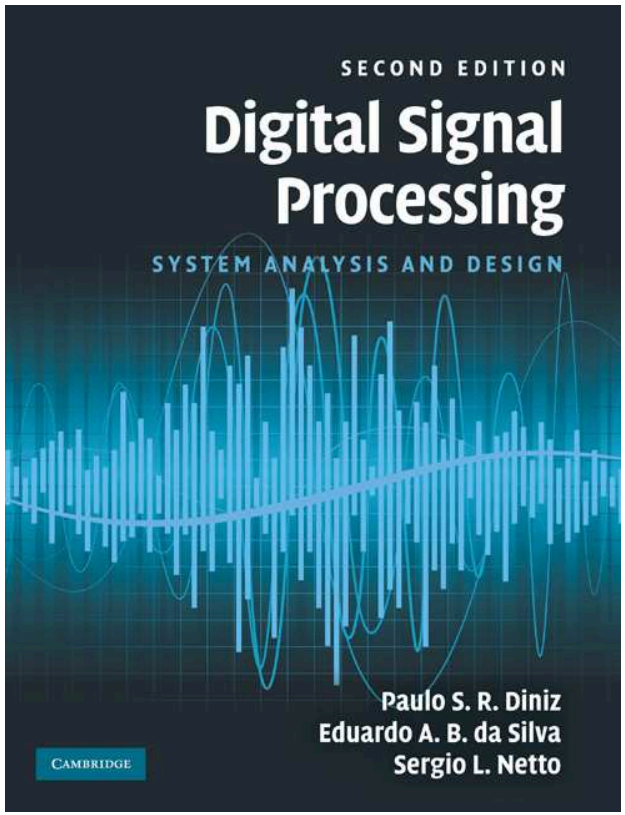
Discrete Transforms



Paulo S. R. Diniz

Eduardo A. B. da Silva

Sergio L. Netto

diniz,eduardo,sergioln@lps.ufrj.br

September 2010

Contents

Introduction

- In Chapter 2, we saw that discrete-time signals and systems can be characterized in the frequency domain by their Fourier transform.
- Also, as seen in Chapter 2, one of the main advantages of discrete-time signals is that they can be processed and represented in digital computers.
- However, when we examine the definition of the Fourier transform in equation (**??**),

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \tag{1}$$

  we notice that such a characterization in the frequency domain depends on the continuous variable ω.
  - This implies that the Fourier transform, as it is, is not suitable for the processing of discrete-time signals in digital computers.
- We need a transform depending on a discrete-frequency variable that, if possible, preserves the idea and information expressed by equation (1).

Introduction

- This can be obtained from the Fourier transform itself in a very simple way, by sampling uniformly the continuous-frequency variable ω.
  - With this, we obtain a mapping of a signal depending on a discrete-time variable n to a transform depending on a discrete-frequency variable k.
- Such a mapping is referred to as the discrete Fourier transform (DFT).

Discrete Fourier transform

- The Fourier transform of a sequence $x(n)$ is given by equation (1).
- Since $X(e^{j\omega})$ is periodic with period 2π, it is convenient to sample it with a sampling frequency equal to an integer multiple of its period, that is, taking N uniformly spaced samples between 0 and 2π.
  - Let us use the frequencies $\omega_k = \frac{2\pi}{N}k$, $k \in \mathbb{Z}$.
- This sampling process is equivalent to generating a Fourier transform $X'(e^{j\omega})$ such that

$$X'(e^{j\omega}) = X(e^{j\omega}) \sum_{k=-\infty}^{\infty} \delta\left(\omega - \frac{2\pi}{N}k\right) \tag{2}$$

Applying the convolution theorem seen in Chapter 2, the above equation becomes

$$x'(n) = \mathcal{F}^{-1}\left\{X'(e^{j\omega})\right\} = x(n) * \mathcal{F}^{-1}\left\{\sum_{k=-\infty}^{\infty} \delta\left(\omega - \frac{2\pi}{N}k\right)\right\} \tag{3}$$

Discrete Fourier transform

and since

$$\mathcal{F}^{-1}\left\{\sum_{k=-\infty}^{\infty}\delta\left(\omega-\frac{2\pi}{N}k\right)\right\}=\frac{N}{2\pi}\sum_{p=-\infty}^{\infty}\delta\left(n-Np\right) \tag{4}$$

equation (3) becomes

$$x'(n)=x(n)*\frac{N}{2\pi}\sum_{p=-\infty}^{\infty}\delta\left(n-Np\right)=\frac{N}{2\pi}\sum_{p=-\infty}^{\infty}x\left(n-Np\right) \tag{5}$$

- Equation (5) indicates that, from equally spaced samples of the Fourier transform, we are able to recover a signal $x'(n)$ consisting of a sum of periodic repetitions of the original discrete signal $x(n)$.

Discrete Fourier transform

- In this case, the period of the repetitions is equal to the number, N, of samples of the Fourier transform in one period.
  - It is then easy to see that if the length, L, of $x(n)$ is larger than N, then $x(n)$ can not be obtained from $x'(n)$.
  - On the other hand, if $L \leq N$, then $x'(n)$ is a precise periodic repetition of $x(n)$, and therefore $x(n)$ can be obtained by isolating one complete period of N samples of $x'(n)$, as given by

$$x(n) = \frac{2\pi}{N} x'(n), \quad \text{for } 0 \leq n \leq N-1 \tag{6}$$

- From the above discussion, we can draw two important conclusions:
  - The samples of the Fourier transform can provide an effective discrete representation, in frequency, of a finite-length discrete-time signal.
  - This representation is useful only if the number of samples, N, of the Fourier transform in one period is greater than or equal to the original signal length L.

Discrete Fourier transform

- It is interesting to note that equation (5) is of the same form the equation that gives the Fourier transform of a sampled continuous-time signal.
  - That equation implies that, in order for a continuous-time signal to be recoverable from its samples, aliasing has to be avoided.
    - This can be done by using a sampling frequency greater than two times the bandwidth of the analog signal.
  - Similarly, equation (5) implies that one can recover a digital signal from the samples of its Fourier transform provided that the signal length, L, is smaller than or equal to the number of samples, N, taken in one complete period of its Fourier transform.

Discrete Fourier transform

- We can express $x(n)$ directly as a function of the samples of $X(e^{j\omega})$ by manipulating equation (2) in a different manner to that described above. We begin by noting that equation (2) is equivalent to

$$X'(e^{j\omega}) = \sum_{k=-\infty}^{\infty} X\left(e^{j\frac{2\pi}{N}k}\right)\delta\left(\omega - \frac{2\pi}{N}k\right) \tag{7}$$

By applying the inverse Fourier transform relation, we have

$$\begin{aligned} x'(n) &= \frac{1}{2\pi}\int_0^{2\pi} X'(e^{j\omega})e^{j\omega n}d\omega \\ &= \frac{1}{2\pi}\int_0^{2\pi} \sum_{k=-\infty}^{\infty} X\left(e^{j\frac{2\pi}{N}k}\right)\delta\left(\omega - \frac{2\pi}{N}k\right)e^{j\omega n}d\omega \\ &= \frac{1}{2\pi}\sum_{k=0}^{N-1} X\left(e^{j\frac{2\pi}{N}k}\right)e^{j\frac{2\pi}{N}kn} \end{aligned} \tag{8}$$

Discrete Fourier transform

- Substituting equation (8) in equation (6), $x(n)$ can be expressed as

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X\left(e^{j\frac{2\pi}{N}k}\right) e^{j\frac{2\pi}{N}kn}, \quad \text{for } 0 \leq n \leq N-1 \tag{9}$$

  - Equation (9) shows how a discrete-time signal can be recovered from its discrete-frequency representation.
  - This relation is known as the inverse discrete Fourier transform (IDFT).

- An inverse expression to equation (9), relating the discrete-frequency representation to the discrete-time signal can be obtained by just rewriting equation (1) for the frequencies $\omega_k = \frac{2\pi}{N}k$, for $k = 0, 1, \ldots, (N-1)$.
  - Since $x(n)$ has finite duration, we assume that its nonzero samples are within the interval $0 \leq n \leq N-1$, and then

$$X\left(e^{j\frac{2\pi}{N}k}\right) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn}, \quad \text{for } 0 \leq k \leq N-1 \tag{10}$$

  - Equation (10) is known as the discrete Fourier transform (DFT).

Discrete Fourier transform

- It is important to point out that, in equation (10), if $x(n)$ has length $L < N$, it has to be padded with zeros up to length N, to adapt the sequence length when calculating its corresponding DFT.
- Referring to Figure 1, we can see that the amount of zero-padding also determines the frequency resolution of the DFT.
  - Figure 1a, shows a signal $x(n)$ consisting of $L = 6$ samples along with its Fourier transform, which distinctively presents two pairs of close peaks within $[0, 2\pi)$.
  - In Figure 1b, we see that sampling the Fourier transform with 8 samples in the interval $[0, 2\pi)$ is equivalent to computing the DFT of $x(n)$ using equation (10) with $N = 8$ samples, which requires $x(n)$ to be padded with $N - L = 2$ zeros.
    - From this figure, we observe that in such a case the two close peaks of the Fourier transform of $x(n)$ could not be resolved by the DFT coefficients.

Discrete Fourier transform

- This fact indicates that the resolution of the DFT should be improved by increasing the number of samples, N, thus requiring $x(n)$ to be padded with even more zeros.
  - In Figure 1c, we can see that the close peaks can be easily identified by the DFT coefficients when N has been increased to 32, corresponding to a zero-padding of $N - L = 32 - 6 = 26$ zeros in $x(n)$.
- We can summarize the issues in the above discussion as follows:
  - The larger the number of zeros padded on $x(n)$ for the calculation of the DFT, the more it resembles its Fourier transform. This happens because of the larger number of samples taken within $[0, 2\pi)$.
  - The amount of zero-padding used depends on the arithmetic complexity allowed by a particular application, because the larger the amount of zero-padding the greater the computational and storage requirements involved in the DFT computation.

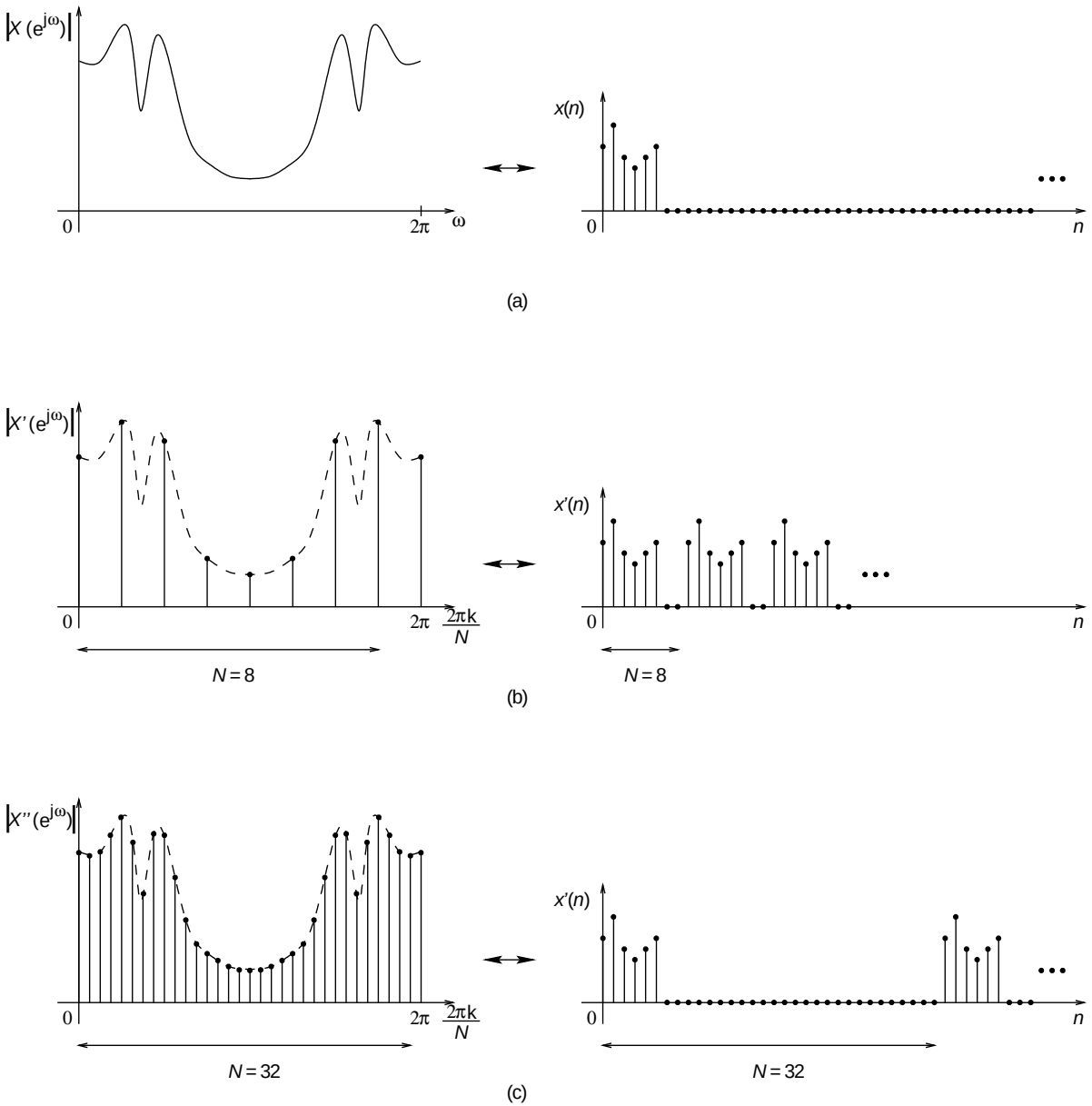

|X (e^jω)|
0
2π
ω
x(n)
0
n
(a)
|X′(e^jω)|
0
2π
2πk/N
N = 8
x′(n)
0
n
N = 8
(b)
|X″(e^jω)|
0
2π
2πk/N
N = 32
x′(n)
0
n
N = 32
(c)

Figure 1:

Discrete Fourier transform

- There is, however, an important observation that has to be made about the discussion related to Figure 1.
  - We have seen in Figure 1b that for $N = 8$ the DFT could not resolve the two close peaks of the Fourier transform.
  - However, since the signal duration is $N = 6$, $x(n)$ can be recovered from the samples of the Fourier transform in Figure 1b using equation (9).
  - With $x(n)$, the Fourier transform $X(e^{j\omega})$, as in Figure 1a, can be computed using equation (1), and therefore the two close peaks could be fully recovered and identified.
- At this point one could ask why one would need to use a DFT of larger resolution if a DFT of size equal to the signal duration would be enough to fully recover the correct Fourier transform.

Discrete Fourier transform

- A justification for the use of a DFT of a larger size than the signal duration is that, for example, for N large enough, one would not need to perform indirect calculations to identify the two close peaks in Figure 1, because they would be represented directly by the DFT coefficients, as depicted in Figure 1c.
- In what follows, we derive an expression relating the Fourier transform of a signal $x(n)$ to its DFT.

Discrete Fourier transform

- Equation (6), which relates the signal $x(n)$ to the signal $x'(n)$ obtainable from the samples of its Fourier transform, can be rewritten as

$$x(n) = \frac{2\pi}{N} x'(n)(u(n) - u(n-N)) \tag{11}$$

- Using the fact that a multiplication in the time domain corresponds to a periodic convolution in the frequency domain, discussed in Chapter 2, we have

$$\begin{aligned} X(e^{j\omega}) &= \frac{1}{2\pi}\frac{2\pi}{N} X'(e^{j\omega}) \circledast \mathcal{F}\{u(n) - u(n-N)\} \\ &= \frac{1}{N} X'(e^{j\omega}) \circledast \left[\frac{\sin\frac{\omega N}{2}}{\sin\frac{\omega}{2}} e^{-j\frac{\omega(N-1)}{2}} \right] \end{aligned} \tag{12}$$

Discrete Fourier transform

- Substituting equation (7) in equation (12), $X(e^{j\omega})$ becomes

$$X(e^{j\omega}) = \frac{1}{N}\left[\sum_{k=-\infty}^{\infty} X\left(e^{j\frac{2\pi}{N}k}\right)\delta\left(\omega - \frac{2\pi}{N}k\right)\right] \circledast \left[\frac{\sin\frac{\omega N}{2}}{\sin\frac{\omega}{2}}\, e^{-j\frac{\omega(N-1)}{2}}\right]$$

$$= \frac{1}{N}\sum_{k=-\infty}^{\infty} X\left(e^{j\frac{2\pi}{N}k}\right)\left[\frac{\sin\frac{(\omega-\frac{2\pi}{N}k)N}{2}}{\sin\frac{(\omega-\frac{2\pi}{N}k)}{2}}\, e^{-j\frac{(\omega-\frac{2\pi}{N}k)(N-1)}{2}}\right]$$

$$= \frac{1}{N}\sum_{k=-\infty}^{\infty} X\left(e^{j\frac{2\pi}{N}k}\right)\left[\frac{\sin\left(\frac{\omega N}{2} - \pi k\right)}{\sin\left(\frac{\omega}{2} - \frac{\pi k}{N}\right)}\, e^{-j\left(\frac{\omega}{2}-\frac{\pi}{N}k\right)(N-1)}\right] \qquad (13)$$

- This equation corresponds to an interpolation formula that gives the Fourier transform of a signal as a function of its DFT.
- One should note, once again, that such a relationship only works when N is larger than the signal length L.

Discrete Fourier transform

- In order to simplify the notation, it is common practice to use $X(k)$ instead of $X\left(e^{j\frac{2\pi}{N}k}\right)$, and to define

$$W_N = e^{-j\frac{2\pi}{N}} \tag{14}$$

- Using this notation, the definitions of the DFT and IDFT, as given in equations (10) and (9), become

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad \text{for } 0 \leq k \leq N-1 \tag{15}$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn}, \quad \text{for } 0 \leq n \leq N-1 \tag{16}$$

respectively.

Discrete Fourier transform

- From the development represented by equations (7)–(10), it can be seen that equation (16) is the inverse of equation (15).
  (This can be shown in an alternative way by substituting equation (16) into equation (15) directly.)
- One should note that if, in the above definitions, $x(n)$ and $X(k)$ are not restricted to being between 0 and $N-1$, then they should be interpreted as being periodic sequences with period N.

Discrete Fourier transform

- From the above, the discrete Fourier transform as expressed in equation (15) can be interpreted in two related ways:
  - As a discrete-frequency representation of finite-length signals whereby a length-N signal is mapped into N discrete-frequency coefficients, which correspond to N samples of the Fourier transform of $x(n)$.
  - As the Fourier transform of a periodic signal having period N. This periodic signal may correspond to the finite-length signal $x(n)$ repeated periodically, not restricting the index n in equation (16) to the interval $0 \leq n \leq N-1$.
- By referring to Chapter 2, one can see that the DFT and inverse DFT are actually a Fourier series pair for the periodic signal.

Discrete Fourier transform

Example 3.1

Compute the DFT of the following sequence:

$$x(n) = \begin{cases} 1, & 0 \leq n \leq 4 \\ -1, & 5 \leq n \leq 9 \end{cases} \tag{17}$$

Discrete Fourier transform

Solution

Before solving this example it is worthy stating a simple but important property of W_N. If k is a multiple of N, then

$$W_N^k = e^{-j\frac{2\pi}{N}k} = e^{-j\frac{2\pi}{\frac{N}{k}}} = W_{\frac{N}{k}} \tag{18}$$

We have that

$$X(k) = \sum_{n=0}^{4} W_{10}^{kn} - \sum_{n=5}^{9} W_{10}^{kn} \tag{19}$$

Discrete Fourier transform

Since $0 \leq k \leq 9$, if $k \neq 0$, we have that

$$\begin{aligned} X(k) &= \frac{1 - W_{10}^{5k}}{1 - W_{10}^{k}} - \frac{W_{10}^{5k} - W_{10}^{10k}}{1 - W_{10}^{k}} \\ &= \frac{2(1 - W_{10}^{5k})}{1 - W_{10}^{k}} \\ &= \frac{2(1 - W_{2}^{k})}{1 - W_{10}^{k}} \\ &= \frac{2(1 - (-1)^{k})}{1 - W_{10}^{k}} \end{aligned} \tag{20}$$

If $k = 0$, the summation in equation (19) becomes

$$X(k) = \sum_{n=0}^{4} W_{10}^{0} - \sum_{n=5}^{9} W_{10}^{0} = \sum_{n=0}^{4} 1 - \sum_{n=5}^{9} 1 = 0 \tag{21}$$

Discrete Fourier transform

By examining equations (20) and (21), we can see that $X(k) = 0$ for k even. Therefore, we have that the DFT can be expressed as

$$X(k) = \begin{cases} 0, & \text{for } k \text{ even} \\ \dfrac{4}{1 - W_{10}^{k}}, & \text{for } k \text{ odd} \end{cases} \tag{22}$$

△

Discrete Fourier transform

Example 3.2

Given the sequence

$$x(n) = \alpha(-a)^n \ , \quad \text{for} \quad 0 \leq n \leq N-1, \tag{23}$$

compute the length-N DFT, supposing that N is even.

Discrete Fourier transform

Solution

1. For $a \neq \pm 1$, since N is even,

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} \alpha(-a)^n W_N^{kn} \\ &= \alpha \frac{1-(-a)^N W_N^{Nk}}{1+aW_N^k} \\ &= \alpha \frac{1-a^N W_N^{Nk}}{1+aW_N^k} \\ &= \alpha \frac{1-a^N}{1+aW_N^k} \end{aligned} \tag{24}$$

Discrete Fourier transform

2. For $a = -1$, then $x(n) = \alpha$ for all n. Then

$$X(k) = \sum_{n=0}^{N-1} \alpha W_N^{kn} \tag{25}$$

Since $0 \leq k < N$, if $k = 0$, we have that

$$X(k) = \sum_{n=0}^{N-1} \alpha = \alpha N \tag{26}$$

If $k \neq 0$,

$$X(k) = \sum_{n=0}^{N-1} \alpha W_N^{kn} = \alpha \frac{1 - W_N^{Nk}}{1 + W_N^k} = 0 \tag{27}$$

From equations (26) and (27), we can write

$$X(k) = \alpha N \delta(k) \tag{28}$$

Discrete Fourier transform

3. For $a = 1$,

$$X(k) = \sum_{n=0}^{N-1} \alpha(-1)^n W_N^{kn} = \sum_{n=0}^{N-1} \alpha(-W_N^k)^n \tag{29}$$

If $k = \frac{N}{2}$, then $W_N^k = -1$, and from the above equation,

$$X(k) = \sum_{n=0}^{N-1} \alpha = \alpha N \tag{30}$$

If $k \neq \frac{N}{2}$, from equation (29), since N is even,

$$X(k) = \alpha \frac{1 - (-W_N^k)^N}{1 + W_N^k} = \alpha \frac{1 - W_N^{kN}}{1 + W_N^k} = 0 \tag{31}$$

From equations (30) and (31), we can write

$$X(k) = \alpha N \delta\left(k - \frac{N}{2}\right) \tag{32}$$

Discrete Fourier transform

- From equation (32), it can be concluded that the signal $x(n)$, for $a = 1$, has spectral contents only at $k = \frac{N}{2}$, which corresponds to $\omega = \pi$, that is the maximum normalized frequency for a discrete signal.
- On the other hand, equation (28) says that for $a = -1$ the signal $x(n)$ has spectral contents only for $k = 0$, that is, for $\omega = 0$.
  - This is indeed clear from the fact that, if $a = -1$ then $x(n) = \alpha$, that is constant and therefore has only the DC component.

$\triangle$

Discrete Fourier transform

Equations (15) and (16) can be written in matrix notation, respectively, as

$$X(k) = \begin{bmatrix} W_N^0 & W_N^k & W_N^{2k} & \cdots & W_N^{(N-1)k} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{bmatrix} \tag{33}$$

$$x(n) = \frac{1}{N} \begin{bmatrix} W_N^0 & W_N^{-k} & W_N^{-2k} & \cdots & W_N^{-(N-1)k} \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix} \tag{34}$$

and then

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^1 & \cdots & W_N^{(N-1)} \\ W_N^0 & W_N^2 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ W_N^0 & W_N^{(N-1)} & \cdots & W_N^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{bmatrix} \tag{35}$$

$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{bmatrix} = \frac{1}{N} \begin{bmatrix} W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^{-1} & \cdots & W_N^{-(N-1)} \\ W_N^0 & W_N^{-2} & \cdots & W_N^{-2(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ W_N^0 & W_N^{-(N-1)} & \cdots & W_N^{-(N-1)^2} \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix} \tag{36}$$

Discrete Fourier transform

By defining

$$\mathbf{x} = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix} \tag{37}$$

and a matrix $\mathbf{W}_N$ such that

$$\{\mathbf{W}_N\}_{ij} = W_N^{ij}, \quad \text{for } 0 \leq i, j \leq N-1 \tag{38}$$

equations (35) and (36) can be rewritten more concisely as

$$\mathbf{X} = \mathbf{W}_N \mathbf{x} \tag{39}$$

$$\mathbf{x} = \frac{1}{N} \mathbf{W}_N^* \mathbf{X} \tag{40}$$

respectively.

Discrete Fourier transform

- Note that the matrix $\mathbf{W}_N$ enjoys very special properties.
  - Equation (38) above implies that this matrix is symmetric, that is, $\mathbf{W}_N^T = \mathbf{W}_N$.
  - Also, the direct and inverse DFT relations in equations (39) and (40) imply that $\mathbf{W}_N^{-1} = \frac{1}{N}\mathbf{W}_N^*$.
- From either equations (15) and (16), or (35) and (36), one can easily conclude that a length-N DFT requires N^2 complex multiplications, including possible trivial multiplications by $W_N^0 = 1$.
  - Since the first row and the first column of the matrices in equations (35) and (36) are equal to 1, we have $(2N - 1)$ elements equal to 1.
  - Therefore, if we discount these trivial cases, the total number of multiplications is equal to $(N^2 - 2N + 1)$. Furthermore, the total number of additions is $N(N - 1)$.

Properties of the DFT

- In this section, we describe the main properties of the direct and inverse DFTs, and provide proofs for some of them.
- Note that since the DFT corresponds to samples of the Fourier transform, its properties closely resemble the ones of the Fourier transform presented in Chapter 2..
- However, from N samples of the Fourier transform, one can only recover a signal corresponding to the periodic repetition of the signal $x(n)$ with period N, as given by equation (5).
  - This makes the properties of the DFT slightly different from the ones of the Fourier transform.

Properties of the DFT

- One should bear in mind that, although the DFT can be interpreted as the Fourier transform of a periodic signal, it is just a mapping from a length-N signal into N frequency coefficients and vice versa.
- However, we often resort to this periodic signal interpretation, since some of the DFT properties follow naturally from it.
  - Note that in this case the periodic signal is the one in equation (5), which is the same one as that obtained by allowing the index n in equation (16) to vary within $(-\infty, \infty)$.

Properties of the DFT

1. **Linearity**

   The DFT of a linear combination of two sequences is the linear combination of the DFT of the individual sequences, that is, if $x(n) = k_1 x_1(n) + k_2 x_2(n)$ then

$$X(k) = k_1 X_1(k) + k_2 X_2(k) \tag{41}$$

   – Note that the two DFTs must have the same length, and thus the two sequences, if necessary, should be zero-padded accordingly in order to reach the same length N.

Properties of the DFT

2. Time-reversal

The DFT of $x(-n)$ is such that

$$x(-n) \longleftrightarrow X(-k) \tag{42}$$

- It must be noted that, if both indexes n and k are constrained to be between 0 and $N-1$, then $-n$ and $-k$ are outside this interval.
- Therefore, consistency with equations (15) and (16) requires that $x(-n) = x(N-n)$ and $X(-k) = X(N-k)$.
- These relations can also be deduced from the fact that we can interpret both $x(n)$ and $X(k)$ as being periodic with period N.

Properties of the DFT

3. Time-shift theorem

The DFT of a sequence shifted in time is such that

$$x(n+l) \longleftrightarrow W_N^{-lk} X(k) \tag{43}$$

– In the definition of the IDFT given in equation (16), if the index n is allowed to vary outside the set $0, 1, \ldots, (N-1)$, then $x(n)$ can be interpreted as being periodic with period N. This interpretation implies that the signal $x(n+l)$ obtained from the inverse DFT of $W_N^{-lk} X(k)$ corresponds to a circular shift of $x(n)$, that is, provided that $1 \leq l \leq N-1$, if

$$y(n) \longleftrightarrow W_N^{-lk} X(k) \tag{44}$$

then

$$y(n) = \begin{cases} x(n+l), & \text{for } 0 \leq n \leq N-l-1 \\ x(n+l-N), & \text{for } N-l \leq n \leq N-1 \end{cases} \tag{45}$$

Properties of the DFT

- This result indicates that $y(n)$ is a sequence whose last l samples are equal to the first l samples of $x(n)$.
  - An example of circular shift is illustrated in Figure 2.
  - A formal proof of this property is provided below for the case $1 \leq l \leq N-1$.

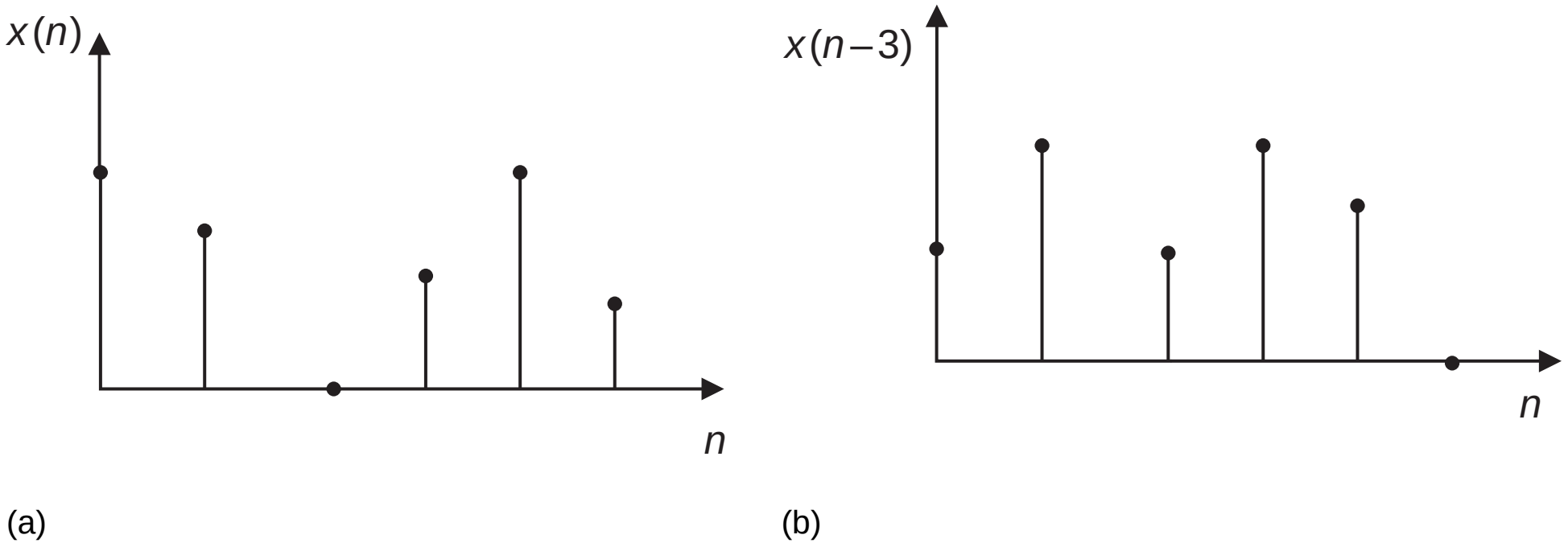


Figure 2: Circular shift of 3 samples: (a) original signal $x(n)$; (b) resulting signal $x(n-3)$.

Properties of the DFT

Proof

$$
\begin{aligned}
X'(k) &= \sum_{n=0}^{N-1} x(n+l) W_N^{nk} \\
&= W_N^{-lk} \sum_{n=0}^{N-1} x(n+l) W_N^{(n+l)k} \\
&= W_N^{-lk} \sum_{m=l}^{N+l-1} x(m) W_N^{mk} \\
&= W_N^{-lk} \left(\sum_{m=l}^{N-1} x(m) W_N^{mk} + \sum_{m=N}^{N+l-1} x(m) W_N^{mk} \right) \qquad (46)
\end{aligned}
$$

Properties of the DFT

Since W_N^k has period N and $x(n+l)$ is obtained from a circular shift of $x(n)$, then the summation from N to $(N+l-1)$ is equivalent to a summation from 0 to $(l-1)$. Therefore

$$X'(k) = W_N^{-lk}\left(\sum_{m=l}^{N-1} x(m)W_N^{mk} + \sum_{m=0}^{l-1} x(m)W_N^{mk}\right)$$

$$= W_N^{-lk}\sum_{m=0}^{N-1} x(m)W_N^{mk} = W_N^{-lk}X(k) \qquad (47)$$

- It is important to note that, since both $x(n)$ and W_N^{kn} are periodic with period N, then the property is still valid for both $l < 0$ and $l \geq N$. □
- A short-hand notation for equation (45) is

$$y(n) = x((n+l) \bmod N) \qquad (48)$$

where the $(n \bmod N)$ represents the remainder of the division of n by N, and is always in between 0 and $(N-1)$.

Properties of the DFT

Example 3.3

Given the DFT of a length-6 sequence $x(n)$ as below:

$$X(k) = \begin{cases} 4, & k = 0 \\ 2, & 1 \leq k \leq 5 \end{cases} \tag{49}$$

(a) Compute $x(n)$.

(b) Determine the length-6 sequence $y(n)$ whose DFT is $Y(k) = W_6^{-2k} X(k)$.

Discrete Fourier transform

Solution

(a)

$$x(n) = \frac{1}{6}\sum_{k=0}^{5} X(k)W_6^{kn} = \frac{1}{6}\left(4W_6^0 + \sum_{k=1}^{5} 2W_6^{kn}\right) \tag{50}$$

If $n = 0$, we have that

$$x(0) = \frac{1}{6}\left(4W_6^0 + \sum_{k=1}^{5} 2W_6^0\right) = \frac{7}{3} \tag{51}$$

For $1 \leq n \leq 5$,

$$x(n) = \frac{1}{6}\left(4 + \frac{2W_6^n - 2W_6^{6n}}{1 - W_6^n}\right) = \frac{1}{6}\left(4 + 2\frac{W_6^n - 1}{1 - W_6^n}\right) = \frac{1}{3} \tag{52}$$

Properties of the DFT

In short-hand notation,

$$x(n) = \frac{1}{3} + 2\delta(n), \quad \text{for } 0 \leq n \leq 5 \tag{53}$$

We can express the above equations using the matrix notation in equation (37) as

$$\mathbf{x} = \begin{bmatrix} \frac{7}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}^T \tag{54}$$

Properties of the DFT

(b) Using the time-shift theorem, we have that if $Y(k) = W_6^{-2k} X(k)$, then

$$y(n) = x((n+2) \bmod 6) = \frac{1}{3} + 2\delta((n+2) \bmod 6) \tag{55}$$

Since $((n+2) \bmod 6) = 0$ for $n = 4$, then we can express $y(n)$ as

$$y(n) = \frac{1}{3} + 2\delta(n-4), \quad 0 \leq n \leq 5 \tag{56}$$

which, in matrix notation is

$$\mathbf{y} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{7}{3} & \frac{1}{3} \end{bmatrix}^T \tag{57}$$

△

Properties of the DFT

4. Circular frequency-shift theorem (modulation theorem)

$$W_N^{ln} x(n) \longleftrightarrow X(k+l) \tag{58}$$

– The proof is analogous to the one of the time-shift theorem, and is left as an exercise to the reader.

– Noting that

$$W_N^{ln} = \cos\left(\frac{2\pi}{N} ln\right) - j \sin\left(\frac{2\pi}{N} ln\right) \tag{59}$$

equation (58) also implies the following properties:

$$x(n) \sin\left(\frac{2\pi}{N} ln\right) \quad \longleftrightarrow \quad \frac{1}{2j}(X(k-l) - X(k+l)) \tag{60}$$

$$x(n) \cos\left(\frac{2\pi}{N} ln\right) \quad \longleftrightarrow \quad \frac{1}{2}(X(k-l) + X(k+l)) \tag{61}$$

Properties of the DFT

5. Circular convolution in time

If $x(n)$ and $h(n)$ are periodic with period N, then

$$\sum_{l=0}^{N-1} x(l)h(n-l) = \sum_{l=0}^{N-1} x(n-l)h(l) \longleftrightarrow X(k)H(k) \tag{62}$$

where $X(k)$ and $H(k)$ are the DFTs of the length-N signals corresponding to one period of $x(n)$ and $h(n)$, respectively.

Properties of the DFT

Proof

If $Y(k) = X(k)H(k)$, then

$$
\begin{aligned}
y(n) &= \frac{1}{N}\sum_{k=0}^{N-1} H(k)X(k)W_N^{-kn} \\
&= \frac{1}{N}\sum_{k=0}^{N-1} H(k)\left(\sum_{l=0}^{N-1} x(l)W_N^{kl}\right)W_N^{-kn} \\
&= \frac{1}{N}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} H(k)x(l)W_N^{(l-n)k} \\
&= \sum_{l=0}^{N-1} x(l)\frac{1}{N}\sum_{k=0}^{N-1} H(k)W_N^{-(n-l)k} \\
&= \sum_{l=0}^{N-1} x(l)h(n-l) \qquad (63)
\end{aligned}
$$

□

Properties of the DFT

- This result is the basis of one of the most important applications of the DFT, which is the ability to compute a discrete convolution in time through the application of the inverse DFT to the product of the DFTs of the two sequences.
- However, one should bear in mind that when computing the DFT all the sequence shifts involved are circular, as depicted in Figure 2.
  - Therefore, we say that the product of two DFTs actually corresponds to a circular convolution in the time domain.
  - In fact, the circular convolution is equivalent to one period of the convolution between one original sequence and the periodic version of the other.
  - It is important to note that, as seen in Chapter 1, linear convolutions are usually the ones of interest in practice.
  - In the next section, we will discuss how linear convolutions can be implemented through circular convolutions, and therefore through the computation of DFTs.

Properties of the DFT

- Since $x(n)$ and $h(n)$ in equation (62) have period N, then their circular convolution $y(n)$ also has period N, and needs to be specified only for $0 \leq n \leq N-1$.
- Therefore, the circular convolution in equation (62) can be expressed as a function of only the samples of $x(n)$ and $h(n)$ between 0 and $N-1$ as

$$\begin{aligned} y(n) &= \sum_{l=0}^{N-1} x(l)h(n-l) \\ &= \sum_{l=0}^{n} x(l)h(n-l) + \sum_{l=n+1}^{N-1} x(l)h(n-l+N), \quad \text{for } 0 \leq n \leq N-1 \end{aligned} \tag{64}$$

which can be rewritten in a compact form as

$$y(n) = \sum_{l=0}^{N-1} x(l)h((n-l) \bmod N) = x(n) \circledast h(n) \tag{65}$$

where $(l \bmod N)$ represents the remainder of the integer division of l by N.

Properties of the DFT

- Equation (65) can be expressed as $y(n) = \mathbf{h}^T\mathbf{x}$ with

$$\mathbf{h} = \begin{bmatrix} h(n \bmod N) \\ h((n-1) \bmod N) \\ h((n-2) \bmod N) \\ \vdots \\ h((n-N+1) \bmod N) \end{bmatrix} \tag{66}$$

and $\mathbf{x}$ as before.

Properties of the DFT

- Therefore, the circular convolution can be put in matrix form as

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \\ y(N-1) \end{bmatrix} = \begin{bmatrix} h(0) & h(N-1) & h(N-2) & \cdots & h(1) \\ h(1) & h(0) & h(N-1) & \cdots & h(2) \\ h(2) & h(1) & h(0) & \cdots & h(3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h(N-1) & h(N-2) & h(N-3) & \cdots & h(0) \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{bmatrix} \tag{67}$$

- Note that each row of the matrix in the above equation is a circular right-shift of the previous row.
- In the remainder of this chapter, unless otherwise stated, it will be assumed that all the sequences are periodic and all the convolutions are circular.

Properties of the DFT

6. Correlation

The DFT of the correlation in time between two sequences is such that

$$\sum_{n=0}^{N-1} h(n)x(l+n) \longleftrightarrow H(-k)X(k) \tag{68}$$

This result is a direct consequence of the convolution and the time-reversal properties. Its proof is left as an exercise to the reader.

Properties of the DFT

7. Complex conjugation

$$x^*(n) \longleftrightarrow X^*(-k) \tag{69}$$

Proof

$$\sum_{n=0}^{N-1} x^*(n) W_N^{kn} = \left(\sum_{n=0}^{N-1} x(n) W_N^{-kn} \right)^* = X^*(-k) \tag{70}$$

Properties of the DFT

8. Real and imaginary sequences

If $x(n)$ is a real sequence, then $X(k) = X^*(-k)$, that is

$$\left.\begin{aligned} \text{Re}\{X(k)\} &= \text{Re}\{X(-k)\} \\ \text{Im}\{X(k)\} &= -\text{Im}\{X(-k)\} \end{aligned}\right\} \tag{71}$$

- The proof can be easily obtained from the definition of the DFT, by using the expression for W_N^{kn} in equation (59).
- When $x(n)$ is imaginary, then $X(k) = -X^*(-k)$, that is

$$\left.\begin{aligned} \text{Re}\{X(k)\} &= -\text{Re}\{X(-k)\} \\ \text{Im}\{X(k)\} &= \text{Im}\{X(-k)\} \end{aligned}\right\} \tag{72}$$

Properties of the DFT

Example 3.4

Show how to compute the DFTs of two real sequences through the computation of only one DFT.

Properties of the DFT

Solution

With the two real sequences $x_1(n)$ and $x_2(n)$ we form the sequence $y(n) = x_1(n) + jx_2(n)$. From the linearity of the DFT, we have that the DFT of $y(n)$ is

$$Y(k) = X_1(k) + jX_2(k) \tag{73}$$

From this equation, we have that

$$\left.\begin{aligned} \text{Re}\{Y(k)\} &= \text{Re}\{X_1(k)\} - \text{Im}\{X_2(k)\} \\ \text{Im}\{Y(k)\} &= \text{Re}\{X_2(k)\} + \text{Im}\{X_1(k)\} \end{aligned}\right\} \tag{74}$$

Using the properties in equation (71) in the above equation, we get

$$\left.\begin{aligned} \text{Re}\{Y(-k)\} &= \text{Re}\{X_1(k)\} + \text{Im}\{X_2(k)\} \\ \text{Im}\{Y(-k)\} &= \text{Re}\{X_2(k)\} - \text{Im}\{X_1(k)\} \end{aligned}\right\} \tag{75}$$

Properties of the DFT

Combining equations (74) and (75) , the DFTs of $x_1(n)$ and $x_2(n)$ can be computed as

$$\left.\begin{aligned}
\mathrm{Re}\{X_1(k)\} &= \frac{1}{2}\left(\mathrm{Re}\{Y(k)\} + \mathrm{Re}\{Y(-k)\}\right) \\
\mathrm{Im}\{X_1(k)\} &= \frac{1}{2}\left(\mathrm{Im}\{Y(k)\} - \mathrm{Im}\{Y(-k)\}\right) \\
\mathrm{Re}\{X_2(k)\} &= \frac{1}{2}\left(\mathrm{Im}\{Y(k)\} + \mathrm{Im}\{Y(-k)\}\right) \\
\mathrm{Im}\{X_2(k)\} &= \frac{1}{2}\left(\mathrm{Re}\{Y(-k)\} - \mathrm{Re}\{Y(k)\}\right)
\end{aligned}\right\} \qquad (76)$$

△

Properties of the DFT

9. Symmetric and antisymmetric sequences

- Symmetric and antisymmetric sequences are of special interest because their DFTs have some interesting properties.
- We have studied before the properties of the Fourier transform relating to symmetric and antisymmetric sequences.
- In this chapter, the meanings of symmetry and antisymmetry are slightly different.
  -
  - This happens because, unlike the Fourier and z transforms, that are applied to infinite-duration signals, the DFT is applied to finite-duration signals.
  - In fact, the DFT can be interpreted as the Fourier transform of a periodic signal formed from the infinite repetition of the finite-duration signal.

Properties of the DFT

- Therefore, before describing the properties of the DFT relating to symmetric and antisymmetric sequences, we give precise definitions for them in the context of the DFT.
  - A sequence is called symmetric (or even) if $x(n) = x(-n)$. Since, for indexes outside the set $0, 1, \ldots, (N-1)$, $x(n)$ can be interpreted as periodic with period N (see equation (16)), then $x(-n) = x(N-n)$. Therefore, symmetry is equivalent to $x(n) = x(N-n)$.
  - A sequence is antisymmetric (or odd) if $x(n) = -x(-n) = -x(N-n)$.
  - A complex sequence is said to be conjugate symmetric if $x(n) = x^*(-n) = x^*(N-n)$.
  - A complex sequence is called conjugate antisymmetric if $x(n) = -x^*(-n) = -x^*(N-n)$.

Properties of the DFT

- Using such concepts, the following properties hold:
  - If $x(n)$ is real and symmetric, $X(k)$ is also real and symmetric.

Proof

From equation (15), $X(k)$ is given by

$$X(k) = \sum_{n=0}^{N-1} x(n) \cos\left(\frac{2\pi}{N} kn\right) - j \sum_{n=0}^{N-1} x(n) \sin\left(\frac{2\pi}{N} kn\right) \qquad (77)$$

Since $x(n) = x(N-n)$, the imaginary part of the above summation is null, because, for N even, we get that

$$
\begin{aligned}
\sum_{n=0}^{N-1} x(n) \sin\left(\frac{2\pi}{N} kn\right) &= \sum_{n=0}^{\frac{N}{2}-1} x(n) \sin\left(\frac{2\pi}{N} kn\right) + \sum_{n=\frac{N}{2}}^{N-1} x(n) \sin\left(\frac{2\pi}{N} kn\right) \\
&= \sum_{n=1}^{\frac{N}{2}-1} x(n) \sin\left(\frac{2\pi}{N} kn\right) + \sum_{n=\frac{N}{2}+1}^{N-1} x(n) \sin\left(\frac{2\pi}{N} kn\right) \\
&= \sum_{n=1}^{\frac{N}{2}-1} x(n) \sin\left(\frac{2\pi}{N} kn\right) \\
&\quad + \sum_{m=1}^{\frac{N}{2}-1} x(N-m) \sin\left[\frac{2\pi}{N} k(N-m)\right] \\
&= \sum_{n=1}^{\frac{N}{2}-1} x(n) \sin\left(\frac{2\pi}{N} kn\right) - \sum_{m=1}^{\frac{N}{2}-1} x(m) \sin\left(\frac{2\pi}{N} km\right) \\
&= 0
\end{aligned}
\tag{78}
$$

Properties of the DFT

Therefore, we have that

$$X(k) = \sum_{n=0}^{N-1} x(n) \cos\left(\frac{2\pi}{N} kn\right) \tag{79}$$

which is real and symmetric (even). The proof for N odd is analogous, and is left as an exercise to the reader.

□

- If $x(n)$ is imaginary and even, then $X(k)$ is imaginary and even.
- If $x(n)$ is real and odd, then $X(k)$ is imaginary and odd.
- If $x(n)$ is imaginary and odd, then $X(k)$ is real and odd.
- If $x(n)$ is conjugate symmetric, then $X(k)$ is real.

Properties of the DFT

Proof

A conjugate symmetric sequence $x(n)$ can be expressed as

$$x(n) = x_e(n) + jx_o(n) \tag{80}$$

where $x_e(n)$ is real and even and $x_o(n)$ is real and odd. Therefore,

$$X(k) = X_e(k) + jX_o(k) \tag{81}$$

From the above properties, $X_e(k)$ is real and even, and $X_o(k)$ is imaginary and odd. Thus, $X(k) = X_e(k) + jX_o(k)$ is real.

□

- If $x(n)$ is conjugate antisymmetric, then $X(k)$ is imaginary.

The proofs of all other of these properties are left as exercises to the interested reader.

Properties of the DFT

Example 3.5

Given the sequence $x(n)$ represented by the matrix below:

$$\mathbf{x} = \begin{bmatrix} 1 & 2 & 3 & 4 & 0 & 0 \end{bmatrix}^T \tag{82}$$

find the sequence $y(n)$ whose length-6 DFT is given by $Y(k) = \text{Re}\{X(k)\}$.

Properties of the DFT

Solution

$$Y(k) = \frac{X(k) + X^*(k)}{2} \tag{83}$$

From the linearity of the DFT,

$$y(n) = \frac{\text{IDFT}\{X(k)\} + \text{IDFT}\{X^*(k)\}}{2} \tag{84}$$

Since $x(n)$ is real and is a sequence of length 6,

$$y(n) = \frac{x(n) + x(-n)}{2} = \frac{x(n) + x(6-n)}{2} \tag{85}$$

Properties of the DFT

Then,

$$\left.\begin{aligned} y(0) &= \frac{x(0)+x(6)}{2} = 1 \\ y(1) &= \frac{x(1)+x(5)}{2} = 1 \\ y(2) &= \frac{x(2)+x(4)}{2} = \frac{3}{2} \\ y(3) &= \frac{x(3)+x(3)}{2} = 4 \\ y(4) &= \frac{x(4)+x(2)}{2} = 1 \\ y(5) &= \frac{x(5)+x(1)}{2} = 1 \end{aligned}\right\} \tag{86}$$

△

Properties of the DFT

10. Parseval's theorem

$$\sum_{n=0}^{N-1} x_1(n)x_2^*(n) = \frac{1}{N}\sum_{k=0}^{N-1} X_1(k)X_2^*(k) \tag{87}$$

Proof

$$\begin{aligned}
\sum_{n=0}^{N-1} x_1(n)x_2^*(n) &= \sum_{k=0}^{N-1}\left(\frac{1}{N}\sum_{k=0}^{N-1} X_1(k)W_N^{-kn}\right)x_2^*(n) \\
&= \frac{1}{N}\sum_{k=0}^{N-1} X_1(k)\sum_{k=0}^{N-1} x_2^*(n)W_N^{-kn} \\
&= \frac{1}{N}\sum_{k=0}^{N-1} X_1(k)X_2^*(k) \qquad (88)
\end{aligned}$$

Properties of the DFT

If $x_1(n) = x_2(n)$, then we have that

$$\sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X(k)|^2 \tag{89}$$

- The above equation can be interpreted as an energy conservation property, since the energy in the discrete-time domain is equal to the energy in the discrete-frequency domain, up to a scaling factor N.

Properties of the DFT

11. Relationship between the DFT and the z transform

- We have defined before the DFT as samples of the Fourier transform, and showed, in equation (13), how to obtain the Fourier transform directly from the DFT.
- Since the Fourier transform corresponds to the z transform for $z = e^{j\omega}$, then clearly the DFT can be obtained by sampling the z transform at $\omega = \frac{2\pi}{N}k$.
- Mathematically, since the z transform $X_z(z)$ of a length-N sequence $x(n)$ is

$$X_z(z) = \sum_{n=0}^{N-1} x(n) z^{-n} \tag{90}$$

if we make $z = e^{j\frac{2\pi}{N}k}$, then we have that

$$X_z\left(e^{j\frac{2\pi}{N}kn}\right) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn} \tag{91}$$

- This equation corresponds to samples of the z transform equally spaced on the unit circle, and is identical to the definition of the DFT in equation (15).

- In order to obtain the z transform from the DFT coefficients $X(k)$, we substitute equation (16) into equation (90), obtaining

$$\begin{aligned} X_z(z) &= \sum_{n=0}^{N-1} x(n) z^{-n} \\ &= \sum_{n=0}^{N-1} \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn} z^{-n} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} X(k) \sum_{n=0}^{N-1} \left(W_N^{-k} z^{-1} \right)^n \\ &= \frac{1}{N} \sum_{k=0}^{N-1} X(k) \frac{1 - W_N^{-kN} z^{-N}}{1 - W_N^{-k} z^{-1}} \\ &= \frac{1 - z^{-N}}{N} \sum_{k=0}^{N-1} \frac{X(k)}{1 - W_N^{-k} z^{-1}} \end{aligned} \tag{92}$$

which, similarly to equation (13) for the Fourier transform, relates the DFT to the z transform.

Digital filtering using the DFT

Linear and circular convolutions

- A linear time-invariant system implements the linear convolution of the input signal with the impulse response of the system.
- Since the Fourier transform of the convolution of two sequences is the product of their Fourier transforms, it is natural to consider the computation of time convolutions in the frequency domain.
- The DFT is the discrete version of the Fourier transform, and therefore should be the transform used for such computations.
- However, as is described in equation (5), the sampling process in the frequency domain forces the signal to be periodic in time.

Digital filtering using the DFT

- We saw previously that this implies that the IDFT of the product of the DFTs of two length-N signals corresponds to the convolution of one sequence with the periodic version of the other.
  - This periodic version is obtained by repeating the length-N signal with period N.
- As seen before, this convolution between a signal and the periodic version of the other is called the circular convolution between the two length-N signals.
  - This means that, using the DFT, one can in principle compute only circular convolutions, and not the linear convolution necessary to implement a linear system.
- In this section, we describe techniques to circumvent this problem and allow us to implement discrete-time linear systems in the frequency domain.

Digital filtering using the DFT

These techniques are essentially based on a very simple trick:

- Supposing that the DFTs are of size N, we have that the circular convolution between two sequences $x(n)$ and $h(n)$ is given by equation (64), which is repeated here for the reader's convenience

$$y(n) = \sum_{l=0}^{N-1} x(l)h(n-l)$$

$$= \sum_{l=0}^{n} x(l)h(n-l) + \sum_{l=n+1}^{N-1} x(l)h(n-l+N), \quad \text{for } 0 \leq n \leq N-1 \tag{93}$$

- If we want the circular convolution to be equal to the linear convolution between $x(n)$ and $h(n)$, the second summation in the above equation must be null, that is

$$c(n) = \sum_{l=n+1}^{N-1} x(l)h(n-l+N) = 0, \quad \text{for } 0 \leq n \leq N-1 \tag{94}$$

Digital filtering using the DFT

Assuming that $x(n)$ has duration L and $h(n)$ has duration K, that is,

$$x(n) = 0, \text{ for } n \geq L\,; \quad h(n) = 0, \text{ for } n \geq K \tag{95}$$

we have that the summation $c(n)$ in equation (94) is different from zero only if both $x(l)$ and $h(n-l+N)$ are nonzero, and this happens if

$$l \leq L-1 \quad \text{and} \quad n-l+N \leq K-1 \tag{96}$$

which implies that

$$n+N-K+1 \leq l \leq L-1 \tag{97}$$

If we then want the summation $c(n)$ to be null for $0 \leq n \leq N-1$, it should be impossible to satisfy equation (97) for $0 \leq n \leq N-1$. This happens when

$$n+N-K+1 > L-1 \quad \text{for} \quad 0 \leq n \leq N-1 \tag{98}$$

Digital filtering using the DFT

Since the most strict case of equation (98) is for $n = 0$, we have that the condition for $c(n) = 0$, $0 \leq n \leq N - 1$, and therefore for the circular convolution to be equivalent to the linear convolution, is

$$N \geq L + K - 1 \tag{99}$$

- Thus, in order to perform a linear convolution using the inverse DFT of the product of the DFT of two sequences, we must choose a DFT size N satisfying equation (99).
- This is equivalent to padding $x(n)$ using at least $K - 1$ zeros and padding $h(n)$ using at least $L - 1$ zeros.
- This zero-padding process is illustrated in Figure 3 for $L = 4$ and $K = 3$, where, after zero-padding, the sequences $x(n)$ and $h(n)$ are denoted as $x_1(n)$ and $h_1(n)$, respectively.

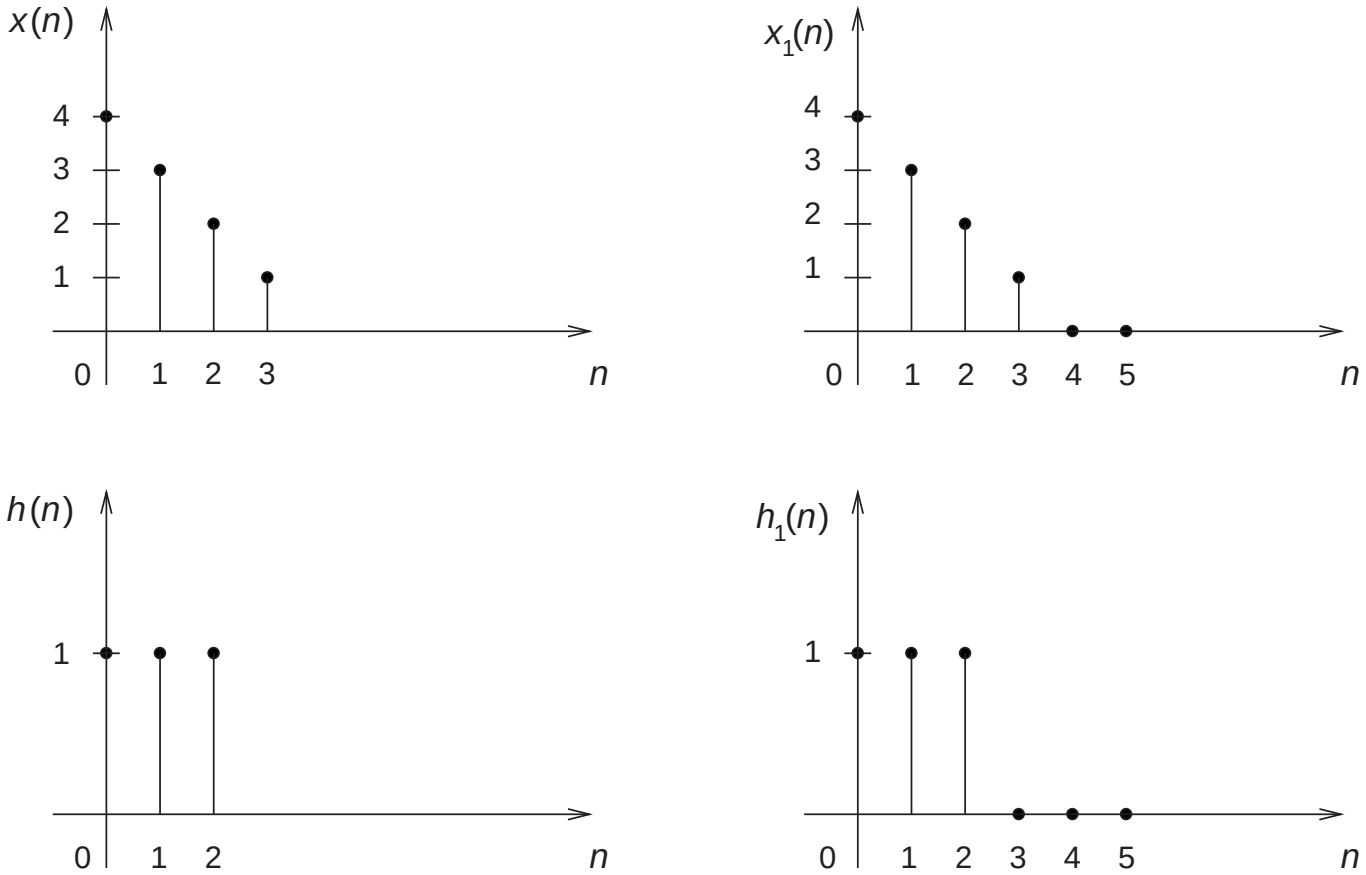


Figure 3: Zero-padding of two sequences in order to perform linear convolution using the DFT: $\mathrm{x_1(n)}$ corresponds to $\mathrm{x(n)}$, and $\mathrm{h_1(n)}$ corresponds to $\mathrm{h(n)}$ after appropriate padding.

The following example will help to clarify the above discussion.

Digital filtering using the DFT

Example 3.6

Referring to Figure 3, compute the linear convolution of the two sequences $x(n)$ and $h(n)$ and compare it with the circular convolution of $x(n)$ and $h(n)$ as well as the circular convolution of $x_1(n)$ and $h_1(n)$.

Digital filtering using the DFT

Solution

First, we compute the linear convolution of $x(n)$ and $h(n)$,

$$y_l(n) = x(n) * h(n) = \sum_{l=0}^{3} x(l)h(n-l) \tag{100}$$

such that

$$\left.\begin{aligned} y_l(0) &= x(0)h(0) = 4 \\ y_l(1) &= x(0)h(1) + x(1)h(0) = 7 \\ y_l(2) &= x(0)h(2) + x(1)h(1) + x(2)h(0) = 9 \\ y_l(3) &= x(1)h(2) + x(2)h(1) + x(3)h(0) = 6 \\ y_l(4) &= x(2)h(2) + x(3)h(1) = 3 \\ y_l(5) &= x(3)h(2) = 1 \end{aligned}\right\} \tag{101}$$

Digital filtering using the DFT

The circular convolution of length $N = 4$ between $x(n)$ and $h(n)$, using equation (65), is equal to

$$y_{c_4}(n) = x(n) \circledast h(n) = \sum_{l=0}^{3} x(l)h((n-l) \bmod 4) \tag{102}$$

such that

$$\left.\begin{aligned} y_{c_4}(0) &= x(0)h(0) + x(1)h(3) + x(2)h(2) + x(3)h(1) = 7 \\ y_{c_4}(1) &= x(0)h(1) + x(1)h(0) + x(2)h(3) + x(3)h(2) = 8 \\ y_{c_4}(2) &= x(0)h(2) + x(1)h(1) + x(2)h(0) + x(3)h(3) = 9 \\ y_{c_4}(3) &= x(0)h(3) + x(1)h(2) + x(2)h(1) + x(3)h(0) = 6 \end{aligned}\right\} \tag{103}$$

Digital filtering using the DFT

We can also use equation (65) to compute the circular convolution of length $N = 6$ of $x_1(n)$ and $h_1(n)$, obtaining

$$y_{c_6}(n) = x_1(n) \circledast h_1(n) = \sum_{l=0}^{5} x(l)h((n-l) \bmod 6) \qquad (104)$$

such that

$$
\left.
\begin{aligned}
y_{c_6}(0) &= x_1(0)h_1(0) + x_1(1)h_1(5) + x_1(2)h_1(4) \\
&\quad + x_1(3)h_1(3) + x_1(4)h_1(2) + x_1(5)h_1(1) \\
&= x_1(0)h_1(0) \\
&= 4 \\
y_{c_6}(1) &= x_1(0)h_1(1) + x_1(1)h_1(0) + x_1(2)h_1(5) \\
&\quad + x_1(3)h_1(4) + x_1(4)h_1(3) + x_1(5)h_1(2) \\
&= x_1(0)h_1(1) + x(1)h(0) \\
&= 7 \\
y_{c_6}(2) &= x_1(0)h_1(2) + x_1(1)h_1(1) + x_1(2)h_1(0) \\
&\quad + x_1(3)h_1(5) + x_1(4)h_1(4) + x_1(5)h_1(3) \\
&= x_1(0)h_1(2) + x(1)h(1) + x(2)h(0) \\
&= 9 \\
y_{c_6}(3) &= x_1(0)h_1(3) + x_1(1)h_1(2) + x_1(2)h_1(1) \\
&\quad + x_1(3)h_1(0) + x_1(4)h_1(5) + x_1(5)h_1(4) \\
&= x_1(1)h_1(2) + x(2)h(1) + x(3)h(0) \\
&= 6 \\
y_{c_6}(4) &= x_1(0)h_1(4) + x_1(1)h_1(3) + x_1(2)h_1(2) \\
&\quad + x_1(3)h_1(1) + x_1(4)h_1(0) + x_1(5)h_1(5) \\
&= x_1(2)h_1(2) + x(3)h(1) \\
&= 3 \\
y_{c_6}(5) &= x_1(0)h_1(5) + x_1(1)h_1(4) + x_1(2)h_1(3) \\
&\quad + x_1(3)h_1(2) + x_1(4)h_1(1) + x_1(5)h_1(0) \\
&= x_1(3)h_1(2) \\
&= 1
\end{aligned}
\right\} \quad (105)
$$

Digital filtering using the DFT

Comparing equations (101) and (105), it is easy to confirm that $y_{c_6}(n)$ corresponds exactly to the linear convolution between $x(n)$ and $h(n)$.

△

- We have now seen how it is possible to implement the linear convolution between two finite-length signals through the DFT.
- However, in practice, it is frequently necessary to implement the convolution of a finite-length sequence with an infinite-length sequence, or even to convolve a short-duration sequence with a long-duration one.
- In both cases, it is not feasible to compute the DFT of a very long or infinite sequence.

Digital filtering using the DFT

- The solution adopted, in these cases, is to divide the long sequence into blocks of duration N, and perform the convolution of each block with the short sequence.
  - In most practical cases, the long or infinite sequence corresponds to the system input and the short-length sequence to the system impulse response.
- However, the results of the convolution of each block must be properly combined so that the final result corresponds to the convolution of the long sequence with the short sequence.
- Two methods for performing this combination are the so-called overlap-and-add and overlap-and-save methods discussed in the sequel.

Overlap-and-add method

- We can describe a signal $x(n)$ decomposed in non-overlapping blocks $x_m(n - mN)$ of length N as

$$x(n) = \sum_{m=0}^{\infty} x_m(n - mN) \tag{106}$$

where

$$x_m(n) = \begin{cases} x(n + mN), & \text{for } 0 \leq n \leq N - 1 \\ 0, & \text{otherwise} \end{cases} \tag{107}$$

that is, each block $x_m(n)$ is nonzero between 0 and $N - 1$, and $x(n)$ is composed of the sum of the blocks $x_m(n)$ shifted to $n = mN$.

Digital filtering using the DFT

- Using equation (106), the convolution of $x(n)$ with another signal $h(n)$ can be written as

$$y(n) = x(n) * h(n) = \sum_{m=0}^{\infty} (x_m(n - mN) * h(n)) = \sum_{m=0}^{\infty} y_m(n - mN) \quad (108)$$

- Note that the above equation implies that $y_m(n)$ is the result of the convolution of $h(n)$ with the mth block $x_m(n)$.

- As seen in Subsection 72, if we want to compute the linear convolutions leading to $y_m(n)$ using DFTs, their lengths must be at least $(N + K - 1)$, where K is the duration of $h(n)$.

- Thus, if $x_m(n)$ and $h(n)$ are zero-padded to length $(N + K - 1)$, then the linear convolutions in equation (108) can be implemented using circular convolutions.

Digital filtering using the DFT

- If $x'_m(n)$ and $h'(n)$ are the zero-padded versions of $x_m(n)$ and $h(n)$, we have that the filtered blocks $y_m(n)$ in equation (108) become

$$y_m(n) = \sum_{l=0}^{N+K-2} x'_m(l)h'(n-l), \quad \text{for } 0 \leq n \leq N+K-2 \tag{109}$$

- Then, from equations (108) and (109), we see that in order to convolve the long $x(n)$ with the length-K $h(n)$ it suffices to:

  (i) Divide $x(n)$ into length-N blocks $x_m(n)$.

  (ii) Zero-pad $h(n)$ and each block $x_m(n)$ to length $N+K-1$.

  (iii) Perform the circular convolution of each block using the length-$(N+K-1)$ DFT.

  (iv) Add the results according to equation (108).

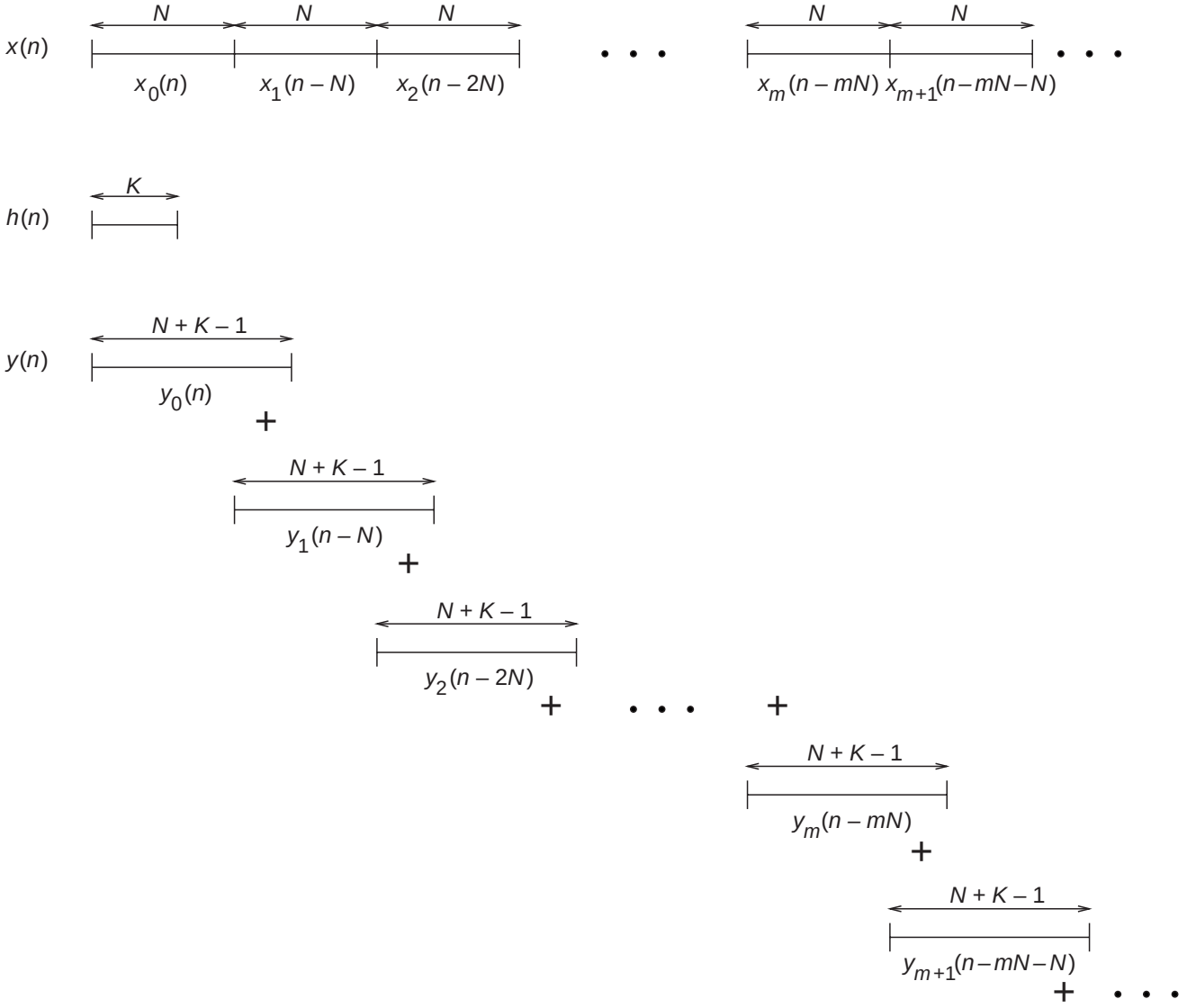


Figure 4: Illustration of the overlap-and-add method.

Digital filtering using the DFT

- Note that in the addition performed in step (iv), there is an overlap of the $K-1$ last samples of $y_m(n-mN)$ and the $K-1$ first samples of $y_{m+1}(n-(m+1)N)$.
- This is why the above procedure is called the overlap-and-add method, which is illustrated schematically in Figure 4.

Digital filtering using the DFT

Example 3.7

Compute graphically the linear convolution of $x(n)$ and $h(n)$ in Figure 5a by splitting $x(n)$ into blocks of two samples and using the overlap-and-add method.

Solution

According to equations (106) and (107), the division of the original signal in Figure 5a in two blocks having length $N = 2$ can be expressed as

$$x(n) = x_0(n) + x_1(n-2) \tag{110}$$

where

$$\left.\begin{aligned} x_0(n) &= \begin{cases} x(n), & \text{for } 0 \leq n \leq 1 \\ 0, & \text{otherwise} \end{cases} \\ x_1(n) &= \begin{cases} x(n+2), & \text{for } 0 \leq n \leq 1 \\ 0, & \text{otherwise} \end{cases} \end{aligned}\right\} \tag{111}$$

The two signals on the right-hand side of equation (110) are depicted in Figure 5b. The desired convolution is then

$$y(n) = (x_0(n) * h(n)) + (x_1(n-2) * h(n)) \tag{112}$$

where each of the signals on the right-hand side of the above equation are depicted in Figure 5d. Their sum $y(n)$ is shown in Figure 5e. △

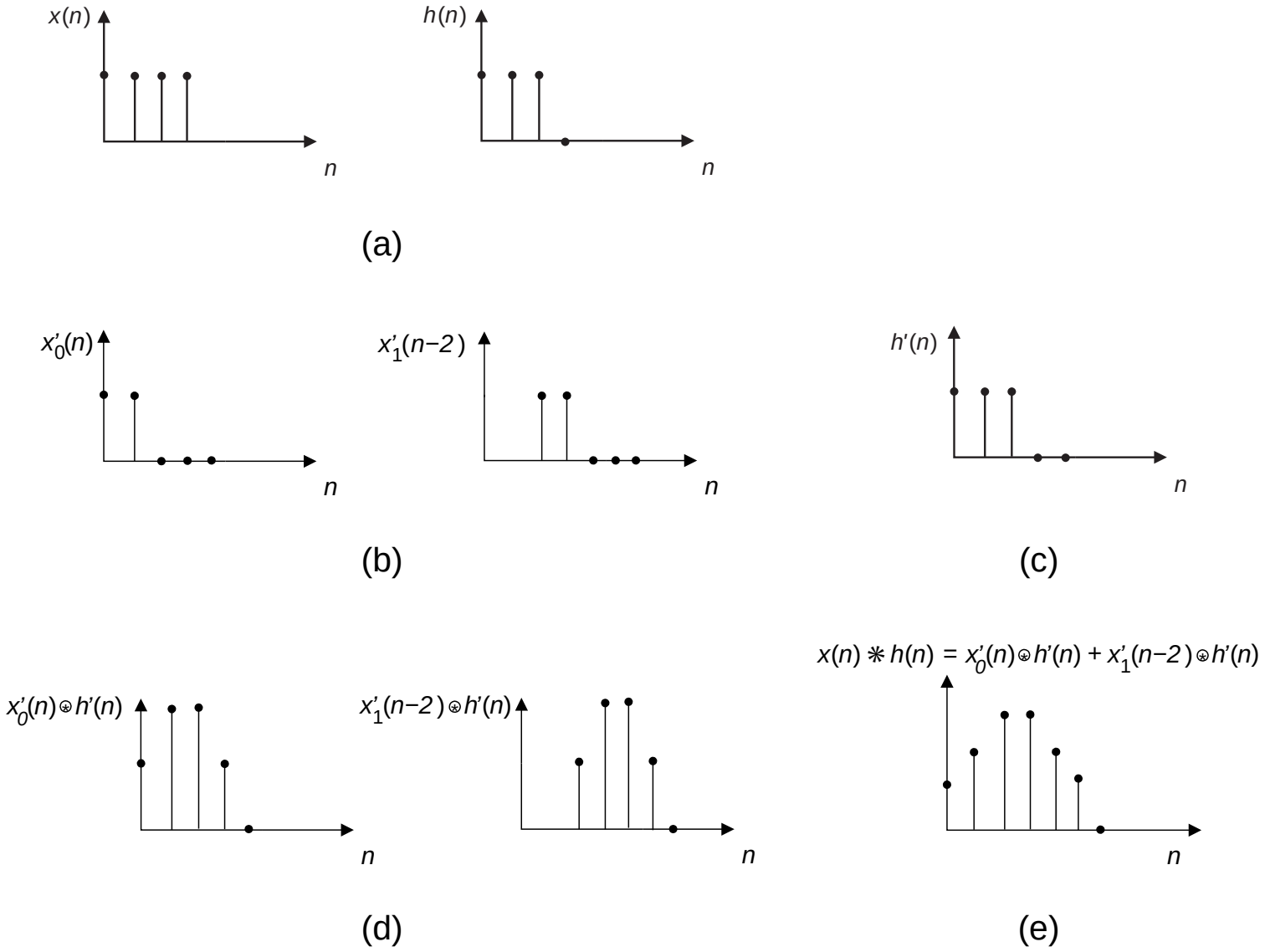


Figure 5: Example of the application of the overlap-and-add method: (a) original sequences; (b) input sequence divided into length-2 blocks. Each block has been zero-padded to length $N = 5$; (c) zero-padded $h(n)$; (d) circular convolution of each zero-padded block with zero-padded $h(n)$; (e) final result (normalized).

Overlap-and-save method

- In the overlap-and-add method, one divides the signal into blocks of length N and computes the convolutions using DFTs of size $(N + K - 1)$, where K is the duration of $h(n)$.
- In the overlap-and-save method, one uses DFTs of length N instead.
  - This poses a problem, because the length-N circular convolution of a length-N block $x_m(n)$ and a length-K $h(n)$ is not equal to their linear convolution.
- To circumvent this, one only uses the samples of the circular convolution that are equal to the ones of the linear convolution.
  - These valid samples can be determined by referring to the expression of the circular convolution

Digital filtering using the DFT

- From there, we see that the condition for computing a linear convolution using a circular convolution is given by equation (94), repeated here for convenience, with $x(n)$ replaced by $x_m(n)$:

$$c(n) = \sum_{l=n+1}^{N-1} x_m(l)h(n-l+N) = 0, \quad \text{for } 0 \leq n \leq N-1 \tag{113}$$

- This equation indicates that the length-N circular convolution, between a length-N block $x_m(n)$ and a length-K $h(n)$, is equal to their linear convolution whenever the above summation is null.

- Since, for $0 \leq n \leq N-1$, we have that $x_m(n) \neq 0$, then $c(n)$ above is null only for n such that all the $h(n)$ in the summation are zero, that is, $h(n-l+N) = 0$ for l in the interval $n+1 \leq l \leq N-1$.

Digital filtering using the DFT

- Since $h(n)$ has length K, then $h(r) = 0$, for $r \geq K$. Since we should have $h(n - l + N) = 0$, then n should be such that $n - l + N \geq K$, that is, $n \geq K - N + l$.
- The most strict case in this inequality is when $l = N - 1$. This implies that the condition in equation (113) is satisfied only when $n \geq K - 1$.
- The conclusion is that the only samples of the length-N circular convolution that are equal to the ones of the linear convolution are for $n \geq K - 1$.
- Therefore, when computing the convolution of the blocks $x_m(n)$ with $h(n)$, the first $K - 1$ samples of the result have to be discarded.
- In order to compensate for the discarded samples, there must be an overlap of an extra $K - 1$ samples between adjacent blocks.

Digital filtering using the DFT

- Thus, the signal $x(n)$ must be divided into blocks $x_m(n)$ of length N such that

$$x_m(n) = \begin{cases} x(n + m(N - K + 1) - K + 1), & \text{for } 0 \leq n \leq N - 1 \\ 0, & \text{otherwise} \end{cases} \tag{114}$$

- Note that the first $K - 1$ samples of $x_m(n)$ are equal to the last $K - 1$ samples of $x_{m-1}(n)$.
- The filtered output of the mth block consists only of the samples of the circular convolution $y_m(n)$ of $x_m(n)$ and $h(n)$ of index larger than or equal to $K - 1$.
- It is important that the original signal $x(n)$ be padded with $K - 1$ zeros at the beginning, since the first $K - 1$ samples of the output are discarded.

Digital filtering using the DFT

- Then, if $h'(n)$ is the version of $h(n)$ zero-padded to length N, the output $y_m(n)$ of each block can be expressed as

$$y_m(n) = \sum_{l=0}^{N-1} x_m(l) h'((n-l) \bmod N) \tag{115}$$

where only the samples of $y_m(n)$ from $n = K-1$ to $n = N-1$ need to be computed.

- Then the output $y(n) = x(n) * h(n)$ is built, for each m, as

$$y(n) = y_m(n - m(N-K+1)) \tag{116}$$

for $m(N-K+1) + K - 1 \leq n \leq m(N-K+1) + N - 1$.

Digital filtering using the DFT

- From equations (115) and (116), we see that, in order to convolve the long $x(n)$ with the length-K $h(n)$ using the overlap-and-save method, it suffices to:

  (i) Divide $x(n)$ into length-N blocks $x_m(n)$ with an overlap of $K-1$ samples as in equation (114). The first block should be zero-padded with $K-1$ zeros at its beginning. If the original signal has length L, then the total number of blocks B should obey

$$B \geq \frac{L+K-1}{N-K+1} \tag{117}$$

  (ii) Zero-pad $h(n)$ to length N.

  (iii) Perform the circular convolution of each block with $h(n)$ (equation (115)) using a length-N DFT.

  (iv) Build the output signal according to equation (116).

- Note that we can interpret step (iv) as the $K-1$ last samples of block $y_m(n)$ being saved in order to replace the discarded $K-1$ first samples of block $y_{m+1}(n)$, thus justifying the terminology overlap-and-save method, illustrated in Figure 6.

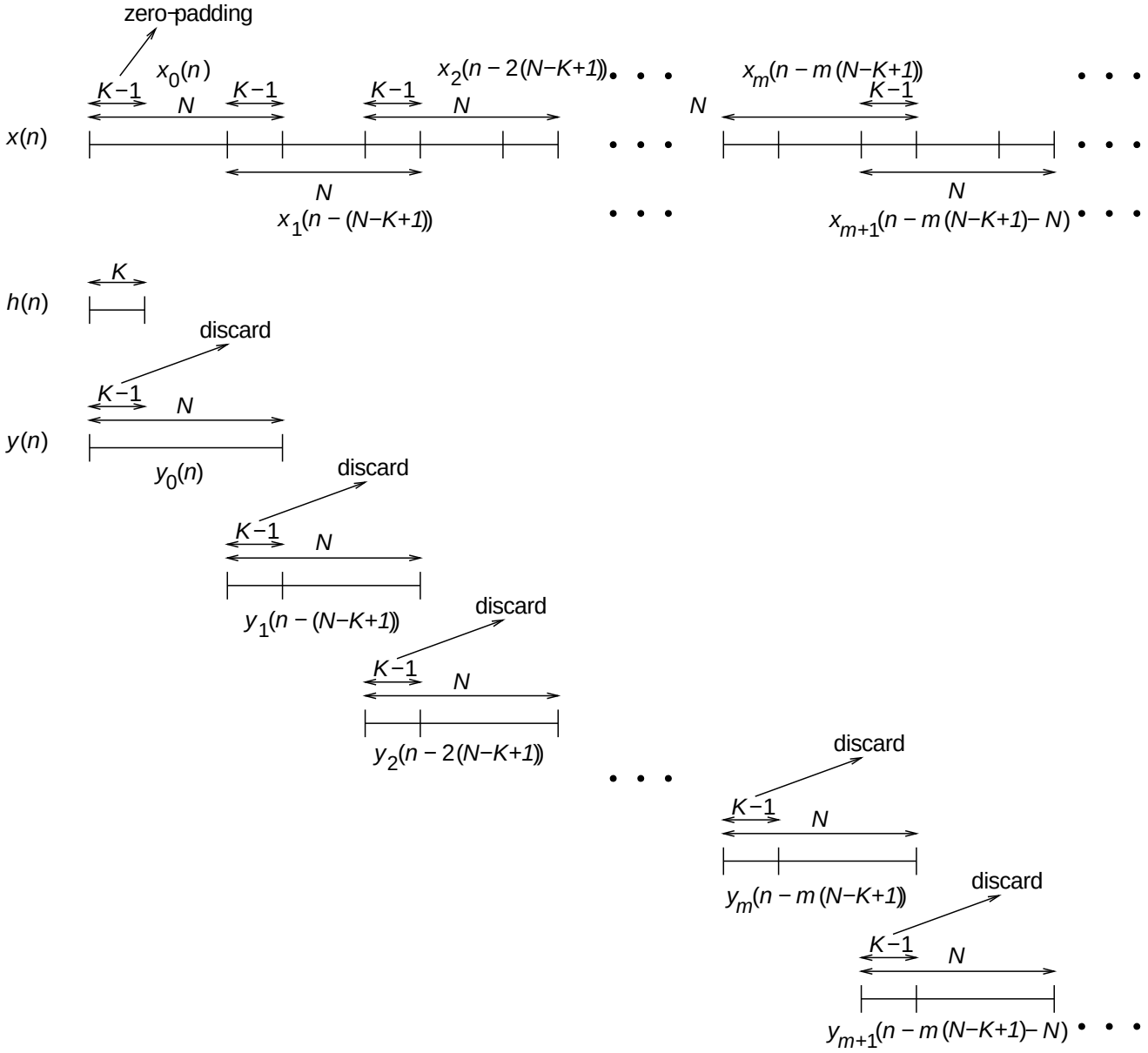


Figure 6: Illustration of the overlap-and-save method.

Digital filtering using the DFT

Example 3.8

Determine, graphically, the linear convolution of $x(n)$ and $h(n)$ in Figure 7a using the DFT, partitioning $x(n)$ into length-6 blocks and using the overlap-and-save method.

Digital filtering using the DFT

Solution

The length of the impulse response $h(n)$ is $K = 3$, $x(n)$ has length $L = 8$, and the DFT has length $N = 6$. From from equation (117), the number of overlapping blocks should be

$$B \geq \frac{8+3-1}{6-3+1} = 2.5 \tag{118}$$

and therefore $B = 3$. The beginning of the first block will be zero-padded with 2 zeros and the last block will be zero-padded with 4 zeros. Therefore, from equation (114), we

have that

$$\left.\begin{aligned} x_0(n) &= \begin{cases} x(n-2), & \text{for } 0 \le n \le 5 \\ 0, & \text{otherwise} \end{cases} \\ x_1(n) &= \begin{cases} x(n+2), & \text{for } 0 \le n \le 5 \\ 0, & \text{otherwise} \end{cases} \\ x_2(n) &= \begin{cases} x(n+6), & \text{for } 0 \le n \le 5 \\ 0, & \text{otherwise} \end{cases} \end{aligned}\right\} \tag{119}$$

These signals are depicted in Figures 7b, 7d, and 7f, respectively. Using equation (115) the signals in Figures 7c, 7e and 7g are computed as

$$y_m(n) = \sum_{l=0}^{5} x_m(l) h'((n-l) \bmod 6), \quad \text{for} \quad 2 \le n \le 5 \quad \text{and} \quad m = 0, 1, 2 \tag{120}$$

The final result in Figure 7h is computed using equation (116) yielding

$$\left.\begin{aligned} y(n) &= y_0(n), \quad \text{for} \quad 2 \leq n \leq 5 \\ y(n) &= y_1(n-4), \quad \text{for} \quad 6 \leq n \leq 9 \\ y(n) &= y_2(n-8), \quad \text{for} \quad 10 \leq n \leq 13 \end{aligned}\right\} \tag{121}$$

Note that in this example, we have that $K = 3$, $N = 6$, and each partial convolution generates $(N - K + 1) = 4$ new samples.

$\triangle$

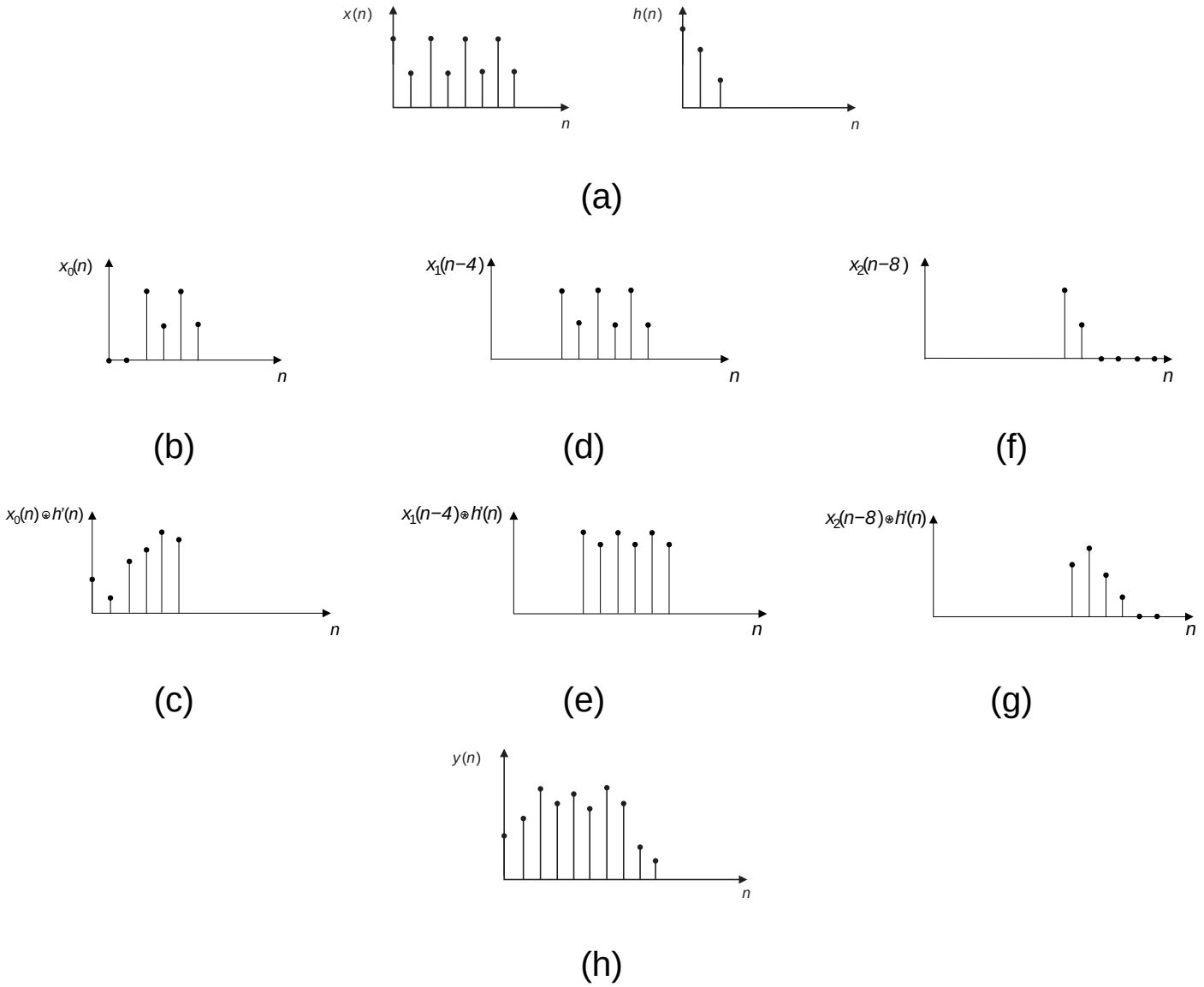


Figure 7: Linear convolution using the DFT and the overlap-and-save method: (a) original sequences; (b) first block; (c) first partial convolution; (d) second block; (e) second partial convolution; (f) third block; (g) third partial convolution; (h) final result.

Fast Fourier transform

- In the previous section, we saw that the DFT is an effective discrete representation in frequency that can be used to compute linear convolutions between two discrete sequences.
- However, by examining the DFT and IDFT definitions in equations (15) and (16), repeated here for convenience

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad \text{for } 0 \leq k \leq N-1 \tag{122}$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn}, \quad \text{for } 0 \leq n \leq N-1 \tag{123}$$

we see that in order to compute the DFT and IDFT of a length-N sequence, one needs about N^2 complex multiplications, that is, the complexity of the DFT grows with the square of the signal length.

Fast Fourier Transform

- This severely limits its practical use for lengthy signals.
- Fortunately, in 1965, Cooley and Tukey proposed an efficient algorithm to compute the DFT, which requires a number of complex multiplications of the order of $N \log_2 N$.
- This may represent a tremendous decrease in complexity.
  - For example, even for signal lengths as low as 1024 samples, the decrease in complexity is of the order of 100 times, that is, two orders of magnitude.
- It is needless to say that the advent of this algorithm opened up an endless list of practical applications for the DFT, ranging from signal analysis to fast linear filtering.
- Today, there is an enormous number of fast algorithms for the computation of the DFT, and they are collectively known as FFT (fast Fourier transform) algorithms.
- In this section we will study some of the most popular types of FFT algorithms.

Radix-2 algorithm with decimation in time

- Suppose we have a sequence $x(n)$ whose length N is a power of two, that is, $N = 2^l$.
- Now, let us express the DFT relation in equation (122) by splitting the summation into two parts, one with the even-indexed $x(n)$ and another with the odd-indexed $x(n)$, obtaining

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{nk} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x(2n) W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) W_N^{(2n+1)k} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x(2n) W_N^{2nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) W_N^{2nk} \end{aligned} \tag{124}$$

Radix-2 algorithm with decimation in time

- If we note that for N even, we have that

$$W_N^{2nk} = e^{-j\frac{2\pi}{N}2nk} = e^{-j\frac{2\pi}{\frac{N}{2}}nk} = W_{\frac{N}{2}}^{nk} \tag{125}$$

  then equation (124) becomes

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n) W_{\frac{N}{2}}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) W_{\frac{N}{2}}^{nk} \tag{126}$$

  and we can see that each summation may represent a distinct DFT of size $N/2$.

- Therefore, a DFT of size N can be computed through two DFTs of size $N/2$, in addition to the multiplications by W_N^k.

- Note that each new DFT has only $N/2$ coefficients, and for the computation of its coefficients we now need solely $(N/2)^2$ complex multiplications.

- In addition, since we have one distinct coefficient W_N^k for each k between 0 and $N-1$, then we need to perform N multiplications by W_N^k.

Radix-2 algorithm with decimation in time

- Therefore, the computation of the DFT according to equation (126) requires

$$2\left(\frac{N}{2}\right)^2 + N = \frac{N^2}{2} + N \tag{127}$$

complex multiplications.

  - Since $\left(N + \frac{N^2}{2}\right)$ is smaller than N^2, for $N > 2$, equation (126) provides a decrease in complexity when compared to the usual DFT computation.

Radix-2 algorithm with decimation in time

- We should also compare the number of complex additions of the two forms of computation.
  - The usual length-N DFT computation needs a total of $N(N-1) = N^2 - N$ additions.
  - In equation (126), we need to compute two DFTs of length $N/2$, and then perform the N additions of the two DFTs after multiplication by W_N^k.
  - Therefore, the total number of complex additions in equation (126) is

$$2\left[\left(\frac{N}{2}\right)^2 - \frac{N}{2}\right] + N = \frac{N^2}{2} \tag{128}$$

    which also corresponds to a reduction in complexity.

- From the above, exploiting the fact that N is a power of 2, it is easy to see that if the procedure shown in equation (126) is recursively applied to each of the resulting DFTs, we may get a very significant reduction in complexity until all the remaining DFTs are of length 2.

Radix-2 algorithm with decimation in time

- The overall procedure is formalized in an algorithm by first rewriting equation (126) as

$$X(k) = X_e(k) + W_N^k X_o(k) \tag{129}$$

where $X_e(k)$ and $X_o(k)$ are, respectively, the DFTs of length $\frac{N}{2}$ of the even- and odd-indexed samples of $x(n)$, that is

$$\left.\begin{aligned} X_e(k) &= \sum_{n=0}^{\frac{N}{2}-1} x(2n) W_{\frac{N}{2}}^{nk} = \sum_{n=0}^{\frac{N}{2}-1} x_e(n) W_{\frac{N}{2}}^{nk} \\ X_o(k) &= \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) W_{\frac{N}{2}}^{nk} = \sum_{n=0}^{\frac{N}{2}-1} x_o(n) W_{\frac{N}{2}}^{nk} \end{aligned}\right\} \tag{130}$$

Radix-2 algorithm with decimation in time

- The DFTs above can be computed by separating $x_e(n)$ and $x_o(n)$ into their even- and odd-indexed samples, as follows

$$\left.\begin{aligned} X_e(k) &= \sum_{n=0}^{\frac{N}{4}-1} x_e(2n) W_{\frac{N}{4}}^{nk} + W_{\frac{N}{2}}^{k} \sum_{n=0}^{\frac{N}{4}-1} x_e(2n+1) W_{\frac{N}{4}}^{nk} \\ X_o(k) &= \sum_{n=0}^{\frac{N}{4}-1} x_o(2n) W_{\frac{N}{4}}^{nk} + W_{\frac{N}{2}}^{k} \sum_{n=0}^{\frac{N}{4}-1} x_o(2n+1) W_{\frac{N}{4}}^{nk} \end{aligned}\right\} \quad (131)$$

such that

$$\left.\begin{aligned} X_e(k) &= X_{ee}(k) + W_{\frac{N}{2}}^{k} X_{eo}(k) \\ X_o(k) &= X_{oe}(k) + W_{\frac{N}{2}}^{k} X_{oo}(k) \end{aligned}\right\} \quad (132)$$

where $X_{ee}(k)$, $X_{eo}(k)$, $X_{oe}(k)$, and $X_{oo}(k)$ correspond now to DFTs of length $\frac{N}{4}$.

Radix-2 algorithm with decimation in time

- Generically, in each step we compute DFTs of length L using DFTs of length $\frac{L}{2}$ as follows

$$X_i(k) = X_{ie}(k) + W_L^k X_{io}(k) \tag{133}$$

- A recursive application of the above procedure can convert the computation of a DFT of length $N = 2^l$ in l steps to the computation of 2^l DFTs of length 1, because each step converts a DFT of length L into two DFTs of length $\frac{L}{2}$ plus a complex product by W_L^k and a complex sum.

- Therefore, supposing that $\mathcal{M}(N)$ and $\mathcal{A}(N)$ are respectively the number of complex multiplications and additions to compute a DFT of length N the following relations hold:

Radix-2 algorithm with decimation in time

$$\mathcal{M}(\mathrm{N}) = 2\mathcal{M}\left(\frac{\mathrm{N}}{2}\right) + \mathrm{N} \tag{134}$$

$$\mathcal{A}(\mathrm{N}) = 2\mathcal{A}\left(\frac{\mathrm{N}}{2}\right) + \mathrm{N} \tag{135}$$

- In order to compute the values of $\mathcal{M}(\mathrm{N})$ and $\mathcal{A}(\mathrm{N})$, we must solve the recursive equations above.
  - The initial conditions are $\mathcal{M}(1) = 1$ and $\mathcal{A}(1) = 0$, since a length-1 DFT needs no additions and a multiplication by W_1^0 (these multiplications are trivial but must be considered in order to maintain coherence with equation (127) ; they will be discounted when stating the final result).

Radix-2 algorithm with decimation in time

- We can compute the number of multiplications employing the change of variables $N = 2^l$ and $T(l) = \frac{\mathcal{M}(N)}{N}$. With it, equation (134) becomes

$$T(l) = T(l-1) + 1 \tag{136}$$

Since $T(0) = \mathcal{M}(1) = 1$, then $T(l) = l + 1$. Therefore, we conclude that

$$\frac{\mathcal{M}(N)}{N} = 1 + \log_2 N \tag{137}$$

and thus

$$\mathcal{M}(N) = N + N \log_2 N \tag{138}$$

- Note that the above equation is coherent with equation (127).
- However, since we do not perform the trivial multiplications necessary to compute the N DFTs of size 1, we conclude that the actual number of multiplications is

$$\mathcal{M}(N) = N \log_2 N \tag{139}$$

Radix-2 algorithm with decimation in time

- In order to compute the number of additions, we can use the same change of variables, that is, we make $N = 2^l$ and $T(l) = \frac{\mathcal{A}(N)}{N}$. With it, equation (135) becomes

$$T(l) = T(l-1) + 1 \tag{140}$$

but this time $T(0) = \mathcal{A}(1) = 0$, and then $T(l) = l$. Therefore, we conclude that

$$\frac{\mathcal{A}(N)}{N} = \log_2 N \tag{141}$$

and thus

$$\mathcal{A}(N) = N \log_2 N \tag{142}$$

that is, the fast Fourier transform (FFT) can be computed using $N \log_2 N$ complex multiplications and additions, which comprises an economy of the order of $\frac{N}{\log_2 N}$, when compared to the direct implementation in equation (122).

Radix-2 algorithm with decimation in time

- This FFT algorithm is known as a decimation-in-time algorithm because it recursively divides the sequence $x(n)$ into subsequences.
- We can devise a graphical representation of the above procedure by noting that in the operation in equation (133) each value of either $X_{ie}(k)$ or $X_{io}(k)$ is used twice.
  - This is so because, if $X_i(k)$ has length L, then $X_{ie}(k)$ and $X_{io}(k)$ have length $\frac{L}{2}$ (and thus period $\frac{L}{2}$).
- The corresponding equations are:

$$X_i(k) = X_{ie}(k) + W_L^k X_{io}(k) \quad (143)$$

$$\begin{aligned} X_i\left(k+\frac{L}{2}\right) &= X_{ie}\left(k+\frac{L}{2}\right) + W_L^{k+\frac{L}{2}} X_{io}\left(k+\frac{L}{2}\right) \\ &= X_{ie}(k) + W_L^{k+\frac{L}{2}} X_{io}(k) \end{aligned} \quad (144)$$

Radix-2 algorithm with decimation in time

- Therefore, if we have the DFTs of length $\frac{L}{2}$, $X_{ie}(k)$ and $X_{io}(k)$, we can compute the length-L DFT $X_i(k)$ by applying equations (143) and (144), for $k = 0, 1, \ldots, \left(\frac{L}{2} - 1\right)$.
- This process is illustrated by the graph in Figure 8.
- Since the FFT algorithm is composed of repetitions of this procedure, it is often called the basic cell of the algorithm.
- Due to the appearance of its graph, the basic cell is often referred to as a butterfly.

Radix-2 algorithm with decimation in time

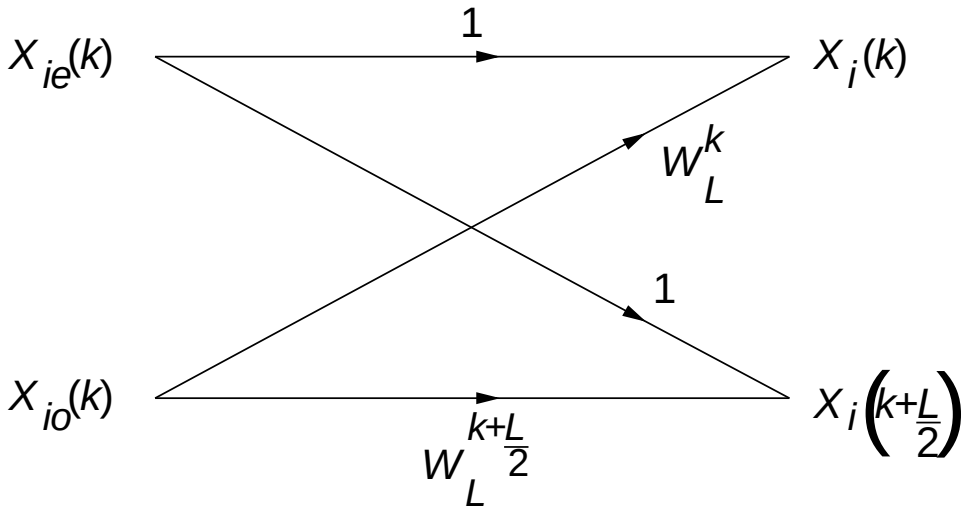


Figure 8: Basic cell of the decimation-in-time FFT algorithm.

- A graph for the complete FFT algorithm above is obtained by repeating the basic cell in Figure 8, for $\mathrm{L} = \mathrm{N}, \frac{\mathrm{N}}{2}, \ldots, 2$ and for $\mathrm{k} = 0, 1, \ldots, \left(\frac{\mathrm{L}}{2} - 1\right)$. Figure 9 illustrates the case when $\mathrm{N} = 8$.

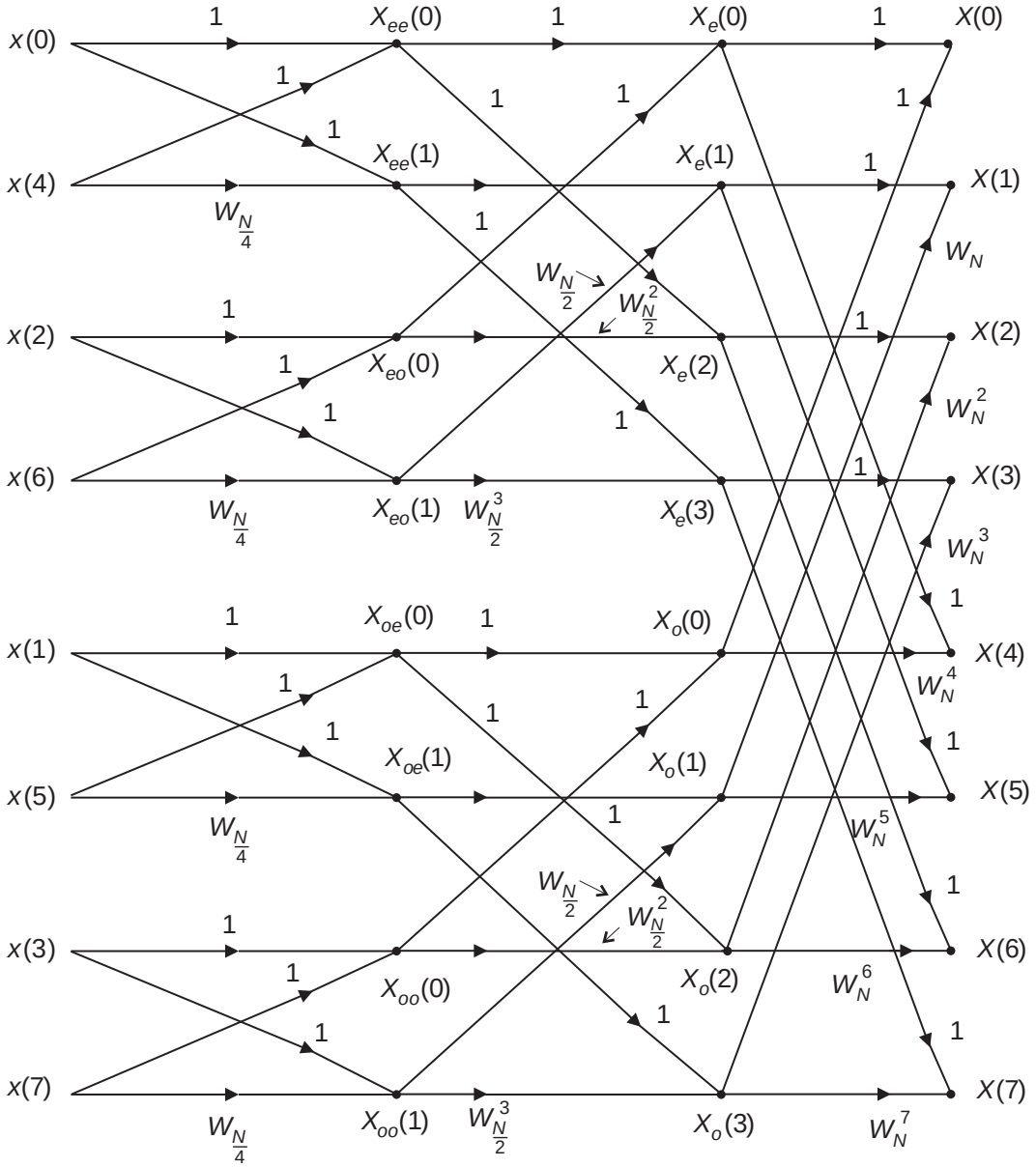


Figure 9: Graph of the decimation-in-time 8-point FFT algorithm.

Radix-2 algorithm with decimation in time

- The graph in Figure 9 has an interesting property. The nodes of one section of the graph depend only on nodes of the previous section of the graph.
  - For example, $X_i(k)$ depends only on $X_{ie}(k)$ and $X_{io}(k)$.
  - In addition, $X_{ie}(k)$ and $X_{io}(k)$ are only used to compute $X_i(k)$ and $X_i(k + \frac{L}{2})$.
  - Therefore, once computed, the values of $X_i(k)$ and $X_i(k + \frac{L}{2})$ can be stored in the same place as $X_{ie}(k)$ and $X_{io}(k)$.
- This implies that the intermediate results of the length-N FFT computation can be stored in a single size-N vector, that is, the results of section l can be stored in the same place as the results of section $l - 1$.
- This is why the computation of the intermediate results of the FFT is often said to be 'in place'.

Radix-2 algorithm with decimation in time

- Another important aspect of the FFT algorithm relates to the ordering of the input vector.
- As seen in Figure 9, the output vector is ordered sequentially while the input vector is not.
- A general rule for the ordering of the input vector can be devised by referring to the FFT graph in Figure 9.
- Moving from right to left, we note that in the second section, the upper half corresponds to the DFT of the even-indexed samples and the lower half to the DFT of the odd-indexed samples.
- Since the even-indexed samples have the least significant bit (LSB) equal to 0 and the odd-indexed samples have the LSB equal to 1, then the DFT of the upper half corresponds to samples with an index having LSB = 0 and the lower half to the DFT of samples with an index having LSB = 1.

Radix-2 algorithm with decimation in time

- Likewise, for each half, the upper half corresponds to the even-indexed samples belonging to this half, that is, the ones with the second LSB = 0. Similarly, the lower half corresponds to the second LSB = 1.
- If we proceed until we reach the input signal on the left, we end up with the uppermost sample having an index with all bits equal to zero, the second one having the first bit equal to one and the others equal to zero, and so on.
- We note, then, that the ordering of the input vector is the ascending order of the bit-reversed indexes.
  - For example, $x(3) = x(011)$ will be put in position 110 of the input, that is, position 6.

Radix-2 algorithm with decimation in time

- An additional economy in the number of complex multiplications of the algorithm can be obtained by noting that, in equation (144)

$$W_L^{k+\frac{L}{2}} = W_L^k W_L^{\frac{L}{2}} = W_L^k W_2 = -W_L^k \tag{145}$$

Then, equations (143) and (144) can be rewritten as

$$\left.\begin{aligned} X_i(k) &= X_{ie}(k) + W_L^k X_{io}(k) \\ X_i\left(k+\tfrac{L}{2}\right) &= X_{ie}(k) - W_L^k X_{io}(k) \end{aligned}\right\} \tag{146}$$

- This allows a more efficient implementation of the basic cell in Figure 8, using one complex multiplication instead of two.
- The resulting cell is depicted in Figure 10.

Radix-2 algorithm with decimation in time

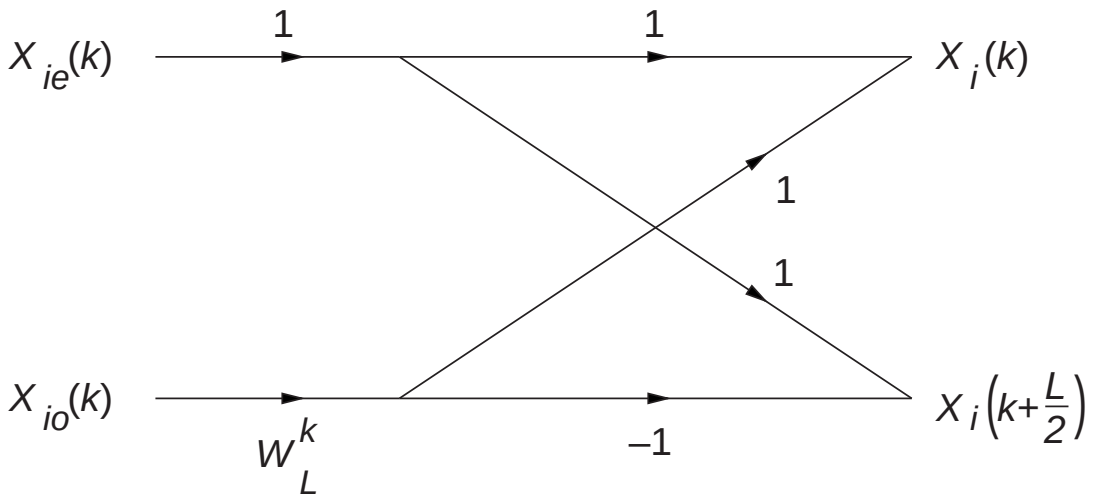


Figure 10: More efficient basic cell of the decimation-in-time FFT algorithm.

- Substituting the basic cells corresponding to Figure 8 by the ones corresponding to Figure 10, we have the more efficient graph for the FFT shown in Figure 11.

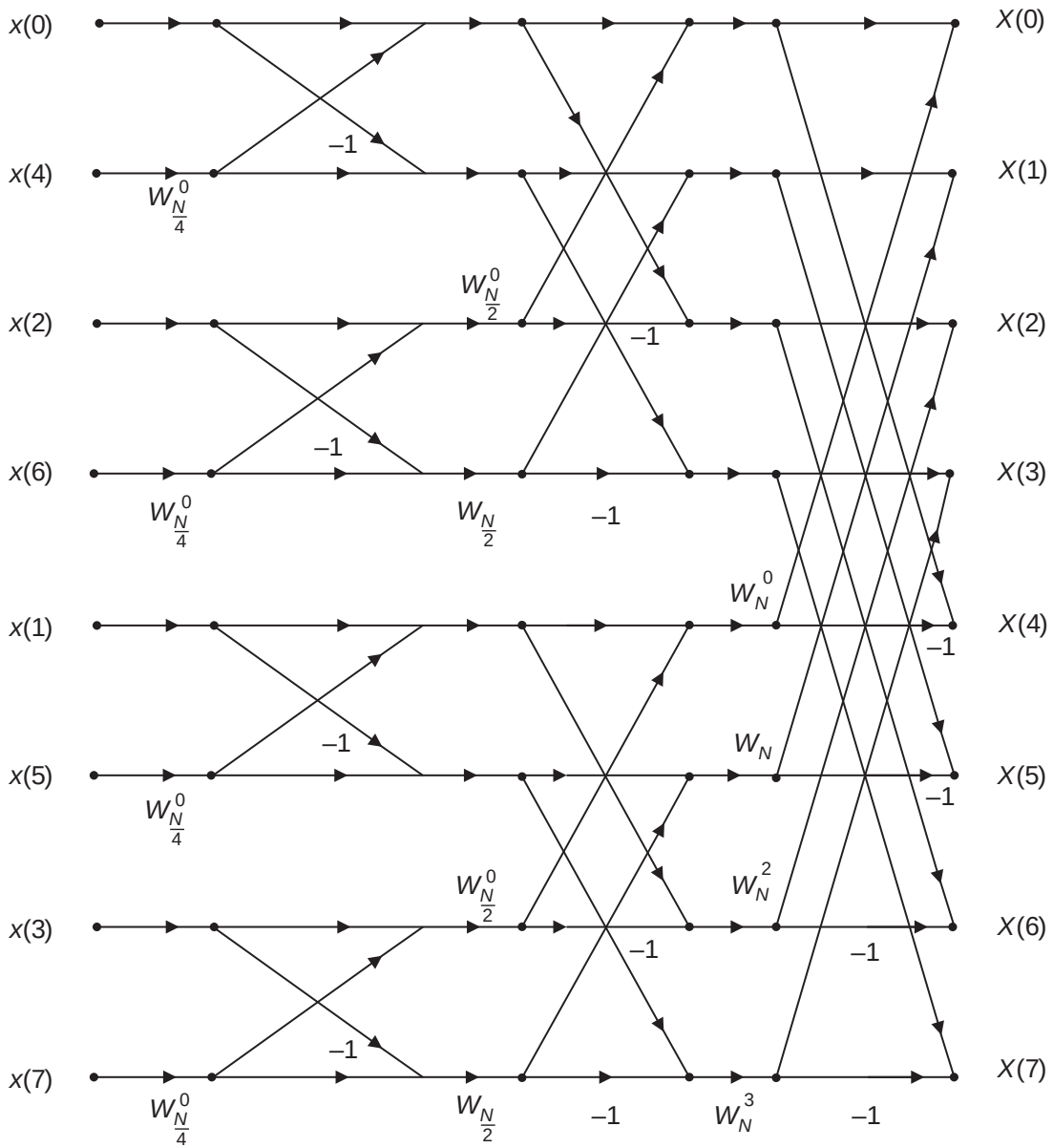


Figure 11: More efficient graph of the decimation-in-time 8-point FFT algorithm. The unmarked branches are equal to 1.

Radix-2 algorithm with decimation in time

- With this new basic cell, the number of complex multiplications has dropped to half, that is, there is a total of $\frac{N}{2}\log_2 N$ complex multiplications.
- In order to perform a more accurate calculation, we have to discount the remaining trivial multiplications.
  - One set of them are by $W_L^0 = 1$.
  - In equation (146), when the DFTs have length L, we have $\frac{N}{L}$ DFTs and thus the term W_L^0 appears $\frac{N}{L}$ times.
  - Then, we have $\frac{N}{2} + \frac{N}{4} + \cdots + \frac{N}{N} = N - 1$ multiplications by 1.
  - The other set of trivial multiplications are the ones by $W_L^{\frac{L}{4}} = -j$.
  - Since, in the first stage, there are no terms equal to $-j$ and, from the second stage on, the number of times a term $-j$ appears is the same as the number of times a term 1 appears, then we have $N - 1 - \frac{N}{2} = \frac{N}{2} - 1$ multiplications by $-j$.

Radix-2 algorithm with decimation in time

- This gives an overall number of nontrivial complex multiplications equal to

$$\mathcal{M}(\mathrm{N}) = \frac{\mathrm{N}}{2}\log_2 \mathrm{N} - \mathrm{N} + 1 - \frac{\mathrm{N}}{2} + 1 = \frac{\mathrm{N}}{2}\log_2 \mathrm{N} - \frac{3}{2}\mathrm{N} + 2 \qquad (147)$$

- Note that the number of complex additions remains $\mathcal{A}(\mathrm{N}) = \mathrm{N}\log_2 \mathrm{N}$.
- If we shuffle the horizontal branches of the graph in Figure 11 so that the input signal is in normal ordering, then the output of the graph comes in bit-reversed ordering.
  - In this case, we have another 'in place' algorithm.
- In fact, there are a plethora of forms for the FFT.
  - For example, there is an algorithm in which the input and output appear in normal ordering, but where there is no possibility of 'in place' computation.
- It is clear from equations (122) and (123) that, in order to compute the inverse DFT it suffices to change, in the graphs of Figures 9 or 11, the terms $W_{\mathrm{N}}^{\mathrm{k}}$ to $W_{\mathrm{N}}^{-\mathrm{k}}$ and divide the output of the graph by N.

Radix-2 algorithm with decimation in time

- An interesting interpretation of the FFT algorithms can be found if we look at the matrix form of the DFT in equation (39),

$$\mathbf{X} = \mathbf{W}_N \mathbf{x} \tag{148}$$

- The matrix form is widely used for describing fast transform algorithms.
- For instance, the decimation-in-time FFT algorithm can be represented in matrix form if we notice that equations (143) and (144), corresponding to the basic cell in Figure 8, can be expressed in matrix form as

$$\begin{bmatrix} X_i(k) \\ X_i(k+\frac{L}{2}) \end{bmatrix} = \begin{bmatrix} 1 & W_L^k \\ 1 & W_L^{k+\frac{L}{2}} \end{bmatrix} \begin{bmatrix} X_{ie}(k) \\ X_{io}(k) \end{bmatrix} \tag{149}$$

- Then the graph in Figure 9 can be expressed as

$$\mathbf{X} = \mathbf{F}_8^{(8)} \mathbf{F}_8^{(4)} \mathbf{F}_8^{(2)} \mathbf{P}_8 \mathbf{x} \tag{150}$$

where

$$\mathbf{P}_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{151}$$

corresponds to the bit-reversal operation onto the indexes of the input vector **x**,

$$\mathbf{F}_8^{(2)} = \begin{bmatrix} 1 & W_2^0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & W_2^1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & W_2^0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & W_2^1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & W_2^0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & W_2^1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & W_2^0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & W_2^1 \end{bmatrix} \tag{152}$$

corresponds to the basic cells of the first stage,

$$\mathbf{F}_8^{(4)} = \begin{bmatrix} 1 & 0 & W_4^0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W_4^1 & 0 & 0 & 0 & 0 \\ 1 & 0 & W_4^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W_4^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & W_4^0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W_4^1 \\ 0 & 0 & 0 & 0 & 1 & 0 & W_4^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W_4^3 \end{bmatrix} \tag{153}$$

corresponds to the basic cells of the second stage, and

$$\mathbf{F}_8^{(8)} = \begin{bmatrix} 1 & 0 & 0 & 0 & W_8^0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W_8^1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W_8^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W_8^3 \\ 1 & 0 & 0 & 0 & W_8^4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W_8^5 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W_8^6 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W_8^7 \end{bmatrix} \tag{154}$$

corresponds to the basic cells of the third stage.

- Comparing equations (148) and (150), one can see the FFT algorithm in Figure 9 as a factorization of the DFT matrix $\mathbf{W}_8$ such that

$$\mathbf{W}_8 = \mathbf{F}_8^{(8)}\mathbf{F}_8^{(4)}\mathbf{F}_8^{(2)}\mathbf{P}_8 \tag{155}$$

- This factorization corresponds to a fast algorithm because the matrices $\mathbf{F}_8^{(8)}$, $\mathbf{F}_8^{(4)}$, $\mathbf{F}_8^{(2)}$, and $\mathbf{P}_8$ have most elements equal to zero.

Radix-2 algorithm with decimation in time

- Because of that, the product by each matrix above can be effected at the expense of at most 8 complex multiplications.
- An exception if the product by $\mathbf{P}_8$, that, being just a permutation, requires no multiplications at all.
- Equations (150)–(155) exemplify the general fact that each different FFT algorithm corresponds to a different factorization of the DFT matrix $\mathbf{W}_N$ into sparse matrices.
  - For example, the reduction in complexity achieved by replacing Figure 8 (equations (143) and (144)) by Figure 10 (equation (146)) is equivalent to factoring the matrix in equation (149) as

$$\begin{bmatrix} 1 & W_L^k \\ 1 & W_L^{k+\frac{L}{2}} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & W_L^k \end{bmatrix} \tag{156}$$

Decimation in frequency

- An alternative algorithm for the fast computation of the DFT can be obtained using decimation in frequency, that is, division of $X(k)$ into subsequences. The algorithm is generated as follows:

$$
\begin{aligned}
X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{nk} \\
&= \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{nk} + \sum_{n=\frac{N}{2}}^{N-1} x(n) W_N^{nk} \\
&= \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{nk} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right) W_N^{\frac{N}{2}k} W_N^{nk} \\
&= \sum_{n=0}^{\frac{N}{2}-1} \left(x(n) + W_N^{\frac{N}{2}k} x\left(n+\frac{N}{2}\right) \right) W_N^{nk}
\end{aligned}
\tag{157}
$$

Decimation in frequency

- We can now compute the even and odd samples of $X(k)$ separately, that is

$$X(2l) = \sum_{n=0}^{\frac{N}{2}-1} \left(x(n) + W_N^{Nl} x\left(n + \frac{N}{2}\right) \right) W_N^{2nl}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} \left(x(n) + x\left(n + \frac{N}{2}\right) \right) W_N^{2nl} \qquad (158)$$

for $l = 0, 1, \ldots, \left(\frac{N}{2} - 1\right)$, and

Decimation in frequency

$$
\begin{aligned}
X(2l+1) &= \sum_{n=0}^{\frac{N}{2}-1} \left(x(n) + W_N^{(2l+1)\frac{N}{2}} x\left(n+\frac{N}{2}\right) \right) W_N^{(2l+1)n} \\
&= \sum_{n=0}^{\frac{N}{2}-1} \left(x(n) - x\left(n+\frac{N}{2}\right) \right) W_N^{(2l+1)n} \\
&= \sum_{n=0}^{\frac{N}{2}-1} \left[\left(x(n) - x\left(n+\frac{N}{2}\right) \right) W_N^{n} \right] W_N^{2ln} \qquad (159)
\end{aligned}
$$

for $l = 0, 1, \ldots, \left(\frac{N}{2} - 1\right)$.

Decimation in frequency

- Equations (158) and (159) can be recognized as DFTs of length $\frac{N}{2}$, since $W_N^{2ln} = W_{\frac{N}{2}}^{ln}$. Before computing these DFTs, we have to compute the two intermediate signals $s_e(n)$ and $s_o(n)$, of length $\frac{N}{2}$, given by

$$\left.\begin{aligned} s_e(n) &= x(n) + x\left(n + \frac{N}{2}\right) \\ s_o(n) &= \left(x(n) - x\left(n + \frac{N}{2}\right)\right) W_N^n \end{aligned}\right\} \tag{160}$$

- Naturally, the above procedure can be repeated for each of the DFTs of length $\frac{N}{2}$, thus generating DFTs of length $\frac{N}{4}$, $\frac{N}{8}$, and so on.

Decimation in frequency

- Therefore, the basic cell used in the decimation-in-frequency computation of the DFT is characterized by

$$\left.\begin{aligned} S_{ie}(n) &= S_i(n) + S_i\left(n+\frac{L}{2}\right) \\ S_{io}(n) &= \left(S_i(n) - S_i\left(n+\frac{L}{2}\right)\right) W_L^n \end{aligned}\right\} \tag{161}$$

  where $S_{ie}(n)$ and $S_{io}(n)$ have length $\frac{L}{2}$, and $S_i(n)$ has length L.

- The graph of such a basic cell is depicted in Figure 12.

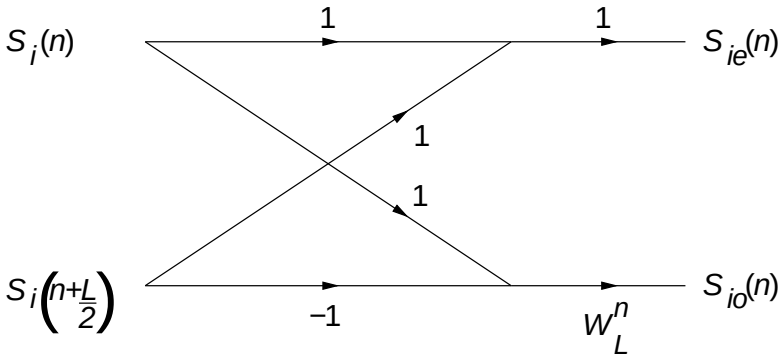


Figure 12: Basic cell of the decimation-in-frequency FFT algorithm.

Decimation in frequency

- Figure 13 shows the complete decimation-in-frequency FFT algorithm for $N = 8$.
- It is important to note that, in this algorithm, the input sequence $x(n)$ is in normal ordering and the output sequence $X(k)$ is in bit-reversed ordering.
- Also, comparing Figures 11 and 13, it is interesting to see that one graph is the transpose of the other.

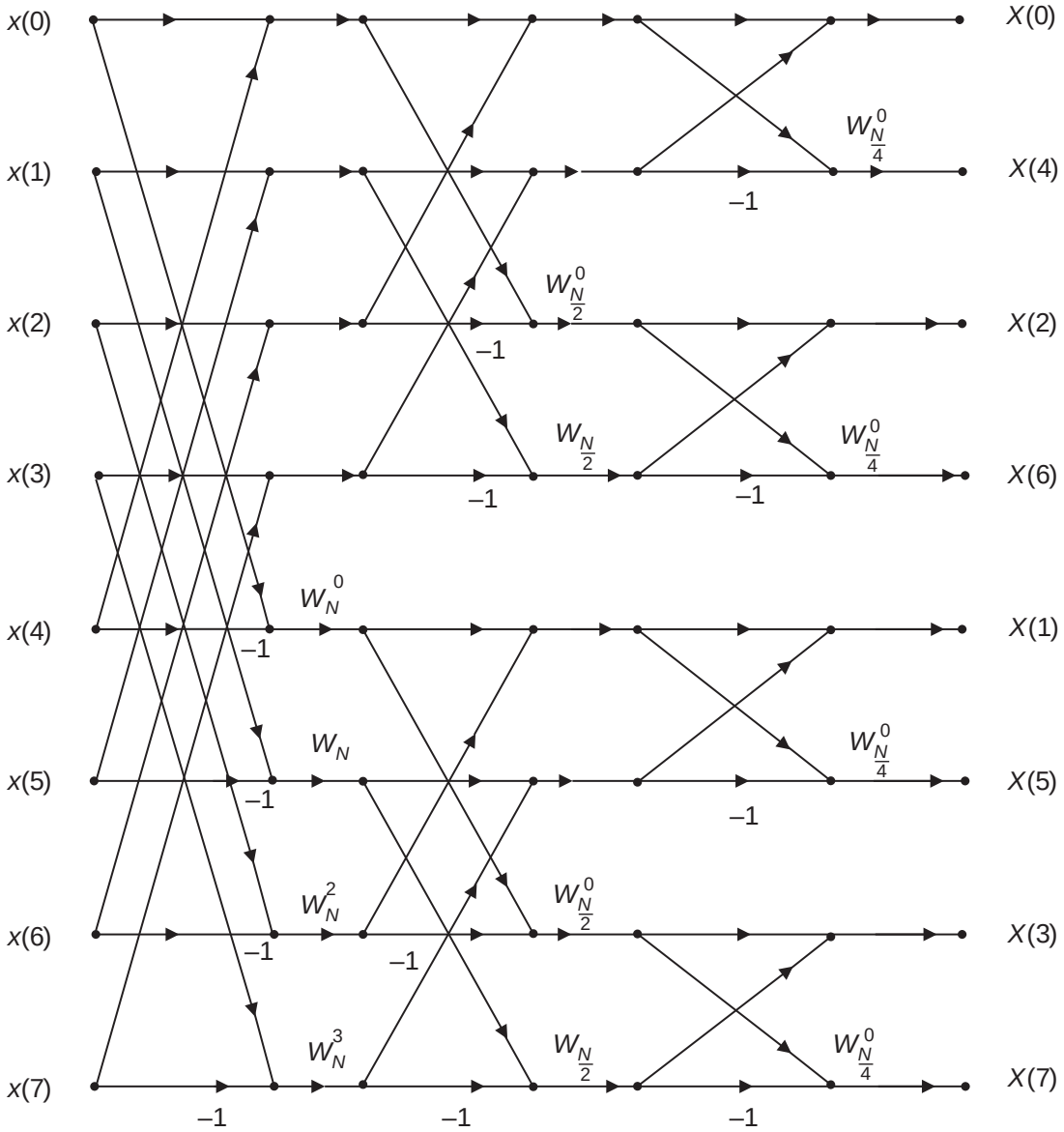


Figure 13: Graph of the decimation-in-frequency 8-point FFT algorithm.

Radix-4 algorithm

- If $N = 2^{2l}$, instead of using radix-2 algorithms, we can use radix-4 FFT algorithms, which can give us additional economy in the required number of complex multiplications.
- The derivation of the radix-4 algorithms parallels those of the radix-2 ones.
- If we use decimation-in-time, a length-N sequence is divided into 4 sequences of length $\frac{N}{4}$ such that

Radix-4 algorithm

$$
\begin{aligned}
X(k) &= \sum_{m=0}^{\frac{N}{4}-1} x(4m) W_N^{4mk} + \sum_{m=0}^{\frac{N}{4}-1} x(4m+1) W_N^{(4m+1)k} \\
&\quad + \sum_{m=0}^{\frac{N}{4}-1} x(4m+2) W_N^{(4m+2)k} + \sum_{m=0}^{\frac{N}{4}-1} x(4m+3) W_N^{(4m+3)k} \\
&= \sum_{m=0}^{\frac{N}{4}-1} x(4m) W_{\frac{N}{4}}^{mk} + W_N^k \sum_{m=0}^{\frac{N}{4}-1} x(4m+1) W_{\frac{N}{4}}^{mk} \\
&\quad + W_N^{2k} \sum_{m=0}^{\frac{N}{4}-1} x(4m+2) W_{\frac{N}{4}}^{mk} + W_N^{3k} \sum_{m=0}^{\frac{N}{4}-1} x(4m+3) W_{\frac{N}{4}}^{mk} \\
&= \sum_{l=0}^{3} W_N^{lk} \sum_{m=0}^{\frac{N}{4}-1} x(4m+l) W_{\frac{N}{4}}^{mk}
\end{aligned}
\tag{162}
$$

- We can rewrite the above equation as

$$X(k) = \sum_{l=0}^{3} W_N^{lk} F_l(k) \tag{163}$$

where each $F_l(k)$ can be computed using 4 DFTs of length $\frac{N}{16}$, as shown below

$$\begin{aligned}
F_l(k) &= \sum_{m=0}^{\frac{N}{4}-1} x(4m+l) W_{\frac{N}{4}}^{mk} \\
&= \sum_{q=0}^{\frac{N}{16}-1} x(16q+l) W_{\frac{N}{4}}^{4qk} + W_{\frac{N}{4}}^{k} \sum_{q=0}^{\frac{N}{16}-1} x(16q+4+l) W_{\frac{N}{4}}^{4qk} \\
&\quad + W_{\frac{N}{4}}^{2k} \sum_{q=0}^{\frac{N}{16}-1} x(16q+8+l) W_{\frac{N}{4}}^{4qk} + W_{\frac{N}{4}}^{3k} \sum_{q=0}^{\frac{N}{16}-1} x(16q+12+l) W_{\frac{N}{4}}^{4qk} \\
&= \sum_{r=0}^{3} W_{\frac{N}{16}}^{rk} \sum_{q=0}^{\frac{N}{16}-1} x(16q+4r+l) W_{\frac{N}{16}}^{qk}
\end{aligned} \tag{164}$$

Radix-4 algorithm

- This process is recursively applied until we have to compute $\frac{N}{4}$ DFTs of length 4.
- From equation (163), we can see that the basic cell implements the equations below when computing a DFT $S(k)$ of length L using 4 DFTs of length $\frac{L}{4}$, $S_l(k),\ l = 0, 1, 2, 3$.

$$\left.\begin{aligned}
S(k) &= \sum_{l=0}^{3} W_L^{lk} S_l(k) \\
S\left(k + \tfrac{L}{4}\right) &= \sum_{l=0}^{3} W_L^{l(k+\frac{L}{4})} S_l(k) = \sum_{l=0}^{3} W_L^{lk} (-j)^l S_l(k) \\
S\left(k + \tfrac{L}{2}\right) &= \sum_{l=0}^{3} W_L^{l(k+\frac{L}{2})} S_l(k) = \sum_{l=0}^{3} W_L^{lk} (-1)^l S_l(k) \\
S\left(k + \tfrac{3L}{4}\right) &= \sum_{l=0}^{3} W_L^{l(k+\frac{3L}{4})} S_l(k) = \sum_{l=0}^{3} W_L^{lk} (j)^l S_l(k)
\end{aligned}\right\} \quad (165)$$

- The corresponding graph for the radix-4 butterfly is shown in Figure 14.

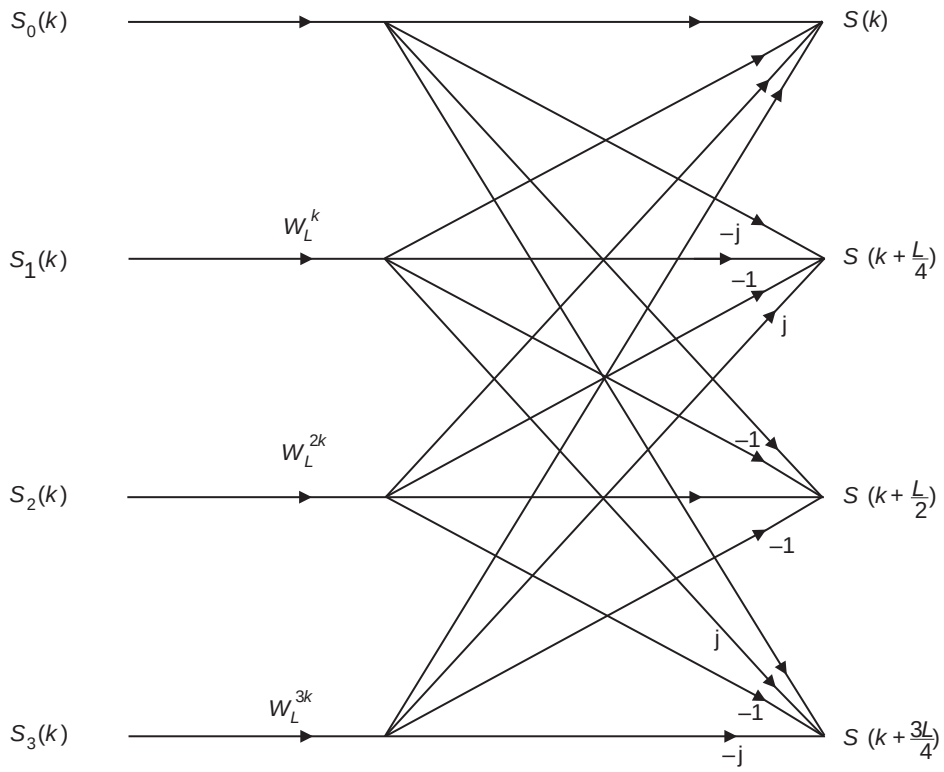


Figure 14: Basic cell of the radix-4 FFT algorithm.

- As an illustration of the radix-4 algorithm, Figure 15 shows a sketch of the computation of a DFT of length 64 using a radix-4 FFT.

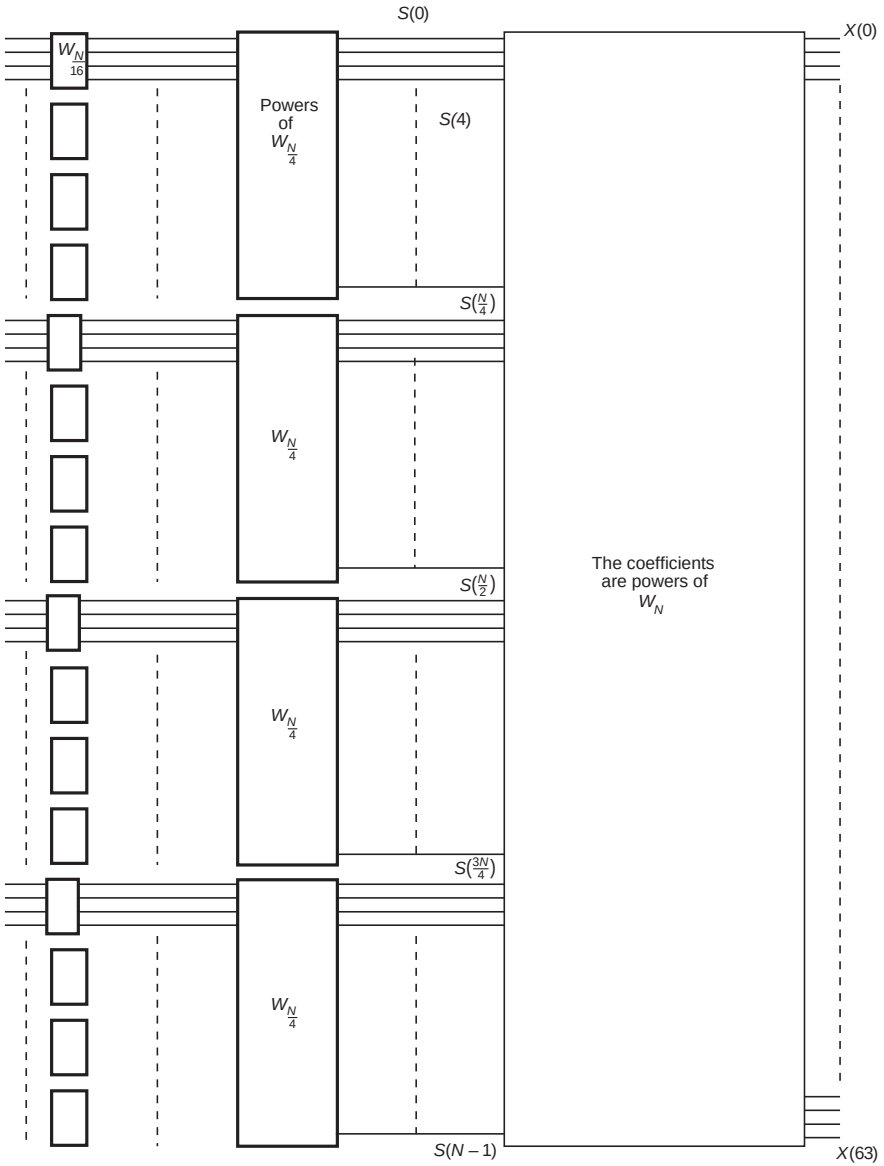


Figure 15: Sketch of a length-64 DFT using a radix-4 FFT algorithm.

Radix-4 algorithm

- As can be deduced from Figure 14 and equation (164), at each stage of the application of the radix-4 FFT algorithm we need N complex multiplications and $3\mathrm{N}$ complex additions, giving a total number of complex operations of

$$\mathcal{M}(\mathrm{N}) = \mathrm{N}\log_4 \mathrm{N} = \frac{\mathrm{N}}{2}\log_2 \mathrm{N} \tag{166}$$

$$\mathcal{A}(\mathrm{N}) = 3\mathrm{N}\log_4 \mathrm{N} = \frac{3\mathrm{N}}{2}\log_2 \mathrm{N} \tag{167}$$

- Apparently, the radix-4 algorithms do not present any advantage when compared to radix-2 algorithms.

Radix-4 algorithm

- However, the number of additions in the radix-4 basic cell can be decreased if we note from Figure 14 and equation (165) that the quantities

$$\left.\begin{aligned} & W_L^0 S_0(k) + W_L^{2k} S_2(k) \\ & W_L^0 S_0(k) - W_L^{2k} S_2(k) \\ & W_L^k S_1(k) + W_L^{3k} S_3(k) \\ & W_L^k S_1(k) - W_L^{3k} S_3(k) \end{aligned}\right\} \quad (168)$$

  are each computed twice, unnecessarily.

- By exploiting this fact, the more economical basic cell for the radix-4 algorithm shown in Figure 16 results, and we decrease the number of complex additions to $2N$ per stage, instead of $3N$.

Radix-4 algorithm

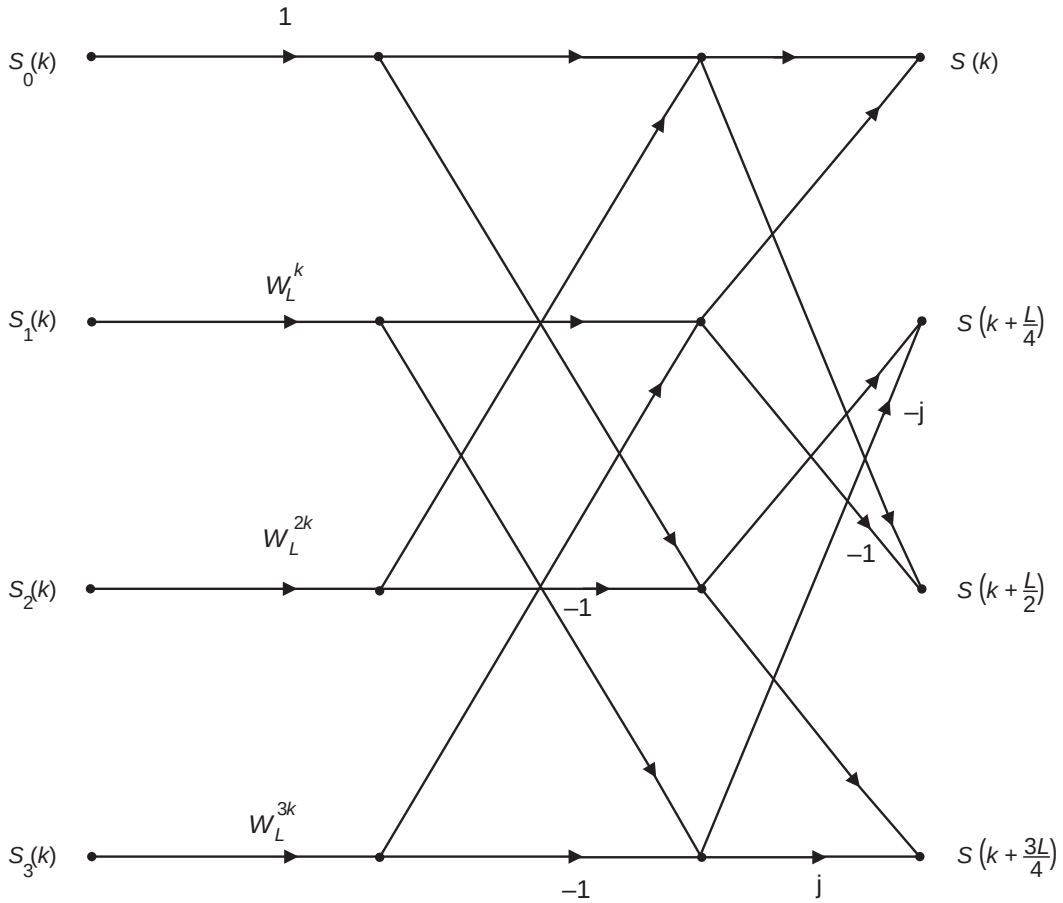


Figure 16: More efficient basic cell of the radix-4 FFT algorithm.

Radix-4 algorithm

- The number of multiplications can be further decreased if we do not consider the multiplications by W_N^0.
- There is one of them in each basic cell, and three more of them in the basic cells corresponding to the index $k = 0$.
- Since there are $\log_4 N$ stages, we have, in the former case, $\frac{N}{4} \log_4 N$ elements W_N^0, while the number of elements corresponding to $k = 0$ is $3(1 + 4 + 16 + \cdots + \frac{N}{4}) = (N - 1)$.
- Therefore, the total number of multiplications is given by

$$\mathcal{M}(N) = N \log_4 N - \frac{N}{4} \log_4 N - N + 1 = \frac{3}{8} N \log_2 N - N + 1 \qquad (169)$$

- Some additional trivial multiplications can also be detected.

Radix-4 algorithm

- If we then compare equations (147) and (169), we note that the radix-4 algorithm can be more economical, in terms of the number of overall complex multiplications, than the radix-2 algorithm.
- It is important to point out that, in general, the shorter the length of the DFTs of the basic cell of a FFT algorithm, the more efficient it is.
  - The exceptions to this rule are the radix-4, -8, -16,. . ., algorithms, where we can obtain a smaller number of multiplications than in radix-2 algorithms.
- Although the radix-4 algorithm introduced here was based on the decimation-in-time approach, a similar algorithm based on the decimation-in-frequency method could be readily obtained.

Fast Fourier Transform

Example 3.9

Derive the butterfly of the radix-3 DFT exploiting the possible savings in the number of multiplications and additions.

Fast Fourier Transform

Solution

$$\begin{aligned} X(k) &= \sum_{n=0}^{2} x(n) W_N^{nk} \\ &= x(0) + W_3^k x(1) + W_3^{2k} x(2) \\ &= x(0) + e^{-j\frac{2\pi}{3}k} x(1) + e^{-j\frac{4\pi}{3}k} x(2) \end{aligned} \tag{170}$$

- Using the above equation and discounting the trivial multiplications, one can compute the radix-3 butterfly using 6 additions (2 for each value of k) and 4 complex multiplications (2 for $k = 1$ and 2 for $k = 2$).

Fast Fourier Transform

- By further developing equation (170), we have

$$
\begin{aligned}
X(0) &= x(0) + x(1) + x(2) && (171)\\
X(1) &= x(0) + e^{-j\frac{2\pi}{3}} x(1) + e^{-j\frac{4\pi}{3}} x(2)\\
&= x(0) + e^{-j\frac{2\pi}{3}} x(1) + e^{j\frac{2\pi}{3}} x(2)\\
&= x(0) + W_3 x(1) + W_3^{-1} x(2) && (172)\\
X(2) &= x(0) + e^{-j\frac{4\pi}{3}} x(1) + e^{-j\frac{8\pi}{3}} x(2)\\
&= x(0) + e^{j\frac{2\pi}{3}} x(1) + e^{\frac{4\pi}{3}} x(2)\\
&= x(0) + e^{j\frac{2\pi}{3}} \left(x(1) + e^{j\frac{2\pi}{3}} x(2) \right)\\
&= x(0) + W_3^{-1} \left(x(1) + W_3^{-1} x(2) \right) && (173)
\end{aligned}
$$

- From equations (171) to (173), we can compute the radix-3 butterfly using the graph in Figure 17, that uses 6 additions and 3 complex multiplications, a saving of 1 complex multiplication relative to the solution in equation (170).

Fast Fourier Transform

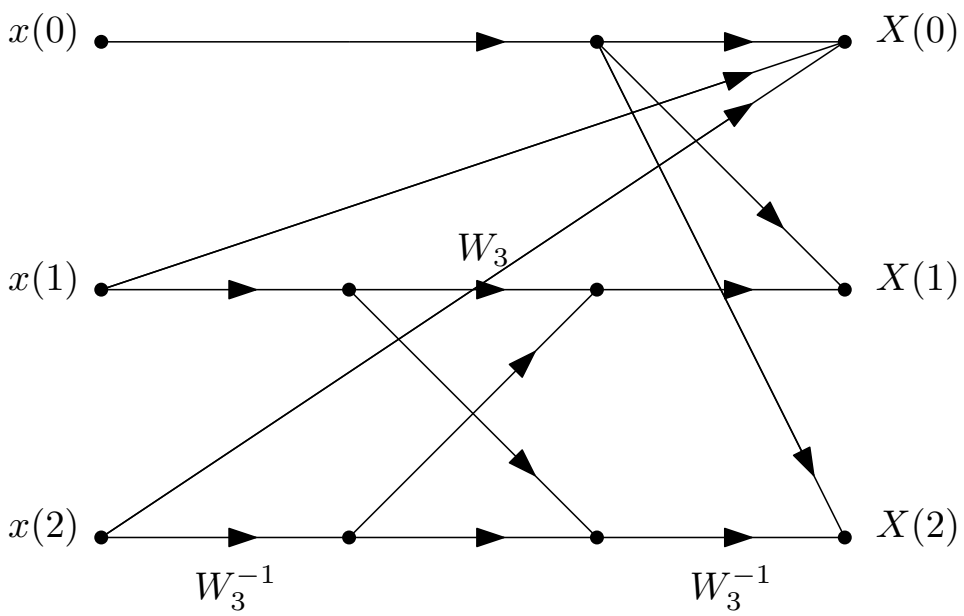


Figure 17: Efficient butterfly for the radix-3 DFT in Example 153.

△

Algorithms for arbitrary values of N

- Efficient algorithms for the computation of DFTs having a generic length N are possible provided that N is not prime.
- In such cases, we can decompose N as a product of factors

$$N = N_1 N_2 N_3 \cdots N_l = N_1 N_{2\to l} \tag{174}$$

  where $N_{2\to l} = N_2 N_3 \cdots N_l$.
- We can then initially divide the input sequence into N_1 sequences of length $N_{2\to l}$, thus writing the DFT of $x(n)$ as

$$
\begin{aligned}
X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{nk} \\
&= \sum_{m=0}^{N_{2\to 1}-1} x(N_1 m) W_N^{mN_1 k} + \sum_{m=0}^{N_{2\to 1}-1} x(N_1 m+1) W_N^{mN_1 k+k} + \cdots \\
&\quad + \sum_{m=0}^{N_{2\to 1}-1} x(N_1 m+N_1-1) W_N^{mN_1 k+(N_1-1)k} \\
&= \sum_{m=0}^{N_{2\to 1}-1} x(N_1 m) W_{N_{2\to 1}}^{mk} + W_N^k \sum_{m=0}^{N_{2\to 1}-1} x(N_1 m+1) W_{N_{2\to 1}}^{mk} \\
&\quad + W_N^{2k} \sum_{m=0}^{N_{2\to 1}-1} x(N_1 m+2) W_{N_{2\to 1}}^{mk} + \cdots \\
&\quad + W_N^{(N_1-1)k} \sum_{m=0}^{N_{2\to 1}-1} x(N_1 m+N_1-1) W_{N_{2\to 1}}^{mk} \\
&= \sum_{r=0}^{N_1-1} W_N^{rk} \sum_{m=0}^{N_{2\to 1}-1} x(N_1 m+r) W_{N_{2\to 1}}^{mk}
\end{aligned}
\tag{175}
$$

Algorithms for arbitrary values of N

- This equation can be interpreted as being the computation of a length-N DFT using N_1 DFTs of length $N_{2\rightarrow l}$.
- We then need $N_1 N_{2\rightarrow l}^2$ complex multiplications to compute the N_1 mentioned DFTs, plus $N(N_1 - 1)$ complex multiplications to compute the products of W_N^{rk} with the N_1 DFTs.
- We can continue this process by computing each of the N_1 DFTs of length $N_{2\rightarrow l}$ using N_2 DFTs of length $N_{3\rightarrow l}$, where $N_{3\rightarrow l} = N_3 N_4 \cdots N_l$, and so on, until all the DFTs have length N_l.

Algorithms for arbitrary values of N

- It can be shown that in such a case the total number of complex multiplications is given by

$$\mathcal{M}(\mathrm{N}) = \mathrm{N}(\mathrm{N}_1 + \mathrm{N}_2 + \cdots + \mathrm{N}_{l-1} + \mathrm{N}_l - l) \tag{176}$$

  - For example, if $\mathrm{N} = 63 = 3 \times 3 \times 7$ we have that $\mathcal{M}(\mathrm{N}) = 63(3 + 3 + 7 - 3) = 630$.
  - It is interesting to note that in order to compute a length-64 FFT we need only 384 complex multiplications if we use a radix-2 algorithm.

- This example reinforces the idea that we should, as a rule of thumb, divide N into factors as small as possible.

- In practice, whenever possible, the input sequences are zero-padded to force N to be a power of 2.

- As seen in the example above, this usually leads to an overall economy in the number of multiplications.

Alternative techniques for determining the DFT

- The algorithms for efficient computation of the DFT presented in the previous subsections are generically called FFT algorithms.
- They were, for a long time, the only known methods to enable the efficient computation of long DFTs.
- In 1976, however, Winograd showed that there are algorithms with smaller complexity than the FFTs.
- These algorithms are based on convolution calculations exploring ‘number-theoretic’ properties of the addresses of the data.
- These algorithms are denominated Winograd Fourier transform (WFT) algorithms.
- In terms of practical implementations, the WFT has a smaller number of complex multiplications than the FFT, at the expense of a more complex algorithm.

Alternative techniques for determining the DFT

- This gives an advantage to the WFT in many cases. However, the FFTs are more modular, which is an advantage in hardware implementations, especially in very large scale integration (VLSI).
- The main disadvantage of the WFT in this case is the complexity of the control path.
- Therefore, when implemented in special-purpose digital signal processors (DSPs) the FFTs have a clear advantage.
- This is so because multiplications are not a problem in such processors, and the more complex algorithms of the WFT make it slower than the FFTs in most cases.
- Another class of techniques for the computation of convolutions and DFTs is given by the number-theoretic transform (NTT), which explores number-theoretic properties of the data.
- NTT techniques have practical implementations in machines with modular arithmetic.
- Also, they are useful for computations using hardware based on residue arithmetic.

Other discrete transforms

- As seen in the previous sections, the DFT is a natural discrete-frequency representation for finite-length discrete signals, consisting of uniformly spaced samples of the Fourier transform.
- The direct and inverse DFTs can be expressed in matrix form as given by equations (39) and (40), repeated here for the reader's convenience

$$\mathbf{X} = \mathbf{W}_N \mathbf{x} \tag{177}$$

$$\mathbf{x} = \frac{1}{N} \mathbf{W}_N^* \mathbf{X} \tag{178}$$

where $\{\mathbf{W}_N\}_{ij} = W_N^{ij}$.

- Now, let $\mathbf{A}_N$ be a matrix of dimensions $N \times N$, such that

$$\mathbf{A}_N^{-1} = \gamma \mathbf{A}_N^{*^T} \tag{179}$$

where the superscript ^{T} and asterisk denote the matrix transposition and complex conjugate operations, respectively, and γ is a constant.

Other discrete transforms

- Using $\mathbf{A}_N$, we can generalize the definition in equations (177) and (178) to

$$\mathbf{X} = \mathbf{A}_N \mathbf{x} \tag{180}$$

$$\mathbf{x} = \gamma \mathbf{A}_N^{*^T} \mathbf{X} \tag{181}$$

- Equations (180) and (181) represent several discrete transforms which are frequently employed in signal processing applications.
- In the next subsection we will see that Parseval's theorem is also valid for discrete transforms in general, and discuss some of its implications.

Discrete transforms and Parseval's theorem

- Before proceeding, it is interesting to define some important operations between two vectors $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{C}^N$:
  - The inner product between $\mathbf{v}_2$ and $\mathbf{v}_1$ is defined as

$$\langle \mathbf{v}_2, \mathbf{v}_1 \rangle = \mathbf{v}_1^{*^T} \mathbf{v}_2 \tag{182}$$

  - The norm of vector $\mathbf{v}$ is defined as

$$\|\mathbf{v}\|^2 = \langle \mathbf{v}, \mathbf{v} \rangle = \mathbf{v}^{*^T} \mathbf{v} \tag{183}$$

  - The angle θ between two vectors $\mathbf{v}_2$ and $\mathbf{v}_1$ is defined as

$$\cos\theta = \frac{\langle \mathbf{v}_2, \mathbf{v}_1 \rangle}{\|\mathbf{v}_1\| \|\mathbf{v}_2\|} = \frac{\mathbf{v}_1^{*^T} \mathbf{v}_2}{\|\mathbf{v}_1\| \|\mathbf{v}_2\|} \tag{184}$$

Discrete transforms and Parseval's theorem

- Given two length-N signals $x_1(n)$ and $x_2(n)$ ($\mathbf{x}_1$ and $\mathbf{x}_2$ in vector form), and their respective transforms $X_1(k)$ and $X_2(k)$ ($\mathbf{X}_1$ and $\mathbf{X}_2$ in vector form), according to equations (180) and (181), we have that

$$\begin{aligned}
\sum_{k=0}^{N-1} X_1(k)X_2^*(k) &= \mathbf{X}_2^{*^T}\mathbf{X}_1 \\
&= (\mathbf{A}_N\mathbf{x}_2)^{*^T}\mathbf{A}_N\mathbf{x}_1 = \mathbf{x}_2^{*^T}\mathbf{A}_N^{*^T}\mathbf{A}_N\mathbf{x}_1 \\
&= \mathbf{x}_2^{*^T}\left(\frac{1}{\gamma}\mathbf{A}_N^{-1}\right)\mathbf{A}_N\mathbf{x}_1 = \frac{1}{\gamma}\mathbf{x}_2^{*^T}\mathbf{x}_1 \\
&= \frac{1}{\gamma}\sum_{n=0}^{N-1} x_1(n)x_2^*(n) \qquad (185)
\end{aligned}$$

- The above equation when $\gamma = \frac{1}{N}$ is equivalent to the Parseval's relation in equation (87).

Discrete transforms and Parseval's theorem

- If $x_1(n) = x_2(n) = x(n)$, equation (185) becomes

$$\|\mathbf{X}\|^2 = \frac{1}{\gamma} \|\mathbf{x}\|^2 \tag{186}$$

- An interesting property of transforms as defined by equations (180) and (181) is related to the angle between two vectors as defined by equation (184).

Discrete transforms and Parseval's theorem

- We have that the angles $\theta_{\mathbf{x}}$ between $\mathbf{x}_1$ and $\mathbf{x}_2$ and $\theta_{\mathbf{X}}$ between their transforms $\mathbf{X}_1$ and $\mathbf{X}_2$ satisfy

$$\begin{aligned}
\cos\theta_{\mathbf{x}} &= \frac{\mathbf{x}_1^{*^{\mathrm{T}}}\mathbf{x}_2}{\|\mathbf{x}_1\|\,\|\mathbf{x}_2\|} \\
&= \frac{(\gamma\mathbf{A}_N^{*^{\mathrm{T}}}\mathbf{X}_1)^{*^{\mathrm{T}}}(\gamma\mathbf{A}_N^{*^{\mathrm{T}}}\mathbf{X}_2)}{\sqrt{\gamma}\|\mathbf{X}_1\|\,\sqrt{\gamma}\|\mathbf{X}_2\|} \\
&= \frac{\gamma\mathbf{X}_1^{*^{\mathrm{T}}}\mathbf{A}_N\gamma\left(\frac{1}{\gamma}\mathbf{A}_N^{-1}\right)\mathbf{X}_2}{\gamma\|\mathbf{X}_1\|\,\|\mathbf{X}_2\|} \\
&= \frac{\mathbf{X}_1^{*^{\mathrm{T}}}\mathbf{X}_2}{\|\mathbf{X}_1\|\,\|\mathbf{X}_2\|} \\
&= \cos\theta_{\mathbf{X}} \qquad (187)
\end{aligned}$$

that is, transforms do not change the angles between vectors.

Discrete transforms and Parseval's theorem

- A special case of transforms is when $\gamma = 1$. These are referred to as unitary transforms.
- For unitary transforms, equation (186) means that the energy in the transform domain is equal to the energy in the time domain, or, equivalently, unitary transforms do not change the length of vectors.
- If we consider this property together with the one expressed by equation (187), we see that neither angles nor lengths are changed under unitary transforms.
  - This is equivalent to saying that unitary transforms are just rotations in $\mathbb{C}^N$.

Discrete transforms and Parseval's theorem

- Note that $\gamma = \frac{1}{N}$ for the DFT as defined in equations (177) and (178).
  - A unitary definition of the DFT would be

$$\mathbf{X} = \frac{1}{\sqrt{N}}\mathbf{W}_N\mathbf{x} \tag{188}$$

$$\mathbf{x} = \frac{1}{\sqrt{N}}\mathbf{W}_N^*\mathbf{X} \tag{189}$$

where $\{\mathbf{W}_N\}_{ij} = W_N^{ij}$. This unitary version is often used when one needs strict energy conservation between time and frequency domains.

Discrete transforms and orthogonality

- Two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ are orthogonal if their inner product is null, that is,

$$\langle \mathbf{v}_2, \mathbf{v}_1 \rangle = \mathbf{v}_1^{*^T} \mathbf{v}_2 = 0 \tag{190}$$

- Equation (190), together with equation (184), implies that the angle between two orthogonal vectors is $\theta = \frac{\pi}{2}$.

- The transforms as defined in equation (179) have an interesting interpretation when we consider the orthogonality concept.

- We can express matrix $\mathbf{A}_N$ as composed by its rows:

$$\mathbf{A}_N = \begin{bmatrix} \mathbf{a}_0^{*^T} \\ \mathbf{a}_1^{*^T} \\ \vdots \\ \mathbf{a}_{N-1}^{*^T} \end{bmatrix} \tag{191}$$

where $\mathbf{a}_k^{*^T}$ is the kth row of matrix $\mathbf{A}_N$.

Discrete transforms and orthogonality

- As we have seen above, the transform definition in equations (180) and (181) implies equation (179), which is equivalent to

$$\mathbf{A}_N \mathbf{A}_N^{*^T} = \frac{1}{\gamma} \mathbf{I}_N \tag{192}$$

Expressing $\mathbf{A}_N$ as in equation (191), the above equation becomes

$$
\begin{aligned}
\mathbf{A}_N \mathbf{A}_N^{*^T} &= \begin{bmatrix} \mathbf{a}_0^{*^T} \\ \mathbf{a}_1^{*^T} \\ \vdots \\ \mathbf{a}_{N-1}^{*^T} \end{bmatrix} \begin{bmatrix} \mathbf{a}_0 & \mathbf{a}_1 & \cdots & \mathbf{a}_{N-1} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{a}_0^{*^T}\mathbf{a}_0 & \mathbf{a}_0^{*^T}\mathbf{a}_1 & \cdots & \mathbf{a}_0^{*^T}\mathbf{a}_{N-1} \\ \mathbf{a}_1^{*^T}\mathbf{a}_0 & \mathbf{a}_1^{*^T}\mathbf{a}_1 & \cdots & \mathbf{a}_1^{*^T}\mathbf{a}_{N-1} \\ \vdots & \vdots & \cdots & \vdots \\ \mathbf{a}_{N-1}^{*^T}\mathbf{a}_0 & \mathbf{a}_{N-1}^{*^T}\mathbf{a}_1 & \cdots & \mathbf{a}_{N-1}^{*^T}\mathbf{a}_{N-1} \end{bmatrix} \\
&= \frac{1}{\gamma}\mathbf{I}_N \\
&= \frac{1}{\gamma}\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}
\end{aligned}
\tag{193}
$$

Discrete transforms and orthogonality

which is the same as saying that

$$\mathbf{a}_k^{*^T}\mathbf{a}_l = \frac{1}{\gamma}\delta(k-l) \tag{194}$$

- Therefore, we can conclude that the rows of a transform matrix $\mathbf{A}_N$ are orthogonal, that is, the angle between any pair of distinct rows is equal to $\frac{\pi}{2}$.
- In addition, the constant γ is such that

$$\gamma = \frac{1}{\mathbf{a}_k^{*^T}\mathbf{a}_k} = \frac{1}{\|\mathbf{a}_k\|^2} \tag{195}$$

Discrete transforms and orthogonality

- Now, expressing the direct transform in equation (180) as a function of the rows of $\mathbf{A}_N$ we have that

$$\mathbf{X} = \begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} \mathbf{a}_0^{*^T} \\ \mathbf{a}_1^{*^T} \\ \vdots \\ \mathbf{a}_{N-1}^{*^T} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{a}_0^{*^T}\mathbf{x} \\ \mathbf{a}_1^{*^T}\mathbf{x} \\ \vdots \\ \mathbf{a}_{N-1}^{*^T}\mathbf{x} \end{bmatrix} \tag{196}$$

that is,

$$X(k) = \mathbf{a}_k^{*^T}\mathbf{x} = \langle \mathbf{x}, \mathbf{a}_k \rangle \tag{197}$$

Discrete transforms and orthogonality

- Likewise, expressing the inverse transform in equation (181) as a function of the columns of $\mathbf{A}_N^{*^T}$, we have that

$$\begin{aligned}
\mathbf{x} &= \gamma \begin{bmatrix} \mathbf{a}_0 & \mathbf{a}_1 & \cdots & \mathbf{a}_{N-1} \end{bmatrix} \mathbf{X} \\
&= \gamma \begin{bmatrix} \mathbf{a}_0 & \mathbf{a}_1 & \cdots & \mathbf{a}_{N-1} \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix} \\
&= \sum_{k=0}^{N-1} \gamma X(k) \mathbf{a}_k
\end{aligned} \tag{198}$$

- The above equation means that a transform expresses a vector $\mathbf{x}$ as a linear combination of N orthogonal vectors $\mathbf{a}_k$, for $k = 0, 1, \ldots, (N-1)$.

Discrete transforms and orthogonality

- The coefficients of this linear combination are proportional to the transform coefficients $\mathrm{X}(\mathrm{k})$.
- Furthermore, from equation (197), $\mathrm{X}(\mathrm{k})$ is equal to the inner product between the vector **x** and the vector $\mathbf{a}_\mathrm{k}$.
- We take this interpretation one step further by observing that orthogonality of the vectors $\mathbf{a}_\mathrm{k}$ imply that γ is given by equation (195).

Discrete transforms and orthogonality

- Replacing this value of γ and the value of $X(k)$ from equation (197) in equation (198), we get

$$\begin{aligned}
\mathbf{x} &= \sum_{k=0}^{N-1} \underbrace{\left(\frac{1}{\|\mathbf{a}_k\|^2}\right)}_{\gamma} \underbrace{\left(\mathbf{a}_k^{*T}\mathbf{x}\right)}_{X(k)} \mathbf{a}_k \\
&= \sum_{k=0}^{N-1} \left(\frac{\mathbf{a}_k^{*T}\mathbf{x}}{\|\mathbf{a}_k\|}\right) \frac{\mathbf{a}_k}{\|\mathbf{a}_k\|} \\
&= \sum_{k=0}^{N-1} \left\langle \mathbf{x}, \frac{\mathbf{a}_k}{\|\mathbf{a}_k\|} \right\rangle \frac{\mathbf{a}_k}{\|\mathbf{a}_k\|}
\end{aligned} \tag{199}$$

- From the above equation, one can interpret the transform as a representation of a signal using an orthogonal basis of vectors $\mathbf{a}_k$, for $k = 0, 1, \ldots, (N-1)$.
  - The transform coefficient $X(k)$ is proportional to the component of vector **x** in the direction of the basis vector $\mathbf{a}_k$, which corresponds to the projection **p** of **x** on the unit-norm vector $\frac{\mathbf{a}_k}{\|\mathbf{a}_k\|}$, that is, $\mathbf{p} = \left\langle \mathbf{x}, \frac{\mathbf{a}_k}{\|\mathbf{a}_k\|} \right\rangle$.

Discrete transforms and orthogonality

- If the transform is unitary, then $\gamma = 1$, which implies, from equation (195), that $\|\mathbf{a}_k\| = 1$.
- In this case, vectors $\mathbf{a}_k$ are said to form an orthonormal basis, such that equation (199) becomes

$$\mathbf{x} = \sum_{k=0}^{N-1} \langle \mathbf{x}, \mathbf{a}_k \rangle \, \mathbf{a}_k \tag{200}$$

that is, the transform coefficient $X(k)$ is equal to the projection of vector $\mathbf{x}$ on the unit-norm basis vector $\mathbf{a}_k$.

Discrete transforms and orthogonality

- The canonical basis is formed by the vectors $\mathbf{e}_k$, for $k = 0, 1, \ldots, (N-1)$, such that

$$\begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_{N-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{201}$$

- The vectors $\mathbf{e}_k$ form an orthonormal set that is an orthonormal basis of $\mathbb{C}^N$.
  - In this basis, the nth component of any vector $\mathbf{x}$ can be expressed as

$$x(n) = \mathbf{e}_n^{*^T}\mathbf{x} = \langle \mathbf{x}, \mathbf{e}_n \rangle \tag{202}$$

- From the above equation, we see that the samples of $\mathbf{x}$ are its projections (or coordinates) on the canonical basis.

Discrete transforms and orthogonality

- Since all orthonormal basis are rotations of one another, and any unitary transform is a projection on an orthonormal basis, this confirms the statement made earlier that all orthonormal transforms are just rotations in $\mathbb{C}^{\mathrm{N}}$.
- In the remainder of this section we describe some of the most commonly employed discrete transforms.

Discrete cosine transform

- The length-N discrete cosine transform (DCT) of a signal $x(n)$ can be defined as

$$C(k) = \alpha(k) \sum_{n=0}^{N-1} x(n) \cos\left[\frac{\pi(n+\frac{1}{2})k}{N}\right], \quad \text{for } 0 \leq k \leq N-1 \tag{203}$$

where

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}}, & \text{for } k = 0 \\ \sqrt{\frac{2}{N}}, & \text{for } 1 \leq k \leq N-1 \end{cases} \tag{204}$$

- Accordingly, the inverse DCT is given by

$$x(n) = \sum_{k=0}^{N-1} \alpha(k) C(k) \cos\left[\frac{\pi(n+\frac{1}{2})k}{N}\right], \quad \text{for } 0 \leq n \leq N-1 \tag{205}$$

- Note that the DCT is a real transform, that is, it maps a real signal into real DCT coefficients.

Discrete cosine transform

- From equations (203)–(205), we can define the DCT matrix $\mathbf{C}_N$ by

$$\{\mathbf{C}_N\}_{kn} = \alpha(k)\cos\left[\frac{\pi(n+\frac{1}{2})k}{N}\right] \tag{206}$$

and the matrix form of the DCT becomes

$$\mathbf{c} = \mathbf{C}_N\mathbf{x} \tag{207}$$

$$\mathbf{x} = \mathbf{C}_N^T\mathbf{c} \tag{208}$$

- Note that for the above equations to be valid, $\mathbf{C}_N^{-1} = \mathbf{C}_N^T$, which, together with the fact that $\mathbf{C}_N$ is a real matrix, implies that the matrix $\mathbf{C}_N$ is unitary.
- As seen earlier, the Parseval theorem is valid, and the DCT can be considered a rotation in $\mathbb{C}^N$ (and also in $\mathbb{R}^N$, since it is a real transform).

Discrete cosine transform

- The DCT basis functions $\mathbf{c}_k$ are the conjugate transpose of the rows of $\mathbf{C}_N$ (see equation (191)), and are thus given by

$$\begin{aligned} \mathbf{c}_k(n) &= \{\mathbf{C}_N\}^*_{kn} \\ &= \alpha(k)\cos\left[\frac{\pi(n+\frac{1}{2})k}{N}\right] \\ &= \alpha(k)\cos\left(\frac{2\pi}{2N}kn + \frac{\pi}{2N}k\right) \end{aligned} \tag{209}$$

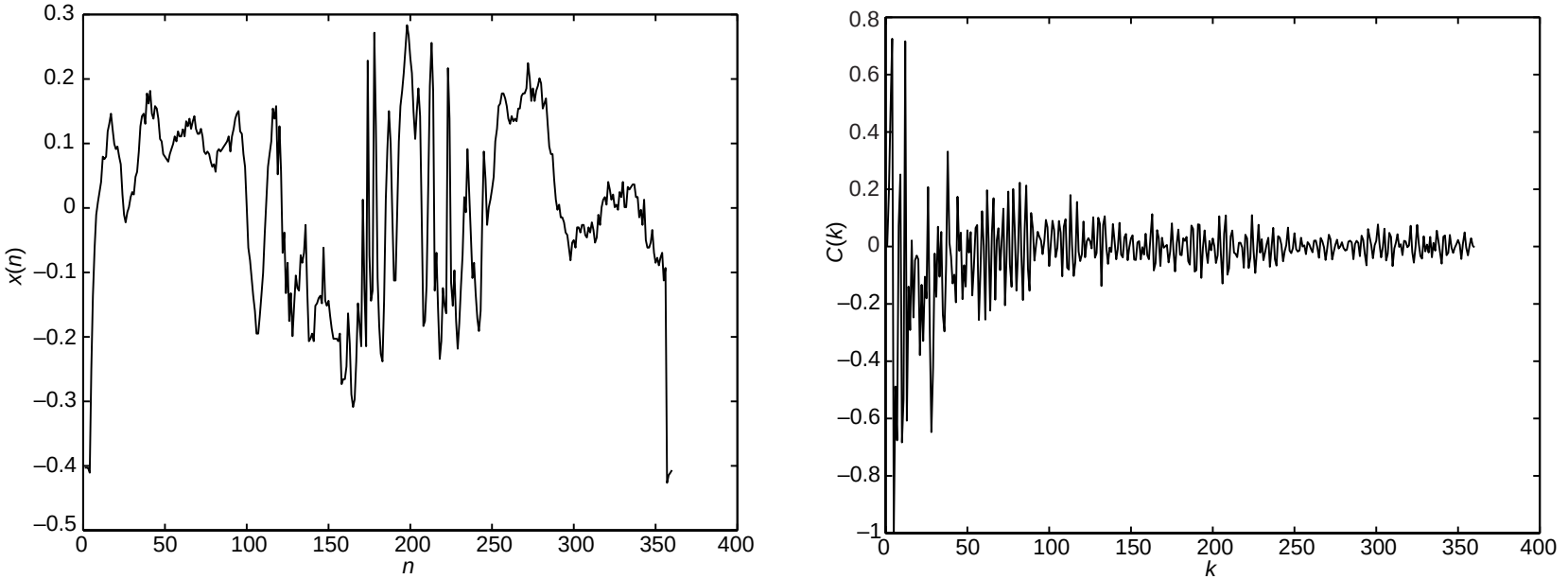
  that are sinusoids of frequencies $\omega_k = \frac{2\pi}{2N}kn$, for $k = 0, 1, \ldots, (N-1)$.

  – Thus, the DCT, besides being a rotation in $\mathbb{R}^N$, decomposes a signal as a sum of N real sinusoids of frequencies given by ω_k above.

- The DCT enjoys another very important property: When applied to signals such as voice and video, most of the transform energy is concentrated in few coefficients.

Discrete cosine transform

- For example, in Figure 18a, we can see a digital signal $x(n)$ corresponding to one line of a digitized television signal, and in Figure 18b, we show its DCT coefficients $C(k)$.
- It can be seen that the energy of the signal more or less evenly spread among its samples, while it is mostly concentrated in the first transform coefficients.
- Due to this property, the DCT is widely used in video compression schemes, because the coefficients with lowest energy can be discarded during transmission without introducing significant distortion in the original signal.
- In fact, the DCT is part of most of the digital video broadcasting systems in operation in the world.



(a) (b)

Figure 18: The DCT of a video signal: (a) discrete-time video signal $x(n)$; (b) DCT of $x(n)$.

Discrete cosine transform

- Since the DCT is based on sinusoids and $\cos(x) = \frac{1}{2}(e^{jx} + e^{-jx})$, then in the worst case the DCT of $x(n)$ can be computed using one length-$2N$ DFT.
- This can be deduced by observing that

$$\cos\left[\frac{\pi(n+\frac{1}{2})k}{N}\right] = \cos\left(\frac{2\pi}{2N}kn + \frac{\pi}{2N}k\right)$$

$$= \frac{1}{2}\left[e^{j\left(\frac{2\pi}{2N}kn+\frac{\pi}{2N}k\right)} + e^{-j\left(\frac{2\pi}{2N}kn+\frac{\pi}{2N}k\right)}\right]$$

$$= \frac{1}{2}\left(W_{2N}^{\frac{k}{2}}W_{2N}^{kn} + W_{2N}^{-\frac{k}{2}}W_{2N}^{-kn}\right) \quad (210)$$

which implies that

$$C(k) = \alpha(k) \sum_{n=0}^{N-1} x(n) \cos\left[\frac{\pi(n+\frac{1}{2})k}{N}\right]$$

$$= \frac{1}{2}\left(\alpha(k) W_{2N}^{\frac{k}{2}} \sum_{n=0}^{N-1} x(n) W_{2N}^{kn} + \alpha(k) W_{2N}^{-\frac{k}{2}} \sum_{n=0}^{N-1} x(n) W_{2N}^{-kn}\right)$$

$$= \frac{1}{2}\alpha(k)\left(W_{2N}^{\frac{k}{2}} \text{DFT}_{2N}\left\{\hat{x}(n)\right\}(k) + W_{2N}^{-\frac{k}{2}} \text{DFT}_{2N}\left\{\hat{x}(n)\right\}(-k)\right) \quad (211)$$

for $0 \leq k \leq N-1$, where $\hat{x}(n)$ is equal to $x(n)$ zero-padded to length $2N$.

- Note that the second term in the above equation is equivalent to the first one computed at the index $-k$.
- Therefore, we actually need to compute only one length-$2N$ DFT.

- One can see that the algorithm in equation (211) has a complexity of one length-$2N$ DFT, plus the $2N$ multiplications by W_{2N}^{k}, which gives a total of $(2N + 2N \log_2 2N)$ complex multiplications.

Discrete cosine transform

- However, there are other fast algorithms for the DCT which give a complexity of the order of $N \log_2 N$ real multiplications.
- The graph of one popular fast DCT algorithm for $N = 8$ is shown in Figure 19.

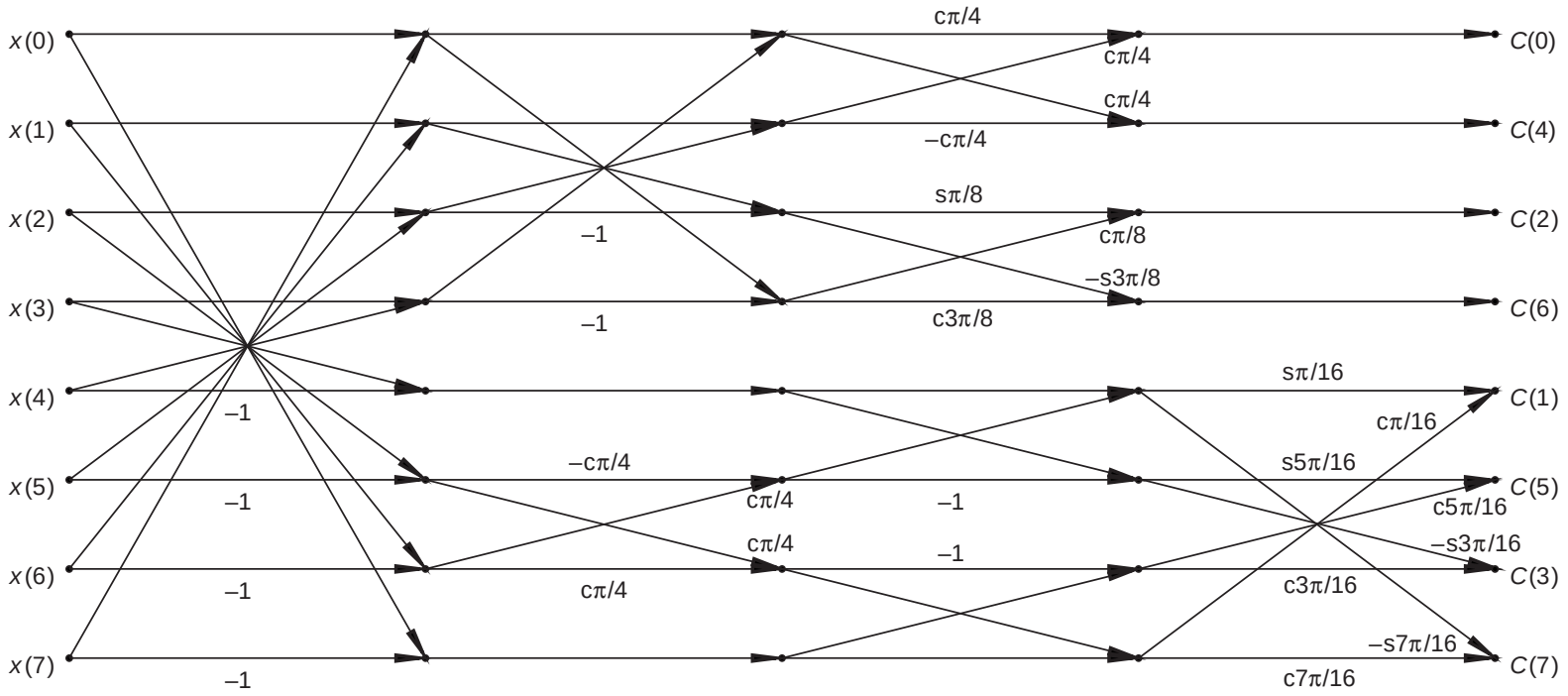


Figure 19: A fast algorithm for the computation of a length-8 DCT. In this graph, cx corresponds to $\cos x$ and sx corresponds to $\sin x$.

Discrete cosine transform

- This graph corresponds to the following factorization of $\mathbf{C}_8$:

$$\mathbf{C}_8 = \mathbf{P}_8\mathbf{A}_4\mathbf{A}_3\mathbf{A}_2\mathbf{A}_1 \tag{212}$$

where

Discrete cosine transform

$$\mathbf{A}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{213}$$

Discrete cosine transform

$$\mathbf{A}_2 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\cos\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & \cos\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (214)$$

Discrete cosine transform

$$\mathbf{A}_3 = \begin{bmatrix} \cos\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 & 1 & 0 & 0 & 0 & 0 \\ \cos\frac{\pi}{4} & -\cos\frac{\pi}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sin\frac{\pi}{8} & \cos\frac{\pi}{8} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\sin\frac{3\pi}{8} & \cos\frac{3\pi}{8} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad (215)$$

Discrete cosine transform

$$\mathbf{A}_4 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sin\frac{\pi}{16} & 0 & 0 & \cos\frac{\pi}{16} \\ 0 & 0 & 0 & 0 & 0 & \sin\frac{5\pi}{16} & \cos\frac{5\pi}{16} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & 0 \\ 0 & 0 & 0 & 0 & -\sin\frac{7\pi}{16} & 0 & 0 & \cos\frac{7\pi}{16} \end{bmatrix} \quad (216)$$

Discrete cosine transform

and $\mathbf{P}_8$ is given by equation (151), corresponding to the placement in normal ordering of the bit-reversed indexes of the output vector **c**.

- Note that the operation corresponding to $\mathbf{P}_8$ is not shown in Figure 19.
- After discounting the trivial multiplications, we have that, for a generic $\mathrm{N} = 2^{\mathrm{l}}$, the numbers of real multiplications $\mathcal{M}(\mathrm{N})$ and real additions $\mathcal{A}(\mathrm{N})$ in this fast implementation of the DCT are

$$\mathcal{M}(\mathrm{N}) = \mathrm{N}\log_2 \mathrm{N} - \frac{3\mathrm{N}}{2} + 4 \tag{217}$$

$$\mathcal{A}(\mathrm{N}) = \frac{3\mathrm{N}}{2}(\log_2 \mathrm{N} - 1) + 2 \tag{218}$$

A family of sine and cosine transforms

- The DCT is a particular case of a more general class of transforms, whose transform matrix has its rows composed of sines or cosines of increasing frequency.
- Such transforms have a number of applications including the design of filter banks (Chapter 9).
- In what follows, we present a brief description of the so-called even forms of the cosine and sine transforms.
- There are four types of even cosine transforms and four types of even sine transforms.

A family of sine and cosine transforms

- Their transform matrices are referred to as $\mathbf{C}_N^{\text{I–IV}}$ for the cosine transforms and $\mathbf{S}_N^{\text{I–IV}}$ for the sine transforms. Their (k, n) elements are given by

$$\{\mathbf{C}_N^x\}_{kn} = \sqrt{\frac{2}{N+\epsilon_1}}\,(\alpha_{N+\epsilon_1}(k))^{\epsilon_2}\,(\alpha_{N+\epsilon_1}(n))^{\epsilon_3}\cos\left[\frac{\pi(k+\epsilon_4)(n+\epsilon_5)}{N+\epsilon_1}\right] \tag{219}$$

$$\{\mathbf{S}_N^x\}_{kn} = \sqrt{\frac{2}{N+\epsilon_1}}\,(\alpha_{N+\epsilon_1}(k))^{\epsilon_2}\,(\alpha_{N+\epsilon_1}(n))^{\epsilon_3}\sin\left[\frac{\pi(k+\epsilon_4)(n+\epsilon_5)}{N+\epsilon_1}\right] \tag{220}$$

  where

$$\alpha_\gamma(k) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{for } k = 0 \text{ or } k = \gamma \\ 1, & \text{for } 1 \le k \le \gamma - 1 \end{cases} \tag{221}$$

- The set $(\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5)$ defines the transform.

A family of sine and cosine transforms

- Table 1 gives those values for all even cosine and sine transforms.
- One should note that all these transforms have fast algorithms and are unitary, that is $\mathbf{A}^{-1} = \mathbf{A}^{*^{\mathsf{T}}}$.
- For instance, the DCT defined in Subsection 182 is the same as $\mathbf{C}_{\mathrm{N}}^{\mathrm{II}}$.

A family of sine and cosine transforms

Table 1: Definition of the even cosine and sine transforms.

	ϵ_1	ϵ_2	ϵ_3	ϵ_4	ϵ_5
$\mathbf{C}^{\mathrm{I}}_{kn}$	-1	1	1	0	0
$\mathbf{C}^{\mathrm{II}}_{kn}$	0	1	0	0	$\frac{1}{2}$
$\mathbf{C}^{\mathrm{III}}_{kn}$	0	0	1	$\frac{1}{2}$	0
$\mathbf{C}^{\mathrm{IV}}_{kn}$	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$
$\mathbf{S}^{\mathrm{I}}_{kn}$	1	0	0	0	0
$\mathbf{S}^{\mathrm{II}}_{kn}$	0	1	0	0	$-\frac{1}{2}$
$\mathbf{S}^{\mathrm{III}}_{kn}$	0	0	1	$-\frac{1}{2}$	0
$\mathbf{S}^{\mathrm{IV}}_{kn}$	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$

Discrete Hartley transform

- The discrete Hartley transform (DHT) can be viewed as a real counterpart of the DFT when it is applied to real signals.
- The definition of a length-N direct DHT is

$$H(k) = \sum_{n=0}^{N-1} x(n) \operatorname{cas}\left(\frac{2\pi}{N} kn\right), \quad \text{for } 0 \leq k \leq N-1 \tag{222}$$

  where $\operatorname{cas} x = \cos x + \sin x$.

- Similarly, the inverse DHT is determined by

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} H(k) \operatorname{cas}\left(\frac{2\pi}{N} kn\right), \quad \text{for } 0 \leq k \leq N-1 \tag{223}$$

Discrete Hartley transform

- The Hartley transform is attractive due to the following properties:
  - If the input signal is real, the DHT is real.
  - The DFT can be easily derived from the DHT and vice versa.
  - It has efficient fast algorithms.
  - It has a convolution-multiplication property.
- The first property follows trivially from the definition of the DHT.

Discrete Hartley transform

- The other properties can be derived from the relations between the DFT $X(k)$ and the DHT $H(k)$ of a real sequence $x(n)$, which are

$$H(k) = \text{Re}\{X(k)\} - \text{Im}\{X(k)\} \tag{224}$$

$$X(k) = \mathcal{E}\{H(k)\} - j\mathcal{O}\{H(k)\} \tag{225}$$

where the operators $\mathcal{E}\{\cdot\}$ and $\mathcal{O}\{\cdot\}$ correspond to the even and odd parts, respectively, that is

$$\mathcal{E}\{H(k)\} = \frac{H(k) + H(-k)}{2} \tag{226}$$

$$\mathcal{O}\{H(k)\} = \frac{H(k) - H(-k)}{2} \tag{227}$$

Discrete Hartley transform

- The convolution property (the fourth property above) can be stated more precisely as, given two arbitrary length-N sequences $x_1(n)$ and $x_2(n)$, the DHT of their length-N circular convolution $y(n)$ is given by (from equations (224) and (62))

$$Y(k) = H_1(k)\mathcal{E}\{H_2(k)\} + H_1(-k)\mathcal{O}\{H_2(k)\} \tag{228}$$

  where $H_1(k)$ is the DHT of $x_1(n)$ and $H_2(k)$ is the DHT of $x_2(n)$.

- This result is especially useful when the sequence $x_2(n)$ is even, which leads to

$$Y(k) = H_1(k)H_2(k) \tag{229}$$

- This equation only involves real functions while equation (62) relates complex functions to determine $Y(k)$.

- Therefore, in the case $x_2(n)$ is even, it is advantageous to use equation (229) instead of equation (62) for performing convolutions.

Discrete Hartley transform

- To conclude, we can say that the DHT, as a signal representation, is not as widely used in practice as the DFT or the DCT.
- However, it has been used in a large range of applications as a tool to perform fast convolutions, as well as a tool to compute FFTs and fast DCTs.

Hadamard transform

- The Hadamard transform, also known as the Walsh-Hadamard transform, is a transform that is not based on sinusoidal functions, unlike the other transforms seen so far.
- Instead, the elements of its transform matrix are either 1 or -1. When $N = 2^n$, its transform matrix is defined by the following recursion:

$$\mathbf{H}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\mathbf{H}_n = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{H}_{n-1} & \mathbf{H}_{n-1} \\ \mathbf{H}_{n-1} & -\mathbf{H}_{n-1} \end{bmatrix} \tag{230}$$

- From the above equations, it is easy to see that the Hadamard transform is unitary.

Hadamard transform

- For example, a length-8 Hadamard transform matrix is

$$\mathbf{H}_3 = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \tag{231}$$

- An important aspect of the Hadamard transform is that, since the elements of its transform matrix are only either 1 or -1, then the Hadamard transform needs no multiplications for its computation, leading to simple hardware implementations.

Hadamard transform

- Because of this fact, the Hadamard transform has been used in digital video schemes in the past, although its compression performance is not as high as that of the DCT.
- Nowadays, with the advent of specialized hardware to compute the DCT, the Hadamard transform is used in digital video only in specific cases.
- However, one important area of application of the Hadamard transform today is in carrier division multiple access (CDMA) systems for mobile communications, where it is employed as channelization code in synchronous communications systems.

Hadamard transform

- The Hadamard transform also has a fast algorithm. For example, the matrix $\mathbf{H}_3$ can be factored as:

$$\mathbf{H}_3 = \mathbf{A}_8^3 \tag{232}$$

where

$$\mathbf{A}_8 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \tag{233}$$

- Therefore, the number of additions of a fast Hadamard transform algorithm is of the order of $\mathrm{N} \log_2 \mathrm{N}$.

Other important transforms

Wavelet transforms

- Wavelet transforms constitute a different class of transforms which have become very popular in the past years.
- They come from the area of functional analysis, and in digital signal processing are usually studied under the discipline of multirate signal processing.
- Wavelet transforms are studied in Chapter 10 of this book.

Other important transforms

Karhunen-Loève transform

- As seen earlier, the DCT is widely used in image and video compression because it has the ability to concentrate the energy of a signal in a few transform coefficients.
- A question that naturally arises is which is the transform that maximizes this energy concentration.
- Given a statistical distribution of an ensemble of signals, the optimum transform in terms of this energy compaction capability is the Karhunen-Loève transform (KLT).
- It is defined as the transform that diagonalizes the autocorrelation matrix of a discrete random process.
- From the above, we see that there is a different KLT for each of the different signal statistics.

Other important transforms

- However, it can be shown that the DCT approximates the KLT when the signals can be modeled as Gauss-Markov processes with correlation coefficients near to 1.
- This is a reasonably good model for several useful signals; probably the best example of it is given by video signals.
- For them, the DCT is indeed approximately optimum in terms of energy compaction capability, which explains its widespread use.

Signal representations

- In Chapters 1–3, we have dealt with several forms of signal representations, both continuous and discrete.
- We now summarize the main characteristics of those representations.
- We classify them in terms of both the time and frequency variables as continuous or discrete, as well as real, imaginary, or complex.
- Also, we classify the representations as being periodic or nonperiodic.
- The time-frequency relationship for each transform is shown in Figures 20–25.

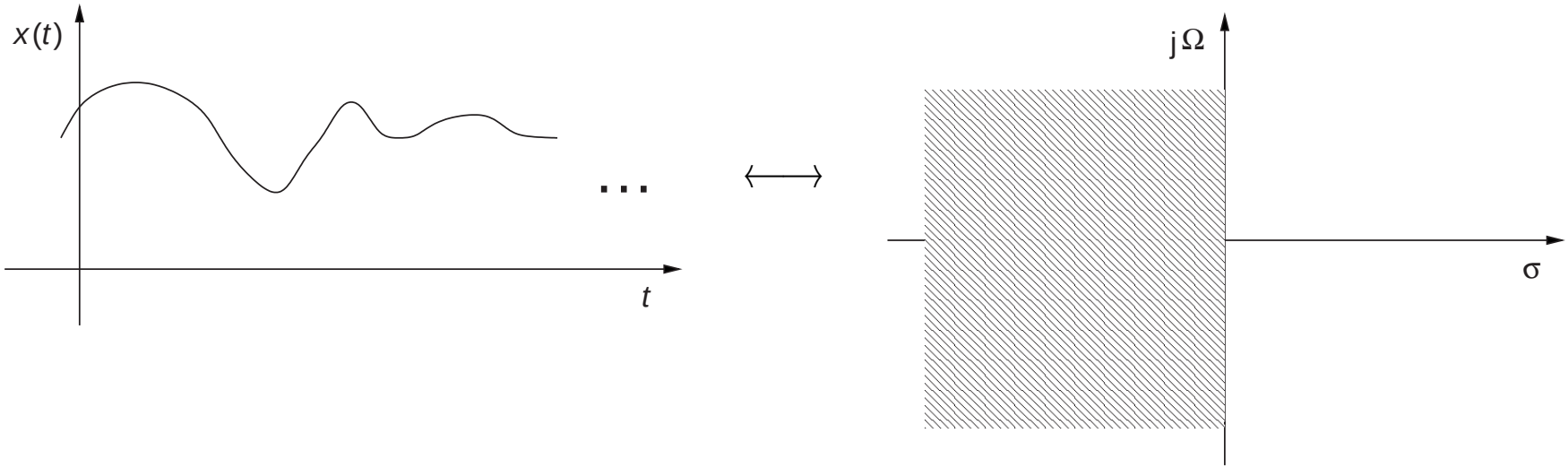
Signal representations

Laplace transform

$$X(s) = \int_{-\infty}^{\infty} x(t) e^{-st} dt \quad \longleftrightarrow \quad x(t) = \frac{1}{2\pi} e^{\sigma t} \int_{-\infty}^{\infty} X(\sigma + j\omega) e^{j\omega t} d\omega \tag{234}$$

- Time domain: nonperiodic function of a continuous and real-time variable.
- Frequency domain: nonperiodic function of a continuous and complex frequency variable.

Signal representations



(a) (b)

Figure 20: Laplace transform: (a) continuous-time signal; (b) domain of the corresponding Laplace transform.

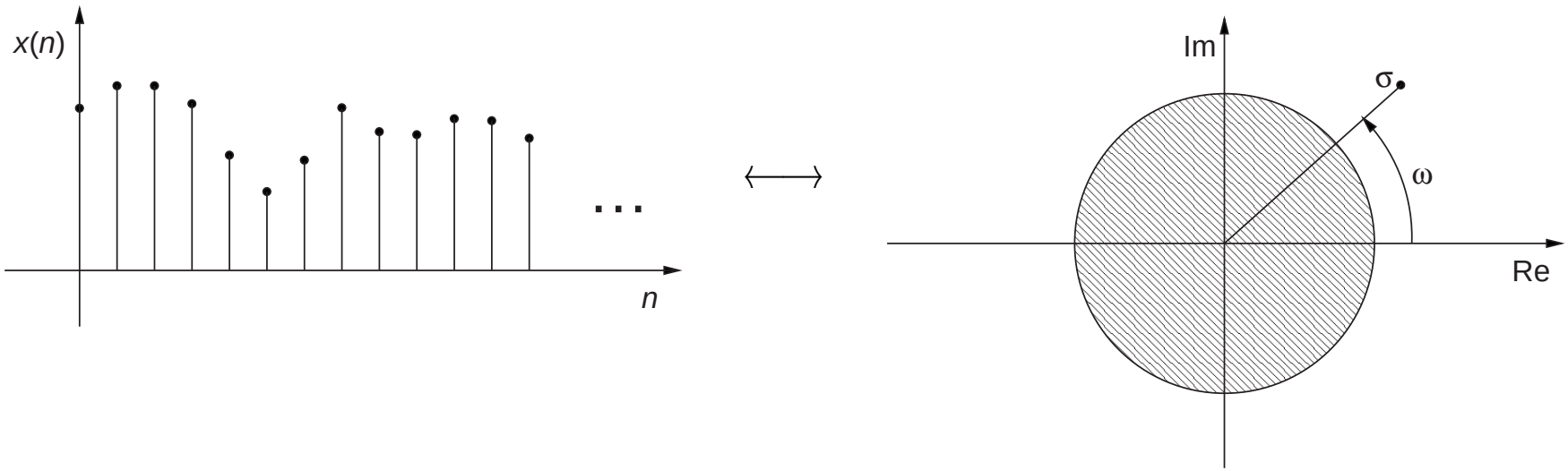
Signal representations

z transform

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n} \quad \longleftrightarrow \quad x(n) = \frac{1}{2\pi j} \oint_C X(z) z^{n-1} dz \tag{235}$$

- Time domain: nonperiodic function of a discrete and integer time variable.
- Frequency domain: nonperiodic function of a continuous and complex frequency variable.

Signal representations



(a) (b)

Figure 21: z transform: (a) discrete-time signal; (b) domain of the corresponding z transform.

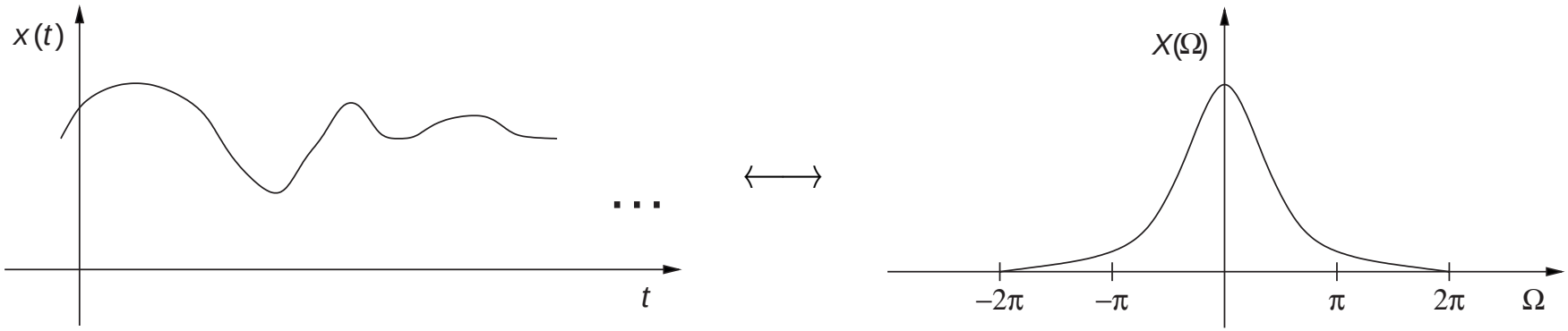
Signal representations

Fourier transform (continuous time)

$$X(\Omega) = \int_{-\infty}^{\infty} x(t) e^{-j\Omega t} dt \quad \longleftrightarrow \quad x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\Omega) e^{j\Omega t} d\Omega \qquad (236)$$

- Time domain: nonperiodic function of a continuous and real-time variable.
- Frequency domain: nonperiodic function of a continuous and imaginary frequency variable.

Signal representations



(a) (b)

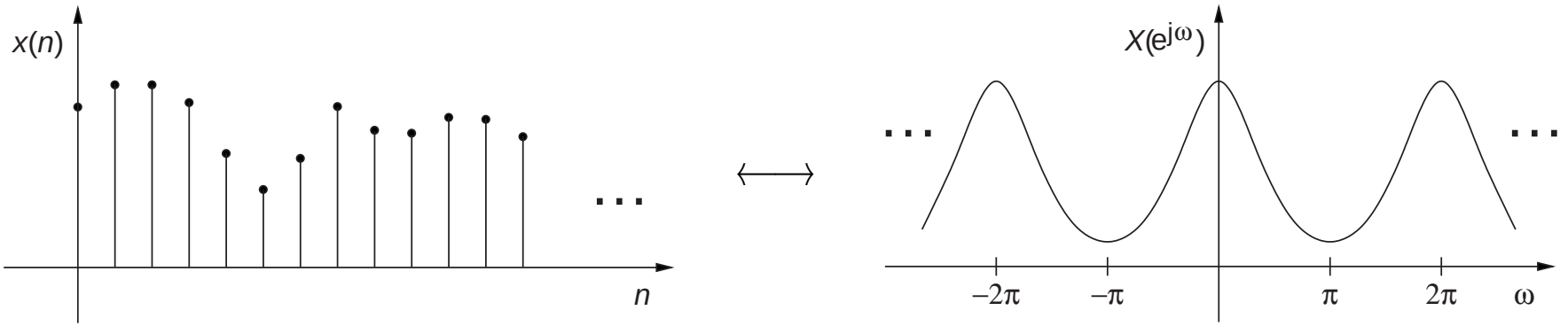
Figure 22: Fourier transform: (a) continuous-time signal; (b) corresponding Fourier transform.

Signal representations

Fourier transform (discrete time)

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \quad \longleftrightarrow \quad x(n) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(e^{j\omega})e^{j\omega n}d\omega \qquad (237)$$

- Time domain: nonperiodic function of a discrete and integer time variable.
- Frequency domain: periodic function of a continuous frequency variable.



(a) (b)

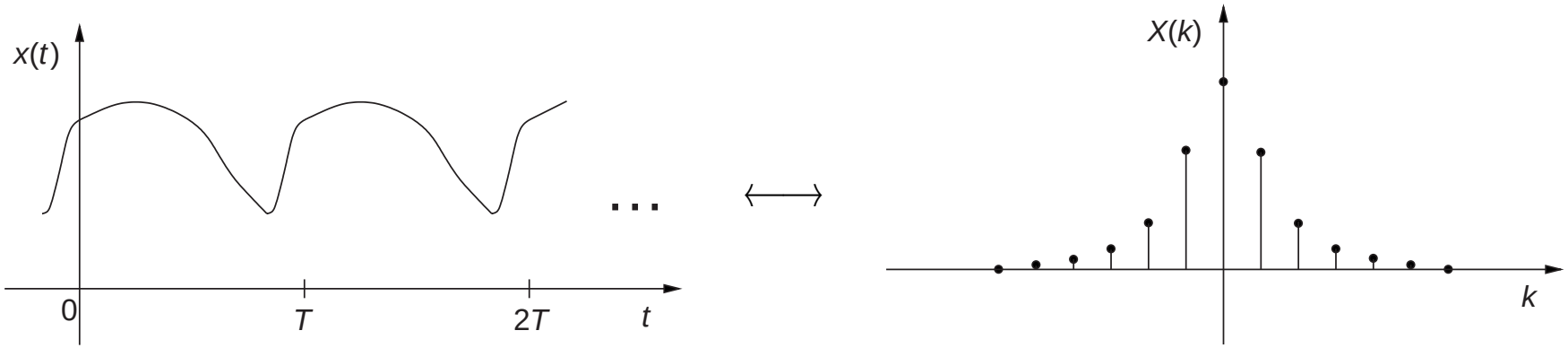
Figure 23: Discrete-time Fourier transform: (a) discrete-time signal; (b) corresponding discrete-time Fourier transform.

Signal representations

Fourier series

$$X(k) = \frac{1}{T}\int_0^T x(t)e^{-j\frac{2\pi}{T}kt}dt \quad \longleftrightarrow \quad x(t) = \sum_{k=-\infty}^{\infty} X(k)e^{j\frac{2\pi}{T}kt} \tag{238}$$

- Time domain: periodic function of a continuous and real-time variable.
- Frequency domain: nonperiodic function of a discrete and integer frequency variable.



(a) (b)

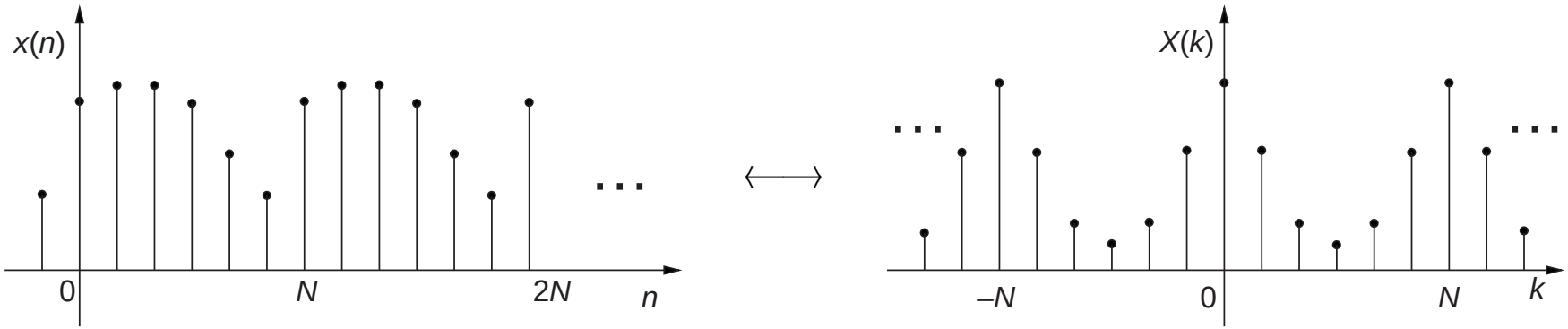
Figure 24: Fourier series: (a) continuous-time periodic signal; (b) corresponding Fourier series.

Signal representations

Discrete Fourier transform

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn} \quad \longleftrightarrow \quad x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi}{N}kn} \tag{239}$$

- Time domain: periodic function of a discrete and integer time variable.
- Frequency domain: periodic function of a discrete and integer frequency variable.



(a) (b)

Figure 25: Discrete Fourier transform: (a) discrete-time signal; (b) corresponding discrete Fourier transform.

Do-it-yourself: Discrete transforms

Experiment 3.1:

- We have seen that the output signal $y(n)$ of a linear, time-invariant, causal filter can be determined by the linear convolution between the input signal $x(n)$ and the system impulse response $h(n)$.
- In this experiment, we investigate several ways to perform such an operation using MATLAB.
- We work here with short-length sequences $x(n)$ and $h(n)$, such as:

```
x = ones(1,10);
h = [1 2 3 4 5 6 7 8 9 10];
```

  so that the reader can obtain the desired output algebraically in advance.
- Later, we compare the performance of each method seen below for longer sequences.

Do-it-yourself: Discrete transforms

- Given $x(n)$ and $h(n)$, perhaps the easiest way, but not necessarily the most numerically efficient one, is to employ the command `conv`:

  ```
  y1 = conv(x,h);
  ```

  whose input arguments can be swapped since the convolution operation is symmetric.

- For a general digital filter with transfer function

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{a_0 + a_1 z^{-1} + \cdots + a_N z^{-N}}, \tag{240}$$

  with $a_0 \neq 0$, one can determine the output $y(n)$ to the input $x(n)$ using the command `filter`.

- For this command, the input arguments are two vectors, containing the numerator and denominator coefficients of $H(z)$, and the input signal.

Do-it-yourself: Discrete transforms

- In this experiment, where the impulse response $h(n)$ is given, we may use the command filter assuming that $A(z) = 1$ and associating $h(n)$ to the numerator-coefficient vector, such as:

```
y2 = filter(h,1,x);
```

- It is interesting to notice that the `filter` command forces the length of the output vector to be the same as of the input.

- Therefore, if one wants to determine all non-zero samples of $y(n)$, we must force $x(n)$ to have the desired output length by padding the original input with the proper number of zeros:

```
xaux = [x zeros(1,length(h)-1)];
y3 = filter(h,1,xaux);
```

Do-it-yourself: Discrete transforms

- As mentioned earlier, one can implement the digital filtering operation via the frequency domain using the FFT.
- To avoid the circular convolution, one must first pad the $\mathtt{x}(\mathtt{n})$ and $\mathtt{h}(\mathtt{n})$ vectors with the proper numbers of zeros.
- The easiest way to determine these number of zeros is to remember that the length of the desired output signal should be the length of the linear convolution of $\mathtt{x}(\mathtt{n})$ and $\mathtt{h}(\mathtt{n})$, that is

$$\mathtt{length}(\mathtt{y}) = \mathtt{length}(\mathtt{x}) + \mathtt{length}(\mathtt{h}) - 1. \tag{241}$$

Do-it-yourself: Discrete transforms

- Hence, we must guarantee that the FFTs of $x(n)$ and $h(n)$ are determined with this length, as performed by the following script:

```
length_y = length(x) + length(h) - 1;
X = fft(x,length_y);
H = fft(h,length_y);
Y4 = X.*H;
y4 = ifft(Y4);
```

  - In old MATLAB versions, numerical errors tended to accumulate throughout the filtering process, generating a complex output sequence, even when $x(n)$ and $h(n)$ were real signals.
  - In these cases, one was forced to use the `real` command to store only the real part of the result. Current versions of MATLAB get rid of the spurious imaginary part of `y4` automatically.

Do-it-yourself: Discrete transforms

- As mentioned before, this whole experiment was based on short $x(n)$ and $h(n)$ signals to allow the reader to follow closely all computations performed in MATLAB.
- In practice, if the lengths of these signals are sufficiently short (both of them around 100 coefficients), the simplest and fastest way to perform the digital filtering is with the `conv` command.
- If, however, both signals become too lengthy, the frequency domain becomes quite advantageous in terms of numerical computations.
- In the case where only one of the signals has a long duration, the overlap-and-add method described earlier can be implemented as:

  ```
  y5 = fftfilt(h,xaux);
  ```

  where the FFT size and the segmentation of $x(n)$ are automatically chosen to guarantee efficient execution.

Do-it-yourself: Discrete transforms

- The reader is encouraged to experiment all forms above to perform digital filtering with distinct input signals and impulse responses.
  - Particularly, he/she may increase the length of these signals, to the order of thousands of samples, in order to verify how the frequency domain becomes an important tool in several practical situations.

Do-it-yourself: Discrete transforms

Experiment 3.2:

- Let us now employ the frequency domain to analyze the contents of a given signal $x(n)$ composed by a 10-Hz sinusoid corrupted by noise, with $F_s = 200$ samples/s for an interval of 1 second, as given by:

```
fs = 200; f = 10;
time = 0:1/fs:(1-1/fs);
k = 0;
x = sin(2*pi*f.*time) + k*randn(1,fs);
figure(1);
plot(time,x);
```

  where the parameter k controls the amount of noise present in $x(n)$.

Do-it-yourself: Discrete transforms

- Figure 26 shows examples of $x(n)$ and Figure 27 depicts the absolute value of the corresponding FFT for distinct values of k.
- Figure 26 indicates that the sinusoidal component is clearly observed in the time domain for small amounts of noise, such as when $k \leq 0.5$.
  - For larger amounts of noise, as when $k = 1.5$, the frequency domain can then be employed to detect the sinusoidal component, as well as to estimate its frequency value, from the position of the dominating peaks in Figure 27.
  - However, when k is too large, the sinusoid becomes masked by the noisy component even in the frequency domain, as observed in Figure 27d.

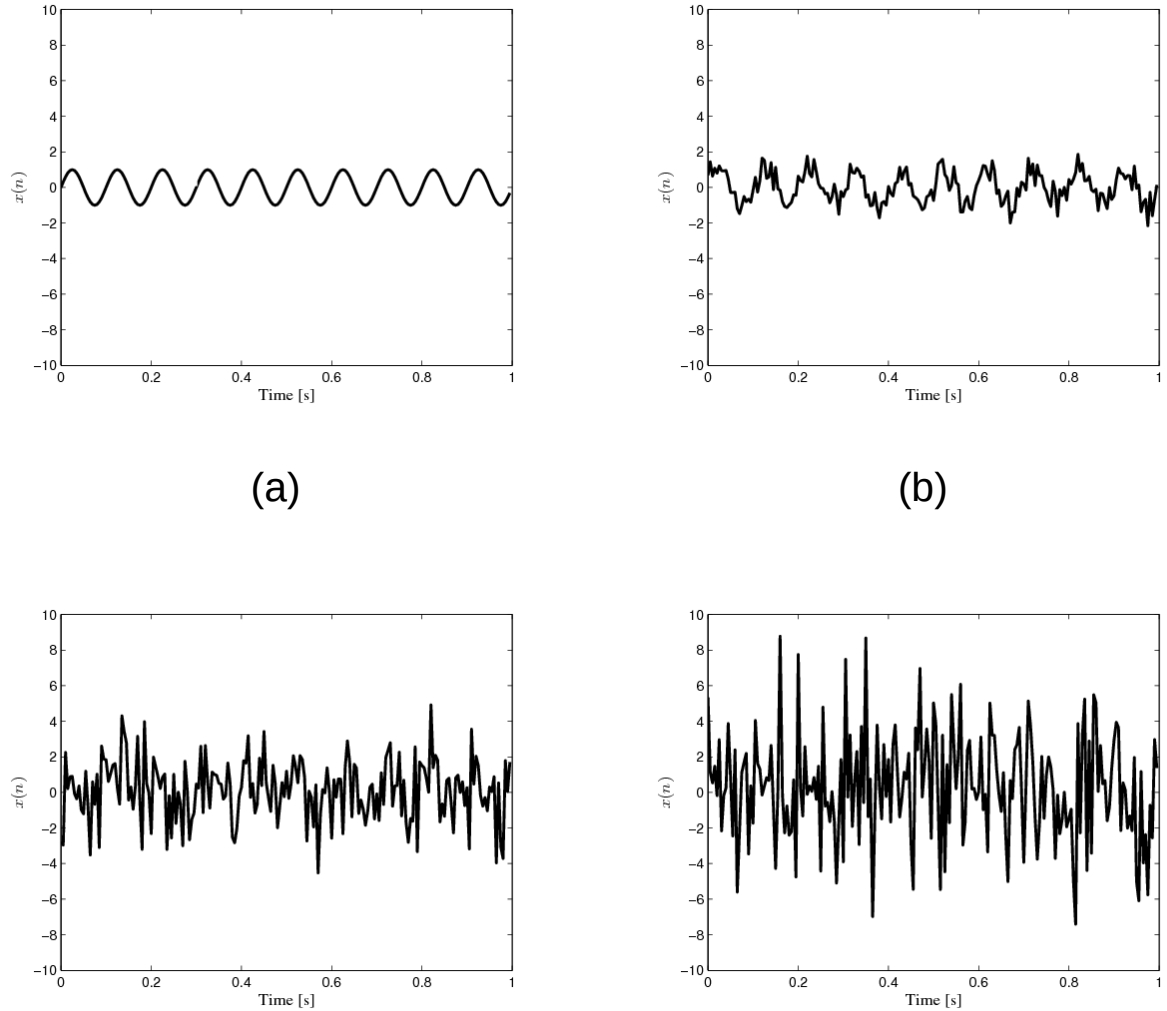


Figure 26: Sinusoidal signal corrupted with different levels of noise: (a) $k = 0$; (b) $k = 0.5$; (c) $k = 1.5$; (d) $k = 3$.

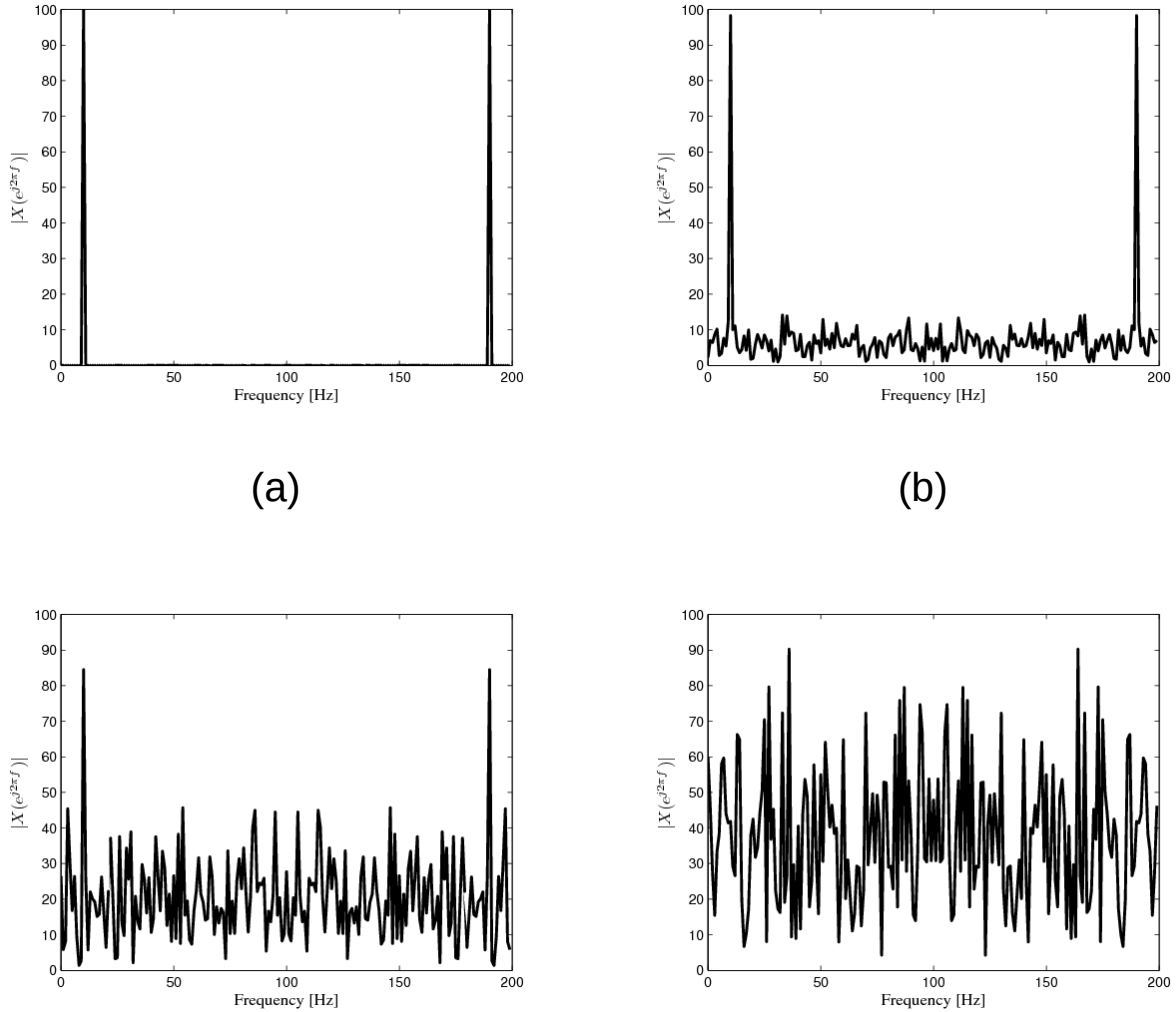


Figure 27: Absolute value of FFT of sinusoidal signal corrupted with different levels of noise: (a) $k = 0$; (b) $k = 0.5$; (c) $k = 1.5$; (d) $k = 3$.

Do-it-yourself: Discrete transforms

- One way to deal with the large-noise case is to estimate the spectrum for several different time segments of $x(n)$ and average the results.
- By doing so, the 10-Hz sinusoidal peaks are present in all FFTs whereas the noise peaks are randomly located in the different FFTs.
- Therefore, the averaging operation tends to preserve the FFT peaks corresponding to the 10-Hz sinusoid while attenuating the peaks due to the noise component.
- This approach is left as an exercise for the reader.