{cases} W_N^{k(N-1-n)} & 0 \le n \le N-1 \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

then one has

$$y_k(N-1) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn}\ , \tag{2.7}$$

$$= X(k)\ . \tag{2.8}$$

Note that $h_k(n) = W_N^{k(N-1-n)}$ represents a reversal of the values W_N^{kn} for $n = 0, \ldots, N-1$, which in turn is the k-th row of the DFT matrix. Therefore, the DFT of a length N sequence $\{x(n)\}$ can be interpreted as the output of a bank of N FIR filters each of length N sampled at time $l = N-1$.

Moreover, the impulse responses $h_k(n)$ are directly related to each other through DFT-modulation:

$$h_k(n) = W_N^{k(N-1-n)} \cdot p(n)$$

where the filter $h_0(n) = p(n)$ is given by

$$p(n) = \begin{cases} 1 & 0 \le n \le N-1 \\ 0 & \text{otherwise}\ . \end{cases} \tag{2.9}$$

This filter is called a *rectangular window* as it is not tapered at its ends. It follows that the Z-transforms of the filters are also simply related:

$$H_k(z) = \sum_{n=0}^{N-1} h_k(n)\, z^{-n} \tag{2.10}$$

$$= \sum_{n=0}^{N-1} W_N^{k(N-1-n)}\, p(n)\, z^{-n} \tag{2.11}$$

$$= W_N^{-k} \sum_{n=0}^{N-1} W_N^{-kn}\, p(n)\, z^{-n} \tag{2.12}$$

$$= W_N^{-k} \sum_{n=0}^{N-1} p(n) \left(W_N^k z\right)^{-n} \tag{2.13}$$

$$= W_N^{-k}\ P\left(W_N^k z\right) \tag{2.14}$$

where $P(z) = \sum_{n=0}^{N-1} p(n)\, z^{-n}$. That is, if each filter $h_k(n)$ in an N-channel filter bank is taken to be the time-flip of the k-th row of the DFT matrix, then their Z-transforms are given by $H_k(z) = W_N^{-k} P(W_N^k z)$. $H_0(z) = P(z)$, $H_1(z) = W_N^{-1} P(W_N z)$, etc. It is instructive to view the frequency responses of the N filters $h_k(n)$, as the frequency responses of the filters $H_k(z)$ indicate the effect of the DFT on a sequence. The magnitude of the frequency response of $H_k(z)$ and the zero plot in the z-plane are given in Fig. 2.4. Note that the zeros of $H_k(z)$ in the z-plane are simply rotated by $2\pi/N$, and that the frequency responses are shifted by the same

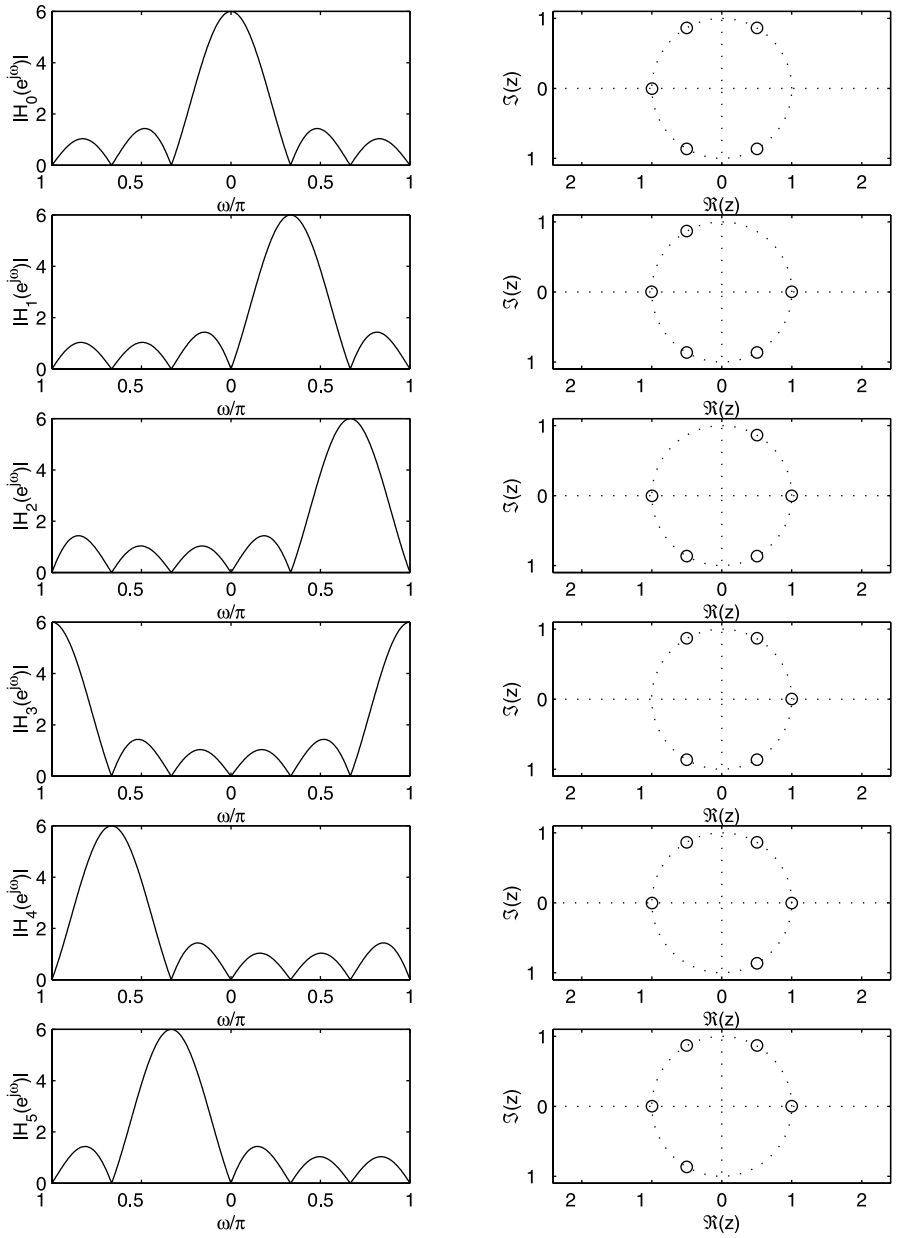


FIGURE 2.4
The magnitude of the frequency response of the filters $h_k(n)$ for $k = 0, \ldots, 5$, corresponding to a 6-point DFT. Shown on the right are the zeros of $H_k(z)$.

amount. The figure makes clear the way in which the DFT performs a frequency decomposition of a signal.

The frequency response of the filter h_k is given by $H_k(e^{j\omega})$, the discrete-time Fourier transform (DTFT) of the impulse response:

$$H_k(e^{j\omega}) = \sum_{n=0}^{N-1} h_k(n)\, e^{-j\omega n} \,. \tag{2.15}$$

The frequency response of the rectangular window $p(n)$ is given by

$$P\left(e^{j\omega}\right) = \sum_{n=0}^{N-1} 1 \cdot e^{-j\omega n} \tag{2.16}$$

$$= \frac{1 - e^{-jN\omega}}{1 - e^{-j\omega}} \tag{2.17}$$

$$= \frac{e^{-j\omega N/2}\left(e^{j\omega N/2} - e^{-j\omega N/2}\right)}{e^{-j\omega/2}\left(e^{j\omega/2} - e^{-j\omega/2}\right)} \tag{2.18}$$

$$= e^{-j\omega(N-1)/2} \cdot \frac{\sin\frac{N}{2}\omega}{\sin\frac{1}{2}\omega} \,. \tag{2.19}$$

The function $\sin(\frac{N}{2}\omega)/\sin(\frac{1}{2}\omega)$ is called the *digital sinc* function, for its resemblance to the usual sinc function.

2.5 Selected Properties of the DFT

Of the many properties the DFT possesses, the symmetry properties are some of the most useful when using the DFT for compression.

Because the DFT operates on finite-length data sequences, it is useful to define two types of symmetries as follows. When $\{x(n)\}$ is periodically extended outside the range $n = 0, \ldots, N-1$, the following definitions for symmetric and anti-symmetric sequences are consistent with their usual definitions for sequences that are not finite in length.

Symmetry: Let $\{x(n)\}$ be a real-valued length N data sequence, for $n = 0, \ldots, N-1$, then $\{x(n)\}$ is *symmetric* if

$$x(N-n) = x(n), \qquad n = 1, \ldots, N-1 \,.$$

Note that an *even*-length N symmetric sequence $\{x(n)\}$ is fully described by its first $N/2+1$ values. For example, a length 6 symmetric sequence is fully determined by its first 4 values as illustrated in Fig. 2.5. On the other hand, an *odd*-length N symmetric sequence $\{x(n)\}$ is fully described by its first $(N+1)/2$ values. For

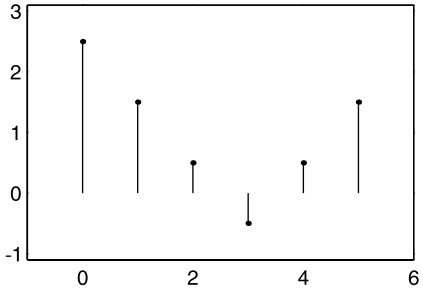


FIGURE 2.5
Illustration of even-length symmetric sequence.

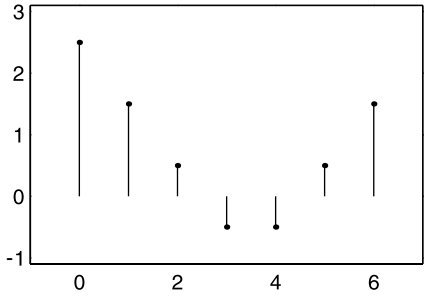


FIGURE 2.6
Illustration of odd-length symmetric sequence.

example, a length 7 symmetric sequence is fully determined by its first 4 values as illustrated in Fig. 2.6. For both even- and odd-length sequences, the number of values that determine a length N symmetric sequence is $\lfloor N/2+1 \rfloor$ where $\lfloor k \rfloor$ denotes the greatest integer smaller than or equal to k.

Anti-symmetry: A real-valued length N data sequence is *anti*-symmetric if

$$x(0) = 0 \qquad \text{and} \qquad x(N-n) = -x(n), \qquad n = 1, \ldots, N-1 \, .$$

Note that an *even*-length N anti-symmetric sequence $\{x(n)\}$ is fully described by $N/2-1$ values. For example, a length 6 anti-symmetric sequence is fully determined by 2 values (see Fig. 2.7). On the other hand, an *odd*-length N anti-symmetric sequence $\{x(n)\}$ is fully described by $(N-1)/2$ values. For example, a length 7 anti-symmetric sequence is fully determined by 3 values (see Fig. 2.8). For both even- and odd-length sequences, the number of values that determine a length N anti-symmetric sequence is $\lceil N/2-1 \rceil$ where $\lceil k \rceil$ denotes the smallest integer greater than or equal to k.

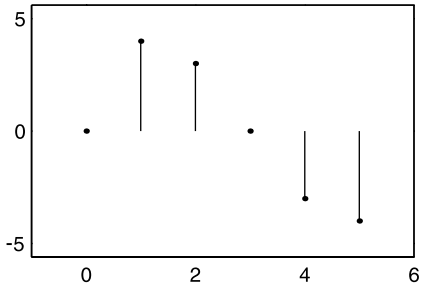


FIGURE 2.7
Illustration of even-length anti-symmetric sequence.

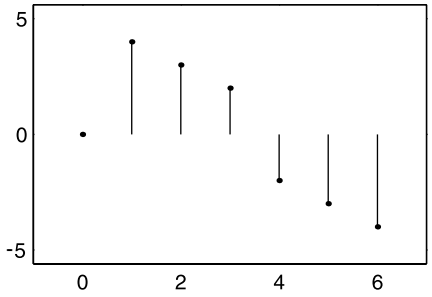


FIGURE 2.8
Illustration of odd-length anti-symmetric sequence.

2.5.1 Symmetry Properties

To state the symmetry properties of the DFT, it is useful to introduce the notation $\{X_r(k)\}$ and $\{X_i(k)\}$ for the real and imaginary parts of $\{X(k)\}$. Similarly, $\{x_r(n)\}$ and $\{x_i(n)\}$ are used to denote the real and imaginary parts of $\{x(n)\}$.

If $\{x(n)\}$ is a length N data vector and ...

1. if $\{x(n)\}$ is *real-valued,* then

$$X(k) = X^*(N-k), \qquad k = 1, \ldots, N-1 \ ,$$

   i.e., the real part of $\{X(k)\}$ is symmetric, and the imaginary part of $\{X(k)\}$ is anti-symmetric.

2. if $\{x(n)\}$ is *real-valued and symmetric,* then

$$X(k) = X_r(k) = X_r(N-k), \quad k = 1, \ldots, N-1 \ ,$$

   i.e., $\{X(k)\}$ is purely real and symmetric.

3. if $\{x(n)\}$ is *real-valued and anti-symmetric,* then

$$X(k) = j\, X_i(k) = -j\, X_i(N-k), \quad k = 1, \ldots, N-1 \ ,$$

i.e., $\{X(k)\}$ is purely imaginary and anti-symmetric.

4. if $\{x(n)\}$ is *purely imaginary,* then

$$X(k) = -X^*(N-k), \qquad k = 1, \ldots, N-1 \ ,$$

i.e., the real part of $\{X(k)\}$ is anti-symmetric, and the imaginary part of $\{X(k)\}$ is symmetric.

5. if $\{x(n)\}$ is *purely imaginary and* $\{x_i(n)\}$ *is symmetric,* then

$$X(k) = j\, X_i(k) = j\, X_i(N-k), \quad k = 1, \ldots, N-1 \ ,$$

i.e., $\{X(k)\}$ is purely imaginary and symmetric.

6. if $\{x(n)\}$ is *purely imaginary and* $\{x_i(n)\}$ *is anti-symmetric,* then

$$X(k) = X_r(k) = -X_r(N-k), \quad k = 1, \ldots, N-1 \ ,$$

i.e., $\{X(k)\}$ is purely real and anti-symmetric.

These properties are summarized in Table 2.1.

These properties explain why the total number of parameters needed to describe the original data sequence $\{x(n)\}$ is the same after the DFT is performed. For example, consider a real-valued length 6 sequence $\{x(n)\}$ and its DFT:

$$\mathbf{x} = \begin{bmatrix} 1 \\ 3 \\ 5 \\ 6 \\ 7 \\ 2 \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} 24.0000 \\ -8.5000 \\ -1.5000 \\ 2.0000 \\ -1.5000 \\ -8.5000 \end{bmatrix} + j \begin{bmatrix} 0 \\ 0.8660 \\ -2.5981 \\ 0 \\ 2.5981 \\ -0.8660 \end{bmatrix} .$$

It is clear that there are a total of 6 distinct values in the DFT coefficients $\{X(k)\}$ for this example.

In general, for a length N real-valued sequence $\{x(n)\}$, the symmetric $\{X_r(k)\}$ is determined by $\lfloor N/2+1 \rfloor$ values, and the anti-symmetric $\{X_i(k)\}$ is determined by $\lceil N/2-1 \rceil$ values. Therefore, even though the DFT $\{X(k)\}$ of a length N real-valued sequence $\{x(n)\}$ is complex-valued, it is fully determined by exactly N values. The number of parameters is the same in both $\{x(n)\}$ and $\{X(k)\}$.

Recall that an even-length real-valued symmetric sequence $\{x(n)\}$ is determined by its first $N/2+1$ values. By the symmetry property above, the same is true for the DFT $\{X(k)\}$. An odd-length real-valued symmetric sequence $\{x(n)\}$ is determined by its first $(N+1)/2$ values. By the symmetry property above, the same is true for the DFT $\{X(k)\}$. The symmetry properties for real-valued symmetric sequences are especially useful because they can be used to develop useful DFT-based transforms that yield real-valued coefficients.

Table 2.1 DFT Symmetry Properties

$\mathbf{x}$ is purely real		$\mathbf{X}_r$ is symmetric,	$\mathbf{X}_i$ is anti-symmetric
$\mathbf{x}$ is purely real,	$\mathbf{x}_r$ is symmetric	$\mathbf{X}_r$ is symmetric,	$\mathbf{X}$ is purely real
$\mathbf{x}$ is purely real,	$\mathbf{x}_r$ is anti-symmetric	$\mathbf{X}$ is purely imaginary,	$\mathbf{X}_i$ is anti-symmetric
$\mathbf{x}$ is purely imaginary		$\mathbf{X}_r$ is anti-symmetric,	$\mathbf{X}_i$ is symmetric
$\mathbf{x}$ is purely imaginary,	$\mathbf{x}_i$ is symmetric	$\mathbf{X}$ is purely imaginary,	$\mathbf{X}_i$ is symmetric
$\mathbf{x}$ is purely imaginary,	$\mathbf{x}_i$ is anti-symmetric	$\mathbf{X}_r$ is anti-symmetric,	$\mathbf{X}$ is purely real

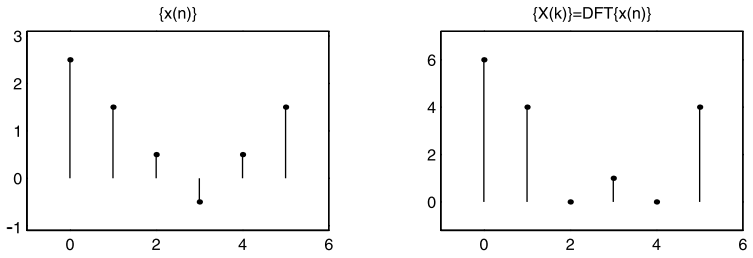


FIGURE 2.9
Illustration of DFT symmetry property for an even-length sequence.

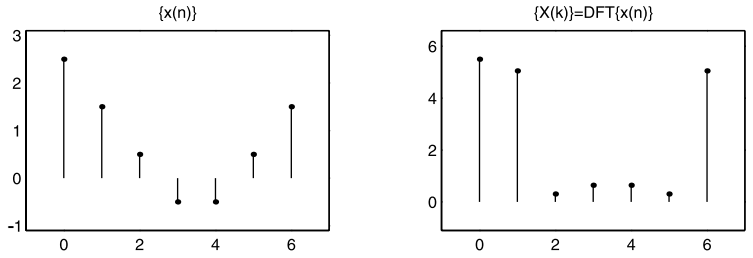


FIGURE 2.10
Illustration of DFT symmetry property for an odd-length sequence.

2.6 Real-Valued DFT-Based Transforms

In most applications the data are real-valued. For this reason, it can be beneficial to use the DFT in a specialized way so that it gives real values. This can be accomplished by suitably extending the given data sequence $\{x(n)\}$ so that it exhibits the necessary symmetry that makes the DFT $\{X(k)\}$ real-valued.

For example, given a length N real-valued sequence $\{x(n)\}$, which does not necessarily possess any symmetries, one can construct a symmetric sequence by symmetrically extending $\{x(n)\}$. There is more than one way to symmetrically extend a given sequence, depending on how the end points are treated. Different symmetric

extensions give rise to the different types of DFT-based signal transforms that map real-valued sequences to real-valued sequences. One class of DFT-based real transforms is the *discrete cosine and sine transforms.* In fact, 16 different cosine and sine transforms are described in [32].

One way to symmetrically extend a finite length N sequence is illustrated in Fig. 2.11. The result is a symmetric sequence $\{x_1(n)\}$ of even length $2N-2$. $\{X_1(k)\}$,

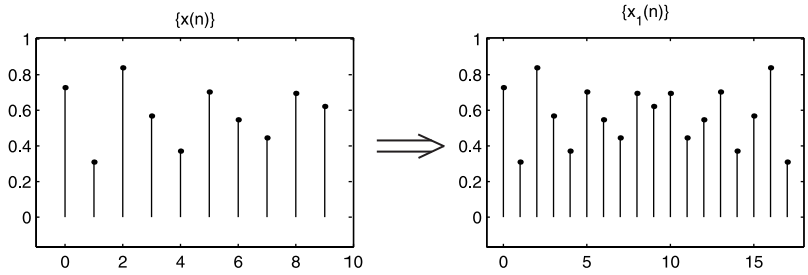


FIGURE 2.11
Illustration of symmetric extension.

the DFT of $\{x_1(n)\}$, is therefore real-valued symmetric and is determined by its first N values (see Fig. 2.12). Because $\{X_1(k)\}$ is determined by its first N values, this procedure gives an N-point real transform. The inverse of this transform is obtained by performing the same steps in reverse sequence. Given the first N values of $\{X_1(k)\}$, construct a symmetric extension as above to obtain a length $2N-2$ sequence $\{X_1(k)\}$, take the inverse DFT of the resulting sequence to obtain the length $2N-2$ sequence $\{x_1(n)\}$, from which $\{x(n)\}$ can be extracted.

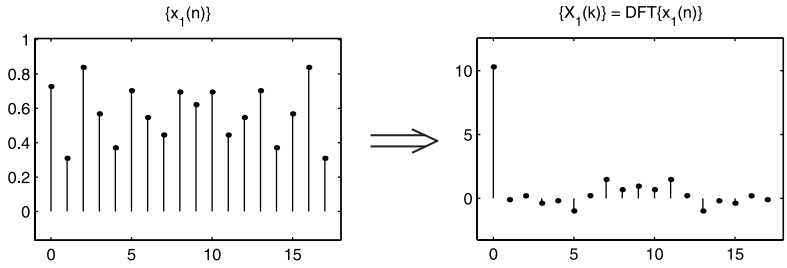


FIGURE 2.12
Illustration of symmetric extension.

The transform formulae can be found explicitly using the DFT formulae together with the symmetric extension.

$$X_1(k) = \text{DFT}\{x_1(n)\} \tag{2.20}$$

$$= \sum_{n=0}^{2N-3} x_1(n)\, W_{2N-2}^{kn} \tag{2.21}$$

$$= x(0) + \sum_{n=1}^{N-2} x(n)\left[W_{2N-2}^{nk} + W_{2N-2}^{2N-2-n}\right] + x(N-1)\, W_{2N-2}^{(N-1)k} \tag{2.22}$$

$$= x(0) + 2\sum_{n=1}^{N-2} x(n)\cos\left(\frac{nk\pi}{N-1}\right) + (-1)^k x(N-1) \tag{2.23}$$

where we have used the simplification $W_{2N-2}^{k(2N-2-n)} = W_{2N-2}^{-nk}$. Often the first and last values, $x(0)$ and $x(N-1)$, are scaled by $\sqrt{2}$ so that the transform is orthogonal. The inverse can also be derived in a similar way.

It is very interesting to look at the type of frequency analysis this type of discrete cosine transform (DCT) [1, 42, 61] performs, as was done for the DFT in Fig. 2.4. In Fig. 2.13, the frequency responses corresponding to this DCT are shown. Note that the plots of zeros in the z-plane are especially simple.

Another way to symmetrically extend a finite length N sequence is illustrated in Fig. 2.14. The result is a symmetric sequence $\{x_2(n)\}$ of odd length $2N-1$. $\{X_2(k)\}$, the DFT of $\{x_2(n)\}$, is therefore real-valued symmetric and is determined by its first N values (see Fig. 2.15). Because $\{X_2(k)\}$ is determined by its first N values, this procedure gives an N-point real transform. The inverse of this transform is obtained by performing the same steps in reverse sequence. Given the first N values of $\{X_2(k)\}$, construct a symmetric extension as above to obtain a length $2N-1$ sequence $\{X_2(k)\}$, and take the inverse DFT of the resulting sequence to obtain the length $2N-1$ sequence $\{x_2(n)\}$, from which $\{x(n)\}$ can be extracted.

Now consider a symmetric extension by simply mirroring the entire length N sequence,

$$\{x_1(n)\} = [x(0), \ldots, x(N-1), x(N-1), \ldots, x(0)]$$

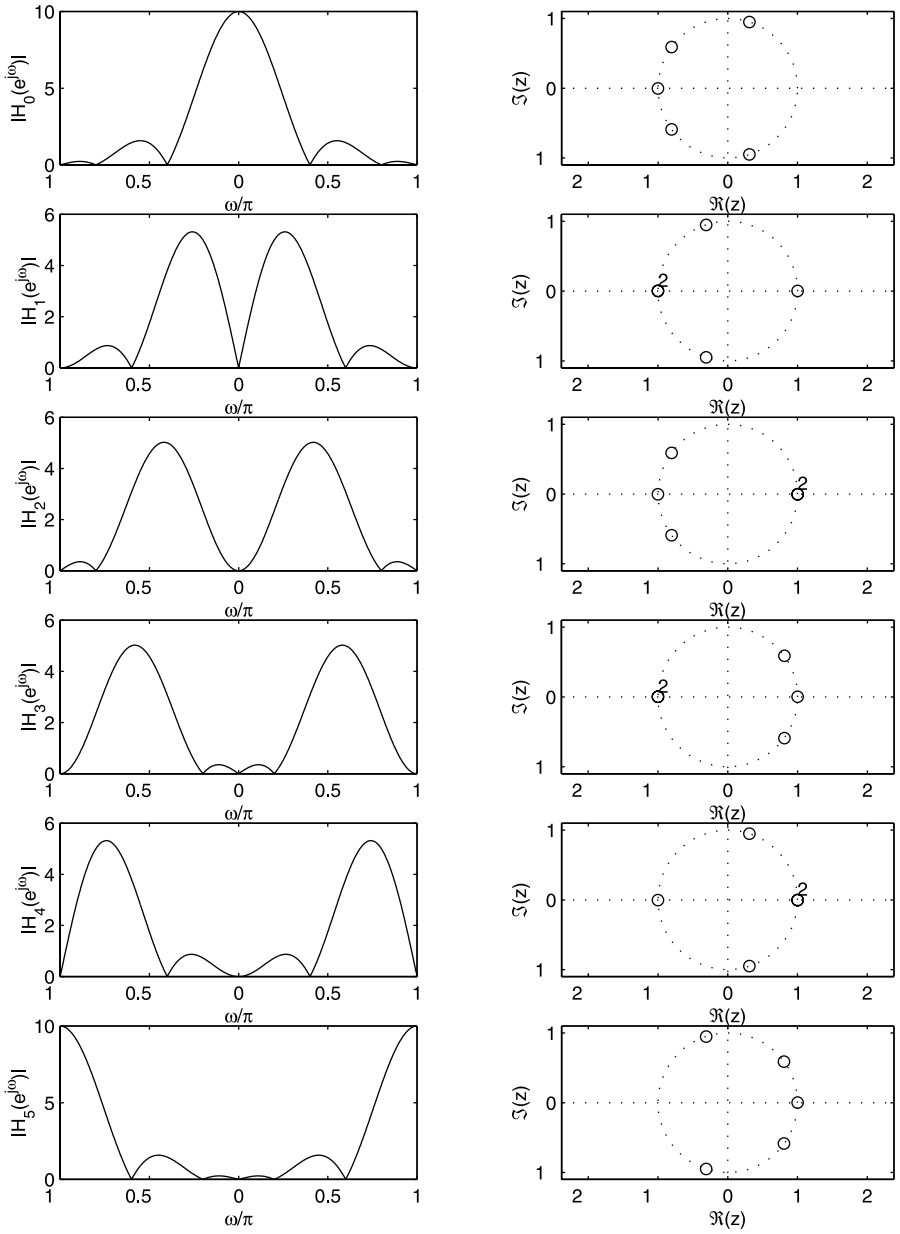


FIGURE 2.13
The discrete cosine transform (I) basis vectors illustrated in the frequency domain and in the z-plane. $N = 6$.

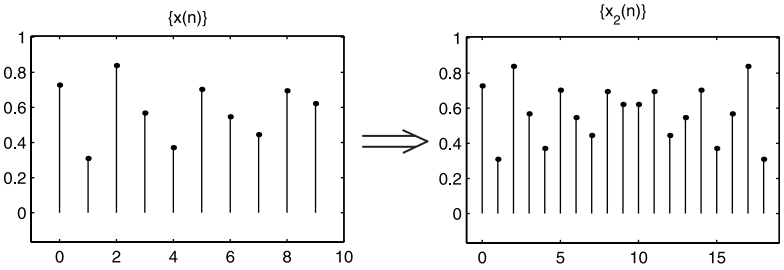


FIGURE 2.14
Illustration of DFT symmetry property.

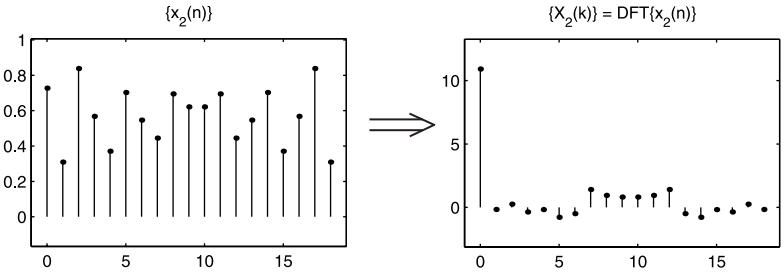


FIGURE 2.15
Illustration of DFT symmetry property.

for $0 \leq n \leq 2N-1$ (a length $2N$ sequence). The DFT of this sequence becomes

$$
\begin{aligned}
X_1(k) &= \mathrm{DFT}\{x_1(n)\} && (2.24)\\
&= \sum_{n=0}^{2N-1} x_1(n) W_{2N}^{kn} && (2.25)\\
&= \sum_{n=0}^{N-1} x(n) \left[W_{2N}^{kn} + W_{2N}^{k(2N-1-n)} \right] && (2.26)\\
&= \sum_{n=0}^{N-1} x(n) W_{2N}^{-0.5k} \left[W_{2N}^{k(n+0.5)} + W_{2N}^{k(2N-0.5-n)} \right] && (2.27)\\
&= W_{2N}^{-0.5k} \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi}{N} \cdot k \cdot (n+0.5) \right) && (2.28)
\end{aligned}
$$

The phase factor $W_{2N}^{-0.5k}$ can be neglected in applications since it carries no information about the signal. Since the transform length is N, the frequency index has the range $k = 0, \ldots, N-1$, so that a quadratic cosine transform matrix is obtained. The

DCT thus obtained is a so-called DCT type II. Its transform matrix is

$$\mathbf{D}_{II}(k,n) := \sqrt{2/N}\cos\left(\frac{\pi}{N}k(n-0.5)\right)$$

for $n, k = 0, \ldots, N-1$. To make this transform matrix orthogonal, its first row is usually scaled to

$$\mathbf{D}_{II}(0,n) := \sqrt{1/N}$$

for $k = 0$. This transform divides the frequency axis as illustrated in Fig. 2.16. It can be seen that the width of the resulting frequency bins or bands is π/N, except for the lowest band for $k = 0$, as it is centered around DC. This results in a lowpass filter bandwidth of $1/(2N)$. The highest band for $k = N-1$ is centered at $\pi(1-1/N)$, which means the required bandwidth to cover the entire frequency axis up to π is $2/N$. This means that for the design of filter banks with uniform frequency width for all bands, a shift of the frequency grid by $1/2$ would be suitable, so that the lowest band covers more bandwidth, and the highest band needs to cover less, as illustrated in Fig. 2.17. This results in a DCT type IV; its orthogonal transform matrix is

$$\mathbf{D}_{IV}(k,n) := \sqrt{2/N}\cos\left(\frac{\pi}{N}(k+0.5)(n+0.5)\right) . \tag{2.29}$$

Similarly a discrete sine transform of types II and IV are obtained by applying a DFT to the sequence

$$\{x_1\} = [x(0), \ldots, x(N-1), -x(N-1), \ldots, -x(0)]$$

for $0 \le n \le 2N-1$. The resulting transform matrix for a DST type IV is

$$\mathbf{S}_{IV}(k,n) := \sqrt{2/N}\sin\left(\frac{\pi}{N}(k+0.5)(n+0.5)\right) . \tag{2.30}$$

Efficient ways to obtain DCTs with the help of FFTs can be found, for example, in Malvar [31].

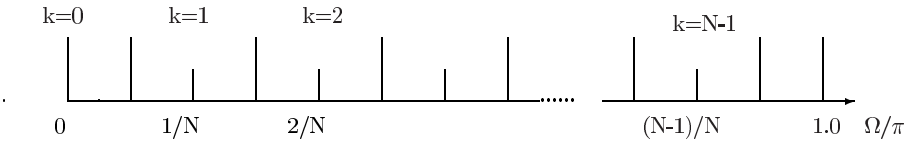


FIGURE 2.16
The distribution of bands with a DCT II. Horizontally is the normalized frequency Ω/π. The band edges are marked with long vertical lines, and the band centers with short lines.

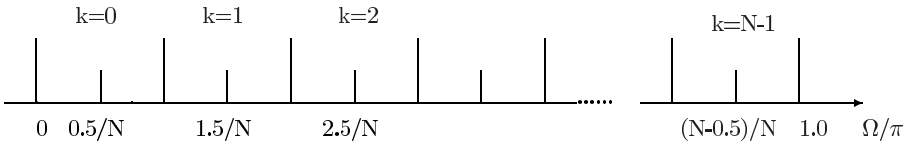


FIGURE 2.17
The distribution of bands with a DCT IV. The band edges are again marked with long vertical lines, and the band centers with short lines.

2.7 The Fast Fourier Transform

A fast Fourier transform (FFT) is any fast algorithm for computing the DFT. As stated earlier, FFT algorithms have a tremendous impact on computational aspects of signal processing. To introduce the FFT, recall the definition of the DFT in Eq. (2.1) and suppose the data vector $\{x(n)\}$ is of even length N. The basic derivation of the FFT begins by splitting the sum into two parts — one part for the even-indexed values $\{x(2n)\}$ and one part for the odd-indexed values $\{x(2n+1)\}$

$$X(k) = \sum_{\substack{n=0 \\ n\ even}}^{N-1} x(n)\, W_N^{nk} + \sum_{\substack{n=0 \\ n\ odd}}^{N-1} x(n)\, W_N^{nk}$$

which can be written as

$$X(k) = \sum_{n=0}^{N/2-1} x(2n)\, W_N^{2nk} + \sum_{n=0}^{N/2-1} x(2n+1)\, W_N^{(2n+1)k}$$

or as

$$X(k) = \sum_{n=0}^{N/2-1} x(2n)\, W_N^{2nk} + W_N^k \sum_{n=0}^{N/2-1} x(2n+1)\, W_N^{2nk}\,.$$

Note that W_N^{2nk} can be rewritten as follows:

$$W_N^{2nk} = e^{-j2\pi(2nk)/N} \tag{2.31}$$

$$= e^{-j2\pi(nk)/(N/2)} \tag{2.32}$$

$$= W_{N/2}^{nk}\,. \tag{2.33}$$

Hence the DFT values $\{X(k)\}$ can be written as

$$X(k) = \sum_{n=0}^{N/2-1} x(2n)\, W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} x(2n+1)\, W_{N/2}^{nk}\,.$$

Note that the first sum is the length $N/2$ DFT of the sequence $\{x(2n)\}$ and the second sum is the length $N/2$ DFT of the sequence $\{x(2n+1)\}$. Defining these sequences as $\{x_0(n)\} = \{x(2n)\}$ and $\{x_1(n)\} = \{x(2n+1)\}$ for $n = 0, \ldots, N-1$ makes them both sequences of length $N/2$. Then one has

$$X(k) = X_0(k) + W_N^k X_1(k), \qquad k = 0, \ldots, N-1 \ ,$$

where $\{X_0(k)\}$ and $\{X_1(k)\}$ are the DFTs of $\{x_0(n)\}$ and $\{x_1(n)\}$, respectively. It should be noted that in the definition of the length N DFT, $\{X(k)\}$ was defined for $k = 0, \ldots N-1$. As $\{x_0(n)\}$ is a sequence of length $N/2$, its DFT is also of length $N/2$, and therefore $\{X_0(k)\}$ would be defined for $k = 0, \ldots, N/2-1$. However, as noted in Section 2.1, when k is taken outside this range, the DFT coefficients are periodic — so $X_0(k) = X_0(k - N/2)$ for values of k from $N/2$ to $N-1$. Likewise for $X_1(k)$.

This expression shows how a length N DFT can be computed using two length $N/2$ DFTs. After taking the two length $N/2$ DFTs it remains only to multiply the result of the second DFT with the terms W_N^k and to add the results. The multipliers W_N^k are known as *twiddle factors.*

If $N/2$ can be further divided by 2, then the same procedure can be used to calculate the length $N/2$ DFTs. To determine the arithmetic complexity of this algorithm for computing the DFT, let $A(N)$ denote the number of complex additions for computing the DFT of a length N complex sequence $\{x(n)\}$. Let N be a power of 2, $N = 2^K$. Then, according to the above procedure, one has

$$A(N) = 2\,A(N/2) + N$$

as N complex additions are required to put the two length $N/2$ DFTs back together. Note that a length 2 DFT is simply a sum and difference:

$$X(0) = x(0) + x(1)$$
$$X(1) = x(0) - x(1) \ .$$

Hence, the starting condition is $A(2) = 2$. [Or one can use $A(1) = 0$.] Then solving the recursive equation yields

$$A(N) = N \log_2 N \quad \text{complex additions.}$$

Similarly, one has a recursive formula for complex multiplications:

$$M(N) = 2\,M(N/2) + N/2$$

which gives

$$M(N) = \frac{N}{2} \log_2 N \quad \text{complex multiplications.}$$

In fact, this number can be reduced by a more careful examination of the multipliers W_N^k (the twiddle factors). In particular, the numbers 1, -1, j, and $-j$ will be among

the twiddle factors W_N^k, when k is a multiple of $N/4$ — and so these multiplications need not be performed. Taking this into account, one has the following formulae for the number of real additions and real multiplications of the DFT of a sequence whose length is a power of 2:

$$A_r(N) = 7\frac{N}{2}\log_2 N - 5N + 8 \tag{2.34}$$

$$M_r(N) = 3\frac{N}{2}\log_2 N - 5N + 8 \tag{2.35}$$

where a complex addition counts as two real additions, and a complex multiplication counts as three real additions and three real multiplications.

The advantage of the efficient algorithm for computing the DFT is a reduction from an arithmetic complexity of N^2 for direct calculation to a complexity of $N \log_2 N$. This is a fundamental improvement in the complexity, and historically it led to many new developments in signal processing that would not otherwise have been possible or practical. Due to its fundamental quickening in calculating the DFT, the efficient algorithm for its computation is called the *fast Fourier transform* or FFT.

Many variations and enhancements of this basic algorithm have been developed in the literature and used in practice, and they are collectively called FFTs. Of particular note is the *split radix FFT* [16, 50, 56], which is a refinement of the algorithm that attains the lowest computational complexity of practical FFT variants for lengths that are powers of 2. FFT algorithms can be developed for lengths that are not powers of 2. Some types of FFTs, called *prime factor FFTs,* do not require the use of twiddle factors [9, 52, 53] and therefore have a reduced computational complexity (this is possible when the length N is factored into relatively prime integers. It is not applicable for lengths that are powers of 2). Implementations of the FFT for real-valued data are described in Sorenson et al. [51]. Most FFT algorithms depend on the ability to factor N, the length of the data vector $\{x(n)\}$; for prime-length DFTs a separate approach is needed to combine shorter FFTs. The algorithms for prime-length FFTs are based on work by Rader and Winograd [40, 60, 59]. FFT programs for prime lengths are discussed in several publications [25, 29, 46]. Descriptions of the different types of FFTs are available in several books [4, 7, 20, 10, 33, 35, 36, 54, 28] and book chapters [8, 18, 19, 49]. The complexity theory associated with the FFT is described in Winograd [60] and Heideman [22]. A comparison of different FFT implementations on DSP chips is described in Meyer and Schwarz [34].

A relevant issue in practice is the trade-off between computational complexity and implementation complexity. The right balance must be obtained for the best results and some FFT algorithms with improved computational complexity are more complex to implement than others. Moreover, for the fastest results, the variant of the FFT chosen should be matched to the hardware on which it will run. Methods for choosing the best variant of the FFT from among a family of FFTs have been the subject of recent research [23, 24, 21].

2.8 The DFT in Coding Applications

In coding applications the DFT is used in two broad classes — in power spectrum estimation and in subband coding, where it is used in the implementation of complex-, cosine- or sine-modulated filter banks. As an illustration, audio coding will be considered in the following.

In audio coding, the real-valued audio signal is decomposed into a number of subbands with a filter bank. The subband signals are then adaptively quantized and encoded [47, 6]. The subband decomposition has the purpose of obtaining a more efficient description of the signal (redundancy reduction) and applying a psycho-acoustic model to control the quantization noise such that it will be inaudible (irrelevance reduction); see Fig. 2.18.

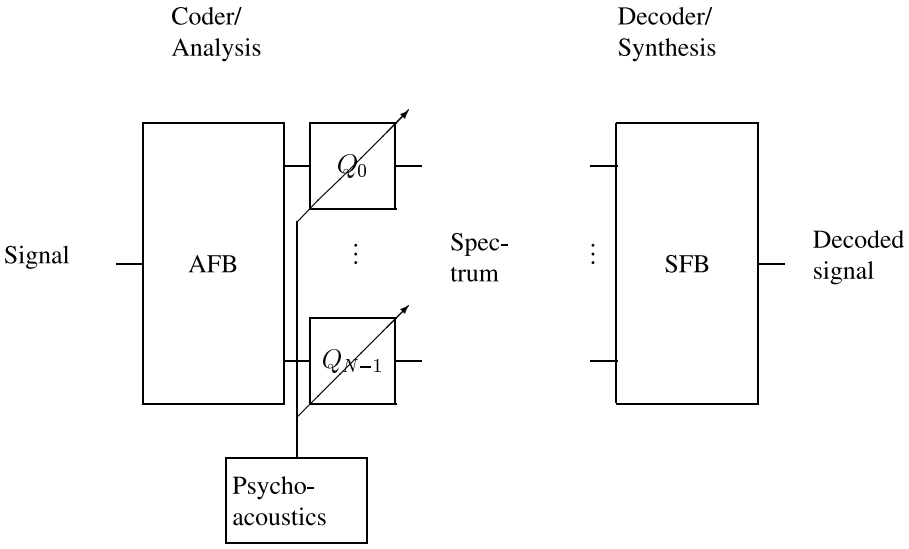


FIGURE 2.18
Audio coding based on filter banks, AFB: analysis filter bank, SFB: synthesis filter bank.

In audio coding, the subband decomposition is usually obtained with a filter bank called *modified discrete cosine transform* MDCT. It can often be switched between differing numbers of bands, for example, between 128 and 1024 bands. The MDCT is used, for example, in ASPEC, MPEG, MUSICAM, and PAC audio coders [30]. ASPEC and MUSICAM were later combined into MPEG-1 layer III, also known as MP3.

One way in which the DFT is used in subband coding is for the implementation of filter banks. Since the filters $h_k(n) = e^{j2\pi kn/N}, n = 0, \ldots, N-1, k = 0, \ldots, N-1$, can be seen as a rectangular window of length N multiplied with the exponential, the

frequency localization is not very good, as can be seen in Fig. 2.4. Since this frequency localization is very important in coding applications, the DFT is used only indirectly in coding applications, e.g., for implementing the MDCT. The output of the MDCT is real valued for real-valued inputs, and its subband filter impulse responses $h_k(n)$ are longer and have a nonrectangular shape, such that the frequency localization is better than for the DFT. The MDCT filter bank can be implemented using a DCT of length N, which in turn can be implemented using FFTs of length $N/2$ [31].

In audio coding the DFT is also used as a complex filter bank. The psycho-acoustic model, used to control the quantization step size, needs to detect and estimate signals (sinusoids) in the subbands, i.e., it needs a reliable estimate of the time-varying power spectrum, with a time and frequency resolution as similar to the MDCT as possible. This is most reliably done with a complex valued spectral decomposition because it provides the phase and magnitude of signals in the subbands at every time step. To estimate the spectrum, only the magnitude of the subband signal is needed.

This would not be possible with a real-valued filter bank because in such a filter bank a sinusoid in a subband is still a sinusoid after filtering, which will pass through zero at certain times — so it may not be detected. That is, the estimated power of the signal at that frequency and time would be lower than it should be. That is why some audio coders [e.g., MPEG-AAC (Advanced Audio Coder) [30]] possess an FFT parallel to the MDCT as input to the psycho-acoustic model. But a problem is the insufficient frequency localization of the FFT, which reduces the accuracy of the psycho-acoustic model.

The so called Balian-Low theorem states that the rectangular window of the DFT gives rise to the only orthogonal FIR filter bank with complex Fourier modulation and critical sampling [57] (every N input samples produce N output samples). However, for the time-varying spectral estimation required for the psycho-acoustic model, critical sampling is not a constraint. That is why, for example, in the perceptual audio coder (PAC) [30] the input of the psycho-acoustic model is a complex signal, which is taken from two filter banks. The real part of the signal is the output of the real-valued MDCT filter bank with a cosine modulation function. Hence, only an appropriate imaginary part corresponding to this signal is needed to obtain a complete complex subband signal — which will have improved frequency localization and therefore a more accurate psycho-acoustic model. This imaginary part of the subband signal can be obtained by using a second filter bank which is based on the same window function as the MDCT, but with a sine modulation function instead of a cosine modulation function. Interestingly, this sine-modulated filter bank alone is again a perfect reconstruction (PR) filter bank, as is the cosine-modulated MDCT filter bank. These two filter banks, in parallel, can be seen as one complex filter bank which is twice oversampled. Hence the limitation the Balian-Low theorem no longer applies, as the filter bank system is not critically sampled.

2.9 The DFT and Filter Banks

Because the frequency content of many signals changes with time, it is often more desirable to first partition a signal into blocks and then apply the DFT to each block individually. This *block-wise* DFT leads to a point of view based on filter banks. If the independent variable of the input signal is time (e.g., an audio signal), then this results in a time-frequency representation. If the input data is arranged in a matrix

$$\mathbf{x} = \begin{bmatrix} x(0) & x(N) & x(2N) & \cdots \\ x(1) & x(N+1) & x(2N+1) & \cdots \\ \vdots & & & \\ x(N-1) & x(2N-1) & x(3N-1) & \cdots \end{bmatrix}$$

and $\mathbf{F}_N$ is the DFT matrix, then the block-wise DFT can be written as

$$\mathbf{X} = \mathbf{F}_N \cdot \mathbf{x} \tag{2.36}$$

where each column of the matrix $\mathbf{X}$ is a DFT spectrum. Clearly this operation is easily inverted with

$$\mathbf{x} = (\mathbf{F}_N)^{-1} \cdot \mathbf{X} \,. \tag{2.37}$$

Depending on the amount of data, the matrices for $\mathbf{X}$ and $\mathbf{x}$ can be quite large. To simplify the mathematical description and to obtain a more general formulation, the Z-transform can be used. Then each block, or time frame, of $\mathbf{X}$ and $\mathbf{x}$ is associated with a power of z^{-1}, and the data becomes a vector of polynomials in z^{-1},

$$\mathbf{x}(z) = \begin{bmatrix} x(0) + x(N)\,z^{-1} + x(2N)\,z^{-2} + \cdots \\ x(1) + x(N+1)\,z^{-1} + x(2N+1)\,z^{-2} + \cdots \\ \vdots \\ x(N-1) + x(N+N-1)\,z^{-1} + x(2N+N-1)\,z^{-2} + \cdots \end{bmatrix} .$$

This leads to

$$\mathbf{X}(z) = \mathbf{F}_N \cdot \mathbf{x}(z) \tag{2.38}$$

and

$$\mathbf{x}(z) = (\mathbf{F}_N)^{-1} \cdot \mathbf{X}(z) \,. \tag{2.39}$$

These equations are quite similar to Eqs. (2.36) and (2.37), but now the data $\mathbf{x}$ and $\mathbf{X}$ are in the form of a simple vector instead of a possibly infinite matrix. The operation of applying the DFT to blocks of the signal can now also be viewed as a filter bank, as seen in Fig. 2.19. The symbol $\downarrow N$ means a downsampling operation, i.e., only every N-th sample is let through. This figure shows an analysis filter bank on the left which

corresponds to Eq. (2.36), and a synthesis filter bank on the right which corresponds to Eq. (2.37). Since the DFT is invertible, the signal $\{x(n)\}$ can be directly obtained from the block-wise DFT coefficients using the inverse DFT on each block. This inverse can also be interpreted in terms of filter banks as illustrated by the synthesis filter bank in Fig. 2.19.

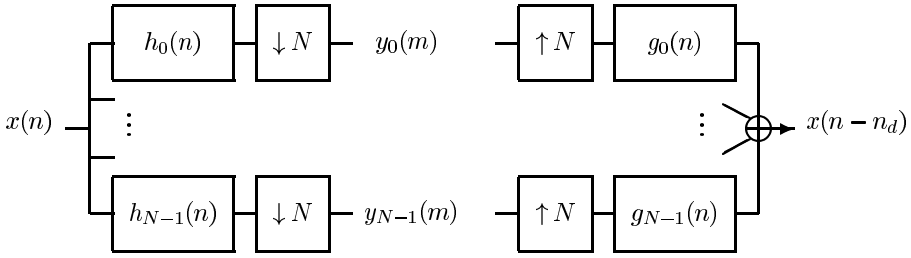


FIGURE 2.19
An N-channel filter bank with critical downsampling, perfect reconstruction, and a system delay of n_d samples.

Because the analysis filter bank is invertible, it is said to have the perfect reconstruction (PR) property. Because the total number of samples in the input signal $\{x(n)\}$ equals the total number of samples in the subbands (the N channels), it is said to be critically sampled. In coding applications, critical downsampling is important because it leads to an accurate and complete description of a signal with the least possible number of samples, and it leads to computationally efficient implementations. The analysis filter bank is used in the encoder, and the synthesis filter bank in the decoder.

To see that the matrix formulation can also be represented by a filter bank structure (see also Vaidyanathan [55]), consider the following. For simplicity, we assume a time-shifted sequence $\{x(n+N-1)\}$. The filtering (convolution) and downsampling operation can be written as

$$y_k(m) = \sum_n h_k(n) \cdot x(mN + N - 1 - n), \qquad 0 \le k \le N-1\,. \tag{2.40}$$

On the other hand, Eq. (2.36) can also be written as

$$\mathbf{X}_{k,m} = \sum_{n=0}^{N-1} W_N^{kn} \cdot x(mN+n), \qquad 0 \le k \le N-1\,. \tag{2.41}$$

If this equation is compared to Eq. (2.40), it can be seen by a substitution of the index variable that they are identical if the filters $\{h_k(n)\}$ are defined as

$$h_k(n) = W_N^{k(N-1-n)} = W_N^{-k} W_N^{-kn} \quad \text{for } n = 0, \ldots N-1$$

and

$$h_k(n) = 0 \quad \text{otherwise}$$

[see also Eq. (2.6)]. It was noted in Section 2.4 that these filters are complex-modulated versions of the rectangular window function. The resulting frequency responses of $\{h_k(n)\}$ are frequency-shifted versions of the frequency response of the rectangular window function $p(n)$, as can also be seen in Fig. 2.4. The block-wise interpretation of this DFT-modulated filter bank leads to an efficient algorithm for its implementation using an FFT.

The rectangular window does not have a good frequency localization because of its limited length and its rectangular shape. Fig. 2.4 shows that the main lobe of the frequency response (its passband) is quite wide, and the side lobes are not very low — the stopband attenuation is not very high. A solution is to increase the window length and to give it a different shape, such that the passband becomes more narrow and the stopband attenuation is improved (see Bellanger [2]). To this end, first consider a general window function $\{p(n)\}$ of length N, the shape of which is not necessarily rectangular. ($\{p(n)\}$ denotes the analysis *prototype* filter or window function; the synthesis prototype filter will be denoted by $\{q(n)\}$.) The filters $\{h_k(n)\}$ in this case are given by

$$h_k(n) = W_N^{-k} \cdot W_N^{-kn} \cdot p(n) \tag{2.42}$$

or in terms of Z-transforms, as

$$H_k(z) = W_N^{-k}\ H_a\left(W_N^k z\right)\ .$$

The analysis equation can then be written using a diagonal matrix as

$$\mathbf{X}(z) = \mathbf{F}_N \cdot \begin{bmatrix} p(N-1) & 0 & \cdots & 0 \\ 0 & p(N-2) & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & & p(0) \end{bmatrix} \cdot \mathbf{x}(z)\ . \tag{2.43}$$

The diagonal matrix is also called a *filter matrix,* denoted by $\mathbf{F}_a$ for the analysis. The inverse gives the equation for the synthesis stage

$$\mathbf{x}(z) = \begin{bmatrix} 1/p(N-1) & 0 & \cdots & 0 \\ 0 & 1/p(N-2) & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & & 1/p(0) \end{bmatrix} \cdot \mathbf{F}_N^{-1} \cdot \mathbf{X}(z)\ . \tag{2.44}$$

The analysis window function $p(n)$ leads to a synthesis window function of $1/p(n)$, e.g., the synthesis window is the point-wise inverse of the analysis window. Consequently, a window with improved frequency localization properties in the analysis stage can lead to worse frequency localization in the synthesis stage, which is often not desired. Also, the limited length of N of the window still is an important limiting factor in the design of better window functions.

When the filter $p(n)$ is longer than N, say LN, then Eq. (2.41) becomes

$$\mathbf{X}_{k,m} = \sum_{n=0}^{LN-1} W_N^{kn} \cdot p(LN-1-n)\, x(mN+n) \; .$$

Since $W_N^{k(n+N)} = W_N^{kn}$, we can replace n by $lN+n$ to obtain

$$\mathbf{X}_{k,m} = \sum_{n=0}^{N-1} W_N^{kn} \cdot \sum_{l=0}^{L} p(LN-1-n-lN)\, x(mN+n+lN) \; .$$

The inner sum can be interpreted as a convolution, which is written as a product in the z-domain, with

$$P_n(z) = \sum_{l=0}^{L-1} p(n+lN) \cdot z^{-l}$$

$$X_n(z) = \sum_{l=0}^{\infty} x(n+lN) \cdot z^{-l} \; .$$

This leads to

$$\mathbf{X}_k(z) = \sum_{n=0}^{N-1} W_N^{kn} \cdot P_{N-1-n}(z) \cdot X_n(z)$$

so that Eq. (2.43) becomes

$$\mathbf{X}(z) = \mathbf{F}_N \cdot \begin{bmatrix} P_{N-1}(z) & 0 & 0 & \cdots \\ 0 & P_{N-2}(z) & 0 & \cdots \\ \vdots & & & \\ 0 & \cdots & 0 & P_0(z) \end{bmatrix} \cdot \mathbf{x}(z) \; . \tag{2.45}$$

At this point it becomes clear that the synthesis requires the inverse functions $1/P_n(z)$, which represent IIR filters, whose stability is difficult to control. Consequently, a critically sampled filter bank based on filters $\{h_k(n)\}$ that are related through DFT modulation, as in Eq. (2.42), can have the perfect reconstruction property with FIR filters in both the analysis stage *and* the synthesis stage only if the filters are not longer than the downsampling rate N and have no overlap in time with neighboring blocks. To obtain FIR synthesis for longer filters, the filter bank must have a different structure.

2.9.1 Cosine-Modulated Filter Banks

We saw that a discrete cosine transform is obtained by applying a DFT to a symmetrically extended real valued signal. This suggests that a DCT would lead to a different

filter matrix $\mathbf{F}_a$, with elements off the diagonal. In many applications, as in video, audio, or speech coding, the signal is indeed represented as real values. Now it would be interesting to see the shape of the resulting filter matrix for a filter bank based on a DCT IV modulation [compare to Eq. (2.29)]. In this case, the filters $\{h_k(n)\}$ are modulated with cosine functions (the factor $\sqrt{2/N}$ is neglected for simplicity),

$$h_k(n) = \cos\Big(\frac{\pi}{N}(k+0.5)(n+0.5)\Big) \cdot p(LN-1-n)\ , \tag{2.46}$$

and the transform (the subband signals) can be written as

$$\mathbf{X}_{k,m} = \sum_{n=0}^{LN-1} \cos\Big(\frac{\pi}{N}(k+0.5)(n+0.5)\Big) \cdot p(LN-1-n)\, x(mN+n)\ . \tag{2.47}$$

We will exploit the symmetries embodied in the identities

$$\begin{aligned} &\cos\Big(\frac{\pi}{N}(k+0.5)((n+N)+0.5)\Big) \\ &\quad = -\cos\Big(\frac{\pi}{N}(k+0.5)((N-1-n)+0.5)\Big) \end{aligned} \tag{2.48}$$

and

$$\cos\Big(\frac{\pi}{N}(k+0.5)((n+2N)+0.5)\Big) = -\cos\Big(\frac{\pi}{N}(k+0.5)(n+0.5)\Big) \tag{2.49}$$

This means that every second block of N input samples "reverses the direction" of the cosine transform. A close examination of these symmetries and replacing n by $n + 2lN$ and $n + N + 2lN$ shows that the analysis equation (2.47) can be written as a type of *folding* operation followed by a cosine transform, as can be seen in the following.

Again the filtering can be written more easily in the z-domain, with

$$P_n(z) = \sum_{l=0}^{L-1} p(n+2lN) \cdot z^{-l}$$

with $n = 0, \ldots, N-1$,

$$X_n(z) = \sum_{l=0}^{\infty} x(n+lN) \cdot z^{-l}\ .$$

with $n = 0, \ldots, 2N-1$. Using $\mathbf{D}_{IV}$ as the DCT IV matrix leads to

$$\mathbf{X}(z) = \mathbf{D}_{IV} \cdot \mathbf{F}_a(z) \cdot \mathbf{x}(z)$$

where

$$\mathbf{F}_a(z) = \begin{bmatrix} z^{-1}P_{2N-1}\left(-z^2\right) & & 0 & & P_0\left(-z^2\right) \\ & \ddots & & \iddots & \\ & & z^{-1}P_{N+N/2}\left(-z^2\right) & P_{N/2-1}\left(-z^2\right) & \\ 0 & & P_{N/2}\left(-z^2\right) & z^{-1}P_{N+N/2-1}\left(-z^2\right) & 0 \\ & \iddots & & & \ddots \\ P_{N-1}\left(-z^2\right) & & 0 & & z^{-1}P_N\left(-z^2\right) \end{bmatrix} . \tag{2.50}$$

This form of $\mathbf{F}_a(z)$ assumes that the window length factor L is even, which can always be obtained by appending zeros. The filter matrix $\mathbf{F}_a(z)$ has a bi-diagonal structure, i.e., it has nonzero elements not only on the diagonal but also on the antidiagonal. This means a window function can be designed such that the inverse of the filter matrix leads to FIR filters. An example is the classical MDCT or TDAC filter bank [37]. It results from inserting an additional phase shift of $N/2$ in the modulating cosine function:

$$\mathbf{X}_{k,m} = \sum_{n=0}^{2N-1} \cos\left(\frac{\pi}{N}(k+0.5)(n+0.5+N/2)\right) \cdot p(LN-1-n)\, x(mN+n) \; .$$

This phase shift leads to a shift of the structure of the filter matrix downwards by $N/2$. For example, for a window function $p(n)$ for $n = 0, \ldots, 2N-1$, the filter matrix has the following form,

$$\mathbf{F}_a(z) = \begin{bmatrix} 0 & z^{-1}p(1.5N) & z^{-1}p(1.5N-1) & 0 \\ & \iddots & & \ddots \\ z^{-1}p(2N-1) & & 0 & z^{-1}p(N) \\ p(N-1) & & & -p(0) \\ & \ddots & & \iddots \\ 0 & p(N/2) & -p(N/2-1) & 0 \end{bmatrix}$$

The inverse for the synthesis matrix is

$$z^{-1}\mathbf{F}_a^{-1}(z) = \begin{bmatrix} 0 & q(0) & z^{-1}q(N) & 0 \\ & \iddots & & \ddots \\ q(N/2-1) & & 0 & z^{-1}q(1.5N-1) \\ q(N/2) & & & -z^{-1}q(1.5N) \\ & \ddots & & \iddots \\ 0 & q(N-1) & -z^{-1}q(2N-1) & 0 \end{bmatrix}$$

with

$$q(n) = \frac{p(n)}{p(2N-1-n)p(n) + p(N-1-n)p(N+n)}$$
$$q(N+n) = \frac{p(N+n)}{p(2N-1-n)p(n) + p(N-1-n)p(N+n)}$$

where $n = 0, \ldots, N-1$. This inverse is used in the synthesis filter bank to reconstruct the signal, e.g., in a decoder. The synthesis side has a filter matrix with the same shape as the analysis side, so the synthesis filter bank is again a cosine-modulated filter bank, with $q(n)$ as its window function. Observe that $q(n) = p(n)$ if the denominator for the computation of the inverse becomes one.

The DCT leads to a filter matrix which has a form enabling us to design filter banks with critical sampling and FIR filters for analysis as well as for the synthesis. Therefore filter banks based on DCTs are the predominant tools for time-frequency decomposition in audio coding.

To design filter banks with longer filters and more freedom in the design process, the filter matrix Eq. (2.50) can be written as a product of simpler matrices. These simpler matrices can be unitary, such that the product is a unitary matrix, whose inverse is then obtained by simply transposing it and replacing z by z^{-1} [31]. Or these simpler matrices can be bi-orthogonal, so that the resulting filter bank is bi-orthogonal [45]. The latter is a more general solution, which enables us to design, for example, filter banks with a lower end-to-end delay than unitary or orthogonal filter banks [43, 44].

2.9.2 Complex DFT-Based Filter Banks

A disadvantage of the DCT is that it delivers no phase or magnitude information, as the DFT does. For example, in audio coding the magnitudes of the subband signals are needed as inputs to psycho-acoustic models which control the quantization process, as seen in Fig. 2.18. Such is the basic structure of, for example, the PAC audio coder. The DCT can be seen as the real part of a DFT of a real valued signal, so what is needed is the imaginary part to obtain complex subband signals and hence their magnitudes. The imaginary part can be obtained by using a filter bank based on a DST. For a cosine-modulated filter bank with a DCT IV, the corresponding sine-modulated filter bank uses a DST IV (2.30). The equality

$$\sin\left(\frac{\pi}{N}(k+0.5)(n+0.5)\right) = \cos\left(\frac{\pi}{N}(k+0.5)(n-N+0.5)\right)$$

shows, that the sine modulation function has the same symmetries in time n as the cosine modulation function [Eqs. (2.48) and (2.49)] but is shifted by N samples. This leads to the same conditions on the window function for perfect reconstruction, so the same window function can be used for the cosine- and the sine-modulated filter banks, hence for the real and imaginary parts of the resulting complex valued filter bank. This is important for obtaining the precise magnitude and phase information of a signal.

In audio coding, the signal consists of real values. If the input signal to the complex filter bank consists of complex values, as in applications such as synthetic aperture radar (SAR) [30], the filter bank needs to cover positive as well as negative frequencies to obtain perfect reconstruction. If AFB_C is the output of the cosine-modulated analysis filter bank, and AFB_S is the output of the sine-modulated filter bank, then the positive frequencies are obtained by taking $AFB_C - jAFB_S$ and the negative frequencies by $AFB_C + jAFB_S$, similar to the DFT. This means the analysis filter bank consists of $2N$ bands

$$[AFB_C - jAFB_S, AFB_C + jAFB_S] \ .$$

The synthesis filter bank for perfect reconstruction has an analogous structure,

$$[SFB_C + jSFB_S, SFB_C - jSFB_S] \ ,$$

where SFB_C, SFB_S are the outputs of the synthesis filter banks.

It is easy to see that this synthesis filter bank leads to perfect reconstruction if the cosine and sine filter banks have the perfect reconstruction property of their own. Observe that this is not the only solution for perfect reconstruction since the filter bank is, in effect, oversampled at twice the rate. But this solution for the synthesis has an advantage because it has an analogous structure, hence similar properties, as the analysis part, which is often desirable in coding applications.

Figs. 2.21–2.23 show a comparison of the frequency responses of the window functions of a direct FFT approach, as used in the MPEG-AAC audio coder as input for the psycho-acoustic model, and the complex filter bank. Fig. 2.22 shows the frequency response of a 1024 band FFT filter bank, and Fig. 2.21 shows the frequency response of a complex low-delay filter bank with 1024 bands, an analysis/synthesis delay of 2047 samples, and filter length of 4096 taps. Figs. 2.23 and 2.24 show an enlargement with the passband on the left. The passband of the complex filter bank is narrower, and the stopband attenuation is much higher than with the direct FFT application.

Figs. 2.25–2.27 show an application example for a stereo audio signal that is encoded and decoded at two different bit rates. Fig. 2.25 shows a piece of the original audio signal (jazz music), the left channel, sampled at 32000 samples/s. In this uncompressed representation, each sample is represented with a 16 bit integer number, which leads to a bit rate of $16 \cdot 2 \cdot 32000 = 1024$ kb/s. Fig. 2.26 shows that signal, but coded and decoded with a bit rate of 67 kb/s for the stereo signal (i.e., 35 kb/s per channel, or a compression ratio of over 14). The resulting audio quality is comparable to FM radio. It can be seen that there are slight differences to the original, but most of the differences are still inaudible because of the application of the psycho-acoustic model. Fig. 2.27 shows the signal at 30 kb/s stereo (a compression ratio of over 34). The resulting quality is comparable to AM radio. There are now more pronounced differences to the original; it is much smoother, which means it contains fewer high frequencies. Here the difference to the original is easy to hear, but the psycho-acoustic model is used such that the audible distortions are minimized.

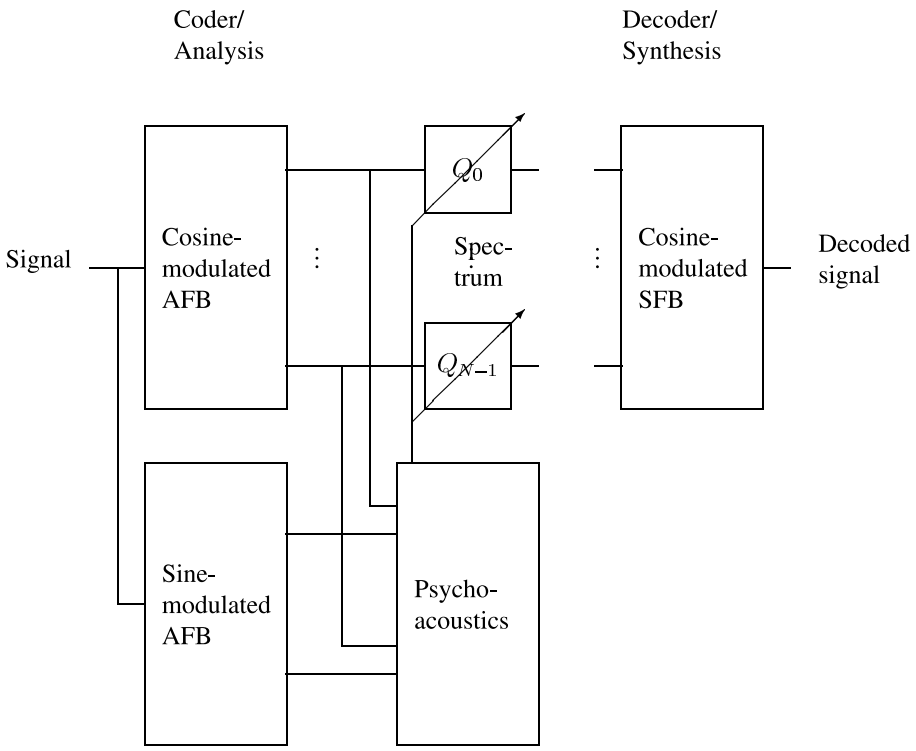


FIGURE 2.20
Audio coding based on filter banks, AFB: analysis filter bank, SFB: synthesis filter bank.

2.10 Conclusion

This chapter introduced the DFT and some of its basic properties. Even though it is a complex-valued transform, because of its symmetry properties, the DFT of a real-valued N-point signal can be represented again by N real values. A set of real-valued discrete cosine transforms can be derived using the DFT. The derivation of a fast algorithm for computing the DFT (the FFT) was also described here.

The DFT has many applications in coding. For example, the FFT is used for the efficient implementation of DCTs, the MDCT, and low delay filter banks. Furthermore, the complex output is used for power spectrum estimation, in particular, to drive psycho-acoustic models in audio coding, and it can be used to implement complex-valued filter banks for improved power spectrum estimation.

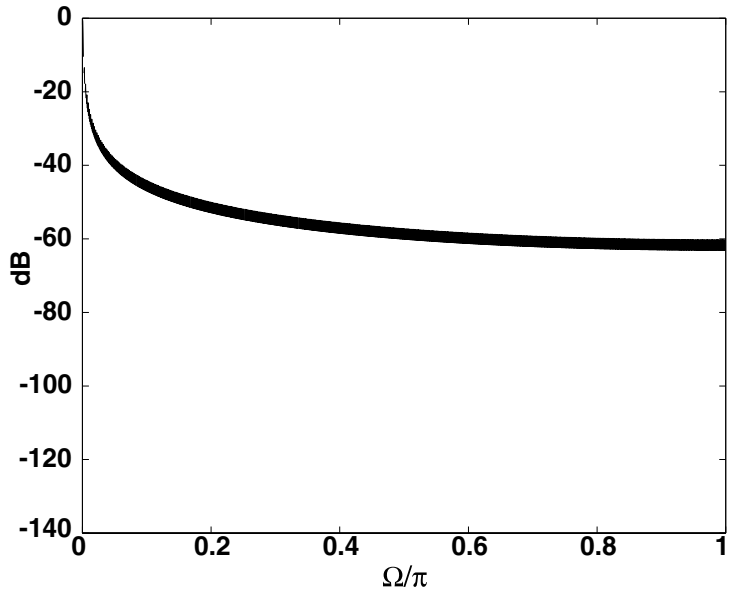


FIGURE 2.21
Magnitude of the frequency response of the rectangular window of a DFT of length 1024.

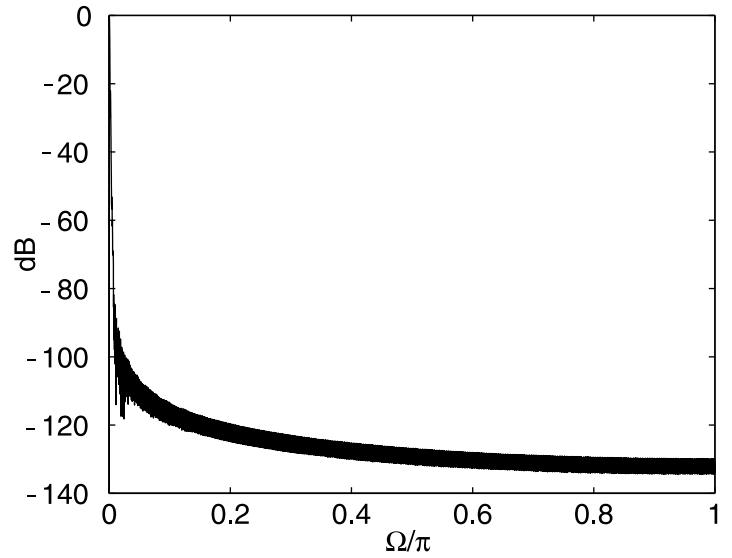


FIGURE 2.22
Magnitude of the frequency response of the window of a low delay filter bank with 1024 bands and filter length 4096.

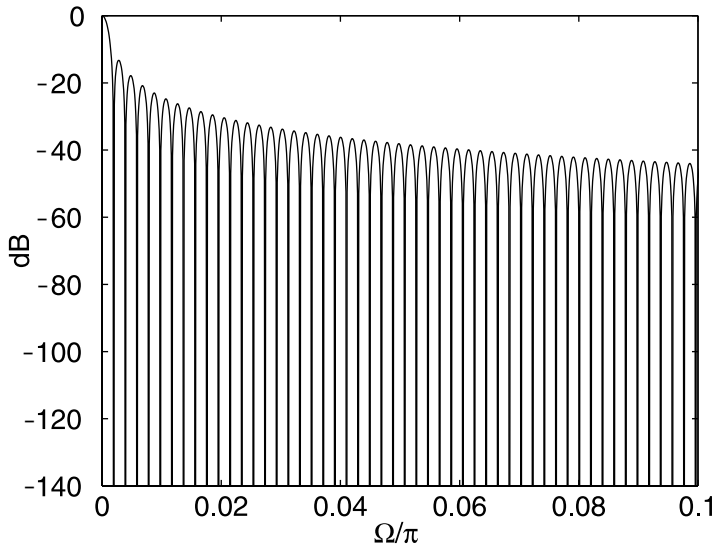


FIGURE 2.23
Enlargement of the first part of the magnitude of the frequency response of the rectangular window of the DFT.

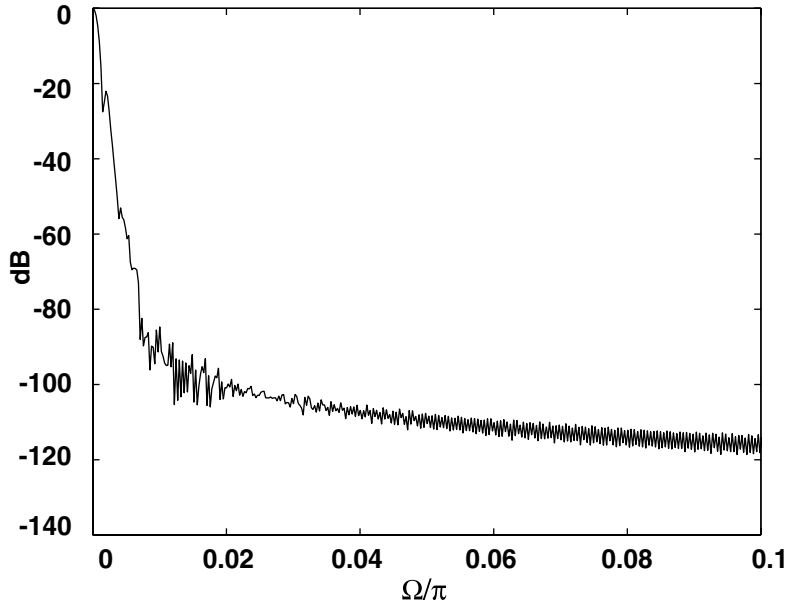


FIGURE 2.24
Enlargement of the first part of the magnitude of the frequency response of the window of the low delay filter bank.

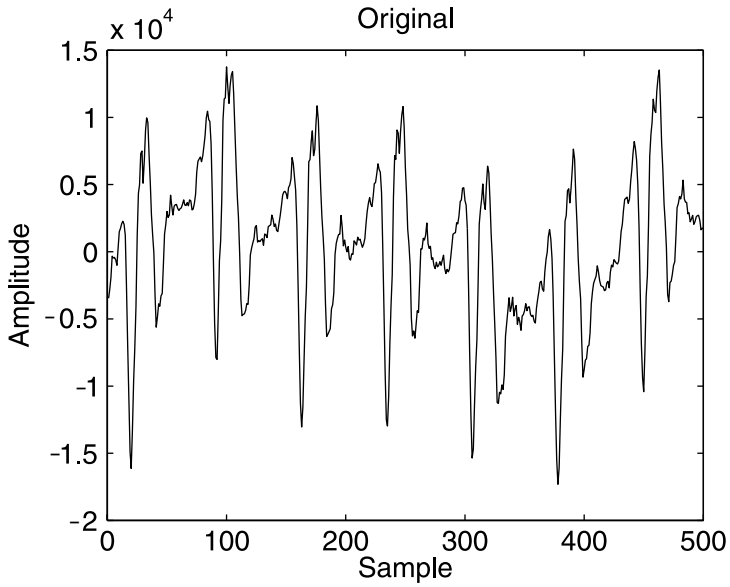


FIGURE 2.25
A piece of an example audio signal, sampled at 32 khz. Shown is the left channel of the stereo signal.

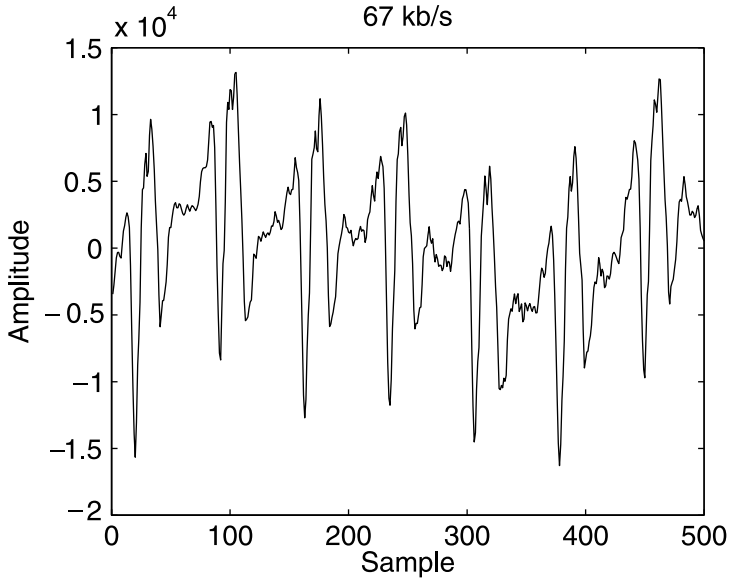


FIGURE 2.26
The stereo audio signal, coded and decoded with 67 kb/s. The left channel is shown.

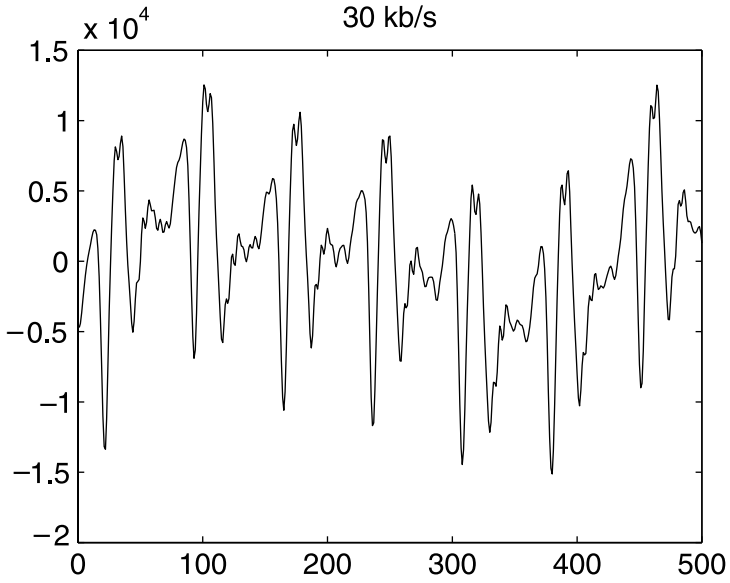


FIGURE 2.27
The left channel of the stereo audio signal, coded and decoded, but with 30 kb/s.

2.11 FFT Web sites

The following list reflects some of the available software and information on Web sites devoted to the FFT (September 1999).

- FFTW
  `http://www.fftw.org/index.html`
  `http://www.fftw.org/benchfft/doc/ffts.html`
- FFTPACK
  `http://www.netlib.org/fftpack/`
- FFT for Pentium (Bernstein)
  `ftp://koobera.math.uic.edu/www/djbfft.html`
- FFT software (comp.speech FAQ Q2.4)
  `http://svr-www.eng.cam.ac.uk/comp.speech/Section2/Q2.4.html`
- One-dimensional real fast Fourier transforms
  `http://www.hr/josip/DSP/fft.html`

- FXT package FFT code (Arndt)
  http://www.jjj.de/fxt/
- FFT (Don Cross)
  http://www.intersrv.com/~dcross/fft.html
- Public domain FFT code
  http://risc1.numis.nwu.edu/ftp/pub/transforms/
  http://risc1.numis.nwu.edu/fft/
- DFT (Paul Bourke)
  http://www.swin.edu.au/astronomy/pbourke/
  sigproc/dft/
- FFT code for TMS320 processors
  ftp://ftp.ti.com/mirrors/tms320bbs/
- Fast Fourier Transforms (Kifowit)
  http://ourworld.compuserve.com/homepages/
  steve_kifowit/fft.htm
- Nielsen's MIXFFT page
  http://home.get2net.dk/jjn/fft.htm
- Parallel FFT homepage
  http://www.arc.unm.edu/Workshop/FFT/fft/fft.html
- FFT public domain algorithms
  http://www.arc.unm.edu/Workshop/FFT/fft/fft.html
- Numerical recipes
  http://www.nr.com/
- General purpose FFT package
  http://momonga.t.u-tokyo.ac.jp/õoura/fft.html
- FFT links
  http://momonga.t.u-tokyo.ac.jp/õoura/fftlinks.html
- FFT, performance, accuracy, and code (Mayer)
  http://www.geocities.com/ResearchTriangle/8869/
  fft_summary.html
- Prime-length FFT
  http://www.dsp.rice.edu/software/RU-FFT/
  pfft/pfft.html
- Notes on the FFT (Burrus)
  http://www.dsp.rice.edu/research/fft/fftnote.asc

- Yahoo FFT Web site list
  `http://dir.yahoo.com/Science/Mathematics/Software/Fast_Fourier_Transform__FFT_/`

References

[1] Ahmed, N., Natarajan, T., and Rao, K.R., Discrete cosine transform, *IEEE Trans. Comput.,* 23:90–93, 1974, also in [41].

[2] Bellanger, M., *Digital Processing of Signals, Theory and Practice,* John Wiley & Sons, Chichester, NY, 1989.

[3] Bergland, G.D., A guided tour of the fast Fourier transform, *IEEE Spectrum,* 6:41–52, 1969, also in [39].

[4] Blahut, R.E., *Fast Algorithms for Digital Signal Processing.* Addison-Wesley, Reading, MA, 1985.

[5] Bracewell, R.N., *The Fourier Transform and its Applications,* McGraw Hill, Reading, MA, 1986.

[6] Brandenburg, K. and Bosi, M., Overview of MPEG audio: current and future standards for low bit rate audio coding, *J. Audio Eng. Soc.,* 45(1/2):4–21, 1997.

[7] Brigham, E.O., *The Fast Fourier Transform and its Applications,* Prentice-Hall, Englewood Cliffs, NJ, 1988.

[8] Burrus, C.S., Efficient Fourier transform and convolution algorithms, in Lim, J.S. and Oppenheim, A.V., Eds., *Advanced Topics in Signal Processing,* Prentice-Hall, Englewood Cliffs, NJ, 1988.

[9] Burrus, C.S. and Eschenbacher, P.W., An in-place, in-order prime factor FFT algorithm, *IEEE Trans. on Acoust., Speech, Signal Proc.,* 29(4):806–817, 1981.

[10] Burrus, C.S. and Parks, T.W., *DFT/FFT and Convolution Algorithms,* John Wiley & Sons, Chichester, NY, 1985.

[11] Cochran, J.W., Favin, D.L., Helms, H.D., Kaenel, R.A., Lang, W.W., Maling, G.C., Nelson, D.E., Rader, C.M., and Welch, P.D., What is the fast Fourier transform?, *IEEE Trans. Audio Electroacoust.,* 15:45–55, 1967, also in [39].

[12] Cohen, L., *Time-Frequency Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1995.

[13] Cooley, J.W., Lewis, P.A.W., and Welch, P.D., Historical notes on the fast Fourier transform, *IEEE Trans. Audio Electroacoust.,* 15:76–79, 1967, also in [39].

[14] Cooley, J.W., Lewis, P.A.W., and Welch, P.D., The finite Fourier transform, *IEEE Trans. Audio Electroacoust.,* 17:77–85, 1969, also in [39].

[15] Cooley, J.W. and Tukey, J.W., An algorithm for the machine calculation of complex Fourier series, *Mathematics of Computation,* 19(90):297–301, 1965, also in [39].

[16] Duhamel, P., Implementation of "split radix" FFT algorithm, *IEEE Trans. on Acoust., Speech, Signal Proc.,* 34(2):285–295, 1986.

[17] Duhamel, P. and Vetterli, M., Fast Fourier transforms: a tutorial review and a state of the art, *Signal Processing,* 19:259–299, 1990, also in [18].

[18] Duhamel, P. and Vetterli, M., Fast Fourier transforms: a tutorial review and a state of the art, in Madisetti, V.K. and Williams, D.B., Eds., *The Digital Signal Processing Handbook,* chapter 7, CRC Press, 1998, also appears as [17].

[19] Elliott, D.F., Fast Fourier transforms, in Elliott, D.F., Ed., *Handbook of Digital Signal Processing,* Chapter 7, pages 527–631, Academic Press, New York, 1987.

[20] Elliott, D.F. and Rao, K.R., *Fast Transforms: Algorithms, Analyses, Applications,* Academic Press, New York, 1982.

[21] Frigo, M. and Johnson, S.G., FFTW, FFT software developed at MIT, `http://www.fftw.org/index.html`.

[22] Heideman, M.T., *Multiplicative Complexity, Convolution, and the DFT,* Springer-Verlag, New York, Berlin, 1988.

[23] Johnson, H.W. and Burrus, C.S., The design of optimal DFT algorithms using dynamic programming, *IEEE Trans. on Acoust., Speech, Signal Proc.,* 31(2):378–387, 1983.

[24] Johnson, J., Automatic implementation and generation of FFT algorithms, SIAM parallel processing FFT session, March 1999, see SPIRAL webpage `http://www.ece.cmu.edu/~spiral/`.

[25] Jones, K.J., Prime number DFT computation via parallel circular convolvers, *IEE Proceedings, Part F,* 137(3):205–212, 1990.

[26] Karp, T. and Fliege, N.J., MDFT filter banks with perfect reconstruction, in *IEEE International Symposium on Circuits and Systems,* Seattle, WA, 1995.

[27] Lim, J.S. and Oppenheim, A.V., *Advanced Topics in Signal Processing,* Prentice-Hall, Englewood Cliffs, NJ, 1988.

[28] Van Loan, C., *Computational Frameworks for the Fast Fourier Transform,* SIAM, Philadelphia, PA, 1992.

[29] Lu, C., Cooley, J.W., and Tolimieri, R., FFT algorithms for prime transform sizes and their implementations on VAX, IBM3090VF, and IBM RS/6000, *IEEE Trans. on Acoust., Speech, Signal Proc.,* 41(2):638–648, 1993.

[30] Madisetti, V.K. and Williams, D.B., *The Digital Signal Processing Handbook,* CRC Press and IEEE Press, Boca Raton, FL, 1997.

[31] Malvar, H., *Signal Processing with Lapped Transforms,* Artech House, Boston, MA, London, 1992.

[32] Martucci, S., Symmetric convolution and the discrete sine and cosine transforms, *IEEE Trans. on Signal Processing,* 42:1038–1051, 1994.

[33] McClellan, J.H. and Rader, C.M., *Number Theory in Digital Signal Processing,* Prentice-Hall, Englewood Cliffs, NJ, 1979.

[34] Meyer, R. and Schwarz, K., FFT implementation on DSP-chips, theory and practice, in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing (ICASSP),* 1503–1506, April 1990.

[35] Myers, D.G., *Digital Signal Processing: Efficient Convolution and Fourier Transform Techniques,* Prentice-Hall, Englewood Cliffs, NJ, 1990.

[36] Nussbaumer, H.J., *Fast Fourier Transform and Convolution Algorithms,* Springer-Verlag, New York, Berlin, 1982.

[37] Princen, J.P. and Bradley, A.B., Analysis/synthesis filter bank design based on time domain aliasing cancellation, *IEEE Trans. on Signal Processing,* 34(10):1153–1161, 1986.

[38] Proakis, J.G., Rader, C.M., Ling, F., and Nikias, C.L., *Advanced Digital Signal Processing,* Macmillan, New York, 1992.

[39] Rabiner, L.R. and Rader, C.M., Eds., *Digital Signal Processing,* IEEE Press, Piscataway, NJ, 1972.

[40] Rader, C.M., Discrete Fourier transform when the number of data samples is prime, *Proc. IEEE*, 56(6):1107–1108, 1968.

[41] Rao, K.R., Ed., *Discrete Transforms and Their Applications,* Krieger, Malabar, FL, 1990.

[42] Rao, K.R. and Yip, P., *Discrete Cosine Transform: Algorithms, Advantages, Applications,* Academic Press, New York, 1990.

[43] Schuller, G., Time-varying filter banks with variable system delay, in *Proc. IEEE ICASSP*, Vol. 3, 2469–2472, Munich, Germany, 1997.

[44] Schuller, G. and Karp, T., Modulated filter banks with arbitrary system delay: efficient implementations and the time-varying case, *IEEE Trans. on Signal Processing,* 48(3), 2000.

[45] Schuller, G.D.T. and Smith, M.J.T., New framework for modulated perfect reconstruction filter banks, *IEEE Trans. on Signal Processing,* 44(8):1942–1954, 1996.

[46] Selesnick, I.W. and Burrus, C.S., Automatic generation of prime length FFT programs, *IEEE Trans. on Signal Processing,* 44(1):14–24, 1996.

[47] Sinha, D., Johnston, J.D., Dorward, S., and Quackenbush, S., The perceptual audio coder (PAC), in Madisetti, V. and Williams, D.B., Eds., *The Digital Signal Processing Handbook,* chapter 42, CRC Press and IEEE Press, Boca Raton, FL, 1997.

[48] Smith, W.W. and Smith, J.M., *Handbook of Real-Time Fast Fourier Transforms,* IEEE Press, Piscataway, NJ, 1995.

[49] Sorensen, H.V. and Burrus, C.S., Fast DFT and convolution algorithms, in Mitra, S.K. and Kaiser, J.F., Eds., *Handbook For Digital Signal Processing,* chapter 8, 491–610, John Wiley & Sons, New York, 1993.

[50] Sorenson, H.V., Heideman, M.T., and Burrus, C.S., On computing the split-radix FFT, *IEEE Trans. on Acoust., Speech, Signal Proc.,* 34(1):152–156, 1986.

[51] Sorenson, H.V., Jones, D.L., Heideman, M.T., and Burrus, C.S., Real-valued fast Fourier transform algorithms, *IEEE Trans. on Acoust., Speech, Signal Proc.,* 35(6):849–863, 1987.

[52] Temperton, C., Implementation of a self-sorting in-place prime factor FFT algorithm, *J. of Computational Physics,* 58:283–299, 1985.

[53] Temperton, C., Self-sorting in-place fast Fourier transforms, *SIAM J. on Scientific and Statistical Computing,* 12(4):808–823, 1991.

[54] Tolimieri, R., An, M., and Lu, C., *Algorithms for Discrete Fourier Transform and Convolution,* Springer-Verlag, New York, Berlin, 1989.

[55] Vaidyanathan, P.P., *Multirate Systems and Filter Banks,* Prentice-Hall, Englewood Cliffs, NJ, 1992.

[56] Vetterli, M. and Duhamel, P., Split-radix algorithms for length-p^m DFTs, *IEEE Trans. on Acoust., Speech, Signal Proc.,* 37(1):57–64, 1989.

[57] Vetterli, M. and Kovačević, J., *Wavelets and Subband Coding,* Prentice-Hall, Englewood Cliffs, NJ, 1995.

[58] Wickershauser, M.L., *Adapted Wavelet Analysis from Theory to Software,* A.K. Peters, Wellesley, MA, 1994.

[59] Winograd, S., Some bilinear forms whose multiplicative complexity depends on the field of constants, *Mathematical Systems Theory,* 10:169–180, 1977.

[60] Winograd, S., *Arithmetic Complexity of Computations,* SIAM, Philadelphia, PA, 1980.

[61] Yip, P. and Rao, K.R., Fast discrete transforms, in Elliott, D.F., Ed., *Handbook of Digital Signal Processing,* chapter 6, 481–525, Academic Press, New York, 1987.