06 The Discrete Fourier Transform

06.01 Definition of the Discrete Fourier Transform (DFT)

We have discussed the discrete-time Fourier transform, which enables the transition from the time domain to the frequency domain. The primary drawback of the DTFT is that its inverse transform necessitates the evaluation of an integral, making it unsuitable for computer implementation.
In contrast, the direct discrete-time Fourier transform does not require integral evaluation but involves only the computation of multiplications and additions. This is because the DTFT is a periodic function in ω and can be expanded in the Fourier series as the sum of an infinite number of generalized sinusoids, represented by $e^{-j\omega n}$.
Now, we will consider another transform applicable to periodic sequences (and later, as we will see, to finite-duration sequences) – the Discrete Fourier Transform (DFT). Due to the periodicity of the sequences being transformed, both the direct DFT transform and the inverse DFT (IDFT) transform only require the sum of complex numbers and do not involve any integration. Therefore, they are suitable for computer implementation.

Considering a periodic sequence $x_p(n)$ with a period of N (where $x_p(n) = x_p(n+N)$ for all n), we define the Discrete Fourier Transform (DFT) of $x_p(n)$ as follows

$$X_p(k) = \sum_{n=0}^{N-1} x_p(n) e^{-j\frac{2\pi}{N}nk}.$$

It is easy to verify that $X_P(k)$ is a complex periodic sequence with a period of N:

$$X_p(k) = X_p(k+N) \qquad \forall k.$$

The Inverse Discrete Fourier Transform (IDFT) is expressed as:

$$x_p(n) = \frac{1}{N} \sum_{k=0}^{N-1} X_p(k) e^{j\frac{2\pi}{N}nk}.$$

Let's demonstrate that these relations can be derived from one another, indicating that they are reciprocal transforms.
We take the expression of the IDFT and multiply it by $e^{-j\frac{2\pi}{N}nl}$:

$$x_p(n)\, e^{-j\frac{2\pi}{N}nl} = \frac{1}{N} \sum_{k=0}^{N-1} X_p(k) e^{j\frac{2\pi}{N}nk}\, e^{-j\frac{2\pi}{N}nl}$$

$$\sum_{n=0}^{N-1} x_p(n) e^{-j\frac{2\pi}{N}nl} = \sum_{n=0}^{N-1} \frac{1}{N} \sum_{k=0}^{N-1} X_p(k) e^{j\frac{2\pi}{N}nk}\, e^{-j\frac{2\pi}{N}nl} =$$

$$= \sum_{k=0}^{N-1} X_p(k) \frac{1}{N} \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}n(k-l)}$$

For $k = l$:

$$\frac{1}{N}\sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}n(k-l)} = \frac{1}{N}\sum_{n=0}^{N-1} 1 = 1$$

For $k \neq l$:

$$\frac{1}{N}\sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}n(k-l)} = \frac{1}{N}\frac{1-e^{j\frac{2\pi}{N}N(k-l)}}{1-e^{j\frac{2\pi}{N}(k-l)}} = 0.$$

Thus, for all k and l (with $0 \leq k \leq N-1$ and $0 \leq l \leq N-1$):

$$\frac{1}{N}\sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}n(k-l)} = \delta(k-l).$$

Substituting this expression into the preceding equation:

$$\sum_{n=0}^{N-1} x_p(n)e^{-j\frac{2\pi}{N}nl} = \sum_{k=0}^{N-1} X_p(k)\frac{1}{N}\sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}n(k-l)} = \sum_{k=0}^{N-1} X_p(k)\delta(k-l) = X_p(l)$$

Q.E.D.

Hence, the two transformations are reciprocal. Using the DFT, we can pass from the time domain to the frequency domain, achieving a completely equivalent representation of our sequence.

Considering only periodic sequences may seem like a severe restriction. However, in reality, the Discrete Fourier Transform (DFT) is commonly applied to finite-length sequences. Any finite-length sequence of length N can be transformed into a periodic sequence with a period of N by periodically repeating its samples. For any causal finite-length sequence $x(n)$ satisfying

$$x(n) = 0 \qquad \forall n < 0 \text{ and } n \geq N$$

we can associate the periodic sequence:

$$x_p(n) = x(\langle n \rangle_N)$$

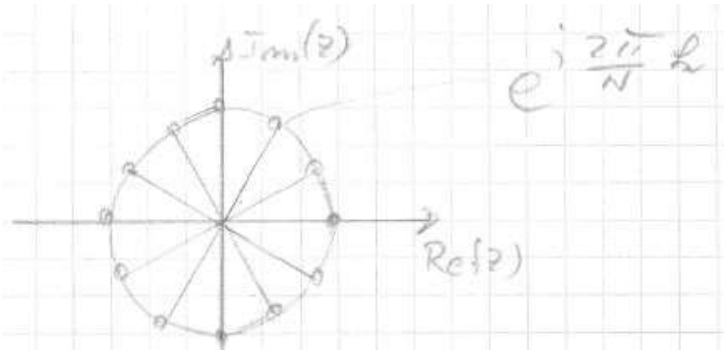
where $\langle n \rangle_N$ indicates the n modulus N (i.e., the remainder of the division of n by N). There exists a bijective correspondence between finite-length sequences of length N and periodic sequences with a period of N. Consequently, we can also apply the DFT to finite-length sequences.

06.02 The relation between DFT, DTFT, and Z Transform

A finite-length sequence of length N has Z-transform given by

$$X(z) = \sum_{n=0}^{N-1} x(n)z^{-n}.$$

Let's compute this transform at N uniformly spaced points on the unit circle:



$$X(z)\Big|_{z=e^{j\frac{2\pi}{N}k}} = \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}nk} = X_p(k)$$

Thus, we find the DFT of $x(n)$.
For finite-length sequences, the DFT can be obtained by evaluating the Z-Transform at N uniformly spaced points on the unit circle, i.e., for

$$z = e^{j\frac{2\pi}{N}k}, \text{ with } k = 0, 1, \ldots, N-1.$$

As the DTFT coincides with the Z-Transform evaluated on the unit circle, the DFT can be obtained by sampling the DTFT at N points uniformly spaced on the interval $[0, 2\pi]$:

$$X_p(k) = X(e^{j\omega})\Big|_{\omega=\frac{2\pi}{N}k}.$$

When dealing with finite-length sequences, it is also possible to express the Z-Transform (and the DTFT) in terms of the samples of the DFT:

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

$$= \sum_{n=0}^{N-1} \left(\frac{1}{N} \sum_{k=0}^{N-1} X_p(k)e^{j\frac{2\pi}{N}nk} \right) z^{-n} =$$

$$= \sum_{k=0}^{N-1} \frac{X_p(k)}{N} \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}nk} z^{-n} =$$

$$= \sum_{k=0}^{N-1} \frac{X_p(k)}{N} \frac{1 - z^{-N}}{1 - e^{j\frac{2\pi}{N}k} z^{-1}}$$

This is the *interpolation formula of Lagrange*. A similar relation also holds for the DTFT.
Thus, we observe that for finite-length sequences, the representations through DTFT, Z-Transform, and DFT are equivalent and interchangeable. The DFT serves as a straightforward means for assessing the spectral content of a sequence, making it well-suited for computer implementation.

For a sequence $x(n)$ of length N, a more detailed evaluation of the spectrum $X(e^{j\omega})$ can be achieved by employing a DFT on L samples, where $L > N$. In other words, it suffices to periodically extend the sequence $x(n)$ with a period $L > N$ and then apply the DFT to this extended periodic sequence.

This property proves valuable when considering fast convolution algorithms, which apply the DFT to finite-length sequences of L samples, where L is a power of 2, i.e., $L = 2^k$.

What happens when we consider an infinite-length sequence $x(n)$ with a DTFT and we sample its DTFT at N uniformly spaced points on the unit circle, and then invert these samples using the IDFT?
Remember that, when we sample a continuous-time signal at uniformly spaced intervals, in the frequency domain, we obtain the periodic repetition of the continuous-time spectrum, leading to the problem of aliasing. The coefficients of the DFT can be considered as samples of a continuous function in ω, representing the DTFT $X(e^{j\omega})$.
This frequency-domain sampling operation results in the periodic repetition of the signal $x(n)$ in the time domain, leading to aliasing in the time domain if $x(n)$ has infinite length.
Let's consider $X(e^{j\omega})$ and take N uniformly spaced samples in the interval $[0, 2\pi]$. We interpret these samples as the samples of the DFT:

$$X_p(k) = X(e^{j\omega})\Big|_{\omega=\frac{2\pi}{N}k} \qquad \text{with } 0 \leq k \leq N-1$$

$$= \sum_{n=-\infty}^{+\infty} x(n) e^{-j\frac{2\pi}{N}nk}.$$

Let's apply the IDFT:

$$x_p(n) = \frac{1}{N}\sum_{k=0}^{N-1} X_p(k) e^{j\frac{2\pi}{N}kn} =$$

$$= \frac{1}{N}\sum_{k=0}^{N-1}\sum_{m=-\infty}^{+\infty} x(m) e^{-j\frac{2\pi}{N}mk} e^{j\frac{2\pi}{N}kn} =$$

$$= \sum_{m=-\infty}^{+\infty} x(m) \frac{1}{N}\sum_{k=0}^{N-1} e^{-j\frac{2\pi}{N}(m-n)k}$$

We know that:

$$\frac{1}{N}\sum_{k=0}^{N-1} e^{-j\frac{2\pi}{N}(m-n)k} = \begin{cases} 1 & \text{when } m-n = lN, \text{ with } l \in \mathbb{Z} \\ 0 & \text{otherwise} \end{cases}$$

$$= \sum_{l=-\infty}^{+\infty} \delta(n-m-lN).$$

Hence,

$$x_p(n) = \sum_{m=-\infty}^{+\infty} x(m) \sum_{l=-\infty}^{+\infty} \delta(n-m-lN) =$$

$$= \sum_{l=-\infty}^{+\infty}\sum_{m=-\infty}^{+\infty} x(m)\delta(n-m-lN) =$$

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

The signal $x_p(n)$ is obtained as the sum of the signal $x(n)$ and its periodic repetitions, each time-shifted by a multiple of N. If the length of the signal $x(n)$ exceeds N, $x_p(n)$ is subject to aliasing errors in the time domain. Only signals $x(n)$ with a length less than or equal to N remain unaffected by aliasing.

DFT in matrix form

The DFT can be represented in matrix form through the following relation:

$$\mathbf{X} = \mathbf{D}_N \cdot \mathbf{x}.$$

Here, $\mathbf{X}$ is a column vector consisting of the first N samples of the DFT $X_p(k)$, and $\mathbf{x}$ is a column vector composed of the first N samples of $x_p(n)$.

$$\mathbf{X} = \begin{bmatrix} X_p(0) \\ X_p(1) \\ \vdots \\ X_p(N-1) \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_p(0) \\ x_p(1) \\ \vdots \\ x_p(N-1) \end{bmatrix}$$

The matrix $\mathbf{D}_N$, known as the *DFT matrix*, is defined as follows:

$$\mathbf{D}_N = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W_N^1 & W_N^2 & W_N^3 & \dots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & W_N^6 & \dots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & W_N^{3(N-1)} & \dots & W_N^{(N-1)^2} \end{bmatrix}$$

where $W_n = e^{-j\frac{2\pi}{N}}$.

The validity of this relation can be easily confirmed using the definition of the DFT:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}nk} = \sum_{n=0}^{N-1} x(n) W_N^{nk}.$$

Similarly, the IDFT can be expressed as:

$$\mathbf{x} = \mathbf{D}_N^{-1} \cdot \mathbf{X}$$

where the expression for the IDFT matrix $\mathbf{D}_N^{-1}$ is given by

$$\mathbf{D}_N^{-1} = \frac{1}{N} \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W_N^{-1} & W_N^{-2} & W_N^{-3} & \dots & W_N^{-(N-1)} \\ 1 & W_N^{-2} & W_N^{-4} & W_N^{-6} & \dots & W_N^{-2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & W_N^{-(N-1)} & W_N^{-2(N-1)} & W_N^{-3(N-1)} & \dots & W_N^{-(N-1)^2} \end{bmatrix}.$$

It can be verified that $\mathbf{D}_N^{-1} = \frac{1}{N}\mathbf{D}_N^*$.

06.03 Properties of the DFT

Linearity

If $x_1(n) \overset{DFT}{\longleftrightarrow} X_1(k)$ and $x_2(n) \overset{DFT}{\longleftrightarrow} X_2(k)$, then

$$a_1 x_1(n) + a_2 x_2(n) \overset{DFT}{\longleftrightarrow} a_1 X_1(k) + a_2 X_2(k).$$

Circular time-shift
If $x_p(n) \overset{DFT}{\longleftrightarrow} X_p(k)$ then, for $n_0 \in \mathbb{Z}$,

$$x_p(n - n_0) \overset{DFT}{\longleftrightarrow} X_p(k) e^{-j\frac{2\pi}{N} n_0 k}.$$

Proof:

$$\text{DFT}[x_p(n-n_0)] = \sum_{n=0}^{N-1} x_p(n-n_0) e^{-j\frac{2\pi}{N}nk} =$$

(consider $m = n - n_0$)

$$= \sum_{m=-n_0}^{N-1-n_0} x_p(m) e^{-j\frac{2\pi}{N}mk} e^{-j\frac{2\pi}{N}n_0 k} =$$

$$= \sum_{m=-n_0}^{-1} x_p(m) e^{-j\frac{2\pi}{N}mk} e^{-j\frac{2\pi}{N}n_0 k} + \sum_{m=0}^{N-1-n_0} x_p(m) e^{-j\frac{2\pi}{N}mk} e^{-j\frac{2\pi}{N}n_0 k} =$$

$$= \sum_{m=N-n_0}^{N-1} x_p(m) e^{-j\frac{2\pi}{N}mk} e^{-j\frac{2\pi}{N}n_0 k} + \sum_{m=0}^{N-1-n_0} x_p(m) e^{-j\frac{2\pi}{N}mk} e^{-j\frac{2\pi}{N}n_0 k} =$$

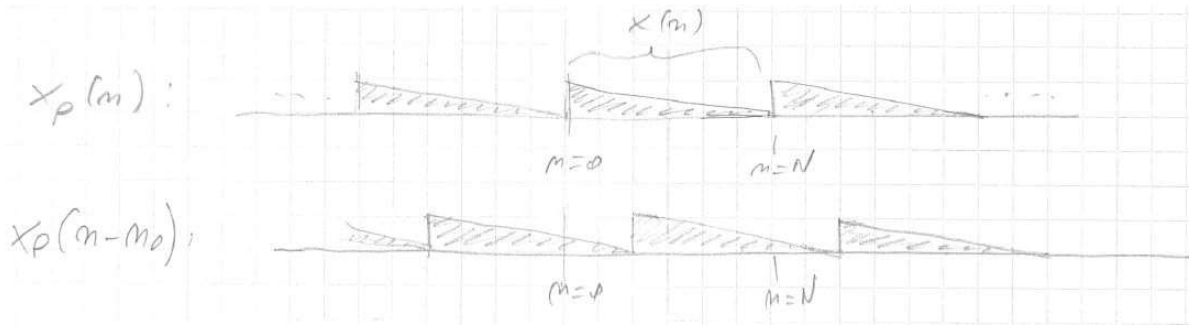
(because $e^{-j\frac{2\pi}{N}mk} = e^{-j\frac{2\pi}{N}(m+N)k}$ and $x_p(m) = x_p(m+N)$)

$$= \sum_{m=0}^{N-1} x_p(m) e^{-j\frac{2\pi}{N}mk} e^{-j\frac{2\pi}{N}n_0 k} =$$

$$= X_p(k) e^{-j\frac{2\pi}{N}n_0 k}$$

Q.E.D.

Note the distinction from the case of the discrete-time Fourier transform: $x_p(n)$ represents a periodic sequence or the periodic repetition of a finite-length sequence.



When considering the window from time 0 to $N-1$, it becomes apparent that the samples that 'exit' from one side 'enter' from the other side. For this reason we talk about a *circular time-shift*.
If $\langle m \rangle_N$ denotes the modulus of m with respect to N, and $x_p(n)$ represents the periodic repetition of a finite-length sequence $x(n)$, then we have:

$$x_p(n - n_0) = x_p(\langle n - n_0 \rangle_N) = x(\langle n - n_0 \rangle_N).$$

$$x_p(\langle n-n_0\rangle_N) \overset{DFT}{\longleftrightarrow} X_p(k)e^{-j\frac{2\pi}{N}n_0k}.$$

Circular frequency shift
If $x_p(n) \overset{DFT}{\longleftrightarrow} X_p(k)$ then, for $k_0 \in \mathbb{Z}$,

$$x_p(n)e^{j\frac{2\pi}{N}nk_0} \overset{DFT}{\longleftrightarrow} X_p(k-k_0) = X_p(\langle k-k_0\rangle_N).$$

Proof:

$$\sum_{n=0}^{N-1} x_p(n)e^{j\frac{2\pi}{N}nk_0}e^{-j\frac{2\pi}{N}nk} = \sum_{n=0}^{N-1} x_p(n)e^{-j\frac{2\pi}{N}n(k-k_0)} = X_p(k-k_0)$$

Conjugation
If $x_p(n) \overset{DFT}{\longleftrightarrow} X_p(k)$ then

$$x_p^*(n) \overset{DFT}{\longleftrightarrow} X_p^*(-k) = X_p^*(\langle -k\rangle_N).$$

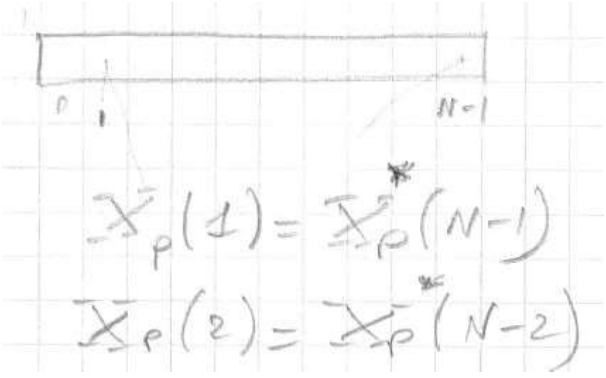
Proof:

$$\sum_{n=0}^{N-1} x_p^*(n)e^{-j\frac{2\pi}{N}nk} = \left[\sum_{n=0}^{N-1} x_p(n)e^{j\frac{2\pi}{N}nk}\right]^* =$$
$$\left[\sum_{n=0}^{N-1} x_p(n)e^{-j\frac{2\pi}{N}n(-k)}\right]^* = [X_p(-k)]^*.$$

If $x_p(n)$ is real, then $x_p^*(n) = x_p(n)$ and

$$X_p(k) = X_p^*(-k) = X_p^*(\langle -k\rangle_N) = X_p^*(\langle N-k\rangle_N) = X_p^*(N-k).$$

We term the DFT of a real sequence $x_p(n)$ as *circular conjugate-symmetric*.



If $x_p(n)$ is real, then:
$\text{Re}\{X_p(k)\} = \text{Re}\{X_p(N-k)\}$, i.e., it is *circular symmetric*;
$\text{Im}\{X_p(k)\} = -\text{Im}\{X_p(N-k)\}$, i.e., it is *circular anti-symmetric*;
$|X_p(k)| = |X_p(N-k)|$, i.e., it is *circular symmetric*;
$\arg\{X_p(k)\} = -\arg\{X_p(N-k)\}$, i.e., it is *circular anti-symmetric*.

If $x_p(n)$ is real and symmetric, i.e., if $x_p(n) = x_p(-n) = x_p(\langle -n \rangle_N) = x_p(N-n)$, then $X_p(k)$ is also real and symmetric.

Proof:

$$X_p(k) = \text{DFT}\{x_p(n)\} = \sum_{n=0}^{N-1} x_p(n) e^{-j\frac{2\pi}{N}nk}$$

$$\text{DFT}\{x_p(N-n)\} = \sum_{n=0}^{N-1} x_p(N-n) e^{-j\frac{2\pi}{N}nk} =$$

(Consider $m = N - n$, i.e., $n = N - m$)

$$= \sum_{m=1}^{N} x_p(m) e^{-j\frac{2\pi}{N}(N-m)k} =$$

(since $x_p(N) = x_p(0)$ and $e^{-j\frac{2\pi}{N}(N-N)} = e^{-j\frac{2\pi}{N}(N-0)} = 1$)

$$= \sum_{m=0}^{N-1} x_p(m) e^{-j\frac{2\pi}{N}(N-m)k} =$$

$$= \sum_{m=0}^{N-1} x_p(m) e^{j\frac{2\pi}{N}mk} = X_p^*(k)$$

$$\text{DFT}\left\{\frac{1}{2}[x_p(n) + x_p(N-n)]\right\} = \text{DFT}\left\{x_p(n)\right\} =$$

$$= \sum_{m=0}^{N-1} x_p(m) \frac{e^{j\frac{2\pi}{N}mk} + e^{-j\frac{2\pi}{N}mk}}{2} =$$

$$= \sum_{m=0}^{N-1} x_p(m) \cos\left[\frac{2\pi}{N}mk\right] \qquad \in \mathbb{R}$$

Q.E.D.

If $x_p(n)$ is real and anti-symmetric, i.e., if $x_p(n) = -x_p(-n) = -x_p(\langle -n \rangle_N) = -x_p(N-n)$, then $X_p(k)$ is imaginary and anti-symmetric.

In general, the DFT transforms complex sequences into complex sequences. However, we can efficiently use a single DFT transformation to compute the DFT of two real sequences.

Consider two real sequences, $x(n)$ and $y(n)$,

$$x(n) \overset{DFT}{\longleftrightarrow} X(k)$$

$$y(n) \overset{DFT}{\longleftrightarrow} Y(k)$$

Now, let's form the sequence

$$z(n) = x(n) + jy(n).$$

By leveraging the linearity property of the DFT:

$$Z(k) = X(k) + jY(k)$$

$$Z^*(N-k) = X^*(N-k) - jY^*(N-k)$$

Given that $x(n)$ and $y(n)$ are real sequences:

$$X^*(N-k) = X(k)$$

$$Y^*(N-k) = Y(k)$$

$$Z^*(N-k) = X(k) - jY(k)$$

Consequently, we obtain the following expressions:

$$X(k) = \frac{Z(k) + Z^*(N-k)}{2}$$

$$Y(k) = \frac{Z(k) - Z^*(N-k)}{2j}$$

$$\mathsf{Re}\{X(k)\} = \frac{\mathsf{Re}\{Z(k)\} + \mathsf{Re}\{Z(N-k)\}}{2}$$
$$\mathsf{Im}\{X(k)\} = \frac{\mathsf{Im}\{Z(k)\} - \mathsf{Im}\{Z(N-k)\}}{2}$$
$$\mathsf{Re}\{Y(k)\} = \frac{\mathsf{Im}\{Z(k)\} + \mathsf{Im}\{Z(N-k)\}}{2}$$
$$\mathsf{Im}\{Y(k)\} = \frac{-\mathsf{Re}\{Z(k)\} + \mathsf{Re}\{Z(N-k)\}}{2}$$

Parseval Theorem

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

Given a finite length sequence $g(n)$ of length N, we can evaluate its energy from the DFT $G(k)$.
Proof:

$$\sum_{n=0}^{N-1} |g(n)|^2 = \sum_{n=0}^{N-1} g(n) g^*(n) =$$
$$= \sum_{n=0}^{N-1} g(n) \frac{1}{N} \sum_{k=0}^{N-1} G^*(k) e^{-j\frac{2\pi}{N}nk} =$$
$$= \frac{1}{N} \sum_{k=0}^{N-1} G^*(k) \sum_{n=0}^{N-1} g(n) e^{-j\frac{2\pi}{N}nk} =$$
$$= \frac{1}{N} \sum_{k=0}^{N-1} |G(k)|^2.$$

Q.E.D.

06.04 The Fast Fourier Transform (FFT)

The Fast Fourier Transform (FFT) algorithms are efficient methods for computing the Discrete Fourier Transform (DFT). In particular, we will explore the original algorithms proposed by Cooley and Tukey in 1965, which specifically address sequences of length $N = 2^L$, i.e., of a power of 2 length.
The fundamental strategy of the FFT lies in computing the Discrete Fourier Transform (DFT) of a sequence by leveraging the DFTs of some reduced-length sequences.
The computation of $X(k) = \sum_{k=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}nk}$ requires N complex multiplications and $N-1$ complex additions. If we aim to compute the values of $X(k)$ for $k = 0, 1, \ldots, N-1$, the overall computational cost includes:

- N^2 complex multiplications,
- $N(N-1)$ complex additions.

But if we break the computation of the length N DFT into the computation of two DFTs of length $N/2$, we have a total of

- $2 \cdot \left(\frac{N}{2}\right)^2$ complex multiplications,
- $2 \cdot \left(\frac{N}{2}\right) \cdot \left(\frac{N}{2} - 1\right)$ complex additions.

Thus, we halve the operations.
Considering $N = 2^L$, the sequence of length $N/2$ can itself be split into two sequences, and each of these sequences can further be split into two, and so on, until we have sequences of only two samples.
In the following, we denote $W_N = e^{-j\frac{2\pi}{N}}$, such that

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

We will consider two algorithms:

- Decimation-in-time FFT.
- Decimation in frequency FFT.

Decimation-in-time FFT

This algorithm is derived by splitting the sequence $x(n)$ into two subsequences:

- The sequence of elements with even indices, denoted as $x_1(n) = x(2n)$, where $n = 0, 1, \ldots, \frac{N}{2} - 1$.
- The sequence of elements with odd indices, denoted as $x_2(n) = x(2n+1)$, where $n = 0, 1, \ldots, \frac{N}{2} - 1$.

We have,

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

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

$$= \sum_{n=0}^{N/2-1} x_1(n) W_N^{2nk} + \sum_{n=0}^{N/2-1} x_2(n) W_N^{(2n+1)k}$$

Since $W_N^{2nk} = e^{-j\frac{2\pi}{N}2nk} = e^{-j\frac{2\pi}{\frac{N}{2}}nk} = W_{\frac{N}{2}}^{nk}$ it is

$$X(k) = \sum_{n=0}^{N/2-1} x_1(n) W_{\frac{N}{2}}^{nk} + W_N^k \sum_{n=0}^{N/2-1} x_2(n) W_{\frac{N}{2}}^{nk}$$

But $\sum_{n=0}^{N/2-1} x_1(n) W_{\frac{N}{2}}^{nk}$ and $\sum_{n=0}^{N/2-1} x_2(n) W_{\frac{N}{2}}^{nk}$represent the DFTs computed on $\frac{N}{2}$ samples of $x_1(n)$ and $x_2(n)$, respectively. Thus,

$$X(k) = X_1(k) + W_N^k X_2(k).$$

The DFT $X(k)$ is the sum of two DFTs of length $\frac{N}{2}$, where the second DFT is multiplied by W_N^k. Note that $X_1(k)$ and $X_2(k)$ are periodic sequences of period $\frac{N}{2}$. Thus, if we consider $0 \leq k \leq \frac{N}{2} - 1$,

$$X(k) = X_1(k) + W_N^k X_2(k),$$

$$X(k + \frac{N}{2}) = X_1(k + \frac{N}{2}) + W_N^{(k+\frac{N}{2})} X_2(k + \frac{N}{2}) =$$

$$= X_1(k) + W_N^{(k+\frac{N}{2})} X_2(k).$$

$$W_N^{(k+\frac{N}{2})} = e^{j\frac{2\pi}{N}(k+\frac{N}{2})} = e^{j\frac{2\pi}{N}k} \cdot e^{j\frac{2\pi}{N}\cdot\frac{N}{2}} = -W_N^k$$

$$X(k + \frac{N}{2}) = X_1(k) - W_N^k X_2(k).$$

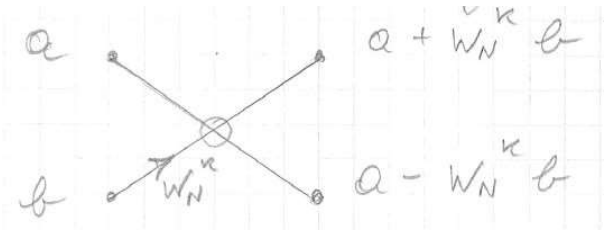
To summarize, the DFT of N samples can be computed using two DFTs on $\frac{N}{2}$ samples with the following equations:

$$X(k) = X_1(k) + W_N^k X_2(k),$$

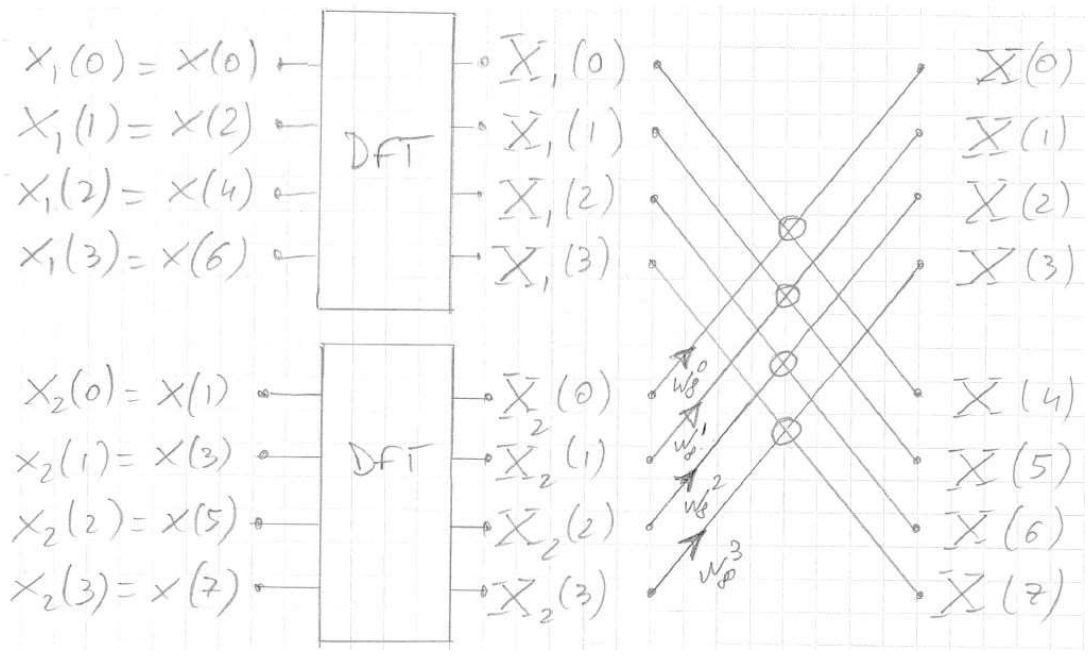
$$X(k + \frac{N}{2}) = X_1(k) - W_N^k X_2(k),$$

where $k = 0, 1, \ldots, \frac{N}{2} - 1$.

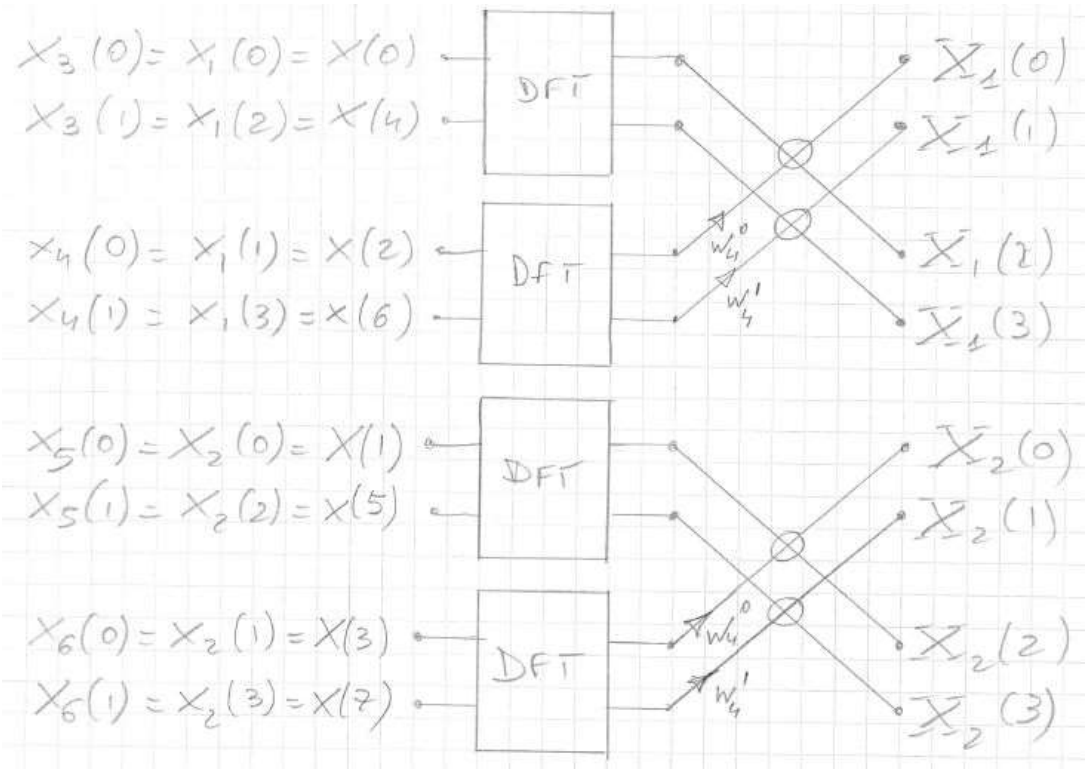
These operations form the fundamental steps of the FFT, and they are visually represented through the following *butterfly diagram*:



For an 8-sample DFT, the following operations take place:



Now, the $\frac{N}{2}$-sample length sequences can be further divided into sequences of $\frac{N}{4}$ samples, and so on, until we have sequences with only two samples.



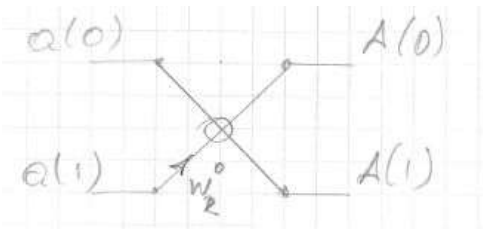
We use the term 'decimation in time' for this algorithm because the length-N sequence is successively decimated into sequences of length $\frac{N}{2}$, $\frac{N}{4}$, $\frac{N}{8}$, ..., 2.

The 2 samples DFT can be computed with a single butterfly. Indeed, given the sequence of two samples $\{a_0, a_1\}$,

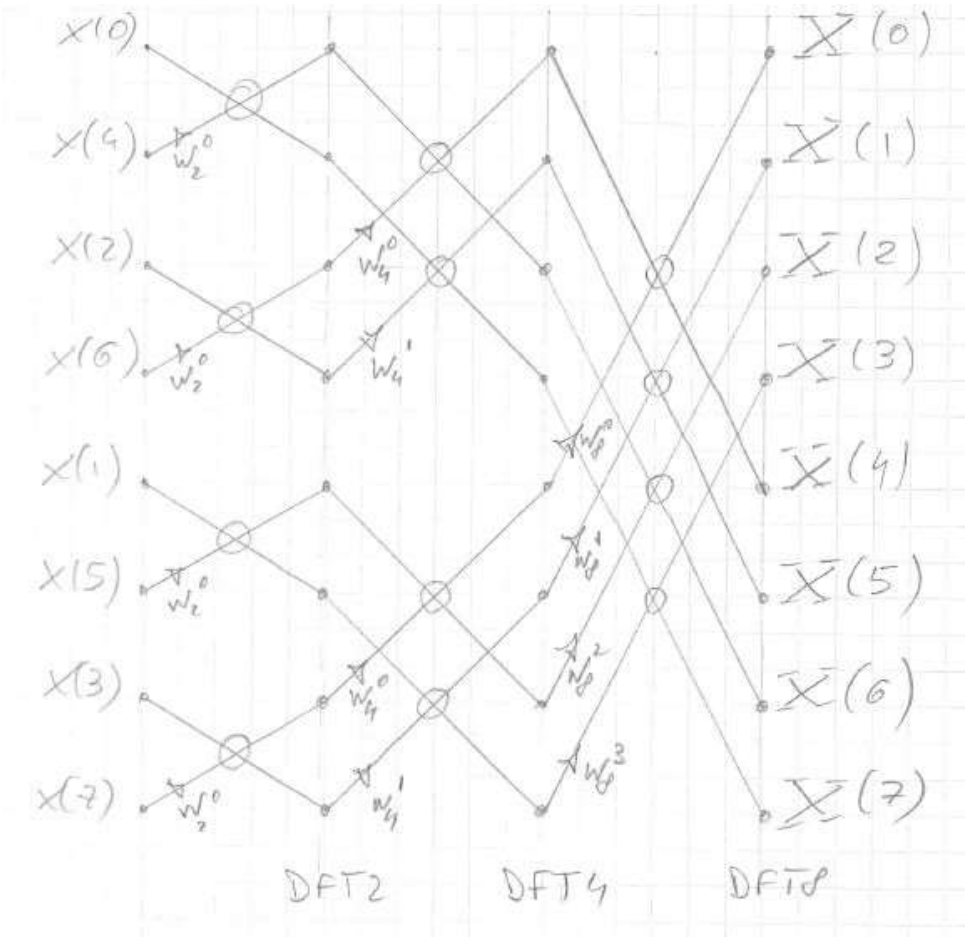
$$A(k) = \sum_{n=0}^{1} a(n) W_2^{nk} = \sum_{n=0}^{1} a(n) e^{-j\frac{2\pi}{2}nk} = a(0) + e^{-j\pi k} a(1)$$

$$A(0) = a(0) + a(1) = a(0) + W_2^0 a(1)$$

$$A(1) = a(0) - a(1) = a(0) - W_2^0 a(1)$$



Eventually, the computation of the 8-sample DFT involves the following operations:



To compute a DFT of length N (where N is a power of 2), we have $\log_2(N)$ butterfly stages, each composed of $\frac{N}{2}$ butterflies. Consequently, each stage involves $\frac{N}{2}$ complex multiplications and N complex additions/subtractions (neglecting the zero cost of multiplications by W_N^0). In total, the FFT computation requires:

- $\frac{N}{2}\log_2(N)$ complex multiplications,
- $N\log_2(N)$ complex additions.

When compared to the direct computation of the DFT, even for small N, this method yields significant computational savings.

Computational cost of the FFT

	N	4	8	16	32	64	128	256	512	1.024
FFT	$\times$	4	12	32	80	192	448	1.024	2.392	5.120
	$+$	8	24	64	160	384	896	2.048	4.696	10.240
DFT	$\times$	12	72	240	992	4.032	16.256	65.280	261.632	1.047.532
	$+$	16	64	256	1.024	4.096	16.384	65.536	262.144	1.048.576

Take note of the specific sample order at the input of the FFT. By decimating the sequence $x(n)$ $\log_2(N)$ times, we obtain a permuted sequence that serves as the input for the decimation-in-time FFT. For an

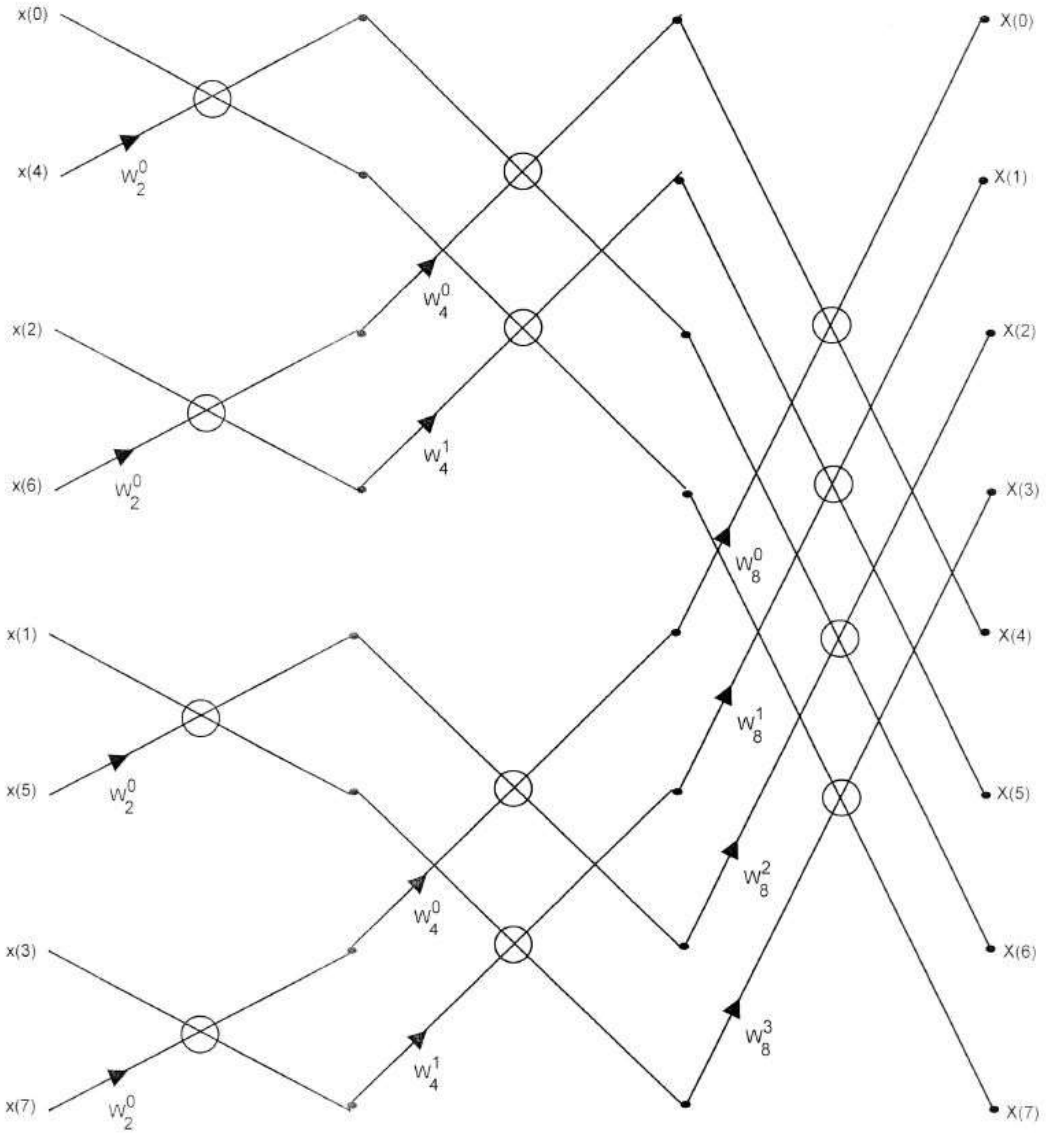


Figure 06.01: Decimation-in-time FFT

8-sample FFT, starting with

$$x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)$$

we rearrange it to:

$$x(0), x(4), x(2), x(6), x(1), x(5), x(3), x(7).$$

The permuted sequence can be easily constructed using the *bit-reversal rule*.
Let's denote the permuted sequence, which serves as the input of the FFT, as $y(n)$. Here, $y(n) = x(m)$, where the index m is obtained by representing n in binary form and inverting the bits of the representation:

0	000	$\longrightarrow$	000	0
1	001	$\longrightarrow$	100	4
2	010	$\longrightarrow$	010	2
3	011	$\longrightarrow$	110	6
4	100	$\longrightarrow$	001	1
5	101	$\longrightarrow$	101	5
6	110	$\longrightarrow$	011	3
7	111	$\longrightarrow$	111	7

Applying the decimation-in-time FFT algorithm to the sequence permuted according to the bit-reversal rule, we obtain the DFT samples in their natural order.

Decimation-in-frequency FFT
It is the dual algorithm to the decimation-in-time FFT.
It can be obtained by splitting the sequence $x(n)$ into two adjacent sequences:

- $x_1(n) = x(n)$ with $n = 0, 1, \ldots, \frac{N}{2} - 1$,
- $x_2(n) = x(n + \frac{N}{2})$ with $n = 0, 1, \ldots, \frac{N}{2} - 1$.

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

(considering $m = n + \frac{N}{2}$ in the second summation)

$$= \sum_{n=0}^{N/2-1} x_1(n) W_N^{nk} + \sum_{n=0}^{N/2-1} x_2(n) W_N^{(n+\frac{N}{2})k} =$$

$$= \sum_{n=0}^{N/2-1} \left[x_1(n) + W_N^{\frac{N}{2}k} x_2(n) \right] W_N^{nk}$$

But $W_N^{\frac{N}{2}k} = e^{-j\frac{2\pi}{N}\frac{N}{2}k} = e^{-j\pi k}$. Thus, we consider two cases: when k is even and when k is odd.
For k even:

$$X(2k) = \sum_{n=0}^{N/2-1} \left[x_1(n) + x_2(n) \right] W_N^{n2k} =$$

$$= \sum_{n=0}^{N/2-1} [x_1(n) + x_2(n)] W_{\frac{N}{2}}^{nk}$$

The even terms of $X(k)$ are determined by the $\frac{N}{2}$ points DFT of $x_1(n) + x_2(n)$.

For k odd:

$$X(2k+1) = \sum_{n=0}^{N/2-1} [x_1(n) - x_2(n)] W_N^{n(2k+1)} =$$

$$= \sum_{n=0}^{N/2-1} [x_1(n) - x_2(n)] W_N^n W_{\frac{N}{2}}^{nk}$$

The odd terms of $X(k)$ are determined by the $\frac{N}{2}$ points DFT of $(x_1(n) - x_2(n)) \cdot W_N^n$.

Thus, the procedure for computing the N-sample DFT consists of constructing two sequences of $\frac{N}{2}$ samples:

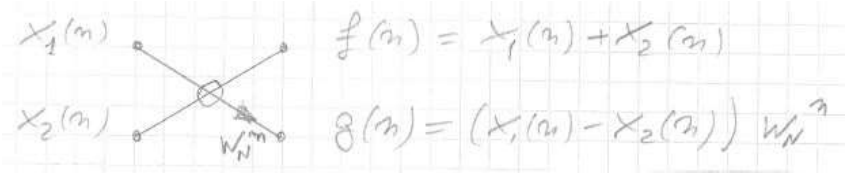
$$f(n) = x_1(n) + x_2(n)$$

$$g(n) = (x_1(n) - x_2(n)) W_N^n$$

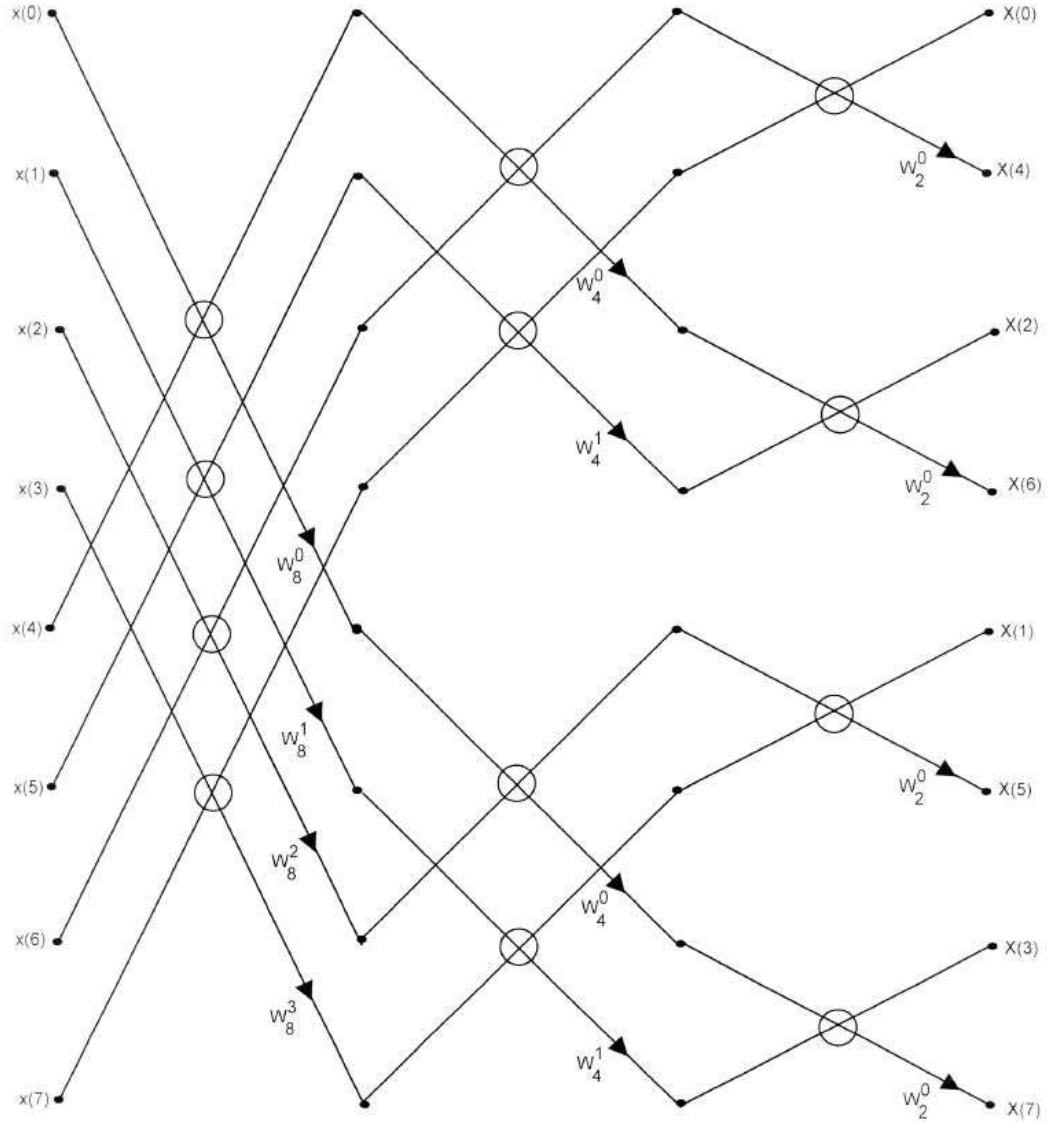
with $n = 0, 1, \ldots, N/2 - 1$, and then computing their $\frac{N}{2}$ samples DFT. The DFT of $f(n)$ provides the even terms of $X(k)$, and the DFT of $g(n)$ provides the odd terms of $X(k)$.

The procedure can be repeated to transition from the $\frac{N}{2}$-sample DFT to the $\frac{N}{4}$-sample DFT, and so on, until reaching DFTs of only 2 samples, which can be computed directly.

Again, the fundamental operations of the FFT can be illustrated using a butterfly diagram, albeit different from the previous one:



For an 8-samples DFT, we obtain the following scheme



In this case, while the input sequence follows the natural order, the FFT output samples are ordered according to the bit-reversal rule.

This is called the decimation-in-frequency algorithm because it involves decimating the DFT into sequences of odd and even terms.

The computational cost of this algorithm is identical to that of the decimation-in-time algorithm.

IDFT computed with the FFT

If we compare the expression of the IDFT with that of the DFT:

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

$$X(k) = \sum_{n=0}^{N-1} x(n) W^{nk},$$

we observe that, apart from the constant factor $\frac{1}{N}$, the expressions of the direct transform and the inverse transform are almost identical. The fast algorithms for computing the DFT can also be applied for computing the IDFT, with the only modification being to replace $W_2, W_4, \ldots, W_N$ with $W_2^{-1}, W_4^{-1}, W_N^{-1}$, respectively, and to divide the result by N.
Alternatively, we can directly apply the FFT algorithm to compute the IDFT. In this case,

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-nk} = \frac{1}{N} \left[\sum_{k=0}^{N-1} X^*(k) W_N^{nk} \right]^*$$

The IDFT can be computed by transforming the conjugate of $X(k)$ using the FFT, conjugating the resulting sequence, and dividing it by N.

FFT of real sequences
Up to now, we have considered the DFT of complex sequences. When dealing with real sequences, it is possible to achieve additional computational savings by exploiting the symmetry properties of these sequences.
For example, in the context of the decimation-in-time FFT, the two DFTs on $\frac{N}{2}$ real samples can be computed with a single FFT on $\frac{N}{2}$ complex samples. This FFT involves the transformation of the sequence $z(n)$ defined as

$$z(0) = x(0) + jx(1) = x_1(0) + jx_2(0)$$

$$z(1) = x(2) + jx(3) = x_1(1) + jx_2(1)$$

$$z(2) = x(4) + jx(5) = x_1(2) + jx_2(2)$$

$$\vdots$$

$$z(N/2-1) = x(N-2) + jx(N-1) = x_1(N/2-1) + jx_2(N/2-1)$$

We have:

$$Z(0) = X_1(0) + jX_2(0)$$

$$Z(1) = X_1(1) + jX_2(1)$$

$$\vdots$$

$$Z(N/2-1) = X_1(N/2-1) + jX_2(N/2-1)$$

Utilizing the symmetry properties of real sequences, we derive the following relations:

$$X_1(k) = \frac{Z(k) + Z^*(N/2-k)}{2}$$

$$X_2(k) = \frac{Z(k) - Z^*(N/2-k)}{2j}$$

Thus, the computation of $X_1(k)$ and $X_2(k)$ from $Z(k)$ requires only additions, subtractions, and division by 2.
Then, we can apply the last butterfly stage to $X_1(k)$ and $X_2(k)$ to compute $X(0), X(1), \ldots, X(N/2)$. The terms $X(N/2+1), \ldots, X(N-1)$ can be computed without any additional operations by exploiting the conjugate symmetry property of $X(k)$.

If, as is often the case in practice, we need to compute the N-sample DFT of two real sequences $x(n)$ and $y(n)$, we can efficiently apply a single FFT to the sequence $z(n) = x(n) + jy(n)$. The DFTs of $x(n)$ and $y(n)$, denoted as $X(k)$ and $Y(k)$, can then be computed using the following formulas:

$$X(k) = \frac{Z(k) + Z^*(N-k)}{2}.$$

$$Y(k) = \frac{Z(k) - Z^*(N-k)}{2j}.$$

06.05 Linear convolution and circular convolution

We defined the convolution between two infinite-length sequences, $x(n)$ and $h(n)$, as

$$y(n) = \sum_{m=-\infty}^{+\infty} x(m)h(n-m) = \sum_{m=-\infty}^{+\infty} h(m)x(n-m) = x(n) \circledast h(n).$$

This convolution is also called *linear convolution*. When the two sequences have finite lengths N, i.e., $x(n) = 0 \quad \forall n < 0$ and $\forall n \geq N$, and $h(n) = 0 \quad \forall n < 0$ and $\forall n \geq N$, the convolution is given by

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

In this case, the resulting sequence $y(n)$ also has a finite length equal to $2N-1$, i.e. $x(n) = 0 \; \forall n < 0$ and $\forall n \geq 2N-1$. Indeed,

$$y(0) = x(0)h(0)$$

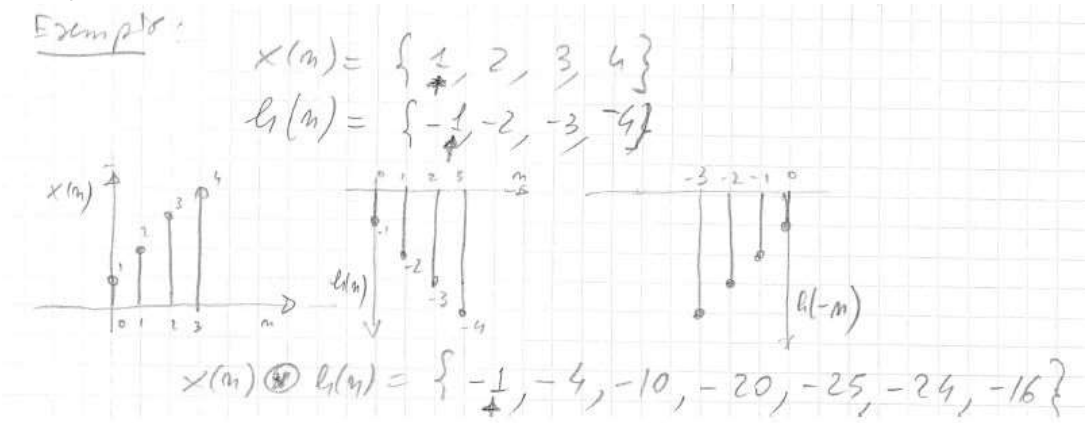
$$y(1) = x(0)h(1) + x(1)h(0)$$

$$y(2) = x(0)h(2) + x(1)h(1) + x(2)h(0)$$

$$\vdots$$

$$y(2N-2) = x(N-1)h(N-1)$$

Example:

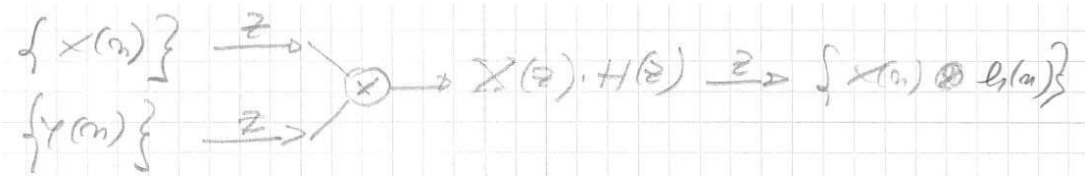


We have seen that an important property of DTFT and Z-Transform is the transformation of linear convolution into the product of the transforms of the two sequences:

$$x(n) \circledast h(n) \overset{DTFT}{\longleftrightarrow} X(e^{j\omega})H(e^{j\omega})$$

$$x(n) \circledast h(n) \overset{Z}{\longleftrightarrow} X(z)H(z)$$

Thus, it is possible to evaluate the convolution by computing the transform of $x(n)$ and of $h(n)$, multiplying these transforms, and then taking the inverse transform of the product:



A similar property holds for the DFT, but it involves a modified definition of convolution. The DFT applies to periodic sequences or sequences that are the periodic extension of finite-length sequences.
Given two periodic sequences with the same period N (or given two finite-length sequences with a duration less than or equal to N, periodically extended with period N), we define the *circular convolution* of the two sequences as the sequence $y_p(n)$ given by

$$y_p(n) = x_p(n) \textcircled{N} h_p(n) =$$

$$= \sum_{m=0}^{N-1} x_p(m) h_p(n-m) =$$

$$= \sum_{m=0}^{N-1} h_p(m) x_p(n-m) =$$

$$= \sum_{m=0}^{N-1} h_p(m) x_p(\langle n-m \rangle_N),$$

which is a periodic sequence of period N.
The circular convolution satisfies the commutative property and the distributive property:

$$x(n) \textcircled{N} h(n) = h(n) \textcircled{N} x(n)$$

$$(x_1(n) + x_2(n)) \textcircled{N} h(n) = x_1(n) \textcircled{N} h(n) + x_2(n) \textcircled{N} h(n)$$

The evaluation of the circular convolution involves the same operations we have seen for the linear convolution. If we want to compute

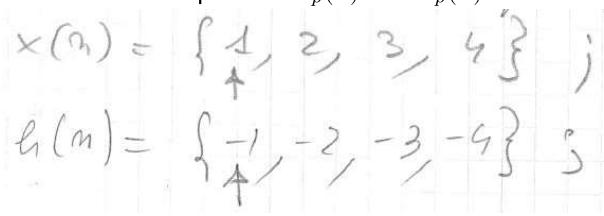
$$\sum_{m=0}^{N-1} x_p(m) h_p(n-m)$$

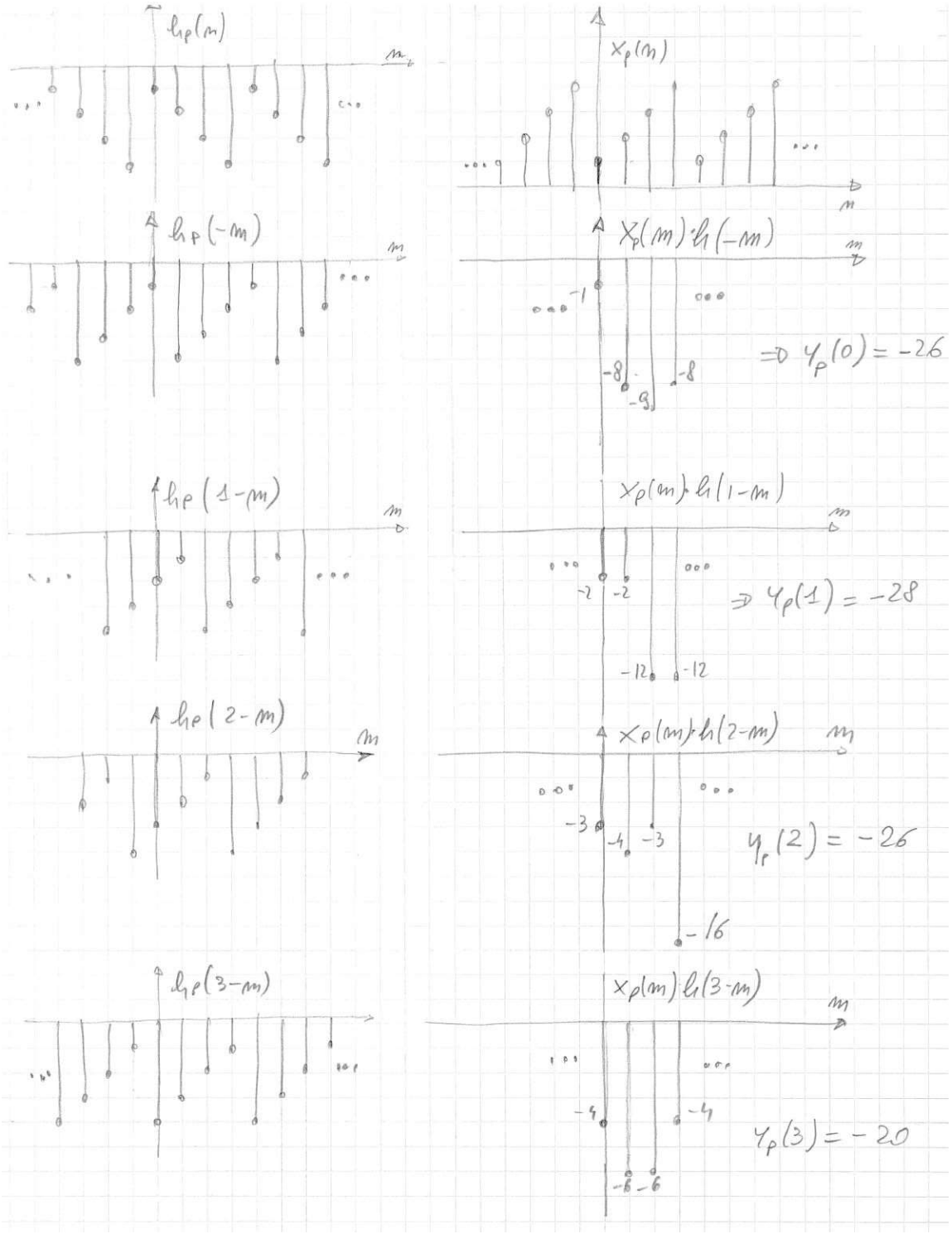
we need to:

1. Fold $h_p(n)$ to get $h_p(-m)$,
2. Delay $h_p(-m)$ by n samples to get $h_p(n-m)$,
3. Compute the sample-by-sample product between $x_p(m)$ and $h_p(n-m)$,
4. Sum all terms for $m = 0$ till $m = N-1$.

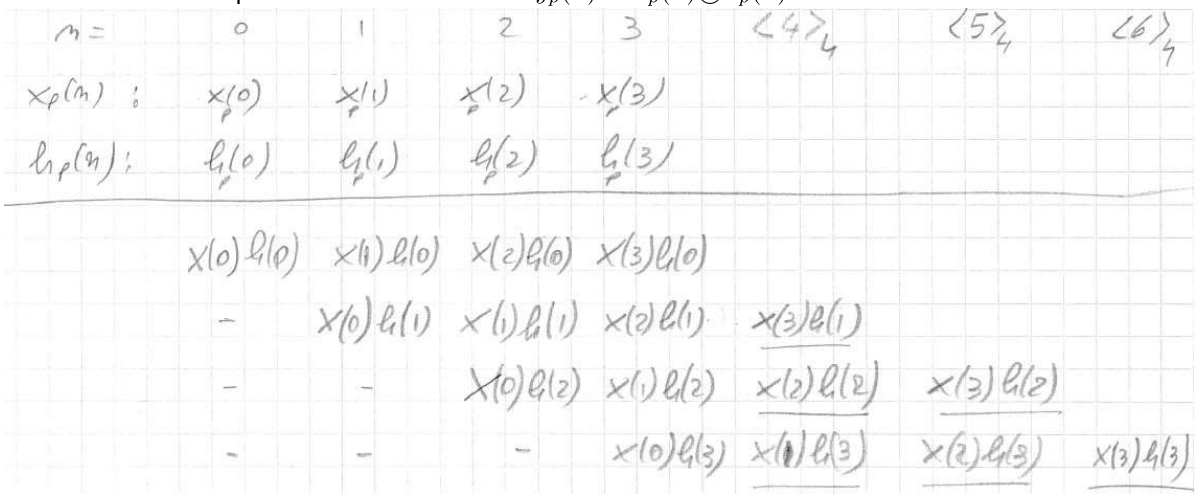
The main difference with the linear case comes from the fact that the sequences $x_p(n)$ and $h_p(n)$ are periodic with a period of N and that the sum is performed only over one period (from 0 to $N-1$).

Consider the sequences $x_p(n)$ and $h_p(n)$ which derive from the periodic repetition of

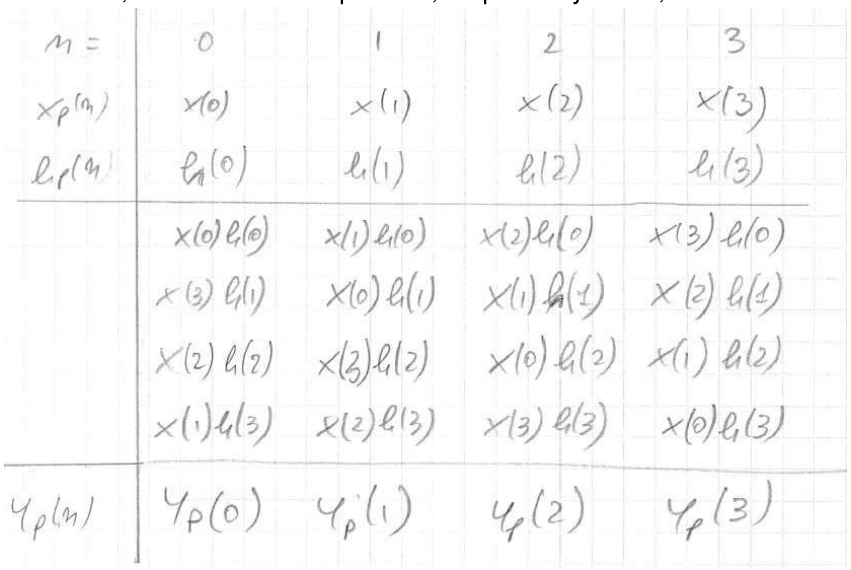




For computing the circular convolution, we can apply the tabular method we have seen for the linear convolution, with the only difference that we have to wrap around in the window $[0, N-1]$ the partial products of the second, third, ..., N-th row, that fall outside this window.
Let us consider the period from 0 to $N-1$ of $y_p(n) = x_p(n) \textcircled{N} h_p(n)$:



In the second row, the partial product $x(3)h(1)$ must be taken back to the first position. In the third row, the two partial products $x(2)h(2)$ and $x(3)h(2)$ must be taken back to the first and second position, respectively. In the forth row, the partial products $x(1)h(3)$, $x(2)h(3)$, and $x(3)h(3)$ must be taken back to the first, second and third position, respectively. Thus, we have:



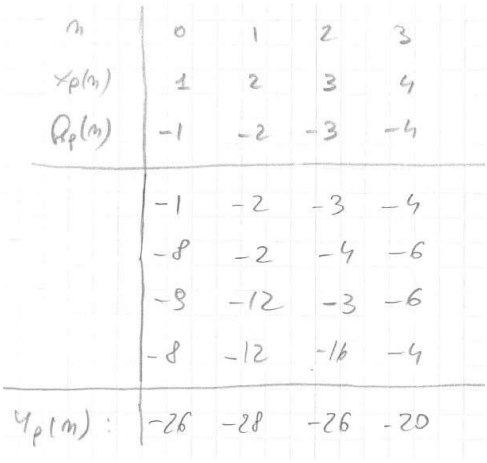
$$y_p(0) = x(0)h(0) + x(3)h(1) + x(2)h(2) + x(1)h(3)$$

$$y_p(1) = x(1)h(0) + x(0)h(1) + x(3)h(2) + x(2)h(3)$$

$$y_p(2) = x(2)h(0) + x(1)h(1) + x(0)h(2) + x(3)h(3)$$

$$y_p(3) = x(3)h(0) + x(2)h(1) + x(1)h(2) + x(0)h(3)$$

Example:



Property: The DFT transforms the circular convolution into the product of the transforms of the two sequences:

$$x_p(n) \textcircled{N} h_p(n) \overset{DFT}{\longleftrightarrow} X_p(k)H_p(k)$$

Proof:

$$x_p(n) \textcircled{N} h_p(n) = \sum_{m=0}^{N-1} x_p(m)h_p(n-m)$$

$$\text{DFT}\left\{x_p(n) \textcircled{N} h_p(n)\right\} = \sum_{n=0}^{N-1}\left(\sum_{m=0}^{N-1} x_p(m)h_p(n-m)\right)e^{-j\frac{2\pi}{N}nk} =$$

$$= \sum_{m=0}^{N-1} x_p(m)\left(\sum_{n=0}^{N-1} h_p(n-m)e^{-j\frac{2\pi}{N}(n-m)k}\right)e^{-j\frac{2\pi}{N}mk} =$$

$$= \sum_{m=0}^{N-1} x_p(m)H_p(k)e^{-j\frac{2\pi}{N}mk} =$$

$$= H_p(k)\sum_{m=0}^{N-1} x_p(m)e^{-j\frac{2\pi}{N}mk} = H_p(k)X_p(k)$$

Q.E.D.

This property is used for computing the *fast convolution* of two sequences. The linear convolution between two sequences can be obtained by means of the circular convolution of suitable periodic sequences, with the circular convolution computed in the DFT domain.

Linear convolution of two finite-length sequences

Consider first the computation of the linear convolution between two finite-length sequences.

Let us call $x(n)$ a sequence of length N, $x(n) = 0\ \forall n < 0$ and $\forall n \geq N$, and let us call $h(n)$ a sequence of length M, $x(n) = 0\ \forall n < 0$ and $\forall n \geq M$. Then, the linear convolution of $x(n)$ and $h(n)$,

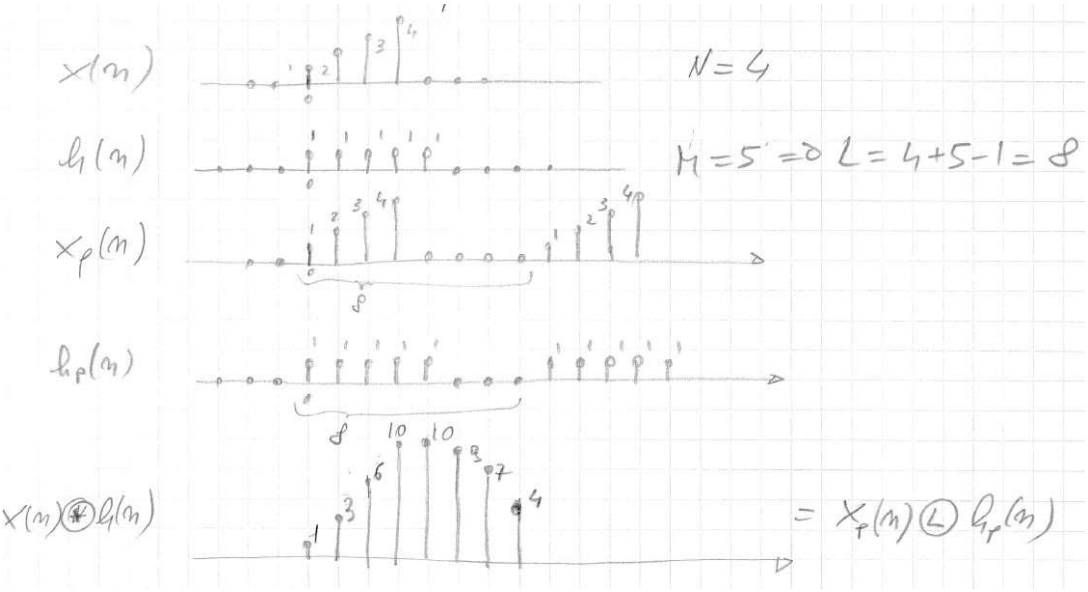
$$y(n) = \sum_{m=0}^{N-1} x(m)h(n-m)$$

has length $L = N + M - 1$. Indeed,

$$y(0) = x(0)h(0)$$

$$y(L-1) = x(N)h(M) = y(N+M-2)$$

Let us consider the two sequences $x_p(n)$ and $h_p(n)$ obtained from the periodic repetition of $x(n)$ and $h(n)$, respectively, with period L.



On the fundamental period:

$$y_p(n) = x_p(n) \textcircled{L} h_p(n) = \sum_{m=0}^{L-1} x_p(m) h_p(n-m) = \sum_{m=0}^{N-1} x(n) h(n-m) = x(n) \circledast h(n)$$

$y_p(n) = x_p(n) \textcircled{L} h_p(n)$ coincides with the periodic repetition with period L of the sequence $y(n) = x(n) \circledast h(n)$:

$$y(n) = x(n) \circledast h(n) = y_p(n) = x_p(n) \textcircled{L} h_p(n) \qquad \text{for } 0 \leq n < L.$$

Theorem:
Let $x(n)$ be a sequence of length N, $x(n) = 0\ \forall n < 0$ and $\forall n \geq N$, and let $h(n)$ be a sequence of length M, $h(n) = 0\ \forall n < 0$ and $\forall n \geq M$. Let us consider the two sequences $x_p(n)$ and $h_p(n)$ obtained from the periodic repetition of $x(n)$ and $h(n)$, respectively, with period $L = M + N - 1$. Then, for $0 \leq n \leq L-1$, it holds that

$$x_p(n) \textcircled{L} h_p(n) = x(n) \circledast h(n).$$

Proof: By construction, it is:

- $x_p(n) = x(n)$ for $0 \leq n \leq L-1$.
- $x_p(n) = 0$ for $N \leq n \leq L-1$.
- $h_p(n) = h(n)$ for $-(N-1) \leq n \leq L-1$.
  ($h_p(n) = 0$ for $-(N-1) \leq n < 0$).

Thus, for $0 \leq n \leq L-1$,

$$x_p(n) \textcircled{L} h_p(n) = \sum_{m=0}^{L-1} x_p(m) h_p(n-m) =$$

$$= \sum_{m=0}^{N-1} x_p(m) h_p(n-m) =$$

$$= \sum_{m=0}^{N-1} x(m) h_p(n-m)$$

For $0 \leq n \leq L-1$ and $0 \leq m \leq N-1$

$$-(N-1) \leq n-m \leq L-1 \qquad \Longrightarrow \qquad h_p(n-m) = h(n-m)$$

$$x_p(n) \textcircled{L} h_p(n) = \sum_{m=0}^{N-1} x(m) h(n-m) = x(n) \circledast h(n).$$

Q.E.D.

We can compute the linear convolution of two finite-length sequences by means of the following operations:

1. Compute the FFT on L points of $x(n)$ and $h(n)$.
2. Multiply $X(k)$ and $H(k)$ for $0 \leq k < L$.
3. Compute the inverse FFT transform on L points of $X(k)H(k)$.

The fast convolution technique with the FFT is efficient when the two sequences $x(n)$ and $h(n)$ have similar lengths ($N \simeq M$). If this is not true, we have to add a large number of zeros to the shortest sequence, increasing the computational cost considerably.
If we assume to know *a priori* $H(k)$, since the computational cost of the FFT is $\frac{L}{2}\log_2(L)$ complex multiplications and $L\log_2(L)$ complex additions, the total cost of the fast convolution algorithm is

$$2 \cdot \left(\frac{L}{2}\log_2(L)\right) + L = L\left(\log_2(L)+1\right) \qquad \text{complex multiplications,}$$

$$2 \cdot (L\log_2(L)) \qquad \text{complex additions.}$$

This computational cost must be compared with that of the direct computation of the convolution on L output samples which is:

$$L \cdot M \qquad \text{complex multiplications,}$$

$$L \cdot (M-1) \qquad \text{complex additions.}$$

In general, $\log_2(L)+1 \ll M$ and we have a significant computational saving.
Note that to apply this technique, we must process the entire input sequence before applying the fast convolution algorithm. Thus, there is a delay introduced, which is equal to the length of the input sequence

---

Linear convolution of a finite length sequence with an infinite length sequence.

The fast convolution method for finite-length sequences can be extended to compute the convolution between a finite-length sequence $h(n)$ and an infinite-length sequence $x(n)$. This is a very common situation: the FIR filtering of an infinite-length signal or a finite-length signal with very long duration. The trick used is to split the sequence $x(n)$ into many *segments* of finite length. For each of these segments, we can evaluate the linear convolution with a technique similar to that we have seen for finite-length sequences. There are two possible approaches:

- Overlap-add method
- Overlap-save method

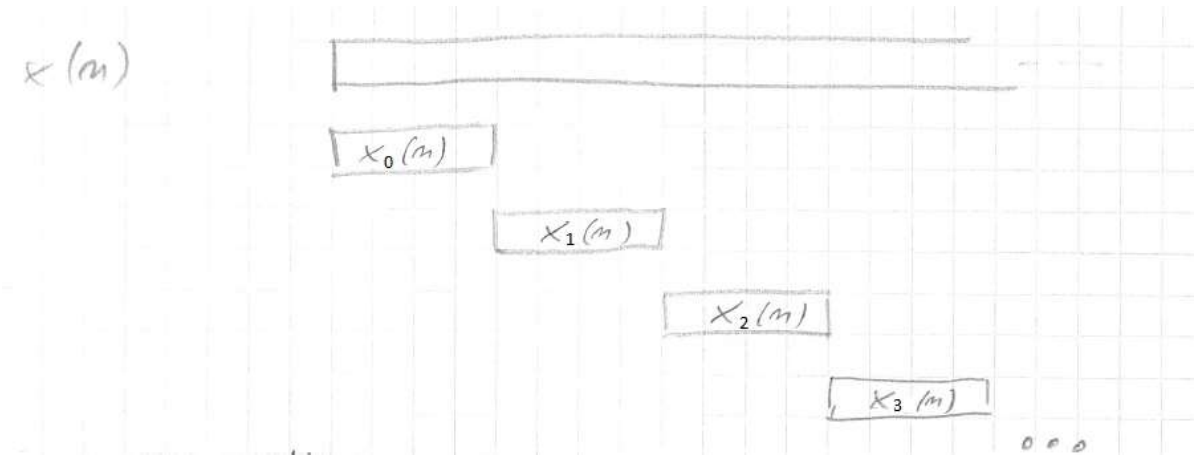
Overlap-add method.

In this method, the input sequence $x(n)$ is split into the sum of adjacent non-overlapping sequences of finite length N. Assuming the sequence $x(n)$ to be causal,

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

with

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



But, we have:

$$y(n) = \sum_{l=0}^{M-1} h(l)x(n-l) = \sum_{l=0}^{M-1} h(l) \sum_{m=0}^{+\infty} x_m(n - l - mN) =$$

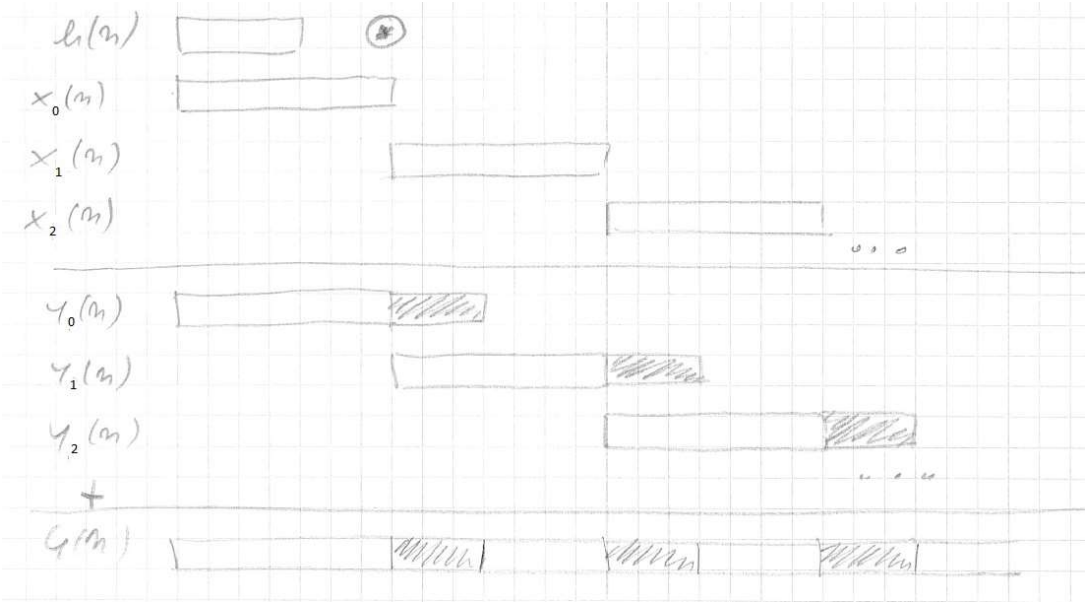
$$= \sum_{m=0}^{+\infty} \left[\sum_{l=0}^{M-1} h(l)x_m(n - mN - l) \right] = \sum_{m=0}^{+\infty} y_m(n - mN)$$

where

$$y_m(n) = \sum_{l=0}^{M-1} h(l)x_m(n-l) = h(n) \circledast x_m(n)$$

$y_m(n)$ is the convolution of two finite-length sequences. It can be computed by means of the FFT of the two sequences, the product of the FFTs, and an inverse FFT. If $h(n)$ has length M and $x_m(n)$ has

length N, $y_m(n)$ has length $L = M + N - 1 > N$. Thus, the sequences $y_m(n)$ overlap with the adjacent ones over $M - 1$ samples.



Since $y(n) = \sum_{m=0}^{+\infty} y_m(n - mN)$, the tails that temporally overlap are added together. For this reason, the method is called the 'overlap-add' method.

Thus, the overlap-add method is applied with the following procedure:

1. Split $x(n)$ in adjacent segments of length N.
2. Add $M-1$ zeros to each segment in order to obtain the sequences $x_m(n)$ of length $L = M+N-1$ (typically, L is a power of 2).
3. Compute $X_m(k)$ with FFT.
4. Compute $Y_m(k) = H(k)X_m(k)$ (with $H(k)$ computed once and *a priori*).
5. Compute the inverse FFT of $Y_m(k)$: $y_m(n) = \text{IFFT}\{Y_m(k)\}$.
6. Add (on $M - 1$ points) the overlapping tails to get $y(n) = \sum_{m=0}^{+\infty} y_m(n - mN)$.

Using Cooley and Tukey algorithms for computing the FFT and IFFT, it is convenient to use the decimation-in-frequency algorithm for the FFT and the decimation-in-time algorithm for the IFFT. In this way, we can avoid permutations with the bit-reversal rule. From the sequence with natural order $x_m(n)$, we obtain the sequence with bit-reversal order $X_m(k)$. We can think that $H(k)$ has also been obtained with the same decimation-in-frequency algorithm, and its samples are ordered with the bit-reversal rule. Multiplying element by element $H(k)$ and $X_m(k)$ and computing the IFFT with the decimation-in-time algorithm, we obtain $y_m(n)$ with the natural order.

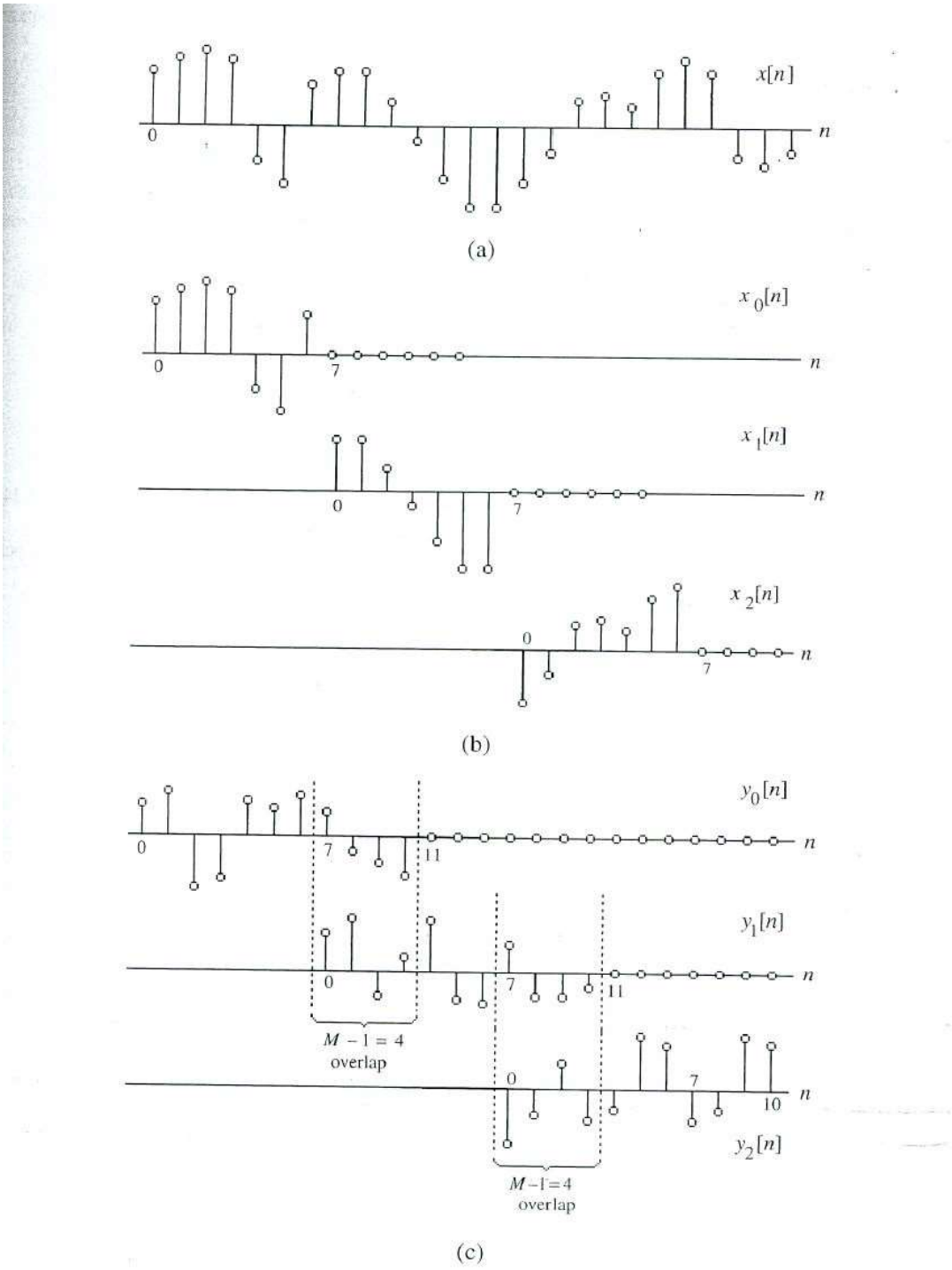


Figure 5.14: (a) Original $x[n]$, (b) segments $x_m[n]$ of $x[n]$, and (c) linear convolution of $x_m[n]$ with $h[n]$.

(From S. K. Mitra, “Digital signal processing: a computer based approach”, McGraw Hill, 2011)

Overlap-save method

Consider $h(n)$ of length M, $h(n) = 0\ \forall n < 0$ and $\forall n \geq M$, and $g(n)$ of length $L = M + N - 1$, $g(n) = 0$

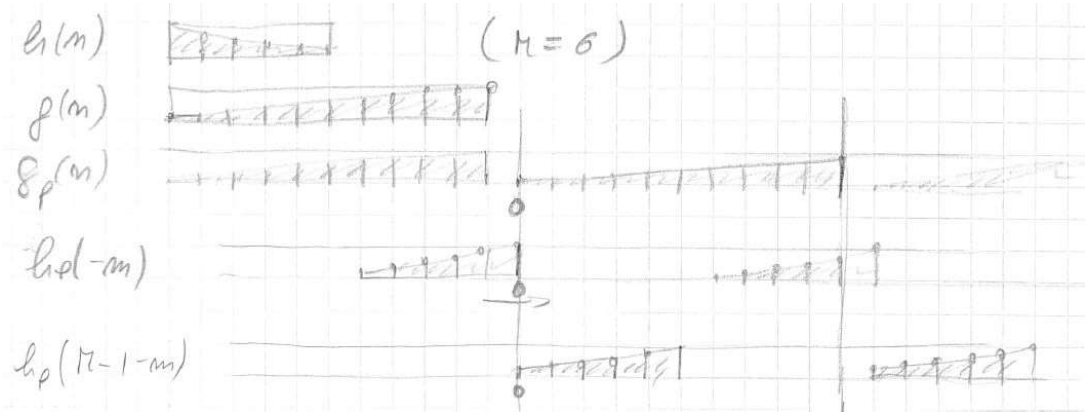
$\forall n < 0$ and $\forall n \geq L$. Let $h_p(n)$ and $g_p(n)$ be obtained from the periodic repetition of $h(n)$ and $g(n)$ with period L. The circular convolution $h_p(n) \textcircled{L} g_p(n) = y_p(n)$ is given by:

$$y_p(n) = \sum_{m=0}^{L-1} g_p(m) h_p(n-m) =$$

$$\text{for } M-1 \leq n \leq L-1: \qquad = \sum_{m=0}^{L-1} g(m) h(n-m) = y(n)$$

$$\text{for } 0 \leq n < M-1: \qquad \neq \sum_{m=0}^{L-1} g(m) h(n-m) = y(n)$$

For $M-1 \leq n \leq L-1$, $y_p(n)$ coincides with the linear convolution, while for the first $M-1$ samples, $y_p(n)$ differs from the linear convolution. These first $M-1$ samples are "affected" by the periodic repetition of the signals.



Theorem: Let $x(n)$ be a sequence of length L, $x(n) = 0\ \forall n < 0$ and $\forall n \geq L$. Similarly, let $h(n)$ be a sequence of length M, where $h(n) = 0$ for all $n < 0$ and $n \geq M$. Consider the sequences $x_p(n)$ and $h_p(n)$ obtained by the periodic repetition of $x(n)$ and $h(n)$, respectively, with a period of L. For $M-1 \leq n \leq L-1$, we have:

$$x_p(n) \textcircled{L} h_p(n) = x(n) \circledast h(n).$$

Proof:

Define $N = L - M + 1$ (i.e., $L = M + N - 1$) . By construction it is

- $x_p(n) = x(n)$ for $0 \leq n \leq L-1$.
- $h_p(n) = h(n)$ for $-(N-1) \leq n \leq L-1$.

Thus, for $M-1 \leq n \leq L-1$

$$x_p(n) \textcircled{L} h_p(n) = \sum_{m=0}^{L-1} x_p(m) h_p(n-m) =$$

$$= \sum_{m=0}^{L-1} x(m) h_p(n-m)$$

For $M-1 \leq n \leq L-1$ and $0 \leq m \leq L-1$,

$$M-1-(L-1) \leq n-m \leq L-1$$

$$M-1-(L-1) = M-1-(M+N-1-1) = -(N-1)$$

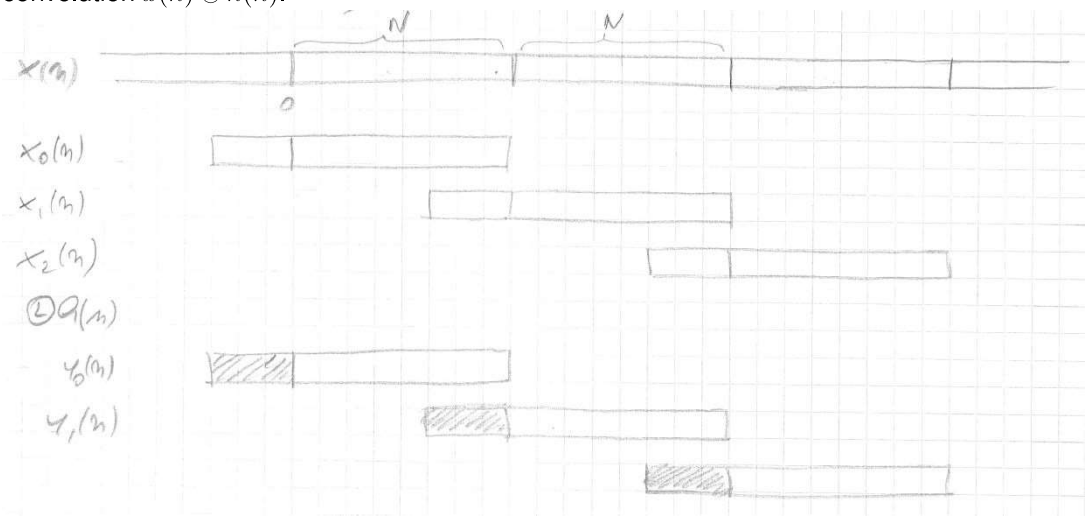
Thus,

$$-(N-1) \leq n-m \leq L-1 \quad \Longrightarrow \quad h_p(n-m) = h(n-m)$$

$$x_p(n) \textcircled{L} h_p(n) = \sum_{m=0}^{L-1} x(m)h(n-m) = x(n) \circledast h(n).$$

Q.E.D.

The overlap-save method exploits this property by splitting $x(n)$ into sequences $x_m(n)$ of length L, where the first $M-1$ samples of each sequence repeat the last $M-1$ samples of the previous sequence $x_{m-1}(n)$.
By computing the circular convolution on L samples, the first $M-1$ samples of $y_p(n)$ are affected by temporal aliasing and are discarded, while the remaining N samples coincide with those of the linear convolution $x(n) \circledast h(n)$.



(The darkened terms are discarded because they are affected by aliasing). Note that

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

Summarizing the overlap-save method is composed of the following operations:

1. Build the sequence $x_m(n)$ composed by the last $M-1$ samples of $x_{m-1}(n)$ and N new samples of $x(n)$.
2. Transform with the FFT $x_m(n)$ to obtain $X_m(k)$.
3. Multiply $X_m(k) \cdot H(k) = Y_m(k)$.
4. Compute the inverse transform $y_m(n) = \text{IFFT}\{Y_m(k)\}$.

5. Discard the first $M-1$ samples of $y_m(n)$ and keep the last N samples that provide $y(n)$.

Also in this case, it is convenient to use a decimation-in-frequency FFT for the direct transform and a decimation-in-time FFT for the inverse transform.

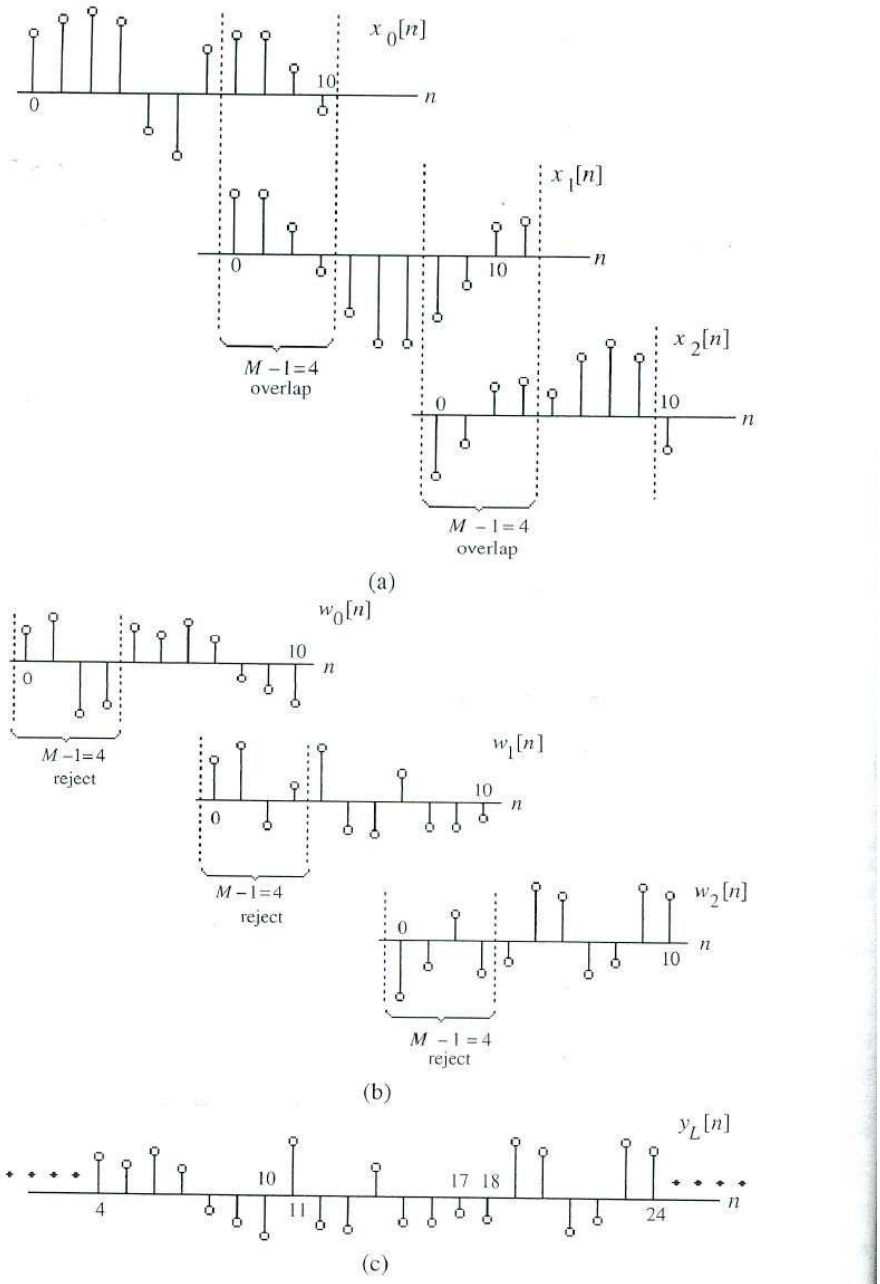


Figure 5.16: Illustration of the overlap-save method. (a) Overlapped segments of the sequence $x[n]$ of Figure 5.14(a), (b) sequences generated by an 11-point circular convolution, and (c) sequence obtained by rejecting the first four samples of $w_i[n]$ and abutting the remaining samples.

(From S. K. Mitra, "Digital signal processing: a computer based approach", McGraw Hill, 2011)

Computational cost of the overlap-save method

Since we work with an infinite-length input sequence $x(n)$ (or a very long finite-length input sequence),

it is convenient to refer to the complexity per single output sample.
By applying the convolution sum directly, the computational cost is equal to M complex multiplications and $M-1$ additions per sample.
In the case of the overlap-save technique, the computational cost is

$$\left[2\left(\frac{L}{2}\log_2(L)\right)+L\right]/N \qquad \text{complex multiplications,}$$

$$\left[2L\log_2(L)\right]/N \qquad \text{complex additions.}$$

Indeed, we have two FFTs of length L, L products $X(k)H(k)$, and from these, we obtain N output samples.
Also in this case it is convenient to choose $N > M$ and almost equal to M.

Example: Consider $h(n)$ of length $M = 128$. The convolution sum costs 128 multiplications and 127 additions per output sample. Assuming $L = 256$ ($N = 129$), the overlap-save technique has a cost of 17.86 multiplications and 31.75 additions per output sample. With $L = 512$ ($N = 385$), the overlap-save technique has a cost of 13.29 multiplications and 23.93 additions per output sample. Further increments in L do not lead to any further reduction in computational complexity.
Note that the higher is N (and thus L), the greater the delay introduced by the convolution using the overlap-add or overlap-save method.

Convolution of real sequences
If the FIR filter is real, and the sequence $x(n)$ is also real, it is possible to achieve significant computational savings by simultaneously computing the convolution of two consecutive segments. Indeed, given $x_m(n)$ and $x_{m+1}(n)$, we can construct

$$x_m(n) + jx_{m+1}(n)$$

and apply the fast convolution technique to this sequence. The corresponding output signal is

$$y_m(n) + jy_{m+1}(n).$$

By separating the real and the imaginary parts, we can compute the output signal of the two consecutive sequences.

06.06 Frequency analysis with DTFT and DFT

In order to compute the spectrum of a continuous-time or discrete-time signal, we need all the signal samples from $-\infty$ to $+\infty$. Nevertheless, in practice, a signal is always observed only within a finite temporal window. If the signal is analog, we first filter it with an anti-aliasing (lowpass) filter, and then sample it with a sampling frequency $F_s \geq 2B$, where B is the bandwidth of the signal, to obtain a discrete-time signal. The length of the discrete-time signal is then limited (i.e., truncated) to only L samples (i.e., to a window of TL seconds, with T being the sampling period and $T = 1/F_s$).
The finite-length interval limits the *frequency resolution*, i.e., it restricts our ability to distinguish between two frequencies with a frequency separation lower than $\frac{1}{LT}$.
Let us denote the infinite length sequence we want to analyze as $x(n)$. Limiting the sequence duration to L samples, i.e., to the interval $0 \leq n \leq L-1$, is equivalent to multiplying sample by sample $x(n)$ with a rectangular window function $w(n)$ of length L:

$$\hat{x}(n) = x(n) \cdot w(n)$$

$$w(n) = \begin{cases} 1 & 0 \leq n \leq L-1 \\ 0 & \text{otherwise.} \end{cases}$$

In this way the signal spectrum to be analyzed, $\hat{X}(e^{j\omega})$ is given by

$$\hat{X}(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{+\pi} X(e^{j\theta}) W(e^{j(\omega-\theta)}) \mathrm{d}\theta$$

Let us prove the last relation. We know that

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{+\pi} X(e^{j\theta}) e^{j\theta n} \mathrm{d}\theta$$

$$\text{DTFT}\left[x(n)w(n)\right] = \sum_{n=-\infty}^{+\infty} x(n)w(n)e^{-j\omega n} =$$

$$= \sum_{n=-\infty}^{+\infty} w(n) \frac{1}{2\pi} \int_{-\pi}^{+\pi} X(e^{j\theta}) e^{j\theta n} \mathrm{d}\theta e^{-j\omega n} =$$

$$= \frac{1}{2\pi} \int_{-\pi}^{+\pi} X(e^{j\theta}) \sum_{n=-\infty}^{+\infty} w(n) e^{-j(\omega-\theta)n} \mathrm{d}\theta =$$

$$= \frac{1}{2\pi} \int_{-\pi}^{+\pi} X(e^{j\theta}) W(e^{j(\omega-\theta)}) \mathrm{d}\theta$$

The last integral is also known as the *convolution integral*.

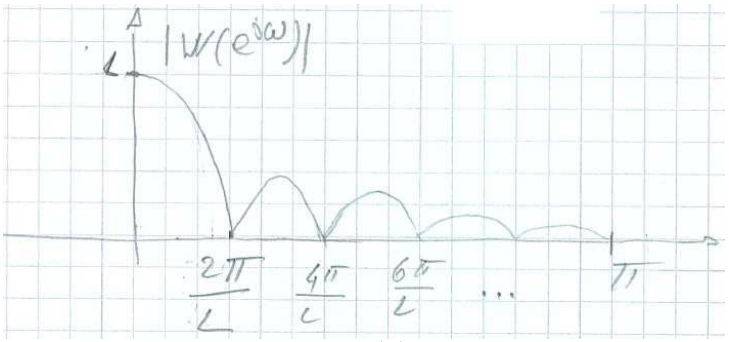
The spectrum of the rectangular function $W(e^{j\omega})$ is given by

$$\text{DTFT}\left[w(n)\right] = \sum_{n=0}^{L-1} e^{-j\omega n} =$$

$$= \frac{1 - e^{-j\omega L}}{1 - e^{-j\omega}} = \Big/ \cdot \frac{e^{j\omega\frac{L}{2}} e^{-j\omega\frac{L}{2}}}{e^{j\frac{\omega}{2}} e^{-j\frac{\omega}{2}}}$$

$$= \frac{e^{j\omega\frac{L}{2}} - e^{-j\omega\frac{L}{2}}}{e^{j\frac{\omega}{2}} - e^{j\frac{\omega}{2}}} \cdot e^{-j\omega\frac{L-1}{2}} =$$

$$= \frac{j\sin\left(\omega\frac{L}{2}\right)}{j\sin\left(\frac{\omega}{2}\right)} \cdot e^{-j\omega\frac{L-1}{2}}$$



This is a function similar to $\frac{\sin(x)}{x}$. In the convolution with $X(e^{j\omega})$, it has the effect of spreading the spectrum of $x(n)$ over all frequencies. This phenomenon is called *leakage*.
Let us consider an example:

$$x(n) = \cos(\omega_0 n) = \frac{1}{2}\left[e^{-j\omega_0 n} + e^{j\omega_0 n}\right].$$

For the frequency shift property:

$$\hat{X}(e^{j\omega}) = \frac{1}{2}\left[W(e^{j(\omega+\omega_0)}) + W(e^{j(\omega-\omega_0)})\right]$$

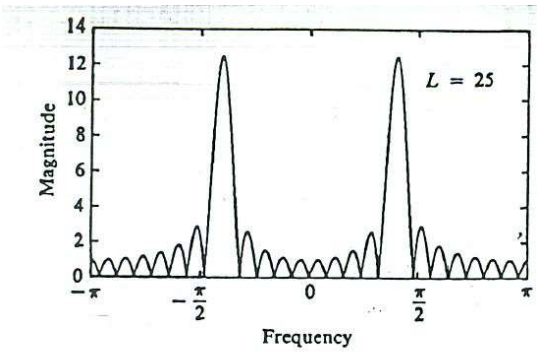


Figure 5.12 Magnitude spectrum for $L = 25$ and $n = 2048$, illustrating the occurrence of leakage.

We see that the spectrum, instead of being a line (a Dirac pulse at $\pm\omega_0$), forms a lobe whose width depends on the number of samples L that we are observing. Generally, the greater the value of L, the better the spectral resolution. For instance, consider

$$x(n) = \cos(\omega_1 n) + \cos(\omega_2 n)$$

with $\omega_1 \simeq \omega_2$,

$$\hat{X}(e^{j\omega}) = \frac{1}{2}\left[W(e^{j(\omega+\omega_1)}) + W(e^{j(\omega+\omega_2)}) + W(e^{j(\omega-\omega_1)}) + W(e^{j(\omega-\omega_2)})\right]$$

Only when $|\omega_1 - \omega_2| \geq \frac{2\pi}{L}$ (which is half of the lobe width), it is possible to distinguish two separate lobes.

Example:

$$x(n) = \cos(\omega_0 n) + \cos(\omega_1 n) + \cos(\omega_2 n)$$

with $\omega_0 = 0.2\pi$, $\omega_1 = 0.22\pi$, and $\omega_2 = 0.6\pi$

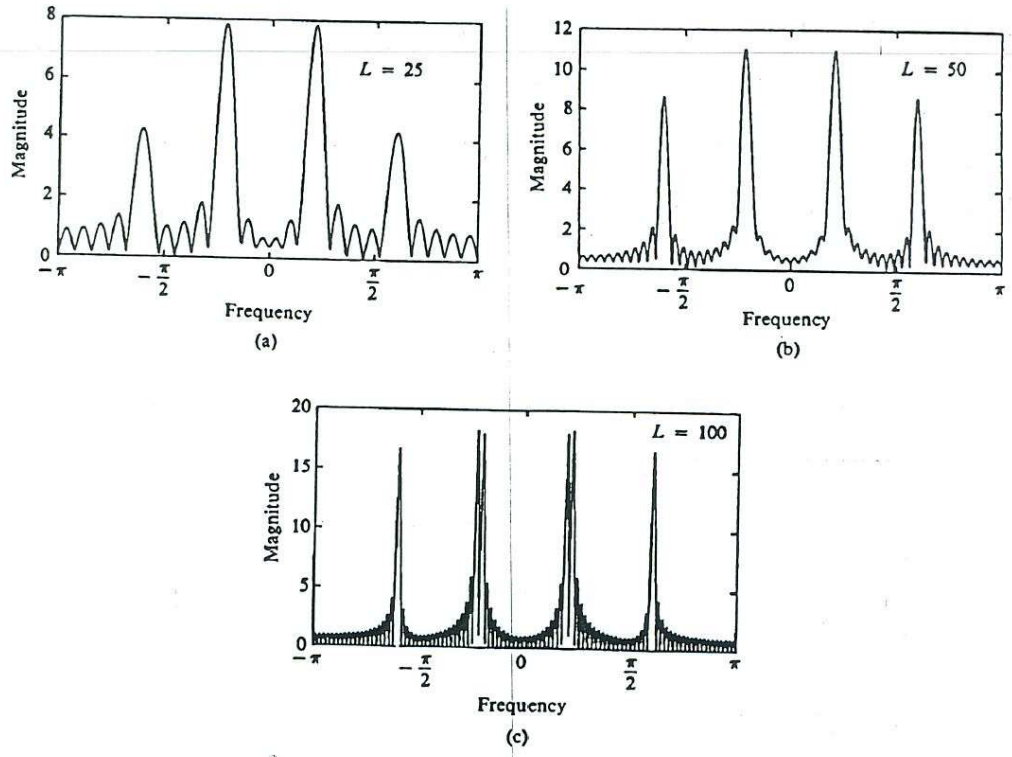


Figure 5.13 Magnitude spectrum for the signal given by (5.4.8), as observed through a rectangular window.

If we increase the observation window, i.e., L, we can reduce the width of the main lobe of $W(e^{j\omega})$, but the secondary lobes are not attenuated; that is, the leakage effect remains. To reduce these lobes and minimize leakage, we can use an appropriate window function $w(n)$

For example, a commonly used window function is the *Hann window* (or *Hanning window*), which is a raised cosine function:

$$w(n) = \begin{cases} \frac{1}{2}\left[1 - \cos\left(\frac{2\pi n}{L-1}\right)\right] & \text{for } 0 \leq n \leq L-1 \\ 0 & \text{otherwise,} \end{cases}$$

or the *Hamming window*:

$$w(n) = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{L-1}\right) & \text{for } 0 \leq n \leq L-1 \\ 0 & \text{otherwise.} \end{cases}$$

For these window functions, the secondary lobes are more attenuated than with the rectangular window. The reduction of the secondary lobes is obtained at the expense of an enlargement of the main lobe, i.e., at the expense of resolution loss (which can be compensated by increasing L). Since

$$\hat{X}(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{+\pi} X(e^{j\theta})W(e^{j(\omega-\theta)})\mathrm{d}\theta$$

if the spectrum of $w(n)$ is narrow compared to that of $x(n)$, the window function leads only to a negligible lowpass effect (a smoothing effect) on the spectrum $X(e^{j\omega})$. On the contrary, if $w(n)$ has a large main lobe (as in the case of small L), the spectrum of $w(n)$ will mask the details of $X(e^{j\omega})$.

Typically, the spectrum is evaluated with the DFT, which, for finite-length sequences of length L, coincides with the DTFT estimated at L uniformly spaced points on the interval $[0, 2\pi]$:

$$\text{DFT}[\hat{x}(n)] = \hat{x}(e^{j\omega})\Big|_{\omega=\frac{2\pi}{L}k}$$

By extending $\hat{x}(n)$ with $N - L$ zeros (i.e., by computing the DFT on N points), we can increase the resolution in estimating $\hat{X}(e^{j\omega})$ as much as we desire. Nevertheless, it's important to note that the DFT provides samples of $\hat{X}(e^{j\omega})$, and increasing N does not result in a better estimation of $X(e^{j\omega})$. The frequency resolution in the estimation of $X(e^{j\omega})$ depends only on the length of the observation window L.

Window functions used in practice and their spectra

TABLE 8.1 WINDOW FUNCTIONS FOR FIR FILTER DESIGN

Name of window	Time-domain sequence, $h(n), 0 \le n \le M-1$
Bartlett (triangular)	$1 - \dfrac{2\left\lvert n - \frac{M-1}{2}\right\rvert}{M-1}$
Blackman	$0.42 - 0.5\cos\dfrac{2\pi n}{M-1} + 0.08\cos\dfrac{4\pi n}{M-1}$
Hamming	$0.54 - 0.46\cos\dfrac{2\pi n}{M-1}$
Hanning	$\dfrac{1}{2}\left(1 - \cos\dfrac{2\pi n}{M-1}\right)$
Kaiser	$\dfrac{I_0\left[\alpha\sqrt{\left(\frac{M-1}{2}\right)^2 - \left(n - \frac{M-1}{2}\right)^2}\right]}{I_0\left[\alpha\left(\frac{M-1}{2}\right)\right]}$
Lanczos	$\left\{\dfrac{\sin\left[2\pi\left(n - \frac{M-1}{2}\right)\Big/(M-1)\right]}{2\pi\left(n - \frac{M-1}{2}\right)\Big/\left(\frac{M-1}{2}\right)}\right\}^L \quad L > 0$
	$1, \left\lvert n - \dfrac{M-1}{2}\right\rvert \le \alpha\dfrac{M-1}{2} \quad 0 < \alpha < 1$
Tukey	$\dfrac{1}{2}\left[1 + \cos\left(\dfrac{n - (1+\alpha)(M-1)/2}{(1-\alpha)(M-1)/2}\pi\right)\right]$
	$\alpha(M-1)/2 \le \left\lvert n - \dfrac{M-1}{2}\right\rvert \le \dfrac{M-1}{2}$

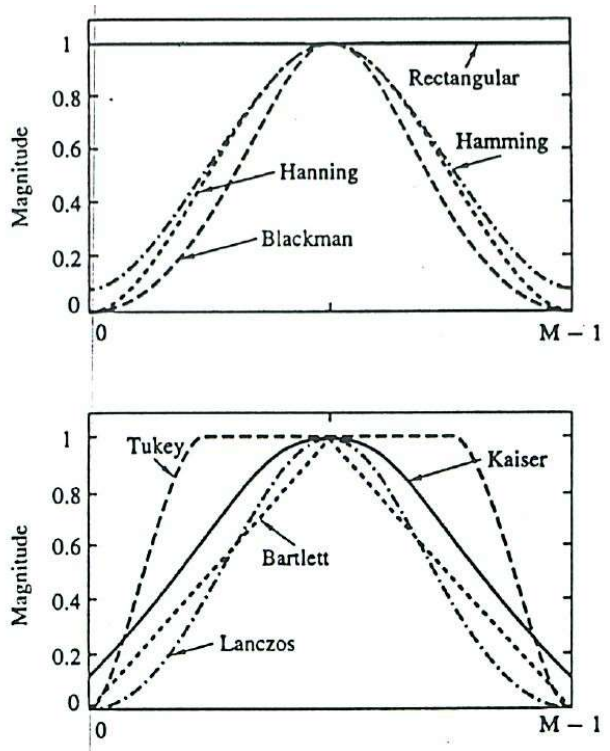


Figure 8.5 Shapes of several window functions.

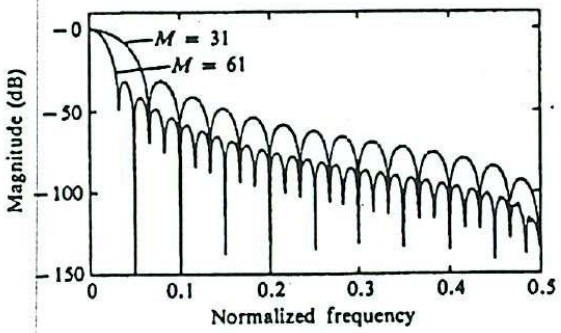


Figure 8.6 Frequency responses of Hanning window for (a) $M = 31$ and (b) $M = 61$.

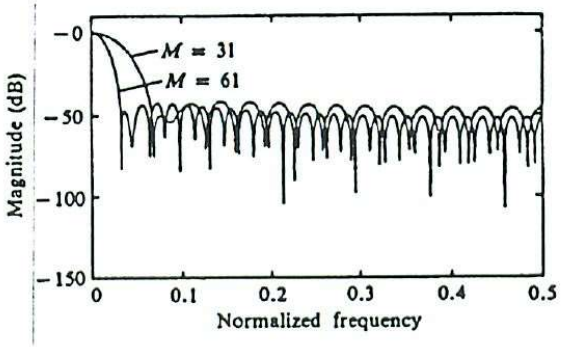


Figure 8.7 Frequency responses for Hamming window for (a) $M = 31$ and (b) $M = 61$.

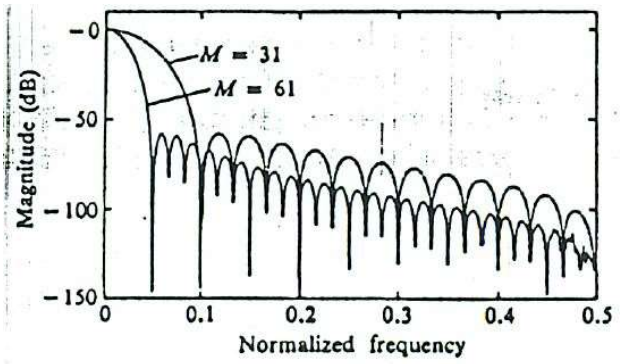


Figure 8.8 Frequency responses for Blackman window for (a) $M = 31$ and (b) $M = 61$.

TABLE 8.2 IMPORTANT FREQUENCY-DOMAIN CHARACTERISTICS OF SOME WINDOW FUNCTIONS

Type of window	Approximate transition width of main lobe	Peak sidelobe (dB)
Rectangular	$4\pi/M$	−13
Bartlett	$8\pi/M$	−27
Hanning	$8\pi/M$	−32
Hamming	$8\pi/M$	−43
Blackman	$12\pi/M$	−58

06.07 The Short-Time Fourier Transform - STFT

The Short-Time Fourier Transform (STFT) is a commonly used tool for the analysis, modification, and synthesis of signals with time-varying characteristics. It is often employed in speech and audio processing. Given an input signal $x(n)$, data segments are extracted at regular intervals using a time-limited window $w(m)$. The signal segments or *frames* can be expressed as

$$x_l(m) = w(m)x(m + lL); \qquad 0 \leq m \leq N - 1,$$

where N is the window length, l is the frame index, L is the *hop size*, i.e., the spacing in samples between two consecutive frames, with $L \leq N$ in general. Thus, two consecutive frames may overlap over $L - N$ samples. The index m is the local time index, i.e., an index relative to the start of the sliding window, while the 'global' time index of $x_l(m)$ is

$$n = m + lL.$$

For each signal frame, the discrete Fourier transform is computed as follows:

$$X(k, l) = \sum_{m=0}^{N-1} x_l(m)e^{-j\frac{2\pi}{K}nk} =$$

$$= \sum_{m=0}^{N-1} w(m)x(m + lL)e^{-j\frac{2\pi}{K}nk},$$

where K is the DFT size, with $K \geq N$ (performing the DFT on a larger number of points than the window size can enhance spectrum visualization or account for processing at a later stage). The STFT $X(k, l)$ characterizes the local time-frequency behavior of the signal around time lL and bin k. For a continuous sampling rate F_S, the discrete indexes correspond to the continuous time lL/F_S and frequency kF_S/K.

The STFT can be thought of as the spectral representation of a time slice of the input signal. By interpreting $X(k, l)$ as a function of the frequency k for each value of the time index l, the STFT corresponds to a series of time-localized spectra. Alternatively, we can view the STFT as a function of time for each frequency. Interpreting $X(k, l)$ as a time series that is a function of l for each bin k, the STFT then corresponds to a filter bank that decomposes the input signal into subbands (with one subband for each bin). We will delve into filter banks later in the course. In any case, these two interpretations are depicted with respect to the time-frequency plane in the following figure.

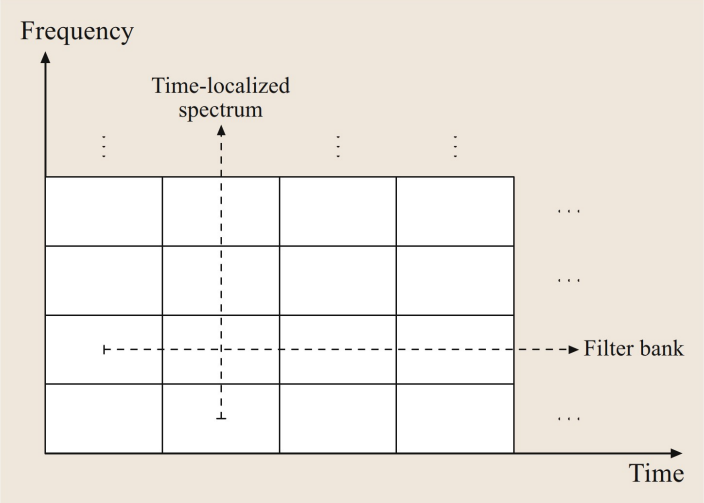


Fig. 12.1 Interpretations of the short-time Fourier transform as a series of time-localized spectra (*vertical*) and as a bank of bandpass filters (*horizontal*)

(Figure taken from Benesty, Jacob, M. Mohan Sondhi, and Yiteng Huang, eds. Springer handbook of speech processing. Berlin: Springer, 2008.)

Note that we can increase the spectral resolution by extending the length of the window N, but this leads to lower resolution in time. Conversely, we can enhance time resolution by reducing the length of the window N, but it results in lower resolution in frequency. Therefore, a compromise must be reached between the two requirements of achieving high time resolution or high frequency resolution.

The STFT is an analysis tool: it provides a representation of the signal and can reveal information about the signal. Very often, the squared magnitude of the STFT, $|X(k,l)|^2$ is visually represented using an image called a *spectrogram*. The X-axis of the spectrogram corresponds to time, and the Y-axis corresponds to frequency, with $|X(k,l)|^2$ depicted using gray levels or false colors:

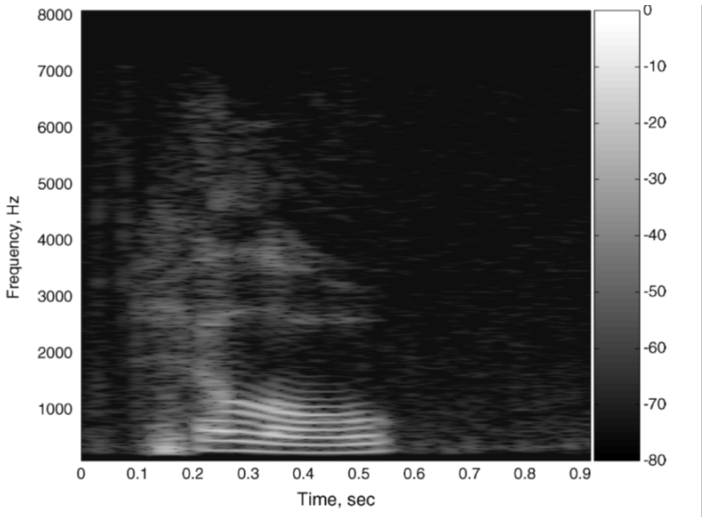
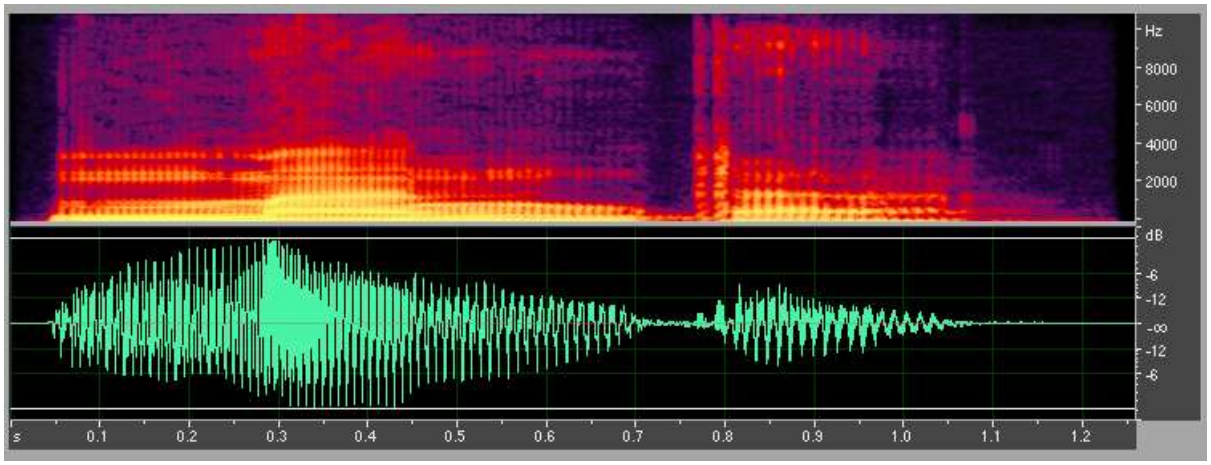


Figure 2.16 Spectrogram of English word "hello"

(Figure taken from Ian Vince McLoughlin, "Speech and Audio Processing"- Cambridge Univesity Press, 2016)



Spectrogram of the word ”manta” (Figure taken from Wikipedia)

The STFT allows for useful modifications of the signal, some guided by the information captured by the STFT itself. This includes techniques such as (i) *speech enhancement*, which aims to improve the signal-to-noise ratio or the intelligibility of the speech signal; (ii) *time-scale modification*, which can be applied to alter the duration of a speech or audio signal without changing its character (i.e., the pitch), such as playing the signal faster or slower; (iii) *pitch modification* that can be employed to change the pitch without altering the time scale. In these scenarios, the STFT of the input signal is modified to achieve the desired effect. To generate the modified signal, an appropriate *synthesis operation* is needed. Ideally, such a synthesis operation should perfectly reconstruct the original signal if no STFT-domain modification is carried out. A synthesis procedure based on this *perfect reconstruction* property is described in the following.
The reconstruction operation is essentially the reverse of the analysis operation. First, the inverse Discrete Fourier Transform (IDFT) of each local spectrum is computed. Then, the resulting signal frames are aggregated to synthesize the signal. If the DFT size is sufficiently large ($K \geq N$), the IDFT simply returns the windowed signal segment:

$$\hat{x}_l(m) = IDFT\{X(k,l)\} = w(m)x(m+lL), \qquad 0 \leq m \leq N-1$$

considering the local time m. In the global time n, considering $m = n - lL$ we have

$$\hat{x}_l(n-lL) = w(n-lL)x(n) \qquad lL \leq n \leq lL+N-1$$

The output signal reconstruction can then be obtained by an overlap-add operation, possibly adopting a *synthesis window*. Denoting $v(n)$ as the synthesis window, the overlap-add reconstruction is given by

$$\hat{x}(n) = \sum_l v(n-lL)\hat{x}_l(n-lL) = \sum_l v(n-lL)w(n-lL)x(n)$$

To obtain the output signal $\hat{x}(n)$, each frame generated by the IDFT is weighted by the synthesis window $v(m)$ and added to the neighboring windows in the parts that overlap in time. Since $x(n)$ is not a function of l, we have

$$\hat{x}(n) = x(n)\sum_l v(n-lL)w(n-lL).$$

Perfect reconstruction is achieved if the analysis and synthesis windows satisfy the constraint

$$\sum_l v(n-lL)w(n-lL) = 1$$

In many cases, $v(n)$ is not specified, and the equivalent synthesis window is a rectangular window or length N. Then, the constraint becomes simply

$$\sum_l w(n-lL) = 1.$$

Several perfect reconstruction windows that satisfy this condition have been studied in the literature. For example, the rectangular or the triangular windows, and the Blackman-Harris family, which includes the Hann and Hamming windows. These are also referred to as *windows with the overlap-add property* and will be denoted by $w_{\mathrm{PR}}(n)$ in the following. It is worth noting that any window function satisfies the overlap-add property when $L = 1$; for $L = N$, the only window that has the overlap-add property is a rectangular window of length N; for $L > N$, there are time gaps between successive frames and no window can have the overlap-add property.

There are several methods to design analysis and synthesis windows that satisfy

$$\sum_l v(n-lL)w(n-lL) = 1.$$

The simplest and most common approach is that of considering

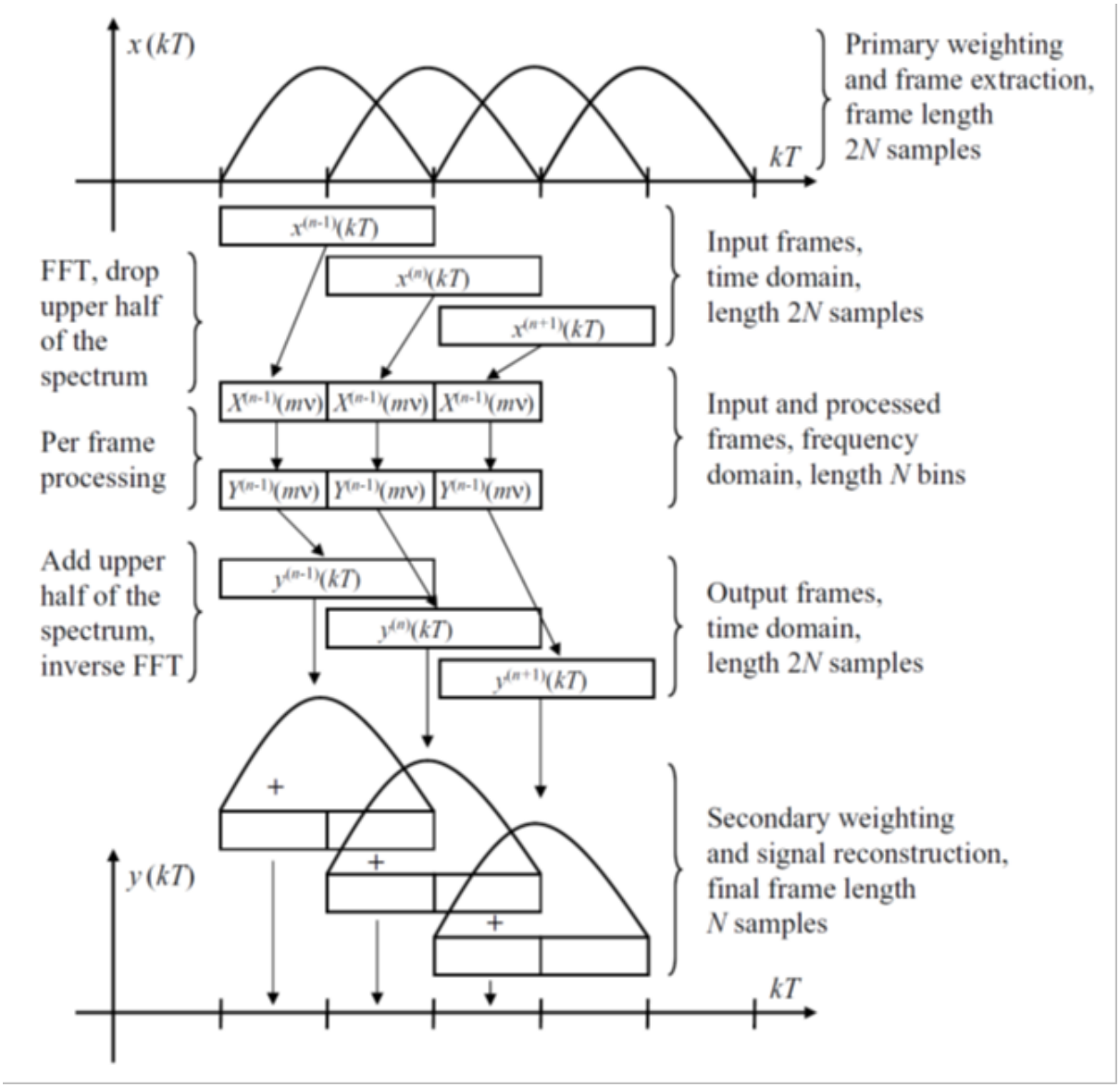
$$v(m) = w(m) = \sqrt{w_{\mathrm{PR}}(n)}.$$

Another approach involves using a perfect reconstruction window $w_{\mathrm{PR}}(m)$ and an arbitrary window $b(m)$, which is strictly nonzero over the time support of $w_{\mathrm{PR}}(m)$. In this case, $b(m)$ is employed as the analysis window, and

$$v(n) = \frac{w_{\mathrm{PR}}(m)}{b(n)}$$

is used as the synthesis window.

The following figure illustrates the operations involved in the analysis, processing, and synthesis using the STFT and the overlap-add method:



(Figure taken from Ian Vince McLoughlin, “Speech and Audio Processing”- Cambridge Univesity Press, 2016)

06.08 The Discrete Cosine Transform - DCT

The DFT is not the only transform that requires only multiplications and additions for its computation.
Let us consider first the expressions of the direct and inverse DFTs:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}nk}$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi}{N}nk}$$

These expressions can also be written as

$$X(k) = \sum_{n=0}^{N-1} x(n) f(n,k)$$

$$x(n) = \sum_{k=0}^{N-1} X(k) g(n,k)$$

where for the DFT it is

$$f(n,k) = e^{-j\frac{2\pi}{N}nk}$$

$$g(n,k) = \frac{e^{j\frac{2\pi}{N}nk}}{N}$$

By choosing different sequences $f(n,k)$ and $g(n,k)$, we obtain different transforms that still require only additions and multiplications. Obviously, in order to have a pair of reciprocal transforms, $g(n,k)$ and $f(n,k)$ must be linked in some way.
We have already discussed the matrix representation of the DFT If we define:

$$\mathbf{X} = \begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}$$

we have

$$\mathbf{X} = \mathbf{F} \cdot \mathbf{x} \qquad \text{and} \qquad \mathbf{x} = \mathbf{G} \cdot \mathbf{X}$$

where

$$\mathbf{F} = \begin{bmatrix} f(0,0) & f(1,0) & \dots & f(N-1,0) \\ f(0,1) & f(1,1) & \dots & f(N-1,1) \\ \vdots & \vdots & & \vdots \\ f(0,N-1) & f(1,N-1) & \dots & f(N-1,N-1) \end{bmatrix}$$

(because we consider $f(n,k)$ and we sum on n),

$$\mathbf{G} = \begin{bmatrix} g(0,0) & g(0,1) & \dots & g(0,N-1) \\ g(1,0) & g(1,1) & \dots & g(1,N-1) \\ \vdots & \vdots & & \vdots \\ g(N-1,0) & g(N-1,1) & \dots & g(N-1,N-1) \end{bmatrix}$$

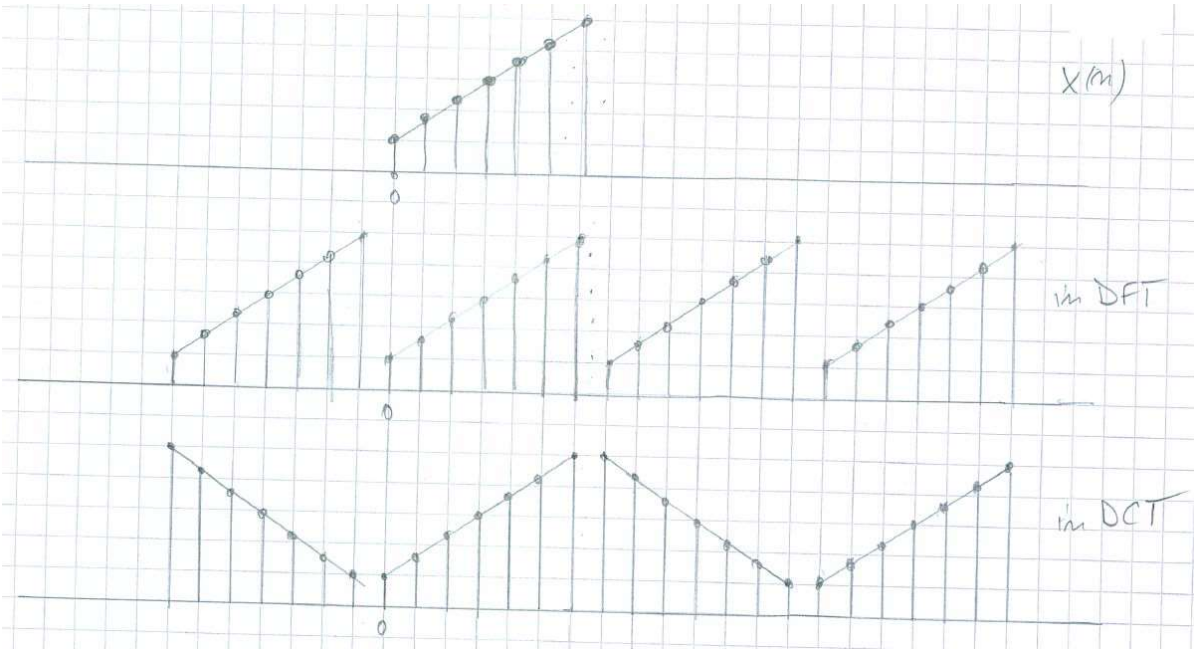
(because we consider $g(n,k)$ and we sum on k).
The two transforms are reciprocal if and only if

$$\mathbf{F}^{-1} = \mathbf{G}.$$

In the field of data compression, many different transforms of this family are considered. All these transforms differ in the choice of the sequences $f(n,k)$ and $g(n,k)$. With these transforms, we exploit a property of natural signals (and sequences): natural signals tend to concentrate most of their energy at low frequencies. Given a natural sequence (i.e., a sequence obtained by sampling a natural signal), we can compress this sequence (i.e., we can code this sequence with a reduced number of bits) by considering the DFT of the sequence and coding the low frequencies with an adequate number of bits while the other frequencies are coded with a reduced number of bits.
The DCT is a real transform for real sequences (i.e., for the DCT $f(n,k)$ and $g(n,k)$ are real) which offers two significant characteristics:

- it obtains a good compression of the energy in a few terms,
- it has fast computation algorithms (similar to the FFT).

In reality there are eight cosine transform. Here we will study the *DCT type 2* transform, which is the most used in practice both for image and audio compression (it is used in JPEG, MPEG, H.261, in MP3, i.e., in MPEG layer 3 audio coding).
The DCT is related to the DFT. Indeed, here we derive it from the DFT.
Given a sequence $x(n)$ of finite length N, we have seen that the DFT transform the sequence $x_p(n)$, obtained from the periodic repetition of $x(n)$ with period N. Also in the DCT a periodic extension of the sequence $x(n)$ is considered, but it is an even-order periodic extension. I.e.,



The sequence transformed by the DCT is more regular: we do not have those relevant steps as in the DFT. Thus, this transformation is able to compact the energy towards the low frequencies better than

the DFT and is more suitable for data compression. In the DCT (of type 2), the periodic repetition with period $2N$ of the sequence

$$y(n) = \begin{cases} x(n) & \text{for } 0 \leq n \leq N-1 \\ x(2N-1-n) & \text{for } N \leq n \leq 2N-1 \end{cases}$$

is considered.

Let us compute the DFT of this sequence:

$$Y(k) = \sum_{n=0}^{2N-1} y(n) W_{2N}^{nk} = \sum_{n=0}^{N-1} x(n) W_{2N}^{nk} + \sum_{n=N}^{2N-1} x(2N-1-n) W_{2N}^{nk}$$

In the second sum, consider the following variables change $m = 2N-1-n$, and thus $n = 2N-1-m$. For $n = N$, it is $m = N-1$, and for $n = 2N-1$, it is $m = 0$.

$$Y(k) = \sum_{n=0}^{N-1} x(n) W_{2N}^{nk} + \sum_{m=0}^{N-1} x(m) W_{2N}^{(2N-1-m)k} =$$

$$\{m = n\} = \sum_{n=0}^{N-1} x(n) W_{2N}^{nk} + \sum_{n=0}^{N-1} x(n) W_{2N}^{(-1-n)k} = \Big/ \cdot W_{2N}^{k/2} W_{2N}^{-k/2}$$

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

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

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

Thus,

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

with $0 \leq k \leq 2N-1$.

By definition, the Type 2 DCT is given by

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

with $0 \leq k \leq N-1$ (for $N \leq k \leq 2N-1$ we find the same samples).

Since $Y(k) = W_{2N}^{-k/2} C(k)$, we have $C(k) = W_{2N}^{k/2} Y(k)$. From this relation, we see that we can compute the samples $C(k)$ by building the sequence $y(n)$ and by computing its FFT. This is a fast algorithm, but for computing a real transform of a real sequence, it requires complex computations. There exist other algorithms for the fast computation of the DCT (similar to FFT) that require only real operations.

IDCT

For computing the inverse DCT, we exploit the IDFT of $Y(k)$. Let us consider

$$y(n) = \frac{1}{2N} \sum_{k=0}^{2N-1} Y(k) W_{2N}^{-nk} = \frac{1}{2N} \left\{ Y(0) + \sum_{k=1}^{N-1} Y(k) W_{2N}^{-nk} + Y(N) W_{2N}^{-nk} + \sum_{k=N+1}^{2N-1} Y(k) W_{2N}^{-nk} \right\}$$

In the last term, let us consider the change of variables $l = 2N - k$. For $k = N + 1$, it is $l = N - 1$. For $k = 2N - 1$, $l = 1$.

$$y(n) = \frac{1}{2N}\left\{Y(0) + \sum_{k=1}^{N-1} Y(k)W_{2N}^{-nk} + Y(N)W_{2N}^{-nk} + \sum_{l=1}^{N-1} Y(2N-l)W_{2N}^{-n(2N-l)}\right\}$$

Since we assume $y(n)$ to be real, for the properties of conjugate symmetry of $Y(k)$

$$Y(2N-k) = Y^*(k)$$

and the two sums are not independent.

$Y(0) = \sum_{n=0}^{2N-1} y(n)$ is the mean value of the real signal.

$Y(N) = \sum_{n=0}^{2N-1} y(n)W_{2N}^{-Nn} = 0$ because $W_{2N}^{Nn} = +1$ when n is even, and $W_{2N}^{Nn} = -1$ when n is odd. The sequence $y(n)$ is symmetric and for each $y(n)$ with n even, there is an identical term with n odd.

Thus,

$$y(n) = \frac{1}{2N}\left\{Y(0) + \sum_{k=1}^{N-1}\left[Y(k)W_{2N}^{-nk} + Y^*(k)(W_{2N}^{-nk})^*\right]\right\} =$$

$$= \frac{1}{2N}\left\{Y(0) + \sum_{k=1}^{N-1} 2\ \text{Re}\left[Y(k)W_{2N}^{-nk}\right]\right\}$$

Taking the first N terms we have

$$x(n) = \frac{1}{2N}\left\{Y(0) + \sum_{k=1}^{N-1} 2\ \text{Re}\left[Y(k)W_{2N}^{-nk}\right]\right\}$$

But $Y(k) = C(k)W_{2N}^{-k/2}$, thus

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

and the IDCT is

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

For more information study:

S. K. Mitra, "Digital Signal Processing: a computer based approach," 4th edition, McGraw-Hill, 2011

Chapters 5.2-5.3, pp. 201-211

Chapter 5.7, pp. 224-230

Chapter 11.3.2-4 pp. 621-632

Chapter 5.4 pp. 211-216

Chapter 5.7 pp. 226-228

Chapter 5.12 pp. 249-254

Benesty, Jacob, M. Mohan Sondhi, and Yiteng Huang, eds. Springer handbook of speech processing. Berlin: Springer, 2008

Chapter 12.1, pp. 230-232