00 A brief overview of the course

00.01 The course organization

Digital Signal and Image Processing is a 9 CFU course in the first year of the Laurea Degree in Computer Engineering. The course aims to introduce fundamental concepts of signals and images, present principles and methodologies of signal analysis and signal processing through discrete-time systems, and illustrate the main techniques for audio, image, and video signal compression.
The lecture notes posted on the e-learning platform are an expanded transcription of the face-to-face lessons at the University of Trieste. Like every course, these lectures have been prepared based on specific textbooks. At the end of each topic, you will find a reference to the book and chapter from which the lessons are derived. You are encouraged to read/study these book chapters to enhance your preparation.
In particular, the first part of the course introduces the fundamentals of Digital Signal Processing. The lessons primarily follow the content of the book:
Sanjit K. Mitra, "Digital Signal Processing: a computer based approach," 4th edition, McGraw-Hill, 2011. ISBN-10: 0071289461; ISBN-13: 978-0071289467

The second part of the course introduces the fundamentals of Digital Image Processing. The lessons are based on the content presented in the book:
Rafael C. Gonzalez and Richard E. Woods, "Digital Image Processing," 4th edition, Pearson, 2007. ISBN 10: 1-292-22304-9; ISBN 13: 978-1-292-22304-9

Many examples shown during the lessons will utilize MATLAB, employing resources from the Signal Processing Toolbox and the Image Processing Toolbox.
The final exam will be oral and will include open questions and exercises.

01 Signals and signal processing

The course involves the processing of signals and images through digital (numeric) methods, specifically, Digital Signal Processing (DSP).

A signal is a function of independent variables, such as time, distance, or position. For instance, voice and music signals represent air pressure as a function of time in a specific position in space. In the case of a black-and-white image, it represents light intensity as a function of two spatial coordinates. A TV video signal comprises a sequence of images, referred to as frames, and is a function of three variables: two spatial coordinates and time.

Most signals we encounter are generated by natural means (e.g., voice, music, images, seismic waves), but there are signals that are produced synthetically or generated by computers (e.g., musical synthesis, ...).

Signals carry information, and the purpose of digital signal processing is to extract the useful information carried by a signal.

Digital signal processing deals with the mathematical representation of signals, and the operations, algorithms, applied to signals in order to extract the information they carry. We will see that the signal representation can be done by basic functions in the original independent variable domain (the time domain), or can be performed by means of basic functions in a transformed domain. Similarly, digital signal processing can be carried out in the original independent variable domain (in the time domain) or in a transformed domain.

In our lectures, we will study both the representation and processing of signals using digital means.

01.01 Characterization and classification of signals

Signals can be classified based on the nature of the independent variables or the values of the function that defines the signal.

A signal can be *mono-dimensional* or *multi-dimensional* (1-D, 2-D, 3-D, ...). A mono-dimensional signal is a function of a single independent variable, while an M-dimensional signal is a function of M independent variables. For example, voice is a 1-D signal, a function of the independent variable 'time.' An image is a 2-D signal, a function of two spatial coordinates. A video is a 3-D signal, a function of three independent variables: two spatial coordinates and time.

A signal can also be classified as *mono-channel* or *multi-channel*. A mono-channel signal is a scalar function of the independent variable, whereas a multi-channel signal is a vector function of the independent variable. For example, a grayscale image is a scalar function of two spatial coordinates, represented as $I(x, y)$, while a color image is a vector function (with three components) of the spatial

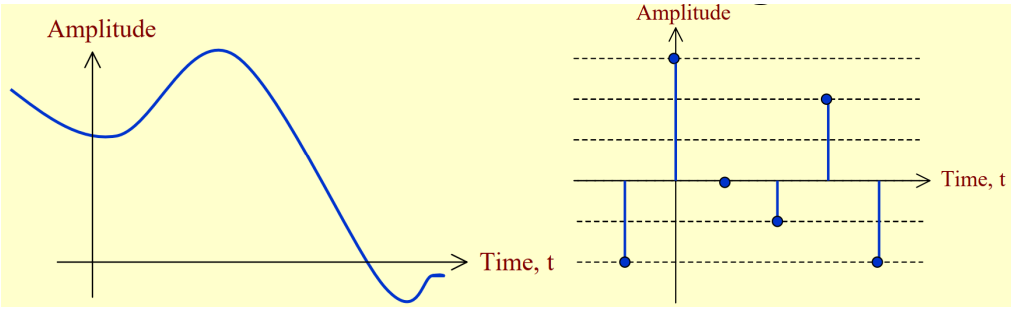
coordinates:

$$\begin{bmatrix} R(x,y) \\ G(x,y) \\ B(x,y) \end{bmatrix},$$

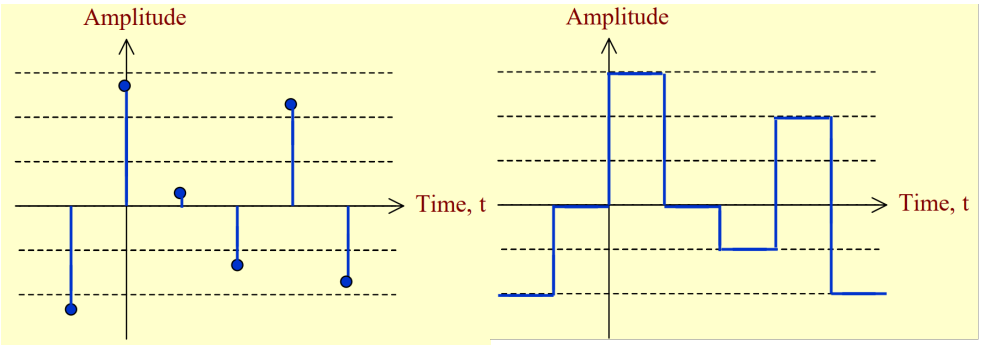
where R, G, and B represent the intensity of the three primary colors: red, green, and blue.
In the first part of the course, we will study only mono-dimensional and mono-channel signals.
Although the independent variable isn't necessarily time, it is a common practice to refer to it as *time* for these signals.
The signal value at a specific time instant is called *amplitude*.
The variation of amplitude as a function of the independent variable, i.e., as a function of time, is called *waveform*.
If the independent variable is continuous ($t \in \mathbb{R}$), the signal is called a *continuous-time signal*. If the independent variable is discrete ($t \in \mathbb{N}$ or $t \in \mathbb{Z}$), the signal is called a *discrete-time signal*.
A continuous-time signal is defined for each time instant. On the contrary, a discrete-time signal is defined only for certain time instants, while it does not exist between these time instants. A discrete time signal is a sequence of numbers.
Similarly, a signal can have a *continuous amplitude* or a *discrete amplitude*. A continuous amplitude signal can assume an infinite number of amplitude values (in $\mathbb{R}$ or $\mathbb{C}$). On the contrary, a signal with discrete amplitude can assume only a finite number of values (which can be coded with a finite number of binary digits).
A continuous-time signal with continuous amplitude is called an *analog signal*. Many natural signals are analog signals (e.g., voice, music, seismic waves).
A discrete-time signal with discrete amplitude is called a *digital signal*. Digital signals can be generated artificially, but there also exist natural digital signals (for example, the number of sunspots in one year or the number of cars passing on a road in the last month). Moreover, digital signals can be generated from analog signals through the following operations:

- sampling the signal,
- quantizing the samples,
- coding the quantized samples.

For example, the music stored in an audio CD is generated with this procedure.
A discrete-time signal with continuous amplitude is often referred to as a *sample-data signal*.
A continuous-time signal with discrete amplitude is commonly known as a *quantized boxcar signal*.
A signal can be classified as a *real signal* or a *complex signal* based on the amplitude of the signal, whether it belongs to the set of real numbers ($\mathbb{R}$) or the set of complex numbers ($\mathbb{C}$).
In the following, we will denote the independent variable of a continuous-time 1-D signal as t, and the independent variable of a discrete-time signal as n. We will treat continuous-time signals as continuous functions and discrete-time signals as sequences (of numbers). For example, $u_a(t)$ represents a 1-D continuous-time signal, and $\{v(n)\}$ represents a 1-D discrete-time signal. Each element $v(n)$ of the sequence $\{v(n)\}$ is called a *sample*. In many applications, a discrete-time signal is generated from a



(a) A continuous time signal (b) A digital signal



(c) A sample-data signal (d) A quantized boxcart signal

Figure 01.01: (From S. K. Mitra, "Digital signal processing: a computer based approach", McGraw Hill, 2011)

continuous-time signal by sampling the signal at uniformly spaced intervals T:

$$u_a(t) \rightarrow u_a(nT).$$

We will often represent the sampled signal $u_a(nT)$ simply as $u(n)$.

A final classification of signals arises from the way we know and treat them. We define a signal as *deterministic* when it is uniquely described by a mathematical expression, a look-up table, or a well-defined rule. For example,

$$x(n) = a^n.$$

We define a signal as *random* if it is generated in a random fashion and evolves in an unpredictable way. For most of the course, we will primarily focus on studying deterministic discrete-time signals.

Signals originating from the real world are typically analog signals, which means they have continuous-time and continuous amplitude. However, Digital Signal Processing focuses on digital signals, i.e., signals with discrete-time and discrete amplitude. For simplicity, we will consider these signals as discrete-time signals with continuous amplitude. Later in the course, we will study the effects of quantization on signals.

01.02 Digital signal processing: pros and cons

Processing a signal involves treating the signal with an appropriate system, such as a device, circuit, or algorithm implemented by a program on a computer or any other processor. Many of the signals found in nature are analog signals, and these signals can be directly processed by suitable analog systems. In this case, both the input and output signals of the analog system are analog:

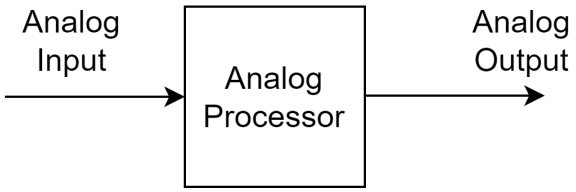


Figure 01.02: (From S. K. Mitra, "Digital signal processing: a computer based approach", McGraw Hill, 2011)

Digital signal processing offers an alternative method for processing analog signals through three fundamental steps:

1. Analog to Digital (A/D) conversion.
2. Digital processing.
3. Digital to Analog (D/A) conversion.

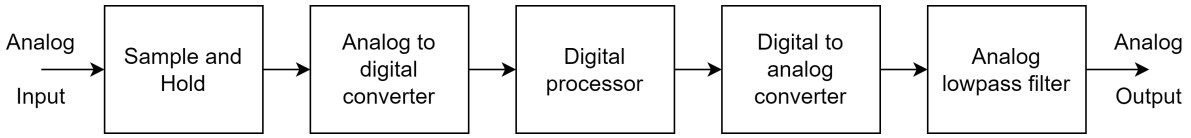


Figure 01.03: (From S. K. Mitra, "Digital signal processing: a computer based approach", McGraw Hill, 2011)

The input signal undergoes processing through a Sample-and-Hold circuit, which samples the signal and maintains its value constant between samples, facilitating analog-to-digital conversion (this operation is often considered part of the A/D converter). A/D conversion involves quantizing and coding the samples. The coded samples can be numerically processed by a digital processor, such as a computer, a processor with architecture specialized for digital signal processing (a Digital Signal Processor), or a digital electronic circuit. Subsequently, the output signal of the digital processor is converted back to analog form with a D/A converter, typically generating an analog signal formed with continuous steps. An analog lowpass filter is employed to eliminate undesired high-frequency components (i.e., the steps) and generate the desired analog output signal. Refer to Fig. 01.04.
Conceptually, the entire process appears more complex than analog signal processing. Why do we prefer to process analog signals digitally?

Pros of digital signal processing

1. Digital circuits are less sensitive to component tolerances and are independent of temperature and aging effects on components. They can be produced in large volumes without the need for

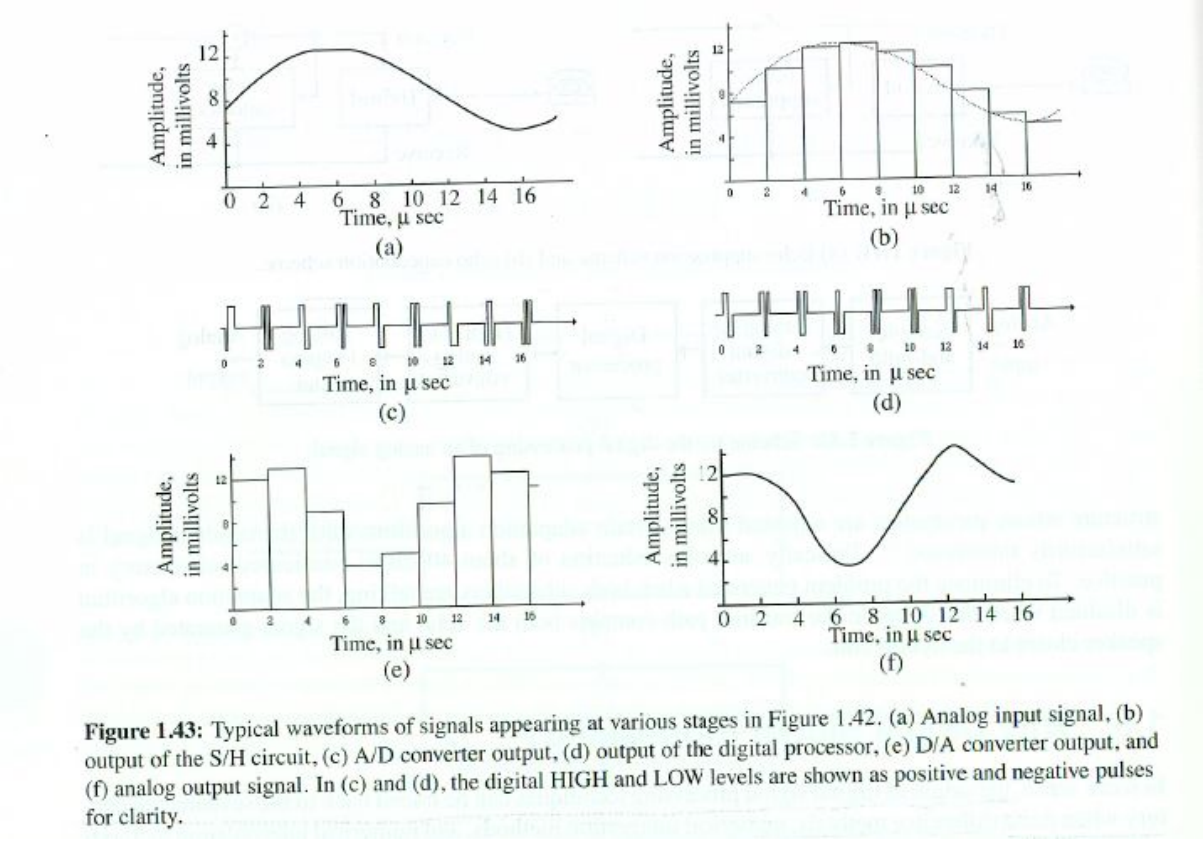


Figure 1.43: Typical waveforms of signals appearing at various stages in Figure 1.42. (a) Analog input signal, (b) output of the S/H circuit, (c) A/D converter output, (d) output of the digital processor, (e) D/A converter output, and (f) analog output signal. In (c) and (d), the digital HIGH and LOW levels are shown as positive and negative pulses for clarity.

Figure 01.04: (From S. K. Mitra, “Digital signal processing: a computer based approach”, McGraw Hill, 2011)

any adjustments during construction or later in use. They can be manufactured in the form of VLSI or ULSI circuits, enabling the implementation of very complex systems on a single chip.

2. Numeric filtering enables improved robustness and accuracy. By increasing the precision of operations, it becomes possible to arbitrarily reduce the noise introduced by the system.

3. Digital systems allow the implementation of much more complex signal processing algorithms than those of analog systems.

4. It becomes possible to have a software implementation of the system, which allows for easy reconfiguration of the digital processor functions (simply by changing the program). The same processor can process different signals through time-sharing of the processor.

5. It is easy to design systems that automatically adapt the internal parameters to the specific signal, known as adaptive filters.

6. Digital signals can be stored almost indefinitely without any loss of information.

7. Digital systems generally have a much lower economic cost than analog systems that implement the same operations.

Cons of digital signal processing

1. The system complexity is higher due to the pre- and post-processing operations (Sample-and-hold, A/D, D/A, lowpass filtering).
2. There are some limitations due to the speed of A/D and D/A converters and digital signal processors.
3. Digital signal processors necessarily use active devices that consume electric power.
4. There are some undesired effects originating from the use of algorithms implemented with finite-precision arithmetic.

Applications of digital signal processing

- Voice (voice coding, voice recognition, voice synthesis)
- Music (coding, filtering, equalization, audio effects, music transcription, synthesis)
- Images and Video (coding, filtering, pattern recognition, ...)
- Communications (modulation, demodulation, equalization, ...)
- Medicine (Electrocardiography, Electroencephalogram, Echography, TAC, ...)
- Seismology (seismic data analysis, petrol research, ...)
- Radar and sonar
- Vibration control systems in buildings or engines
- Automated guidance systems
- Economy and finance
- Big data
- ...

For more information study:

S. K. Mitra, ”Digital Signal Processing: a computer based approach,” 4th edition, McGraw-Hill, 2011
Chapter 1.1, pp. 1-3
Chapter 1.5, pp. 37-40

02 Discrete-time signals in the time domain

02.01 Time domain representation

In digital signal processing, signals are sequences of numbers (called samples) function of an independent variable (called time), which is an integer in the interval $[-\infty, +\infty]$.

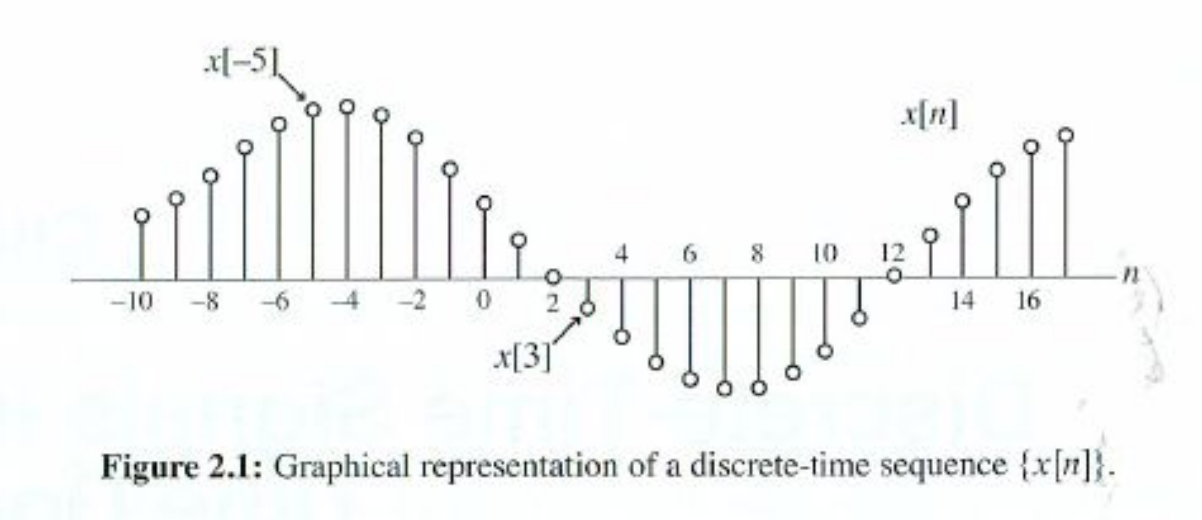


Figure 2.1: Graphical representation of a discrete-time sequence $\{x[n]\}$.

Figure 02.01: (From S. K. Mitra, "Digital signal processing: a computer based approach", McGraw Hill, 2011)

In the following, we will denote the generic sequence as $\{x(n)\}$, where $x(n)$ represents the sample of the sequence at time n [1]. Note that the sequence $x(n)$ is defined exclusively for integer values of n and *does not exist* for non-integer values of n.
We will represent or define a sequence through the use of

- a mathematical law:

$$\{x(n)\} = e^{|n|}$$

$$\{x(n)\} = \begin{cases} 2 & n = 0 \\ 1 & n \neq 0 \end{cases}$$

- a sequence of numbers between $\{\ \ \}$:

$$\{x(n)\} = \quad \{\ldots, 0.95, -0.2, \underset{\uparrow}{2.1}, 1.2, -3.2, \ldots\}$$

where the arrow denotes the element at $n = 0$, with elements to the left of the arrow corresponding to $n < 0$, and elements to the right corresponding to $n > 0$.

[1] Later, when there will be no ambiguity, we will directly represent our sequence as $x(n)$.

The sequence $\{x(n)\}$ is commonly generated by sampling a continuous-time signal $x_a(t)$ (an analog signal) at uniformly spaced intervals:

$$x(n) = x_a(t)\Big|_{t=nT} = x_a(nT).$$

The interval time T that separates two consecutive samples is referred to as the *sampling period*. Its reciprocal is known as the *sampling frequency*:

$$F_T = \frac{1}{T}.$$

The sampling period is measured in seconds (s) and has the physical dimension of time. The sampling frequency is measured in cycles per second, denoted as Hertz (Hz).
In either scenario, whether the sequence $\{x(n)\}$ is derived from sampling a continuous-time signal or generated through alternative methods, $x(n)$ is referred to as the n-th sample of the sequence.
If the sequence takes only real values, it is referred to as a *real* sequence.
If the sequence assumes complex values, it is called a *complex* sequence.

Length of a discrete-time sequence
Discrete-time signals, i.e., sequences, possess either finite or infinite length.
A finite-length sequence is defined only within the interval

$$N_1 \leq n \leq N_2$$

where $-\infty < N_1 \leq N_2 < +\infty$, and the sequence has length (or duration):

$$N = N_2 - N_1 + 1.$$

A sequence of length N comprises only N samples. It can be transformed into an infinite-length sequence by assigning 0 values outside the $[N_1, N_2]$ interval. This operation is known as zero-padding.
There are three types of infinite-length sequences:

- *Causal* sequences, when $x(n) = 0 \; \forall n < 0$. (The sequence has non-zero element only for $n \geq 0$).
- *Anti-causal* sequences, when $x(n) = 0 \; \forall n > 0$.
- *Two-sided* sequences, with non-zero elements both for $n < 0$ and $n \geq 0$.

In the following, we will frequently examine finite-length causal sequences, which are defined solely in the interval $[0, N-1]$.

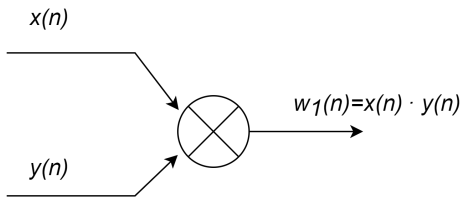
02.02 Operations on sequences

Given two sequences $x(n)$ and $y(n)$ we define the following operations:

- The *product* of two sequences:

$$w_1(n) = x(n) \cdot y(n),$$

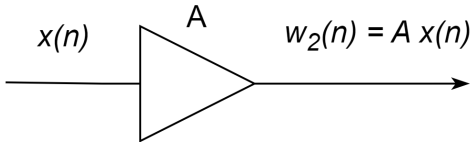
whose samples are given by the product of the corresponding samples of $x(n)$ and $y(n)$. This operation is also called *modulation*.



- The *scalar multiplication* of one sequence for a constant A:

$$w_2(n) = Ax(n)$$

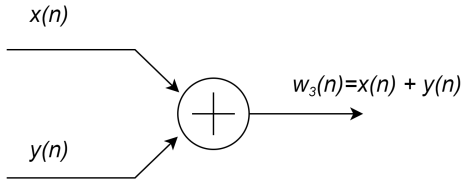
where each sample of $x(n)$ is multiplied for the scalar constant A.



- The *addition* of two sequences:

$$w_3(n) = x(n) + y(n),$$

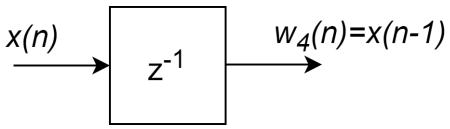
whose samples are given by the addition of the corresponding samples of $x(n)$ and $y(n)$.



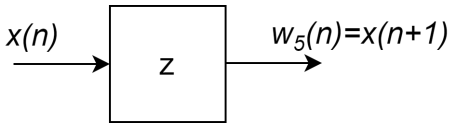
- The *time-shift* :

$$w_4(n) = x(n - N).$$

If $N > 0$, we say that the sequence has been delayed by N samples.
If $N < 0$, we say that the sequence has been time advanced of $|N|$ samples.
In systems processing real-time natural signals, it is possible to delay the signal, but time advancement is not feasible. Conversely, recorded signals can be time-shifted in both directions.
Unit delay:



Unit advance:



For examples of delayed signals, see Figure 02.02.

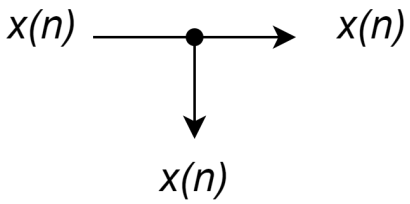
- The *time-reversal* or *folding* operation:

$$w_5(n) = x(-n)$$

which is the time-reversed version of the sequence.

For folding examples, see Figure 02.03.

In the block schemes we will study, there is another operation known as the *pick-off node*, which, when applied to a sequence, generates two other identical sequences.



Most digital signal processing systems are implemented using just these six operations on sequences

---

02.03 Classification of sequences

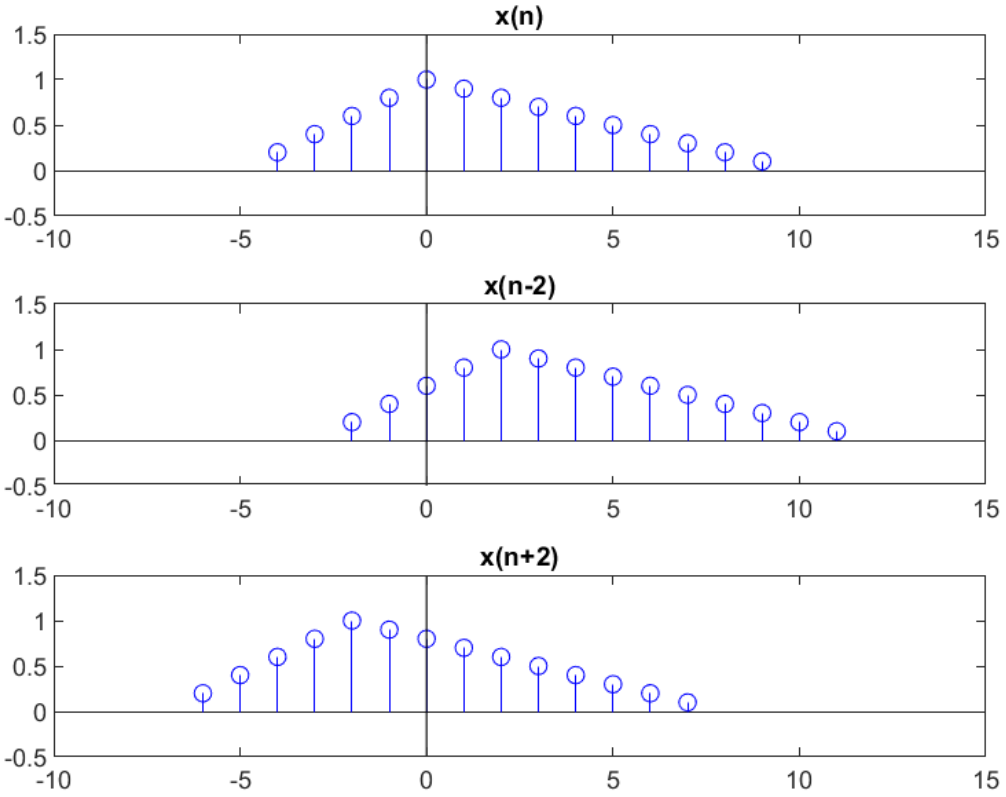
Sequences can be classified according to their symmetry properties:

- A real signal is called *symmetric* or *even* if:

$$x(n) = x(-n)$$

- A real signal is called *anti-symmetric* or *odd* if:

$$x(n) = -x(-n)$$



$$\{x(n)\} = \quad \{\ldots, -3, -2, -1, \underset{\uparrow}{0}, 2, 4, \ldots\}$$

$$\{x(n-2)\} = \quad \{\ldots, -3, -2, \underset{\uparrow}{-1}, 0, 2, 4, \ldots\}$$

$$\{x(n+2)\} = \quad \{\ldots, -3, -2, -1, 0, 2, \underset{\uparrow}{4}, \ldots\}$$

Figure 02.02: Delayed or time advanced sequences

- A real signal can be decomposed in the addition of an even and an odd signal:

$$x(n) = x_{\text{ev}}(n) + x_{\text{od}}(n)$$

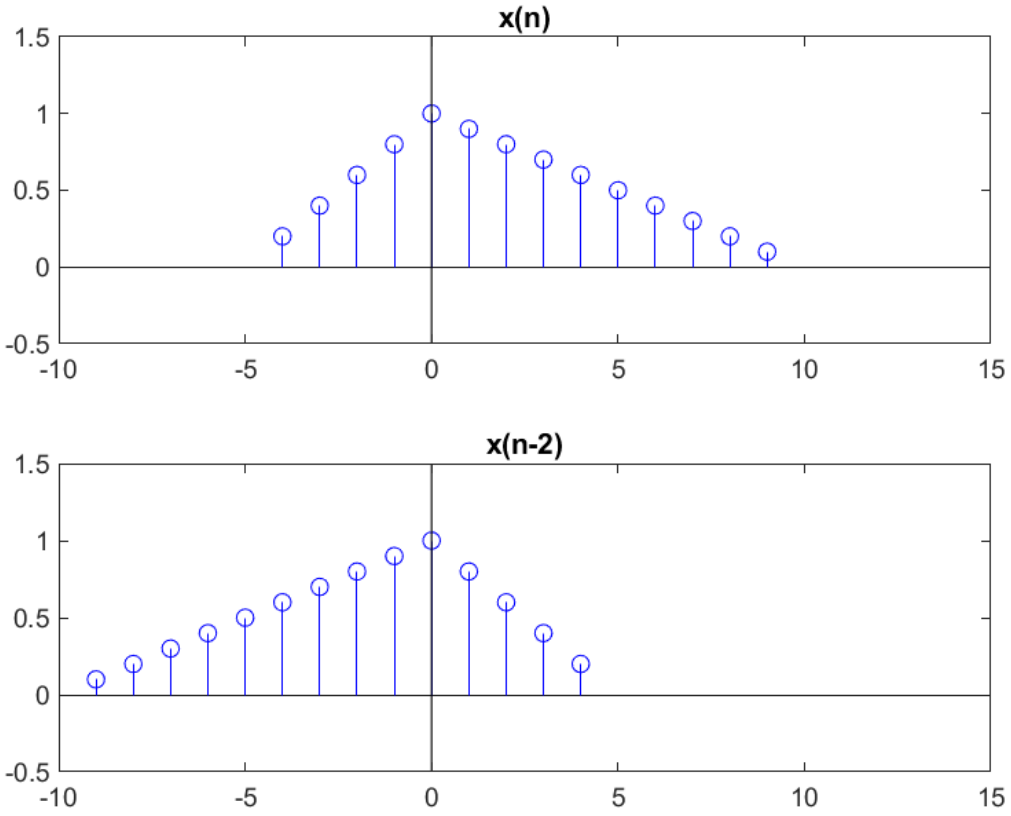
$$x_{\text{ev}}(n) = \frac{1}{2}\big[x(n) + x(-n)\big]$$

$$x_{\text{od}}(n) = \frac{1}{2}\big[x(n) - x(-n)\big]$$

- A complex signal is called *conjugate-symmetric* if

$$x(n) = x^*(-n),$$

  which means that the real part of $x(n)$ is even and the imaginary part is odd.



$$\{x(n)\} = \underset{\uparrow}{\{\ldots, -3, -2, -1, 0, 2, 4, \ldots\}}$$

$$\{x(-n)\} = \underset{\uparrow}{\{\ldots, 4, 2, 0, -1, -2, -3, \ldots\}}$$

Figure 02.03: Time reversed sequences

- A complex signal is called *conjugate-antisymmetric* if

$$x(n) = -x^*(-n),$$

  which means that the real part of $x(n)$ is odd and the imaginary part is even.

- A complex signal can be decomposed in the addition of a conjugate-symmetric and a conjugate-antisymmetric signal:

$$x(n) = x_{\text{cs}}(n) + x_{\text{ca}}(n)$$

$$x_{\text{cs}}(n) = \frac{1}{2}\big[x(n) + x^*(-n)\big]$$

$$x_{\text{ca}}(n) = \frac{1}{2}\big[x(n) - x^*(-n)\big]$$

Sequences can also be classified according to their periodicity or aperiodicity.

- A sequence such that $x_p(n) = x_p(n + kN)$ for all n, with $N \in \mathbb{N}$, $N > 0$, and $k \in \mathbb{Z}$, is called a *periodic sequence* with period N.
  The smallest $N > 0$ for which $x_p(n) = x_p(n + kN)$ is called *fundamental period* of the sequence.
  A sequence that is not periodic is called *aperiodic*.

Another classification comes from the energy and power content of the signals.

- The energy E_x of a signal $x(n)$ is:
$$E_x = \sum_{n=-\infty}^{+\infty} \left|x(n)\right|^2.$$
  This definition applies both to real and to complex sequences.
  A finite length sequence has always finite energy. An infinite length sequence can have finite or infinite energy.
  For example, the sequence
$$x_1(n) = \begin{cases} \dfrac{1}{n} & n \geq 1 \\ 0 & n \leq 0 \end{cases}$$
  has energy $E_x = \displaystyle\sum_{n=1}^{+\infty} \left(\frac{1}{n}\right)^2 = \frac{\pi^2}{6}$. On the contrary, the sequence
$$x_2(n) = \begin{cases} \dfrac{1}{\sqrt{n}} & n \geq 1 \\ 0 & n \leq 0 \end{cases}$$
  has energy $E_x = \displaystyle\sum_{n=1}^{+\infty} \left(\frac{1}{n}\right) = +\infty$.
- The *average power* of an aperiodic signal is:
$$P_x = \lim_{K \to +\infty} \frac{1}{2K+1} \sum_{n=-K}^{K} \left|x(n)\right|^2.$$
  The average power can be related to the energy by defining the energy in the interval $[-K, K]$:
$$E_{x,K} = \sum_{n=-K}^{K} \left|x(n)\right|^2,$$
$$P_x = \lim_{K \to +\infty} \frac{1}{2K+1} E_{x,K}.$$
  From this relation we see that a signal with fixed energy has zero average power.
  The average power of an infinite-length sequence can be finite or infinite.
  For example, the signal $x(n) = a$ for all n has average power $P_x = a^2$.
  The average power of a periodic signal $x_p(n)$ of period N is
$$P_x = \frac{1}{N} \sum_{n=0}^{N-1} \left|x_p(n)\right|^2$$
  A signal with finite energy is called an *energy signal*.
  A signal with finite average power is called a *power signal*.

Other classifications that we will encounter in the upcoming lessons include the following:

- A sequence is called *bounded* if there exists a constant B_x such that

$$\big|x(n)\big| \leq B_x \qquad \forall n.$$

- A sequence is called *absolutely summable* if

$$\sum_{n=-\infty}^{+\infty} \big|x(n)\big| < +\infty.$$

- A sequence is called *square-summable* if

$$\sum_{n=-\infty}^{+\infty} \big|x(n)\big|^2 < +\infty.$$

  An example of a sequence that is square-summable but not absolutely summable is the *sinc* sequence:

$$x(n) = \begin{cases} \dfrac{\sin[\omega_c n]}{\omega_c \pi n} & n \neq 0 \\ \dfrac{\omega_c}{\pi} & n = 0 \end{cases}$$

02.04 Basic sequences

- The *unit sample* sequence $\delta(n)$, also called *discrete-time impulse* or *unit impulse*, is defined by

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

  Thus,

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

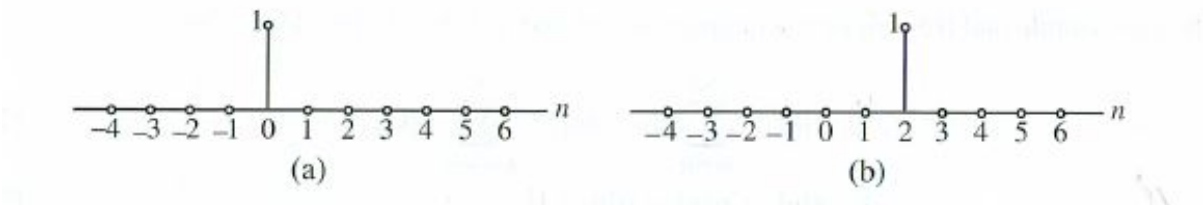


Figure 2.20: (a) The unit sample sequence $\{\delta[n]\}$ and (b) the shifted unit sample sequence $\{\delta[n-2]\}$.

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

Any sequence can be represented as the sum of infinite unit impulses, each shifted in time and appropriately weighted.
For example,

$$\begin{array}{cc} \{\ldots, 0.95, -0.2, 1.2, -3.2, 1.4 \ldots\} & = \ldots 0.95 \cdot \{\delta(n+2)\} - 0.2 \cdot \{\delta(n+1)\} + 1.2 \cdot \{\delta(n)\} \\ \uparrow & -3.2 \cdot \{\delta(n-1)\} + 1.4\{\delta(n-2)\} + \ldots \end{array}$$

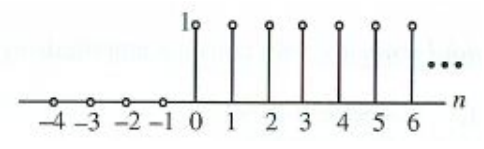
As a general rule, we have

$$\{a(n)\} = \sum_{m=-\infty}^{+\infty} a(m)\{\delta(n-m)\}$$

where $\delta(n-m)$ are the time-shifted unit impulses and $a(m)$ are the corresponding weights. In what follows we will neglect the curly brackets of the formula.

- The *unit step* sequence is defined by

$$\mu(n) = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases} .$$



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

Note that:

$$\mu(n) = \sum_{m=0}^{+\infty} \delta(n-m)$$

$$\delta(n) = \mu(n) - \mu(n-1)$$

- The real *sinusoidal* sequence is defined by

$$x(n) = A\cos\left(\omega_0 n + \phi\right)$$

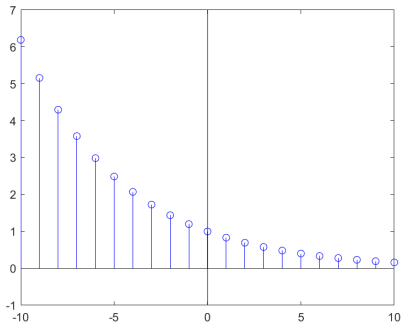
$$= A\cos\left(2\pi f_0 n + \phi\right)$$

$\omega_0 = 2\pi f_0$ is called *normalized angular frequency* or simply angular frequency.
f_0 is called *normalized frequency* or simply frequency.
ϕ is called *initial phase*.
A is the *amplitude* of the sinusoidal signal.

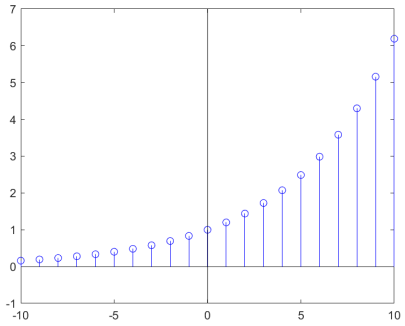
- The real *exponential* sequence is defined by

$$x(n) = Aa^n \qquad A, a \in \mathbb{R}$$

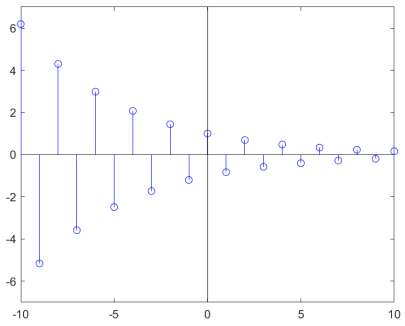
If $0 < a < 1$, it is an exponentially decreasing sequence.



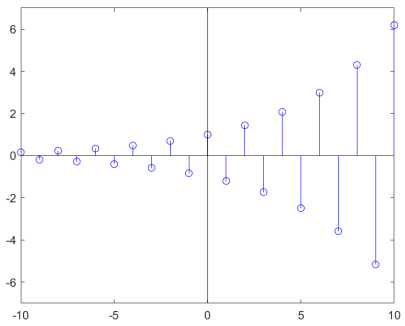
If $a > 1$, it is an exponentially increasing sequence.



If $-1 < a < 0$, it is an alternated exponentially decreasing sequence.



If $a < -1$, it is an alternated exponentially increasing sequence.



- The *complex exponential sequence* is defined by

$$x(n) = Aa^n \qquad a = r \cdot e^{j\omega_0} = e^{\sigma_0 + j\omega_0}$$

with $A, a \in \mathbb{C}$.
Since $A = \big|A\big| \cdot e^{j\phi}$, we can also write:

$$x(n) = \big|A\big| \cdot e^{\sigma_0 n} \cdot e^{j\left(\omega_0 n + \phi\right)}$$

$$= \big|A\big| e^{\sigma_0 n} \Big[\cos\left(\omega_0 n + \phi\right) + j \sin\left(\omega_0 n + \phi\right) \Big].$$

The real and imaginary parts of the complex exponential sequence are sinusoids with amplitude that increase or decrease exponentially.
A notable special case of the complex exponential sequence is the *generalized sinusoidal sequence* defined by

$$x(n) = e^{j\left(\omega_0 n + \phi\right)} = \cos\left(\omega_0 n + \phi\right) + j \sin\left(\omega_0 n + \phi\right).$$

- *Property:* A sinusoidal (or generalized sinusoidal) sequence is periodic if and only if the normalized frequency f_0 is a rational number, i.e., $f_0 \in \mathbb{Q}$.
Proof: A sequence $x(n)$ is periodic if and only if $x(n) = x(n+N)$ for some $N > 0$ and for all n. Let us impose this equality. In our case:

$$A \cdot \cos\left[2\pi f_0 n + \phi\right] = A \cdot \cos\left[2\pi f_0 (n+N) + \phi\right]$$

Thus, the arguments can differ only by a multiple of 2π:

$$2\pi f_0 (n+N) + \phi = 2\pi f_0 n + \phi + 2\pi k$$

with $k \in \mathbb{Z}$. By simplifying the last identity we arrive to:

$$f_0 N = k \qquad \Longrightarrow \qquad f_0 = \frac{k}{N} \in \mathbb{Q}.$$

Q.E.D.

- *Property:* Two sinusoidal sequences with the same amplitude and phase, whose angular frequencies differ for a multiple of 2π, are equal.

Proof: Let us consider

$$x_1(n) = A \cos\left(\omega_1 n + \phi\right)$$

$$x_2(n) = A \cos\left(\omega_2 n + \phi\right)$$

with $\omega_2 = \omega_1 + k \cdot 2\pi$ and $k \in \mathbb{Z}$. Thus

$$x_2(n) = A \cos\left(\omega_1 n + k 2\pi n + \phi\right) =$$

$$= A \cos\left(\omega_1 n + \phi\right) = x_1(n)$$

Q.E.D.

- In the sinusoidal sequence

$$x(n) = A\cos\big(\omega_0 n + \phi\big),$$

the frequency of oscillations in $x(n)$ increases with ω_0 varying from 0 to π.
The frequency of oscillations in $x(n)$ is maximum for $\omega_0 = \pi$ (since $x(n) = \ldots, -A, +A, -A, +A, \ldots$).
The frequency of oscillations in $x(n)$ decreases with ω_0 varying from π to 2π.
Eventually, this behavior repeats (with period 2π) in the intervals $[2\pi, 4\pi]$, $[4\pi, 6\pi]$, etc..

Let us consider the sinusoidal function

$$x_a(t) = A\cos(\Omega t + \phi) = A\cos(2\pi f t + \phi).$$

Let us sample it with a sampling frequency $F_s = \dfrac{1}{T}$:

$$x(n) = x_a(t)\Big|_{t=nT} = A\cos(2\pi f n T + \phi)$$

$$= A\cos(2\pi \frac{f}{F_s} n + \phi)$$

This is a sinusoidal sequence with normalized frequency $f_0 = \dfrac{f}{F_s}$. It explains the origin of the term 'normalized frequency': it is normalized with respect to the sampling frequency.
If $0 < 2\pi\dfrac{f}{F_s} < \pi$, i.e., $F_s > 2f$, the sinusoidal sequence follows the behavior of the sinusoidal function. In other words, as f increases, f_0 increases, and the frequency of oscillation increases.
On the contrary, if $2\pi\dfrac{f}{F_s} > \pi$, the sinusoidal sequence is unable to accurately follow the behavior of the sinusoidal function.
We will explore an important theorem, the sampling theorem, which states that a continuous-time signal can be faithfully reconstructed from his samples provided that the signal is sampled with a frequency at least twice its maximum frequency.

For more information study:

S. K. Mitra, "Digital Signal Processing: a computer based approach," 4th edition, McGraw-Hill, 2011
Chapter 2.1, pp. 41-45
Chapter 2.2, pp. 46-49
Chapter 2.3.3, pp. 58-62
Chapter 2.4, pp. 62-68

03 Discrete-time signals in the frequency domain

We have observed that each sequence can be expressed in the time domain as the weighted sum of infinite impulse sequences shifted in time:

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

In this chapter, we will explore an alternative representation of sequences through the weighted sum of infinite complex exponential sequences of the form $e^{-j\omega n}$, where ω represents the normalized angular frequency. This approach allows us to achieve a meaningful representation of sequences in the *frequency domain* and introduces the concept of signal *spectrum*.

We will initially explore continuous-time signals and introduce the Continuous-Time Fourier Transform (CTFT). Subsequently, we will transition to the discrete-time domain and introduce the Discrete-Time Fourier Transform (DTFT). Finally, we will examine the relationship between these transforms, particularly in the case of sampled signals.

Jean Baptiste Joseph Fourier, a French mathematician born in 1768, was the first to comprehend (in the early 1800s) that every periodic function can be represented as the sum of an infinite series of appropriately weighted sine and cosine functions at different frequencies.

03.01 The Fourier series

Let us assume that $x_p(t)$ is a complex function ($x_p(t) \in \mathbb{C}$), periodic with period T, continuous in t (with $t \in \mathbb{R}$). Then,

$$x_p(t) = \sum_{n=-\infty}^{+\infty} c_n \cdot e^{j\frac{2\pi}{T}nt}$$

where

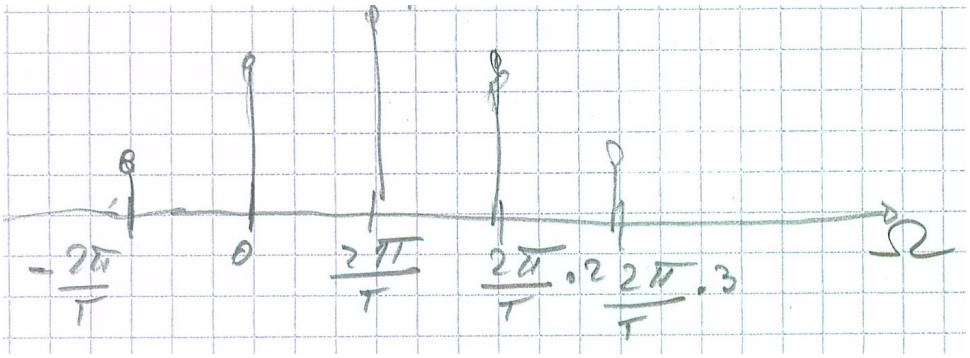
$$c_n = \frac{1}{T}\int_{-\frac{T}{2}}^{+\frac{T}{2}} x_p(t)\, e^{-j\frac{2\pi}{T}nt}\,\mathrm{d}t$$

For the Euler's formula

$$e^{j\,\theta} = \cos\theta + j\,\sin\theta,$$

and $x_p(n)$ is the sum of infinite sine and cosine functions at different frequencies, with each sine/cosine function multiplied by an appropriate weight coefficient.

Note that we can represent the function $x_p(t)$ in the angular frequency domain $\frac{2\pi}{T}n = \Omega$ by associating to every discrete frequency $\frac{2\pi}{T}n$ the corresponding coefficient c_n:



The periodic signal is represented in the frequency domain by an infinite number of discrete "lines", corresponding to the coefficients of the Fourier series expansion of the signal. These lines are uniformly spaced and separated by $\frac{2\pi}{T}$.
Note that by increasing the period of the periodic function, the lines tend to become narrower (more frequent). Thus, we can understand that aperiodic functions (satisfying certain conditions we will see later) can also be expressed as the sum of sine and cosine functions, or, more precisely, as the sum of complex exponential functions $e^{j\Omega t}$ (whose frequency varies continuously) multiplied by a frequency-varying weight function.
Thus, we arrive at the Continuous-Time Fourier Transform (CTFT).

03.02 The Continuous-Time Fourier Transform

The frequency domain representation of a continuous-time signal $x_a(t)$ is given by the Continuous-Time Fourier Transform (CTFT), defined as[1]

$$X_a(j\Omega) = \int_{-\infty}^{+\infty} x_a(t)\, e^{-j\Omega t} \mathrm{d}t$$

The CTFT is also referred to as the *Fourier spectrum*, or simply *spectrum*, of the continuous-time signal.
The continuous-time signal $x_a(t)$ can be reconstructed from its CTFT by means of the *inverse* continuous-time Fourier transform (ICTFT), defined as

$$x_a(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} X_a(j\Omega) e^{j\Omega t} \mathrm{d}\Omega.$$

Note the bijective mapping between the signal $x_a(t)$ and its transform:

$$x_a(t) \overset{\text{CTFT}}{\longleftrightarrow} X_a(j\Omega)$$

Ω is a real variable representing the continuous-time angular frequency, measured in rad/s.
The inverse transform can be interpreted as the linear combination of infinitesimally small complex exponential signals of the form $\frac{1}{2\pi} e^{j\Omega t} \mathrm{d}\Omega$.

[1] Why $X_a(j\Omega)$ and not $X_a(\Omega)$? It is simply a convention. Another transform, the Laplace transform $X_a(s)$, is defined on the entire complex plane, and when evaluated on the imaginary axis (i.e., for $s = j\Omega$), it yields the CTFT.

We can also express the transform in polar form:

$$X_a(j\Omega) = \left|X_a(j\Omega)\right| \cdot e^{j\theta_a(\Omega)},$$

where $\theta_a(\Omega) = \arg\{X_a(j\Omega)\}$.
$\left|X_a(j\Omega)\right|$ is referred to as the *magnitude spectrum*, and $\theta_a(\Omega)$ is called the *phase spectrum*. Both $\left|X_a(j\Omega)\right|$ and $\theta_a(\Omega)$ are real functions of the angular frequency Ω.
Note that not all signals admit the CTFT. The integral $\int_{-\infty}^{+\infty} \cdot \mathrm{d}t$ may not converge. The CTFT exists if the continuous-time signal $x_a(t)$ satisfies the *Dirichlet conditions*:

1. The signal has a finite number of discontinuities and a finite number of maxima and minima in any finite interval.
2. The signal is absolutely integrable, i.e.,

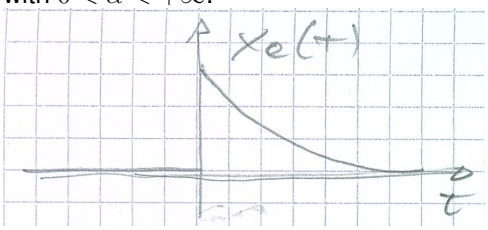
$$\int_{-\infty}^{+\infty} \left|x_a(t)\right| < +\infty$$

If these conditions are satisfied, $\int_{-\infty}^{+\infty} x_a(t)e^{-j\Omega t}\mathrm{d}t$ converges and $x_a(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} X_a(j\Omega)e^{j\Omega t}\mathrm{d}\Omega$ apart from the discontinuity points.

Example:

$$x_a(t) = \begin{cases} e^{-\alpha t} & t \geq 0 \\ 0 & t < 0 \end{cases},$$

with $0 < \alpha < +\infty$.



This signal satisfies the Dirichlet conditions. Indeed, it has a unique discontinuity and

$$\int_{-\infty}^{+\infty} \left|x_a(t)\right| = \int_0^{+\infty} e^{-\alpha t}\mathrm{d}t = -\left.\frac{e^{-\alpha t}}{\alpha}\right|_0^{+\infty} = 0 - \left(-\frac{1}{\alpha}\right) = \frac{1}{\alpha}$$

$$\mathrm{CTFT}[x_a(t)] = X_a(j\Omega) = \int_0^{+\infty} e^{-\alpha t}\, e^{-j\Omega t}\mathrm{d}t = \int_0^{+\infty} e^{-(\alpha+j\Omega)t}\mathrm{d}t$$

$$= -\left.\frac{1}{\alpha+j\Omega}e^{-(\alpha+j\Omega)t}\right|_0^{+\infty} = \frac{1}{\alpha+j\Omega} \quad \Big/ \cdot \frac{\alpha-j\Omega}{\alpha-j\Omega}$$

$$= \frac{\alpha-j\Omega}{\alpha^2+\Omega^2}$$

$$\left|X_a(j\Omega)\right| = \sqrt{\mathrm{Re}\{\}^2 + \mathrm{Im}\{\}^2} = \frac{1}{\sqrt{\alpha^2+\Omega^2}}$$

$$\theta_a(\Omega) = \arctan\frac{\mathrm{Im}\{\}}{\mathrm{Re}\{\}} = -\arctan\frac{\Omega}{\alpha}$$

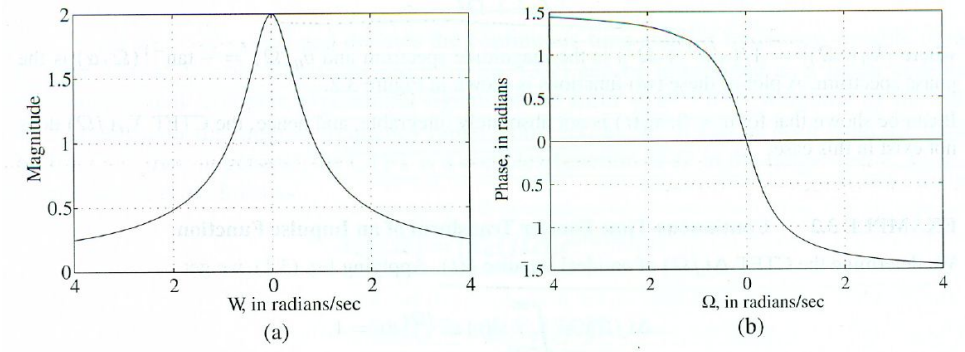


Figure 3.2: (a) Magnitude and (b) phase of $X_a(j\Omega) = 1/(0.5/\text{sec} + j\Omega)$.

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

CTFT of a Dirac delta function

The *Dirac delta function* $\delta(t)$ is a function of the continuous-time variable t with notable properties, defined as:

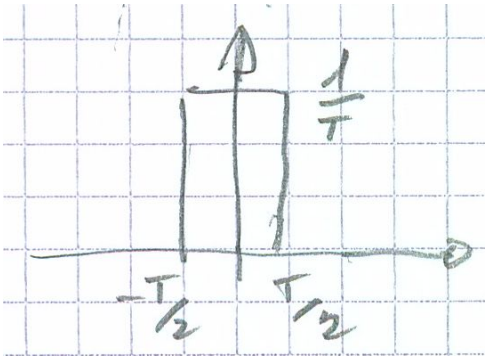
$$\delta(t) = \begin{cases} 0 & t \neq 0 \\ +\infty & t = 0 \end{cases}$$

with

$$\int_{-\infty}^{+\infty} \delta(t)\mathrm{d}t = 1.$$

It is the limit, as T approaches 0, of the rectangular pulse

$$\Delta_T(t) = \begin{cases} \dfrac{1}{T} & -\dfrac{T}{2} \leq t \leq +\dfrac{T}{2} \\ 0 & \text{elsewhere} \end{cases}$$



Properties of Dirac delta function:

$$\int_{-\infty}^{+\infty} f(t)\,\delta(t)\mathrm{d}t = f(0)$$

$$\int_{-\infty}^{+\infty} f(t)\,\delta(t-t_0)\mathrm{d}t = f(t_0)$$

$$\text{CTFT}\{\delta(t)\} = \Delta(j\Omega) = \int_{-\infty}^{+\infty} \delta(t)e^{-j\Omega t}\mathrm{d}t = 1$$

$$\text{CTFT}\{\delta(t-t_0)\} = \int_{-\infty}^{+\infty} \delta(t-t_0)e^{-j\Omega t}\mathrm{d}t = e^{-j\Omega t_0}$$

Linearity of the CTFT

If $F_a(j\Omega)$ is the CTFT of $f_a(t)$ and $G_a(j\Omega)$ is the CTFT of $g_a(t)$, the CTFT of $x_a(t) = \alpha f_a(t) + \beta g_a(t)$, with α and β constants, is

$$X_a(j\Omega) = \alpha F_a(j\Omega) + \beta G_a(j\Omega).$$

Proof:

$$X_a(j\Omega) = \int_{-\infty}^{+\infty} x_a(t)\, e^{-j\Omega t} \mathrm{d}t =$$

$$= \int_{-\infty}^{+\infty} [\alpha f_a(t) + \beta g_a(t)]\, e^{-j\Omega t} \mathrm{d}t =$$

$$= \alpha \int_{-\infty}^{+\infty} f_a(t)\, e^{-j\Omega t} \mathrm{d}t + \beta \int_{-\infty}^{+\infty} g_a(t)\, e^{-j\Omega t} \mathrm{d}t =$$

$$= \alpha F_a(j\Omega) + \beta G_a(j\Omega)$$

Q.E.D.

Time-shift property

If $G_a(j\Omega)$ is the CTFT of $g_a(t)$, the CTFT of $x_a(t) = g_a(t - t_0)$, with t_0 constant, is

$$G_a(j\Omega) e^{-j\Omega t_0}.$$

Proof:

$$X_a(j\Omega) = \int_{-\infty}^{+\infty} g_a(t - t_0)\, e^{-j\Omega t} \mathrm{d}t =$$

Let us introduce the change of variables $t' = t - t_0$, i.e., $t = t' + t_0$:

$$= \int_{-\infty}^{+\infty} g_a(t')\, e^{-j\Omega(t' + t_0)} \mathrm{d}t' =$$

$$= \int_{-\infty}^{+\infty} g_a(t')\, e^{-j\Omega t'} \mathrm{d}t' e^{-j\Omega t_0} =$$

$$= G_a(j\Omega) e^{-j\Omega t_0}.$$

Q.E.D.

Symmetry property of the CTFT

The CTFT of a real signal $x_a(t) \in \mathbb{R}$ satisfies the property

$$X_a(-j\Omega) = X_a^*(j\Omega)$$

where x^* is the conjugate of x.

Proof:

$$X_a(j\Omega) = \int_{-\infty}^{+\infty} x_a(t) e^{-j\Omega t} \mathrm{d}t$$

$$X_a^*(j\Omega) = \int_{-\infty}^{+\infty} x_a^*(t) e^{j\Omega t} \mathrm{d}t$$

$$= \int_{-\infty}^{+\infty} x_a(t) e^{-j(-\Omega)t} \mathrm{d}t$$
$$= X_a(-j\Omega).$$

Q.E.D.

Energy density spectrum

The total energy E_x of a continuous-time signal $x_a(t)$ is given by

$$E_x = \int_{-\infty}^{+\infty} \left|x_a(t)\right|^2 \mathrm{d}t.$$

An absolutely integrable signal, i.e., a signal for which $\int_{-\infty}^{+\infty} \left|x_a(t)\right| \mathrm{d}t < +\infty$, has finite energy, but there exist signals with finite energy which are not absolutely integrable. Moreover, there are signals with infinite energy (for example, periodic signals).

Consider a finite energy signal that admits CTFT. We can express the energy as a function of $X_a(j\Omega)$:

$$E_x = \int_{-\infty}^{+\infty} \left|x_a(t)\right|^2 \mathrm{d}t = \int_{-\infty}^{+\infty} x_a(t) x_a^*(t) \mathrm{d}t =$$
$$\int_{-\infty}^{+\infty} x_a(t) \left[\frac{1}{2\pi} \int_{-\infty}^{+\infty} X_a^*(j\Omega) e^{-j\Omega t} \mathrm{d}\Omega\right] \mathrm{d}t$$

where we have replaced $x_a^*(t)$ with the ICTFT. Let us exchange the two integrations:

$$E_x = \frac{1}{2\pi} \int_{-\infty}^{+\infty} X_a^*(j\Omega) \int_{-\infty}^{+\infty} x_a(t) e^{-j\Omega t} \mathrm{d}t \mathrm{d}\Omega$$

The inner integral is the CTFT of $x_a(t)$. Thus

$$E_x = \frac{1}{2\pi} \int_{-\infty}^{+\infty} X_a^*(j\Omega) X_a(j\Omega) \mathrm{d}\Omega = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \left|X_a(j\Omega)\right|^2 \mathrm{d}\Omega$$

The integral from $-\infty$ to $+\infty$ of the squared magnitude spectrum equals the signal energy. Thus, the following theorem is proved.

Parseval's Theorem:

$$E_x = \int_{-\infty}^{+\infty} \left|x_a(t)\right|^2 \mathrm{d}t = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \left|X_a(j\Omega)\right|^2 \mathrm{d}\Omega$$

$S_{xx}(\Omega) = \left|X_a(j\Omega)\right|^2$ is also called *Energy density spectrum* of the signal $x_a(t)$, and it provides the energy content of the signal at angular frequency Ω.

The energy of the signal $x_a(t)$ in a frequency range $\Omega_a \leq \Omega \leq \Omega_b$ can be computed by integrating $S_{xx}(\Omega)$ over the interval $[\Omega_a, \Omega_b]$:

$$E_{x,r} = \frac{1}{2\pi} \int_{\Omega_a}^{\Omega_b} S_{xx}(\Omega) \mathrm{d}\Omega.$$

Band limited signals

A full-band continuous-time signal $x_a(t)$ has a spectrum occupying the entire frequency range. A signal is called *band-limited* if its spectrum occupies only a portion of the frequency range $-\infty < \Omega < +\infty$.

An *ideal band-limited signal* has spectrum that is zero outside a certain range $\Omega_a \leq \left|\Omega\right| \leq \Omega_b$, with $0 \leq \Omega_a, \Omega_b \leq +\infty$, i.e.,

$$X_a(j\Omega) = \begin{cases} 0 & 0 \leq |\Omega| < \Omega_a \\ 0 & \Omega_b < |\Omega| < +\infty \end{cases}$$

[Why do we consider $|\Omega|$ instead of Ω? Because if the signal $x_a(t)$ is real, $X_a(-j\Omega) = X_a^*(j\Omega)$, i.e., the spectrum has conjugate symmetry. If $X_a(j\Omega) \neq 0$ in a range $[\Omega_a, \Omega_b]$, then $X_a(j\Omega) \neq 0$ in $[-\Omega_b, -\Omega_a]$.]
It is not possible to generate an ideal band-limited signal, but in most applications, it suffices to ensure that the signal has sufficiently low energy outside the interval $[\Omega_a, \Omega_b]$ of interest.
A signal is called *low-pass* if its spectrum occupies the frequency range $0 \leq |\Omega| \leq \Omega_p$, where Ω_p is called the signal *bandwidth*.
A signal is called *high-pass* if its spectrum occupies the frequency range $\Omega_p \leq |\Omega| \leq +\infty$, and the signal bandwidth extends from Ω_p to ∞.
Eventually, a signal is called *passband* if its spectrum occupies the frequency range $0 < \Omega_L \leq |\Omega| \leq \Omega_H < +\infty$, where $\Omega_H - \Omega_L$ is the signal bandwidth.

03.03 The Discrete-Time Fourier Transform

The frequency domain representation of a discrete-time signal is given by the *Discrete-Time Fourier Transform* (DTFT). This transform expresses a sequence as a weighted combination of complex exponential sequences of the form $e^{j\omega n}$, where ω is the *normalized angular frequency*, and $\omega \in \mathbb{R}$.
If it exists, the DTFT of a sequence is unique and the original sequence can be recovered from its transform with an inverse transform operation.
The DTFT $X(e^{j\omega})$ of a sequence $x(n)$ is defined by:

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

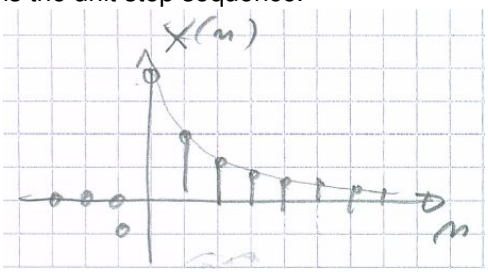
Example:
Let us compute the DTFT of the unit impulse sequence $\delta(n)$,

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

$$\Delta(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} \delta(n)e^{-j\omega n} = 1 \cdot e^{-j\omega 0} = 1$$

Example:
Let us compute the DTFT of the causal exponential sequence $x(n) = \alpha^n \mu(n)$, where $|\alpha| < 1$ and $\mu(n)$

is the unit step sequence.



$$X(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} \alpha^n \mu(n) e^{-j\omega n} =$$

$$= \sum_{n=0}^{+\infty} \alpha^n e^{-j\omega n} =$$

$$= \sum_{n=0}^{+\infty} \left(\alpha e^{-j\omega}\right)^n =$$

$$= \frac{1}{1 - \alpha e^{-j\omega}}$$

since $|\alpha e^{j\omega}| = |\alpha| < 1$.

To derive this result, we have used the following identity:

$$\sum_{n=0}^{+\infty} q^n = \frac{1}{1-q} \qquad \text{if } |q| < 1.$$

We will encounter this exponential series frequently. The sum can be easily computed using the following steps:

$$S = \sum_{n=0}^{M-1} q^n = 1 + q + q^2 + \ldots + q^{M-1}$$

$$S \cdot q = q + q^2 + \ldots + q^M$$

$$S - S \cdot q = S(1-q) = 1 - q^M$$

$$S = \frac{1 - q^M}{1 - q}.$$

If $|q| < 1$ and $M \longrightarrow +\infty$, then $q^M \longrightarrow 0$ and $S \longrightarrow \frac{1}{1-q}$.

Q.E.D.

From the definition of the DTFT of a sequence $x(n)$, we can notice that it is a continuous function of the normalized angular frequency ω. Unlike the CTFT, the DTFT is a periodic function of ω with a period of 2π. Indeed,

$$X(e^{j(\omega+2\pi k)}) = \sum_{n=-\infty}^{+\infty} x(n) e^{-j(\omega+2\pi k)n} =$$

$$= \sum_{n=-\infty}^{+\infty} x(n) e^{-j\omega n} \cdot e^{-j2\pi kn} =$$

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

where we used the fact that $e^{-j2\pi kn} = 1$ for k and n integers.

Note that $\sum_{n=-\infty}^{+\infty} x(n)e^{-j\omega n}$ is the Fourier series expansion of the periodic function $X(e^{j\omega})$.

Thus, the coefficients of the Fourier series, $x(n)$, can be computed from $X(e^{j\omega})$ using the Fourier integral:

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

which is the *Inverse Discrete-Time Fourier Transform* (IDTFT).

The IDTFT can be interpreted as the linear combination of infinitesimally small complex exponential signals of the form $\frac{1}{2\pi}e^{j\omega n}\mathrm{d}\omega$ weighted by the complex function $X(e^{j\omega})$.

Let us verify that

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

By replacing the definition of $X(e^{j\omega})$ with

$$X(e^{j\omega}) = \sum_{l=-\infty}^{+\infty} x(l)e^{-j\omega l}$$

we have

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

by interchanging $\int$ and $\sum$:

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

For computing this integral, we will consider two cases.

If $n \neq l$,

$$\int_{-\pi}^{+\pi} e^{j\omega(n-l)}\mathrm{d}\omega = \int_{-\pi}^{+\pi} \cos\big[\omega(n-l)\big]\mathrm{d}\omega + j\int_{-\pi}^{+\pi} \sin\big[\omega(n-l)\big]\mathrm{d}\omega =$$

$$= \frac{\sin\big[\omega(n-l)\big]}{(n-l)}\bigg|_{-\pi}^{+\pi} =$$

$$= \frac{2\sin\big[\pi(n-l)\big]}{(n-l)} = 0.$$

Here, we have first exploited the fact that $\sin\big[\omega(n-l)\big]$ is an odd function and has $\int_{-\pi}^{+\pi}\mathrm{d}\omega = 0$. Finally, we have used the fact that $\sin\big[\pi(n-l)\big] = 0$.

If $n = l$,

$$\int_{-\pi}^{+\pi} e^{j\omega(n-l)}\mathrm{d}\omega = \int_{-\pi}^{+\pi} e^{j\omega 0}\mathrm{d}\omega = 2\pi.$$

Thus, in general, it is

$$\int_{-\pi}^{+\pi} e^{j\omega(n-l)}\mathrm{d}\omega = 2\pi\delta(n-l)$$

where $\delta(n)$ is the unit impulse sequence.
Eventually, we have

$$\frac{1}{2\pi}\sum_{l=-\infty}^{+\infty} x(l)\int_{-\pi}^{+\pi} e^{j\omega(n-l)}\mathrm{d}\omega = \sum_{l=-\infty}^{+\infty} x(l)\delta(n-l) = x(n)$$

Q.E.D.

In general, the DTFT of a sequence is a complex function of the real variable ω:

$$X(e^{j\omega}) = X_{\mathrm{re}}(e^{j\omega}) + jX_{\mathrm{im}}(e^{j\omega})$$

with $X_{\mathrm{re}}(e^{j\omega})$ and $X_{\mathrm{im}}(e^{j\omega}) \in \mathbb{R}$.

$$X_{\mathrm{re}}(e^{j\omega}) = \frac{1}{2}\left[X(e^{j\omega}) + X^*(e^{j\omega})\right],$$
$$X_{\mathrm{im}}(e^{j\omega}) = \frac{1}{2j}\left[X(e^{j\omega}) - X^*(e^{j\omega})\right].$$

The DTFT can be expressed in polar form as

$$X(e^{j\omega}) = \left|X(e^{j\omega})\right| e^{j\theta(\omega)}$$

where $\left|X(e^{j\omega})\right|$ is called *magnitude spectrum*, and $\theta(\omega)$ is called the *phase spectrum*:

$$\theta(\omega) = \arctan\frac{X_{\mathrm{im}}(e^{j\omega})}{X_{\mathrm{re}}(e^{j\omega})}$$

Note that there is an indetermination of $2\pi k$, with $k \in \mathbb{Z}$, in the knowledge of $\theta(\omega)$. Indeed, $\theta(\omega)$ is an angle measured in radians.
Like $X(e^{j\omega})$, also $X_{\mathrm{re}}(e^{j\omega})$, $X_{\mathrm{im}}(e^{j\omega})$, $\left|X(e^{j\omega})\right|$, and $\theta(\omega)$ are periodic functions of ω with period 2π. Thus, it suffices to know these functions for $-\pi < \omega \leq \pi$ to know them everywhere.

Symmetry properties of the DTFT of a real sequence
If $x(n) \in \mathbb{R}$:

$$X(e^{-j\omega}) = X^*(e^{j\omega}).$$

In fact:

$$X^*(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} x^*(n)e^{j\omega n}$$
$$= \sum_{n=-\infty}^{+\infty} x(n)e^{-j(-\omega)n}$$
$$= X(e^{-j\omega}).$$

For this relation, if $x(n) \in \mathbb{R}$ we have that

$$X_{\mathrm{re}}(e^{-j\omega}) = X_{\mathrm{re}}(e^{j\omega}) \qquad \text{is an even function,}$$

$$X_{\mathrm{im}}(e^{-j\omega}) = -X_{\mathrm{im}}(e^{j\omega}) \qquad \text{is an odd function,}$$

$$\left|X(e^{-j\omega})\right| = \left|X(e^{j\omega})\right| \qquad \text{is an even function,}$$

$$\theta(-\omega) = -\theta(\omega) \qquad \text{is an odd function.}$$

If $x(n)$ is real and even ($x(n) = x(-n)$),

$$X(e^{j\omega}) = X_{\mathrm{re}}(e^{j\omega}) \qquad \text{is an real function.}$$

Indeed:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{-1} x(n)e^{j\omega n} + x(0) + \sum_{n=1}^{+\infty} x(n)e^{j\omega n}$$

(Change the variable of the first sum as follows $n' = -n$)

$$= \sum_{n'=1}^{+\infty} x(-n')e^{-j\omega n'} + x(0) + \sum_{n=1}^{+\infty} x(n)e^{j\omega n}$$

$$= x(0) + \sum_{n=1}^{+\infty} x(n)\left[e^{-j\omega n} + e^{j\omega n}\right]$$

$$= x(0) + \sum_{n=1}^{+\infty} x(n)2\cos(\omega n)$$

Q.E.D.

If $x(n)$ is real and odd ($x(-n) = -x(n)$),

$$X(e^{j\omega}) = jX_{\mathrm{im}}(e^{j\omega}) \qquad \text{is an imaginary function.}$$

Convergence conditions

The series that defines the Fourier transform could or could not converge. We say that the DTFT exists if its series converges according to some criteria.

Sufficient conditions for the existence of the DTFT of a sequence $x(n)$ are the following:

- $x(n)$ is absolutely summable, i.e.,
$$\sum_{-\infty}^{+\infty} |x(n)| < +\infty$$
In this case, we talk about *uniform convergence*.
- $x(n)$ is a finite energy signal
$$\sum_{-\infty}^{+\infty} |x(n)|^2 < +\infty$$
In this case, we talk about *mean square convergence*.

These two conditions are quite restrictive. Note that the first condition is stronger than the second. An absolutely summable signal is always a finite energy signal because:

$$\sum_{-\infty}^{+\infty} |x(n)|^2 \leq \left(\sum_{-\infty}^{+\infty} |x(n)|\right)^2.$$

Example

An absolutely summable signal is the following causal exponential sequence

$$x(n) = \alpha^n \mu(n)$$

for $|\alpha| < 1$.

Indeed,

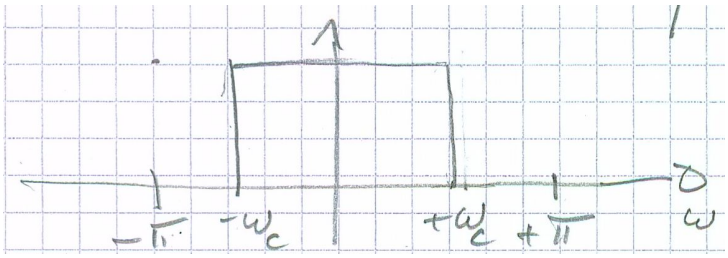
$$\sum_{n=-\infty}^{+\infty} |\alpha^n \mu(n)| = \sum_{n=0}^{+\infty} |\alpha^n| = \frac{1}{1-|\alpha|} < +\infty.$$

We have already seen that its DTFT is $\dfrac{1}{1-\alpha e^{-j\omega}}$.

Example

A signal that is not absolutely summable (but has a DTFT) is the signal with the following ideal low-pass DTFT:

$$H_{\mathrm{LP}}(e^{j\omega}) = \begin{cases} 1 & 0 \leq |\omega| \leq \omega_c \\ 0 & \omega_c < |\omega| \leq \pi \end{cases}$$



Let us compute the corresponding signal, which we will encounter frequently.

Let us compute the IDTFT:

$$h_{\mathrm{LP}}(n) = \frac{1}{2\pi} \int_{-\pi}^{+\pi} H_{\mathrm{LP}}(e^{j\omega}) e^{j\omega n} \mathrm{d}\omega,$$

for $-\infty < n < +\infty$ and $n \neq 0$:

$$\begin{aligned} h_{\mathrm{LP}}(n) &= \frac{1}{2\pi} \int_{-\omega_c}^{+\omega_c} e^{j\omega n} \mathrm{d}\omega \\ &= \frac{1}{2\pi} \left[\frac{e^{j\omega_c n}}{jn} - \frac{e^{-j\omega_c n}}{jn} \right] \\ &= \frac{\sin(\omega_c n)}{\pi n}, \end{aligned}$$

for $n = 0$

$$h_{\mathrm{LP}}(n) = \frac{1}{2\pi} \int_{-\omega_c}^{+\omega_c} \mathrm{d}\omega = \frac{\omega_c}{\pi}$$

Thus,

$$h_{\mathrm{LP}}(n) = \begin{cases} \dfrac{\omega_c}{\pi} & n = 0 \\ \dfrac{\sin(\omega_c n)}{\pi n} & n \neq 0 \end{cases}$$

Since

$$\frac{\sin \omega_c n}{\pi n} = \frac{\omega_c}{\pi} \frac{\sin(\omega_c n)}{\omega_c n},$$

and since for the L'Hopital's rule $\lim_{x\to 0}\frac{\sin x}{x}=1$, assuming by convention that $\frac{\sin(\omega_c 0)}{\omega_c 0}=1$, we can write for all n

$$h_{\mathrm{LP}}(n)=\frac{\sin(\omega_c n)}{\pi n}.$$

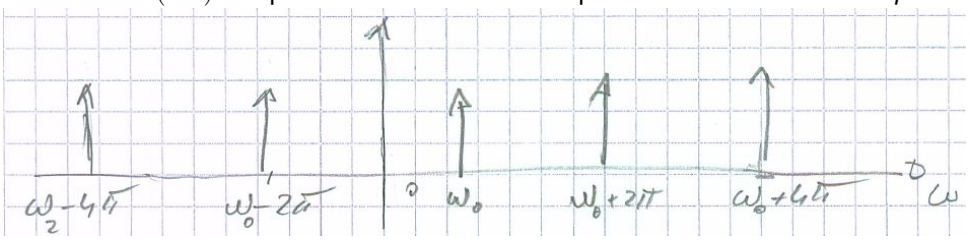
Note that $\sum_{-\infty}^{+\infty}|h_{\mathrm{LP}}(n)|=+\infty$, i.e., the sequence is not absolutely summable, but it admits DTFT because it is a finite energy sequence.

The DTFT can be defined also for a certain class of sequences that are not absolutely summable, nor have finite energy, e.g., the unit step sequence, the sinusoidal sequence, or the complex exponential sequence. In these cases the expression of the DTFT involves Dirac delta functions.

Example
The DTFT of the complex exponential sequence $x(n)=e^{j\omega_0 n}$ is

$$X(e^{j\omega})=\sum_{k=-\infty}^{+\infty}2\pi\delta(\omega-\omega_0+2\pi k)$$

The DTFT $X(e^{j\omega})$ is a periodic function in ω with period 2π and it is called a *periodic pulse train*.



Let us prove this relation by computing the IDTFT:

$$x(n)=\frac{1}{2\pi}\int_{-\pi}^{+\pi}\sum_{k=-\infty}^{+\infty}2\pi\delta(\omega-\omega_0+2\pi k)e^{j\omega n}\mathrm{d}\omega$$

If $-\pi<\omega_o\leq\pi$, the only non-null function in the interval $[-\pi,+\pi]$ is $\delta(\omega-\omega_0)$, thus

$$x(n)=\frac{1}{2\pi}2\pi\int_{-\pi}^{+\pi}\delta(\omega-\omega_0)e^{j\omega n}\mathrm{d}\omega=e^{j\omega_0 n}$$

for the properties of $\delta(\omega)$.

Q.E.D

03.04 Properties of DTFT

Let us assume that $G(e^{j\omega})$ is the DTFT of a sequence $g(n)$ and that $H(e^{j\omega})$ is the DTFT of a sequence $h(n)$. The following properties hold.

Linearity property

If $x(n) = \alpha g(n) + \beta h(n)$, with α and β constants, the DTFT of $x(n)$ is $X(e^{j\omega}) = \alpha G(e^{j\omega}) + \beta H(e^{j\omega})$.

$$\alpha g(n) + \beta h(n) \overset{DTFT}{\longleftrightarrow} \alpha G(e^{j\omega}) + \beta H(e^{j\omega})$$

Proof:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} (\alpha g(n) + \beta h(n))\, e^{-j\omega n}$$

$$= \alpha \sum_{n=-\infty}^{+\infty} g(n) e^{-j\omega n} + \beta \sum_{n=-\infty}^{+\infty} h(n) e^{-j\omega n}$$

$$= \alpha G(e^{j\omega}) + \beta H(e^{j\omega})$$

Q.E.D.

Time reversal

The DTFT of $g(-n)$ is $G(e^{-j\omega})$.

$$g(-n) \overset{DTFT}{\longleftrightarrow} G(e^{-j\omega})$$

Proof:

$$\sum_{n=-\infty}^{+\infty} g(-n) e^{-j\omega n} = \sum_{n'=-\infty}^{+\infty} g(n') e^{-j\omega(-n')} =$$

(where we have considered the following change of variables $n' = -n$)

$$= \sum_{n'=-\infty}^{+\infty} g(n') e^{-j(-\omega)n'}$$

Q.E.D.

Time-shifting

The DTFT of $g(n - n_0)$ is $e^{-j\omega n_0} G(e^{j\omega})$.

$$g(n - n_0) \overset{DTFT}{\longleftrightarrow} e^{-j\omega n_0} G(e^{j\omega})$$

Proof:

$$\sum_{n=-\infty}^{+\infty} g(n - n_0) e^{-j\omega n} = \sum_{n'=-\infty}^{+\infty} g(n') e^{-j\omega(n' + n_0)} =$$

(where we have considered the following change of variables $n' = n - n_0$)

$$= \sum_{n'=-\infty}^{+\infty} g(n') e^{-j\omega n'} e^{-j\omega n_0}$$

Q.E.D.

Frequency-shifting

The DTFT of $e^{j\omega_0 n} g(n)$ is $G(e^{j(\omega - \omega_0)})$.

$$e^{j\omega_0 n} g(n) \overset{DTFT}{\longleftrightarrow} G(e^{j(\omega - \omega_0)})$$

Proof:

$$\sum_{n=-\infty}^{+\infty} g(n)e^{j\omega_0 n}e^{-j\omega n} = \sum_{n=-\infty}^{+\infty} g(n)e^{-j(\omega-\omega_0)n}$$

Q.E.D.

Frequency differentiation

The DTFT of $ng(n)$ is $j\dfrac{\mathrm{d}G(e^{j\omega})}{\mathrm{d}\omega}$.

$$ng(n) \overset{DTFT}{\longleftrightarrow} j\frac{\mathrm{d}G(e^{j\omega})}{\mathrm{d}\omega}$$

Proof: Omitted.

Modulation

The DTFT of $g(n) \cdot h(n)$ is $\dfrac{1}{2\pi}\displaystyle\int_{-\pi}^{+\pi} G(e^{j\theta})H(e^{j(\omega-\theta)})\mathrm{d}\theta$.

$$g(n) \cdot h(n) \overset{DTFT}{\longleftrightarrow} \frac{1}{2\pi}\int_{-\pi}^{+\pi} G(e^{j\theta})H(e^{j(\omega-\theta)})\mathrm{d}\theta$$

Proof:

$$y(n) = g(n) \cdot h(n)$$

$$Y(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} g(n)h(n)e^{-j\omega n}$$

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

(where we have replaced $g(n)$ with its IDTFT. By exchanging $\sum$ and $\int$ ...)

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

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

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

Q.E.D.

The last integral is called *convolution integral*. We will introduce the concept of convolution in one of the following lessons.

Parseval theorem

$$\sum_{n=-\infty}^{+\infty} g(n)h^*(n) = \frac{1}{2\pi}\int_{-\pi}^{+\pi} G(e^{j\omega})H^*(e^{j\omega})\mathrm{d}\omega$$

Proof: (Similar to the previous one)

$$\sum_{n=-\infty}^{+\infty} g(n)h^*(n) = \sum_{n=-\infty}^{+\infty} g(n)\left(\frac{1}{2\pi}\int_{-\pi}^{+\pi} H^*(e^{j\omega})e^{-j\omega n}\mathrm{d}\omega\right)$$

$$= \frac{1}{2\pi} \int_{-\pi}^{+\pi} H^*(e^{j\omega}) \left(\sum_{n=-\infty}^{+\infty} g(n) e^{-j\omega n} \right) \mathrm{d}\omega$$

$$= \frac{1}{2\pi} \int_{-\pi}^{+\pi} G(e^{j\omega}) H^*(e^{j\omega}) \mathrm{d}\omega$$

Q.E.D.

The linearity property, the time-shift property, the frequency shift property, the modulation property and the Parseval theorem find similar formulations also with the CTFT.

Examples using these theorems:

Let us compute the DTFT of the finite length exponential sequence:

$$y(n) = \begin{cases} \alpha^n & 0 \leq n < M-1 \\ 0 & \text{otherwise} \end{cases}$$

with $|\alpha| < 1$.

$$y(n) = \alpha^n \mu(n) - \alpha^n \mu(n-M)$$

$$= \alpha^n \mu(n) - \alpha^M \alpha^{n-M} \mu(n-M)$$

We already know that:

$$\alpha^n \mu(n) \overset{DTFT}{\longleftrightarrow} \frac{1}{1-\alpha e^{-j\omega}}$$

For the linearity and the time shift properties we have

$$Y(e^{j\omega}) = \frac{1}{1-\alpha e^{-j\omega}} - \alpha^M \frac{e^{-j\omega M}}{1-\alpha e^{-j\omega}}$$

$$= \frac{1-\alpha^M e^{-j\omega M}}{1-\alpha e^{-j\omega}}$$

Let us compute the DTFT of the causal sequence $v(n)$ defined by the finite difference equation:

$$d_0 v(n) + d_1 v(n-1) = p_0 \delta(n) + p_1 \delta(n-1)$$

with $d_0 \neq 0$, and $d_1 \neq 0$.

Let us apply the DTFT to both terms of the finite difference equation. By exploiting the linearity and time-shift properties and remembering that DTFT$\{\delta(n)\} = 1$, we have

$$d_0 V(e^{j\omega}) + d_1 e^{-j\omega} V(e^{j\omega}) = p_0 + p_1 e^{-j\omega}$$

Thus,

$$V(e^{j\omega}) = \frac{p_0 + p_1 e^{-j\omega}}{d_0 + d_1 e^{-j\omega}}.$$

Let us compute the DTFT of $y(n) = (-1)^n \alpha^n \mu(n)$ with $|\alpha| < 1$.

It is also $y(n) = e^{j\pi n} (\alpha^n \mu(n)) = e^{j\pi n} x(n)$, where we considered $x(n) = \alpha^n \mu(n)$.

We can apply the frequency-shifting property:

$$Y(e^{j\omega}) = X(e^{j(\omega-\pi)}) = \frac{1}{1-\alpha e^{-j(\omega-\pi)}} = \frac{1}{1+\alpha e^{-j\omega}}.$$

Energy density spectrum

The total energy of a sequence $g(n)$ is given by

$$E_g = \sum_{n=-\infty}^{+\infty} |g(n)|^2 .$$

By applying the Parseval theorem with $h(n) = g(n)$ we have

$$E_g = \frac{1}{2\pi} \int_{-\pi}^{+\pi} \left|G(e^{j\omega})\right|^2 \mathrm{d}\omega.$$

$S_{gg}(e^{j\omega}) = \left|G(e^{j\omega})\right|^2$ is called *energy density spectrum*, and it defines how the energy of the sequence is distributed over the frequency spectrum.

Example:

Let us compute the energy of the low-pass ideal signal

$$\sum_{n=-\infty}^{+\infty} |h_{\mathrm{LP}}(n)|^2 = \frac{1}{2\pi} \int_{-\pi}^{+\pi} \left|H_{\mathrm{LP}}(e^{j\omega})\right|^2 \mathrm{d}\omega = \frac{1}{2\pi} \int_{-\omega_c}^{+\omega_c} \mathrm{d}\omega = \frac{\omega_c}{\pi} < +\infty$$

Band-limited signals

In general, the spectrum of a discrete-time signal is defined on the entire frequency range $-\pi < \omega \leq +\pi$.

A band-limited signal has a spectrum that is limited to part of this range.

An ideal *low-pass* signal has

$$X(e^{j\omega}) \begin{cases} \neq 0 & 0 \leq |\omega| \leq \omega_p \\ = 0 & \omega_p < |\omega| \leq \pi \end{cases}$$

ω_p is called signal *bandwidth*.

An ideal *high-pass* signal has

$$X(e^{j\omega}) \begin{cases} = 0 & 0 \leq |\omega| < \omega_p \\ \neq 0 & \omega_p \leq |\omega| \leq \pi \end{cases}$$

The signal bandwidth is given by $\pi - \omega_p$.

An ideal *passband* signal has

$$X(e^{j\omega}) \begin{cases} = 0 & 0 \leq |\omega| < \omega_a \\ \neq 0 & \omega_a \leq |\omega| \leq \omega_b \\ = 0 & \omega_b < |\omega| < \pi \end{cases}$$

The signal bandwidth is given by $\omega_b - \omega_a$.

03.05 The sampling theorem

Most of the signals we encounter in the real world are continuous-time signals (e.g. voice, music, images). Digital signal processing algorithms are often applied to process these continuous-time signals. To do that, signals are first sampled and coded with an A/D converter, then they are digitally processed, and finally, they are converted back to the analog form with a D/A converter.
We want to study the relations that link the continuous-time signals with the corresponding discrete-time signals.
Let us consider a continuous-time signal $g_a(t)$. We assume to uniformly sample it with sampling period T. We obtain the sequence:

$$g(n) = g_a(nT) \qquad -\infty < n < +\infty.$$

The sampling frequency is $F_T = \frac{1}{T}$. The continuous-time Fourier transform, CTFT, of $g_a(t)$ is

$$G_a(j\Omega) = \int_{-\infty}^{+\infty} g_a(t) e^{-j\Omega t} \mathrm{d}t,$$

and the DTFT of $g(n)$ is

$$G(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} g(n) e^{-j\omega n}.$$

We want to find the relation that exists between $G_a(j\Omega)$ and $G(e^{j\omega})$.
Mathematically, we can consider the sampling operations as the product of a signal $g_a(t)$ and a periodic pulse train $p(t)$, where

$$p(t) = \sum_{n=-\infty}^{+\infty} \delta(t - nT),$$

composed of an infinite sequence of Dirac impulses uniformly spaced in time and separated by the period T.
The multiplication between $g_a(t)$ and $p(t)$ results in a continuous-time function

$$g_p(t) = g_a(t) \cdot p(t) = \sum_{n=-\infty}^{+\infty} g_a(nT)\, \delta(t - nT),$$

which is also composed of infinite Dirac impulses, placed at time $t = nT$, and weighted by the sample value $g_a(nT)$.
We will provide two expressions for the CTFT of $g_p(t)$. The first one is derived directly from the last expression of $g_p(n)$ considering that the CTFT of $\{\delta(t - nT)\}$ is $e^{-j\Omega nT}$.
For the linearity property of the CTFT:

$$G_p(j\Omega) = \sum_{n=-\infty}^{+\infty} g_a(nT) e^{-j\Omega nT}.$$

By comparing this relation with the expression of $G(e^{j\omega})$, we see that

$$G(e^{j\omega}) = G_p(j\Omega)|_{\Omega=\frac{\omega}{T}}$$

$$G_p(j\Omega) = G(e^{j\omega})\big|_{\omega=\Omega T}$$

The DTFT $G(e^{j\omega})$ coincides with the CTFT of $g_p(n)$, apart from a frequency axis normalization. This normalization maps the point at $\omega = 2\pi$ of $G(e^{j\omega})$ to the point at $\Omega_T = 2\pi F_T$ of $G_p(j\Omega)$.
In what follows, we find a second form for $G_p(j\Omega)$.
Assume you have two continuous time signals $a(t)$ and $b(t)$ with CTFT $A(j\Omega)$ and $B(j\Omega)$, respectively. The CTFT of the product $y(t) = a(t) \cdot b(t)$ is

$$Y(j\Omega) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} A(j\Psi) \cdot B\big(j(\Omega - \Psi)\big)\mathrm{d}\Psi$$

Proof:

$$Y(j\Omega) = \int_{-\infty}^{+\infty} a(t) \cdot b(t) e^{-j\Omega t}\mathrm{d}t =$$

$$= \int_{-\infty}^{+\infty} \frac{1}{2\pi}\int_{-\infty}^{+\infty} A(j\Psi)e^{j\Psi t}\mathrm{d}\Psi \cdot b(t)e^{-j\Omega t}\mathrm{d}t =$$

(obtained by replacing $a(t)$ with its ICTFT)

$$= \frac{1}{2\pi}\int_{-\infty}^{+\infty} A(j\Psi)\int_{-\infty}^{+\infty} b(t)e^{-j(\Omega-\Psi)t}\mathrm{d}t\mathrm{d}\Psi =$$

$$= \frac{1}{2\pi}\int_{-\infty}^{+\infty} A(j\Psi)B\big(j(\Omega - \Psi)\big)\mathrm{d}\Psi$$

Q.E.D.

We can use the property we have just proved to evaluate the CTFT of

$$g_p(t) = g_a(t) \cdot p(t).$$

We assume knowing the CTFT of $g_a(t)$. Let us compute the CTFT of the pulse train $p(t)$. In what follows, we prove that

$$P(j\Omega) = \frac{2\pi}{T}\sum_{k=-\infty}^{+\infty} \delta(\Omega - k\frac{2\pi}{T})$$

i.e., the CTFT of $p(t)$ is also a uniformly spaced pulse train.

Proof:
Note that the pulse train $p(t)$ is a periodic signal with a period of T. Thus, we can expand $p(t)$ with the Fourier series:

$$p(t) = \sum_{k=-\infty}^{+\infty} c_k e^{j\frac{2\pi}{T}kt}$$

with

$$c_k = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} p(t)e^{-j\frac{2\pi}{T}kt}\mathrm{d}t =$$

$$= \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} \delta(t)e^{-j\frac{2\pi}{T}kt}\mathrm{d}t =$$

(because in the interval $[-\frac{T}{2}, +\frac{T}{2}]$ only one pulse, $\delta(t)$, is different from 0)

$$= \frac{1}{T} e^{-j\frac{2\pi}{T}k0} = \frac{1}{T}$$

Thus,

$$p(t) = \frac{1}{T} \sum_{k=-\infty}^{+\infty} e^{j\frac{2\pi}{T}kt}$$

and for the linearity of the CTFT

$$P(j\Omega) = \frac{1}{T} \sum_{k=-\infty}^{+\infty} \text{CTFT}\left\{e^{j\frac{2\pi}{T}kt}\right\}.$$

We can easily verify that

$$\text{CTFT}\left\{e^{j\frac{2\pi}{T}kt}\right\} = 2\pi\delta(\Omega - \frac{2\pi}{T}k)$$

Indeed, the IDFT of the right-hand side is

$$\frac{1}{2\pi} \int_{-\infty}^{+\infty} 2\pi\delta(\Omega - \frac{2\pi}{T}k) e^{j\Omega t} \mathrm{d}\Omega = e^{j\frac{2\pi}{T}kt}$$

(for the properties of the Dirac function).
Replacing CTFT $\left\{e^{j\frac{2\pi}{T}kt}\right\}$ in the expression of $P(j\Omega)$, we have

$$P(j\Omega) = \frac{2\pi}{T} \sum_{k=-\infty}^{+\infty} \delta(\Omega - k\frac{2\pi}{T}).$$

We can now compute $G_p(j\Omega)$

$$G_p(j\Omega) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} G_a(j\Psi) \cdot P\big(j(\Omega - \Psi)\big) \mathrm{d}\Psi =$$

$$= \frac{1}{2\pi} \int_{-\infty}^{+\infty} G_a(j\Psi) \cdot \frac{2\pi}{T} \sum_{k=-\infty}^{+\infty} \delta(\Omega - \Psi - k\frac{2\pi}{T}) \mathrm{d}\Psi =$$

$$= \frac{1}{T} \sum_{k=-\infty}^{+\infty} \int_{-\infty}^{+\infty} G_a(j\Psi) \cdot \delta(\Omega - \Psi - k\frac{2\pi}{T}) \mathrm{d}\Psi =$$

$$= \frac{1}{T} \sum_{k=-\infty}^{+\infty} G_a\big(j(\Omega - k\frac{2\pi}{T})\big) =$$

$$= \frac{1}{T} \sum_{k=-\infty}^{+\infty} G_a\big(j(\Omega - k\Omega_T)\big)$$

with $\Omega_T = \frac{2\pi}{T}$.
Consequently,

$$G(e^{j\omega}) = G_p(j\Omega)|_{\Omega=\frac{\omega}{T}} =$$

$$= \frac{1}{T} \sum_{k=-\infty}^{+\infty} G_a\big[j\big(\frac{\omega}{T} + k\Omega_T\big)\big] =$$

$$= \frac{1}{T} \sum_{k=-\infty}^{+\infty} G_a[j(\frac{\omega + 2\pi k}{T})].$$

The DTFT is given by the periodic repetition, with a period of 2π, of the continuous spectrum $G_a\left(j\frac{\omega}{T}\right)$. $G_a\left(j\frac{\omega}{T}\right)$ is identical to $G_a(j\Omega)$, but the frequency axis has been normalized such that $\omega = 2\pi$ corresponds to the angular sampling frequency $\Omega_T = \frac{2\pi}{T}$.
In order to avoid any overlap between the repeated spectra, it must be

$$G_a\left(j\frac{\omega}{T}\right) = 0 \quad \text{for} \quad |\omega| > \pi,$$

or, equivalently,

$$G_a(j\Omega) = 0 \quad \text{for} \quad |\Omega| > \frac{\pi}{T} = \frac{\Omega_T}{2}.$$

If this condition is satisfied, the discrete spectrum reproduces in the band $[-\pi, +\pi]$ the continuous spectrum.
If this condition is not satisfied, we have a distortion caused by the overlap of the tails of the spectrum. This distortion is called *aliasing*.
The frequency $\frac{1}{2T} = \frac{F_T}{2}$ is called *Nyquist frequency*.
Similarly, the angular frequency $\frac{\Omega_T}{2}$ is called *Nyquist angular frequency*.
Let us assume, for example, that our signal occupies the band $[-\Omega_m, +\Omega_m]$.
If $\Omega_T \geq 2\Omega_m$, the different repetitions of the spectrum do not overlap.
If $\Omega_T < 2\Omega_m$, the repetitions of the spectrum overlap, and we have an aliasing error.
See Figure 03.01.
If the continuous-time signal spectrum is preserved in the discrete domain (i.e., if we do not have aliasing), we can reconstruct the original signal from the samples $g(n) = g_a(nT)$.
For this purpose, let us build the signal:

$$g_p(t) = \sum_{n=-\infty}^{+\infty} g(n)\delta(t - nT) =$$

$$= \sum_{n=-\infty}^{+\infty} g_a(nT)\delta(t - nT)$$

We know its CTFT:

$$G_p(j\Omega) = \frac{1}{T} \sum_{k=-\infty}^{+\infty} G_a\left[j(\Omega + k\Omega_T)\right].$$

This continuous spectrum is given by the periodic repetition of the spectrum $G_a(j\Omega)$ with period Ω_T.
If the signal has been sampled with a frequency $\Omega_T > 2\Omega_m$, where Ω_m is the maximum signal frequency of $g_a(t)$, we do not have aliasing, and the repetitions of $G_a(j\Omega)$ do not overlap. Thus, we can faithfully reconstruct the signal $g_a(t)$ by passing the signal $g_p(t)$ through a filter (i.e., a device, a circuit) that lets the spectrum in the band $[-\Omega_m, \Omega_m]$ pass without any alteration, while it stops all signal components outside that frequency range. We will see that such a filter is an ideal low-pass filter with bandwidth Ω_m.
We have just proved the theorem that is at the basis of all signal processing and modern telecommunications, the sampling theorem.

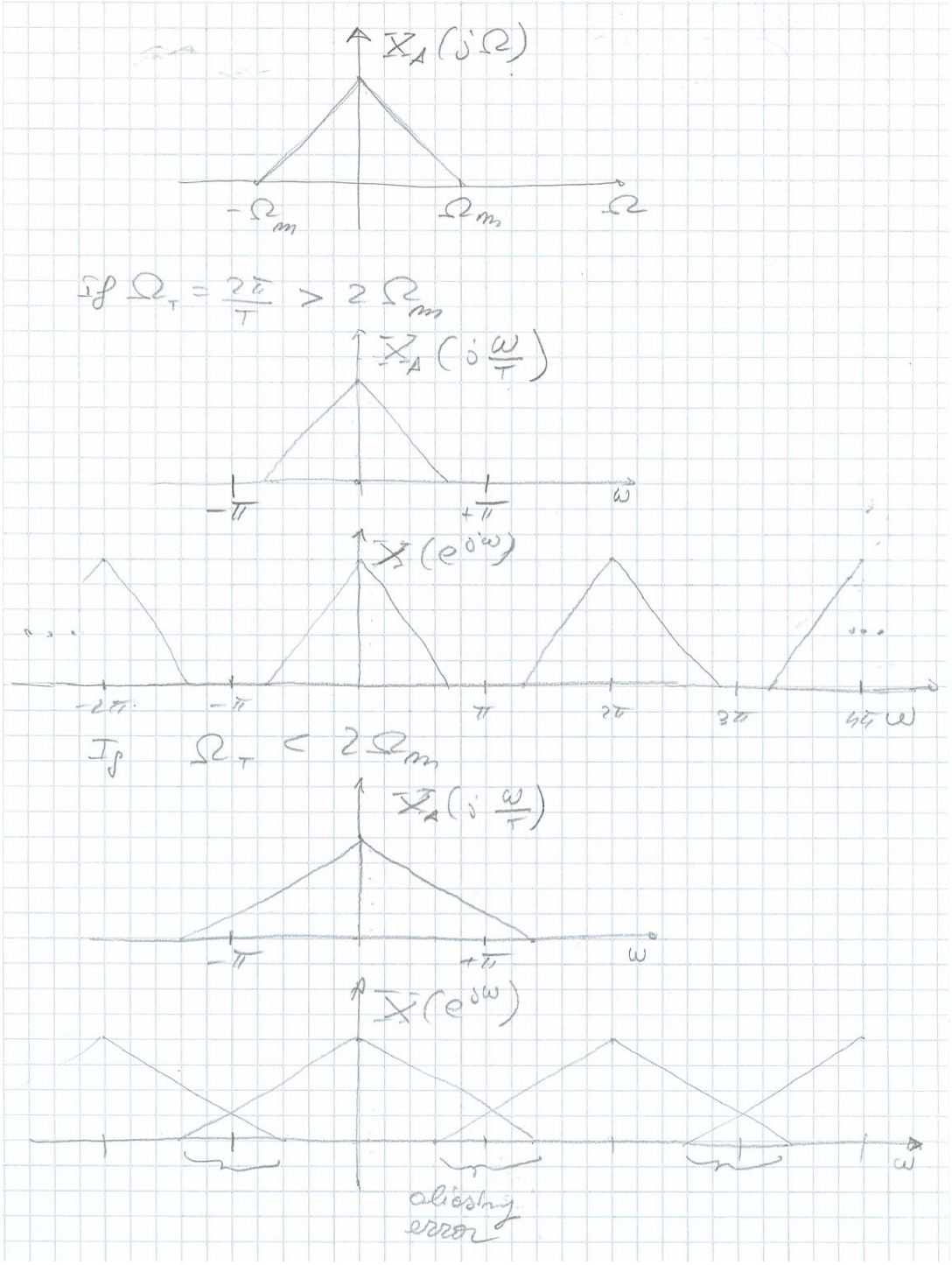


Figure 03.01: Illustration of sampling theorem.

The sampling theorem
(also called *Nyquist-Shannon theorem*)

> Let $g_a(t)$ be a band-limited signal, with $G_a(j\Omega) = 0$ for $|\Omega| > \Omega_m$. Then, $g_a(t)$ is uniquely determined by its samples $g_a(nT)$, (i.e., can be faithfully reconstructed by its samples $g_a(nT)$,)

$-\infty < n < +\infty$, if the angular sampling frequency

$$\Omega_T \geq 2\Omega_m$$

with $\Omega_T = \dfrac{2\pi}{T}$.

In other words, if we want to be able to recover the band-limited signal $g_a(t)$ from its samples, we must sample the signal with a frequency at least twice the signal bandwidth.
The signal can be recovered by generating the signal

$$g_p(t) = \sum_{n=-\infty}^{+\infty} g(n)\delta(t - nT)$$

and by filtering this signal with an ideal low-pass filter.
In general, real-world signals are not band-limited. They occupy an infinite band, but most of their energy is concentrated at the low frequencies.
To avoid distortions due to aliasing, the signal is typically filtered with a low-pass filter before sampling. This filter cuts off high frequencies and generates a band-limited signal. Such a filter is called an *anti-aliasing* filter. After filtering, the signal can be safely sampled with a frequency greater than $2\Omega_m$ and processed as desired.

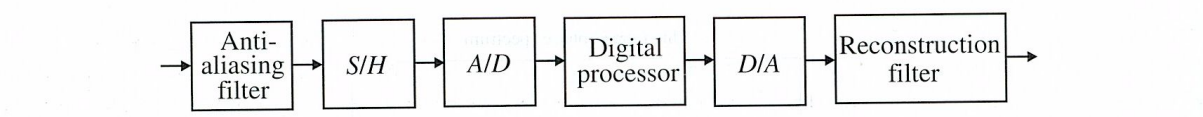


Figure 3.12: Block diagram representation of the discrete-time digital processing of a continuous-time signal.

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

Also the signal that comes out of the Digital to Analog (D/A) converter[2] is filtered with a low-pass filter, called the *reconstruction filter* or *anti-imaging filter*. In this way, all the frequencies (the images) outside the band of the original signal, $[-\Omega_m, +\Omega_m]$ are removed.
For example, to sample the voice in telephone applications, we exploit the fact that most of the voice energy falls between 300 and 3400 Hz. The voice is then sampled at 8kHz, which is greater than $2 \cdot 3.4$kHz.
In audio CDs, the musical signal has a bandwidth ranging from 0 to 20kHz and is sampled at 44.1kHz.
In DAT and DVDs, for the same musical signal, a larger bandwidth has been considered, and it has been sampled at 48kHz.

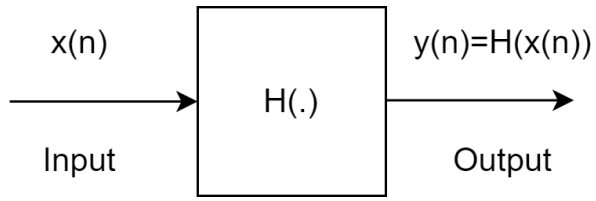
For more information study:

S. K. Mitra, ”Digital Signal Processing: a computer based approach,” 4th edition, McGraw-Hill, 2011
Chapter 3.1, pp. 89-93
Chapter 3.2-3.5, pp. 94-112
Chapter 3.8, pp. 115-124

[2] You can think of this signal as being $g_p(t)$, even though in reality, the Dirac pulses are replaced by rectangular pulses.

04 Discrete-time systems

A system is a device, a circuit, an algorithm implemented on a PC or on any other processor, which associates to the input signal (or the input signals) an output signal (or some output signals).
The function of a discrete-time system is to process one or more sequences, referred to as *input sequences,* with the aim of generating one or more sequences, known as *output sequences.* These output sequences are expected to exhibit certain desired properties or emphasize specific information from the input signals.
In most cases, our systems have a single input and a single output.

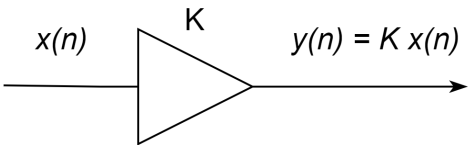


The input signal is typically denoted as $x(n)$, and the corresponding output signal is represented as $y(n)$. Mathematically, a discrete-time system is defined by an operator $H(\cdot)$ that maps each input sequence $x(n)$ to the corresponding output sequence $y(n)$.
In discrete-time systems of practical interest, all signals are digital signals (with discrete-time and discrete amplitude), and the operations on these signals also result in digital signals. Such systems are commonly referred to as *digital filters.* Throughout this discussion, we will interchangeably use the terms 'discrete-time system,' 'discrete system,' and 'digital filter'.
The term 'filter' originates from these systems' initial application in filtering the spectrum of a signal. The system was designed to leave certain frequency components of the signal unaltered while removing, or 'filtering,' other undesired frequencies—similarly to a mechanical filter.

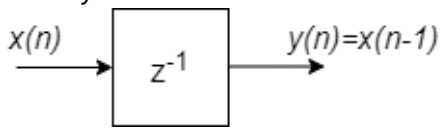
04.01 Examples of simple systems

Examples of basic discrete-time systems are the following:

Constant multiplier:



Unit delay:



Accumulator:

$$y(n) = \sum_{l=-\infty}^{n} x(l)$$

Every time-instant n, the output $y(n)$ is the sum of the input $x(n)$ at time n and of all past input samples. The input-output relationship can also be expressed alternatively as:

$$y(n) = \sum_{l=-\infty}^{n-1} x(l) + x(n) = y(n-1) + x(n).$$

In this form, the output at time n is the sum of the input sample at time n and of the previous value of the output sample, $y(n-1)$.
Another alternative form is the following:

$$y(n) = \sum_{l=-\infty}^{-1} x(l) + \sum_{l=0}^{n} x(l) = y(-1) + \sum_{l=0}^{n} x(l).$$

This form is used when the input signal $x(n)$ is a causal signal (i.e., defined only for $n \geq 0$) and $y(-1)$ is called *initial condition*.

Moving average:

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

$y(n)$ is the mean average value of the last M samples of $x(n)$. This is a very simple filter, commonly used in practice.
It's worth noting that the expression for the moving average can be expressed in recursive form as follows:

$$y(n) = \frac{1}{M} \left(\sum_{l=0}^{M-1} x(n-l) + x(n-M) - x(n-M) \right) =$$

$$= \frac{1}{M} \left(\sum_{l=1}^{M} x(n-l) + x(n) - x(n-M) \right) =$$

$$= y(n-1) + \frac{1}{M} \left(x(n) - x(n-M) \right).$$

It's possible to describe the same system in different ways, corresponding to various implementations. Further, we will observe that the moving average filter behaves like a low-pass filter, with a passband inversely proportional to M (larger M results in a lower passband).

Exponentially weighted running average filter:

$$y(n) = \alpha y(n-1) + x(n)$$

with $0 < \alpha < 1$.

This filter calculates the mean of past signal samples, with a greater emphasis on the most recent samples of $x(n)$. Through successive substitutions, we find that

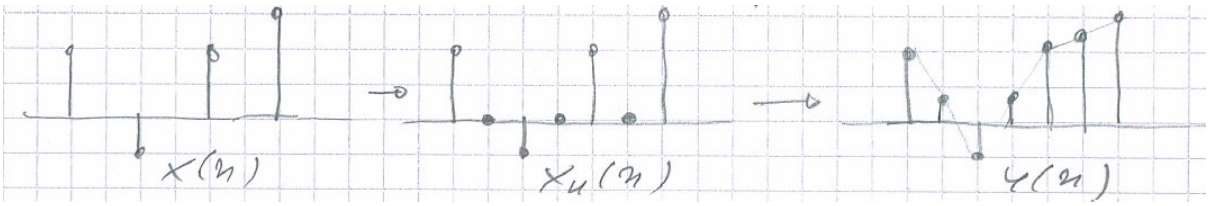
$$y(n) = \sum_{l=0}^{+\infty} \alpha^{n-l} x(n-l).$$

Here, the samples are multiplied by an exponential weight that gradually diminishes as we move away from $x(n)$.

Interpolation filter:

Suppose we have a signal sampled at a frequency of f_c. To obtain the samples of the same signal at a sampling frequency of $2f_c$, we can take the sequence sampled at f_c, insert a zero between each pair of samples, and then filter the resulting sequence with an interpolation filter. The interpolation filter replaces all zero samples with the mean value of the preceding and succeeding samples:

$$y(n) = x_u(n) + \frac{1}{2}\left(x_u(n-1) + x_u(n+1)\right)$$



The technique can be easily extended for interpolation factors of 3, 4, or even higher. For an interpolation factor of 3, the formula becomes:

$$y(n) = x_u(n) + \frac{2}{3}\left(x_u(n-1) + x_u(n+1)\right) + \frac{1}{3}\left(x_u(n-2) + x_u(n+2)\right)$$

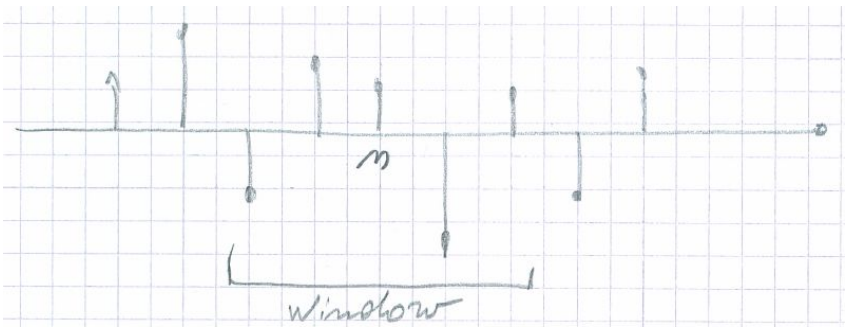
These filters find applications in image processing, particularly for enlarging images. For example, they are used to transition from an image with $N \times N$ pixels to an enlarged image with $2N \times 2N$ pixels.

Median filter:

Consider a set of $2K+1$ numbers. Ordering these numbers by their values, the 'median' is the number at the central position, precisely at position K when counted from 0. Therefore, there are K numbers lower than or equal to the median and K numbers greater than or equal to the median.

The median filter is created by sliding a window of length $2K+1$ over the signal $x(n)$ and selecting the median value within this window:

$$y(n) = \text{med}\left\{x(n-K), x(n-K+1), \ldots, x(n-1), x(n), x(n+1), \ldots, x(n+K)\right\}.$$



If the signal has a finite length, it is extended with zeros in both directions.

$$\{\dots,0,1,2,1,0,\dots\} \overset{\text{med}_3}{\longleftrightarrow} \{\dots,0,1,1,1,0,\dots\}$$

The median filter is widely employed in image processing to eliminate impulsive noises. Notably, it possesses the property of preserving edges, a characteristic that contrasts with the smoothing effect on borders when using low-pass filters like the moving average.

04.02 Classification of discrete-time systems

A discrete-time system is termed *static* or *without memory* if, for every input sequence $\{x(n)\}$ and at every time instant n, the output $y(n)$ depends solely on the input sample at that time, $x(n)$. It does not depend on past or future output samples.
An example of a static system is the multiplier for a constant.
In contrast, a discrete-time system, where the output signal depends on both past and future input samples, is termed *dynamic*.

A discrete-time system is termed *linear* if it satisfies the superposition principle: for any pair of input signals $x_1(n)$ and $x_2(n)$, and for any arbitrary constants a_1 and a_2, if $y_1(n)$ and $y_2(n)$ are the responses to $x_1(n)$ and $x_2(n)$, then the response to the input signal $x(n) = a_1x_1(n) + a_2x_2(n)$ is $a_1y_1(n) + a_2y_2(n)$.

$$x_1(n) \longrightarrow y_1(n)$$

$$x_2(n) \longrightarrow y_2(n)$$

$$a_1x_1(n) + a_2x_2(n) \longrightarrow a_1y_1(n) + a_2y_2(n)$$

Note that this property must hold for every choice of $x_1(n)$, $x_2(n)$, a_1, a_2.
The superposition principle can be separated into two parts:

- Multiplicative property
  It the response to $x(n)$ is $y(n)$, then for all constants K the response to $Kx(n)$ is $Ky(n)$:

  $$x(n) \longrightarrow y(n)$$

  $$Kx(n) \longrightarrow Ky(n)$$

- Additive property
  If the responses to $x_1(n)$ and $x_2(n)$ are $y_1(n)$ and $y_2(n)$, respectively, then the response to $x_1(n) + x_2(n)$ is $y_1(n) + y_2(n)$:

  $$x_1(n) \longrightarrow y_1(n)$$

  $$x_2(n) \longrightarrow y_2(n)$$

  $$x_1(n) + x_2(n) \longrightarrow y_1(n) + y_2(n)$$

Every system that does not satisfy the superposition principle is called *nonlinear*.

Examples: Let us first consider the accumulator:

$$y_1(n) = \sum_{m=-\infty}^{n} x_1(m)$$

$$y_2(n) = \sum_{m=-\infty}^{n} x_2(m)$$

The response to $a_1x_1(n) + a_2x_2(n)$ is

$$y(n) = \sum_{m=-\infty}^{n} (a_1x_1(m) + a_2x_2(m)) =$$

$$= a_1 \sum_{m=-\infty}^{n} x_1(m) + a_2 \sum_{m=-\infty}^{n} x_2(m) =$$

$$= a_1y_1(n) + a_2y_2(n)$$

Thus, the accumulator in this form is linear.
Let us now consider the alternative form of the accumulator:

$$y_1(n) = y_1(-1) + \sum_{m=0}^{n} x_1(m)$$

$$y_2(n) = y_2(-1) + \sum_{m=0}^{n} x_2(m)$$

The response to $a_1x_1(n) + a_2x_2(n)$ is

$$y(n) = y(-1) + \sum_{m=0}^{n} (a_1x_1(m) + a_2x_2(m)) =$$

$$= y(-1) + a_1 \sum_{m=0}^{n} x_1(m) + a_2 \sum_{m=0}^{n} x_2(m)$$

On the contrary, we have

$$a_1y_1(n) + a_2y_2(n) = a_1y_1(-1) + a_2y_2(-1) + a_1 \sum_{m=0}^{n} x_1(m) + a_2 \sum_{m=0}^{n} x_2(m)$$

The two expressions are equal if and only if:

$$a_1y_1(-1) + a_2y_2(-1) = y(-1),$$

but this condition must be satisfied for all a_1, a_2, $x_1(n)$, $x_2(n)$, and for all $y_1(-1)$, $y_2(-1)$, $y(-1)$. Since $y_1(-1)$, $y_2(-1)$, $y(-1)$ are initialization constants, this condition is not generally satisfied unless we assume the system to be initially at rest, i.e., with $y_1(-1) = y_2(-1) = y(-1) = 0$. If the system has zero initial conditions, it is linear. Conversely, if it has an initial condition different from zero, it is a nonlinear system.

Another example of nonlinear system is the median filter.
Let us consider a median filter of length 3.

$$\{x_1(n)\} = \{3, 4, 5\} \longrightarrow \{y_1(n)\} = \{3, 4, 4\}$$

$$\{x_2(n)\} = \{2, -1, -1\} \longrightarrow \{y_2(n)\} = \{0, -1, -1\}$$

$$\{x_1(n)\} + \{x_2(n)\} = \{5, 3, 4\} \longrightarrow \{y_1(n)\} = \{3, 4, 3\}$$

But $\{3, 4, 3\} \neq \{y_1(n)\} + \{y_2(n)\} = \{3, 3, 3\}$.

A system is termed *time-invariant* or *shift-invariant* if, for any input $x(n)$ with a response $y(n)$ and for any constant $k \in \mathbb{Z}$, the response to $x(n-k)$ is $y(n-k)$:

$$x(n) \longrightarrow y(n)$$

$$x(n-k) \longrightarrow y(n-k)$$

Note that this property must hold for every possible choice of $x(n)$ and k.

In the following, we will particularly focus on *Linear Time-Invariant (LTI)* discrete-time systems. LTI systems exhibit both linearity and time-invariance properties. These characteristics make them straightforward to analyze and characterize, facilitating easy design. Consequently, LTI systems find widespread use in processing digital signals.
For these systems, we can explicitly express the rule $H(\cdot)$ that maps the input signal to the output signal. In other words, for LTI systems, we can formulate a mathematical rule that computes the output signal samples based on the knowledge of the input signal samples. Notably, the concepts of impulse response and convolution sum play a crucial role in this context.

04.03 Impulse response and convolution sum

The *impulse response* of a LTI system is defined as the system's response to the unit impulse input signal:

$$x(n) = \delta(n) \longrightarrow y(n) = h(n)$$

We have observed that every sequence $x(n)$ can be represented as the sum of an infinite number of impulses, appropriately scaled:

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

But

$$\delta(n) \longrightarrow h(n)$$

$$\delta(n-m) \longrightarrow h(n-m)$$

(for the time-invariance property)

$$x(m)\delta(n-m) \longrightarrow x(m)h(n-m)$$

(for the multiplicative property)

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

(for the additive property).
Thus, in an LTI system, the output sequence can be calculated from the input sequence using the following relation:

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

This sum is known as the *convolution sum*. Additionally, we say that the signal $x(n)$ is *convolved* with $h(n)$.
The impulse response is sufficient to completely describe LTI systems. Knowing $h(n)$ allows us to determine $y(n)$ for any input $x(n)$.

Properties of the convolution sum

Commutative property:

$$x(n) \circledast h(n) = h(n) \circledast x(n)$$

Proof:

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

Let us consider the change of variable $m' = n - m$, i.e., $m = n - m'$

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

Q.E.D.

Physical interpretation:

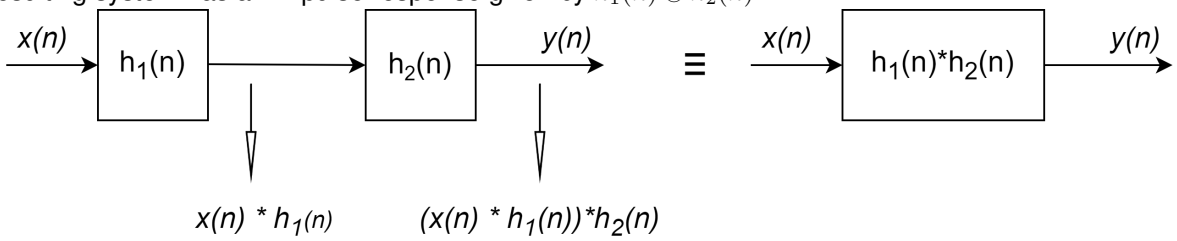
The system with input $x(n)$ and impulse response $h(n)$ has the same response $y(n)$ as the system with input $h(n)$ and impulse response $x(n)$:

Associative property:

$$[x(n) \circledast h_1(n)] \circledast h_2(n) = x(n) \circledast [h_1(n) \circledast h_2(n)]$$

Physical interpretation:

If we consider the cascade of two systems with impulse responses $h_1(n)$ and $h_2(n)$, respectively, the resulting system has an impulse response given by $h_1(n) \circledast h_2(n)$:



Distributive property:

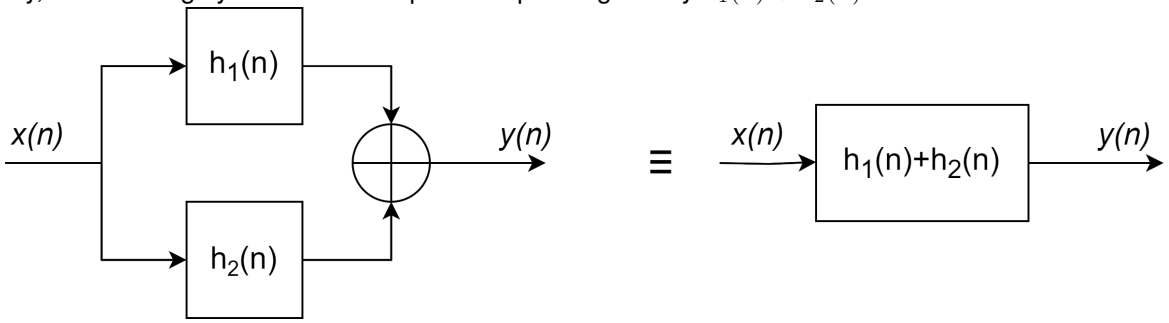
$$[x_1(n) + x_2(n)] \circledast h(n) = x_1(n) \circledast h(n) + x_2(n) \circledast h(n)$$

and also

$$x(n) \circledast [h_1(n) + h_2(n)] = x(n) \circledast h_1(n) + x(n) \circledast h_2(n)$$

Physical interpretation of the last relation:

If we consider the parallel connection of two systems with impulse responses $h_1(n)$ and $h_2(n)$, respectively, the resulting system has an impulse response given by $h_1(n) + h_2(n)$:



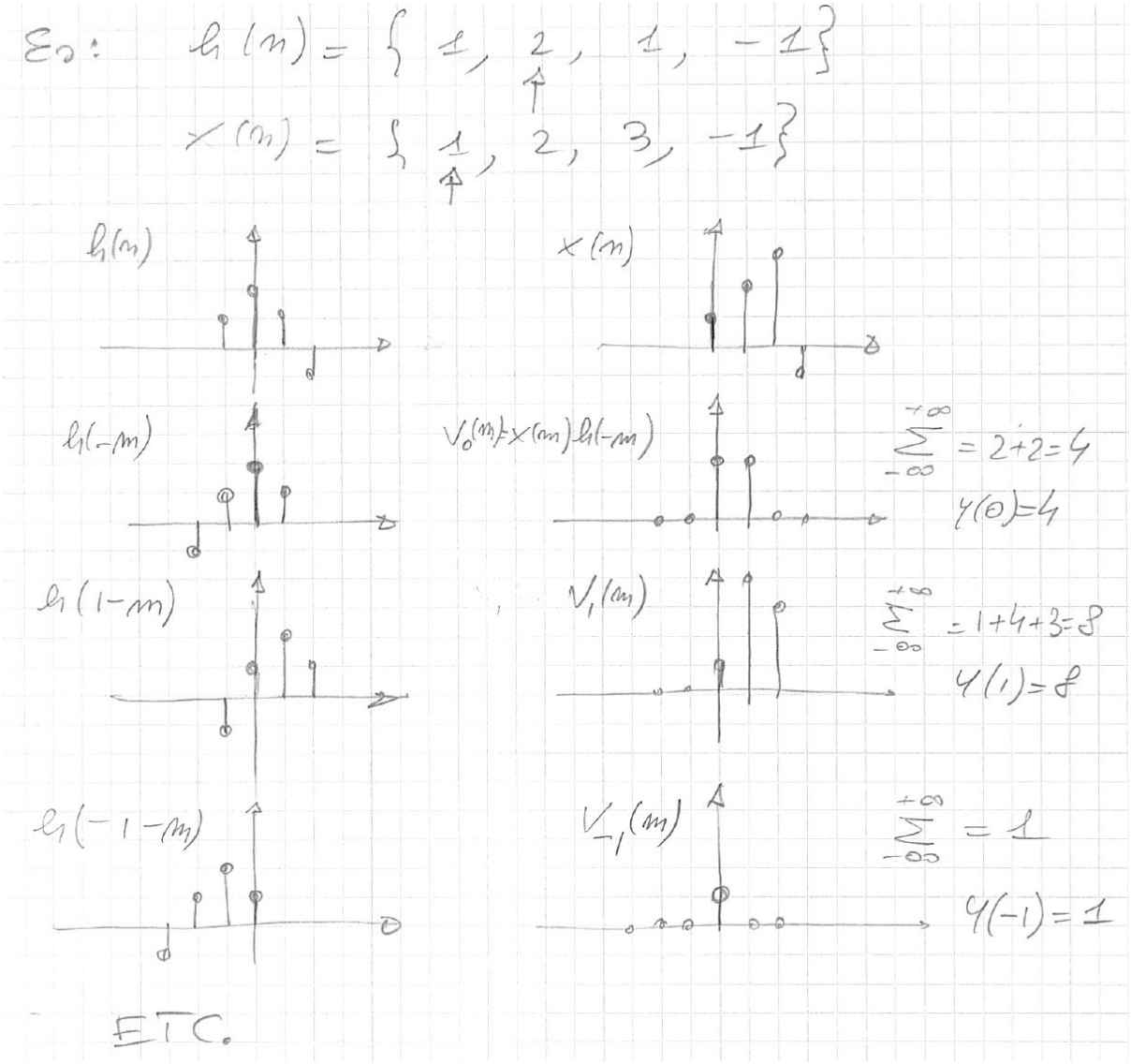
Computation of the convolution sum

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

We can compute $y(n_0) = y(n)\Big|_{n=n_0}$ by means of the following operations:

- Fold the sequences $h(m)$ to obtain $h(-m)$.
- Time-shift $h(-m)$ of n_0 samples to the right for $n_0 > 0$ (time delay), or to the left by $|n_0|$ samples for $n_0 < 0$ (time advancement), resulting in $h(n_0 - m)$.
- Perform element-wise multiplication between $x(m)$ and $h(n_0 - m)$ to get $V_{n_0}(m) = x(m)h(n_0 - m)$.
- Sum of all the terms of $V_{n_0}(m)$.

Conceptually, this method can be applied to any pair of sequences $x(n)$ and $h(n)$. However, in practice, this approach is feasible only when at least one of the two sequences has a finite length.



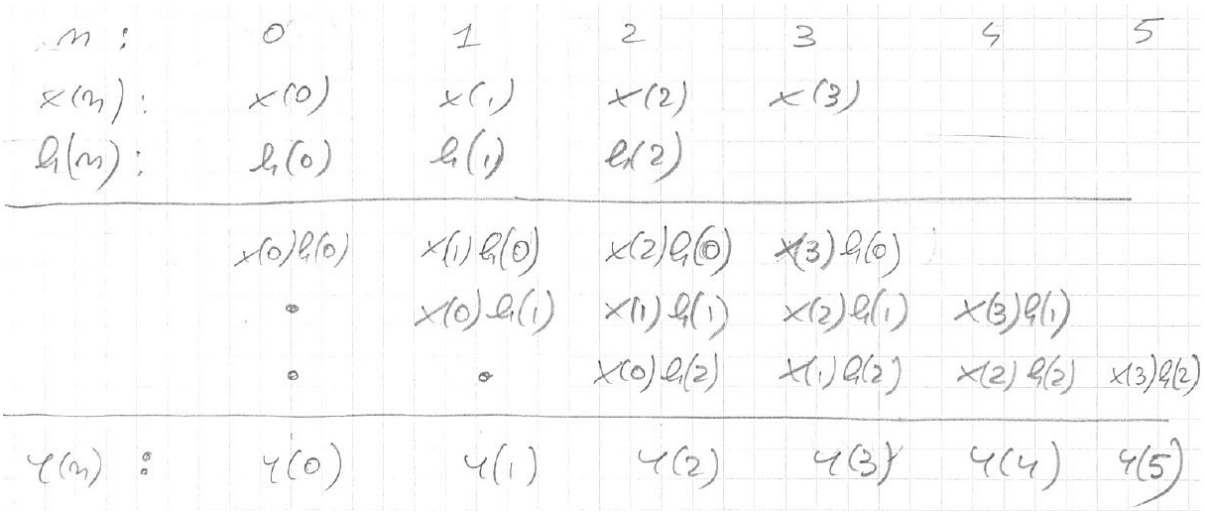
Tabular method for computing the convolution sum.

The convolution sum of two finite-length sequences can be computed using a tabular method, similar to the technique used for elementary school multiplication.

Let us consider the convolution of a sequence $\{x(n)\}$ with a length of 4 for $0 \leq n \leq 3$, and a sequence $\{h(n)\}$ with a length of 3 for $0 \leq n \leq 2$. The resulting sequence has a length of $4 + 3 - 1 = 6$.

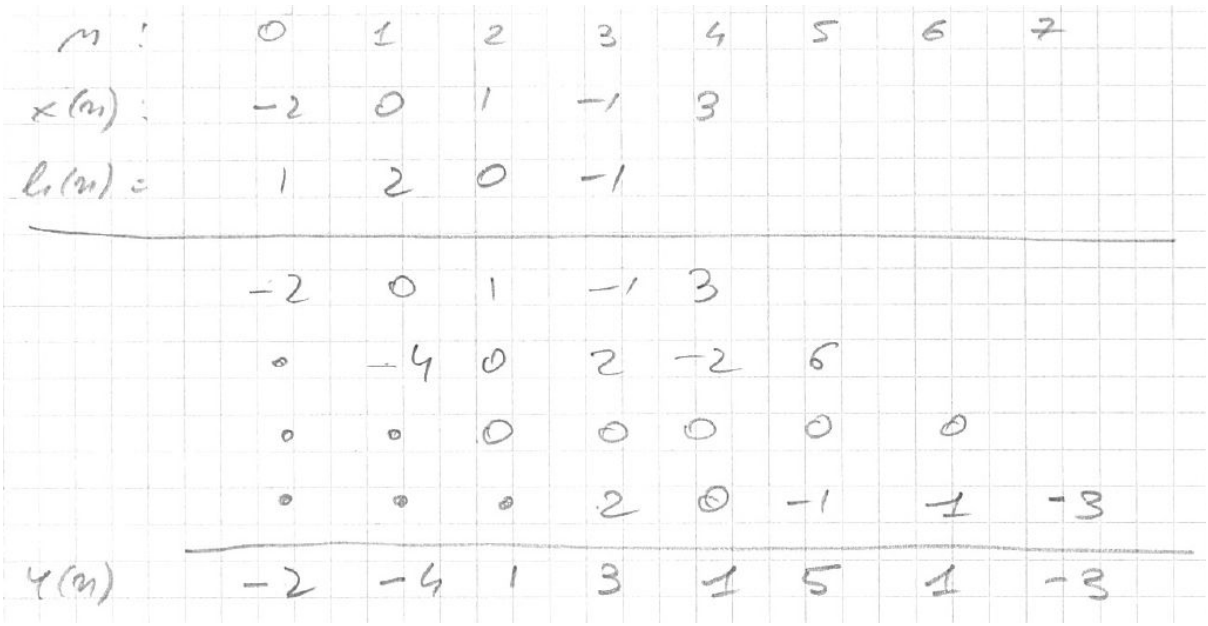
$$y(n) = x(n) \circledast h(n), \qquad 0 \leq n \leq 5.$$

The samples of the two sequences are multiplied using the classical tabular method of multiplications, without considering carry operations between columns:



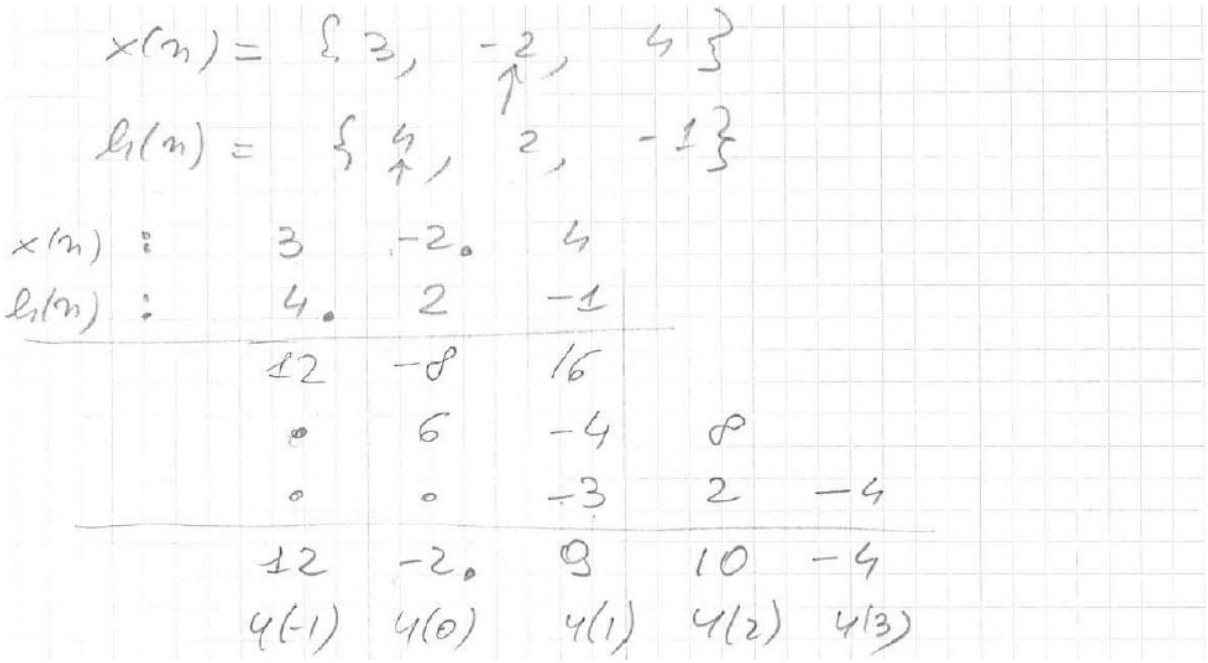
To begin, multiply $h(0)$ with each element of $x(n)$ and fill the first row of the table.
Next, compute the product of $h(1)$ with each element of $x(n)$ and fill the second row of the table, moving one position to the right.
Following that, compute the product of $h(2)$ with each element of $x(n)$ and fill the third row of the table, moving two positions to the right.
Continue this process for all the samples of $h(n)$
Finally, add all the terms along the columns to obtain $y(n)$.
Note that along the columns, we perform the folding and time-shifting by n_0 samples of $x(n)$, i.e., $x(n_0 - m)$.

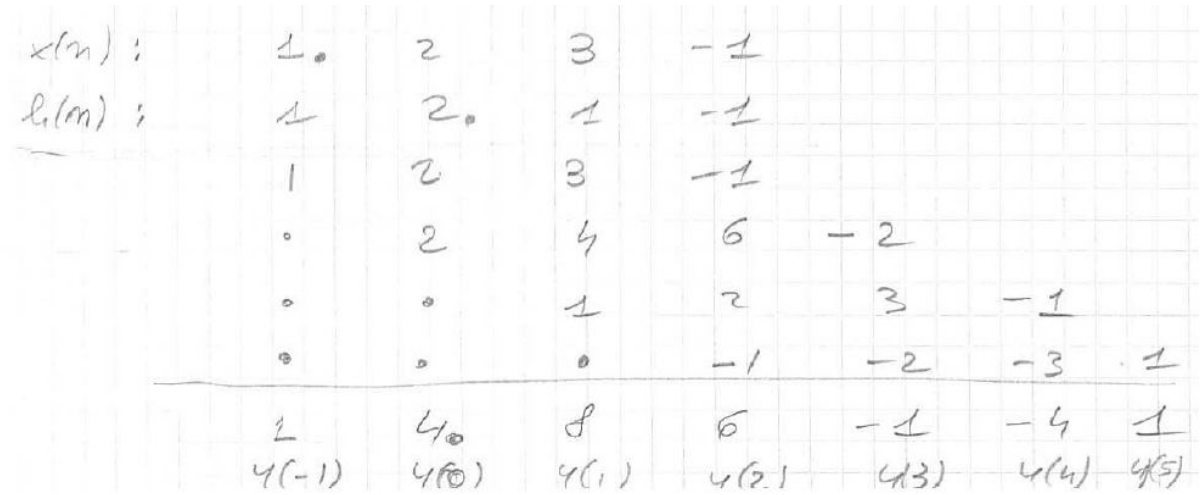
Example:



The tabular method can also be employed to evaluate the convolution of finite-length two-sided sequences. In this case, we can use the decimal point to mark the position of $n = 0$. Determining the position of $y(0)$ can be done by inserting the decimal point in the output sequence following the classical rule of multiplication between decimal numbers. Alternatively, it can be determined by recognizing that the number of terms to the left of $y(0)$ is simply the sum of the number of terms to the left of $x(0)$ and the number of terms to the left of $h(0)$.

Example:





Causal linear time-invariant systems.

A discrete-time system is termed *causal* if, at every time instant n, the output $y(n)$ depends solely on the present and past samples of $x(n)$ (i.e., $x(n)$, $x(n-1)$, $x(n-2)$, etc.), while it remains independent of the future samples of the signal ($x(n+1)$, $x(n+2)$, $x(n+3)$, etc.).

Systems that are not causal are referred to as *noncausal*.

In real-time digital signal processing systems, the observation of future samples of the signal is not possible, rendering noncausal systems unrealizable. Hence, the property of causality is also known as the *realizability property*.

Property: An LTI system is causal if and only if its impulse response is zero for $n < 0$:

$$\text{causality} \iff h(n) = 0 \quad \forall n < 0.$$

Proof:

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

$$= \sum_{m=-\infty}^{-1} h(m) \cdot x(n-m) + \sum_{m=0}^{+\infty} h(m) \cdot x(n-m)$$

The first term depends on the future samples of $x(n)$, while the second term depends only on the present and past samples of $x(n)$. Thus, the system is causal if and only if $h(n) = 0 \ \ \forall n < 0$.

Q.E.D.

An example of a noncausal system is the linear interpolator.

In a causal system, the convolution sum is given by

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

$$= \sum_{m=-\infty}^{n} x(m)h(n-m)$$

In analogy to the causality property of LTI systems, sequences that are zero for all $n < 0$ are referred to as causal.

If the input of a causal LTI system is a causal sequence, the boundaries of the convolution sum are further reduced since we have:

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

Stability of LTI discrete-time systems

A discrete time system is defined *BIBO stable* (Bounded Input Bounded Output stable) if, for every bounded input signal $x(n)$, the output signal $y(n)$ is bounded.

If the input signal is bounded, there exists a constant M_x such that

$$|x(n)| \leq M_x < +\infty \quad \forall n.$$

If the system is BIBO stable, there must exist a constant M_y such that, for any $|x(n)| \leq M_x$,

$$|y(n)| \leq M_y < +\infty \quad \forall n.$$

Property: A LTI discrete-time system is BIBO stable if and only if

$$\sum_{n=-\infty}^{+\infty} |h(n)| < +\infty,$$

i.e., if and only if the impulse response is absolutely summable.

Proof:

First, let's prove that this condition is sufficient for BIBO stability.

If $x(n)$ is bounded, there exists M_x such that

$$|x(n)| \leq M_x < +\infty \quad \forall n.$$

Thus,

$$|y(n)| = \left| \sum_{m=-\infty}^{+\infty} h(m)x(n-m) \right| \leq$$
$$\leq \sum_{m=-\infty}^{+\infty} |h(m)x(n-m)| \leq$$
$$\leq \sum_{m=-\infty}^{+\infty} |h(m)| \, M_x$$

If we define $M_y = M_x \sum_{m=-\infty}^{+\infty} |h(m)|$, it is proved that, if $h(n)$ is absolutely summable, there exists a constant M_y such that

$$|y(n)| \leq M_y < +\infty \quad \forall n.$$

Now, let's prove that it is also a necessary condition. To this purpose, let us assume

$$\sum_{m=-\infty}^{+\infty} |h(m)| = +\infty$$

and let us demonstrate that it is always possible to find a bounded input whose output is not bounded. If $h(n) \in \mathbb{R}$, one of such signals is

$$x(n) = \text{sign}[h(-n)] = \begin{cases} +1 & h(-n) \geq 0 \\ -1 & h(-n) < 0 \end{cases}$$

Surely, $x(n)$ is bounded because $|x(n)| = 1$. If we consider the output of our system for $n = 0$:

$$y(0) = \sum_{m=-\infty}^{+\infty} h(m)x(0-m) =$$

$$= \sum_{m=-\infty}^{+\infty} h(m)\text{sign}[h(m)] =$$

$$= \sum_{m=-\infty}^{+\infty} |h(m)| = +\infty$$

Here, for simplicity, we have considered a real $h(n)$. However, everything holds true for a complex $h(n)$ as well. It is sufficient to consider

$$x(n) = \frac{h^*(-n)}{|h(-n)|}.$$

Finite Impulse Response and Infinite Impulse Response LTI systems

An LTI discrete-time system is termed a *Finite Impulse Response (FIR)* system if its impulse response has a finite length, i.e.

$$h(n) = 0 \quad \forall n < N_1 \text{ or } n > N_2,$$

with $N_1 \leq N_2$. $h(n)$ has only $N = N_2 - N_1 + 1$ elements different from 0. In this case, the convolution sum simplifies to

$$y(n) = \sum_{m=N_1}^{N_2} h(m)x(n-m).$$

As this sum is finite, it can be directly used to compute $y(n)$.

For a causal FIR system of length N:

$$h(n) = 0 \quad \forall n < 0 \text{ or } n \geq N,$$

and the output is given by

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

A system whose impulse response has infinite length (i.e., contains an infinite number of elements different from 0) is referred to as an *Infinite Impulse Response (IIR)* system. In the case of a causal IIR system, the output is given by

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

While FIR systems can be directly implemented using the convolution sum, IIR systems cannot be realized through the convolution sum due to the requirement of an infinite number of additions, multiplications, and memory elements.

In practice, in digital signal processing, we often focus on a subclass of LTI and causal discrete-time systems. This subclass comprises all systems that can be represented by a finite difference equation with constant coefficients, i.e., they can be expressed in the following form:

$$y(n) = \sum_{i=0}^{M} b_i x(n-i) - \sum_{i=1}^{N} a_i y(n-i)$$

for all $n \geq 0$.

For this class of systems, the output can be computed directly from some past samples of the input and output signals. These systems are causal, as they involve only the past samples and the present sample of the input signal.

To compute $y(n)$ from the input signal $x(n)$, knowledge of the N initial conditions, $y(-1)$, $y(-2)$, ..., $y(-N)$, is required. If these N initial conditions are all zero,

$$y(-1) = y(-2) = \ldots = y(-N) = 0,$$

the system is termed *initially at rest.*

The class of systems that can be described by a finite difference equation with constant coefficients includes all causal FIR systems and a subset of causal LTI IIR systems.

There are causal IIR systems that cannot be described by a finite difference equation. For instance, the system with impulse response:

$$h(n) = \begin{cases} \dfrac{1}{n^2} & n > 0 \\ 0 & n \leq 0 \end{cases}$$

is a causal, BIBO-stable system, but it lacks a representation in terms of a finite difference equation.

04.04 Frequency response

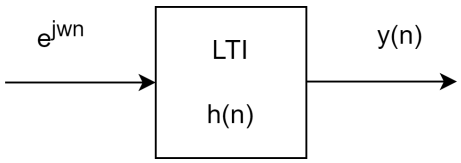
We have seen that every sequence can be represented in the time domain as the weighted sum of an infinite number of unit pulses, shifted in time:

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

This representation leads to an important consequence – the characterization of LTI systems using the impulse response and the convolution sum.

We have also seen that sequences can be represented by means of a weighted sum of an infinite number of complex exponential sequences $\{e^{j\omega n}\}$. This representation leads to another description of LTI systems through the so-called *Frequency Response*.

Let us consider an LTI system with impulse response $h(n)$ and let us excite the system with a complex exponential sequence $e^{j\omega n}$ with $-\infty < n < +\infty$.



$$y(n) = \sum_{m=-\infty}^{+\infty} h(m)e^{j\omega(n-m)} =$$
$$= \sum_{m=-\infty}^{+\infty} h(m)e^{-j\omega m}e^{j\omega n} =$$
$$= H(e^{j\omega}) \cdot e^{j\omega n}.$$

Thus, if we apply a complex exponential sequence $e^{j\omega n}$ to the input of our system, the output is the same exponential sequence multiplied by the complex constant $H(e^{j\omega})$.
$H(e^{j\omega})$ is the Discrete-Time Fourier Transform (DTFT) of the impulse response $h(n)$ and is referred to as the *Frequency Response* of the LTI system.
$|H(e^{j\omega})|$ is termed the *amplitude response* (or *magnitude response*), and arg$H(e^{j\omega})$ is termed the *phase response* of the LTI system.
The frequency response completely characterizes the response of an LTI system in the frequency domain. Indeed,

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

Proof:

$$Y(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} y(n)e^{-j\omega n} =$$
$$= \sum_{n=-\infty}^{+\infty} \left(\sum_{m=-\infty}^{+\infty} h(m)x(n-m) \right) e^{-j\omega n} =$$
$$= \sum_{m=-\infty}^{+\infty} h(m) \sum_{n=-\infty}^{+\infty} x(n-m)e^{-j\omega(n-m)}e^{-j\omega m} =$$
$$= \sum_{m=-\infty}^{+\infty} h(m)X(e^{j\omega})e^{-j\omega m} =$$
$$= X(e^{j\omega}) \sum_{m=-\infty}^{+\infty} h(m)e^{-j\omega m} =$$
$$= X(e^{j\omega})H(e^{j\omega})$$

Q.E.D.

The Discrete-Time Fourier Transform (DTFT) transforms the convolution sum of two sequences into the product of their respective DTFTs. If we know the frequency response of an LTI system, we can calculate the output sequence, denoted as $y(n)$ and representing the response to the input sequence $x(n)$, through the following steps:

1. $X(e^{j\omega}) =$ DTFT $\{x(n)\}$
2. $Y(e^{j\omega}) = X(e^{j\omega})H(e^{j\omega})$

3. $y(n) = \text{IDTFT}\ \{Y(e^{j\omega})\}$

The main inconvenience is represented by the fact that the IDTFT requires the computation of the integral of a continuous function. We will address this inconvenience later by introducing the Discrete Fourier Transform (DFT).

The concept of digital filter

An application of LTI systems is to allow certain frequency components of a sequence to pass without distortions while blocking any other frequency component. Such systems are referred to as *digital filters*. For example, let us consider the low pass filter with frequency response:

$$H(e^{j\omega}) \simeq \begin{cases} 1 & 0 \leq |\omega| \leq \omega_c \\ 0 & \omega_c \leq |\omega| \leq \pi \end{cases}$$

Let the system input be

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

with $0 < \omega_1 < \omega_c < \omega_2 < \pi$. Since $cos(\omega n) = \frac{1}{2}[e^{j\omega n} + e^{-j\omega n}]$, it can be easily proved that:

$$y(n) \simeq A|H(e^{j\omega_1})|cos\left(\omega_1 n + \text{arg}\{H(e^{j\omega_1}\}\right) \simeq Acos(\omega_1 n).$$

Thus, the output comprises only the first cosine component, which lies within the passband of the filter, while the second component outside the passband is eliminated.

Example:

Let us consider the system

$$\begin{cases} y(n) = ay(n-1) + x(n) \\ y(-1) = 0. \end{cases}$$

We aim to calculate the frequency response of this system. If $x(n) = \delta(n)$ then it is easy to observe through successive substitutions that

$$y(n) = h(n) = a^n \qquad \forall n \geq 0.$$

$$H(e^{j\omega}) = \sum_{n=0}^{+\infty} a^n e^{-j\omega n} = \sum_{n=0}^{+\infty} \left(ae^{-j\omega}\right)^n =$$

$$= \frac{1}{1 - ae^{-j\omega}},$$

provided that $|a| < 1$.

$$|H(e^{j\omega})| = \frac{1}{|1 - a\cos(\omega) + ja\sin(\omega)|} =$$

$$= \frac{1}{\sqrt{(1 - a\cos(\omega))^2 + a^2\sin^2(\omega)}} =$$

$$= \frac{1}{\sqrt{1 - 2a\cos(\omega) + a^2}}$$

$$\text{arg } \{H(e^{j\omega})\} = \text{arg } \left\{ \frac{e^{j\omega}}{e^{j\omega} - a} \right\} = \omega - \text{arctan } \left(\frac{\sin(\omega)}{\cos(\omega) - a} \right).$$

When $a > 0$, the filter exhibits a low-pass behavior, while for $a < 0$, it demonstrates a high-pass behavior. This is a first-order system as it involves only one delayed sample of the output. With first-order systems, we can implement either low-pass or high-pass filters.

For more information study:

S. K. Mitra, "Digital Signal Processing: a computer based approach," 4th edition, McGraw-Hill, 2011
Chapter 4.1-4.5, pp. 141-163
Chapter 4.7.1, pp. 173-174
Chapter 4.8.1-4.8.2, pp. 175-177

05 The Z-Transform

05.01 Definition of the Z-Transform

The Z-Transform is a generalization of the discrete-time Fourier transform. It exists for many sequences for which the DTFT does not exist. It is a function of the complex variable z.
Given a sequence $g(n)$, the Z-Transform is expressed as

$$G(z) = \sum_{n=-\infty}^{+\infty} g(n) z^{-n},$$

where $z = \text{Re}\,(z) + j\text{Im}\,(z)$ represents a continuous complex variable.
We will denote the Z-Transform of $g(n)$ also as:

$$\mathcal{Z}\{g(n)\} = G(z) = \sum_{n=-\infty}^{+\infty} g(n) z^{-n}.$$

If we consider $z = r \cdot e^{j\omega}$,

$$G(z) = \sum_{n=-\infty}^{+\infty} g(n) r^{-n} e^{-j\omega n}.$$

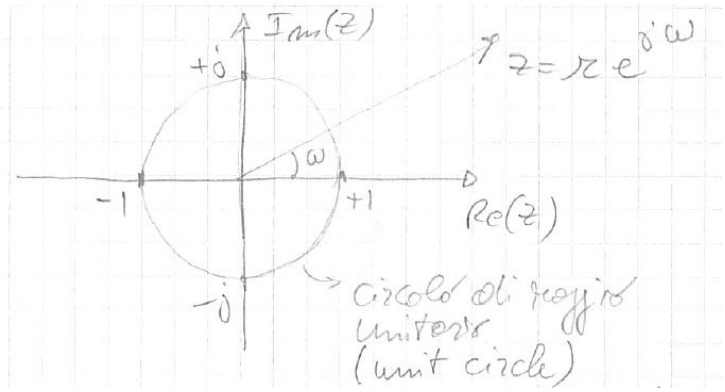
This expression can be interpreted as the Discrete-Time Fourier Transform (DTFT) of the modified sequence $\{g(n)r^{-n}\}$.
When we compare the definition of $G(z)$ with the DTFT of $g(n)$:

$$G(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} g(n) e^{-j\omega n}$$

we observe that

$$G(e^{j\omega}) = G(z)\Big|_{z=e^{j\omega}},$$

assuming these transforms exist.
We can provide a geometrical interpretation to the Z-Transform by considering a point z in the complex plane.



Assigned r and ω, the point $z = re^{j\omega}$ in the complex plane lies at the end of a vector with a magnitude of r and an angle of ω radians with respect to the real axis.
The set of points with $|z| = 1$ and consequently with $z = e^{j\omega}$, forms a circle with radius one (unit circle) in the z plane.
For $r = 1$, $z = e^{j\omega}$ and the Z-Transform $G(z)$ evaluated on the unit circle yields the DTFT of $g(n)$, assuming that this DTFT exists.
For $z = 1$, $G(z) = G(e^{j0})$, i.e., the value $G(e^{j\omega})$ for $\omega = 0$.
For $z = j$, $G(z) = G(e^{j\frac{\pi}{2}})$, i.e., the value $G(e^{j\omega})$ for $\omega = \frac{\pi}{2}$.
For $z = -1$, $G(z) = G(e^{j\pi})$, i.e., the value $G(e^{j\omega})$ for $\omega = \pi$.
For $z = -j$, $G(z) = G(e^{j\frac{3\pi}{2}})$, i.e., the value $G(e^{j\omega})$ for $\omega = \frac{3\pi}{2}$.
By moving along the unit circle from $z = 1$, till $z = -1$, till $z = 1$ again, we obtain all values of $G(e^{j\omega})$ for $0 \leq \omega \leq 2\pi$. As observed earlier, the DTFT is periodic with a period of 2π. Thus, by moving along the unit circle in either a clockwise or anti-clockwise direction, we can compute $G(e^{j\omega})$ for all values of ω within the range $-\infty < \omega < +\infty$.
As with the DTFT, the Z-Transform converges only when specific conditions are met, and typically, it converges within a specific region of the complex z plane.
For a given sequence, the set $\mathcal{R}$ of z values for which the Z-Transform converges is referred to as the *Region of Convergence (ROC)*. In particular, it can be proven that the Z-Transform converges for all values of z for which the sequence $g(n)r^{-n}$ is absolutely summable (with $r = |z|$):

$$\sum_{n=\infty}^{+\infty} \left|g(n)r^{-n}\right| < +\infty.$$

Even if the sequence $g(n)$ is not absolutely summable, $g(n)r^{-n}$ could still be absolutely summable. In other words, even if the sequence $g(n)$ does not have does not have a DTFT, it may have a Z-Transform within a certain region of the complex z plane.
Furthermore, the condition of absolute summability indicates that if there exists a specific $z = re^{j\omega}$ for which the Z-Transform exists (i.e., for which $\sum_{n=\infty}^{+\infty} |g(n)r^{-n}| < +\infty$), then the Z-Transform exists throughout the entire circle with a radius of r.
In general, the Region of Convergence $\mathcal{R}$ forms an annular region in the complex z plane:

$$R_{g-} < |z| < R_{g+}, \qquad \text{with } 0 \leq R_{g-} \leq R_{g+} \leq +\infty.$$

We will observe that different sequences can share the same Z-Transform; however, in such cases, they may have distinct regions of convergence. Therefore, it is crucial to explicitly specify the region of convergence for the Z-Transform.

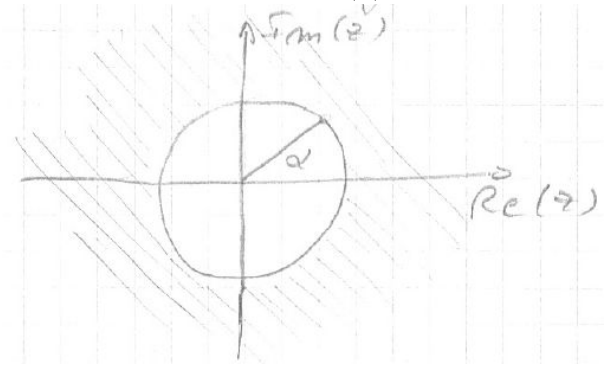
Example:
Let's compute the Z-Transform of the causal sequence $x(n) = \alpha^n \mu(n)$:

$$X(z) = \sum_{n=-\infty}^{+\infty} \alpha^n \mu(n) z^{-n} = \sum_{n=0}^{+\infty} \alpha^n z^{-n}.$$

This series converges if and only if $|\alpha z^{-1}| < 1$, and it converges to

$$X(z) = \frac{1}{1 - \alpha z^{-1}}.$$

The region of convergence is $|z| > \alpha$.



For $\alpha = 1$, $x(n) = \mu(n)$, thus $\mu(z) = \dfrac{1}{1 - z^{-1}}$ with a region of convergence $|z| > 1$.
Whenever the sequence is causal, the region of convergence lies outside a circle centered at the origin of the z plane.

Example:
Let us compute the Z-Transform of the exponential acausal sequence

$$x(n) = -\alpha^n \mu(-n-1)$$

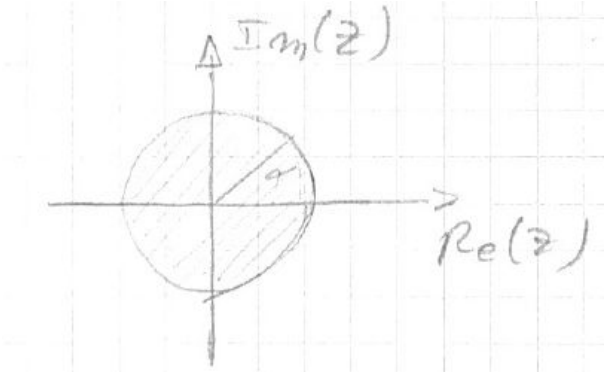
$$X(z) = \sum_{n=-\infty}^{+\infty} x(n) z^{-n} = \sum_{n=-\infty}^{-1} -\alpha^n z^{-n} =$$

(consider $n = -m$)

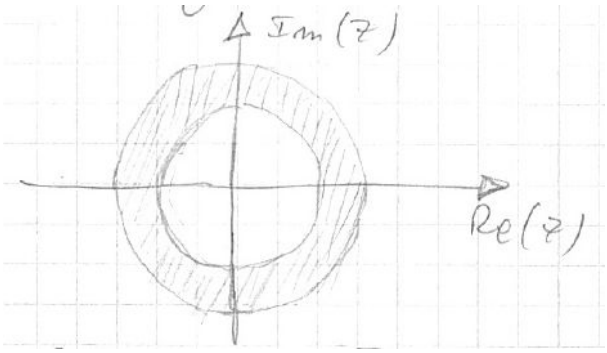
$$= \sum_{m=1}^{\infty} -\alpha^{-m} z^m = -\alpha^{-1} z \sum_{m=0}^{+\infty} \alpha^{-m} z^m =$$

$$= \frac{-\alpha^{-1} z}{1 - \alpha^{-1} z} = \frac{1}{1 - \alpha z^{-1}}$$

provided $|\alpha^{-1} z| < 1$.
We have obtained the same Z-Transform of the previous causal sequence, but now the region of convergence is $|z| < \alpha$.



Note that acausal sequences always have a region of convergence that is a circle centered at the origin of the z plane.
Two-sided sequences consist of a causal part and an acausal part, and their Z-Transform, if it converges at all, has an annular region of convergence.



This can be easily explained, as the causal part has a region of convergence $|z| > R_1$, and the acausal part has a region of convergence $|z| < R_2$. If $R_1 < R_2$, the two-sided sequence has a region of convergence $R_1 < |z| < R_2$. On the contrary, if $R_2 > R_1$ the Z-Transform does not converge.

Example:

Let's compute the Z-Transform of the finite length sequence

$$x(n) = \begin{cases} \alpha^n & M \leq n \leq N-1, \\ 0 & \text{elsewhere.} \end{cases}$$

$$X(z) = \sum_{n=M}^{N-1} \alpha^n z^{-n} =$$

$$= \alpha^M z^{-M} + \alpha^{M+1} z^{-M-1} + \ldots + \alpha^{N-1} z^{-N+1} =$$

$$= \alpha^M z^{-M} \sum_{m=0}^{N-M-1} \alpha^m z^{-m} =$$

$$= \alpha^M z^{-M} \frac{1 - \alpha^{N-M} z^{-(N-M)}}{1 - \alpha z^{-1}} =$$

$$= \frac{\alpha^M z^{-M} - \alpha^N z^{-N}}{1 - \alpha z^{-1}}$$

Since the series is composed of a finite number of terms, the region of convergence coincides with the entire z plane, except possibly from $z = 0$ or $z = \infty$.

In particular, for $N - 1 \geq M \geq 0$, the region of convergence is the entire z plane except at $z = 0$.

If $N - 1 > 0$, $M < 0$, the region of convergence is the entire z plane except at $z = 0$ and $z = \infty$.

If $M \leq N - 1 < 0$, the region of convergence is the entire z plane except at $z = \infty$.

If the region of convergence comprises the unit circle, the sequence admits the DTFT.

Nevertheless, even if a sequence has DTFT, there is no guarantee that the sequence also has a Z-Transform.

For example, the ideal low-pass sequence $h_{LP}(n)$, defined as

$$H(e^{j\omega}) = \begin{cases} 1 & |\omega| < \omega_c, \\ 0 & \text{otherwise} \end{cases},$$

admits DTFT, but it does not have a Z-Transform. Indeed, $h_{LP}(n) r^{-n}$, for any r, is not absolutely summable.

All the Z-Transforms we will consider in this course will always be rational functions of z^{-1}, i.e., the ratio of two polynomials:

$$H(z) = \frac{P(z)}{Q(z)} = \frac{p_0 + p_1 z^{-1} + \ldots + p_M z^{-M}}{d_0 + d_1 z^{-1} + \ldots + d_N z^{-N}}$$

The polynomial degree of the numerator is M, and that of the denominator is N.
An alternative form represents $H(z)$ as the ratio of two polynomials in z:

$$H(z) = z^{N-M} \frac{p_0 z^M + p_1 z^{M-1} + \ldots + p_M}{d_0 z^N + d_1 z^{N-1} + \ldots + d_N}$$

Both relations can be written in a factorized form as[1]

$$H(z) = \frac{p_0}{d_0} \frac{\prod_{l=1}^{M} \left(1 - \xi_l z^{-1}\right)}{\prod_{l=1}^{N} \left(1 - \lambda_l z^{-1}\right)} = \frac{p_0}{d_0} z^{N-M} \frac{\prod_{l=1}^{M} (z - \xi_l)}{\prod_{l=1}^{N} (z - \lambda_l)}$$

where ξ_l and λ_l are the roots of the numerator and denominator polynomials in z.
When $z = \xi_l$, we have $H(\xi_l) = 0$. Thus, the roots of the numerator ξ_l are referred to as the *zeros* of $H(z)$.
When $z = \lambda_l$, we have $H(\lambda_l) = \infty$. Thus, the roots of the denominator λ_l are referred to as the *poles* of $H(z)$.
Note that in our Z-Transform $H(z)$ we have M zeros and N poles.
Moreover, if $N > M$ we have additional $(N - M)$ zeros in the origin; if $M > N$ we have additional $(M - N)$ poles in the origin.
Note also that a rational Z-Transform can be fully characterized by the positions of its poles λ_l and zeros ξ_l, along with the constant term $\frac{p_0}{d_0}$.
It is interesting to observe that the region of convergence of a rational Z-Transform is delimited by the location of its poles.
If $H(z)$ has N poles, each with a different magnitude, we will have $N+1$ possible regions of convergence. For each of these regions of convergence, we have a different sequence $h(n)$ that shares the same Z-Transform $H(z)$.

[1] $\prod_{i=1}^{M} x_i = x_1 \cdot x_2 \cdot \ldots \cdot x_M$.

05.02 The inverse Z-Transform

Let us denote $\mathcal{C}$ a closed contour in the region of convergence that encloses the origin (for example, $\mathcal{C}$ could be a circle centered in the origin within the region of convergence). The expression for the Inverse Z-Transform is given by:

$$g(n) = \frac{1}{2\pi j} \oint_{\mathcal{C}} G(z) z^{n-1} \mathrm{d}z,$$

where $\oint_{\mathcal{C}}$ indicates the integration performed on the closed contour $\mathcal{C}$.
In general, it is challenging to directly evaluate the integral along a closed contour. Additionally, the integral needs to be computed for every time value n.
Fortunately, we will only consider rational Z-Transforms and causal sequences. Under these conditions, there are simple methods that allow us to compute the Z-Transform.

A first method for computing the Z-Transform is the *tabular method*. Very often, the Z-Transform has a simple expression or can be decomposed into the sum of simple expressions. Many books provide tables that offer inverses for these elementary transforms.

$\mathcal{Z}\{x(n)\}$	$x(n)$	R.O.C.
1	$\delta(n)$	z-plane
z^{-N}	$\delta(n-N)$	z-plane - $\{z=0\}$
$\frac{1}{1-z^{-1}}$	$\mu(n)$	$\lvert z\rvert > 1$
$\frac{1}{1-\alpha z^{-1}}$	$\alpha^n \mu(n)$	$\lvert z\rvert > \lvert\alpha\rvert$
$\frac{\alpha z^{-1}}{(1-\alpha z^{-1})^2}$	$n\alpha^n \mu(n)$	$\lvert z\rvert > \lvert\alpha\rvert$
$\frac{1}{(1-\alpha z^{-1})^2}$	$(n+1)\alpha^n \mu(n)$	$\lvert z\rvert > \lvert\alpha\rvert$

For example, if we have

$$H(z) = \frac{0.5z}{z^2 - z + 0.25} \qquad \text{R.O.C. } |z| > 0.5$$

Then,

$$H(z) = \frac{0.5z}{(z-0.5)^2} = \frac{0.5z^{-1}}{(1-0.5z^{-1})^2}.$$

From the table, we can deduce that $h(n) = n\,0.5^n \mu(n)$.

A second method for the Z-Transform inversion is that of the *partial fraction expansion*.
If a sequence is causal, the region of convergence is external to a circle centered in the origin. We can compute the inverse Z-Transform of a rational function $G(z)$ by decomposing it into partial fractions, inverting each of the partial fractions, and summing up all the resulting sequences. We exploit the linearity property of the Z-Transform, which we will study later:
"Let $g(n)$ and $h(n)$ be two sequences with Z-Transform $G(z)$ and $H(z)$, respectively. For any constant

α and β, the Z-Transform of $\alpha g(n) + \beta h(n)$ is $\alpha G(z) + \beta H(z)$ with the region of convergence being the intersection of the regions of convergence of $G(z)$ and $H(z)$."
We consider $G(z) = \dfrac{P(z)}{D(z)}$, where $P(z)$ is a polynomial of degree M and $D(z)$ is a polynomial of degree N.
If $M \geq N$, the rational function $\dfrac{P(z)}{D(z)}$ is called an *improper* fraction.
If $M < N$, the rational function is called a *proper* fraction.
If the fraction is improper, we can decompose it as the sum of a polynomial in z^{-1} plus a proper fraction:

$$G(z) = \sum_{l=0}^{M-N} \eta_l z^{-l} + \frac{P_1(n)}{D(n)},$$

where the degree of $P_1(n)$ is lower than N. To obtain this expression, it suffices to divide $P(z)$ by $D(z)$ (we will cover polynomial division later).
The inverse-transform of $\sum_{l=0}^{M-N} \eta_l z^{-l}$ is known, as it is $\sum_{l=0}^{M-N} \eta_l \delta(n-l)$. Thus, we concentrate our attention only on the inversion of a proper rational function. We will distinguish two cases:

- simple poles,
- multiple poles.

Simple poles:
In most cases, $G(z)$ is a proper rational function with simple poles, i.e., with all distinct poles.
Let us denote λ_k, $1 \leq k \leq N$, the N poles of $G(z)$. Then, $G(z)$ can be rewritten in the following form:

$$G(z) = \sum_{l=1}^{N} \frac{\rho_l}{1 - \lambda_l z^{-1}},$$

where the ρ_l constants are called *residuals* and can be easily computed using:

$$\rho_l = (1 - \lambda_l z^{-1}) G(z) \Big|_{z=\lambda_l}$$

(i.e., by removing the factor $(1 - \lambda_l z^{-1})$ from the denominator and computing the resulting value for $z = \lambda_l$).
Each term $\dfrac{\rho_l}{1 - \lambda_l z^{-1}}$ has a simple inverse: $\rho_l \, \lambda^n \, \mu(n)$. Thus, by exploiting the linearity of the Z-Transform, we get

$$g(n) = \sum_{l=1}^{N} \rho_l \, \lambda_l^n \, \mu(n).$$

Example:

$$H(z) = \frac{1 + 2.0 z^{-1}}{(1 - 0.2 z^{-1}) \cdot (1 + 0.6 z^{-1})}$$

$$H(z) = \frac{\rho_1}{1 - 0.2 z^{-1}} + \frac{\rho_2}{1 + 0.6 z^{-1}}$$

$\lambda_1 = 0.2$ and $\lambda_2 = -0.6$.

$$\rho_1 = \frac{1 + 2.0z^{-1}}{1 + 0.6z^{-1}}\Big|_{z=0.2} = \frac{1 + 2 \cdot (0.2)^{-1}}{1 + 0.6 \cdot (0.2)^{-1}} = \frac{11}{4} = 2.75$$

$$\rho_2 = \frac{1 + 2.0z^{-1}}{1 - 0.2z^{-1}}\Big|_{z=-0.6} = \frac{1 + 2 \cdot (-0.6)^{-1}}{1 - 0.2 \cdot (-0.6)^{-1}} = \frac{0.6 - 2.0}{0.6 + 0.2} = -1.75$$

Thus, we obtain

$$h(n) = 2.75\,(0.2)^n\,\mu(m) - 1.75\,(-0.6)^n\,\mu(n)$$

Another way to compute the residues is to express the numerator of $G(z)$ as a function of the residues ρ_i and then impose the equality of the numerator polynomials. This way, we obtain a system of equations in the unknown ρ_i which, once solved, allows us to derive the partial fraction expansion of $G(z)$.

In the previous example:

$$H(z) = \frac{\rho_1}{1 - 0.2z^{-1}} + \frac{\rho_2}{1 + 0.6z^{-1}} =$$

$$= \frac{\rho_1(1 + 0.6z^{-1}) + \rho_2(1 - 0.2z^{-1})}{(1 - 0.2z^{-1}) \cdot (1 + 0.6z^{-1})} =$$

$$= \frac{\rho_1 + \rho_2 + (0.6\rho_1 - 0.2\rho_2)z^{-1}}{(1 - 0.2z^{-1}) \cdot (1 + 0.6z^{-1})}$$

Imposing:

$$\rho_1 + \rho_2 + (0.6\rho_1 - 0.2\rho_2)z^{-1} = 1 + 2z^{-1}$$

it must be:

$$\begin{cases} \rho_1 + \rho_2 = 1 \\ 0.6\rho_1 - 0.2\rho_2 = 2 \end{cases}$$

Solving this system of equations, we obtain $\rho_1 = 2.75$ and $\rho_2 = -1.75$.

When the rational function has real coefficients, the complex poles always appear in complex-conjugate pairs, and the corresponding residues are also complex-conjugate. Thus, they can be combined into a single fraction of the second order:

$$\frac{\rho_l}{1 - \lambda_l z^{-1}} + \frac{\rho_l^*}{1 - \lambda_l^* z^{-1}} = \frac{p_{0,l} + p_{1,l} z^{-1}}{1 + d_{1,l} z^{-1} + d_{2,l} z^{-2}}$$

Multiple poles:

If $G(z)$ is a rational function with multiple poles (double poles, triple poles, or even of higher order), the partial fraction expansion takes on a different form.

Suppose $z = \nu$ is a pole of multiplicity L (i.e., in the denominator there are L roots equal to ν), and that the other poles are simple. We have

$$G(z) = \sum_{l=1}^{N-L} \frac{\rho_l}{1 - \lambda_l z^{-1}} + \sum_{i=1}^{L} \frac{\gamma_i}{(1 - \nu z^{-1})^i}$$

where the constants γ_i (which are no longer called residues) are computed with the formula:

$$\gamma_i = \frac{1}{(L-i)!(-\nu)^{L-i}} \frac{\mathrm{d}^{L-i}}{\mathrm{d}(z^{-1})^{L-i}} \left[(1-\nu z^{-1})^L G(z)\right] \Big|_{z=\nu},$$

for $1 \leq i \leq L$.

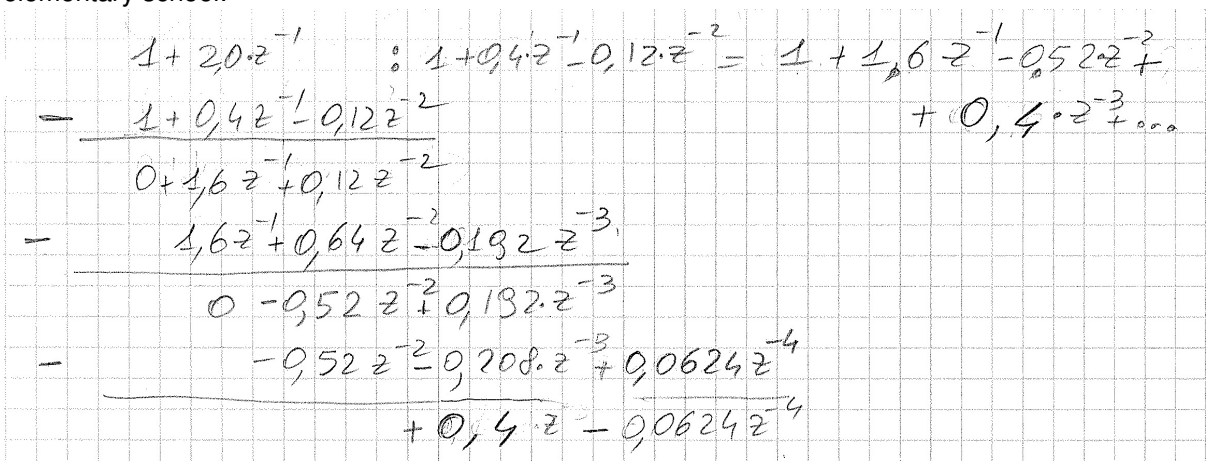
Thus, the computation of γ_i requires eliminating the factor $(1-\nu z^{-1})^L$ from the denominator of $G(z)$ and computing the derivatives with respect to z^{-1} from the first to the $(L-1)$-th. The values of these derivatives and of $(1-\nu z^{-1})^L G(z)$ computed for $z=\nu$ provide the γ_i values. The other residues are computed as before. Eventually, from the tables reported in many books, it is easy to find the inverse transform of $\frac{\gamma_i}{(1-\nu z^{-1})^i}$.

Inverse Z-Transform via long division method.

Another method for computing the inverse of a causal Z-Transform is the *long division*.

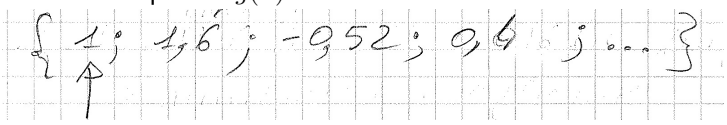
The objective is to obtain the expansion of $G(z)$ in the form of a power series in z^{-1}. If we possess such a series, we immediately know the inverse transform. Indeed, the coefficient that multiplies z^{-n} is the term $g(n)$ of the inverse transform.

If $G(z)$ is rational, we can express the numerator and denominator as polynomial in z^{-1}. Then, we can derive the power series expansion by means of the (long) division of the numerator by the denominator. The division of polynomials applies the same methodology as the divisions you have studied in elementary school:



(How many times does 1 go into 1? 1. Thus we subtract from $1+0.2z^{-1}$ the polynomial $1\cdot(1+0.4z^{-1}-0.12z^{-2})$ and we get the remainder $1.6z^{-1}+0.12z^{-2}$. How many times does 1 go into $1.6z^{-1}$? $1.6z^{-1}$. Thus, we subtract from $1.6z^{-1}+0.12z^{-2}$ the polynomial $1.6z^{-1}\cdot(1+0.4z^{-1}-0.12z^{-2})$ and get the remainder $-0.52z^{-2}+0.192z^{-3}$. How many times does 1 go into $-0.52z^{-2}$? $-0.52z^{-2}$. And so on.)

Thus the sequence $g(n)$ is :

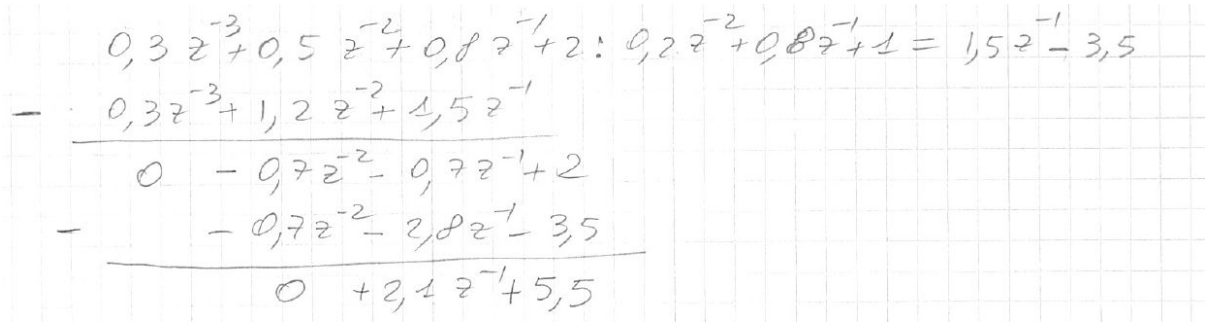


The division between polynomials is also used in the case of the partial fraction expansion for passing from an improper $G(z)$ to an expression composed by a polynomial in z^{-1} (which has a trivial inverse) plus a proper rational function.

For example, consider:

$$G(z) = \frac{2 + 0.8z^{-1} + 0.5z^{-2} + 0.3z^{-3}}{1 + 0.8z^{-1} + 0.2z^{-2}}.$$

We must divide the two polynomials till we get a remainder of degree lower than the denominator. Since we only want to reduce the order in z^{-1} of the numerator, in this case, we apply the same division method we have seen but, compared with the long division, we must reverse the order of the two polynomials:



(Note that $(1.5z^{-1} - 3.5) \cdot (0.2z^{-2} + 0.8z^{-1} + 1) + (2.1z^{-1} + 5.5) = 0.3z^{-3} + 0.5z^{-2} + 0.8z^{-1} + 2$).
Thus,

$$G(z) = -3.5 + 1.5z^{-1} + \frac{5.5 + 2.1z^{-1}}{1 + 0.8z^{-1} + 0.2z^{-2}},$$

and we have

$$g(n) = -3.5\delta(n) + 1.5\delta(n-1) + \rho_1\lambda_1^n\mu(n) + \rho_2\lambda_2^n\mu(n),$$

with λ_1 and λ_2 the roots of $1 + 0.8z^{-1} + 0.2z^{-2}$, and ρ_1 and ρ_2 the corresponding residues.

05.03 Properties of the Z-Transform

Also the Z-Transform has important properties. In the following we will consider

$$g(n) \overset{\mathcal{Z}}{\longleftrightarrow} G(z) \qquad \text{with ROC } \mathcal{R}_g$$

$$h(n) \overset{\mathcal{Z}}{\longleftrightarrow} H(z) \qquad \text{with ROC } \mathcal{R}_h$$

where $\mathcal{R}_g$ is the annular region $R_{g-} < |z| < R_{g+}$, and $\mathcal{R}_h$ is the annular region $R_{h-} < |z| < R_{h+}$.

Conjugation theorem
The Z-Transform of $g^*(n)$ is $G^*(z^*)$ with ROC $\mathcal{R}_g$.
Proof:

$$\mathcal{Z}\{g^*(n)\} = \sum_{n=-\infty}^{+\infty} g^*(n)z^{-n} =$$

$$= \left[\sum_{n=-\infty}^{+\infty} g(n)(z^*)^{-n}\right]^* = G^*(z^*).$$

Moreover, since $|z^*| = |z|$, $G^*(z^*)$ converges for all the z values for which $G(z)$ converges.

Q.E.D.

Time-reversal

The Z-Transform of $x(n) = g(-n)$ is $X(z) = G(1/z)$ with ROC: $R_{g+}^{-1} < |z| < R_{g-}^{-1}$.

Proof:

$$\sum_{n=-\infty}^{+\infty} g(-n)z^{-n} = \sum_{m=-\infty}^{+\infty} g(m)z^{m} =$$

$$= \sum_{m=-\infty}^{+\infty} g(m)\left(z^{-1}\right)^{-m} = G(1/z).$$

$X(z) = G(1/z)$ converges for all z for which $1/z$ belongs to the ROC of $G(z)$. Thus, for all z for which

$$\frac{1}{R_{g+}} < |z| < \frac{1}{R_{g-}}.$$

Q.E.D.

Linearity

The Z-Transform of the sequence $\alpha g(n) + \beta h(n)$ is $\alpha G(z) + \beta H(z)$, with region of convergence that is the intersection between the two ROCs $\mathcal{R}_g$ and $\mathcal{R}_h$, i.e., $\mathcal{R}_g \cap \mathcal{R}_h$.

(If the two regions of convergence do not overlap the Z-Transform of $\alpha g(n) + \beta h(n)$ does not exist!)

Exercise:

Let us compute the Z-Transform of

$$x(n) = r^n \cos(\omega_0 n)\mu(n),$$

$r \in \mathbb{R}$.

Note that

$$x(n) = r^n \frac{e^{j\omega_0 n} + e^{-j\omega_0 n}}{2}\mu(n) =$$

$$= \frac{1}{2}r^n e^{j\omega_0 n}\mu(n) + \frac{1}{2}r^n e^{-j\omega_0 n}\mu(n).$$

Since $\alpha^n\mu(n)$ has Z-Transform $\dfrac{1}{1-\alpha z^{-1}}$ with ROC $|z| > \alpha$,

$$X(z) = \frac{1/2}{1 - re^{j\omega_0}z^{-1}} + \frac{1/2}{1 - re^{-j\omega_0}z^{-1}} =$$

$$= \frac{1}{2}\frac{1 - re^{-j\omega_0}z^{-1} + 1 - re^{j\omega_0}z^{-1}}{(1 - re^{j\omega_0}z^{-1}) \cdot (1 - re^{-j\omega_0}z^{-1})} =$$

$$= \frac{1 - r\cos(\omega_0)z^{-1}}{1 - re^{j\omega_0}z^{-1} - re^{-j\omega_0}z^{-1} + r^2z^{-2}} =$$

$$= \frac{1 - r\cos(\omega_0)z^{-1}}{1 - 2r\cos(\omega_0)z^{-1} + r^2z^{-2}}$$

with ROC $|z| > r$.

Time-shifting

The Z-Transform of the delayed sequence $x(n) = g(n - n_0)$ (with n_0 integer) is $X(z) = z^{-n_0}G(z)$. The region of convergence of $X(z)$ is the same as $G(z)$ apart from $z = 0$ for $n_0 > 0$ or $z = \infty$ for $n_0 < 0$ (because z^{-n_0} adds $|n_0|$ poles in $z = 0$ or $z = \infty$).

Proof:

$$\sum_{n=-\infty}^{+\infty} g(n - n_0)z^{-n} = \sum_{m=-\infty}^{+\infty} g(m)z^{-(m+n_0)} =$$

$$= \sum_{m=-\infty}^{+\infty} g(m)z^{-m} \cdot z^{-n_0} = G(z)z^{-n_0}.$$

Q.E.D.

Multiplication for an exponential sequence

The Z-Transform of the sequence $x(n) = \alpha^n g(n)$ is $X(Z) = G\left(\frac{z}{\alpha}\right)$. The region of convergence is $|\alpha|\mathcal{R}_g$.

Proof:

$$\sum_{n=-\infty}^{+\infty} \alpha^n g(n)z^{-n} = \sum_{n=-\infty}^{+\infty} g(n)\left(\frac{z}{\alpha}\right)^n = G\left(\frac{z}{\alpha}\right).$$

$X(z) = G\left(\frac{z}{\alpha}\right)$ converges for any value z for which $\frac{z}{\alpha} \in \mathcal{R}_g$ and thus when

$$R_{g-}|\alpha| < |z| < R_{g+}|\alpha|.$$

Q.E.D.

Convolution Theorem

The Z-Transform of the convolution sum between two sequences $g(n) \circledast h(n)$ is $G(z) \cdot H(z)$ with a region of convergence that is the intersection between the ROCs of $G(z)$ and $H(z)$, i.e., $\mathcal{R}_g \cap \mathcal{R}_h$.

Proof:

$$X(z) = \sum_{n=-\infty}^{+\infty} x(n)z^{-n} =$$

$$= \sum_{n=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} g(k)h(n - k)z^{-n} =$$

$$= \sum_{k=-\infty}^{+\infty} g(k) \sum_{n=-\infty}^{+\infty} h(n - k)z^{-n} =$$

$$= \sum_{k=-\infty}^{+\infty} g(k) \sum_{m=-\infty}^{+\infty} h(m)z^{-(m+k)} =$$

$$= \sum_{k=-\infty}^{+\infty} g(k)z^{-k}H(z) = G(z)H(z).$$

Moreover, $X(z)$ converges for any value z for which $\sum_{m=-\infty}^{+\infty} h(m)z^{-m}$ and $\sum_{k=-\infty}^{+\infty} g(k)z^{-k}$ converge. Thus, the ROC of $X(z)$ is $\mathcal{R}_g \cap \mathcal{R}_h$.

Q.E.D.

05.04 The Transfer Function

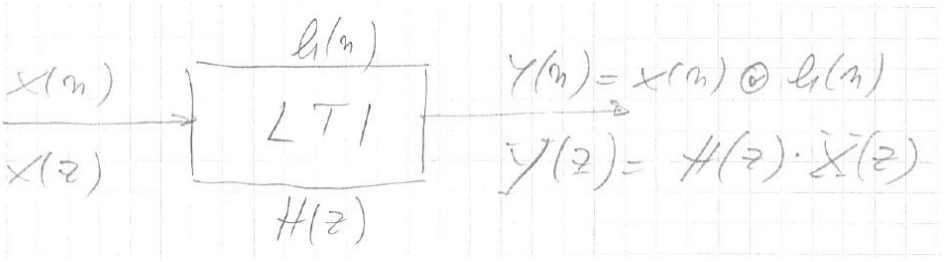
The convolution theorem leads to the concept of *Transfer Function* of an LTI system.
Let us consider an LTI system with impulse response $h(n)$. The input-output relation of the system is

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

with $x(n)$ and $y(n)$ the input and output of the LTI system, respectively.
Let us indicate with $X(z)$, $Y(z)$, and $H(z)$ the Z-transform of $x(n)$, $y(n)$, and $h(n)$, respectively. According to the convolution theorem, the input-output relation in the Z-Transform domain becomes:

$$Y(z) = H(z)X(z),$$

with $H(z) = \sum_{n=-\infty}^{+\infty} h(n)z^{-n}$.



The Z-Transform of the impulse response, $H(z)$, is commonly called *Transfer Function*. Similar to the impulse response, it completely characterizes the LTI system.

$$H(z) = \frac{Y(z)}{X(z)},$$

$H(z)$ is given by the ratio between the Z-Transform of $y(n)$ and $x(n)$.
The transfer function can be seen as a generalization of the frequency response concept. If the region of convergence of $H(z)$ comprises the unit circle:

$$H(e^{j\omega}) = H(z)\Big|_{z=e^{j\omega}}.$$

Thus, the frequency response can be obtained by evaluating the transfer function on the unit circle.
The transfer function can be obtained from the frequency response by *analytic continuation*:

$$H(z) = H(e^{j\omega})\Big|_{\omega=\frac{1}{j}\ln(z)}.$$

We are particularly interested in two types of discrete-time LTI systems:

- FIR filters,
- IIR filters (described by a finite difference equation).

FIR filters

FIR filters have a finite impulse response:

$$y(n) = \sum_{m=-N_1}^{N_2} h(m)x(n-m)$$

with $h(n) = 0 \; \forall n < N_1$, and $\forall n > N_2$.
$h(n)$ has length $N_1 + N_2 + 1$.

$$H(z) = \sum_{n=N_1}^{N_2} h(n)z^{-n} =$$

$$= h(-N_1)z^{+N_1} + h(-N_1+1)z^{+N_1-1} + \ldots + h(0) + h(1)z^{-1} + \ldots + h(N_2-1)z^{-N_2+1} + h(N_2)z^{-N_2}.$$

Properties of FIR filters

- The region of convergence of $H(z)$ coincides with the entire complex plane, possibly excluding 0 and ∞.
- FIR filters are guaranteed to be stable in the BIBO sense:

$$\sum_{n=-\infty}^{+\infty} |h(n)| = \sum_{n=-N_1}^{N_2} |h(n)| < +\infty.$$

- FIR filters are always realizable. Even if the impulse response has non-null terms for $n < 0$, introducing a delay z^{-N_1} ensures realizability.
- For causal filters, the transfer function is given by:

  $$H(z) = h(0) + h(1)z^{-1} + h(2)z^{-2} + \ldots + h(N-1)z^{N-1}.$$

  Thus, the transfer function is a polynomial in z^{-1}, earning them the name *filters with only zeros*.
- Since the input-output relation is non-recursive, quantization errors propagate only from the input to the output in FIR filters.
- We will prove that FIR filters can have linear phase response.

IIR filters

The IIR filters we are interested in are described by a finite difference equation:

$$y(n) = b_0x(n) + b_1x(n-1) + \ldots + b_Mx(n-M) - a_1y(n-1) - a_2y(n-2) - \ldots - a_Nx(n-N).$$

For these filters, it is easy to compute the transfer function using the linearity property and the time-shift property of the Z-Transform. By transforming both members:

$$Y(z) = b_0X(z) + b_1z^{-1}X(z) + b_2z^{-2}X(z) + \ldots b_Mz^{-M}X(z) - a_1z^{-1}Y(z) - a_2z^{-2}Y(z) - \ldots - a_Nz^{-N}Y(z),$$

$$\left(1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-N}\right) Y(z) = \left(b_0 + b_1 z^{-1} + \ldots + b_M z^{-M}\right) X(z)$$

Thus,

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \ldots + b_M z^{-M}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-N}}.$$

The transfer function is a rational function in z^{-1}. It is also:

$$H(z) = z^{M-N} \cdot \frac{b_0 z^M + b_1 z^{M-1} + \ldots + b_M}{z^N + a_1 z^{N-1} + \ldots + a_N}.$$

The roots of the numerator polynomial are called *zeros* of the system. The roots of the denominator polynomial are called *poles*. The denominator polynomial is also called the *characteristic polynomial*.

Properties of IIR filters

- IIR filters or systems can be stable or unstable in BIBO sense. They are stable if and only if

  $$\sum_{n=-\infty}^{+\infty} |h(n)| < +\infty.$$

  IIR systems are stable if and only if the region of convergence of the transfer function includes the unit circle. Indeed,

  $$|H(z)| = \left| \sum_{n=-\infty}^{+\infty} h(n) z^{-n} \right| \leq \sum_{n=-\infty}^{+\infty} \left| h(n) z^{-n} \right| = \sum_{n=-\infty}^{+\infty} |h(n)||z^{-n}|$$

  On the unit circle $|z^{-n}| = 1$:

  $$|H(z)|\Big|_{z=e^{j\omega}} = |H(e^{j\omega})| \leq \sum_{n=-\infty}^{+\infty} |h(n)|.$$

  Thus, if this sum is finite, $|H(z)|\Big|_{z=e^{j\omega}}$ is finite, and the transfer function converges on the unit circle. Conversely, if $H(z)$ converges for $z = e^{j\omega}$, then $\sum_{n=\infty}^{+\infty} |h(n)|$ is finite, and the filter is stable.

- A causal IIR filter is stable if and only if its poles fall within the unit circle.
  Indeed, in case $h(n)$ is causal, the region of convergence of $H(z)$ is external to the circle that encloses all poles. Since for the BIBO stability, the unit circle must belong to the region of convergence, then the poles must all fall within the unit circle.
  We can justify this property also from the observation that (in the partial fraction expansion) to any pole outside the unit circle corresponds an exponentially increasing sequence:

  $$H(z) = \frac{1}{1 - pz^{-1}} \longrightarrow h(n) = p^n \mu(n),$$

  if $|p| > 1$ then $h(n) \to +\infty$.

- The input-output relationship of these filters is always recursive. For this reason, we will see that the quantization errors recirculate.

- We will also see that in causal IIR filters the phase response cannot be linear and that these filters introduce a phase distortion.

Despite these adverse properties, IIR filters adapt better than FIR filters to the realization of the specifications imposed on low-pass, high-pass, and pass-band filters. IIR filters can implement these specifications with a much lower number of coefficients than FIR filters. This property derives from the fact that the transfer function of IIR filters has zeros and poles, while that of FIR filters has only zeros. We will return to these filters later in our course.

For more information study:

S. K. Mitra, "Digital Signal Processing: a computer based approach," 4th edition, McGraw-Hill, 2011
Chapters 6.1-6.4, pp. 277-297
Chapter 6.5, pp. 297-303
Chapter 6.7, pp. 308-311 and pp. 315-319

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:

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

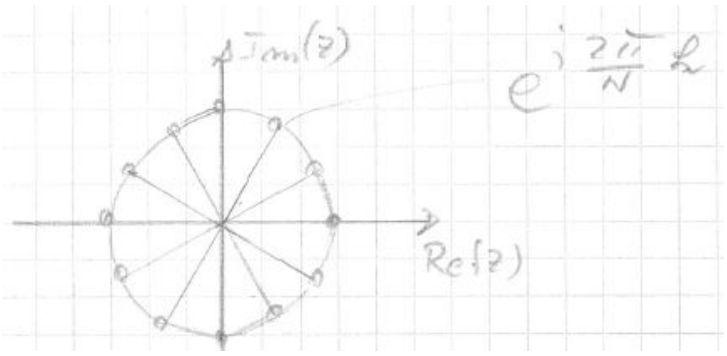
where $\langle n \rangle_N$ indicates the n modulus N (i.e., the remainder of the division of n by N). There exists a bijective correspondence between finite-length sequences of length N and periodic sequences with a period of N. Consequently, we can also apply the DFT to finite-length sequences.

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 $m = n - n_0$)

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

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

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

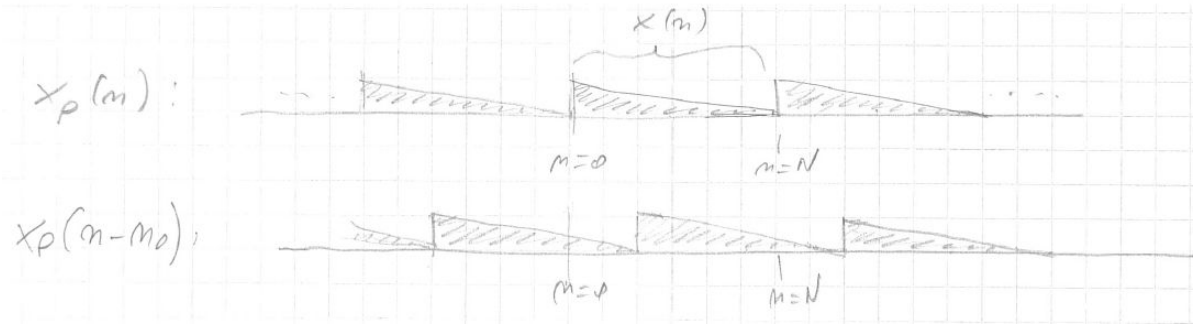
(because $e^{-j\frac{2\pi}{N}mk} = e^{-j\frac{2\pi}{N}(m+N)k}$ and $x_p(m) = x_p(m+N)$)

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

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

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

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

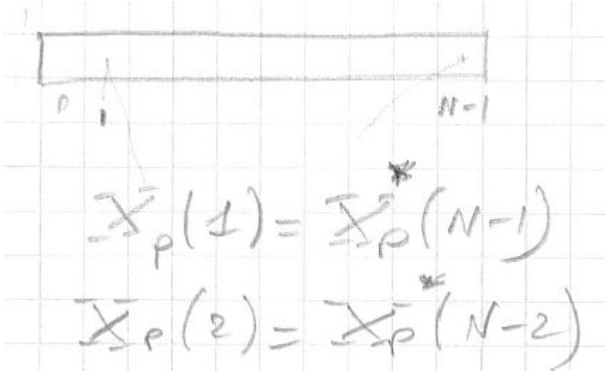
Proof:

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

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

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

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

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

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

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

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 $0 \leq k \leq \frac{N}{2} - 1$,

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

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

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

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

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

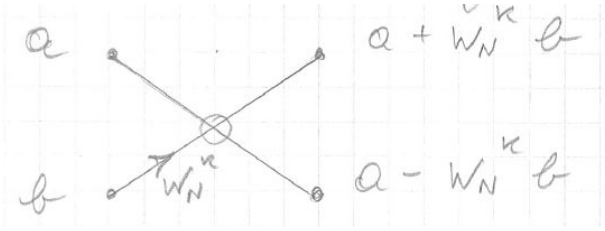
To summarize, the DFT of N samples can be computed using two DFTs on $\frac{N}{2}$ samples with the following equations:

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

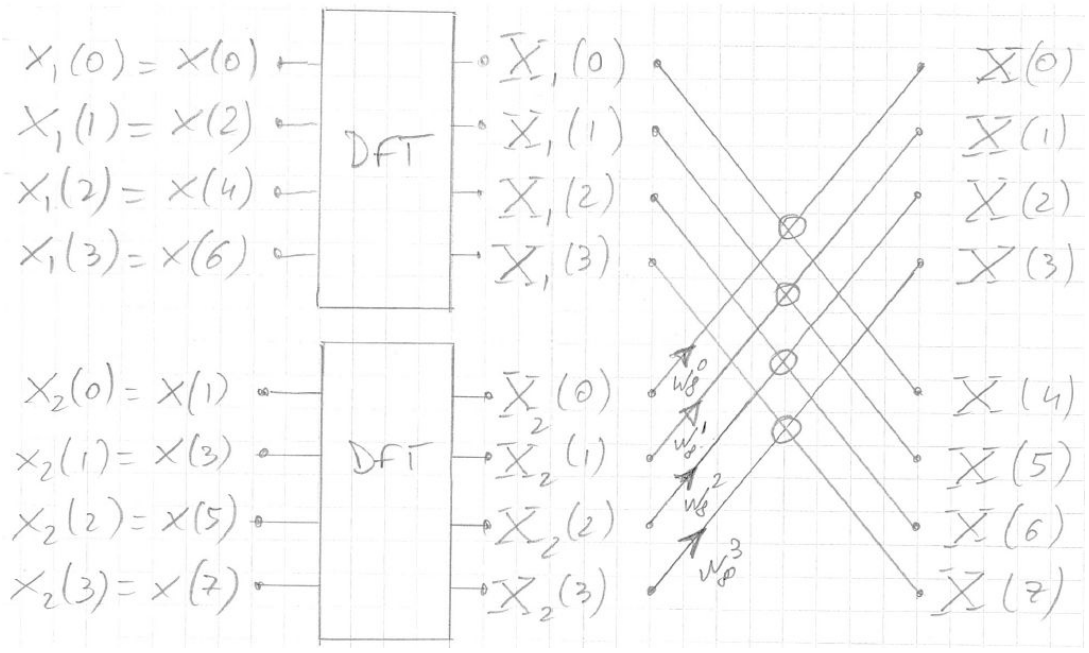
$$X(k + \frac{N}{2}) = X_1(k) - W_N^k X_2(k),$$

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

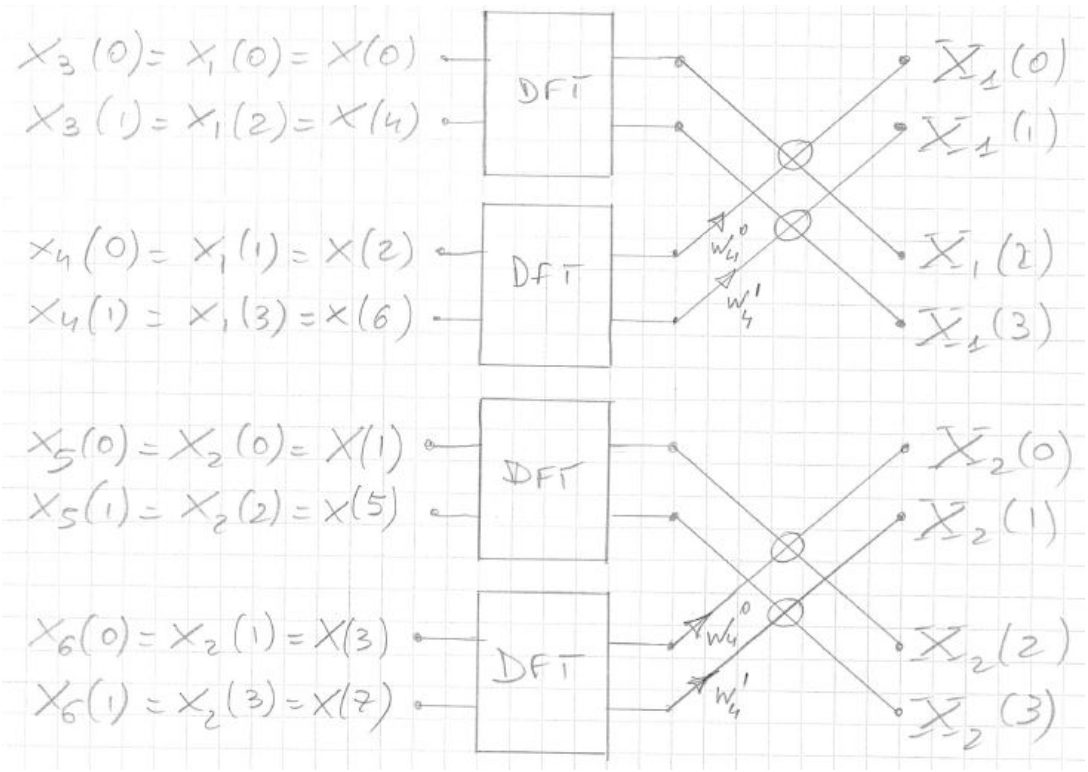
These operations form the fundamental steps of the FFT, and they are visually represented through the following *butterfly diagram*:



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.



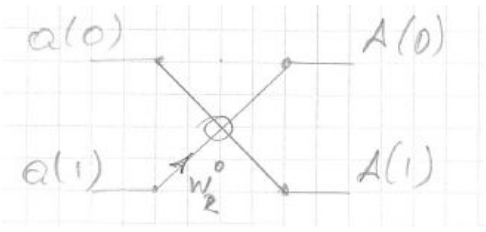
We use the term 'decimation in time' for this algorithm because the length-N sequence is successively decimated into sequences of length $\frac{N}{2}$, $\frac{N}{4}$, $\frac{N}{8}$, ..., 2.

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

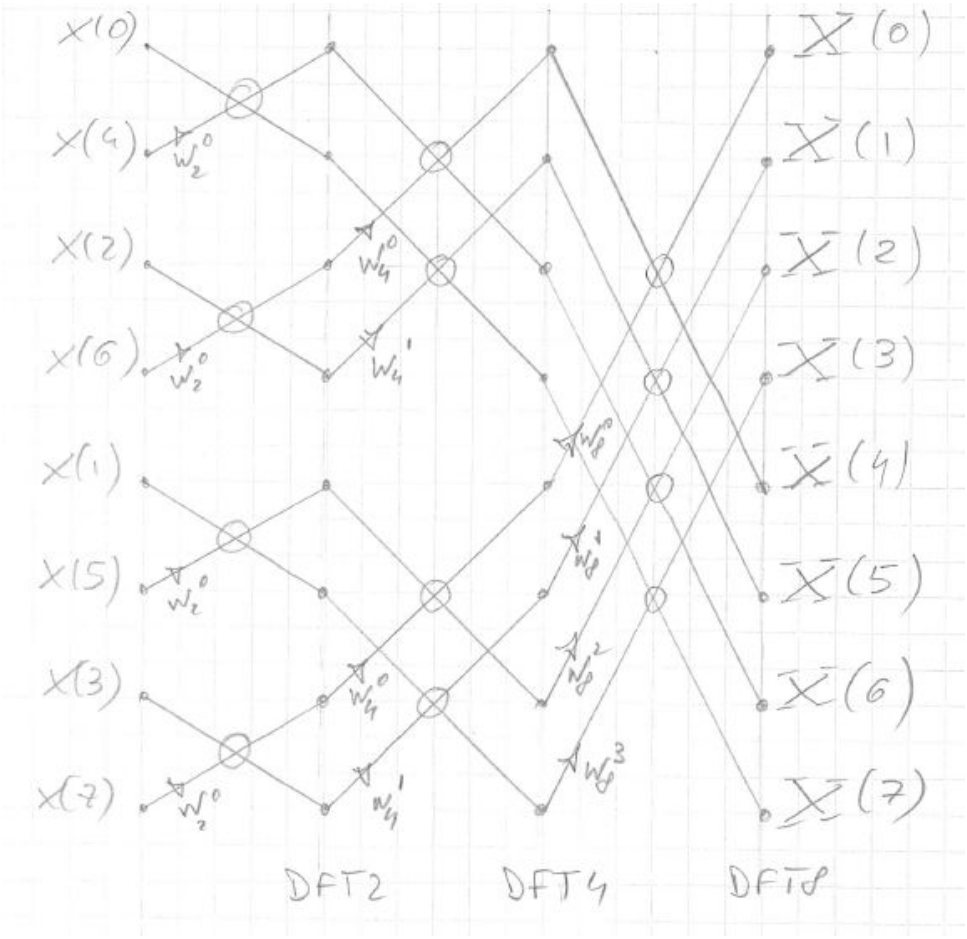
$$A(k) = \sum_{n=0}^{1} a(n) W_2^{nk} = \sum_{n=0}^{1} a(n) e^{-j\frac{2\pi}{2}nk} = a(0) + e^{-j\pi k} a(1)$$

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

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



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

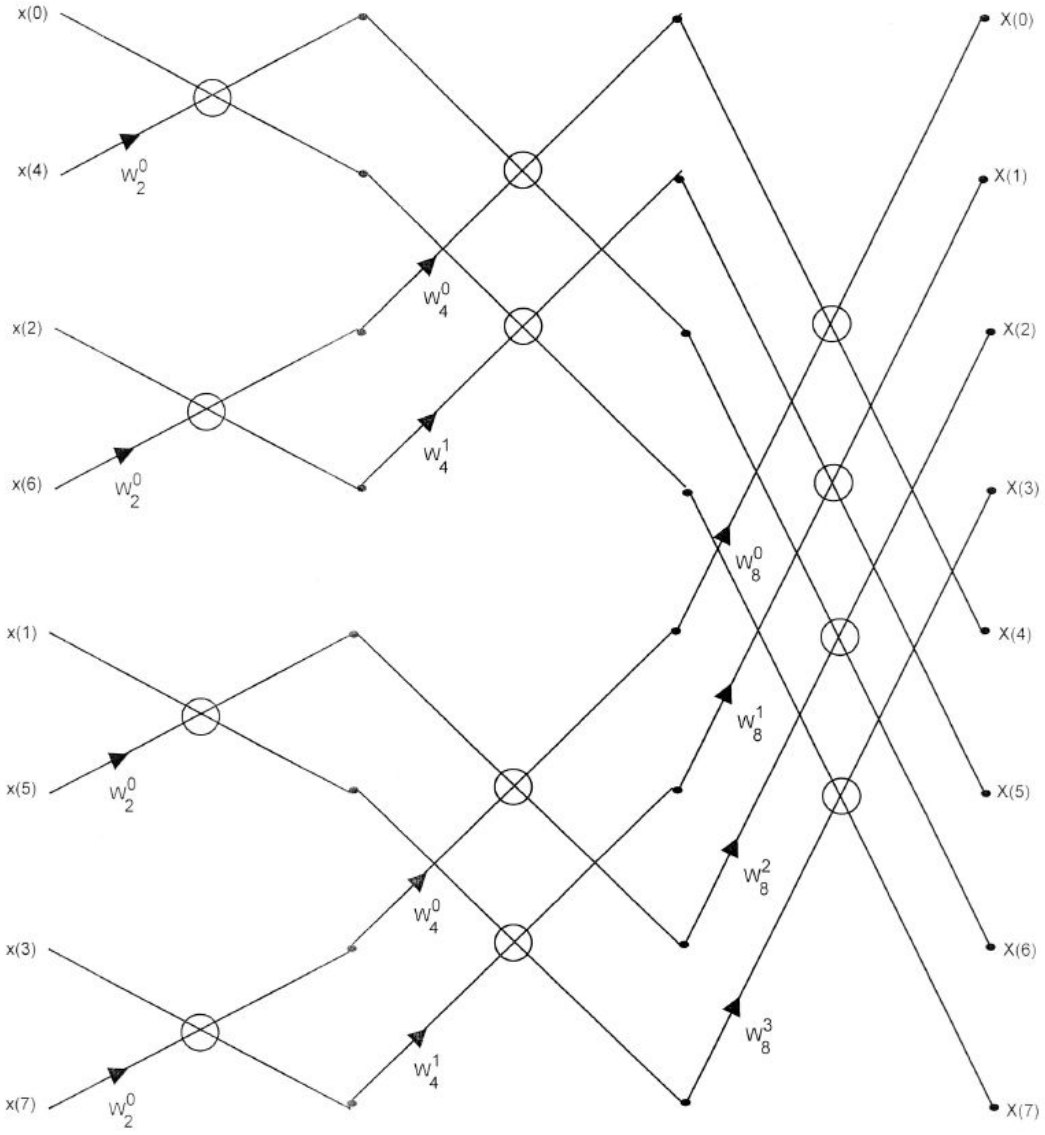


Figure 06.01: Decimation-in-time FFT

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)

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

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

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:

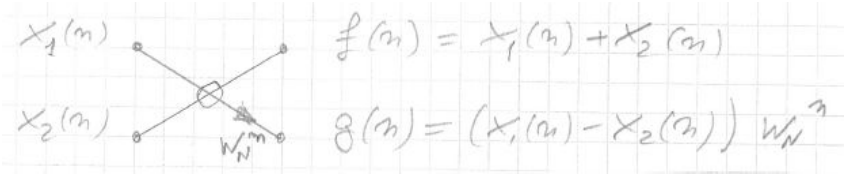
$$f(n) = x_1(n) + x_2(n)$$

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

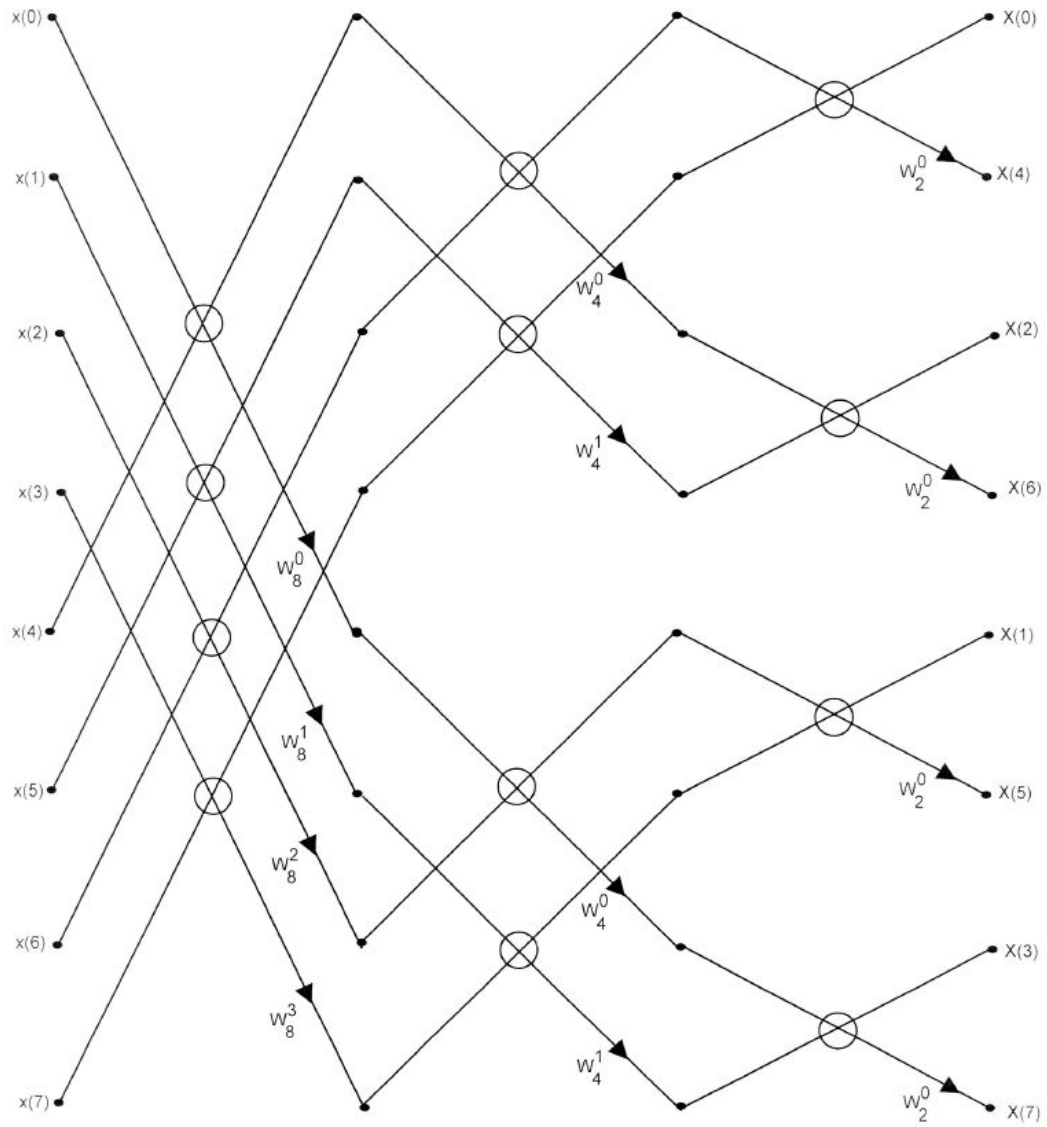
with $n = 0, 1, \ldots, N/2 - 1$, and then computing their $\frac{N}{2}$ samples DFT. The DFT of $f(n)$ provides the even terms of $X(k)$, and the DFT of $g(n)$ provides the odd terms of $X(k)$.

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. $y(n) = 0 \; \forall n < 0$ and $\forall n \geq 2N-1$. Indeed,

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

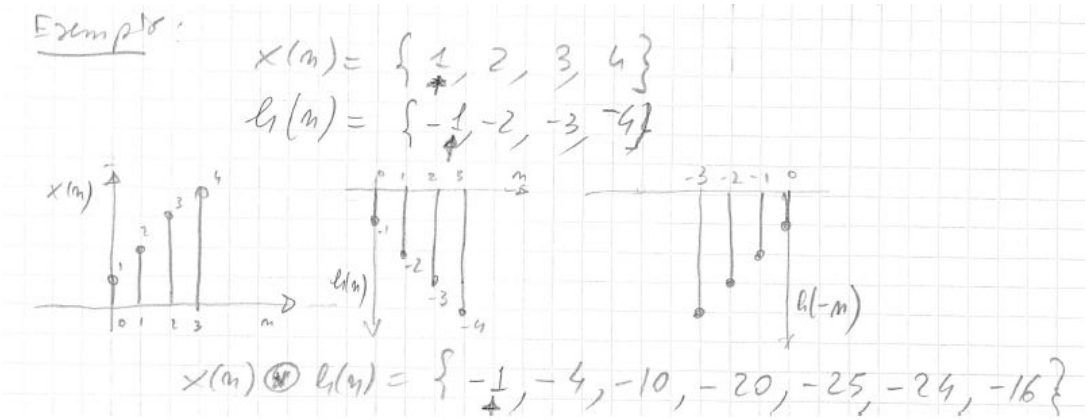
$$y(1) = x(0)h(1) + x(1)h(0)$$

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

$$\vdots$$

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

Example:

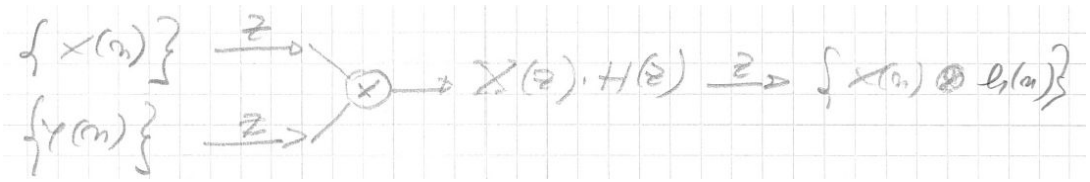


We have seen that an important property of DTFT and Z-Transform is the transformation of linear convolution into the product of the transforms of the two sequences:

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

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

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

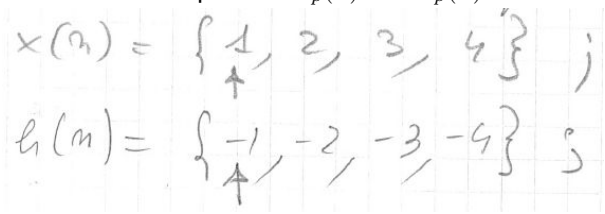
$$\sum_{m=0}^{N-1} x_p(m) h_p(n-m)$$

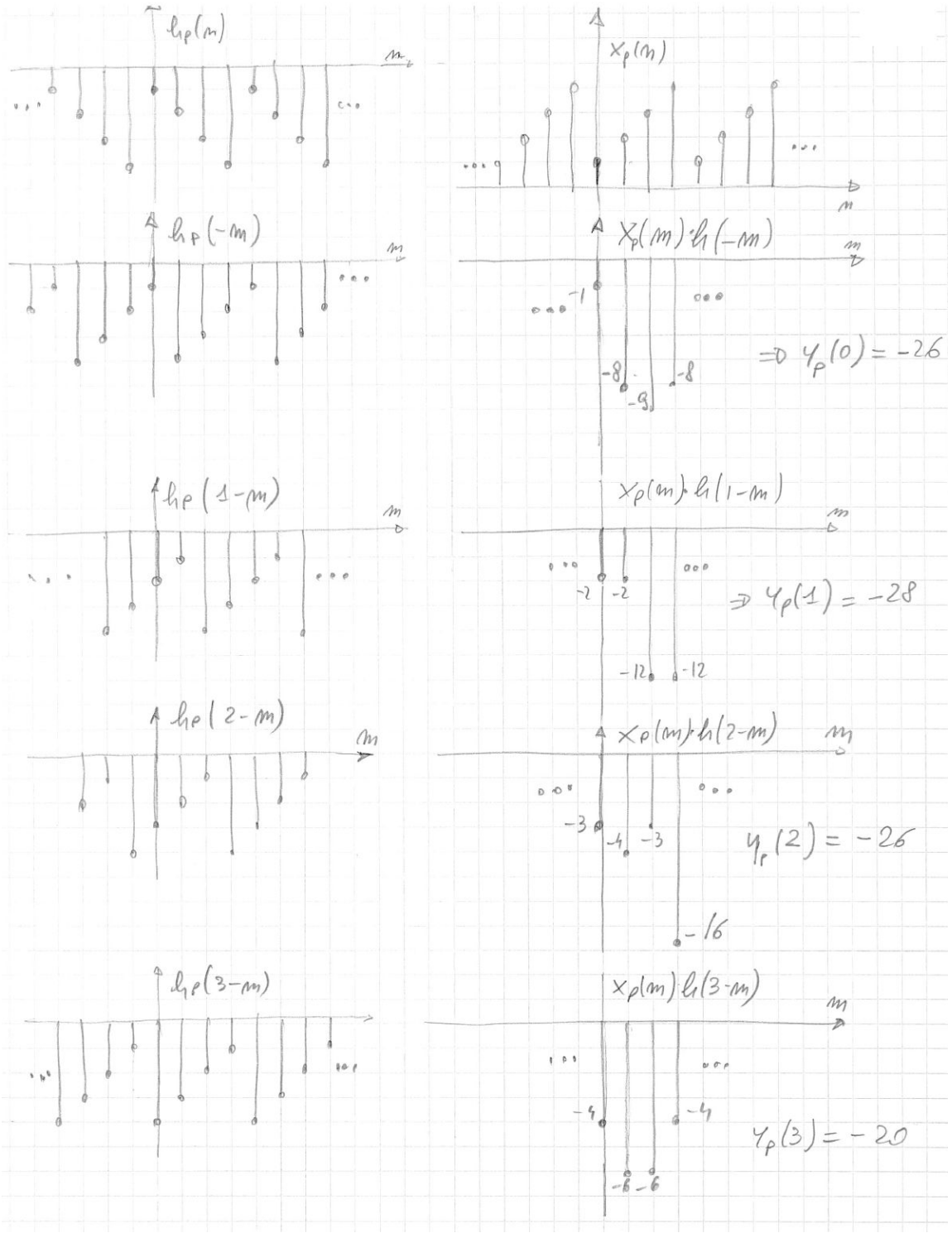
we need to:

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 $m = N-1$.

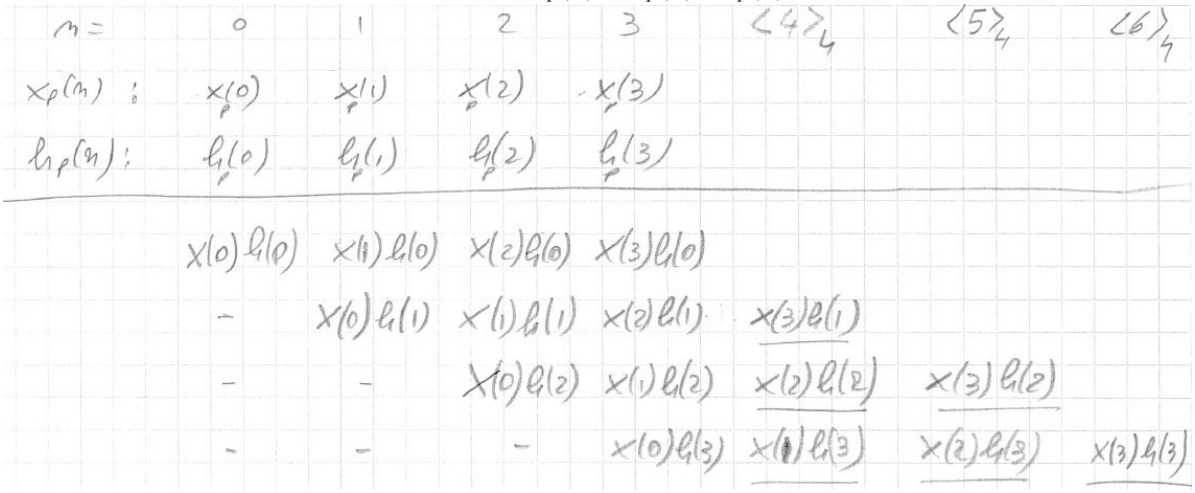
The main difference with the linear convolution comes from the fact that the sequences $x_p(n)$ and $h_p(n)$ are periodic with a period of N and that the sum is performed only over one period (from 0 to $N-1$).

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

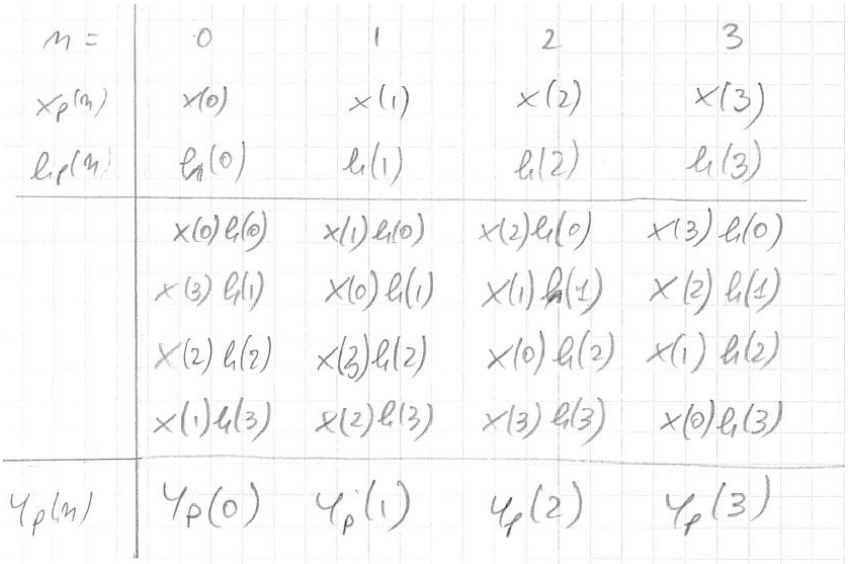




For computing the circular convolution, we can apply the tabular method we have seen for the linear convolution, with the only difference that we have to wrap around in the window $[0, N-1]$ the partial products of the second, third, ..., N-th row, that fall outside this window.
Let us consider the period 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:



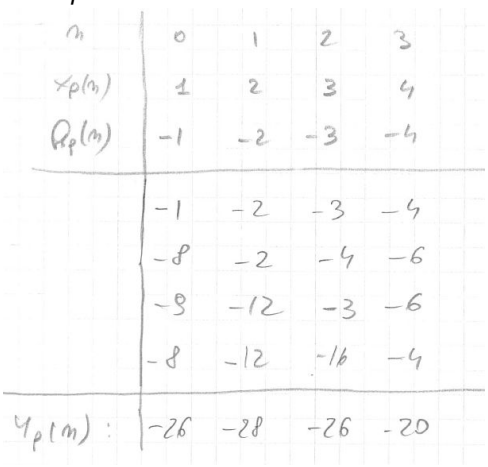
$$y_p(0) = x(0)h(0) + x(3)h(1) + x(2)h(2) + x(1)h(3)$$

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

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

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

Example:



n	0	1	2	3
$x_p(n)$	1	2	3	4
$h_p(n)$	-1	-2	-3	-4
	-1	-2	-3	-4
	-8	-2	-4	-6
	-9	-12	-3	-6
	-8	-12	-16	-4
$y_p(n)$:	-26	-28	-26	-20

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 $h(n)$,

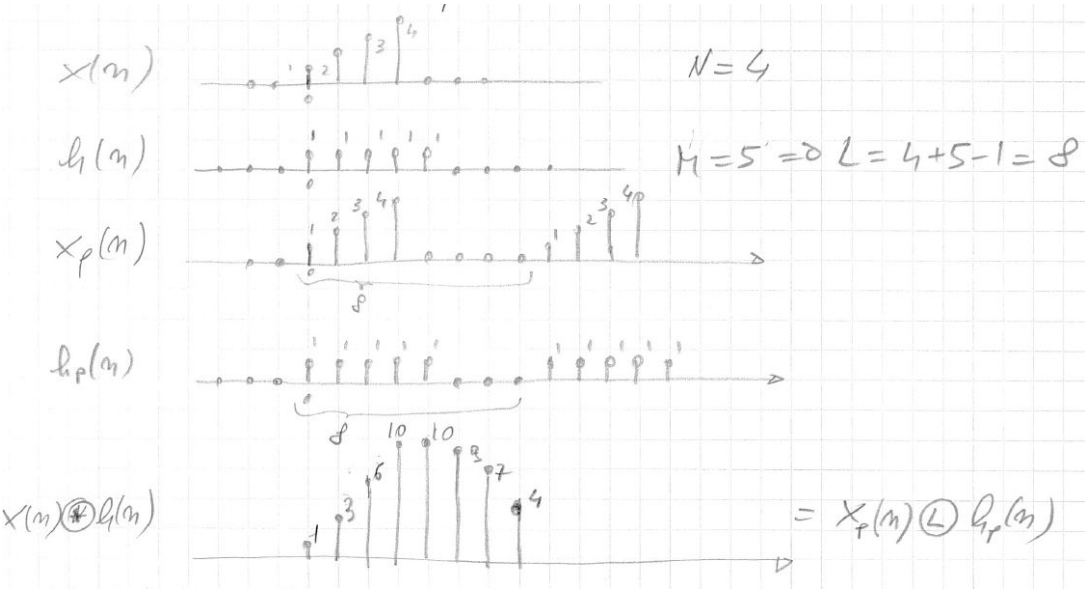
$$y(n) = \sum_{m=0}^{N-1} x(m)h(n-m)$$

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 $0 \leq m \leq N-1$

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

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

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

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

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

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

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

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

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

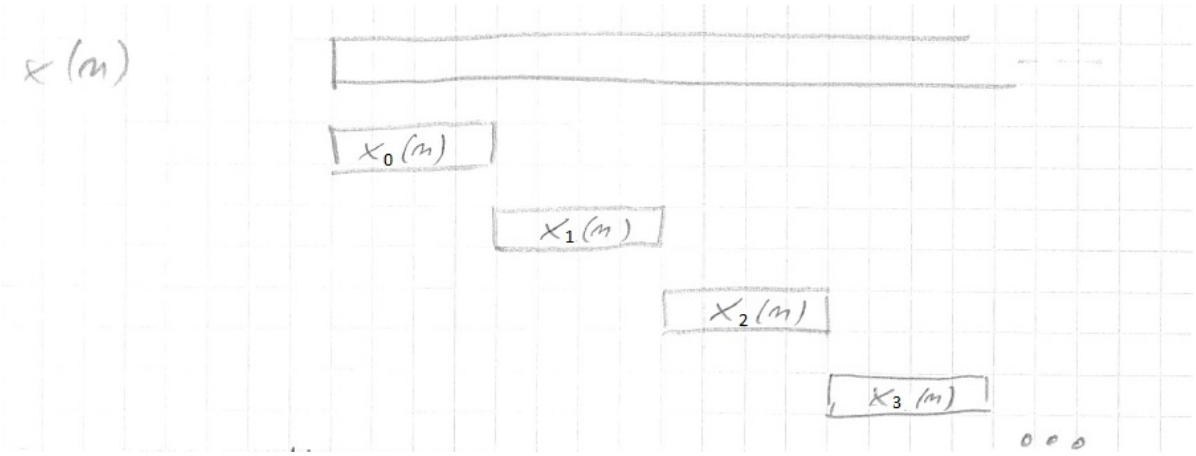
Overlap-add method.

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:

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

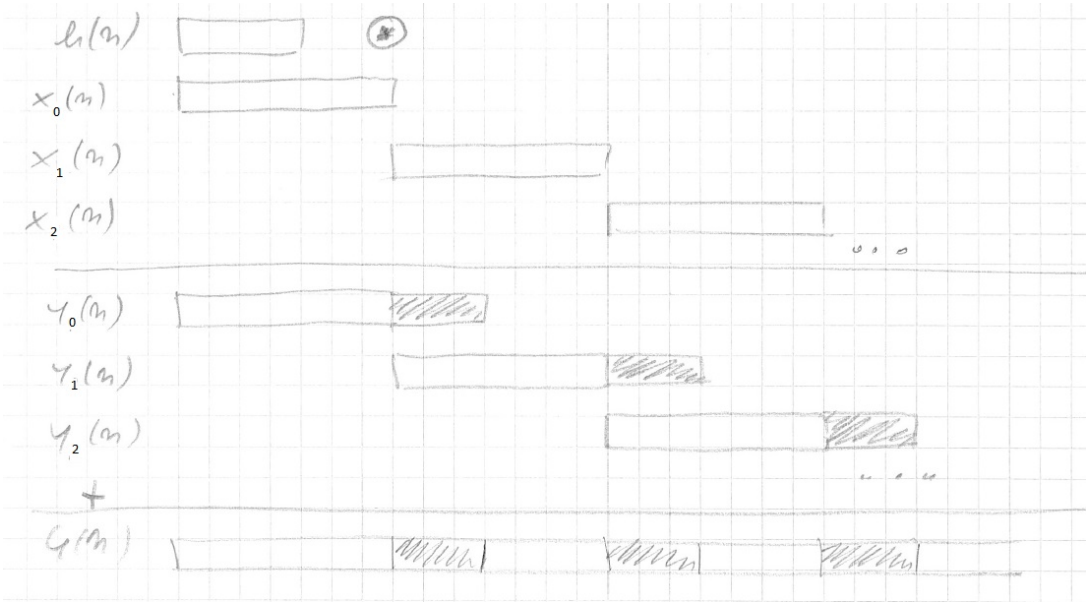
$$= \sum_{m=0}^{+\infty} \left[\sum_{l=0}^{M-1} h(l)x_m(n - mN - l) \right] = \sum_{m=0}^{+\infty} y_m(n - mN)$$

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.

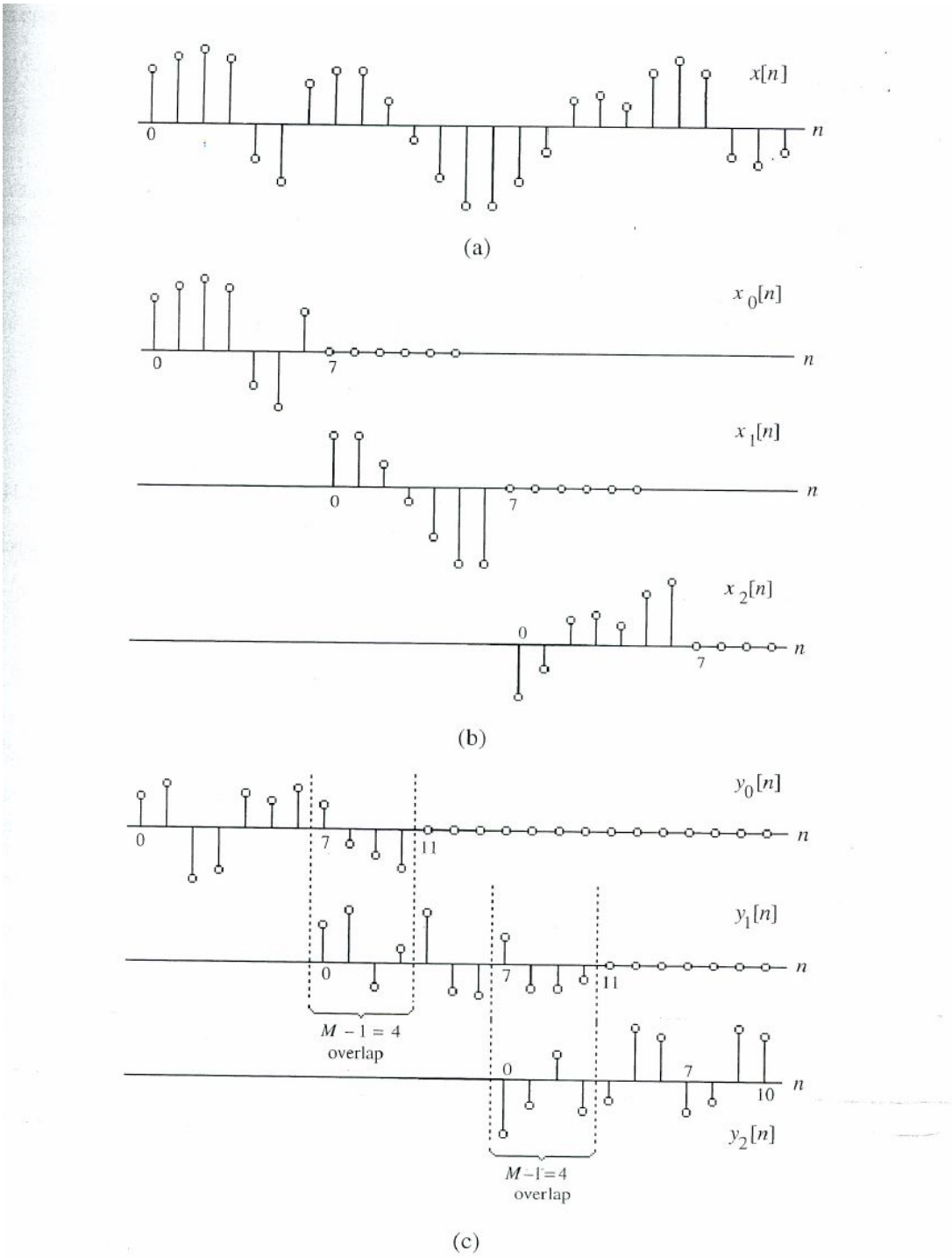


Figure 5.14: (a) Original $x[n]$, (b) segments $x_m[n]$ of $x[n]$, and (c) linear convolution of $x_m[n]$ with $h[n]$.

(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 $L = M + N - 1$, $g(n) = 0$

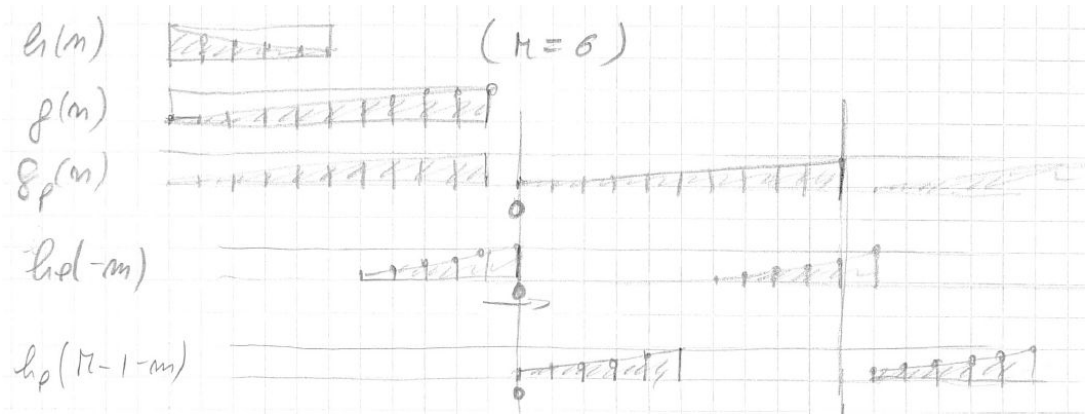
$\forall n < 0$ and $\forall n \geq L$. Let $h_p(n)$ and $g_p(n)$ be obtained from the periodic repetition of $h(n)$ and $g(n)$ with period L. The circular convolution $h_p(n) \textcircled{L} g_p(n) = y_p(n)$ is given by:

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

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

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

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 $0 \leq m \leq L-1$,

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

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

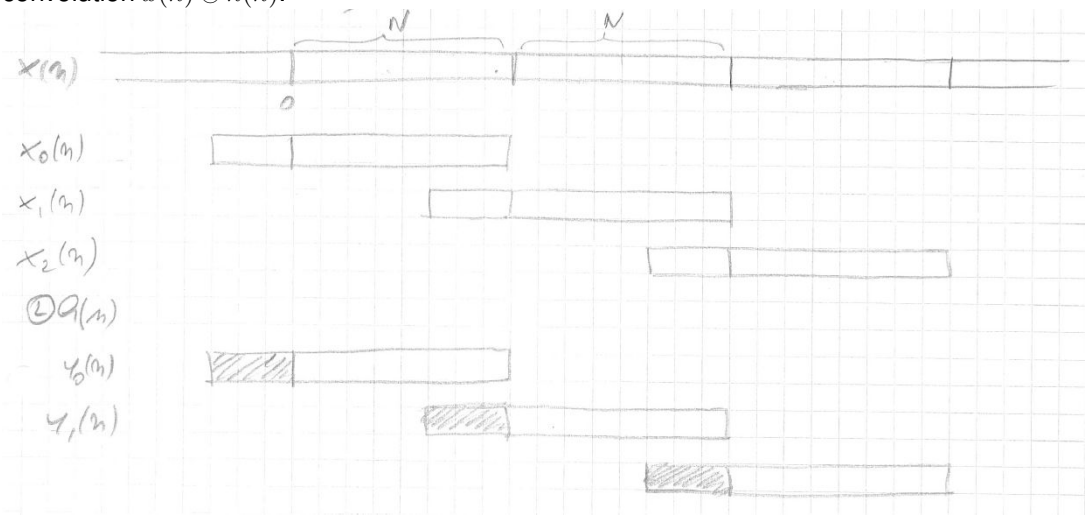
Thus,

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

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

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.

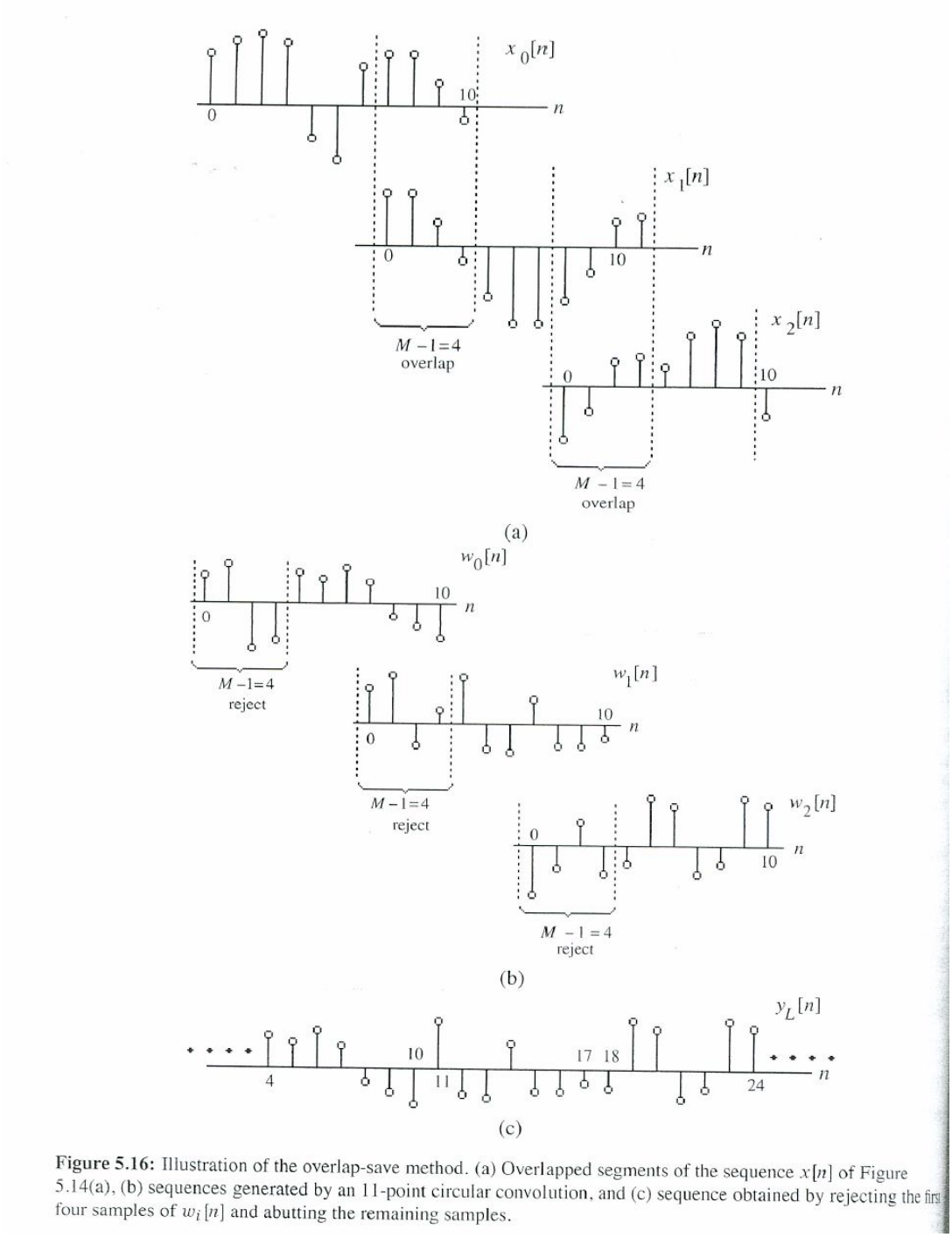


Figure 5.16: Illustration of the overlap-save method. (a) Overlapped segments of the sequence $x[n]$ of Figure 5.14(a), (b) sequences generated by an 11-point circular convolution, and (c) sequence obtained by rejecting the first four samples of $w_i[n]$ and abutting the remaining samples.

(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

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

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

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

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

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

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

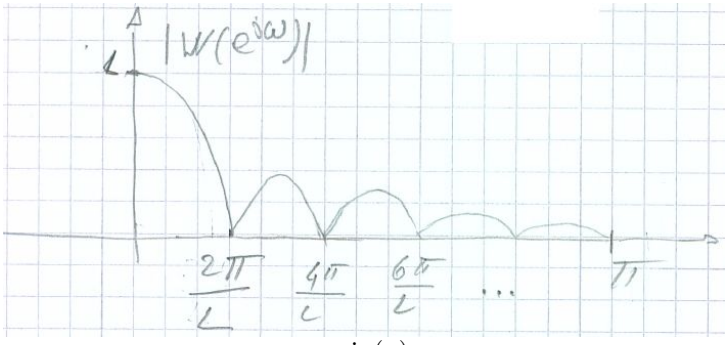
The spectrum of the rectangular function $W(e^{j\omega})$ is given by

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

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

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

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



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:

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

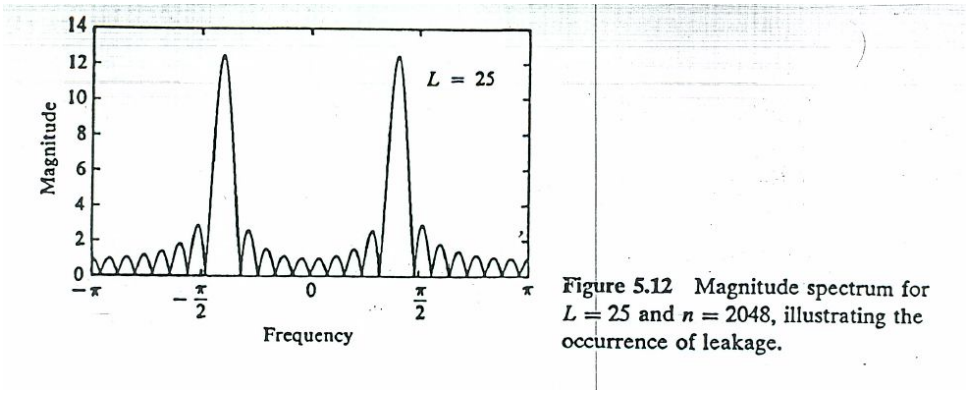


Figure 5.12 Magnitude spectrum for $L = 25$ and $n = 2048$, illustrating the occurrence of leakage.

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 $\omega_2 = 0.6\pi$

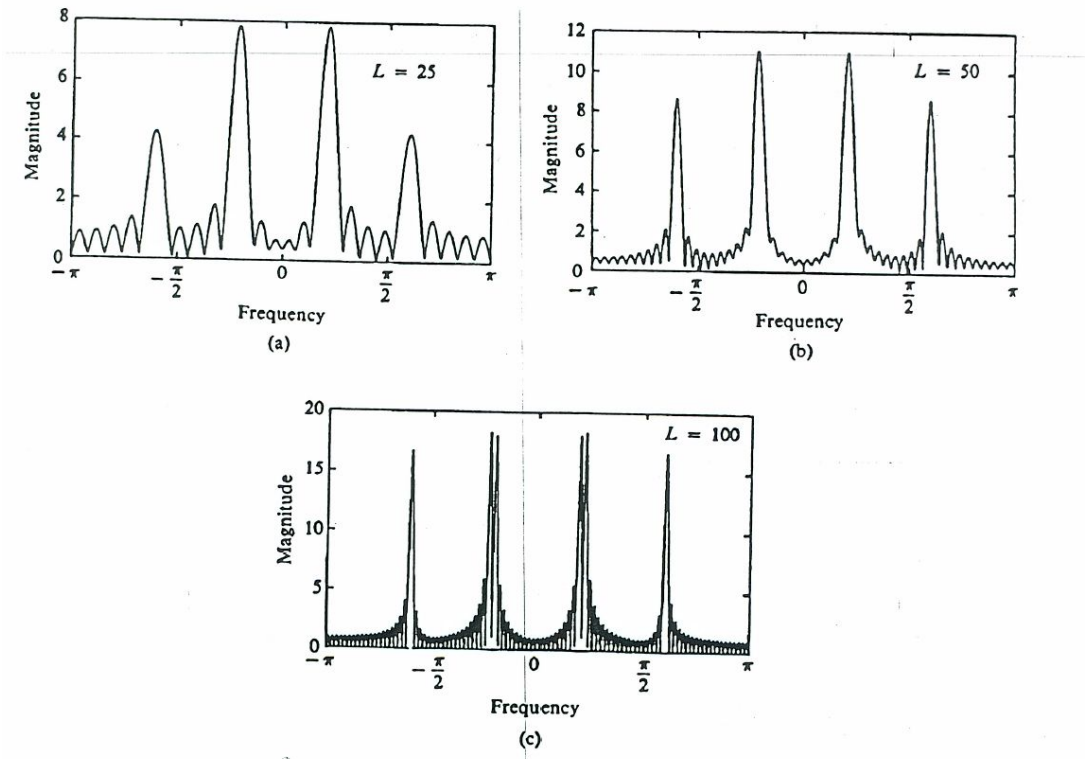


Figure 5.13 Magnitude spectrum for the signal given by (5.4.8), as observed through a rectangular window.

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	$\left\{\dfrac{\sin\left[2\pi\left(n - \frac{M-1}{2}\right)\Big/(M-1)\right]}{2\pi\left(n - \frac{M-1}{2}\right)\Big/\left(\frac{M-1}{2}\right)}\right\}^L \quad L > 0$
Tukey	$1, \left\lvert n - \dfrac{M-1}{2}\right\rvert \le \alpha\dfrac{M-1}{2} \quad 0 < \alpha < 1$ $\dfrac{1}{2}\left[1 + \cos\left(\dfrac{n - (1+\alpha)(M-1)/2}{(1-\alpha)(M-1)/2}\pi\right)\right]$ $\alpha(M-1)/2 \le \left\lvert n - \dfrac{M-1}{2}\right\rvert \le \dfrac{M-1}{2}$

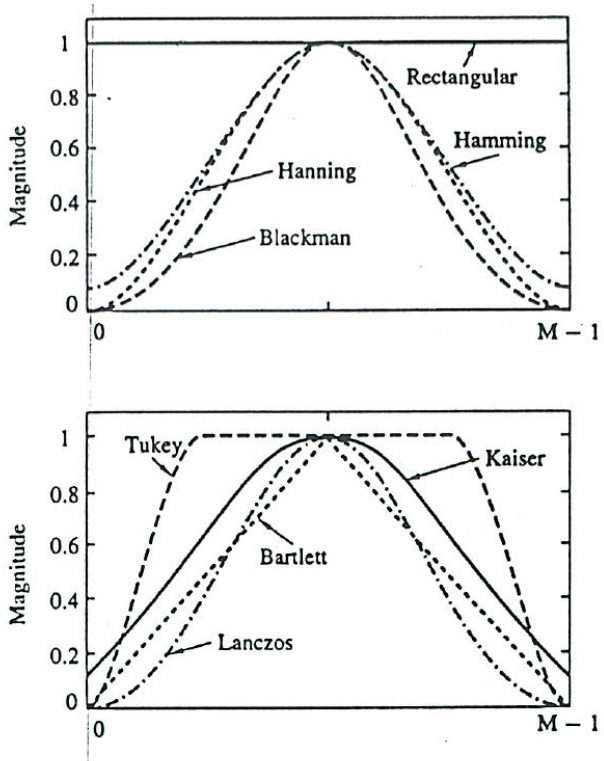


Figure 8.5 Shapes of several window functions.

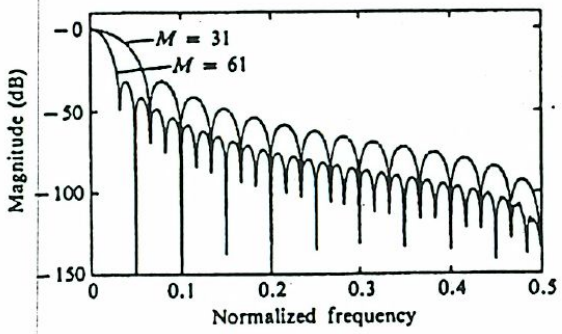


Figure 8.6 Frequency responses of Hanning window for (a) $M = 31$ and (b) $M = 61$.

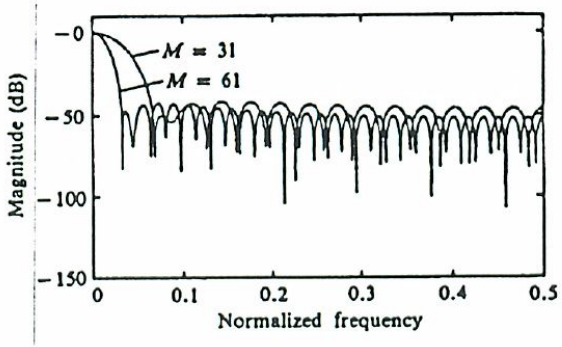


Figure 8.7 Frequency responses for Hamming window for (a) $M = 31$ and (b) $M = 61$.

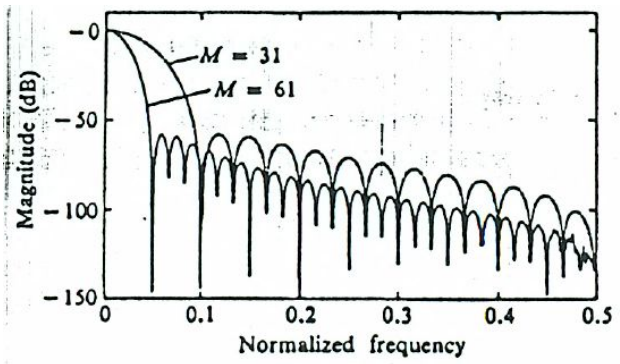


Figure 8.8 Frequency responses for Blackman window for (a) $M = 31$ and (b) $M = 61$.

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.

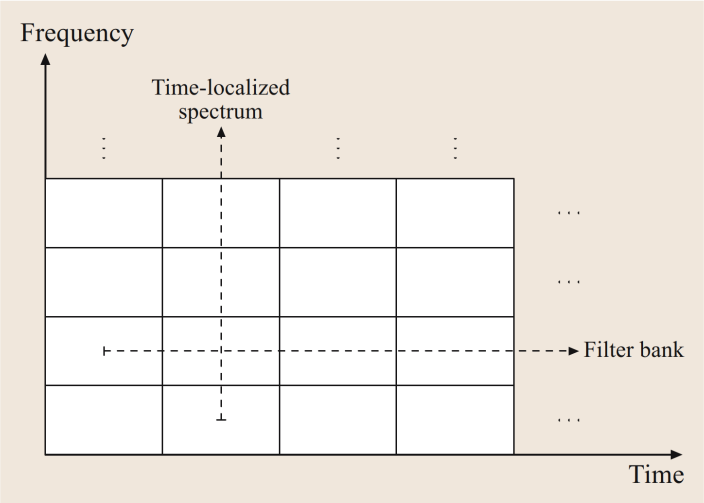


Fig. 12.1 Interpretations of the short-time Fourier transform as a series of time-localized spectra (*vertical*) and as a bank of bandpass filters (*horizontal*)

(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:

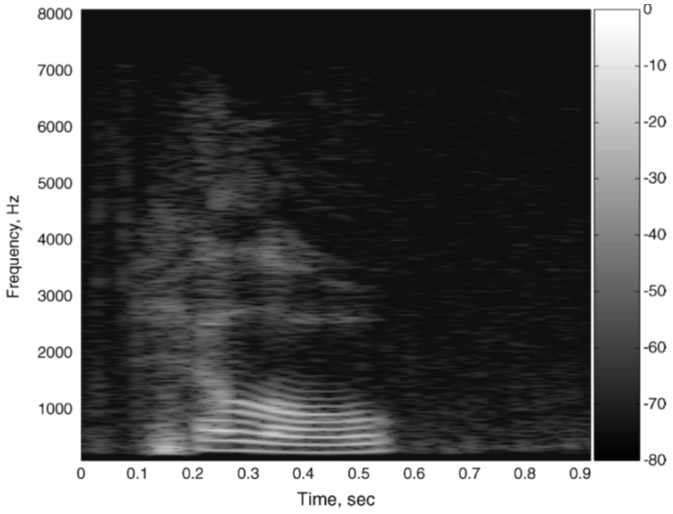
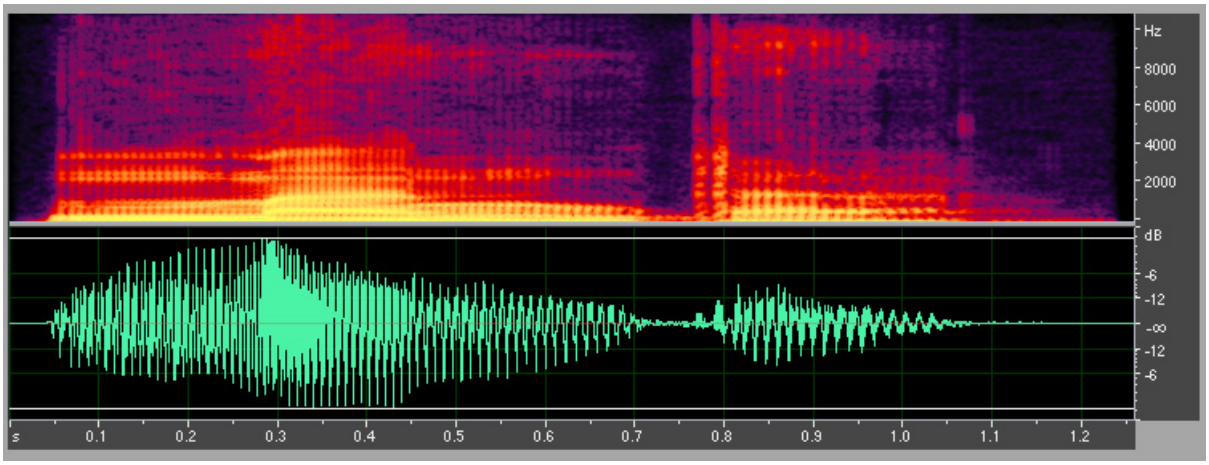


Figure 2.16 Spectrogram of English word "hello"

(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

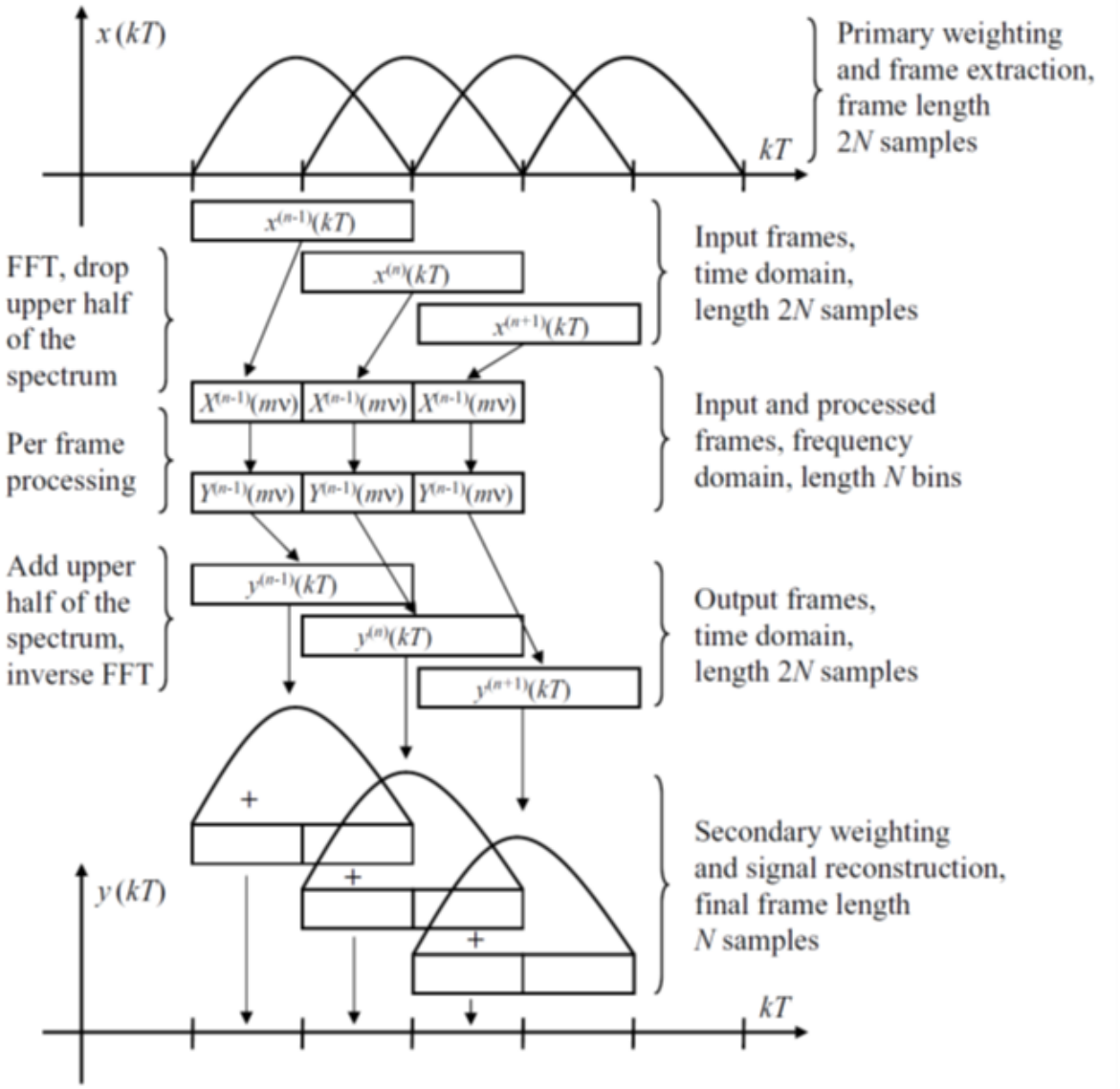
$$v(m) = w(m) = \sqrt{w_{\mathrm{PR}}(m)}.$$

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

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

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),

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

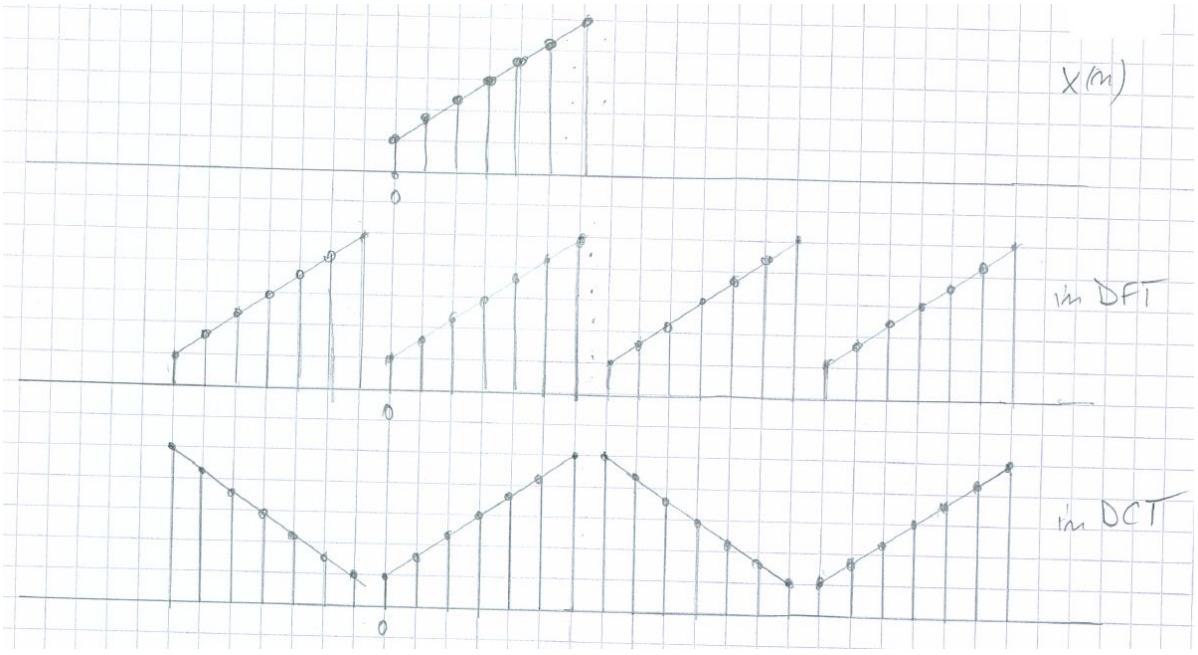
(because we consider $g(n,k)$ and we sum on k).
The two transforms are reciprocal if and only if

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

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,

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

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

Taking the first N terms we have

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

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

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

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

07 Discrete-time LTI systems in the frequency domain

07.01 Ideal filters

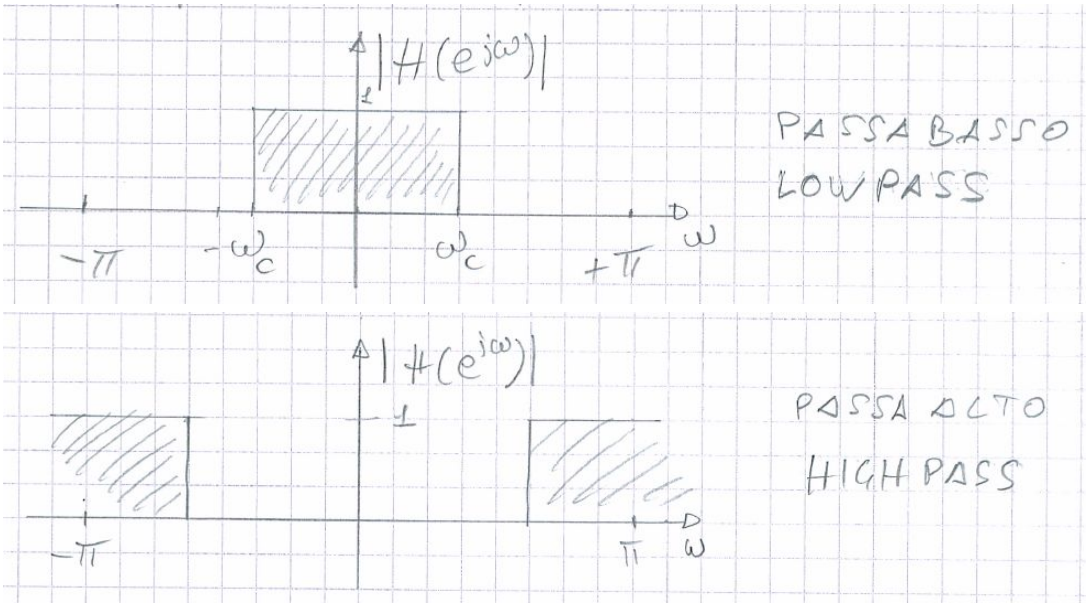
One of the most important applications of LTI systems is allowing the passage of certain frequency components of the signal without any distortion while simultaneously blocking all other frequency components. For this reason, LTI systems are also defined as *'filters.'* The two terms, 'LTI system' and 'filter,' can be considered synonymous. In fact, the term 'filter' is used not only for systems that are frequency-selective but also for all systems that realize an appropriate weighting (a 'spectral shaping') of the signal spectrum:

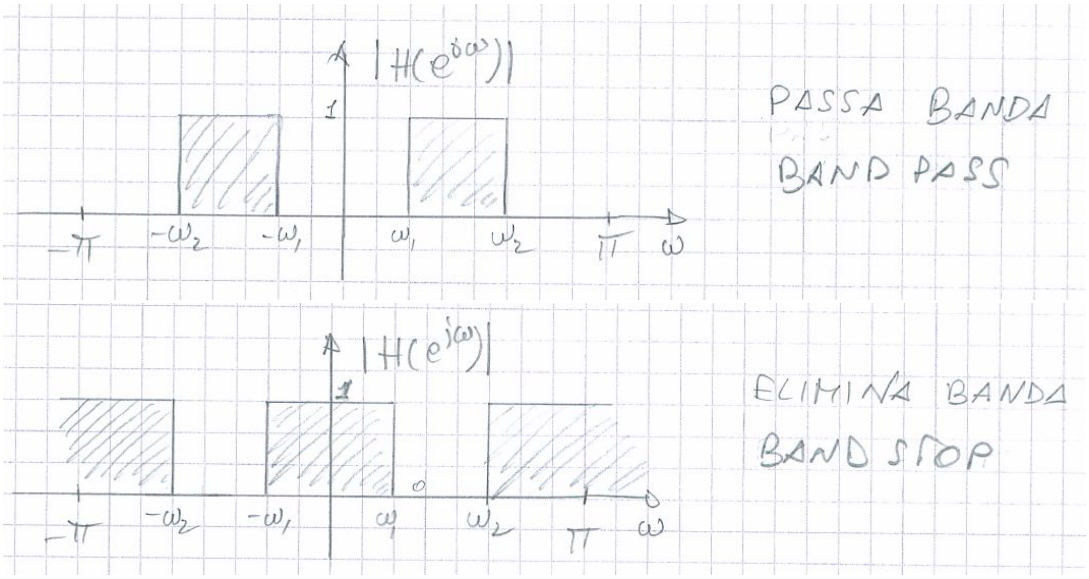
$$Y(e^{j\omega}) = H(e^{j\omega}) \cdot X(e^{j\omega})$$

Filters can be classified according to their frequency domain characteristics as

- lowpass filters,
- highpass filters,
- bandpass filters,
- bandstop filters.

The ideal characteristics of the frequency responses of these filters are as follows:





An *ideal filter* allows certain frequency components to pass unaltered while completely eliminating all other frequency components. Therefore, an ideal filter exhibits a unit magnitude (or amplitude) response in the passband and a zero response in the stopband. Additionally, an ideal filter must feature linear phase in the passband.

Considering a signal $\{x(n)\}$ with a spectrum entirely within the band $\omega_1 \leq |\omega| \leq \omega_2$, let's filter it with a frequency response given by:

$$H(e^{j\omega}) = \begin{cases} e^{-j\omega n_0} & \omega_1 \leq |\omega| \leq \omega_2 \\ 0 & \text{otherwise} \end{cases}$$

In this case, the output signal has a spectrum given by:

$$Y(e^{j\omega}) = H(e^{j\omega}) \cdot X(e^{j\omega}) = e^{-j\omega n_0} X(e^{j\omega}).$$

This implies that:

$$y(n) = x(n - n_0).$$

The output signal coincides with the input signal except for a delay of n_0. Generally, a pure delay is tolerable and is not considered signal distortion. If the ideal filter has linear phase, the signal component in the passband is delayed without distortion. Thus, the ideal phase response is linear in the passband:

$$\theta(\omega) = -\omega n_0.$$

In practice, we are contented with imposing $-\dfrac{\mathrm{d}\theta}{\mathrm{d}\omega}$ to be constant in the passband, i.e., with imposing the group delay to be constant in the passband.

07.02 Phase delay and Group delay

Let's consider a LTI system with frequency response $H(e^{j\omega})$, and let $\theta(\omega)$ be its phase response,

$$H(e^{j\omega}) = |H(e^{j\omega})| \cdot e^{j\theta(\omega)}.$$

For simplicity, let's assume the system to be real, so that

$$|H(e^{j\omega})| = |H(e^{-j\omega})| \qquad \text{and} \qquad \theta(\omega) = -\theta(-\omega)$$

If we consider a sinusoidal sequence with normalized angular frequency ω_0 as system input:

$$x(n) = \cos(\omega_0 n) = \frac{e^{j\omega_0 n} + e^{-j\omega_0 n}}{2},$$

the output of the system is

$$y(n) = \frac{1}{2}|H(e^{j\omega_0})|e^{j\theta(\omega_0)}e^{j\omega_0 n} + \frac{1}{2}|H(e^{-j\omega_0})|e^{j\theta(-\omega_0)}e^{-j\omega_0 n} =$$

$$= \frac{1}{2}|H(e^{j\omega_0})|\left(e^{j(\omega_0 n+\theta(\omega_0))} + e^{-j(\omega_0 n+\theta(\omega_0))}\right) =$$

$$= |H(e^{j\omega_0})|\cos[\omega_0 n + \theta(\omega_0)] =$$

$$= |H(e^{j\omega_0})|\cos\left[\omega_0\left(n + \frac{\theta(\omega_0)}{\omega_0}\right)\right] =$$

$$= |H(e^{j\omega_0})|\cos\left[\omega_0\left(n - t_p(\omega_0)\right)\right]$$

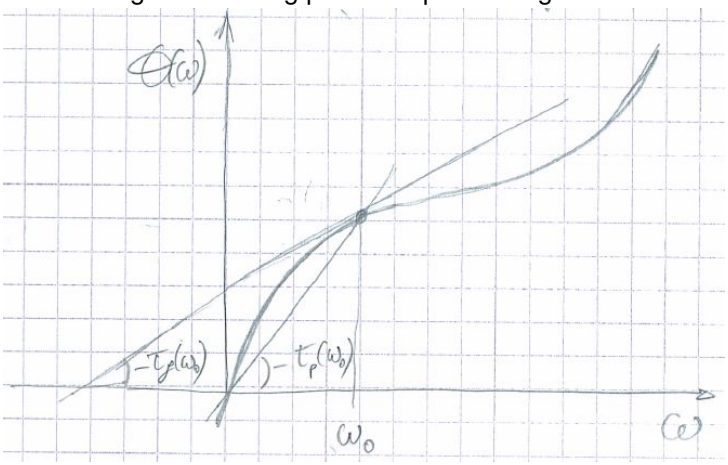
The system output is the same sinusoidal sequence delayed by

$$t_p(\omega_0) = -\frac{\theta(\omega_0)}{\omega_0}.$$

$t_p(\omega)$ is referred to as the *Phase Delay*, representing the delay of a sinusoidal component as it passes from the input to the output of the system.
However, when we consider a signal composed of multiple frequency components (several sinusoids), each component passing through the system experiences a different delay. In such cases, the delay introduced by the system on the signal is assessed using another parameter known as the *Group Delay*, defined as:

$$t_g(\omega) = -\frac{\mathrm{d}\theta(\omega)}{\mathrm{d}\omega}.$$

It's important to note that both $t_p(\omega)$ and $t_g(\omega)$ vary with frequency.
Considering the following phase response diagram:



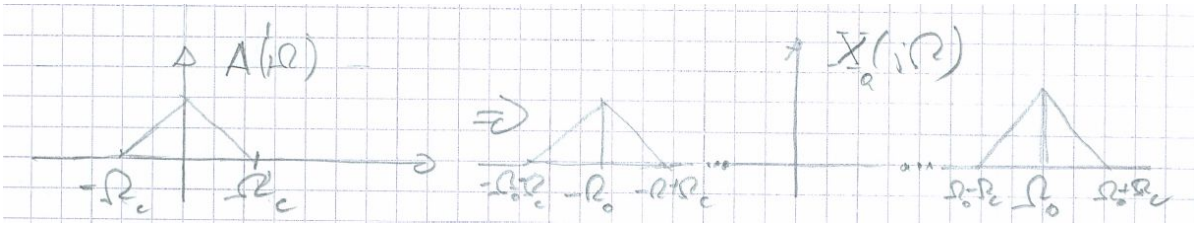
the group delay $t_g(\omega)$ corresponds to the opposite of the slope of $\theta(\omega)$ in ω_0. On the contrary, the phase

delay $t_p(\omega)$ corresponds to the opposite of the slope of the line connecting the origin with the point $(\omega_0, \theta(\omega_0))$.

Why is the delay introduced by the system on the signal evaluated using the group delay $t_g(\omega)$?
This choice is particularly relevant in Amplitude Modulation (AM) systems, where the group delay represents the delay introduced on the modulated signal. In contrast, the phase delay corresponds to the delay of the modulating signal, i.e., the carrier.
Let's return to the continuous-time domain and consider a lowpass signal (for example, think about a musical signal), $a(t)$, with a passband $[-\Omega_c, \Omega_c]$. Now, let's modulate this signal with a carrier having an angular frequency $\Omega_0 \gg \Omega_c$. In other words, we multiply the signal by a sinusoidal signal with an angular frequency Ω_0:

$$x_a(t) = a(t) \cdot \cos(\Omega_0 t) = a(t) \cdot \left[e^{j\Omega_0 t} + e^{-j\Omega_0 t}\right]/2$$

$$X_a(j\Omega) = \frac{1}{2} A\left(j(\Omega - \Omega_0)\right) + \frac{1}{2} A\left(j(\Omega + \Omega_0)\right)$$



The spectrum of the signal $a(t)$ is translated to $\pm\Omega_0$ and occupies the band[1]
$\pm\left[\Omega_0 - \Omega_c, \Omega_0 + \Omega_c\right]$.
Let us assume that the signal $x_a(t)$ passes through an LTI system with frequency response $H(j\Omega)$[2]. Since $\Omega_c \ll \Omega_0$, within the band $[\Omega_0 - \Omega_c, \Omega_0 + \Omega_c]$, we can assume the amplitude response of the LTI system to be constant (for simplicity, let's assume it equals 1). Additionally, we can approximate the phase response with a linear response:

$$\theta(\Omega) \simeq \theta(\Omega_0) + \frac{\mathrm{d}\theta(\Omega)}{\mathrm{d}\Omega}\Big|_{\Omega=\Omega_0} (\Omega - \Omega_0) =$$

$$= -t_p(\Omega_0) \cdot \Omega_0 - t_g(\Omega_0) \cdot (\Omega - \Omega_0)$$

For $\Omega > 0$, the output signal spectrum is:

$$Y_a(j\Omega) = \frac{1}{2} A\left(j(\Omega - \Omega_0)\right) e^{-jt_p(\Omega_0)\Omega_0} e^{-jt_g(\Omega_0)(\Omega - \Omega_0)}.$$

[1] With the AM modulation, the bandwidth of the signal is twice that of the "baseband" signal $a(t)$.

[2] Similar to discrete-time LTI systems, for continuous-time systems, the impulse response and the frequency response are sufficient to completely characterize the LTI system. The impulse response of a continuous-time system, denoted as $h(t)$, is the response to a Dirac pulse $\delta(t)$, and the system output is given by:

$$y(t) = \int_{-\infty}^{+\infty} h(\tau) x(t - \tau) \mathrm{d}\tau.$$

This expression is known as the convolution integral. The frequency response, $H(j\Omega)$, is the CTFT of the impulse response $h(t)$ and can be expressed as:

$$Y(j\Omega) = H(j\Omega) X(j\Omega)$$

where $X(j\Omega)$ and $Y(j\Omega)$ are the CTFTs of $x(t)$ and $y(t)$, respectively. The definitions of phase response, magnitude (or amplitude) response, phase delay, and group delay remain the same as in the discrete-time case.

For $\Omega < 0$, the output signal spectrum is the conjugate symmetric of the spectrum for $\Omega > 0$.
It is easy to verify that the system output is given by:

$$y_a(t) = a(t - t_g)\cos[\Omega_0(t - t_p)]$$

with $t_g = t_g(\Omega_0)$ and $t_p = t_p(\Omega_0)$. In fact, $a(t - t_g)$ has spectrum $A(j\Omega) \cdot e^{-jt_g\Omega}$. When $a(t - t_g)$ is multiplied by $\cos[\Omega_0(t - t_p)] = \dfrac{e^{j\Omega_0(t-t_p)} - e^{-j\Omega_0(t+t_p)}}{2}$, two components with conjugate symmetry are generated: one is centered at Ω_0, the other at $-\Omega_0$. Let's consider the component for $\Omega > 0$, originating from

$$\frac{1}{2}a(t - t_g)e^{j\Omega_0(t-t_p)} = \frac{1}{2}a(t - t_g)e^{-j\Omega_0 t_p}e^{j\Omega_0 t}$$

Due to the linearity and frequency shift properties of the Continuous-Time Fourier Transform (CTFT), the spectrum is:

$$\frac{1}{2}A\left(j(\Omega - \Omega_0)\right)e^{-j(\Omega-\Omega_0)t_g}e^{-j\Omega_0 t_p}$$

(because $e^{-j\Omega_0 t_p}$ is a constant, and the product $e^{j\Omega_0 t}K(t)$ has spectrum $K\left(j(\Omega - \Omega_0)\right)$). which is the expression of $Y_a(j\Omega)$ we has seen before.
Since $y_a(t) = a(t - t_g) \cdot \cos[\Omega_0(t - t_p)]$, we observe that the group delay (t_g) represents the delay of the baseband signal $a(t)$,while the phase delay (t_p) corresponds to the delay of the carrier $\cos[\Omega_0 t]$.

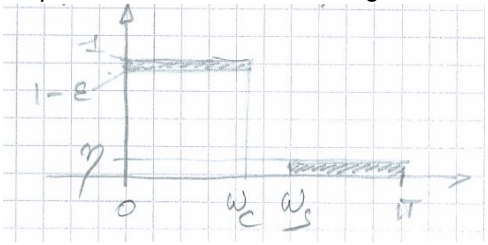
Let's redirect our focus to ideal filters. As discussed earlier, an ideal filter is characterized by a unit amplitude and linear phase in the passband, or, at the very least, it must exhibit a constant group delay in the passband to ensure that all signal components experience the same delay. Unfortunately, ideal filters are not realizable. To illustrate, consider the ideal lowpass filter with the frequency response:

$$H_{\mathrm{LP}}(e^{j\omega}) = \begin{cases} 1 & |\omega| \le \omega_c \\ 0 & \text{otherwise} \end{cases}$$

This filter has an impulse response:

$$h_{\mathrm{LP}}(n) = \frac{\sin(\omega_c n)}{\pi n} \qquad\qquad -\infty < n < +\infty$$

It's essential to note that this filter is not causal and is, in fact, an unstable system because $h_{\mathrm{LP}}(n)$ is not absolutely summable.
To achieve stable and realizable filters, we relax the stringent conditions imposed by ideal filters. One key modification involves introducing a transition band between the passband and the stopband. This enables the magnitude response to gradually decay from its maximum value to zero. Additionally, the magnitude response is permitted to vary within specified bounds in both the passband and the stopband.
For instance, when designing a lowpass filter, a frequency mask that defines bounds for the frequency response similar to the following is often considered:

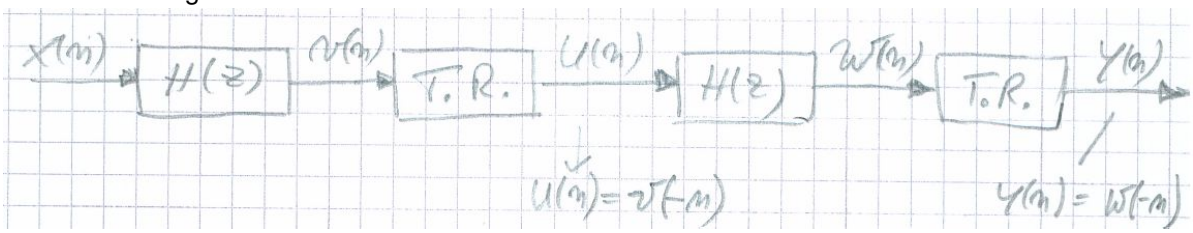


In practice, the following constraints are taken into account for $|H(e^{j\omega})|$:

$$\begin{cases} 1-\epsilon < |H(e^{j\omega})| < 1 & 0 \leq \omega \leq \omega_c \\ |H(e^{j\omega})| < \eta & \omega_s \leq \omega \leq \pi \end{cases}$$

07.03 Zero-phase filters

In many applications, it is crucial to design digital filters in a manner that introduces no phase distortion to the input signal components in the passband. One effective approach to avoiding phase distortions is the implementation of a *zero-phase filter*, characterized by a real positive frequency response. If we do not work in real-time and we process real sequences of finite duration, the zero-phase filtering can be easily implemented if we drop the hypothesis of system causality.
In this block diagram:



the input signal is processed with a filter $H(z)$ having real coefficients; the output of this filter is time-reversed and it is again filtered with the same filter $H(z)$, whose output is folded again. Let us prove that this is a zero-phase system.

$$V(e^{j\omega}) = H(e^{j\omega}) \cdot X(e^{j\omega})$$
$$U(e^{j\omega}) = V^*(e^{j\omega})$$

(because $v(n)$ is real)

$$W(e^{j\omega}) = H(e^{j\omega}) \cdot U(e^{j\omega})$$
$$Y(e^{j\omega}) = W^*(e^{j\omega}) = H^*(e^{j\omega}) \cdot U^*(e^{j\omega}) = H^*(e^{j\omega}) \cdot V(e^{j\omega}) =$$
$$= H^*(e^{j\omega}) \cdot H(e^{j\omega}) \cdot X(e^{j\omega}) = |H(e^{j\omega})|^2 \cdot X(e^{j\omega})$$

The system introduced above implements a filter with a frequency response $|H(e^{j\omega})|^2$, which is positive real, ensuring a zero-phase characteristic.
To design a zero-phase filter with a given magnitude response $A(e^{j\omega})$, one can design a filter with the magnitude response $\sqrt{A(e^{j\omega})}$, without imposing constraints on the phase. Subsequently, the technique described earlier can be applied. However, a notable drawback of this approach is that real-time signal processing becomes impossible. The entire sequence must be recorded before the technique can be applied.
For real-time processing systems, meeting our specifications while ensuring system causality often involves accepting a certain delay introduced by the filter and considering a linear phase response. We will explore that it is always possible to design linear phase FIR filters, whereas achieving linear phase IIR filters is impossible.

07.04 Linear phase FIR filters

In the upcoming discussion, we will demonstrate that a causal FIR filter with real coefficients, having a length of $N+1$, and a transfer function given by

$$H(z) = \sum_{n=0}^{N} h(n) z^{-n} = h(0) + h(1)z^{-1} + \ldots + h(N)z^{-N}$$

(note that now N is the exponent of the polynomial in z^{-1}), exhibits linear phase when the impulse response $h(n)$ is symmetric,

$$h(n) = h(N-n) \qquad \text{for } 0 \leq n \leq N,$$

or is antisymmetric,

$$h(n) = -h(N-n) \qquad \text{for } 0 \leq n \leq N.$$

Considering that the length can be either even or odd, we can categorize linear FIR filters with linear phase into four classes:

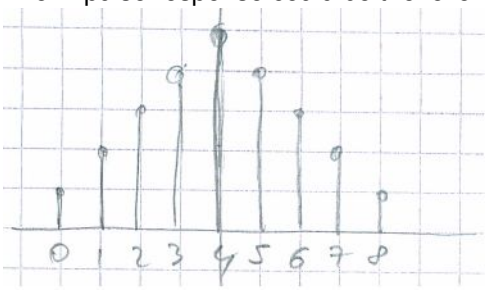
- TYPE 1: $h(n)$ is symmetric and has odd length,
- TYPE 2: $h(n)$ is symmetric and has even length,
- TYPE 3: $h(n)$ is antisymmetric and has odd length,
- TYPE 4: $h(n)$ is antisymmetric and has even length.

Let us analyze one by one these four classes.

TYPE 1: $h(n)$ is symmetric and has odd length. Thus, N is even.
Let us consider for example $N = 8$.

$$H(z) = h(0) + h(1)z^{-1} + h(2)z^{-2} + h(3)z^{-3} + \ldots + h(7)z^{-7} + h(8)z^{-8}$$

The impulse response could be the following,



Here, we have a symmetry axis for $\frac{N}{2} = 4$.
For the symmetry it is $h(0) = h(8)$, $h(1) = h(7)$, $h(2) = h(6)$, $h(3) = h(5)$, and

$$H(z) = h(0)(1 + z^{-8}) + h(1)(z^{-1} + z^{-7}) + h(2)(z^{-2} + z^{-6}) + h(3)(z^{-3} + z^{-5}) + h(4)z^{-4} \Big/ {}_{\cdot z^{4} \cdot z^{-4}}$$

$$H(z) = z^{-4} \cdot \left[h(0)(z^4 + z^{-4}) + h(1)(z^3 + z^{-3}) + h(2)(z^2 + z^{-2}) + h(3)(z^1 + z^{-1}) + h(4)\right]$$

The frequency response is given by

$$H(e^{j\omega}) = e^{-j\omega 4} \cdot [2h(0)\cos(4\omega) + 2h(1)\cos(3\omega) + 2h(2)\cos(2\omega) + 2h(3)\cos(\omega) + h(4)]$$

where we have utilized the identity $e^{j\omega n} + e^{-j\omega n} = 2\cos(\omega n)$.
Note that the expression within the square brackets is real and can take both positive and negative values. Consequently, the phase of the frequency response is linear, given by

$$\theta(\omega) = -4\omega + \beta = -\frac{N}{2}\omega + \beta \qquad \text{with } \beta = 0 \text{ or } \pi$$

and the group delay is constant:

$$t_g = -\frac{\mathrm{d}\theta}{\mathrm{d}\omega} = 4 = \frac{N}{2}.$$

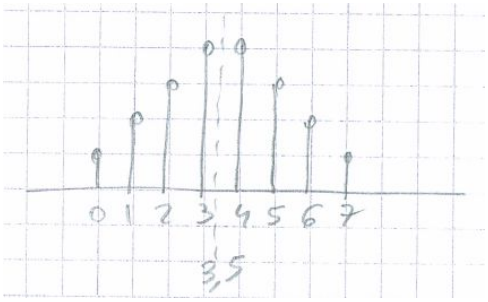
In general, for FIR filters of Type 1, the frequency response is given by

$$H(e^{j\omega}) = e^{-j\frac{N}{2}\omega} \cdot \overline{H}(\omega),$$

$$\overline{H}(\omega) = h(\frac{N}{2}) + 2\sum_{n=1}^{N/2} h(\frac{N}{2} - n)\cos(\omega n)$$

(which is the amplitude response apart from a ± 1 factor).

TYPE 2: $h(n)$ is symmetric and has even length. Thus, N is odd.
Let us consider the case where $N = 7$; here, we have a symmetry axis at $N/2 = 3.5$:



Let us proceed similarly to the previous case. For symmetry, we express $H(z)$ as:

$$H(z) = h(0)(1 + z^{-7}) + h(1)(z^{-1} + z^{-6}) + h(2)(z^{-2} + z^{-5}) + h(3)(z^{-3} + z^{-4}) \Big/_{\cdot z^{7/2} \cdot z^{-7/2}}$$

$$= z^{-7/2}\left[h(0)(z^{7/2} + z^{-7/2}) + h(1)(z^{5/2} + z^{-5/2}) + h(2)(z^{3/2} + z^{-3/2}) + h(3)(z^{1/2} + z^{-1/2})\right],$$

$$H(e^{j\omega}) = e^{-j\frac{7}{2}\omega}\left[2h(0)\cos(\frac{7}{2}\omega) + 2h(1)\cos(\frac{5}{2}\omega) + 2h(2)\cos(\frac{3}{2}\omega) + 2h(3)\cos(\frac{1}{2}\omega)\right].$$

Once again, the term within the square brackets is real, taking either positive or negative values. Consequently, the phase is given by

$$\theta(\omega) = -\frac{7}{2}\omega + \beta = -\frac{N}{2}\omega + \beta \qquad \text{with } \beta = 0 \text{ or } \pi$$

$$t_g = -\frac{\mathrm{d}\theta(\omega)}{\mathrm{d}\omega} = \frac{N}{2}$$

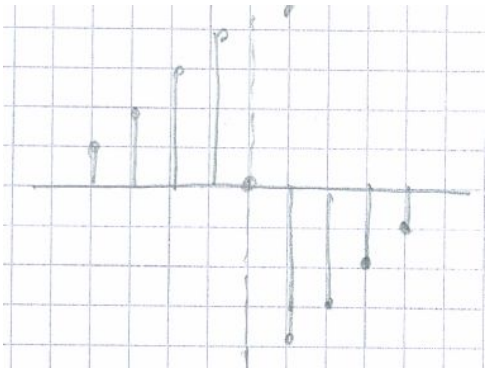
In general, it is

$$H(e^{j\omega}) = e^{-j\omega\frac{N}{2}} \cdot \overline{H}(\omega)$$

$$\overline{H}(\omega) = 2\sum_{n=1}^{(N+1)/2} h(\frac{N+1}{2} - n)\cos[\omega(n-\frac{1}{2})]$$

TYPE 3: $h(n)$ is antisymmetric and has odd length. Thus, N is even.

Here, we have an antisymmetry axis for $N/2$:



Given $h(N-n) = -h(n)$, for $n = \frac{N}{2}$, it follows that $h(\frac{N}{2}) = -h(\frac{N}{2}) = 0$.

Let's consider $N = 8$ and proceed similarly to the previous cases:

$$H(z) = h(0)(1-z^{-8}) + h(1)(z^{-1}-z^{-7}) + h(2)(z^{-2}-z^{-6}) + h(3)(z^{-3}-z^{-5}) \Big/_{\cdot z^4 \cdot z^{-4}}$$

$$= z^{-4}\left[h(0)(z^4-z^{-4}) + h(1)(z^3-z^{-3}) + h(2)(z^2-z^{-2}) + h(3)(z^1-z^{-1})\right]$$

Since $e^{j\omega m} - e^{-j\omega m} = 2j\sin(\omega m) = 2e^{j\pi/2}\sin(\omega m)$,

$$H(e^{j\omega}) = 2e^{-j4\omega}e^{j\pi/2}\left[h(0)\sin(4\omega) + h(1)\sin(3\omega) + h(2)\sin(2\omega) + h(3)\sin(\omega)\right]$$

$$\theta(\omega) = -4\omega + \frac{\pi}{2} + \beta = -\frac{N}{2}\omega + \frac{\pi}{2} + \beta \qquad \text{with } \beta = 0 \text{ or } \pi$$

$$t_g = -\frac{\mathrm{d}\theta(\omega)}{\mathrm{d}\omega} = \frac{N}{2}$$

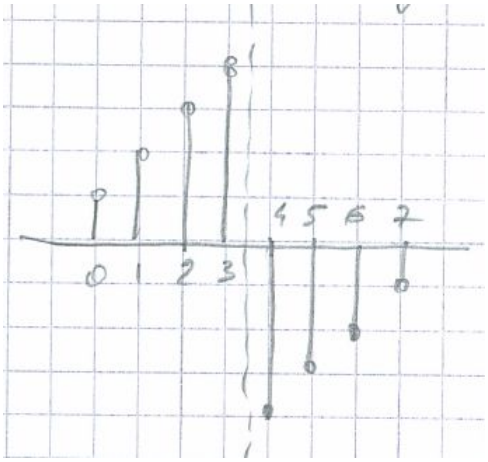
In general, it is

$$H(e^{j\omega}) = e^{-j\frac{N}{2}\omega}e^{j\pi/2} \cdot \overline{H}(\omega)$$

$$\overline{H}(\omega) = 2\sum_{n=1}^{N/2} h(\frac{N}{2} - n)\sin[\omega n]$$

TYPE 4: $h(n)$ is antisymmetric and has even length. Thus, N is odd.

We have an antisymmetry axis for $N/2$:



Let's proceed similarly to the previous cases, considering $N = 7$.

$$H(z) = h(0)(1 - z^{-7}) + h(1)(z^{-1} - z^{-6}) + h(2)(z^{-2} - z^{-5}) + h(3)(z^{-3} - z^{-4}) \Big/_{\cdot z^{7/2} \cdot z^{-7/2}}$$

$$= z^{-7/2}\left[h(0)(z^{7/2} - z^{-7/2}) + h(1)(z^{5/2} - z^{-5/2}) + h(2)(z^{3/2} - z^{-3/2}) + h(3)(z^{1/2} - z^{-1/2})\right]$$

$$H(e^{j\omega}) = e^{-j\frac{7}{2}\omega} e^{j\pi/2} 2\left[h(0)\sin(\frac{7}{2}\omega) + h(1)\sin(\frac{5}{2}\omega) + h(2)\sin(\frac{3}{2}\omega) + h(3)\sin(\frac{1}{2}\omega)\right]$$

Thus, the phase is

$$\theta(\omega) = -\frac{7}{2}\omega + \frac{\pi}{2} + \beta = -\frac{N}{2}\omega + \frac{\pi}{2} + \beta \qquad \text{with } \beta = 0 \text{ or } \pi$$

$$t_g = -\frac{\mathrm{d}\theta(\omega)}{\mathrm{d}\omega} = \frac{7}{2} = \frac{N}{2}$$

In general, it is

$$H(e^{j\omega}) = e^{-j\omega\frac{N}{2}} \cdot e^{j\pi/2} \cdot \overline{H}(\omega)$$

$$\overline{H}(\omega) = 2\sum_{n=1}^{(N+1)/2} h(\frac{N+1}{2} - n)\sin[\omega(n - \frac{1}{2})]$$

Note that $\overline{H}(\omega)$ can take on negative values for certain ω. It represents the amplitude response, with the inclusion of a multiplicative term ± 1. Negative values of $\overline{H}(\omega)$ are commonly observed especially in the stopband.

Zeros' position in linear FIR filters

For the symmetric filters:

$$H(z) = \sum_{n=0}^{N} h(n)z^{-n} = \sum_{n=0}^{N} h(N-n)z^{-n}$$

By introducing the variable change $m = N - n$ in the second equality, we get:

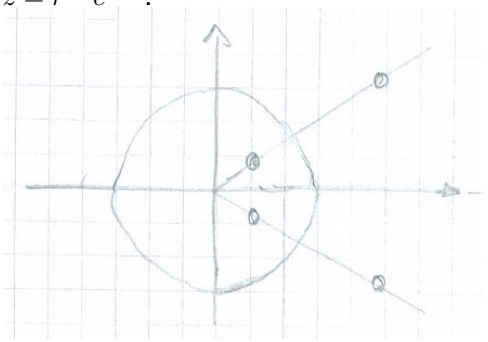
$$H(z) = \sum_{m=0}^{N} h(m)z^{-N+m} = z^{-N}H(z^{-1})$$

Similarly, for the antisymmetric filters, we have

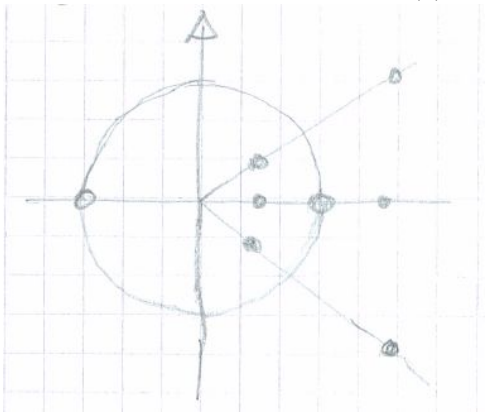
$$H(z) = -z^{-N}H(z^{-1}).$$

A polynomial with constant coefficients that satisfies the condition $H(z) = z^{-N}H(z^{-1})$ is referred to as a *mirror image* polynomial. Conversely, a polynomial with constant coefficients satisfying the condition $H(z) = -z^{-N}H(z^{-1})$ is termed an *antimirror image* polynomial.

For these two properties, if $z = \xi_0$ is a zero of $H(z)$ ($H(\xi_0) = 0$) then $z = \xi_0^{-1}$ is also a zero of $H(z)$. In other words, symmetric and antisymmetric FIR filters exhibit zeros with reciprocal symmetry, known as *mirror image symmetry* with respect to the unit circle. Additionally, if the filter has real coefficients, the zeros also possess conjugate symmetry. If a filter has a pair of conjugate symmetric zeros in $z = re^{\pm j\theta}$, then, due to the *mirror image symmetry* property of the zeros, it must also have a pair of zeros a $z = r^{-1}e^{\pm j\theta}$:



For a zero on the unit circle $z = e^{j\theta}$, its reciprocal coincides with the conjugate. Consequently, the filter can have pairs of zeros on the unit circle in $e^{\pm j\theta}$. For every real zero $z = r$, the filter must have also the reciprocal zero in $z = r^{-1}$. Additionally, zeros in $z = \pm 1$ are reciprocal of themselves and may appear individually in the set of zeros of $H(z)$.



Note that an FIR filter of Type 2 ($h(n)$ symmetric and of even length, N odd) must have at least a zero at -1. This is evident from the condition:

$$H(z) = z^{-N} H(z^{-1})$$

$$H(-1) = (-1)^{-N} H(-1) = -H(-1) = 0$$

Similarly, in Type 3 and 4 FIR filters, there must be at least one zero at $z = +1$ due to the antisymmetry condition:

$$H(z) = -z^{-N} H(z^{-1})$$

$$H(1) = -1^{N} H(1) = -H(1) = 0$$

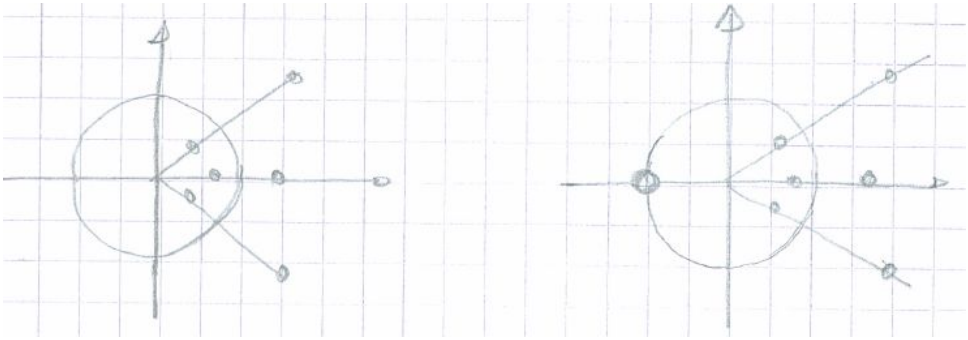
Furthermore, Type 3 FIR filters, which have an odd length and N even, must also have at least one zero at -1.

$$H(-1) = -(-1)^{N} H(-1) = -H(-1) = 0$$

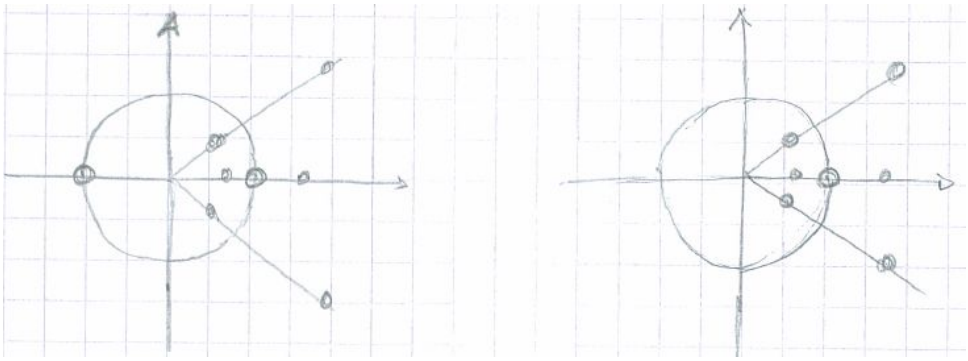
Note that all linear FIR filters with an odd length must have either no zero or an even number of zeros at $+1$ and -1 (because N is even), while all linear FIR filters with an even length must have an odd number of zeros at $+1$ and -1 (since N is odd).
The four cases of linear FIR filters differ in the distribution of zeros at $+1$ and -1. Specifically:

- Type 1 filters: have no zero or an even number of zeros at $+1$ and -1 (N is even).
- Type 2 filters: have no zero or an even number of zeros at $+1$ and an odd number of zeros at -1 (N is odd).
- Type 3 filters: have an odd number of zeros at $+1$ and an odd number of zeros at -1 (N is even).
- Type 4 filters: have an odd number of zeros at $+1$ and no zero or an even number of zeros at -1 (N is odd).

Filters of Type 3 and 4 must have an odd number of zeros at $+1$.This is because the factor associated with a zero at $+1$ is $(1 - z^{-1})$, which imparts antisymmetry to the polynomial. Similarly, Filters of Type 2 and 3 must have an odd number of zeros at -1 (associated with the factor $(1 + z^{-1})$) to achieve the desired value of N – whether odd or even – ensuring the symmetry or antisymmetry of the polynomial.
The zeros in the four cases are the following:



Type 1 Type 2



Type 3 Type 4

Type 1 filters can be used to implement any kind of filters, including lowpass, highpass, passband, and stopband filters. Type 2 filters have $H(e^{j\pi}) = 0$, making them suitable for implementing lowpass and passband filters but not highpass or stopband filters. Type 3 filters with $H(e^{j\pi}) = H(e^{j0}) = 0$ are unsuitable for lowpass, highpass, and stopband filters, but they can be used for passband filters. Type 4 filters with $H(e^{j0}) = 0$ cannot be used for implementing lowpass or stopband filters, but they are suitable for highpass or passband filters.

IIR filters and linear phase

Let us consider the case of a causal IIR filter described by a finite difference equation:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_M z^{-M}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-N}} = \frac{B(z)}{A(z)}$$

We have observed that in the case of FIR filters, the linear phase condition manifests as the mirror image symmetry of the zeros. The same criterion could be applied to derive IIR filters with linear phase. If $A(z)$ and $B(z)$ are mirror image polynomials, then $H(z)$ exhibits linear phase. Unfortunately, the zeros of $A(z)$ are the poles of the system and, if the poles satisfy the mirror image symmetry property, the system is unstable. This is because for every pole inside the unit circle, there must be a pole outside the unit circle. IIR filter design cannot overlook the need to ensure filter stability. Therefore, we shall content ourselves only with approximating linear phase in the filter passband.

07.05 Geometric interpretation of frequency response computation

In the following, we will study various examples of FIR and IIR basic filters. For these filters, using the geometric interpretation of frequency response computation, it is very easy to plot amplitude response or phase response diagrams based on the knowledge of pole and zero locations. Let us consider an IIR filter with a transfer function

$$H(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_M z^{-M}}{1 + a_1 z^{-1} + \ldots + a_N z^{-N}}$$

where for simplicity we have set $a_0 = 1$.

$$H(z) = b_0 \cdot \frac{(1 - z_1 z^{-1}) \cdot (1 - z_2 z^{-1}) \cdot \ldots \cdot (1 - z_M z^{-1})}{(1 - p_1 z^{-1}) \cdot (1 - p_2 z^{-1}) \cdot \ldots \cdot (1 - p_N z^{-1})} =$$

$$= b_0 \cdot z^{N-M} \cdot \frac{(z - z_1) \cdot (z - z_2) \cdot \ldots \cdot (z - z_M)}{(z - p_1) \cdot (z - p_2) \cdot \ldots \cdot (z - p_N)}$$

where $z_1, \ldots, z_M$ are the system zeros and $p_1, \ldots, p_N$ are the system poles.
The frequency response of the system is

$$H(e^{j\omega}) = b_0 \cdot e^{j\omega(N-M)} \frac{(e^{j\omega} - z_1) \cdot (e^{j\omega} - z_2) \cdot \ldots \cdot (e^{j\omega} - z_M)}{(e^{j\omega} - p_1) \cdot (e^{j\omega} - p_2) \cdot \ldots \cdot (e^{j\omega} - p_N)}.$$

The amplitude response and the phase response are given, respectively, by

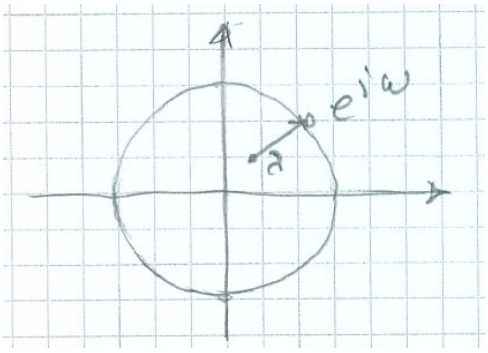
$$|H(e^{j\omega})| = |b_0| \cdot \frac{|e^{j\omega} - z_1| \cdot |e^{j\omega} - z_2| \cdot \ldots \cdot |e^{j\omega} - z_M|}{|e^{j\omega} - p_1| \cdot |e^{j\omega} - p_2| \cdot \ldots \cdot |e^{j\omega} - p_N|},$$

$$\arg H(e^{j\omega}) = \arg b_0 + \omega(N - M) + \arg(e^{j\omega} - z_1) + \arg(e^{j\omega} - z_2) + \ldots + \arg(e^{j\omega} - z_M)$$
$$- \arg(e^{j\omega} - p_1) - \arg(e^{j\omega} - p_2) - \ldots - \arg(e^{j\omega} - p_N).$$

If we examine the frequency response we can notice that the typical factor is

$$(e^{j\omega} - \lambda)$$

with $\lambda = z_i$ or $\lambda = p_i$. If we interpret this factor on the complex plane we have that:
$e^{j\omega}$ is a point on the unit circle,
λ is the zero or pole position,
$e^{j\omega} - \lambda$ is the vector from λ to $e^{j\omega}$.



For ω that goes from 0 to 2π this vector varies in amplitude and phase. From the position of the two points, we can immediately obtain its amplitude and phase behavior. $|e^{j\omega} - \lambda|$ has minimum value when $\omega = \arg\lambda$, and has maximum value when $\omega = \arg\lambda + \pi$.
The amplitude response $|H(e^{j\omega})|$ is given by the product of the modulus of all vectors associated with the zeros, divided by the modulus of all vectors related to the poles, multiplied by the modulus of b_0.
The phase response $\arg H(e^{j\omega})$ is given by the phase of b_0, plus $\omega(N-M)$, plus the phase of all vectors associated with the zeros, minus the phase of all vectors associated with the poles.
When designing a filter that should attenuate a certain frequency range, we shall locate the zeros close to the unit circle around this frequency range. On the contrary, if we have to emphasize certain frequency components of the signal, we shall locate the poles around this frequency range (indeed, the amplitude response maxima correspond to the vectors $z - p_i$ minima).

07.06 Simple digital filters

Lowpass FIR filter

The most simple lowpass filter is the filter that computes the average between samples. Let us consider

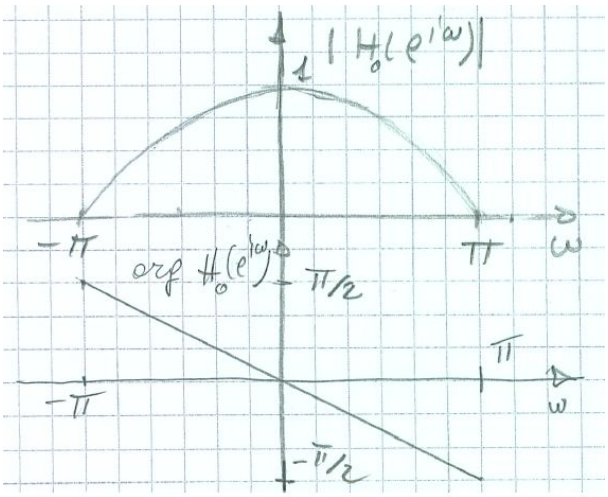
$$y(n) = \frac{1}{2}(x(n) + x(n-1))$$

$$Y(z) = \frac{1}{2}\left(X(z) + z^{-1}X(z)\right)$$

$$H_0(z) = \frac{1}{2}(1 + z^{-1}) = \frac{1}{2}\frac{z+1}{z}$$

This is an FIR filter with a zero at $z = -1$ and a pole at $z = 0$. The vector $e^{j\omega} - \lambda$ related to the pole has always unit amplitude. The vector related to the zero has maximum amplitude for $\omega = 0$ (with amplitude 2) and then its amplitude decreases to 0 as ω goes from 0 to π. The filter is symmetric, and thus its phase is linear.

$$H_0(e^{j\omega}) = \frac{1}{2}(1 + e^{-j\omega}) = \frac{1}{2}e^{-j\omega/2}(e^{j\omega/2} + e^{-j\omega/2}) = e^{-j\omega/2} \cdot \cos(\frac{\omega}{2}).$$



Of particular interest is the frequency ω_c for which

$$|H_0(e^{j\omega_c})| = \frac{1}{\sqrt{2}}|H_0(e^{j\omega})|_{\mathsf{MAX}} = \frac{1}{\sqrt{2}}|H_0(e^{j0})|$$

Let us consider the gain in dB (i.e., the amplitude in dB):

$$G(\omega_c) = 20\log_{10}|H_0(e^{j\omega_c})| =$$

$$= 20\log_{10}|H_0(e^{j0})| - 20\log_{10}(\sqrt{2}) =$$

$$= 0 - 3.0103 \simeq -3\mathrm{dB}.$$

Thus, the frequency ω_c is called the $3\mathrm{dB}$ *cutoff frequency*, because the gain has reduced by $3\mathrm{dB}$ compared with the maximum value.

Imposing,

$$|H_0(e^{j\omega_c})|^2 = \cos^2 \frac{\omega_c}{2} = \frac{1}{2}$$

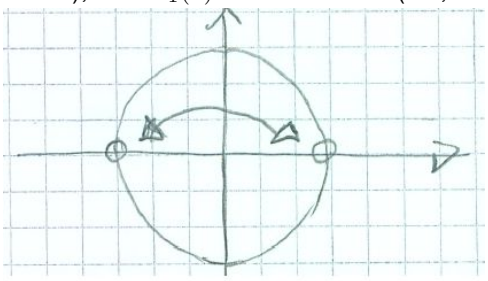
we obtain $\omega_c = \frac{\pi}{2}$.

Highpass FIR filter

The most simple highpass filter can be obtained by replacing z with $-z$ in the previous transfer function:

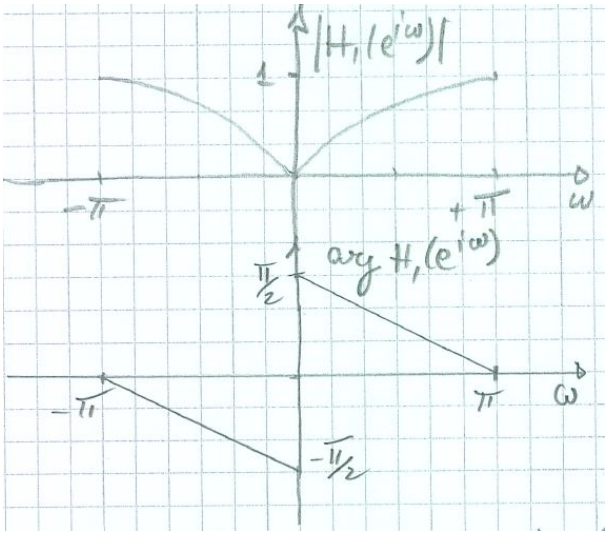
$$H_1(z) = H_0(-z).$$

If we consider the geometric interpretation of the frequency response, we can understand that with this variable change $H_1(z)$ has for $z = 1$ (i.e., for $\omega = 0$) the same behavior of $H_0(z)$ for $z = -1$ (i.e., for $\omega = \pi$), and $H_1(z)$ has for $z = -1$ (i.e., for $\omega = \pi$) the same behavior of $H_0(z)$ for $z = +1$ (i.e., for $\omega = 0$).



$$H_1(z) = \frac{1}{2}(1 - z^{-1})$$

$$H_1(e^{j\omega}) = je^{-j\omega/2} \sin(\frac{\omega}{2})$$



Also in this case the 3dB cutoff frequency falls at $\omega_c = \frac{\pi}{2}$.

We can obtain FIR lowpass or highpass filters with a narrower passband by cascading a certain number of these elementary filters. By considering the cascade of M lowpass filters $H_0(e^{j\omega})$, the resulting frequency response is $H(e^{j\omega}) = H_0^M(e^{j\omega})$ and the 3dB cutoff frequency is given for

$$|H(e^{j\omega_c})| = |H_0(e^{j\omega_c})|^M = \frac{1}{\sqrt{2}}$$

i.e., for

$$|H_0(e^{j\omega_c})| = 2^{-1/(2M)}$$
$$\cos\left(\frac{\omega_c}{2}\right) = 2^{-\frac{1}{2M}}$$
$$\omega_c = 2\arccos\left(2^{-\frac{1}{2M}}\right).$$

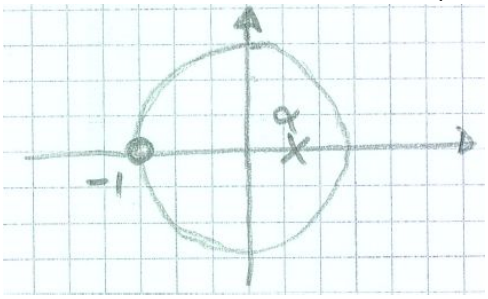
IIR Lowpass filter

A lowpass filter of the first order has a transfer function:

$$H_{\mathrm{LP}}(z) = \frac{1-\alpha}{2}\frac{1+z^{-1}}{1-\alpha z^{-1}},$$

where $|\alpha| < 1$ for the stability of the system.

The filter has a zero at $z = -1$ and a pole at $z = \alpha$.



In the geometric interpretation of the frequency response computation, we see that for ω going from 0 to π the vector $e^{j\omega} - (-1)$ decreases from 2 to 0. On the contrary, the vector $e^{j\omega} - \alpha$ for $\alpha > 0$ increases from $1-\alpha$ till $1+\alpha$. The maximum value and the minimum value of the frequency response are obtained for $\omega = 0$ and $\omega = \pi$, respectively.

$$H_{\mathrm{LP}}(e^{j0}) = 1 \qquad H_{\mathrm{LP}}(e^{j\pi}) = 0$$

$$|H_{\mathrm{LP}}(e^{j\omega})|^2 = \frac{(1-\alpha)^2}{4}\cdot\frac{(1+e^{j\omega})(1+e^{-j\omega})}{(1-\alpha e^{j\omega})(1-\alpha e^{-j\omega})} =$$
$$= \frac{(1-\alpha)^2}{4}\cdot\frac{1+e^{j\omega}+e^{-j\omega}+1}{1-\alpha e^{j\omega}-\alpha e^{-j\omega}+\alpha^2} =$$
$$= \frac{(1-\alpha)^2}{2}\cdot\frac{1+\cos(\omega)}{1-2\alpha\cos(\omega)+\alpha^2}$$

$$\frac{\mathrm{d}|H_{\mathrm{LP}}(e^{j\omega})|^2}{\mathrm{d}\omega} = \frac{(1-\alpha)^2}{2}\cdot\frac{-\sin(\omega)\left(1+\alpha^2-2\alpha\cos(\omega)\right)-(1+\cos(\omega))\left(2\alpha\sin(\omega)\right)}{\left(1-2\alpha\cos(\omega)+\alpha^2\right)^2} =$$
$$= \frac{(1-\alpha)^2}{2}\cdot\frac{-\sin(\omega)(1+\alpha)^2}{\left(1-2\alpha\cos(\omega)+\alpha^2\right)^2}.$$

For $0 \le \omega \le \pi$ the derivative is always negative and, thus, the amplitude response decreases monotonically.

The $3\mathrm{dB}$ cutoff frequency is obtained for $|H_{\mathrm{LP}}(e^{j\omega})|^2 = \frac{1}{2}$.

$$\frac{(1-\alpha)^2}{2}\cdot\frac{1+\cos(\omega_c)}{1-2\alpha\cos(\omega_c)+\alpha^2} = \frac{1}{2}$$

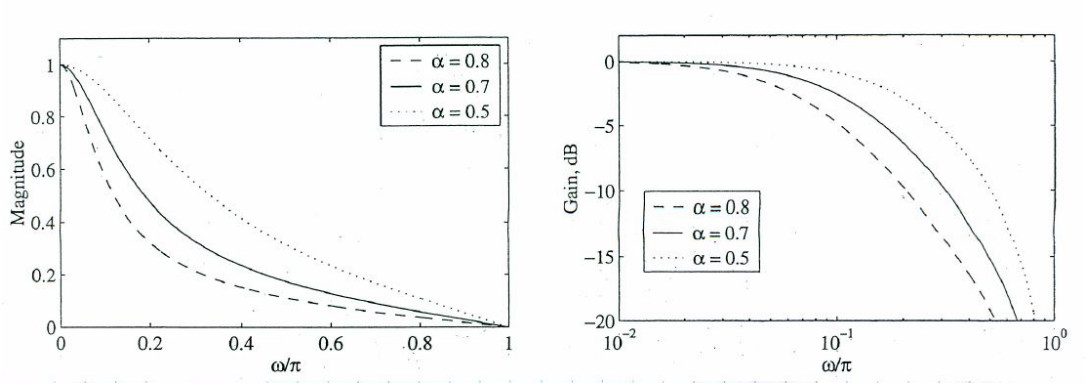
$$(1-\alpha)^2\,(1+\cos(\omega_c)) = 1+\alpha^2-2\alpha\cos(\omega_c)$$

$$(1-2\alpha+\alpha^2)\cos(\omega_c)+2\alpha\cos(\omega_c) = 1+\alpha^2-(1-\alpha)^2$$

$$\cos(\omega_c) = \frac{2\alpha}{1+\alpha^2}$$

If we want a lowpass filter with an assigned cutoff frequency ω_c, we have to solve the previous equation for α. It can be proved that the only stable solution is

$$\alpha = \frac{1-\sin(\omega_c)}{\cos(\omega_c)}$$



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

IIR Highpass filter

An order 1 IIR highpass filter is given by

$$H_{\mathrm{HP}}(z) = \frac{1-\alpha}{2}\,\frac{1-z^{-1}}{1+\alpha z^{-1}}$$

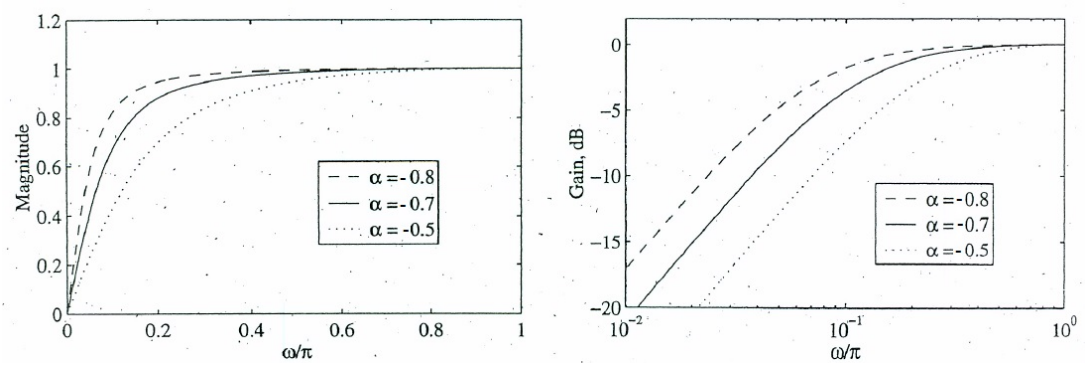
where it must be $|\alpha|<1$ for stability.

This filter has been obtained from the previous lowpass filter by replacing z with $-z$:

$$H_{\mathrm{HP}}(z) = H_{\mathrm{LP}}(-z)$$

Thus, the same properties of the previous filter hold, apart from a frequency shift of π in the frequency response:

$$H_{\mathrm{HP}}(e^{j\omega}) = H_{\mathrm{LP}}(e^{j(\omega+\pi)}).$$



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

IIR Bandpass filter

In order to obtain bandpass or bandstop filters, we must consider at least second-order filters.

A bandpass filter of the second order is given by the following transfer function:

$$H_{\mathrm{BP}}(z) = \frac{1-\alpha}{2}\frac{1-z^{-2}}{1-\beta(1+\alpha)z^{-1}+\alpha z^{-2}}$$

The squared amplitude response is

$$|H_{\mathrm{BP}}(e^{j\omega})|^2 = \frac{(1-\alpha)^2\,(1-\cos(2\omega))}{2\,[1+\beta(1+\alpha)^2\cos(\omega)+2\alpha\cos(2\omega)]}$$

which is 0 for $\omega = 0$ and $\omega = \pi$ and assumes the maximum value 1 for $\omega = \omega_0$, called *center frequency* for the bandpass filter, where

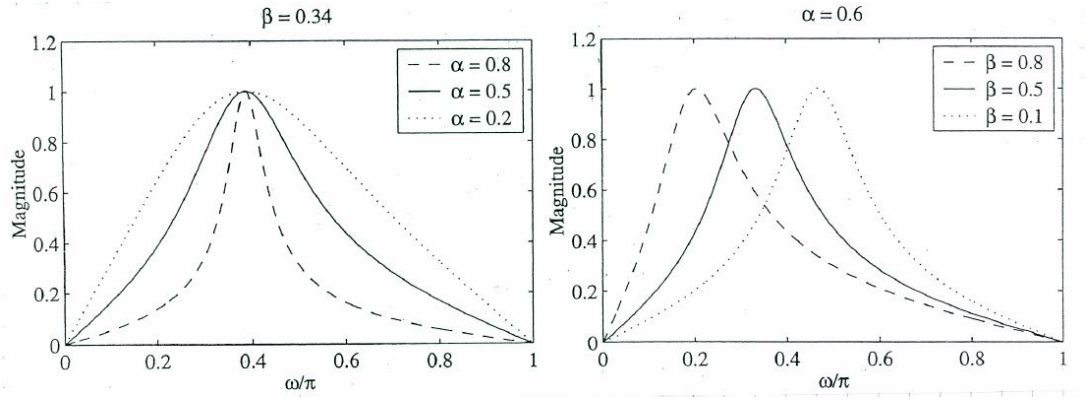
$$\cos(\omega_0) = \beta$$

$$\omega_0 = \arccos(\beta)$$

The frequencies ω_{c1} and ω_{c2} for which $|H_{\mathrm{BP}}(e^{j\omega})|^2 = \frac{1}{2}$ are called $3\mathrm{dB}$ cutoff frequencies and their difference $\omega_{c2} - \omega_{c1}$ is called $3\mathrm{dB}$ *bandwidth*. It can be proved that

$$B_{3\mathrm{dB}} = \omega_{c2} - \omega_{c1} = \arccos\left(\frac{2\alpha}{1+\alpha^2}\right).$$

Thus, β controls the center frequency, while α controls the bandwidth.



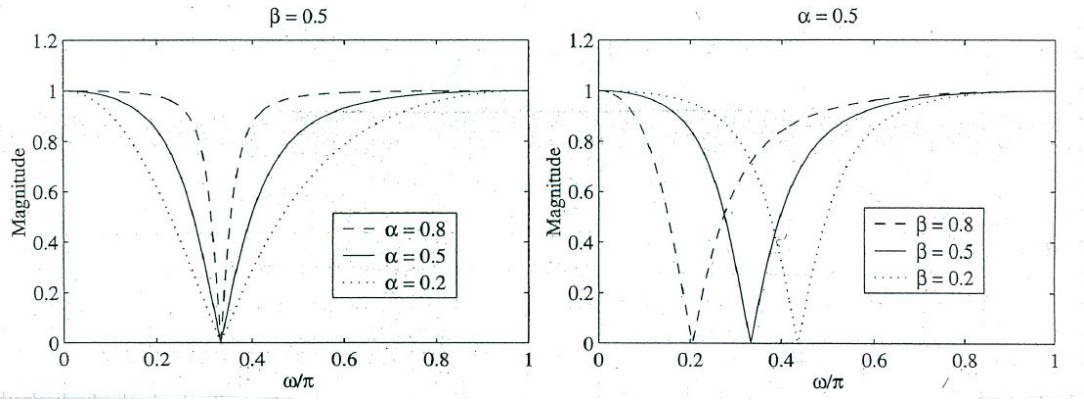
(From S. K. Mitra, “Digital signal processing: a computer based approach”, McGraw Hill, 2011)

IIR Bandstop filter

An second order IIR bandstop filter is given by the following transfer function

$$H_{\mathrm{BP}}(z) = \frac{1+\alpha}{2}\frac{1-2\beta z^{-1}+z^{-2}}{1-\beta(1+\alpha)z^{-1}+\alpha z^{-2}}$$

The amplitude response for different values of α and β is given by:



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

This filter is also called a *notch* filter.
Also in this case α and β separately control the stopband bandwidth and the notch position. The notch frequency is

$$\omega_0 = \arccos(\beta)$$

and the stopband bandwidth is

$$B_{I3\mathrm{dB}} = \arccos\left(\frac{2\alpha}{1+\alpha^2}\right).$$

Higher order IIR filters
By cascading a certain number of filters like the ones we have just introduced, it is possible to obtain filters with steeper rising and falling edges in the frequency domain, i.e., filters with smaller transition bands.
For example, consider the cascade of K IIR lowpass filters:

$$H_{\mathrm{LP}}(z) = \frac{1-\alpha}{2}\frac{1+z^{-1}}{1-\alpha z^{-1}},$$

for which we have seen $\cos(\omega_c) = \dfrac{2\alpha}{1+\alpha^2}$.
In the filter cascade, the resulting transfer function is

$$G_{\mathrm{LP}}(z) = \left(\frac{1-\alpha}{2}\frac{1+z^{-1}}{1-\alpha z^{-1}}\right)^K$$

$$|G_{\mathrm{LP}}(e^{j\omega})|^2 = \left[\frac{(1-\alpha)^2(1+\cos(\omega))}{2(1+\alpha^2-2\alpha\cos(\omega))}\right]^K$$

The $3\mathrm{dB}$ cutoff frequency can be obtained by setting

$$|G_{\mathrm{LP}}(e^{j\omega})|^2 = \frac{1}{2}.$$

By imposing a certain ω_c and solving this equation for α, we obtain that the only stable solution is

$$\alpha = \frac{1+(1-C)\cos(\omega_c) - \sin(\omega_c)\sqrt{2C-C^2}}{1-C+\cos(\omega_c)}$$

with $C = 2^{\frac{K-1}{K}}$.

Example: Let us design a filter with 3dB cutoff frequency $\omega_c = 0.4\pi$.

For $K = 1$, $C = 1$, it is $\alpha = 0.1584$.

For $K = 4$, $C = 1.6818$, it is $\alpha = -0.251$.

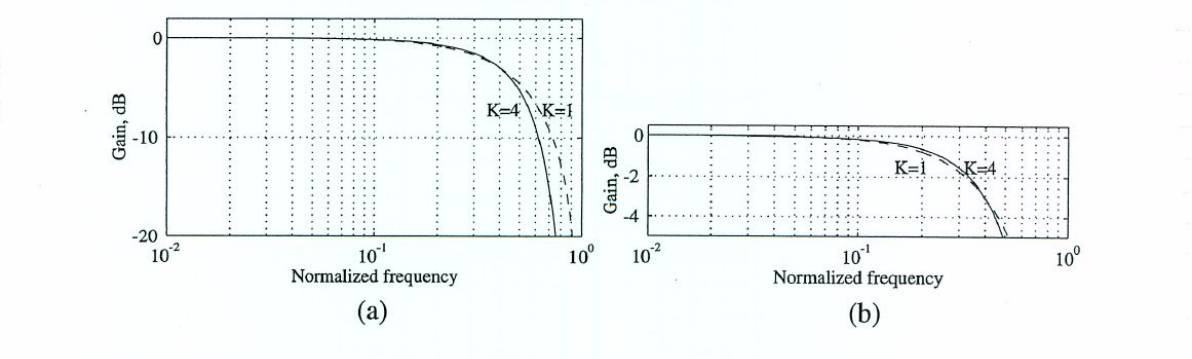


Figure 4.23: (a) Gain responses of a single first-order lowpass filter ($K = 1$) and a cascade of four identical first-order lowpass filters ($K = 4$) with a 3-dB cutoff frequency of $\omega_c = 0.4\pi$. (b) Passband details.

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

07.07 Comb filters

The filters we have seen up to this point have been characterized by a single passband and/or stopband. However, there are several applications where filters with multiple passbands or stopbands are required. Comb filters are an example of such filters.

Comb filters have a periodic frequency response with a period of $\frac{2\pi}{L}$, where L is a positive integer. If $H(z)$ is a transfer function with a single passband and/or stopband, a comb filter can be easily generated by replacing each delay element with L delays, resulting in a transfer function $G(z) = H(z^L)$.

If $|H(e^{j\omega})|$ has a peak at $\omega = \omega_p$, then $|G(e^{j\omega})|$ has L peaks at $\omega = \frac{\omega_p}{L} + \frac{2\pi}{L}k$ where $0 \leq k \leq L-1$, and $0 \leq \omega \leq 2\pi$.

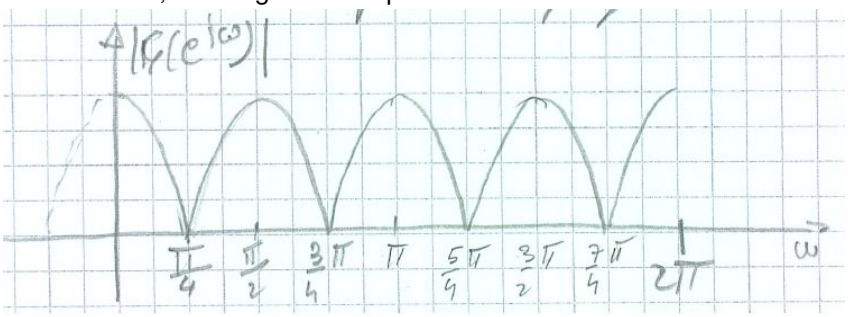
Similarly, if $|H(e^{j\omega})|$ has a notch at $\omega = \omega_0$, then $|G(e^{j\omega})|$ has L notches at $\omega = \frac{\omega_0}{L} + \frac{2\pi}{L}k$ where $0 \leq k \leq L-1$, and $0 \leq \omega \leq 2\pi$.

For example, if we consider

$$H(z) = \frac{1}{2}\left(1 + z^{-1}\right)$$

$$G(z) = \frac{1}{2}\left(1 + z^{-L}\right)$$

and for $L = 4$, the magnitude response is

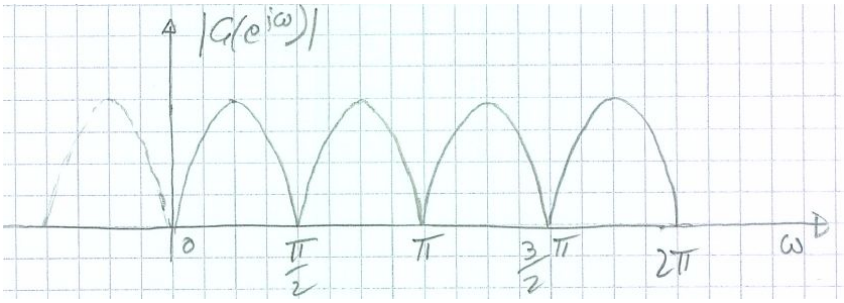


Similarly, if we consider

$$H(z) = \frac{1}{2}\left(1 - z^{-1}\right)$$

$$G(z) = \frac{1}{2}\left(1 - z^{-L}\right)$$

the magnitude response for $L = 4$ is



It is easy to understand the behavior of the frequency response $G(e^{j\omega})$ from the frequency response $H(e^{j\omega})$ because

$$G(e^{j\omega}) = H(e^{j\omega L}).$$

By varying ω from 0 to 2π, $e^{j\omega}$ moves along the unit circle L times, and therefore, the frequency response $G(e^{j\omega})$ coincides with $H(e^{j\omega})$ (which is periodic with period 2π), apart from a frequency axis scaling by a factor of $\frac{1}{L}$.

07.08 All-pass filters

By definition, a transfer function is called *all-pass* if the amplitude response is constant (i.e., one) for all frequencies, that is, if

$$|A(e^{j\omega})| = 1 \quad \forall\omega$$

An all-pass causal transfer function with real coefficients is given by

$$A_M(z) = \frac{d_M + d_{M-1}z^{-1} + \ldots + d_1 z^{-M+1} + d_0 z^{-M}}{d_0 + d_1 z^{-1} + \ldots + d_{M-1}z^{-M+1} + d_M z^{-M}}$$

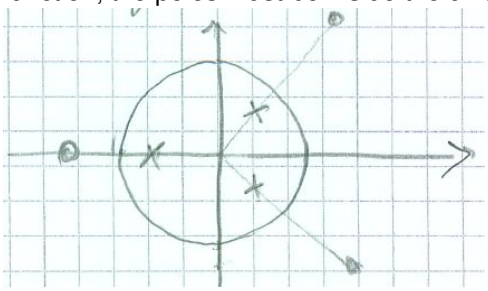
If we call the denominator polynomial $D_M(z)$,

$$D_M(z) = d_0 + d_1 z^{-1} + \ldots + d_{M-1}z^{-M+1} + d_M z^{-M}$$

we have

$$A_M(z) = z^{-M}\frac{D_M(z^{-1})}{D_M(z)}.$$

Note that the denominator polynomial is the *mirror image* polynomial of the numerator, and vice versa. If $z = re^{j\theta}$ is a pole of the transfer function, then $z = \frac{1}{r}e^{-j\theta}$ is a zero. Thus, poles and zeros of an all-pass filter exhibit mirror-image symmetry in the z-plane. By assuming that $A(z)$ is a stable transfer function, the poles must be inside the unit circle and the zeros outside the unit circle.



Let us prove that $A_M(z) = z^{-M}\frac{D_M(z^{-1})}{D_M(z)}$ is an all-pass transfer function. Consider

$$A_M(z^{-1}) = z^{M}\frac{D_M(z)}{D_M(z^{-1})}$$

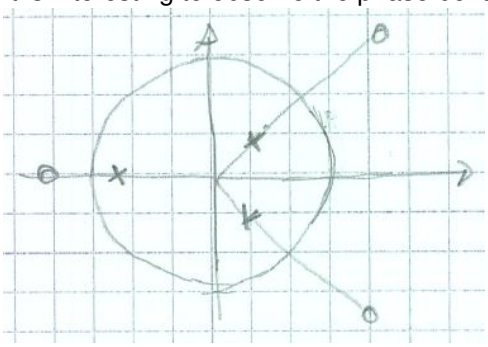
$$A_M(z)\cdot A_M(z^{-1}) = z^{-M}\frac{D_M(z^{-1})}{D_M(z)}\cdot z^{M}\frac{D_M(z)}{D_M(z^{-1})} = 1$$

Then,

$$A_M(e^{j\omega}) \cdot A_M(e^{-j\omega}) = |A_M(e^{j\omega})|^2 = 1$$

Q.E.D.

It is interesting to observe the phase behavior for $0 \leq \omega \leq 2\pi$.



With the geometric interpretation of the phase response, as ω varies from 0 to 2π, the zeros cause phase fluctuations, but the overall phase variation is 0. Conversely, each pole contributes a phase of -2π. Overall, the phase varies from 0 to $-2\pi M$, as ω varies between 0 and 2π. In other words, for ω varying between 0 and π, the phase varies from 0 to $-\pi M$.

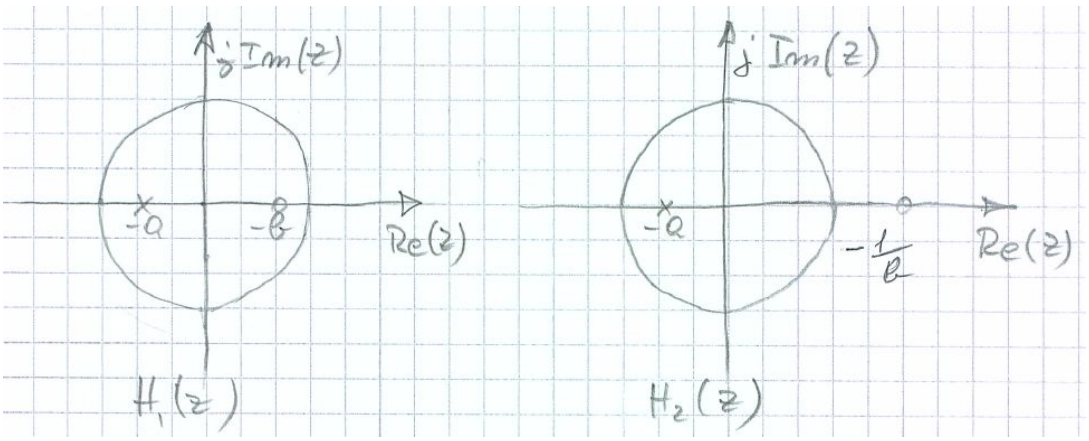
07.09 Minimum-phase and maximum-phase transfer functions

Another useful classification of transfer functions is based on the behavior of the phase response. Let us consider the following two first-order transfer functions with real coefficients:

$$H_1(z) = \frac{z+b}{z+a}$$

$$H_2(z) = \frac{bz+1}{z+a}$$

with $|a| < 1$ and $|b| < 1$.



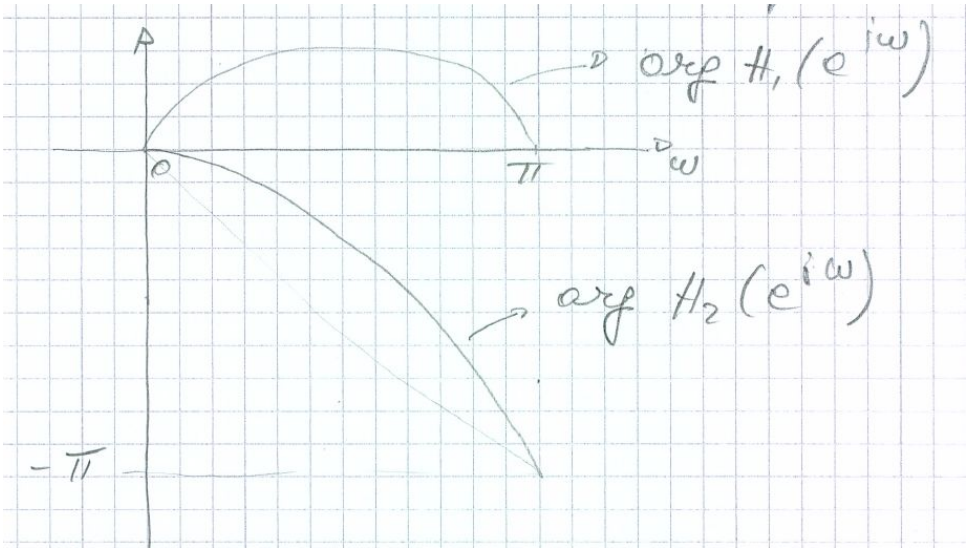
As we can see, both transfer functions have a pole inside the unit circle at $-a$, indicating stability. On the other hand, the zero of $H_1(z)$ falls inside the unit circle (at $z = -b$), while the zero of $H_2(z)$ falls outside the unit circle at $z = -\frac{1}{b}$.
The two transfer functions have the same amplitude response because

$$H_2(z) = H_1(z) \cdot \frac{bz+1}{z+b} = H_1(z) \cdot A(z)$$

with $A(z)$ all-pass. Thus,

$$H_1(e^{j\omega}) \cdot H_1(e^{-j\omega}) = \left|H_1(e^{j\omega})\right|^2 = H_2(e^{j\omega}) \cdot H_2(e^{-j\omega}) = \left|H_2(e^{j\omega})\right|^2$$

but they have different phase responses:



$$\arg\left[H_1(e^{j\omega})\right] = \theta_1(\omega) = \arctan\frac{\sin(\omega)}{b+\cos(\omega)} - \arctan\frac{\sin(\omega)}{a+\cos(\omega)}$$

$$\arg\left[H_2(e^{j\omega})\right] = \theta_2(\omega) = \arctan\frac{b\sin(\omega)}{1+b\cos(\omega)} - \arctan\frac{\sin(\omega)}{a+\cos(\omega)}$$

$$\arg\left[H_2(e^{j\omega})\right] = \arg\left[H_1(e^{j\omega})\right] + \arg\left[A(e^{j\omega})\right]$$

For ω ranging from 0 to π, we observe that $H_2(e^{j\omega})$ undergoes a phase variation of $-\pi$, while $H_1(e^{j\omega})$ undergoes no phase variation, i.e., $H_2(e^{j\omega})$ exhibits an excess phase variation compared to $H_1(e^{j\omega})$. In general, for ω ranging from 0 to π, a causal and stable transfer function with all zeros outside the unit circle experiences an excess phase variation compared to a causal and stable transfer function with the same amplitude response but with all zeros inside the unit circle. Consequently, a transfer function with all zeros inside the unit circle is termed a *minimum-phase transfer function*, while if all zeros lie outside the unit circle, it is termed a *maximum-phase transfer function*.

07.10 Inverse system

Two LTI systems with impulse response $h_1(n)$ and $h_2(n)$ are inverses of each other if

$$h_1(n) \circledast h_2(n) = \delta(n),$$

i.e., if their convolution is the unit impulse function, indicating that the cascade of the two systems (in any order) results in the identity system.
Let us characterize the inverse system (or the inverse filter) in the frequency domain. By taking the Z-transform of both sides of the equality, we have

$$H_1(z) \cdot H_2(z) = 1,$$

which implies

$$H_2(z) = \frac{1}{H_1(z)}$$

and if $H_1(z)$ is rational,

$$H_1(z) = \frac{N(z)}{D(z)} \quad \Longrightarrow \quad H_2(z) = \frac{D(z)}{N(z)},$$

thus $H_2(z)$ is also rational, and the poles (zeros) of the inverse filter are the zeros (poles) of $H_1(z)$. Assuming the inverse system to be causal, it will be stable if and only if $H_1(z)$ is minimum-phase.
Note that by relinquishing the assumption of causality for the inverse filter, the inverse filter is not unique. For example, consider the causal system

$$H_1(z) = \frac{\left(z - \frac{1}{4}\right)\left(z + \frac{1}{5}\right)}{\left(z + \frac{1}{8}\right)\left(z - \frac{1}{7}\right)}$$

with R.O.C.: $|z| > \frac{1}{7}$
The inverse filter has a transfer function

$$H_2(z) = \frac{\left(z + \frac{1}{8}\right)\left(z - \frac{1}{7}\right)}{\left(z - \frac{1}{4}\right)\left(z + \frac{1}{5}\right)}$$

with three possible R.O.C.: (i) $|z| < \frac{1}{5}$, (ii) $\frac{1}{5} < |z| < \frac{1}{4}$, (iii) $|z| > \frac{1}{4}$. Each region of convergence corresponds to a different inverse system. Only the last R.O.C. corresponds to a causal system.

07.11 Deconvolution

If a system has a known causal impulse response $h(n)$ and it is excited by a causal input signal $x(n)$, then, from the knowledge of the output signal $y(n)$ for $n \geq 0$, we can estimate the input signal $x(n)$ using a recursive relation without the need to evaluate the inverse system. In fact, it is

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

Let us assume $h(0) \neq 0$,

$$
\begin{aligned}
y(0) &= h(0)x(0) && \implies && x(0) = \frac{y(0)}{h(0)} \\
y(1) &= h(0)x(1) + h(1)x(0) && \implies && x(1) = \frac{y(1) - h(1)x(0)}{h(0)} \\
y(2) &= h(0)x(2) + h(1)x(1) + h(2)x(0) && \implies && x(2) = \frac{y(2) - h(1)x(1) - h(2)x(0)}{h(0)} \\
y(n) &= \sum_{m=0}^{n} h(m)x(n-m) && \implies && x(n) = \frac{y(n) - \sum_{m=1}^{n} h(m)x(n-m)}{h(0)}.
\end{aligned}
$$

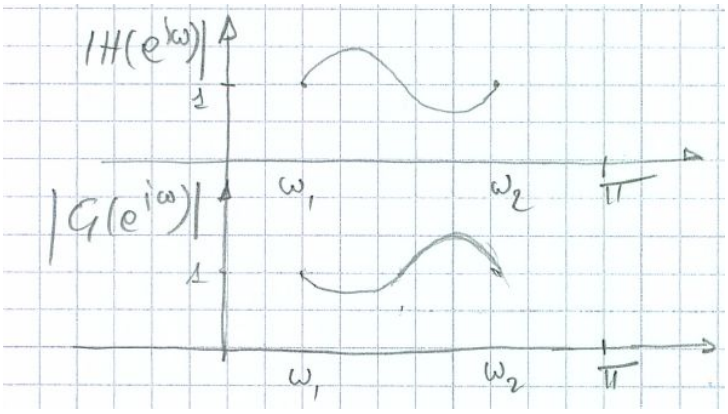
This procedure, which estimates the input signal $x(n)$ from the convolution sum, is called *deconvolution*.

07.12 Amplitude equalizer and phase equalizer

Given a system $H(z)$, a filter $G(z)$ whose amplitude response $|G(e^{j\omega})|$ satisfies the following condition

$$|G(e^{j\omega})| = |H(e^{j\omega})|^{-1} \qquad \forall \omega \in [\omega_1, \omega_2]$$

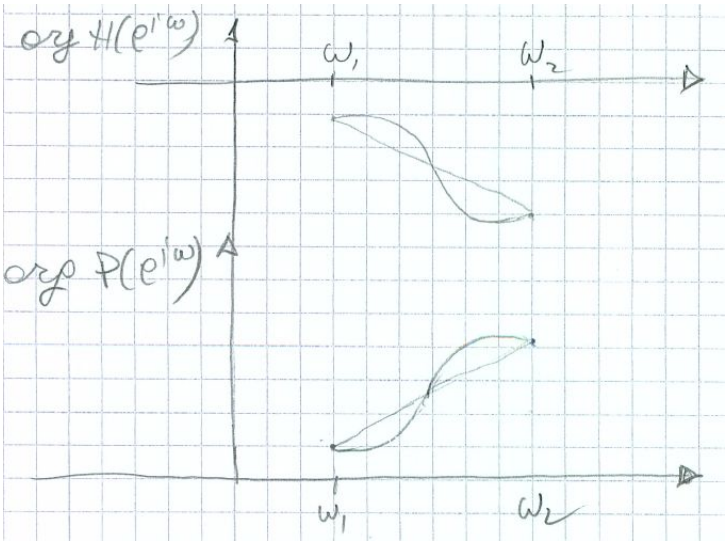
is called an *amplitude equalizer* for the system $H(z)$ in band $[\omega_1, \omega_2]$.



Given a system $H(z)$, a filter $P(z)$ whose phase response $\arg P(e^{j\omega})$ satisfies the following condition

$$\arg P(e^{j\omega}) = -\arg H(e^{j\omega}) - K\omega \qquad \omega \in [\omega_1, \omega_2],$$

with $K \in \mathbb{R}$, is called a *phase equalizer* for the system $H(z)$ in band $[\omega_1, \omega_2]$.



(in this case $K = 0$)

07.13 Stability test for IIR filters

We have seen that a causal IIR filter, described by a finite difference equation or by a rational transfer function, is stable in the BIBO sense if and only if its poles fall inside the unit circle. There exist stability tests that do not require the estimation of the poles.

07.13.1 The stability triangle

For real coefficient IIR filters of second order, we can easily determine if the filter is stable from the coefficients of the denominator. Consider the polynomial

$$D(z) = z^2 + a_1 z + a_2 = (z - z_1)(z - z_2)$$

$$z_{1,2} = -\frac{a_1}{2} \pm \sqrt{\frac{a_1^2 - 4a_2}{4}}$$

and thus

$$a_2 = z_1 \cdot z_2$$

$$a_1 = -(z_1 + z_2)$$

We can prove that the roots z_1 and z_2 fall inside the unit circle if and only if

$$|a_2| < 1,$$

$$|a_1| < 1 + a_2.$$

In fact, if the zeros are complex conjugate the condition $|a_2| < 1$ is necessary and sufficient for stability. If the zeros are real: $a_1^2 - 4a_2 \geq 0$ and it is still necessary that $|a_2| < 1$.

$$|z_{1,2}|_{\max} = \frac{|a_1|}{2} + \sqrt{\frac{a_1^2 - 4a_2}{4}},$$

from which, if $|a_2| < 1$, the filter is stable if and only if $|a_1| < 1 + a_2$.
In fact, if

$$\frac{|a_1|}{2} + \sqrt{\frac{a_1^2 - 4a_2}{4}} < 1$$

then

$$\sqrt{\frac{a_1^2 - 4a_2}{4}} < 1 - \frac{|a_1|}{2}$$

$$\frac{a_1^2 - 4a_2}{4} < 1 - |a_1| + \frac{a_1^2}{4}$$

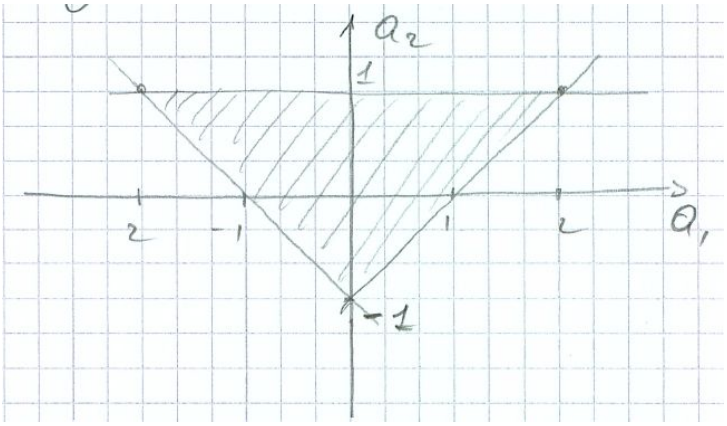
$$-a_2 < 1 - |a_1|$$

$$|a_1| < 1 + a_2.$$

Conversely, if $|a_1| < 1 + a_2$ then

$$|z_{1,2}|_{\max} < \frac{1 + a_2}{2} + \sqrt{\frac{(1 + a_2)^2 - 4a_2}{4}} = 1$$

On the plane a_1, a_2, the region of points for which the filter is stable is a triangle, called the *stability triangle*.



07.13.2 Schur-Cohn stability test

The Schur-Cohn stability test can be applied to polynomials of any order. Let us consider:

$$D_M(z) = \sum_{i=0}^{M} d_i z^{-i} = 1 + d_1 z^{-1} + \ldots + d_M z^{-M}$$

with $d_0 = 1$ for simplicity. Let us take the mirror image polynomial,

$$\widetilde{D}_M(z) = z^{-M} D_M(z^{-1}) = z^{-M} \sum_{i=0}^{M} d_i z^{i} =$$

$$= d_M + d_{M-1} z^{-1} + \ldots + d_1 z^{-M+1} + z^{-M},$$

and let us build the all-pass filter

$$A_M(z) = \frac{\widetilde{D}_M(z)}{D_M(z)}$$

If $D_M(z) = \prod_{i=1}^{M} \left(1 - \lambda_i z^{-1}\right)$, with λ_i the filter poles, then

$$d_M = \prod_{i=1}^{M} \lambda_i (-1)^M.$$

Thus, if we call $K_M = d_M$, a necessary condition for the all-pass filter stability is that

$$|K_M| < 1.$$

Let us assume $|K_M| < 1$ and let us build the all-pass filter

$$\begin{aligned} A_{M-1}(z) &= z \cdot \left[\frac{A_M(z) - K_M}{1 - K_M A_M(z)} \right] \\ &= z \cdot \frac{\widetilde{D}_M(z) - d_M D_M(z)}{D_M(z) - d_M \widetilde{D}_M(z)} = \\ &= z \cdot \frac{(d_M - d_M \cdot 1) + (d_{M-1} - d_M d_1) z^{-1} + \ldots + (1 - d_M^2) z^{-M}}{(1 - d_M^2) + (d_1 - d_M d_{M-1}) z^{-1} + \ldots + (d_M - d_M \cdot 1) z^{-M}} = \\ &= \frac{(d_{M-1} - d_M d_1) + \ldots + (1 - d_M^2) z^{-M+1}}{(1 - d_M^2) + (d_1 - d_M d_{M-1}) z^{-1} + \ldots + (d_{M-1} - d_M d_1) z^{-M+1}}. \end{aligned}$$

It can be proved that the all-pass filter $A_M(z)$ is BIBO stable if and only if $|K_M| < 1$ and $A_{M-1}(z)$ is BIBO stable.

Proof:

The proof is based on the fact that if the real-coefficients all-pass filter $A_M(z)$ is stable then:

$$|A_M(z)| = 1 \quad \text{for} \quad |z| = 1,$$

$$|A_M(z)| < 1 \quad \text{for} \quad |z| > 1,$$

$$|A_M(z)| > 1 \quad \text{for} \quad |z| < 1.$$

We have already seen that the condition $|K_M| < 1$ is necessary for the stability.

In the hypothesis that $|K_M| < 1$ and that $A_M(z)$ is stable, let us prove that $A_{M-1}(z)$ is stable.

If λ_0 is a pole of $A_{M-1}(z)$ then λ_0 is a root of the equation

$$D_M(z) - K_M \widetilde{D}_M(z) = 0$$

i.e., $A_M(\lambda_0) = \dfrac{\widetilde{D}_M(\lambda_0)}{D_M(\lambda_0)} = \dfrac{1}{K_M}$. Since $|K_M| < 1$, $|A_M(\lambda_0)| > 1$, and λ_0 must fall inside the unit circle because of the stability of $A_M(z)$.

Let us prove that $A_M(z)$ is stable when $|K_M| < 1$ and $A_{M-1}(z)$ is stable.

But if λ_0 is a pole of $A_M(z)$, $D_M(\lambda_0) = 0$,

$$A_{M-1}(\lambda_0) = -\lambda_0 \frac{\widetilde{D}_M(\lambda_0)}{K_M \widetilde{D}_M(\lambda_0)} = -\frac{\lambda_0}{K_M}$$

Since $|K_M| < 1$, $\left| \dfrac{1}{\lambda_0} \cdot A_{M-1}(\lambda_0) \right| > 1$, and

$$|A_{M-1}(\lambda_0)| > |\lambda_0|$$

If for absurd we assume $|\lambda_0| > 1$, we also have $|A_{M-1}(\lambda_0)| > 1$, which contradicts the hypothesis of stability of $A_{M-1}(z)$. Thus, it must be $|\lambda_0| < 1$. Q.E.D.

The test procedure can be repeated. Let us consider:

$$A_{M-1}(z) = \frac{d'_{M-1} + d'_{M-2} z^{-1} + \ldots + d'_1 z^{-M+2} + z^{-M+1}}{1 + d'_1 z^{-1} + \ldots + d'_{M-1} z^{-M+1}}$$

with

$$d'_i = \frac{d_i - d_M d_{M-i}}{1 - d_M^2} = \frac{d_i - K_M d_{M-i}}{1 - K_M^2}$$

Let us set $K_{M-1} = d'_{M-1}$ and build

$$A_{M-2}(z) = z \cdot \frac{A_{M-1}(z) - K_{M-1}}{1 - K_{M-1}A_{M-1}(z)}$$

the filter $A_M(z)$ is stable if and only if $|K_M| < 1$, $|K_{M-1}| < 1$, and $A_{M-2}(z)$ is stable.
By iterating this procedure $M-1$ times, given the coefficients K_M, K_{M-1}, ..., K_1 associated with the all-pass filters $A_M(z)$, $A_{M-1}(z)$, ..., $A_1(z)$, the all-pass filter $A_M(z)$ is stable (and the polynomial $D_M(z)$ has all its roots inside the unit circle) if and only if

$$|K_i| < 1 \quad \forall i.$$

For exercise, let's ascertain whether the polynomial $D_4(z)$,

$$D_4(z) = 1 + \frac{1}{3}z^{-1} - \frac{2}{15}z^{-2} - \frac{1}{3}z^{-3} + \frac{1}{3}z^{-4},$$

has roots inside the unit circle.
We apply the Schur-Cohn stability test. To apply the method it suffices to remember that the denominator of $A_{M-1}(z)$ is given by $[D_M(z) - K_M\widetilde{D}_M(z)]/(1-K_M^2)$.
$K_4 = \frac{1}{3}$, and $1 - K_4^2 = 1 - \frac{1}{9} = \frac{8}{9}$.

$$\begin{aligned}
D_3(z) &= [D_4(z) - K_4\widetilde{D}_4(z)]/(1-K_4^2) = \\
&= \left[1 + \frac{1}{3}z^{-1} - \frac{2}{15}z^{-2} - \frac{1}{3}z^{-3} + \frac{1}{3}z^{-4} - \frac{1}{3}\left(\frac{1}{3} - \frac{1}{3}z^{-1} - \frac{2}{15}z^{-2} + \frac{1}{3}z^{-3} + z^{-4}\right)\right] / \frac{8}{9} = \\
&= \left[\left(1 - \frac{1}{9}\right) + \left(\frac{1}{3} + \frac{1}{9}\right)z^{-1} + \left(-\frac{2}{15} + \frac{2}{45}\right)z^{-2} + \left(-\frac{1}{3} - \frac{1}{9}\right)z^{-3} + \left(\frac{1}{3} - \frac{1}{3}\right)z^{-4}\right] / \frac{8}{9} = \\
&= \left[\frac{8}{9} + \frac{4}{9}z^{-1} - \frac{4}{45}z^{-2} - \frac{4}{9}z^{-3}\right] / \frac{8}{9} = \\
&= 1 + \frac{1}{2}z^{-1} - \frac{1}{10}z^{-2} - \frac{1}{2}z^{-3}.
\end{aligned}$$

$K_3 = -\frac{1}{2}$, $1 - K_3^2 = \frac{3}{4}$

$$\begin{aligned}
D_2(z) &= [D_3(z) - K_3\widetilde{D}_3(z)]/(1-K_3^2) = \\
&= \left[1 + \frac{1}{2}z^{-1} - \frac{1}{10}z^{-2} - \frac{1}{2}z^{-3} + \frac{1}{2}\left(-\frac{1}{2} - \frac{1}{10}z^{-1} + \frac{1}{2}z^{-2} + z^{-3}\right)\right] / \frac{3}{4} = \\
&= \left[\left(1 - \frac{1}{4}\right) + \left(\frac{1}{2} - \frac{1}{20}\right)z^{-1} + \left(-\frac{1}{10} + \frac{1}{4}\right)z^{-2} + \left(-\frac{1}{2} + \frac{1}{2}\right)z^{-3}\right] / \frac{3}{4} = \\
&= \left[\frac{3}{4} + \frac{9}{20}z^{-1} + \frac{3}{20}z^{-2}\right] / \frac{3}{4} = \\
&= 1 + \frac{3}{5}z^{-1} + \frac{1}{5}z^{-2}.
\end{aligned}$$

$K_2 = \frac{1}{5}$, $1 - K_2^2 = \frac{24}{25}$

$$\begin{aligned}
D_1(z) &= [D_2(z) - K_2\widetilde{D}_2(z)]/(1-K_2^2) \\
&= \left[1 + \frac{3}{5}z^{-1} + \frac{1}{5}z^{-2} - \frac{1}{5}\left(\frac{1}{5} + \frac{3}{5}z^{-1} + z^{-2}\right)\right] / \frac{24}{25} = \\
&= \left[\left(1 - \frac{1}{25}\right) + \left(\frac{3}{5} - \frac{3}{25}\right)z^{-1} + \left(\frac{1}{5} - \frac{1}{5}\right)z^{-2}\right] / \frac{24}{25} = \\
&= \left[\frac{24}{25} + \frac{12}{25}z^{-1}\right] / \frac{24}{25} =
\end{aligned}$$

$$= 1 + \frac{1}{2} z^{-1}.$$

$K_1 = \frac{1}{2}$.
Since K_1, K_2, K_3, and K_4 have absolute value less than 1, the polynomial has roots inside the unit circle.

For exercise, write a Matlab function that, given the vector of the polynomial coefficients, computes the reflection coefficients K_i of the Schur-Cohn method.

For more information study:

S. K. Mitra, "Digital Signal Processing: a computer based approach," 4th edition, McGraw-Hill, 2011
Chapter 4.9, pp. 185-188
Chapter 7.1, pp. 333-335
Chapter 7.2.1, pp. 342-344
Chapter 7.3, pp. 349-360
Chapter 6.7.3-6.7.4, pp. 312-315
Chapter 7.2.3, pp. 346-349
Chapter 7.6, pp. 385-388
Chapter 7.9, pp. 394-399

08 Digital filter structures

08.01 Basic building blocks

Let us consider an IIR filter described by the finite difference equation

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) - \sum_{k=1}^{N} a_k y(n-k)$$

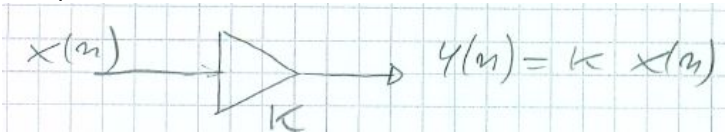
thus with transfer function

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}.$$

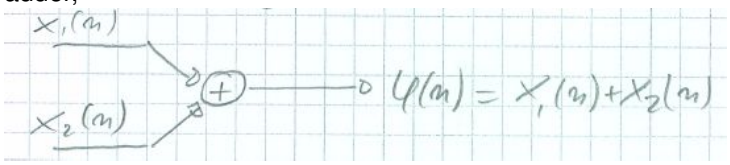
In the following sections, we will explore various methods for implementing both IIR filters and FIR filters. The finite difference equation can be broken down into an equivalent set of finite difference equations. Furthermore, the transfer function can be decomposed into the sum or product of several simpler transfer functions. From any of these sets of equations or transfer functions, we can construct different block diagrams representing the hardware or software operations required to implement the system. Each of these block diagrams is termed a 'realization' of the system or an implementation structure for the system.

The fundamental building blocks of our block diagrams essentially consist of four elements:

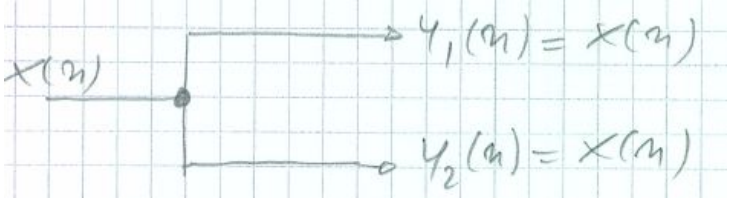
- multiplier,



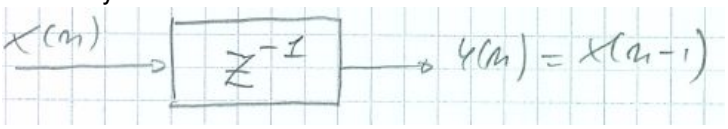
- adder,



- pick-off node,



- unit delay.



The various realizations of a system differ in computation complexity (the number of arithmetic operations required to compute the output) and memory requirements (necessary to store the system parameters, input, and output values). Although they are all equivalent when we work with infinite precision arithmetic, they exhibit different properties when working with finite precision arithmetic.

08.02 FIR filter structures

$$y(n) = \sum_{k=0}^{M-1} b_k x(n-k)$$

$$H(z) = \sum_{k=0}^{M-1} b_k z^{-k}$$

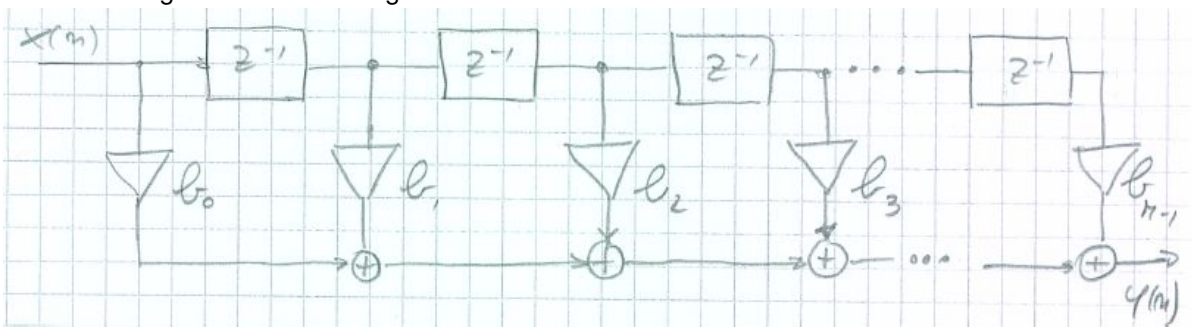
$$h(n) = \begin{cases} b_n & 0 \leq n \leq M-1 \\ 0 & \text{elsewhere} \end{cases}$$

08.02.1 Direct form realization

It directly implements the non-recursive finite difference equation:

$$y(n) = \sum_{k=0}^{M-1} b_k x(n-k) = \sum_{k=0}^{M-1} h(k) x(n-k).$$

The block diagram is the following:



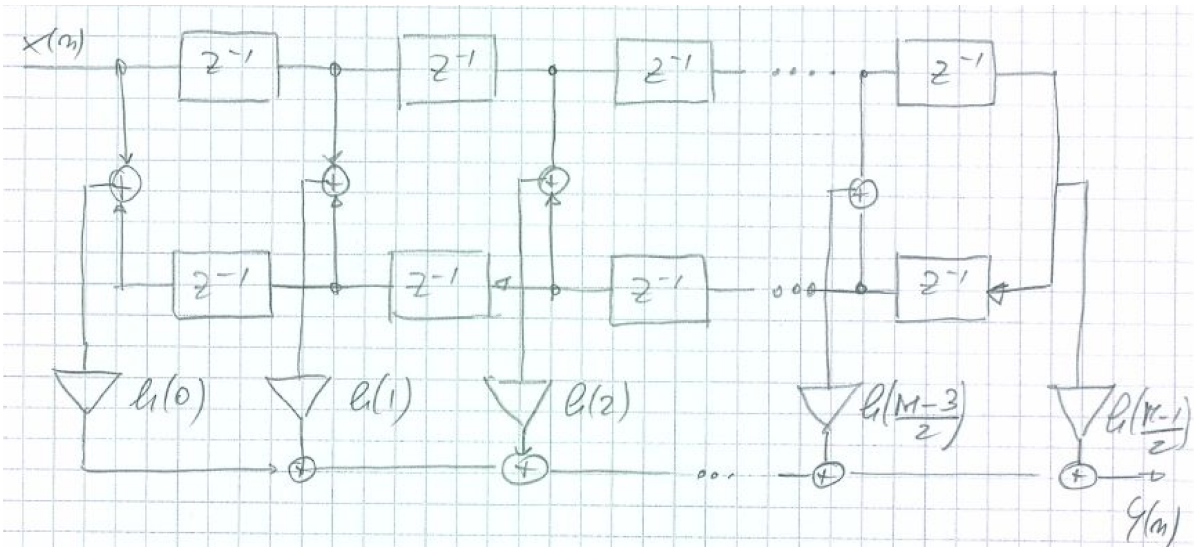
This structure requires $M-1$ unit delay elements, M constant multipliers, and $M-1$ adders.

The filter is also referred to as a *transversal filter* or *tapped delay line* due to its particular structure. When the FIR filter has linear phase (i.e., it is symmetric or anti-symmetric), the structure can be further simplified. Let us consider, for example, a symmetric filter with M odd:

$$y(n) = h(0)x(n) + h(1)x(n-1) + \ldots + h(M-2)x(n-M+2) + h(M-1)x(n-M+1) =$$

$$= h(0)[x(n) + x(N-M+1)] + h(1)[x(n-1) + x(n-M+2)] + \ldots + h(\frac{M-1}{2})x(n-\frac{M-1}{2}),$$

which can be realized with the following structure:



For these systems, the number of products reduces to $\frac{M+1}{2}$ for M odd and $\frac{M}{2}$ for M even, while the number of adders and delay elements remains the same.

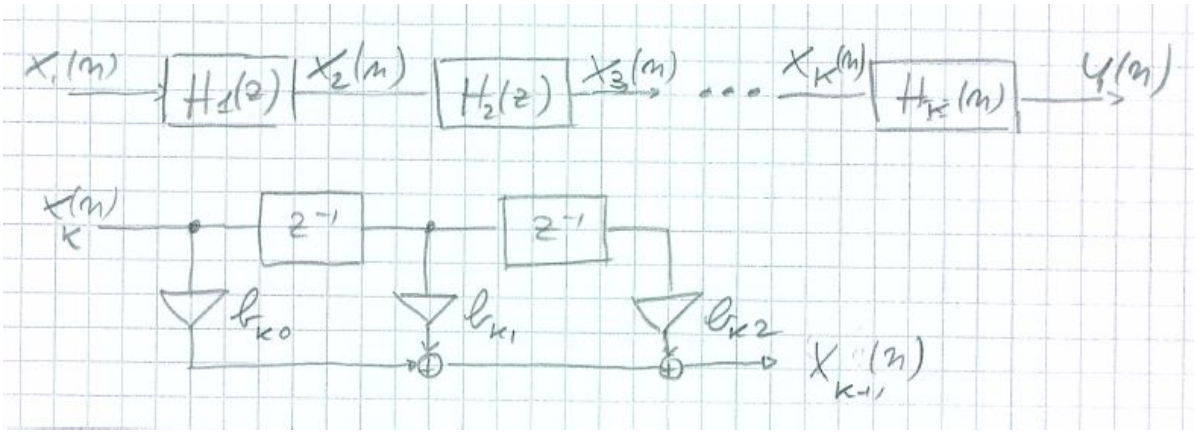
08.02.2 Cascade realization

The cascade realization is achieved by factorizing the transfer function $H(z)$ using transfer functions of second or first order:

$$H(z) = \prod_{k=1}^{K} H_k(z),$$

$$H_k(z) = b_{k0} + b_{k1}z^{-1} + b_{k2}z^{-2},$$

with $k = 1, 2, \ldots, K$, and K is the smallest integer greater than or equal to $\frac{M-1}{2}$.
To determine the zeros of $H_k(z)$, the zeros of $H(z)$ are paired together. Naturally, if the transfer function has real coefficients, it is preferable to pair complex-conjugate zeros together, ensuring that the coefficients b_{ki} are also real. Real zeros of $H(z)$ can be grouped together in any order.



08.02.3 Frequency-sampling structure

This is a realization of FIR filters where the filter parameters are determined by the desired values of the frequency response.
To derive this structure, we specify the desired values of the frequency response at M uniformly spaced points on the unit circle (where M is the filter length), denoted as $e^{j\omega_k}$, with:

$$\omega_k = \frac{2\pi}{M}k, \qquad\qquad k = 0, \dots, M-1,$$

and then compute the impulse response $h(n)$ from these values.
The frequency response is given by

$$H(e^{j\omega}) = \sum_{n=0}^{M-1} h(n)e^{-j\omega n}$$

and for $\omega = \frac{2\pi}{M}k = \omega_k$,

$$H(e^{j\omega_k}) = \sum_{n=0}^{M-1} h(n)e^{-j\frac{2\pi}{M}kn}$$

which is the DFT on M points of the impulse response. Thus, computing the IDFT

$$h(n) = \frac{1}{M}\sum_{k=0}^{M-1} H(e^{j\omega_k})e^{j\frac{2\pi}{M}kn}$$

Let's substitute these values into the transfer function expression:

$$H(z) = \sum_{n=0}^{M-1} h(n)z^{-n} = \sum_{n=0}^{M-1}\left[\frac{1}{M}\sum_{k=0}^{M-1} H(e^{j\omega_k})e^{j\frac{2\pi}{M}kn}\right]z^{-n}$$

$$= \sum_{k=0}^{M-1}\frac{H(e^{j\omega_k})}{M}\left[\sum_{n=0}^{M-1} e^{j\frac{2\pi}{M}kn}z^{-n}\right] =$$

$$= \sum_{k=0}^{M-1}\frac{H(e^{j\omega_k})}{M}\frac{1-z^{-M}}{1-e^{j\frac{2\pi}{M}k}z^{-1}} =$$

$$= (1 - z^{-M}) \cdot \sum_{k=0}^{M-1} \frac{H(e^{j\omega_k})/M}{1 - e^{j\frac{2\pi}{M}k} z^{-1}}$$

This transfer function is characterized by the frequency domain samples, rather than the impulse response samples.
We can then implement the FIR filter as the cascade of two systems:

$$H(z) = H_1(z) \cdot H_2(z),$$

where

$$H_1(z) = 1 - z^{-M},$$

$$H_2(z) = \sum_{k=0}^{M-1} \frac{1}{1 - e^{j\frac{2\pi}{M}k} z^{-1}} \cdot \frac{H(e^{j\omega_k})}{M}.$$

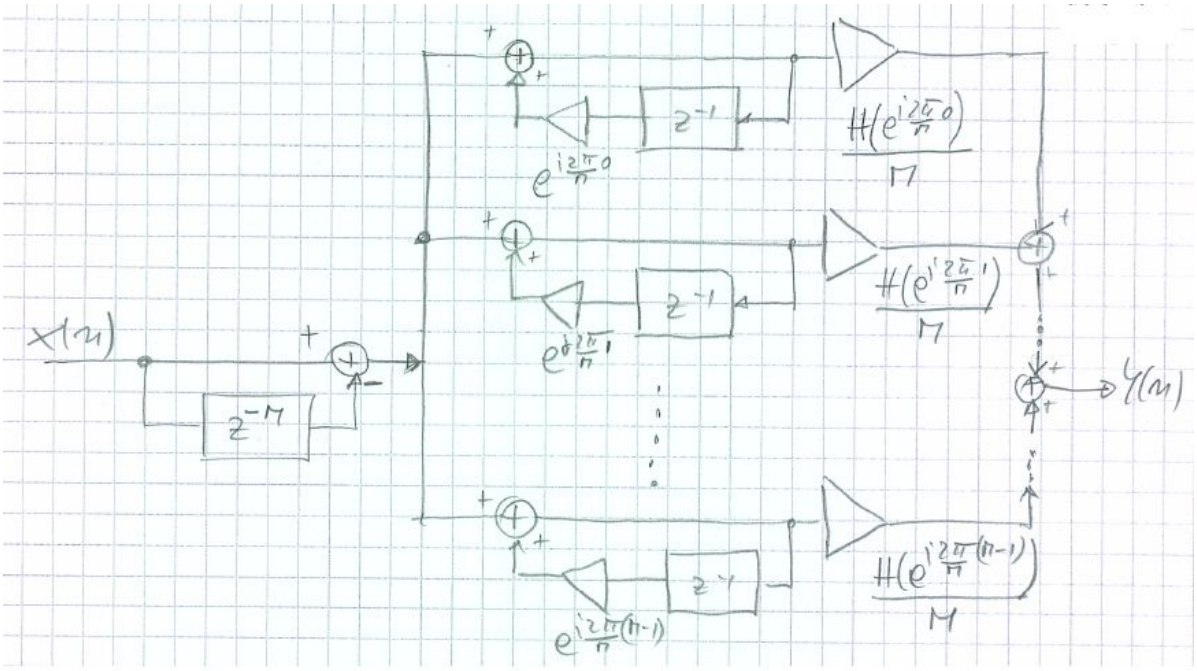
$H_1(z)$ represents an FIR filter and, specifically, it is a comb filter with M zeros uniformly spaced on the unit circle:

$$z_k = e^{j\frac{2\pi}{M}k} \qquad k = 0, \ldots, M-1.$$

$H_2(z)$ is a filter bank comprising a parallel connection of M filters, each having a single pole, and the M poles are again uniformly spaced on the unit circle:

$$p_k = e^{j\frac{2\pi}{M}k} \qquad k = 0, \ldots, M-1.$$

Given a certain desired frequency response, we can implement it directly with the following filter structure:



This realization is particularly advantageous for narrow bandpass filters, where $H(e^{j\omega_k}) = 0$ for many values of k. In such cases, many branches are not implemented, leading to more efficient utilization of resources.

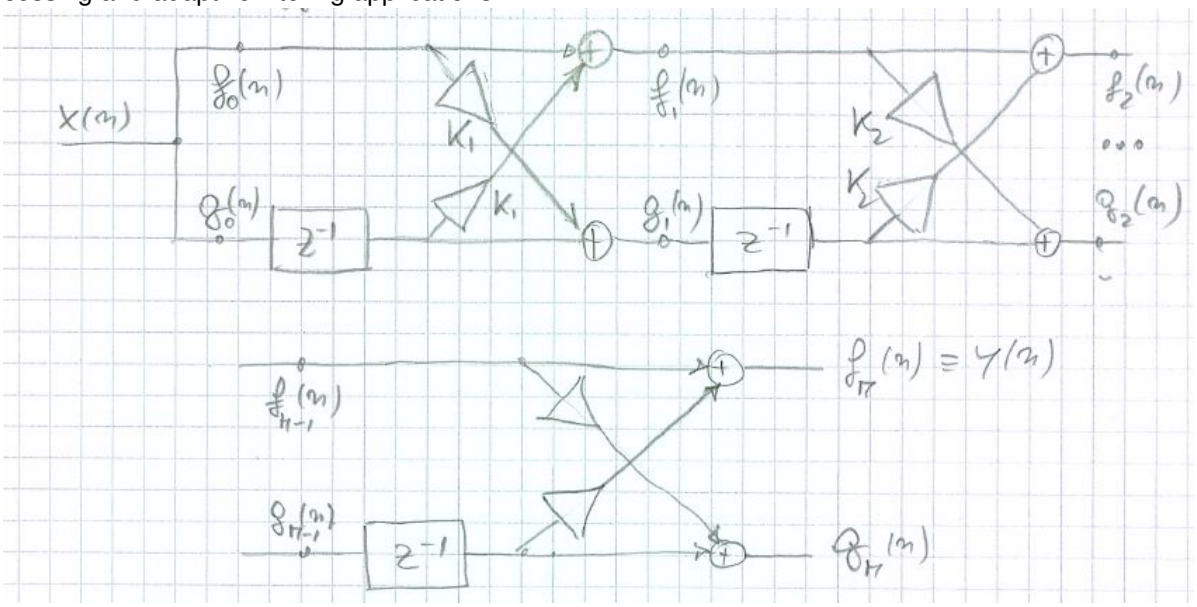
This structure implements filters with complex coefficients. In the case of filters with real coefficients:

$$H(e^{j\omega_k}) = H^*(e^{j\omega_{M-k}})$$

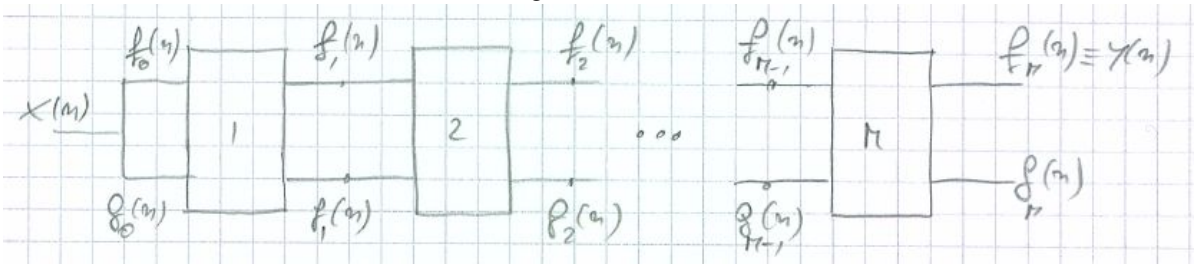
and consequently, the structure can be simplified by combining the complex conjugate branches. Moreover, it's worth noting that with this structure, we implement an FIR filter through the cascade of an FIR filter (the comb filter) and an IIR filter.

08.02.4 Lattice structure

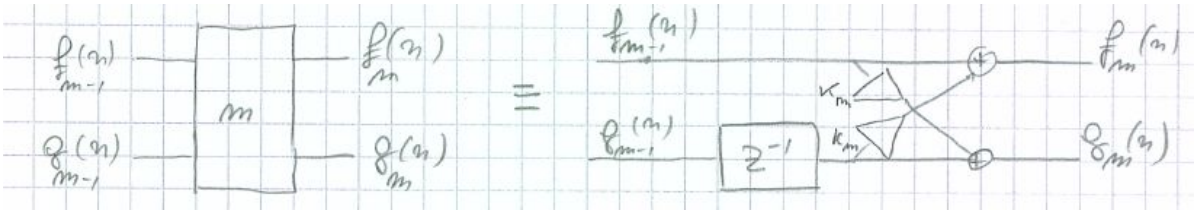
Another structure for implementing FIR filters is the lattice structure, commonly employed in audio processing and adaptive filtering applications.



The structure consists of a cascade of M stages:



where each stage is given by:



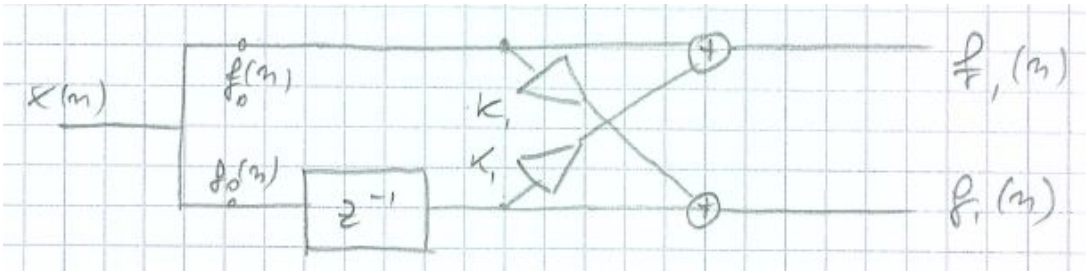
and it implements a filter with the transfer function:

$$H_M(z) = 1 + \alpha_M(1)z^{-1} + \alpha_M(2)z^{-2} + \ldots + \alpha_M(M)z^{-M} = 1 + \sum_{k=1}^{M} \alpha_M(k)z^{-k}$$

where M is the polynomial degree, equal to the number of stages
Studying a block diagram and obtaining a description of the system in terms of finite difference equations is a straightforward process. The following procedure can be followed:

1. Assign names to the block diagram inputs and outputs, as well as to all delay element inputs.
2. Write a finite difference equation for each delay element input and each output. Each equation computes the specific delay element input or block diagram output based on the block diagram inputs and delay element outputs.
3. Simplify the system of finite difference equations if necessary.

Some authors also assign a name (a variable) to each adder and write an equation for each adder, but this step is optional.
Let us study the block diagram of a lattice filter using this procedure, starting from order 1



$$\begin{aligned}
f_0(n) &= x(n) \\
g_0(n) &= x(n) \\
f_1(n) &= f_0(n) + K_1 g_0(n-1) = x(n) + K_1 x(n-1) \\
g_1(n) &= g_0(n-1) + K_1 f_0(n) = x(n-1) + K_1 x(n)
\end{aligned}$$

(Note that the output of the delay element is $g_0(n-1)$).
Thus, the first-order lattice filter is an FIR filter of order 1 with a length of 2. Moreover, if we consider the two transfer functions:

$$\frac{F_1(z)}{X(z)} = 1 + K_1 z^{-1} = D_1(z)$$
$$\frac{G_1(z)}{X(z)} = z^{-1} + K_1 = \widetilde{D}_1(z)$$

we observe that $\widetilde{D}_1(z)$ is the mirror image polynomial of $D_1(z)$, i.e., $\widetilde{D}_1(z) = z^{-1} \cdot D_1(z^{-1})$.
Similarly, if we consider a lattice filter of the second order, we also have the following equations:

$$f_2(n) = f_1(n) + K_2 g_1(n-1) = x(n) + K_1 x(n-1) + K_2 \left(x(n-2) + K_1 x(n-1)\right) =$$

$$= x(n) + (K_1 + K_2K_1)\, x(n-1) + K_2 x(n-2)$$

$$g_2(n) = g_1(n-1) + K_2 f_1(n) = x(n-2) + K_1 x(n-1) + K_2\,(x(n) + K_1 x(n-1)) =$$

$$= K_2 x(n) + (K_1 + K_1K_2)\, x(n-1) + x(n-2),$$

which represents two filters of order 2 with a length of 3.

$$\frac{F_2(z)}{X(z)} = 1 + (K_1 + K_2K_1)\, z^{-1} + K_2 z^{-2} = D_2(z)$$

$$\frac{G_2(z)}{X(z)} = K_2 + (K_1 + K_2K_1)\, z^{-1} + z^{-2} = \widetilde{D}_2(z)$$

where $D_2(z)$ and $\widetilde{D}_2(z)$ are again a pair of mirror-image polynomials.
We can prove by induction that a lattice filter of order m is an FIR filter of order m, with a length of $m+1$, and that the two transfer functions $D_m(z)$ and $\widetilde{D}_m(z)$ form a pair of mirror-image polynomials.
Let us assume

$$D_{m-1}(z) = 1 + \sum_{k=1}^{m-1} \alpha_{m-1}(k) z^{-k} = \sum_{k=0}^{m-1} \alpha_{m-1}(k) z^{-k}$$

$$\widetilde{D}_{m-1}(z) = z^{-(m-1)} + \sum_{k=1}^{m-1} \alpha_{m-1}(k) z^{-(m-1-k)} = \sum_{k=0}^{m-1} \alpha_{m-1}(k) z^{-(m-1-k)}$$

with $\alpha_{m-1}(0) = 1$. We have

$$\frac{F_m(z)}{X(z)} = \frac{F_{m-1}(z)}{X(z)} + K_m z^{-1} \frac{G_{m-1}(z)}{X(z)}$$

$$D_m(z) = D_{m-1}(z) + K_m z^{-1} \widetilde{D}_{m-1}(z) =$$

$$= \sum_{k=0}^{m-1} \alpha_{m-1}(k) z^{-k} + K_m z^{-1} \sum_{k=0}^{m-1} \alpha_{m-1}(k) z^{-(m-1-k)} =$$

$$= \sum_{k=0}^{m-1} \alpha_{m-1}(k) z^{-k} + K_m z^{-1} \sum_{k=0}^{m-1} \alpha_{m-1}(m-1-k) z^{-k} =$$

$$= \sum_{k=0}^{m} \alpha_m(k) z^{-k}$$

where

$$\alpha_m(0) = \alpha_{m-1}(0) = 1,$$

$$\alpha_m(k) = \alpha_{m-1}(k) + K_m \alpha_{m-1}(m-k) \qquad \forall k = 1, \ldots, m-1$$

$$\alpha_m(m) = K_m$$

$$\frac{G_m(z)}{X(z)} = z^{-1} \frac{G_{m-1}(z)}{X(z)} + K_m \frac{F_{m-1}(z)}{X(z)}$$

$$\widetilde{D}_m(z) = z^{-1} \widetilde{D}_{m-1}(z) + K_m D_{m-1}(z)$$

Since

$$\widetilde{D}_{m-1}(z) = z^{-(m-1)} D_{m-1}(z^{-1})$$

$$D_{m-1}(z) = z^{-(m-1)} \widetilde{D}_{m-1}(z^{-1})$$

we have

$$\widetilde{D}_m(z) = z^{-1} \cdot z^{-(m-1)} D_{m-1}(z^{-1}) + K_m z^{-(m-1)} \widetilde{D}_{m-1}(z^{-1}) =$$
$$= z^{-m} \left(D_{m-1}(z^{-1}) + K_m z \widetilde{D}_{m-1}(z^{-1}) \right) =$$
$$= z^{-m} D_m(z^{-1})$$

and thus $\widetilde{D}_m(z)$ is the mirror image polynomial of $D_m(z)$. Q.E.D.

Note that

$$D_m(z) = D_{m-1}(z) + K_m z^{-1} \widetilde{D}_{m-1}(z)$$
$$\widetilde{D}_m(z) = z^{-1} \widetilde{D}_{m-1}(z) + K_m D_{m-1}(z)$$

which in matrix form gives

$$\begin{bmatrix} D_m(z) \\ \widetilde{D}_m(z) \end{bmatrix} = \begin{bmatrix} 1 & K_m z^{-1} \\ K_m & z^{-1} \end{bmatrix} \cdot \begin{bmatrix} D_{m-1}(z) \\ \widetilde{D}_{m-1}(z) \end{bmatrix}$$

and, overall,

$$\begin{bmatrix} D_M(z) \\ \widetilde{D}_M(z) \end{bmatrix} = \begin{bmatrix} 1 & K_M z^{-1} \\ K_M & z^{-1} \end{bmatrix} \cdot \begin{bmatrix} 1 & K_{M-1} z^{-1} \\ K_{M-1} & z^{-1} \end{bmatrix} \cdot \ldots \begin{bmatrix} 1 & K_1 z^{-1} \\ K_1 & z^{-1} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

where we considered $D_0(z) = \widetilde{D}_0(z) = 1$.

Given an FIR lattice filter with coefficients $K_1, K_2, \ldots, K_M$, referred to as *reflection coefficients*, we can compute its transfer function by recursively applying the two relations:

$$D_m(z) = D_{m-1}(z) + K_m z^{-1} \widetilde{D}_{m-1}(z)$$

$$\widetilde{D}_m(z) = z^{-m} D_m(z^{-1})$$

with initial conditions $D_0(z) = \widetilde{D}_0(z) = 1$.

Similarly, given the polynomial

$$D_m(z) = 1 + \sum_{k=1}^{m} \alpha_m(k) z^{-k}$$

we can derive the reflection coefficients $K_1, K_2, \ldots, K_M$ using a recursion.

For what we have seen before,

$$K_M = \alpha_M(M),$$

the last coefficient of $D_m(z)$,

$$K_{M-1} = \alpha_{M-1}(M-1),$$

the last coefficient of $D_{m-1}(z)$, etc. until

$$K_1 = \alpha_1(1).$$

Thus, we can obtain the reflection coefficients by constructing the polynomials $D_m(z)$.

Since,

$$\widetilde{D}_m(z) = z^{-1} \widetilde{D}_{m-1}(z) + K_m D_{m-1}(z)$$

we have

$$z^{-1} \widetilde{D}_{m-1}(z) = \widetilde{D}_m(z) - K_m D_{m-1}(z)$$

From

$$D_m(z) = D_{m-1}(z) + K_m z^{-1} \widetilde{D}_{m-1}(z)$$

we find

$$D_{m-1}(z) = D_m(z) - K_m z^{-1} \widetilde{D}_{m-1}(z) =$$
$$= D_m(z) - K_m \left(\widetilde{D}_m(z) - K_m D_{m-1}(z) \right)$$

If $|K_m| \neq 1$,

$$D_{m-1}(z) = \frac{D_m(z) - K_m \widetilde{D}_m(z)}{1 - K_m^2}.$$

By applying the last equation we can recursively compute the polynomials $D_m(z)$ and the reflection coefficients K_m:

Algorithm

`For` $m = M$ `till` 1

$\widetilde{D}_m(z) = z^{-m} D_m(z^{-1})$

$K_m = [z^m D_m(z)]\Big|_{z=0}$

$D_{m-1}(z) = \dfrac{D_m(z) - K_m \widetilde{D}_m(z)}{1 - K_m^2}$

`End For`

Note that the recursive algorithm used for computing the polynomials $D_m(z)$ is the same algorithm used for the Schur-Cohn stability test.

The recursive procedure works provided that $|K_m| \neq 1$. If $|K_m| = 1$ for any m, the polynomial $D_M(z)$ has a zero on the unit circle. The lattice structure can still be used for the implementation of the filter, but all the polynomial factors $\left(1 + e^{j\theta} z^{-1}\right)$ must be taken out of the transfer function and implemented with one of the previous structures.

08.03 IIR filter structures

08.03.1 Direct form realization of IIR filters

The rational transfer function that characterizes the IIR systems considered in practice,

$$H(z) = \frac{\sum_{i=0}^{M} b_i z^{-i}}{1 + \sum_{k=1}^{N} a_k z^{-k}},$$

can be interpreted as the cascade of two systems $H_1(z)$ and $H_2(z)$,

$$H(z) = H_1(z) \cdot H_2(z)$$

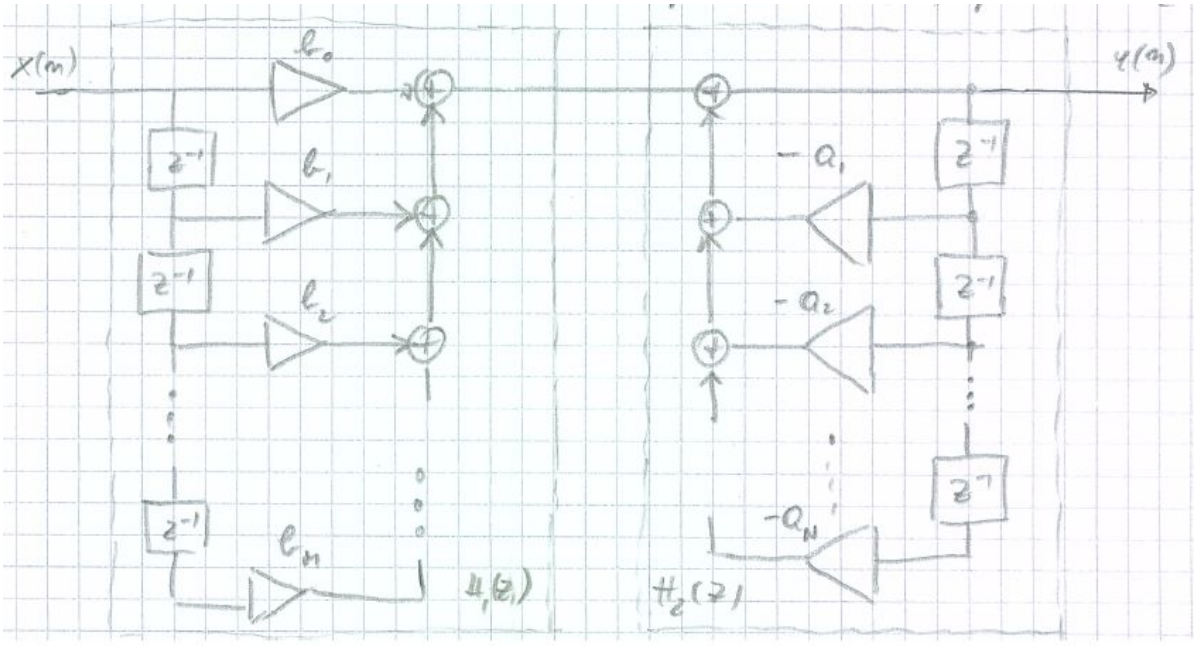
where $H_1(z)$ is an all-zeros system,

$$H_1(z) = \sum_{i=0}^{M} b_i z^{-i},$$

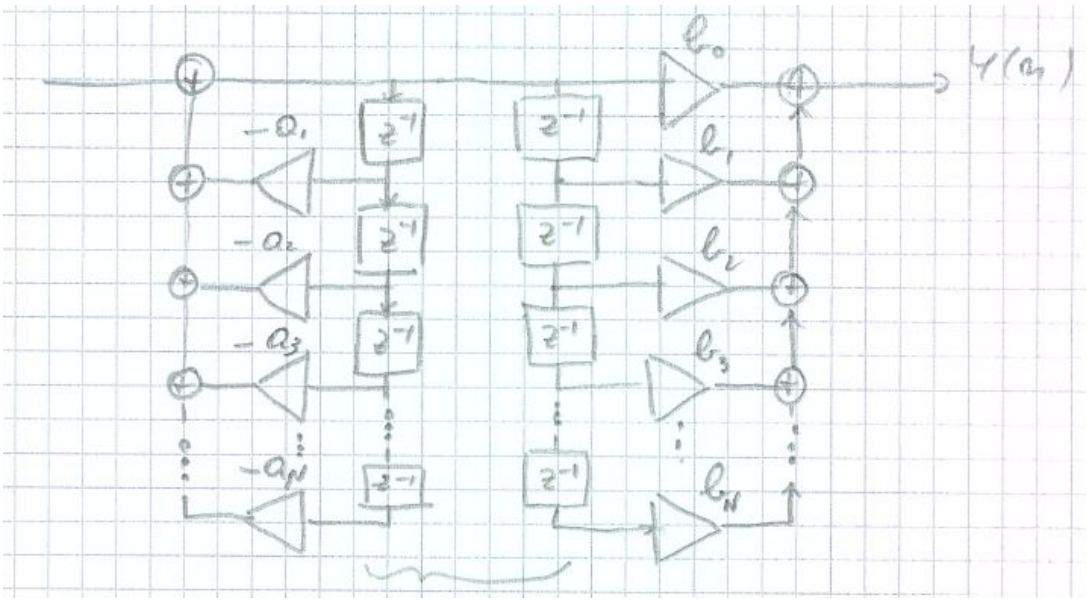
and $H_2(z)$ is an all-poles system,

$$H_2(z) = \frac{1}{1 + \sum_{k=1}^{N} a_k z^{-k}}.$$

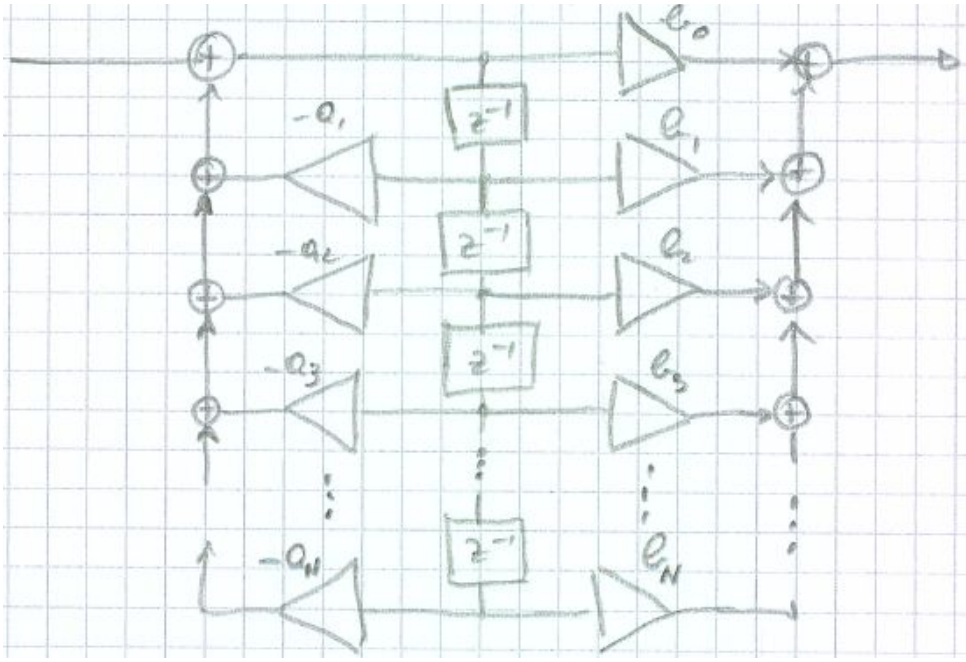
Depending on which of the two systems comes first, we obtain two different direct realizations.
In the *direct form I* realization $H_1(z)$ precedes $H_2(z)$.



The direct form I realization requires $M + N + 1$ multiplications, $M + N$ additions, and $M + N$ delay elements. A more compact structure can be obtained if $H_2(z)$ precedes $H_1(z)$. If $M = N$ we obtain the structure depicted in the following figure.



Note that the delay elements contain the same terms, thus the filter can be realized by combining the delays.

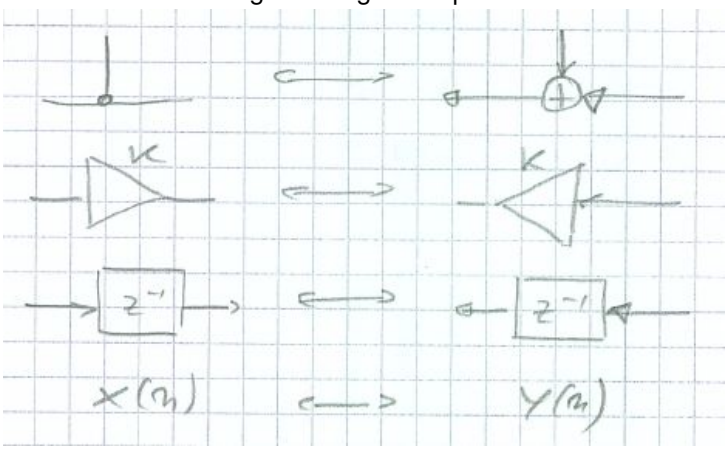


This realization is called the *direct form II* realization. Since this realization minimizes the number of delay elements, it is also referred to as the *canonical realization*.

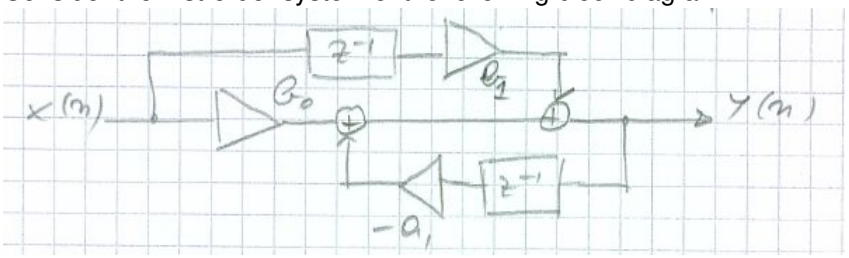
Both direct form realizations have been obtained from the transfer function $H(z)$ without any manipulations, hence the name ”direct form” realization. Unfortunately, both realizations are extremely sensitive to coefficient quantization and are not widely used in practice, except for orders 1 and 2. Indeed, when N is large, even small changes in the filter coefficients due to quantization can lead to significant movements of the system’s zeros and poles (and even to the instability of the IIR filter).

08.03.2 Transposition principle

This principle originates from graph theory and is used to derive additional realization structures.
If we reverse the direction of each branch, exchange the input and output, replace adders with pick-off nodes, and vice versa, in the block diagram implementing the transfer function $H(z)$, the resulting structure remains a realization for $H(z)$.
Note that the following exchanges are performed:



Example:
Consider the first-order system of the following block diagram:

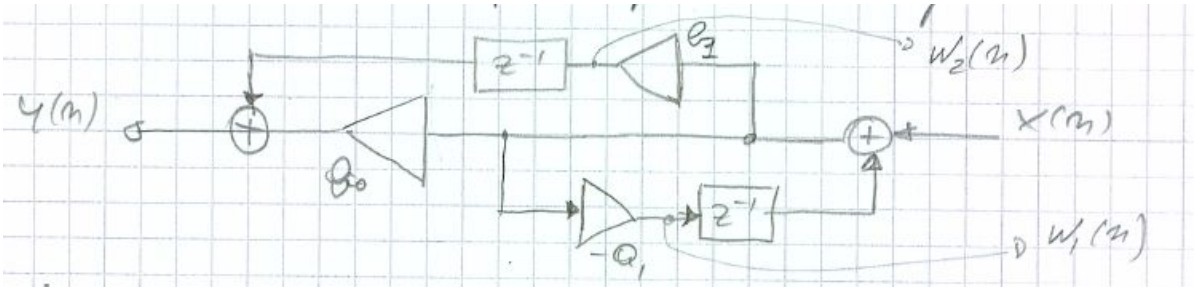


$$y(n) = b_1 x(n-1) + b_0 x(n) - a_1 y(n-1)$$

$$Y(z) = b_1 z^{-1} X(z) + b_0 X(z) - a_1 z^{-1} Y(z)$$

$$\frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}}$$

By applying the transposition principle we obtain the following block diagram.



Let us verify that it implements the same function. Give a name for each of the delay elements inputs: $w_1(n)$ and $w_2(n)$.

$$y(n) = w_2(n-1) + b_0\left(x(n) + w_1(n-1)\right)$$
$$w_2(n) = b_1\left(x(n) + w_1(n-1)\right)$$
$$w_1(n) = -a_1\left(x(n) + w_1(n-1)\right)$$

Thus,

$$Y(z) = z^{-1}W_2(z) + b_0X(z) + B_0z^{-1}W_1(z),$$
$$W_2(z) = b_1X(z) + b_1z^{-1}W_1(z),$$
$$W_1(z) = -a_1X(z) - a_1z^{-1}W_1(z) = \frac{-a_1}{1+a_1z^{-1}}X(z).$$

Let us express $Y(z)$ as a function of $X(z)$ by successive substitutions:

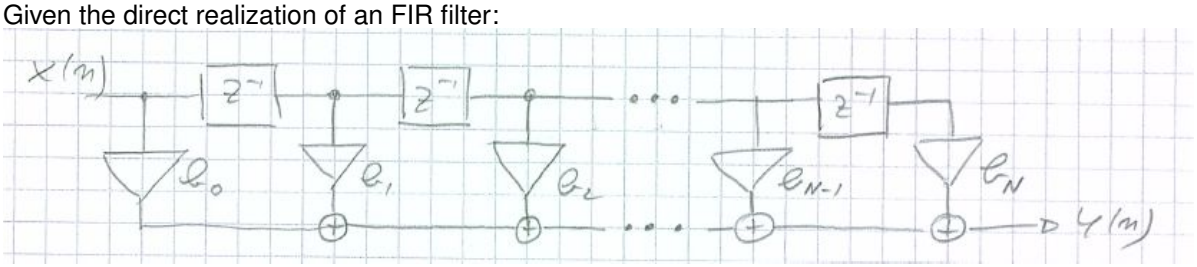
$$W_2(z) = b_1X(z) + b_1z^{-1}\frac{-a_1}{1+a_1z^{-1}}X(z).$$

$$Y(z) = z^{-1}b_1X(z) + b_1z^{-2}\frac{-a_1}{1+a_1z^{-1}}X(z) + b_0X(z) + b_0z^{-1}\frac{-a_1}{1+a_1z^{-1}}X(z) =$$
$$= \frac{z^{-1}b_1[1+a_1z^{-1}] - a_1b_1z^{-2} + b_0[1+a_1z^{-1}] - a_1b_0z^{-1}}{1+a_1z^{-1}} \cdot X(z) =$$
$$= \frac{b_1z^{-1} + b_1a_1z^{-2} - a_1b_1z^{-2} + b_0 + b_0a_1z^{-1} - a_1b_0z^{-1}}{1+a_1z^{-1}} \cdot X(z) =$$
$$= \frac{b_0 + b_1z^{-1}}{1+a_1z^{-1}} \cdot X(z).$$

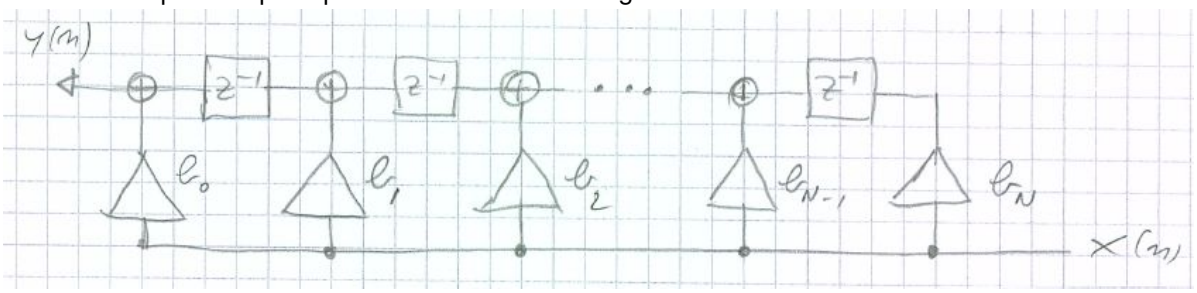
Q.E.D.

Using the transposition principle, we can derive alternative realizations from those we have already seen.

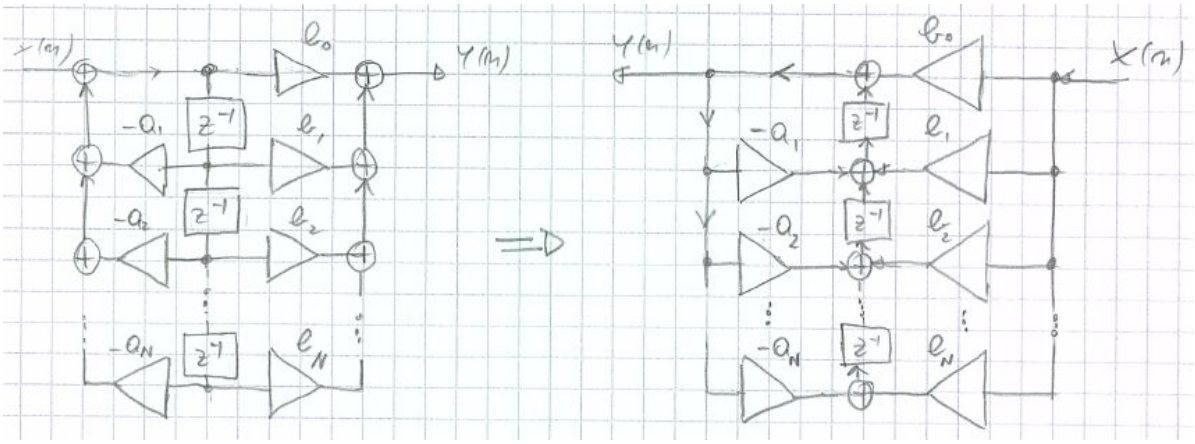
Given the direct realization of an FIR filter:



with the transposition principle we obtain the following realization:

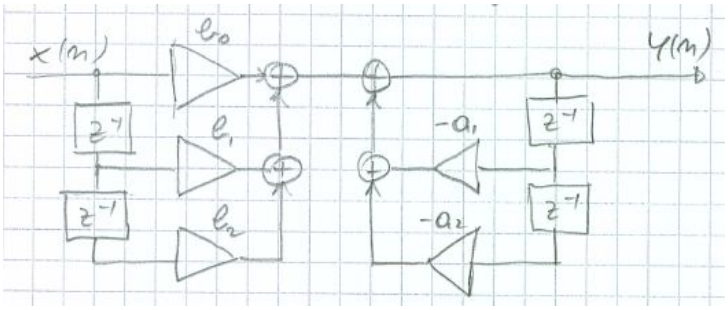


From the direct form II realization of an IIR filter, we obtain the *direct form transposed II* realization.

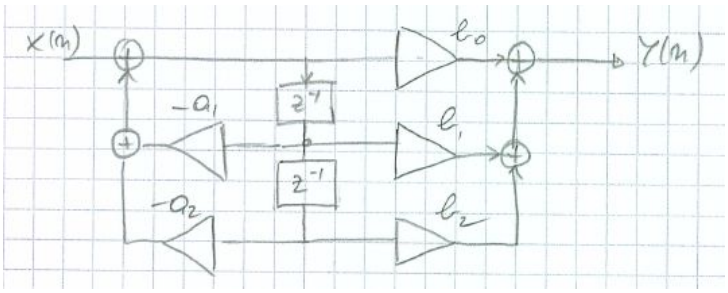


Summarizing, the following realizations are typically used for second-order IIR filters:

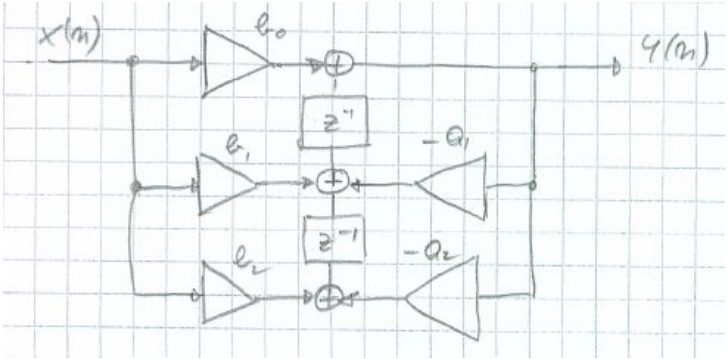
Direct form I:



Direct form II:



Direct form II transposed:



08.03.3 Cascade realization

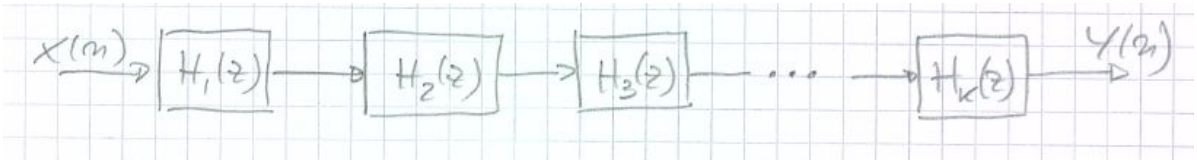
Let's assume $N \geq M$. The IIR transfer function we've seen before can be factorized into the product (or cascade) of second-order transfer functions:

$$H(z) = \prod_{k=0}^{K} H_k(z)$$

where

$$H_k(z) = \frac{b_{k,0} + b_{k,1}z^{-1} + b_{k,2}z^{-2}}{1 + a_{k,1}z^{-1} + a_{k,2}z^{-2}}$$

Note that the coefficient b_0 of $H(z)$ can be distributed among the different sections.
If the system $H(z)$ has real coefficients, then the various $H_k(z)$ can also have real coefficients. When forming the second-order systems, we will group together the complex conjugate poles and zeros. Conversely, real poles and zeros can be grouped in any order.
If $N > M$, some of the subsystems will have $b_{k,2}$ and/or $b_{k,1}$ equal to zero. Moreover, if N is odd, one of the subsystems has $a_{k,2} = 0$ and $b_{k,2} = 0$, meaning it must be of the first order.
Each second-order section can be realized using any of the direct form realizations we have seen before. Poles and zeros can be grouped in any order, and the second-order sections can have any order. Despite all these realizations being equivalent with infinite precision arithmetic, their behavior can differ significantly with finite precision arithmetic.



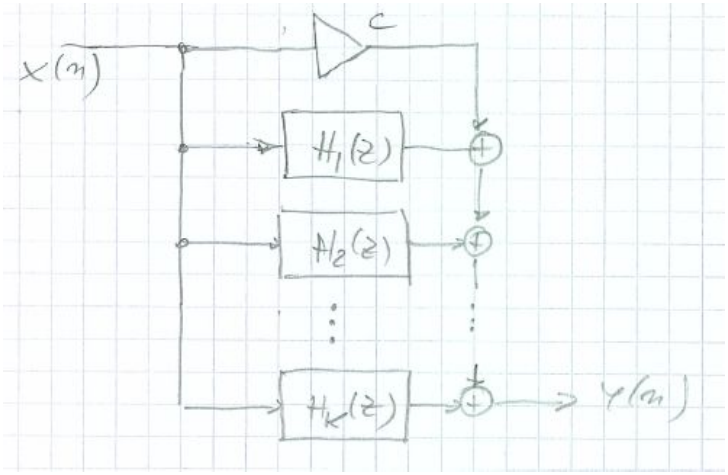
08.03.4 Parallel form realization

Parallel form realizations of IIR filters are derived from the partial fraction expansion of the transfer function. In the case of single poles and $N \geq M$, we have:

$$H(z) = c + \sum_{k=1}^{N} \frac{A_k}{1 - p_k z^{-1}}$$

where p_k are the poles of $H(z)$, A_k are the coefficients of the partial fraction expansion, and $c = \frac{b_N}{a_N}$ (which is non-zero only for $M = N$).
In general, if $H(z)$ has real coefficients, some poles could be complex (forming pairs of complex conjugate poles). In that case, the coefficients A_k will also be complex and appear in conjugate pairs. To simplify computations, we can combine the complex conjugate subsystems to obtain real second-order systems. Additionally, we can combine pairs of branches related to real poles (with any order). Thus, a structure composed of a parallel combination of second-order sections will be obtained. Each second-order section has the following form:

$$H_k(z) = \frac{b_{k,0} + b_{k,1}z^{-1}}{1 + a_{k,1}z^{-1} + a_{k,2}z^{-2}}$$



Example

Let us find the cascade and parallel form realizations of the system described by the transfer function:

$$H(z) = \frac{10\left(1-\frac{1}{2}z^{-1}\right)\left(1-\frac{2}{3}z^{-1}\right)\left(1+2z^{-1}\right)}{\left(1-\frac{3}{4}z^{-1}\right)\left(1-\frac{1}{8}z^{-1}\right)\left(1-(\frac{1}{2}+\frac{1}{2}j)z^{-1}\right)\left(1-(\frac{1}{2}-\frac{1}{2}j)z^{-1}\right)}$$

The cascade realization can indeed be easily obtained from the knowledge of zeros and poles. A possible realization is:

$$H_1(z) = \frac{2\left(1-\frac{1}{2}z^{-1}\right)\left(1-\frac{2}{3}z^{-1}\right)}{\left(1-\frac{3}{4}z^{-1}\right)\left(1-\frac{1}{8}z^{-1}\right)} = \frac{2-\frac{7}{3}z^{-1}+\frac{2}{3}z^{-2}}{1-\frac{7}{8}z^{-1}+\frac{3}{32}z^{-2}}.$$

$$H_2(z) = \frac{5(1+2z^{-1})}{\left(1-(\frac{1}{2}+\frac{1}{2}j)z^{-1}\right)\left(1-(\frac{1}{2}-\frac{1}{2}j)z^{-1}\right)} = \frac{5+10z^{-1}}{1-z^{-1}+\frac{1}{2}z^{-2}}.$$

In order to find the parallel realization we must consider the expansion

$$H(z) = \frac{A_1}{1-\frac{3}{4}z^{-1}} + \frac{A_2}{1-\frac{1}{8}z^{-1}} + \frac{A_3}{1-(\frac{1}{2}+\frac{1}{2}j)z^{-1}} + \frac{A_3^*}{1-(\frac{1}{2}-\frac{1}{2}j)z^{-1}}.$$

$$A_1 = \left.\frac{10\left(1-\frac{1}{2}z^{-1}\right)\left(1-\frac{2}{3}z^{-1}\right)\left(1+2z^{-1}\right)}{\left(1-\frac{1}{8}z^{-1}\right)\left(1-(\frac{1}{2}+\frac{1}{2}j)z^{-1}\right)\left(1-(\frac{1}{2}-\frac{1}{2}j)z^{-1}\right)}\right|_{z=\frac{3}{4}} = 2.93$$

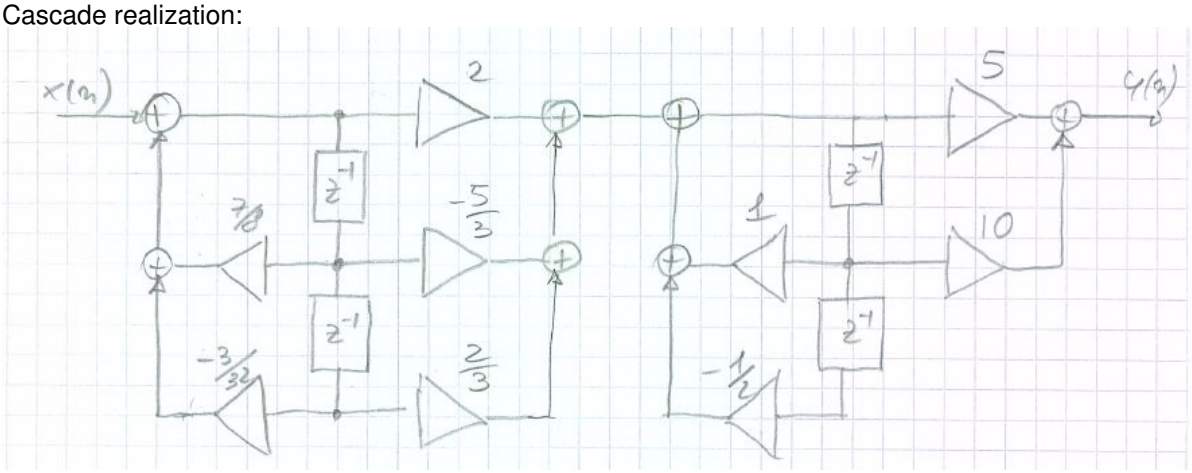
$$A_2 = \left.\frac{10\left(1-\frac{1}{2}z^{-1}\right)\left(1-\frac{2}{3}z^{-1}\right)\left(1+2z^{-1}\right)}{\left(1-\frac{3}{4}z^{-1}\right)\left(1-(\frac{1}{2}+\frac{1}{2}j)z^{-1}\right)\left(1-(\frac{1}{2}-\frac{1}{2}j)z^{-1}\right)}\right|_{z=\frac{1}{8}} = -17.68$$

$$A_3 = \left.\frac{10\left(1-\frac{1}{2}z^{-1}\right)\left(1-\frac{2}{3}z^{-1}\right)\left(1+2z^{-1}\right)}{\left(1-\frac{3}{4}z^{-1}\right)\left(1-\frac{1}{8}z^{-1}\right)\left(1-(\frac{1}{2}-\frac{1}{2}j)z^{-1}\right)}\right|_{z=\frac{1}{2}+\frac{1}{2}j} = 12.25 - j14.57$$

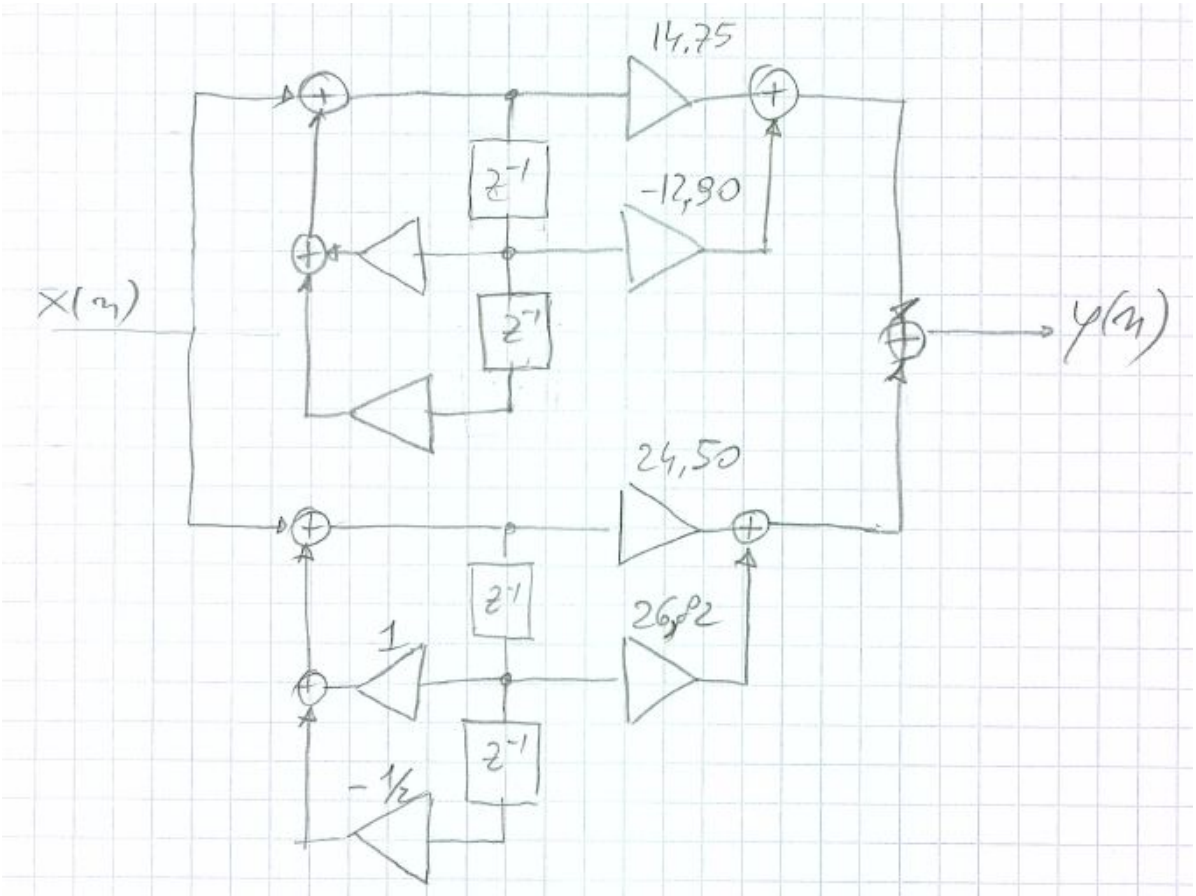
Thus,

$$H(z) = \frac{-14.75-12.90z^{-1}}{1-\frac{7}{8}z^{-1}+\frac{3}{32}z^{-2}} + \frac{24.50+26.82z^{-1}}{1-z^{-1}+\frac{1}{2}z^{-2}}$$

Cascade realization:



Parallel form realization:



08.03.5 Lattice ladder realization

Let's begin by considering the realization of an all-poles system with the transfer function:

$$H(z) = \frac{1}{1 + \sum_{k=1}^{N} a_N(k) z^{-k}} = \frac{1}{D_N(z)}$$

The system is described by the finite difference equation

$$y(n) = -\sum_{k=1}^{N} a_N(k) y(n-k) + x(n).$$

It is interesting to note that if we exchange $x(n)$ and $y(n)$, we obtain the finite difference equation

$$x(n) = -\sum_{k=1}^{N} a_N(k) x(n-k) + y(n),$$

which is:

$$y(n) = x(n) + \sum_{k=1}^{N} a_N(k) x(n-k),$$

the finite difference equation of an FIR filter with transfer function:

$$H'(z) = 1 + \sum_{k=1}^{N} a_N(k) = D_N(z).$$

Supported by this observation, we will employ the lattice structure for FIR filters previously studied in order to derive a lattice structure for IIR filters by exchanging the input and the output. For the FIR filters we had:

$$f_m(n) = f_{m-1}(n) + K_m g_{m-1}(n-1)$$
$$g_m(n) = K_m f_{m-1}(n) + g_{m-1}(n-1)$$

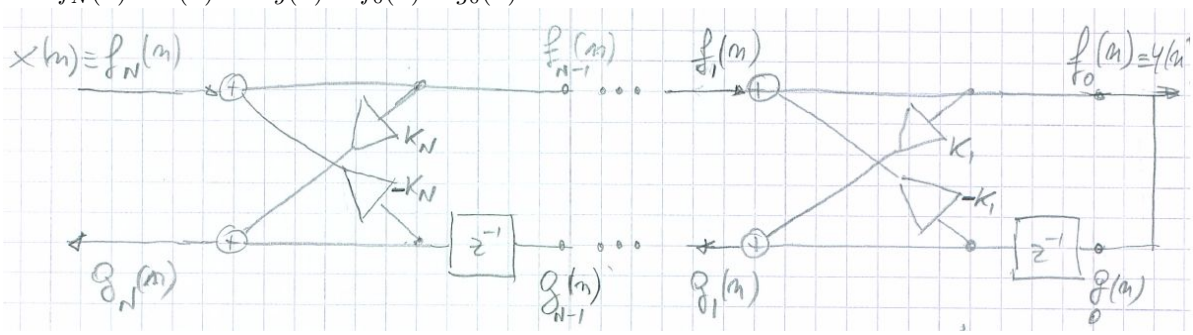
where $f_0(n) = g_0(n) = x(n)$ and $f_N(n) = y(n)$.
For the lattice IIR structure, we consider the same finite difference equations:

$$\begin{cases} f_m(n) = f_{m-1}(n) + K_m g_{m-1}(n-1) \\ g_m(n) = K_m f_{m-1}(n) + g_{m-1}(n-1) \end{cases} \qquad \forall m = 1, \ldots, N$$

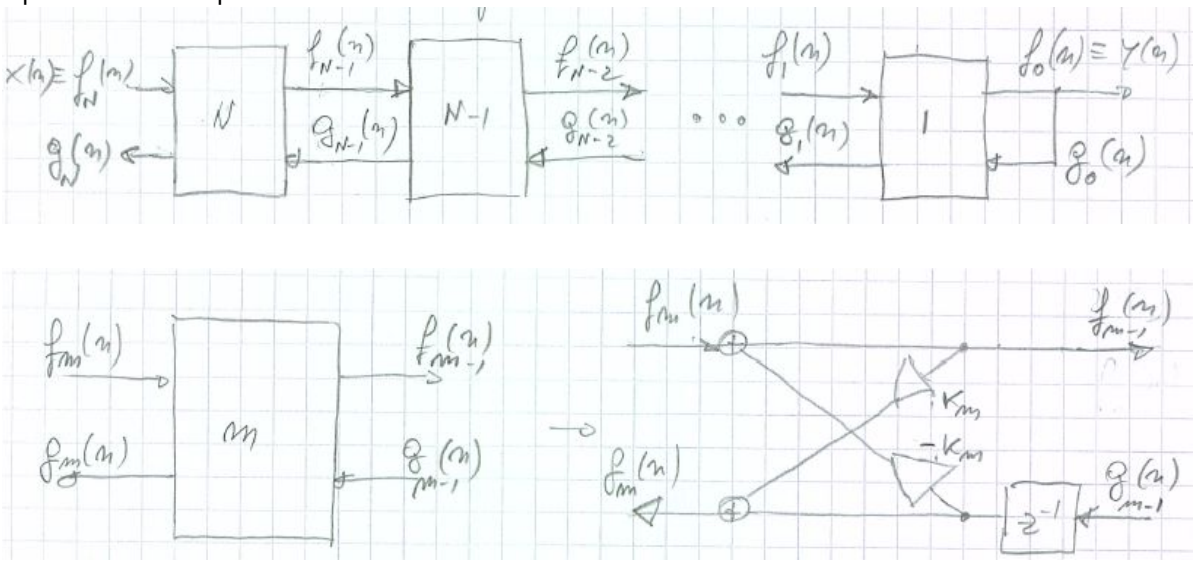
but we take $y(n) = f_0(n) = g_0(n)$ and $f_N(n) = x(n)$. In other words, we consider the system described by:

$$\begin{cases} f_{m-1}(n) = f_m(n) - K_m g_{m-1}(n-1) \\ g_m(n) = K_m f_{m-1}(n) + g_{m-1}(n-1) \end{cases} \quad \forall m = 1, \ldots, N$$

with $f_N(n) = x(n)$ and $y(n) = f_0(n) = g_0(n)$.



Note that now we compute the $f_m(n)$ with decreasing order (for $m = N$ till 0). The structure is again given by the cascade of N equal stages (equal apart from the different factor K_m). Each stage has two inputs and two outputs:



In each block, the inputs are $f_m(n)$ and $g_{m-1}(n)$, while the outputs are $f_{m-1}(n)$ and $g_m(n)$.
Let us verify that these equations describe an all-poles system.
Let us consider first the case of $N = 1$.

$$\begin{cases} f_0(n) = f_1(n) - K_1 g_0(n-1) \\ g_1(n) = K_1 f_0(n) + g_0(n-1) \end{cases}$$

with $f_1(n) = x(n)$ and $y(n) = f_0(n) = g_0(n)$. Now, working in the Z-transform domain:

$$\begin{cases} Y(z) = F_0(z) = F_1(z) - K_1 z^{-1} G_0(z) \\ G_1(z) = K_1 F_0(z) + z^{-1} G_0(z) \end{cases}$$

with $F_1(z) = X(z)$ and $G_0(z) = F_0(z) = Y(z)$. Thus,

$$Y(z) = X(z) - K_1 z^{-1} Y(z)$$

$$\frac{Y(z)}{X(z)} = \frac{1}{1 + K_1 z^{-1}} = \frac{1}{D_1(z)}$$

which is an all-poles transfer function.

$$\frac{G_1(z)}{G_0(z)} = K_1 + z^{-1} = \widetilde{D}_1(z) = z^{-1} D_1(z^{-1}),$$

which is an all-zeros (FIR) transfer function, and $\widetilde{D}_1(z)$ is the mirror image polynomial of $D_1(z)$.

$$\frac{G_1(z)}{X(z)} = \frac{G_1(z)}{F_0(z)} \frac{F_0(z)}{X(z)} = \frac{K_1 + z^{-1}}{1 + K_1 z^{-1}} = \frac{\widetilde{D}_1(z)}{D_1(z)}$$

and it is an all-pass transfer function.

These three properties hold in general. I.e., given the cascade of m stages with $y(n) = f_0(n) = g_0(n)$ and $f_m(n) = x(n)$,

1. $\frac{Y(z)}{X(z)} = \frac{F_0(z)}{F_m(z)} = \frac{1}{D_m(z)}$.
2. $\frac{G_m(z)}{G_0(z)} = \widetilde{D}_m(z)$.
3. $\frac{G_m(z)}{X(z)} = \frac{G_m(z)}{F_m(z)} = \frac{\widetilde{D}_m(z)}{D_m(z)}$.

Note that according to the second property, $g_m(n)$ is an FIR function, i.e., a linear combination, of the past outputs $y(n)$.

Let us prove these properties by induction. Let us assume:

$$\frac{F_0(z)}{F_{m-1}(z)} = \frac{1}{D_{m-1}(z)}$$

$$\frac{G_{m-1}(z)}{G_0(z)} = \widetilde{D}_{m-1}(z)$$

$$\frac{G_{m-1}(z)}{F_{m-1}(z)} = \frac{\widetilde{D}_{m-1}(z)}{D_{m-1}(z)}$$

and

$$\begin{cases} F_{m-1}(z) = F_m(z) - K_m z^{-1} G_{m-1}(z) \\ G_m(z) = K_m F_{m-1}(z) + z^{-1} G_{m-1}(z) \end{cases}.$$

Let us first prove that $\frac{F_0(z)}{F_m(z)} = \frac{1}{D_m(z)}$. But

$$\frac{F_0(z)}{F_m(z)} = \frac{F_0(z)}{F_{m-1}(z)} \cdot \frac{F_{m-1}(z)}{F_m(z)}$$

From $F_{m-1}(z) = F_m(z) - K_m z^{-1} G_{m-1}(z)$, we have

$$\frac{F_m(z)}{F_{m-1}(z)} = 1 + K_m z^{-1} \frac{G_{m-1}(z)}{F_{m-1}(z)} = 1 + K_m z^{-1} \frac{\widetilde{D}_{m-1}(z)}{D_{m-1}(z)} = \frac{D_{m-1}(z) + K_m z^{-1} \widetilde{D}_{m-1}(z)}{D_{m-1}(z)}$$

$$\frac{F_0(z)}{F_m(z)} = \frac{1}{D_{m-1}(z)} \cdot \frac{D_{m-1}(z)}{D_{m-1}(z) + K_m z^{-1} \widetilde{D}_{m-1}(z)} = \frac{1}{D_{m-1}(z) + K_m z^{-1} \widetilde{D}_{m-1}(z)} = \frac{1}{D_m(z)}$$

with $D_m(z) = D_{m-1}(z) + K_m z^{-1} \widetilde{D}_{m-1}(z)$.

Let us verify next that $\frac{G_m(z)}{G_0(z)} = \widetilde{D}_m(z)$. Since $G_m(z) = K_m F_{m-1}(z) + z^{-1} G_{m-1}(z)$, we have

$$\frac{G_m(z)}{G_0(z)} = K_m \frac{F_{m-1}(z)}{G_0(z)} + z^{-1} \frac{G_{m-1}(z)}{G_0(z)} =$$

$$= K_m \frac{F_{m-1}(z)}{F_0(z)} + z^{-1} \frac{G_{m-1}(z)}{G_0(z)} =$$

$$= K_m D_{m-1}(z) + z^{-1} \widetilde{D}_{m-1}(z) = \widetilde{D}_m(z).$$

$\widetilde{D}_m(z)$ is the mirror image polynomial of $D_m(z)$ since:

$$z^{-m} D_m(z^{-1}) = z^{-m} [D_{m-1}(z^{-1}) + K_m z \widetilde{D}_{m-1}(z^{-1})] =$$

$$= z^{-1} z^{-(m-1)} D_{m-1}(z^{-1}) + K_m z^{-(m-1)} \widetilde{D}_{m-1}(z^{-1}) =$$

$$= z^{-1} \widetilde{D}_{m-1}(z) + K_m D_{m-1}(z) = \widetilde{D}_m(z).$$

Eventually, for the last property, we have:

$$\frac{G_m(z)}{F_m(z)} = \frac{G_m(z)}{G_0(z)} \cdot \frac{F_0(z)}{F_m(z)} = \frac{\widetilde{D}_m(z)}{D_m(z)}.$$

Therefore, these properties hold for every m from 1 till N.

It's noteworthy that, given the polynomial $D_N(z)$, both the lattice FIR structure and the lattice IIR structure share the same set of reflection parameters $K_1, K_2, \ldots, K_N$. These reflection coefficients are determined by the polynomial $D_N(z)$. In the lattice FIR filter, $D_N(z)$ is the transfer function, whereas in the lattice IIR filter, it is the denominator of the transfer function.

The algorithms we have covered for FIR filters, used to compute the reflection coefficients from $D_N(z)$ (and to compute $D_N(z)$ from the reflection coefficients) are applicable to IIR lattice filters as well. It's essential to recall that the roots of $D_N(z)$ lie within the unit circle if and only if $K_1, K_2, \ldots, K_N$ are $|K_m| < 1 \; \forall m$. Consequently, the lattice IIR filter maintains stability if and only if the reflection coefficients satisfy $|K_m| < 1 \; \forall m$.

The all-poles lattice filter serves as the foundation for constructing IIR lattice filters with poles and zeros:

$$H(z) = \frac{\sum_{k=0}^{N} c_N(k) z^{-k}}{1 - \sum_{k=0}^{N} a_N(k) z^{-k}} = \frac{C_N(z)}{D_N(z)}$$

When considering the direct form II realization, we observe that it consists of an all-poles filter cascaded with an all-zeros filter. The all-poles filter is described by

$$w(n) = -\sum_{k=1}^{N} a_N(k) w(n-1) + x(n).$$

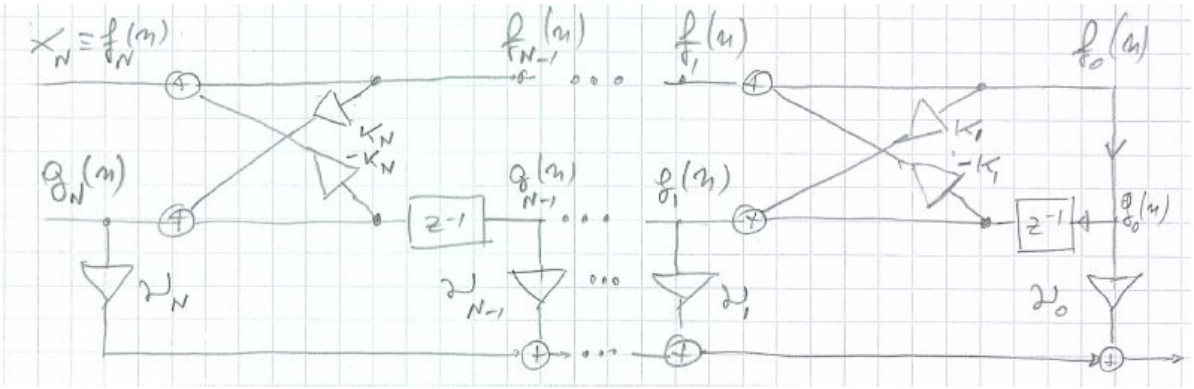
The all-zeros filter is described by

$$y(n) = \sum_{k=0}^{N} c_N(k) w(n-k).$$

Therefore, the output of the IIR filter $\dfrac{C_N(z)}{D_N(z)}$ can be obtained from the linear combination of N past outputs of the all-pole filter $\dfrac{1}{D_N(z)}$. As we've previously observed, the terms $g_m(n)$ in our lattice filter are a linear combination of the past outputs of the system. Thus, to implement a generic IIR transfer function with a lattice filter, we construct the lattice structure that implements $\dfrac{1}{D_N(z)}$ and form an appropriate linear combination of the terms $\{g_m(n)\}$. Specifically, we consider

$$y(n) = \sum_{m=0}^{N} \nu_m g_m(n),$$

where the parameters ν_m form the zeros of $H(z)$.



This structure is referred to as the *Lattice-Ladder* realization.

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{m=0}^{N} \nu_m \frac{G_m(z)}{X(z)} =$$

$$= \sum_{m=0}^{N} \nu_m \frac{G_m(z)}{G_0(z)} \cdot \frac{F_0(z)}{F_N(z)} =$$

$$= \sum_{m=0}^{N} \nu_m \widetilde{D}_m(z) \frac{1}{D_N(z)} =$$

$$= \frac{\sum_{m=0}^{N} \nu_m \widetilde{D}_m(z)}{D_N(z)}$$

Thus, we need to find the coefficients ν_m such that

$$\sum_{m=0}^{N} \nu_m \widetilde{D}_m(z) = C_N(z)$$

However, we have

$$C_N(z) = \sum_{m=0}^{N-1} \nu_m \widetilde{D}_m(z) + \nu_N \widetilde{D}_N(z) =$$

$$= C_{N-1}(z) + \nu_N \widetilde{D}_N(z)$$

where $C_{N-1}(z)$ is an order $N-1$ polynomial. Generally, if we define

$$C_m(z) = \sum_{k=0}^{m} \nu_k \widetilde{D}_k(z)$$

we have

$$C_m(z) = C_{m-1}(z) + \nu_m \widetilde{D}_m(z).$$

The polynomials can be computed recursively from the polynomials $\widetilde{D}_m(z)$, with $m = N, \ldots, 1$. Since the leading coefficient of $\widetilde{D}_m(z)$ is 1, the parameters ν_m are given by

$$\nu_m = c_m(m) \qquad \forall m = N, \ldots, 0$$

i.e., the coefficient of the highest degree monomial of $C_m(z)$. We can recursively compute the coefficients ν_m with the following algorithm:

For $m = N$ till 0

$$\nu_m = [c_m(z) \cdot z^m]\Big|_{z=0}$$

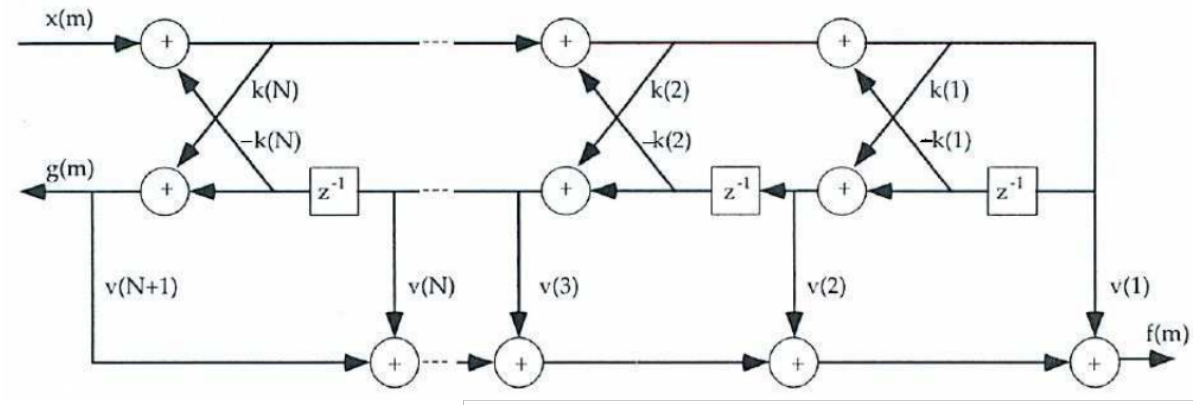
$$c_{m-1}(z) = c_m(z) - \nu_m \widetilde{D}_m(z)$$

End For

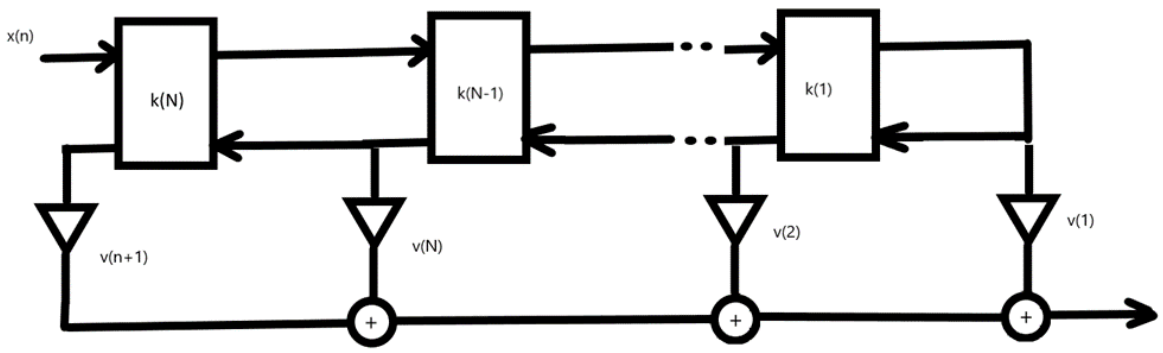
The modularity of the lattice filters, the stability characteristics related to the reflection coefficients, and the robustness in the presence of finite precision arithmetic make these filters very appealing for many applications.
In Matlab, the coefficients of the lattice filter can be obtained with the following function

```
[K,V] = tf2latc(B,A)
```

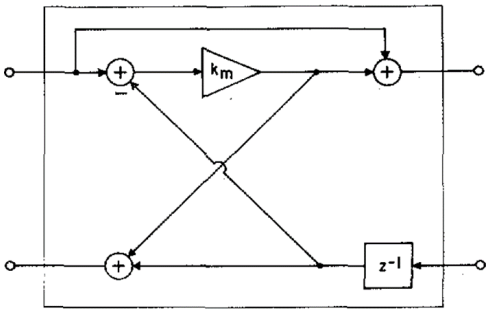
Here, `B` and `A` represent the vectors containing the coefficients of the numerator and denominator polynomials, respectively. In Matlab, elements of vectors are indexed starting from 1. The following figure illustrates the correspondence between coefficients and multipliers:



Enhanced lattice structures have been proposed in the literature. Like before, they consist of a cascade of N identical blocks, with a ladder utilized to implement the transfer function numerator. The reflection coefficients remain consistent with those of the original lattice filter. The alterations lie in the block's implementation and the coefficients of the ladder segment, which nonetheless maintain a relationship with those of the original lattice filter



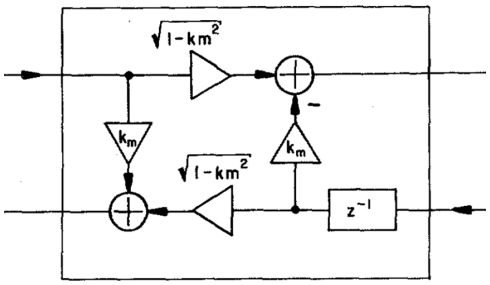
One solution makes use of a single multiplier per block:



The coefficient of the ladder part in Matlab can be obtained as follows:

```
[K,V] = tf2latc(B,A)
for (i=length(V)-1:-1:1)
   V(1:i) = V(1:i)/(1+K(i));
end
```

This code iteratively adjusts the coefficients of the ladder segment based on the obtained reflection coefficients and multipliers.
A computationally more expensive solution, employing four multipliers per block but exhibiting improved behavior when implemented with fixed-point arithmetic, is the normalized lattice structure with the following block:



It offers the notable feature of computing a rotation from the input to the output, making it more computationally robust.
Once more, the coefficients of the ladder part in Matlab can be acquired through the following procedure:

```
[K,V] = tf2latc(B,A)
for (i=length(V)-1:-1:1)
   V(1:i) = V(1:i)/sqrt(1-K(i)^2);
end
```

This code snippet calculates the coefficients of the ladder segment using the obtained reflection coefficients and ensures normalization, thus enhancing the computational robustness of the structure.

For more information study:

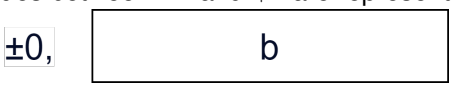
S. K. Mitra, ”Digital Signal Processing: a computer based approach,” 4th edition, McGraw-Hill, 2011
Chapter 8.3.1, pp. 422

Chapter 8.3.4, pp. 426
Chapter 8.3.2, pp. 423-424
Chapter 8.2, pp. 421-422
Chapter 8.4.1, pp. 427-429
Chapter 8.4.2-3, pp. 429-433
Chapter 8.8, pp. 447-452

09 Finite precision arithmetic effects

09.01 Finite precision arithmetic

We have studied various structures for the implementation of digital filters. These structures are equivalent to each other when we operate with infinite precision arithmetic, but they exhibit different properties with finite precision arithmetic.
DSP systems are typically implemented using floating-point or fixed-point arithmetic.
In floating-point arithmetic, the number x is represented using an exponent γ and a mantissa m, such that $0.5 < |m| < 1$ and $x = m \cdot 2^{\gamma}$.
In fixed-point arithmetic, the number x is assumed to belong to a certain interval and is represented with a fixed number of bits, with the decimal point occupying a fixed position within these bits. For example, values between -1 and $+1$ are represented using 1 sign bit and b bits after the decimal point.



The smaller the value of $|x|$, the fewer significant bits are available in the representation of x (thus resulting in a larger relative error in the representation).
When using floating-point arithmetic with a high number of binary digits, we can reach conditions similar to infinite precision arithmetic. However, in cases where finite precision arithmetic is employed (common in hardware or software DSP implementations), it's essential to account for the effects of finite precision. In particular, we need to consider:

1. The effects of quantization of filter coefficients,
2. the A/D conversion noise,
3. the uncorrelated noise due to rounding or truncation during multiplications,
4. the overflow in additions,
5. the limit cycles.

09.02 Filter coefficients quantization

Filter design is typically conducted using infinite precision arithmetic or arithmetic with very high precision. However, in practical implementation, we must quantize the filter coefficients. As a result, the implemented filter no longer matches the transfer function we designed but rather approximates it to

varying degrees, depending on the precision of the arithmetic used (i.e., the number of bits in the coefficient representation) and the sensitivity of the realization to coefficient variations. From this perspective, the direct-form structure of IIR filters exhibits poor behavior, as coefficient quantization can easily lead to system instability. Conversely, lattice filters demonstrate better robustness to coefficient quantization. In situations where quantization involves a very limited number of bits, it becomes necessary to adopt structures highly robust to quantization. Moreover, it is essential to appropriately select the quantized parameters to meet the filter specifications; in this case, the quantized coefficients are directly designed.

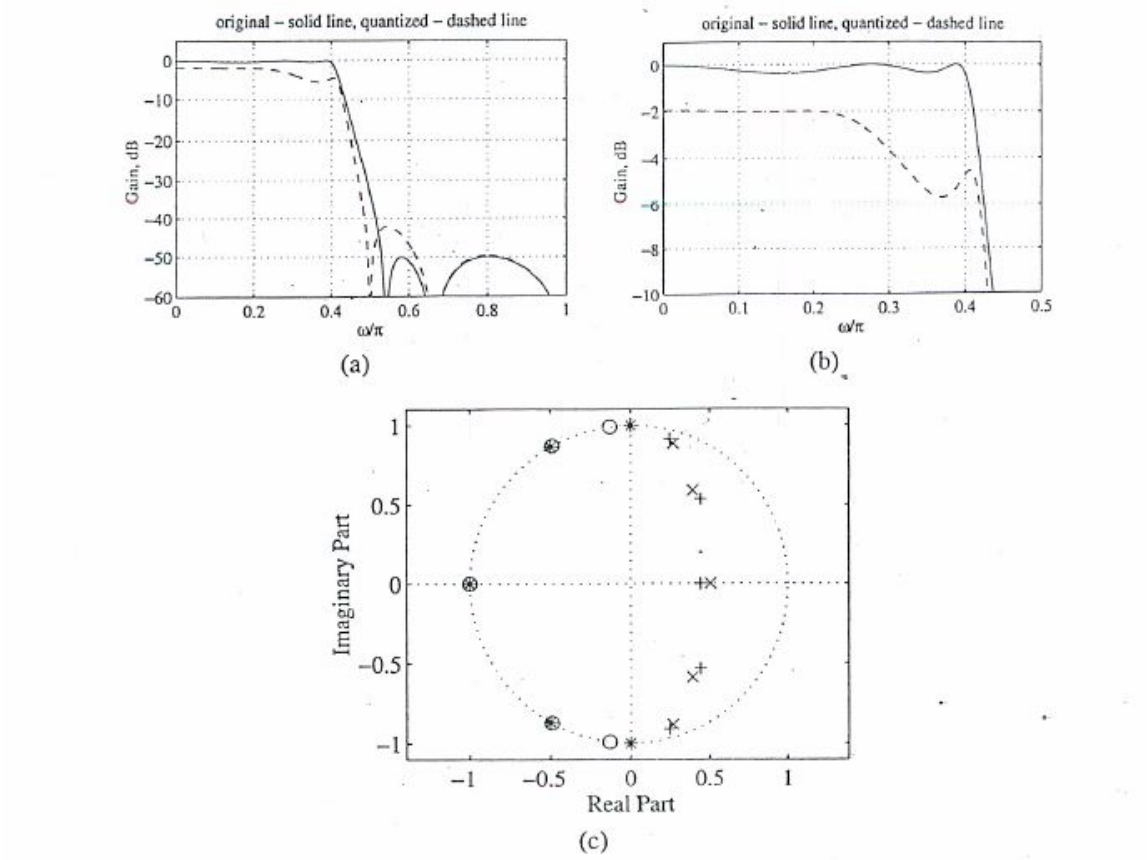


Figure 9.6: Coefficient quantization effects on a fifth-order IIR elliptic lowpass filter implemented in direct form: (a) fullband gain responses with unquantized (shown with solid line) and quantized coefficients (shown with dashed line), (b) passband details, and (c) pole-zero movements: Pole and zero locations of the filter with quantized coefficients denoted by "x" and "c", respectively, and pole and zero locations of the filter with unquantized coefficients denoted by "+" and "*", respectively.

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

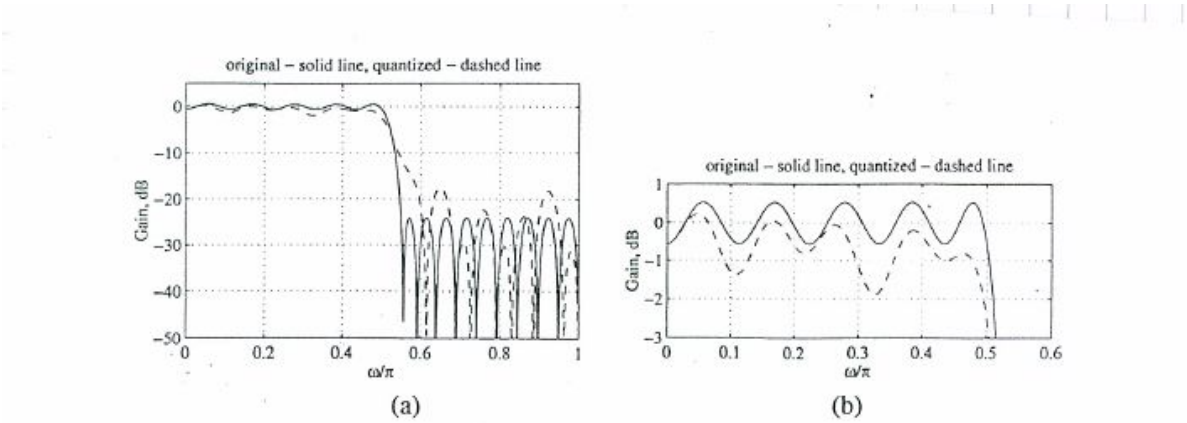
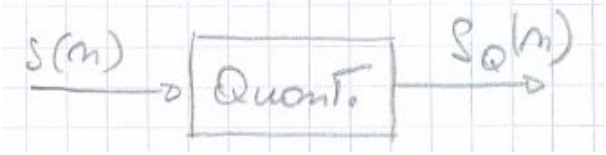


Figure 9.8: Coefficient quantization effects on a 39th-order FIR equiripple lowpass filter implemented in direct form: (a) fullband gain responses with unquantized (shown with solid line) and quantized coefficients (shown with dashed line), and (b) passband details.

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

09.03 A/D conversion noise

The process of analog-to-digital conversion inevitably introduces an error due to signal quantization.

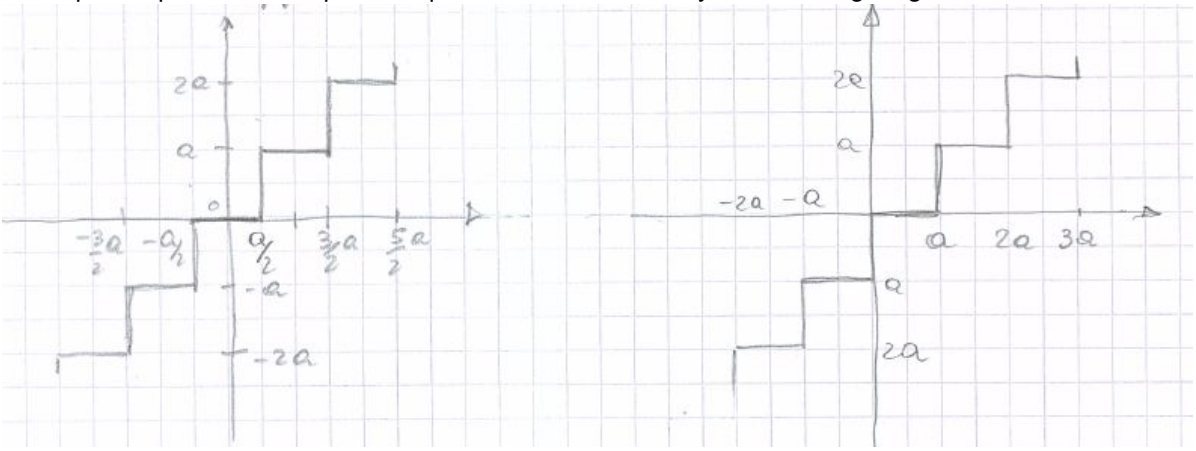


$e(n) = s_Q(n) - s(n)$.

The error introduced by analog-to-digital conversion, called *granular noise*, depends on the number of quantization levels and the type of quantization performed. The signal-to-quantization-noise ratio is influenced by the number of quantization levels.

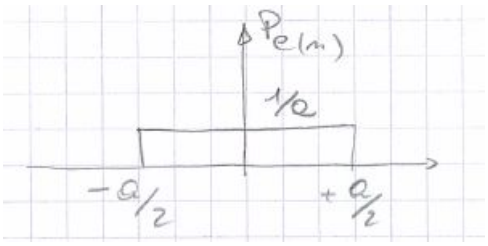
The A/D conversion noise is analyzed using a statistical approach.

The input-output relationship of the quantizer is illustrated by the following diagrams:

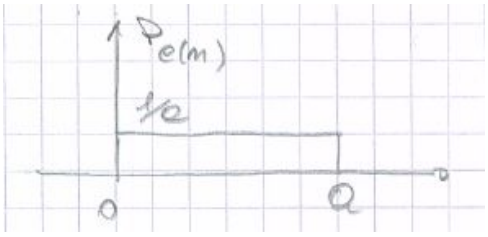


Rounding Truncation

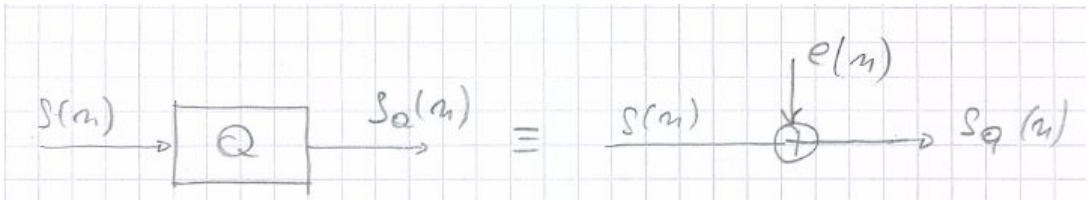
In case of rounding: $-\frac{1}{2}a \leq e(n) < \frac{1}{2}a$.
In case of truncation: $0 \leq e(n) < a$.
Obviously, these expressions assume that there is no saturation of the analog-to-digital converter, i.e., no clipping of the input signal; otherwise, the error can be much larger than a. We must as much as possible avoid saturations.
The quantization errors can be interpreted as generated by a random process akin to white stationary noise, with a constant probability density within the interval $[-\frac{a}{2}, +\frac{a}{2}]$ or $[0, a]$. It is assumed that this noise is uncorrelated with the signal.
For rounding, we have:



$m_e = E[e(n)] = 0$.
$\sigma_e^2 = E[(e(n) - m_e)^2] = \int_{-\frac{a}{2}}^{+\frac{a}{2}} x^2 \frac{1}{a} \mathrm{d}x = \frac{a^2}{12}$.
For truncation, we have:



$m_e = E[e(n)] = \frac{a}{2}$.
$\sigma_e^2 = E[(e(n) - m_e)^2] = \frac{a^2}{12}$.



To evaluate the impact of additive quantization noise on the input signal $s(n)$, we can compute the Signal-to-Quantization-Noise Ratio (SNR) in decibels (dB), which is defined as:

$$\text{SNR}_{\text{A/D}} = 10 \log_{10} \left(\frac{\sigma_s^2}{\sigma_e^2} \right) \text{dB}.$$

Here, σ_s^2 represents the input signal variance (i.e., the power of the input signal), and σ_e^2 denotes the variance of the quantization noise (i.e., the power of the quantization noise).

Consider a quantization scheme with 2^{b+1} levels between $-M$ and $+M$. In this case:

$$a = \frac{2M}{2^{b+1}} = \frac{M}{2^b} \quad \Longrightarrow \quad \sigma_e^2 = \frac{M^2}{12 \cdot 2^{2b}}.$$

This leads to the Signal-to-Quantization-Noise Ratio as:

$$\text{SNR}_{\text{A/D}} = 10\log_{10}\left(\frac{12 \cdot 2^{2b} \cdot \sigma_s^2}{M^2}\right) = 6.02\,b + 10.79 + 10\log_{10}\left(\frac{\sigma_s^2}{M^2}\right) \text{dB}.$$

This equation can help us determine the minimum number of bits b required to achieve a specific signal-to-noise ratio. Notably, we observe that the SNR increases by 6 dB for each additional unit of b.
Is it always possible to prevent saturation of the analog-to-digital converter? In most cases, our signals are random signals with a certain distribution, and there exists a non-zero probability of exceeding a predefined range. Suppose our input signal follows a Gaussian distribution with a zero mean and standard deviation σ_s. The likelihood of a particular analog sample remaining within the range $[-\sigma_s K, +\sigma_s K]$ is given by:

$$2\Phi(K) - 1 = \sqrt{\frac{2}{\pi}} \int_0^K e^{-y^2/2} \mathrm{d}y, \tag{09.01}$$

where $\Phi(.)$ is the cumulative probability of a zero mean unit variance Gaussian distribution.
For $K = 2$, the probability of an analog sample remaining within the range $[-2\sigma_s, +2\sigma_s]$ is 0.9544. This indicates that, on average, out of every $10,000$ input samples, approximately 456 samples fall outside this range.
For $K = 3$, the probability of an analog sample remaining within the range $[-3\sigma_s, +3\sigma_s]$ is 0.9973. This suggests that, on average, out of every $10,000$ input samples, approximately 37 samples fall outside this range.
For $K = 4$, the probability of an analog sample remaining within the range $[-4\sigma_s, +4\sigma_s]$ is 0.999936. This suggests that, on average, out of every $1,000,000$ input samples, approximately 64 samples fall outside this range. This range is generally considered more than sufficient to prevent clipping during conversion.
To prevent saturation of the analog-to-digital converter, scaling the input signal is a common approach. Let's explore its impact. Assuming we scale the input signal by a factor A, the variance of the scaled input, $As(n)$, becomes $A^2\sigma_s^2$. Consequently, the expression for signal-to-quantization noise ratio (SNR) transforms to:

$$\text{SNR}_{\text{A/D}} = 10\log_{10}\left(\frac{12 \cdot 2^{2b} \cdot A^2 \cdot \sigma_s^2}{M^2}\right) = 6.02\,b + 10.79 + 10\log_{10}\left(\frac{\sigma_s^2}{M^2}\right) + 20\log_{10}(A).$$

When $A > 1$, the SNR increases, but so does the probability of overflow. Conversely, $A < 1$ reduces the probability of overflow but also lowers the SNR. To attain the highest possible SNR without distorting the signal, it's essential to align the analog sample range as closely as possible with the full-scale range of the A/D converter.
The quantized signal is commonly processed using a linear time-invariant discrete-time system represented by $H(z)$. By modeling the quantized signal as the sum of the unquantized signal $s(n)$ and a quantization error $e(n)$, and assuming linearity along with uncorrelation between $s(n)$ and $e(n)$, the output comprises two components: $y(n)$ associated with the unquantized input $s(n)$, and $v(n)$ associated

with the quantization error $e(n)$. While $e(n)$ can be assumed to exhibit characteristics of white noise, $v(n)$ does not. Specifically,

$$v(n) = \sum_{m=-\infty}^{+\infty} h(m)e(n-m),$$

where the quantization noise is convolved with the impulse response of the filter. The mean of the output noise $v(n)$ is

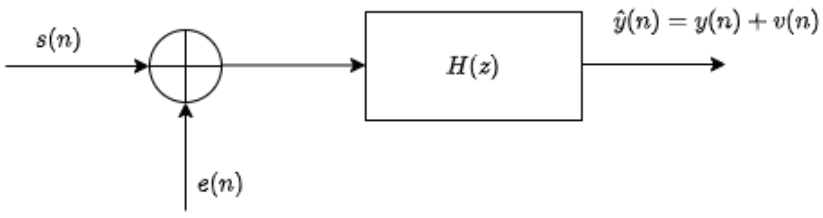
$$m_v = m_e H(e^{j0})$$

and the variance σ_v^2 is

$$\sigma_v^2 = \frac{\sigma_e^2}{2\pi} \int_{-\pi}^{+\pi} |H(e^{j\omega})|^2 \mathrm{d}\omega = \sigma_e^2 \sum_{m=-\infty}^{+\infty} |h(m)|^2.$$

The noise output power spectrum is given by

$$P_{vv}(\omega) = \sigma_e^2 |H(e^{j\omega})|^2.$$

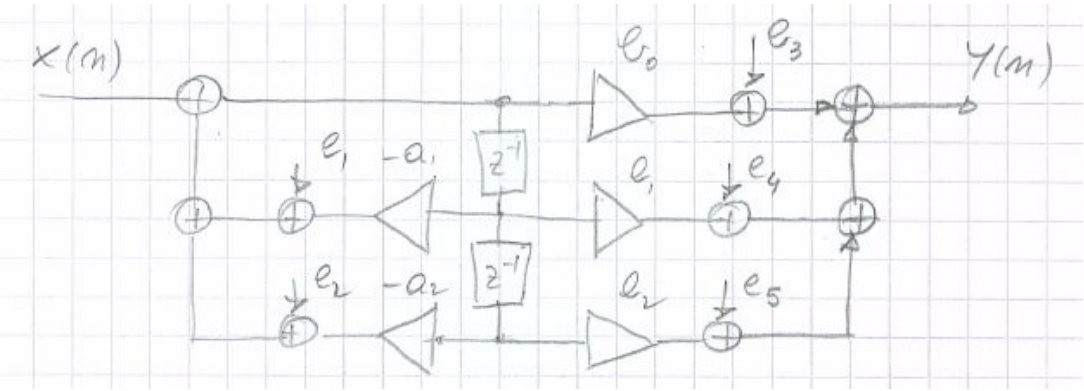


09.04 Uncorrelated noise due to rounding or truncation in multiplications

This noise is analyzed similarly to quantization noise.

When two numbers with N digits are multiplied, the result typically has $2N-1$ digits. To represent this result with N digits, rounding or truncation is necessary. This rounding or truncation process introduces noise, which is statistically treated similarly to quantization noise. The noise is assumed to originate from a white stationary noise source with a constant probability density distribution. It's worth noting that these various noise sources are assumed to be uncorrelated with each other and with the input signal.

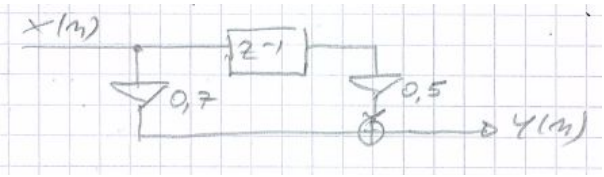
For instance, in the case of a second-order Infinite Impulse Response (IIR) filter, there exists a noise source associated with each multiplier operation.



From the selection of the quantization level, it becomes feasible to ascertain the output signal-to-noise ratio resulting from rounding or truncation.
It's worth noting that different filter structures generally exhibit distinct noise propagation properties.

09.05 Overflow in additions

This issue is especially significant when operating with fixed-point arithmetic.
When dealing with numbers between 0 and 1 in fixed-point representation, the sum of two such numbers may exceed 1, leading to overflow beyond the representation limits (assuming the maximum value is 1). Overflow typically manifests as an impulsive noise with high amplitude at the output of the system. Therefore, it's essential to design the system in advance to minimize the occurrence of overflows or to ensure that they have minimal probability of happening.
To prevent overflows, it is crucial to appropriately scale all signals using constant multiplicative factors.
For example:



If the input signal satisfies $-1 \leq x(n) \leq +1$, then after processing, the output signal $y(n)$ may range from -1.2 to 1.2, posing a risk of overflow when using fixed-point arithmetic with a maximum value of 1. However, by multiplying the input signal $x(n)$ by $\frac{1}{1.2}$, we obtain an output signal that is always bounded between -1 and 1. This scaled output signal is identical to the original signal but scaled by a factor of $\frac{1}{1.2}$. This scaling ensures that the output remains within the representable range without the risk of overflow.
While completely avoiding the overflow problem by appropriately scaling the input signal is indeed possible, it often comes at the cost of significantly penalizing the signal-to-quantization-noise ratio, as it reduces σ_s^2.

In general, we can minimize the probability of overflow at internal nodes by inserting scaling multipliers at selected points in the digital filter structure to appropriately scale the internal signal levels. In many cases, most of these scaling multipliers can be absorbed by the existing multipliers in the structure.

09.06 Dynamic range scaling

Let's consider a digital filter structure, and for a given node r, let the signal to be scaled be denoted as $u_r(n)$. We assume that all fixed-point numbers are represented as binary fractions and that the input sequence $x(n)$ is bounded by unity, i.e., $|x(n)| \leq 1$ for all n. The objective of scaling is to ensure that

$$|u_r(n)| \leq 1$$

for all nodes r and time instances n.

09.06.1 An absolute bound

Let's define the *scaling transfer function* $F_r(z)$ as the transfer function from the input to the r-th node. Its inverse z-transform $f_r(n)$ represents the impulse response from the filter input to the r-th node. Therefore, $u_r(n)$ can be expressed as the convolution of $f_r(n)$ and $x(n)$:

$$u_r(n) = \sum_{k=-\infty}^{+\infty} f_r(k)x(n-k)$$

Then,

$$|u_r(n)| = \left| \sum_{k=-\infty}^{+\infty} f_r(k)x(n-k) \right| \leq \sum_{k=-\infty}^{+\infty} |f_r(n)|.$$

Thus, we have $|u_r(n)| \leq 1$ if

$$\sum_{k=-\infty}^{+\infty} |f_r(n)| \leq 1.$$

It can be proven that the above condition is both necessary and sufficient to guarantee no overflow. If it is not satisfied by the unscaled realization, we can scale the input signal with a multiplier K, where

$$K = \frac{1}{\max_r \sum_{k=-\infty}^{+\infty} |f_r(n)|}.$$

This scaling rule is based on a worst-case bound and significantly reduces the output SNR. More practical and easy-to-use scaling rules can be derived in the frequency domain if some information about the input signals is known a priori. In what follows, we will assume the input $x(n)$ to be a deterministic signal with Fourier transform $X(e^{j\omega})$, and we will derive the bounds in terms of $\mathcal{L}_p$-norms.
The $\mathcal{L}_p$-norm of $X(e^{j\omega})$ is defined as

$$||X||_p = \left(\frac{1}{2\pi} \int_{-\pi}^{+\pi} |X(e^{j\omega})|^p \mathrm{d}\omega \right)^{1/p}$$

In most cases, the values of p used are 1, 2, and ∞. For $p = 2$, $||X||_2$ represents the root-mean-square (rms) value of $X(e^{j\omega})$. For $p = 1$, $||X||_1$ indicates the mean absolute value of $X(e^{j\omega})$. In the case of a continuous $X(e^{j\omega})$, $\lim_{p\to\infty} ||X||_p$ exists and represents the peak absolute value

$$||X||_\infty = \max_{-\pi\leq\omega\leq+\pi} |X(e^{j\omega})|.$$

09.06.2 $\mathcal{L}_\infty$-bound

Since

$$u_r(n) = \frac{1}{2\pi}\int_{-\pi}^{+\pi} F_r(e^{j\omega})X(e^{j\omega})e^{j\omega n}\mathrm{d}\omega,$$

we have

$$\begin{aligned}|u_r(n)| &\leq \frac{1}{2\pi}\int_{-\pi}^{+\pi} |F_r(e^{j\omega})||X(e^{j\omega})|\mathrm{d}\omega\\ &\leq ||F_r||_\infty \frac{1}{2\pi}\int_{-\pi}^{+\pi} |X(e^{j\omega})|\mathrm{d}\omega\\ &\leq ||F_r||_\infty \cdot ||X||_1.\end{aligned}$$

If $||X||_1 \leq 1$, then $|u_r(n)| \leq 1$ if

$$||F_r||_\infty \leq 1.$$

In general, this scaling rule is rarely used since, in practice, $||X||_1 \leq 1$ does not typically hold for most encountered input signals.

09.06.3 $\mathcal{L}_2$-bound

Applying the Schwartz inequality to

$$u_r(n) = \frac{1}{2\pi}\int_{-\pi}^{+\pi} F_r(e^{j\omega})X(e^{j\omega})e^{j\omega n}\mathrm{d}\omega,$$

we obtain that

$$|u_r(n)|^2 \leq \left(\frac{1}{2\pi}\int_{-\pi}^{+\pi} |F_r(e^{j\omega})|^2\mathrm{d}\omega,\right)\cdot\left(\frac{1}{2\pi}\int_{-\pi}^{+\pi} |X(e^{j\omega})|^2\mathrm{d}\omega,\right),$$

i.e.,

$$|u_r(n)|^2 \leq ||F_r||_2 \cdot ||X||_2.$$

If the input signal has finite energy bounded by unity, i.e., $||X||_2 \leq 1$, then preventing adder overflow can be achieved by scaling the filter such that the $\mathcal{L}_2$-norm of the transfer functions from the input to all adder outputs are bounded by unity:

$$||F_r||_2 \leq 1 \quad \forall r.$$

09.06.4 A more general scaling rule

According to the Holder's inequality,

$$|u_r(n)|^2 \leq ||F_r||_p \cdot ||X||_q,$$

for all $p, q \geq 1$ satisfying $(1/p) + (1/q) = 1$. The $\mathcal{L}_\infty$-bound is obtained for $p = \infty$ and $q = 1$. The $\mathcal{L}_2$-bound is obtained for $p = q = 2$. Another useful bound is the $\mathcal{L}_1$-bound obtained for $p = 1$ and $q = \infty$.

Provided $||X||_q < 1$, then $|u_r(n)| \leq 1$ if

$$||F_r||_p \leq 1.$$

In many structures, all scaling multipliers can be absorbed into the existing feedforward multipliers without increasing the total number of multipliers. Let's consider, for example, the commonly used cascade form IIR digital filter structure.

09.06.5 Scaling the Cascade form IIR filter structure

Let us consider first the unscaled IIR filter structure.

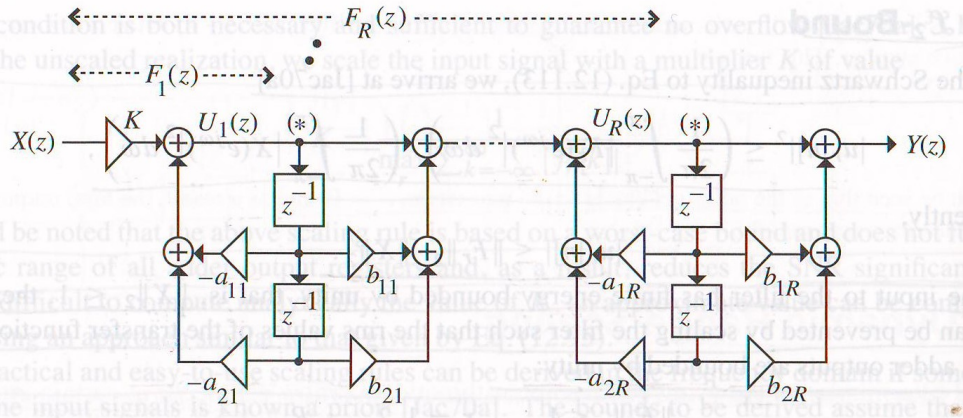


Figure 12.27: An unscaled cascade realization of second-order IIR sections.

$$H(z) = K \prod_{i=1}^{R} H_i(z)$$

where

$$H_i(z) = \frac{B_i(z)}{A_i(z)} = \frac{1 + b_{1i}z^{-1} + b_{2i}z^{-2}}{1 + a_{1i}z^{-1} + a_{2i}z^{-2}}.$$

The nodes to be scaled are those marked with (*) in the figure and correspond to the inputs of the multipliers in each second-order section. The scaling transfer functions are defined as:

$$F_r(z) = \frac{K}{A_r(z)} \prod_{l=1}^{r-1} H_l(z).$$

The scaled transfer function is shown in the following figure.

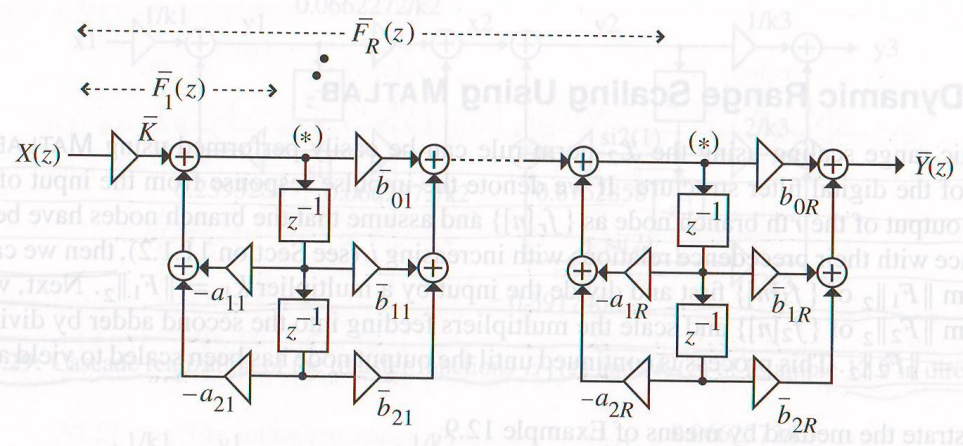


Figure 12.28: The scaled cascade realization.

The scaling process has introduced a new multiplier $\bar{b}_{0l}$ in each second-order section.
Let us denote

$$||F_r||_p = \alpha_r \qquad \forall r$$

$$||H||_p = \alpha_{R+1}$$

and let us choose the scaling constant as

$$\bar{K} = \beta_0 K; \qquad \bar{b}_{lr} = \beta_r b_{lr} \qquad l = 0, 1, 2; \qquad r = 1, 2, \ldots, R.$$

It can be easily verified that

$$\bar{F}_r = \left(\prod_{i=0}^{r-1} \beta_i \right) F_r(z)$$

$$\bar{H} = \left(\prod_{i=0}^{R} \beta_i \right) H(z)$$

With the scaling we what

$$||\bar{F}_r||_p = \left(\prod_{i=0}^{r-1} \beta_i \right) ||F_r||_p = \alpha_r \left(\prod_{i=0}^{r-1} \beta_i \right) = 1, \qquad r = 1, 2, \ldots, R$$

$$||\bar{H}||_p = \left(\prod_{i=0}^{r-1} \beta_i \right) ||H||_p = \alpha_{R+1} \left(\prod_{i=0}^{r-1} \beta_i \right) = 1.$$

Solving these equations we arrive at

$$\beta_0 = \frac{1}{\alpha_1},$$

$$\beta_r = \frac{\alpha_r}{\alpha_{r+1}}, \qquad r = 1, 2, \ldots, R.$$

There are many possible cascade realizations of a higher-order IIR transfer function achieved through various pole-zero pairings and orderings. In fact, for a cascade of R second-order sections, there are $(R!)^2$ different possible realizations. Each of these realizations will have different scaling and output noise power.

What is the optimal pole-zero pairing and ordering of the sections in a cascade realization? There exists a simple heuristic set of rules.

For the pole-zero pairing: First, pair the complex pole pair closest to the unit circle with the nearest complex zero pair. Next, pair the complex pole pair closest to the previous set of poles with its nearest complex zero pair. Repeat this process until all poles and zeros have been paired.

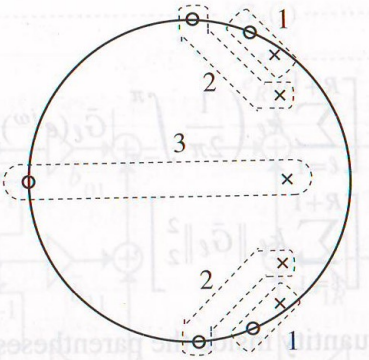


Figure 12.32: Illustration of the pole-zero pairing and ordering rules.

The above procedure will reduce the peak gain of the section characterized by the paired poles and zeros.

As for the ordering of the sections, it depends on the $\mathcal{L}_p$-norm used for scaling. If the $\mathcal{L}_2$-norm is used, the ordering of the paired sections does not significantly influence the output noise power. However, if an $\mathcal{L}_\infty$-scaling is employed, the section with poles closest to the unit circle and exhibiting the most pronounced magnitude response should be placed closest to the output end. The next section should be the one with the next most pronounced magnitude response, and so on. The first section should be the one with the least pronounced magnitude response. The same rule applies in the general case.

09.07 Limit cycles

In recursive systems, the nonlinearities associated with finite precision arithmetic (such as rounding and overflows) can induce periodic oscillations at the system output or yield a constant output, even when the input is zero or remains constant. These phenomena are known as *limit cycles* and are directly caused by errors arising from rounding in multiplications or overflows in additions. The problem is especially notable in IIR filters with poles near the unit circle.

Example:

Consider the system,

$$y(n) = 0.95y(n-1) + x(n),$$

$$H(z) = \frac{1}{1 - 0.95z^{-1}}$$

with $x(n) = 0$ for all $n \geq 0$, $y(-1) = 13$.

Let us assume to round $y(n)$ to an integer value:

n	$y(n)$ exact	$y(n)$ rounded
-1	13	13
0	12.35	$12 \longleftarrow (12.35)$
1	11.73	$11 \longleftarrow (11.4)$
2	11.14	$10 \longleftarrow (10.45)$
3	10.75	$10 \longleftarrow (9.5)$
4	10.05	$10 \longleftarrow (9.5)$
5	9.5	$10 \longleftarrow (9.5)$

Another example is the first-order IIR filter depicted in the following figure, which includes a quantizer after the multiplication

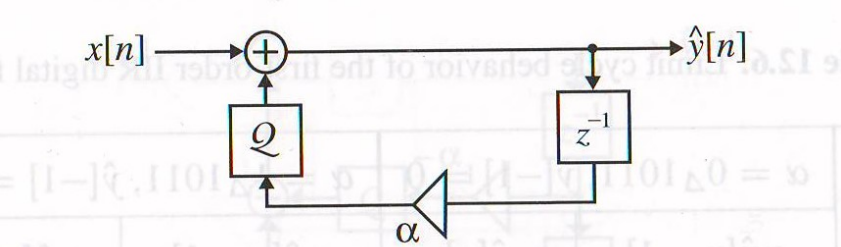


Figure 12.48: A first-order IIR filter with a quantizer.

The digital filter is assumed to be implemented using 6-bit fractional arithmetic with a quantization step of $\delta = 2^{-5}$. The input signal is set as $x(0) = 0.4$, and $x(n) = 0$ for $n > 0$, with the initial condition $y(-1) = 0$.

The following figure shows the limit cycles that can be observed for $\alpha = 0.6$, resulting in a constant non-zero output, and for $\alpha = -0.6$, leading to an oscillatory output.

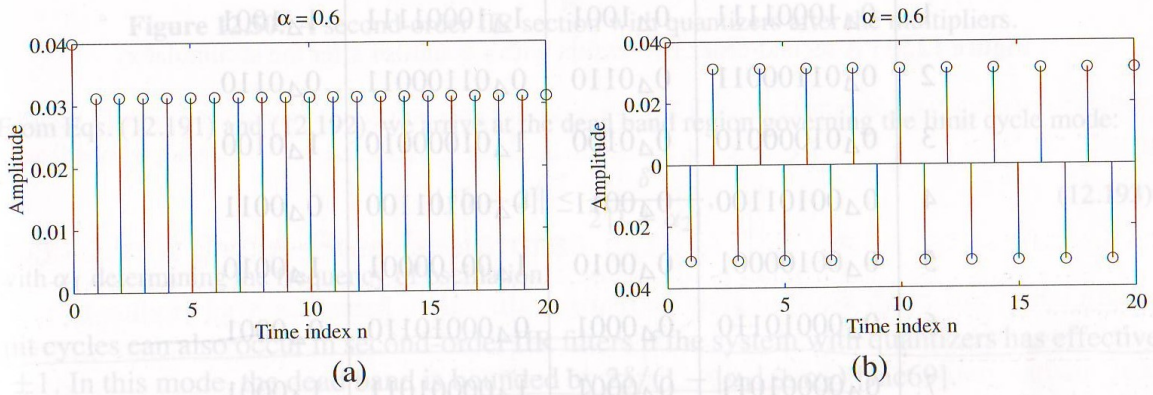


Figure 12.49: Illustration of limit cycles in a first-order IIR digital filter: (a) $\alpha = 0.6$ and (b) $\alpha = -0.6$.

The output signal can become entirely independent of the input in the presence of limit cycles. However,

there are conditions under which these limit cycles can be avoided, such as utilizing truncation towards 0 instead of rounding, or adding a small random noise to the input. Generally, increasing the precision of the arithmetic improves the behavior of the recursive system, thereby reducing the amplitude of the limit cycles.

For more information study:

S. K. Mitra, ”Digital Signal Processing: a computer based approach,” 4th edition, McGraw-Hill, 2011
Chapter 12.1, pp. 664-665
Chapter 12.2, pp. 665-667
Chapter 12.4.1, pp. 668-672
Chapter 12.5, pp. 681-687
Chapter 12.7, pp. 695-699 and 702-705
Chapter 12.11, pp. 719-721

10 Digital filter design

10.01 Digital filter design specifications

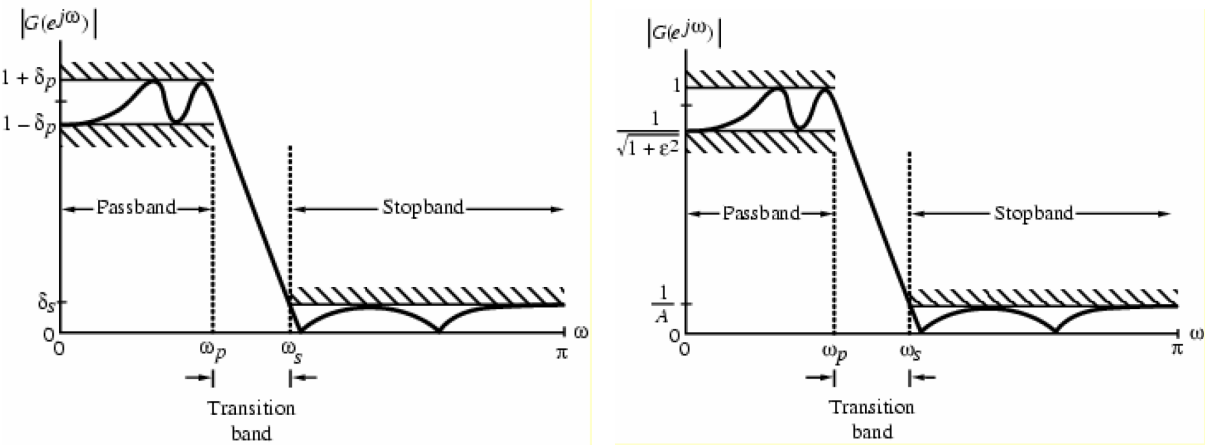
An important problem in digital signal processing is the design of digital filters, that is, the design of a realizable transfer function that can approximate some assigned specifications on the frequency response. When the filter to be designed is IIR, it is also necessary to guarantee that the transfer function is stable.

In digital filter design, first and foremost, we have to obtain a reasonable specification for the frequency response based on the requirements of the system our filter has to operate with. Then, we will have to decide whether we need to design an FIR or IIR filter and how to design it.

As for the specifications, we will derive (or be given) a desired magnitude or phase response. For simplicity, we concentrate our attention only on the design of lowpass, highpass, bandpass, or bandstop filters, and the specifications involve a desired magnitude response.

The specifications in the passband and in the stopband are generally given with some tolerances. Moreover, a transition band is always introduced between the passband and the stopband in order to allow the magnitude response to smoothly transition from the maximum to the minimum value.

The magnitude response specifications for a lowpass filter are typically given using one of the following two methods



(a) (b)

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

In the first method, the maximum deviations of the designed magnitude response from the ideal one are imposed in both the passband and the stopband. In the passband $0 \leq \omega \leq \omega_p$, we require the magnitude response to be bounded as follows:

$$1 - \delta_p \leq |G(e^{j\omega})| \leq 1 + \delta_p.$$

In the stopband $\omega_s \leq \omega \leq \pi$, we require the magnitude response to be

$$|G(e^{j\omega})| \leq \delta_s.$$

ω_p and ω_s are the *passband and stopband edge frequencies*; δ_s and δ_p are called the *peak ripple values*. In most applications, the transfer function we are interested in has real coefficients, and the magnitude response is an even function of ω. Thus, the specifications are given only in the range $0 \leq \omega \leq \pi$.
In the second method, the magnitude response specifications are given in normalized form, with the maximum value of the magnitude in the passband assumed to be 1. In this case, the maximum attenuation in the passband and the minimum attenuation in the stopband are imposed. In the passband $0 \leq \omega \leq \omega_p$, we request

$$\frac{1}{\sqrt{1+\epsilon^2}} \leq |G(e^{j\omega})| \leq 1,$$

while for the stopband $\omega_s \leq \omega \leq \pi$, we request

$$|G(e^{j\omega})| \leq \frac{1}{A}.$$

$\sqrt{1+\epsilon^2}$ is the maximum attenuation imposed in the passband (the attenuation is the reciprocal of the gain). A is the minimum attenuation desired in the stopband. In general, these values are given in dB; that is, we specify the maximum attenuation in dB in the passband as

$$R_p = 20\log_{10}\left(\sqrt{1+\epsilon^2}\right),$$

and the minimum attenuation in dB in the stopband,

$$R_s = 20\log_{10}(A).$$

It is easy to convert between one type of specification and the other, since

$$20\log_{10}\left(\sqrt{1+\epsilon^2}\right) \simeq -20\log_{10}(1-2\delta_p),$$

$$20\log_{10}(A) \simeq -20\log_{10}\delta_s.$$

The passband and stopband edge frequencies are often provided in Hertz, along with the sampling frequency F_c. If F_p and F_s represent the passband and stopband edge frequencies respectively, the normalized angular frequencies ω_p and ω_s are calculated as follows:

$$\omega_p = \frac{F_p}{F_c} \cdot 2\pi,$$

$$\omega_s = \frac{F_s}{F_c} \cdot 2\pi.$$

Based on our specifications, we need to decide whether to design an FIR or IIR filter.
FIR filters offer several advantages. They can achieve exactly linear phase, and they remain stable when coefficients are quantized. However, for specifications with small transition bands, the FIR filter order (and its computational complexity) can be much higher compared to an IIR filter implementing the same specifications. With IIR filters, ensuring stability is a challenge, and exact linear phase cannot be achieved. As a general guideline, if a linear phase filter is required, FIR filters are preferred. Conversely, for applications with small transition bands and strong attenuations in the stopband, IIR filters are preferred.

10.02 IIR filter design

A commonly used method in the design of IIR lowpass, highpass, bandpass, and bandstop filters leverages the extensive knowledge available for their corresponding analog counterparts.
In the analog domain, certain filter categories have been introduced and extensively studied over time. Comprehensive knowledge exists for these filters, including formulas that facilitate easy design based on specifications.
A successful digital filter design technique typically involves initially designing an analog filter that satisfies our specifications, and then transforming it into its digital equivalent without altering the magnitude response characteristics.

10.02.1 The analog domain or continuous-time domain

In the continuous-time domain, similar concepts to those in the discrete-time domain are introduced.
We've previously encountered the continuous-time Fourier transform (CTFT) of an analog signal:

$$X_a(j\Omega) = \int_{-\infty}^{+\infty} x_a(t)e^{-j\Omega}\mathrm{d}t,$$

which has its discrete equivalent in the DTFT.
In the continuous-time domain, the *Laplace Transform* also exists, which serves as the analog equivalent of the Z-Transform:

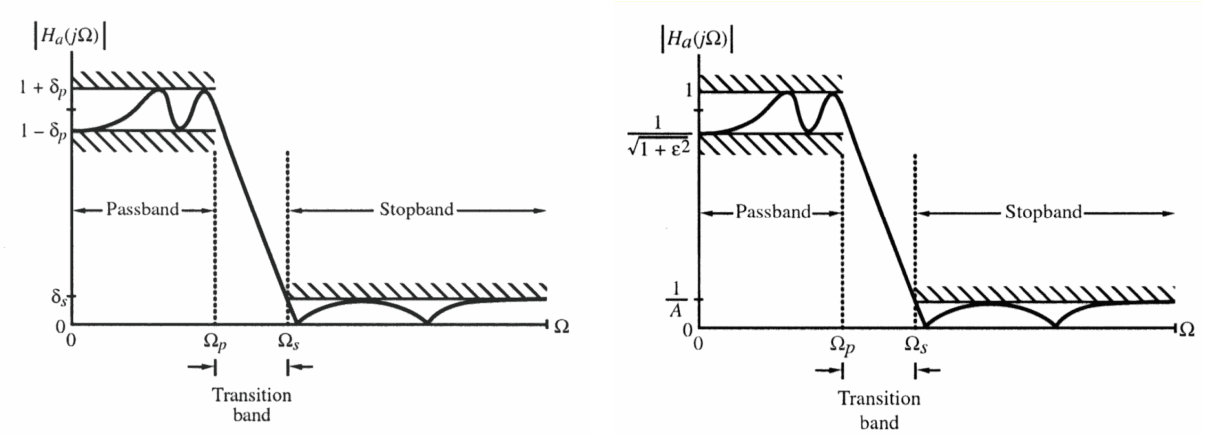
$$X_a(s) = \int_{-\infty}^{+\infty} x_a(t)e^{-st}\mathrm{d}t.$$

This is a function of the complex variable s, defined across the entire complex plane. When we evaluate $X_a(s)$ on the imaginary axis, for $s = j\Omega$, we obtain the CTFT $X_a(j\Omega)$. The imaginary axis for the Laplace transform plays a similar role to the unit circle for the Z-Transform.
Furthermore, in the continuous-time domain, the concept of a linear time-invariant system is introduced. It can be shown that an LTI system is fully characterized by the impulse response $h(t)$, which represents the system's response to a Dirac pulse $\delta(t)$. Indeed, the system's output is given by the *convolution integral*:

$$y_a(t) = \int_{-\infty}^{+\infty} h(\tau)x_a(t-\tau)\mathrm{d}\tau.$$

The CTFT and the Laplace transform possess the property of transforming the convolution integral into the product of the transforms of $h(t)$ and $x_a(t)$.
The CTFT of $h(t)$, $H(j\Omega)$, is termed the *frequency response* of the system. The Laplace transform of $h(t)$, denoted as $H(s)$, is referred to as the *transfer function* of the system.
A continuous-time LTI system described by a differential equation with constant coefficients has a rational transfer function in s. The system is stable in the BIBO sense if and only if the roots of the polynomial fall in the Re$(s) < 0$ semi-plane.

The specifications for analog lowpass, highpass, bandpass, and bandstop filters are similar to those for digital filters. The magnitude response in the passband and stopband cannot be constant and is specified with certain tolerances. Additionally, a transition band is specified between the passband and stopband to allow for a smooth drop-off of the magnitude.
The magnitude $|H_a(j\Omega)|$ of the lowpass filter may be specified as follows:



(a) (b)

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

In the first case, in the passband $0 \leq \Omega \leq \Omega_p$ we require

$$1-\delta_p \leq |H_a(j\Omega)| \leq 1+\delta_p,$$

while in the stopband $\Omega_s \leq \Omega \leq +\infty$, we require

$$|H_a(j\Omega)| \leq \delta_s$$

Here, Ω_p and Ω_s are respectively the *passband edge frequency* and the *stopband edge frequency*. The tolerances δ_p and δ_s are referred to as *ripples*.
In the second case, the specifications for the analog filter are given in a normalized form. The maximum value in the passband is 1, and the passband ripple, $1/\sqrt{1+\epsilon^2}$, is determined by the minimum value of the magnitude in the passband. The maximum stopband ripple is denoted by $1/A$, where A represents the minimum stopband attenuation.
In analog filter theory, two other parameters are defined. The first is the *transition ratio* or *selectivity parameter*, represented by the ratio:

$$k = \frac{\Omega_p}{\Omega_s}.$$

For lowpass filters it is always $k < 1$ The second parameter is the *discrimination parameter* defined as

$$k_1 = \frac{\epsilon}{\sqrt{A^2-1}}.$$

Typically, $k_1 \ll 1$.
In the analog domain, three families of lowpass, highpass, bandpass, and bandstop filters, which have well-established design formulas, have been extensively studied:

- Butterworth filters,

- Chebyshev filters of type I and II,
- Elliptic filters.

10.02.2 Butterworth filters

The magnitude-squared response of an analog lowpass Butterworth filter of order N is given by

$$|H_a(j\Omega)|^2 = \frac{1}{1+\left(\frac{\Omega}{\Omega_c}\right)^{2N}}.$$

It can be easily verified that the first $2N-1$ derivatives of $|H_a(j\Omega)|^2$ at $\Omega = 0$ are equal to zero. Thus, the Butterworth filter is said to have a *maximally flat magnitude* at $\Omega = 0$.
The gain of the Butterworth filter is dB is:

$$\mathcal{G}(\Omega) = 10\log_{10}|H_a(j\Omega)|^2.$$

For $\Omega = 0$, $|H_a(j\Omega)|^2 = 1$ and $\mathcal{G}(\Omega) = 0$ dB.
For $\Omega = \Omega_c$, $|H_a(j\Omega)|^2 = 1/2$ and $\mathcal{G}(\Omega) \simeq -3$ dB. Thus, Ω_C is called the 3-*dB cutoff frequency*.
The derivative of the square magnitude response is always negative for positive values of Ω, and the magnitude response is monotonically decreasing with increasing Ω.
For $\Omega \gg \Omega_c$,

$$|H_a(j\Omega)|^2 \simeq \frac{1}{\left(\frac{\Omega}{\Omega_c}\right)^{2N}}.$$

Let us consider $\Omega_2 = 2\Omega_1$ and $\Omega_1 \gg \Omega_c$. Then,

$$\mathcal{G}(\Omega_2) = -10\log_{10}\left(\frac{\Omega_2}{\Omega_c}\right)^{2N} = -10\log_{10}\left(\frac{2\Omega_1}{\Omega_c}\right)^{2N} = \mathcal{G}(\Omega_1) - 6N \text{ dB}.$$

As a result, the gain in the stopband decreases by 6 dB per octave or 20 dB per decade for an increase of the filter order by one.
This figure illustrates the magnitude response of a Butterworth lowpass filter for different orders N and the same 3-dB cut-off frequency. The larger the order, the faster the transition towards zero of the magnitude response and the narrower the transition band.

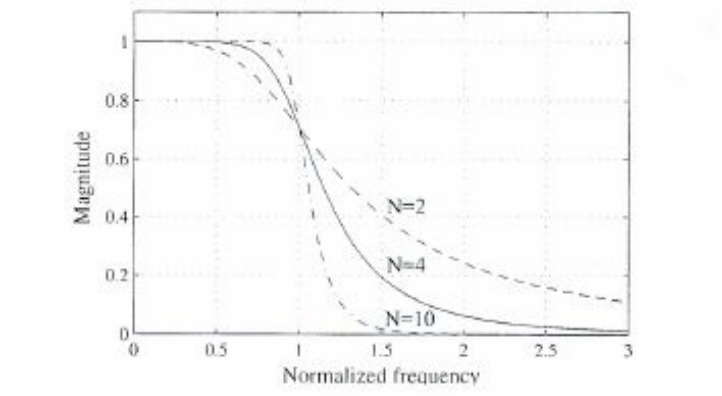


Figure A.3: Typical Butterworth lowpass filter responses.

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

The two parameters fully characterizing a Butterworth filter are Ω_c and N. These are determined from the specified passband edge Ω_p, the minimum passband magnitude $\sqrt{1+\epsilon^2}$, the stopband edge, and minimum stopband attenuation A.

It must be

$$|H_a(j\Omega_p)|^2 = \frac{1}{1+\left(\frac{\Omega_p}{\Omega_c}\right)^{2N}} = \frac{1}{1+\epsilon^2}$$

$$|H_a(j\Omega_s)|^2 = \frac{1}{1+\left(\frac{\Omega_s}{\Omega_c}\right)^{2N}} = \frac{1}{A^2}$$

Solving these equations for the filter order we obtain

$$N = \frac{\log_{10}[(A^2-1)/\epsilon^2]}{\log_{10}(\Omega_s/\Omega_p)} = \frac{\log_{10}(1/k_1)}{\log_{10}(1/k)}$$

Since N must be an integer, the order N is rounded up to the next higher integer, and the final value of Ω_c is then obtained from any one of the first two equations.

The Butterworth filter has an all-poles transfer function

$$H_a(s) = \frac{C}{D_N(s)} = \frac{\Omega_c^N}{\prod_{l=1}^{N}(s-p_l)}$$

with

$$p_l = \Omega_c e^{j[\pi(N+2l-1)/2N]} \qquad l = 1, 2, \ldots, N$$

10.02.3 Chebyshev filters of type I

There are two types of Chebyshev filter: Type 1 and Type 2. In Type 1, the frequency response magnitude is equiripple in the passband and monotonic in the stopband. In Type 2, the frequency response magnitude is monotonic in the passband and maximally flat for $\Omega = 0$ and equiripple in the stopband.

The square-magnitude response of the analog lowpass Type 1 Chebyshev filter of order N is

$$|H_a(j\Omega_p)|^2 = \frac{1}{1+\epsilon^2 T_N^2(\Omega/\Omega_p)},$$

where $T_N(\Omega)$ is the Chebyshev polynomial of order N:

$$T_N(\Omega) = \begin{cases} \cos(N\cos^{-1}\Omega), & |\Omega| \le 1, \\ \cosh(N\cosh^{-1}\Omega), & |\Omega| > 1. \end{cases}$$

Alternatively, the polynomial can be derived with the following recurrence relation

$$T_r(\Omega) = 2\Omega T_{r-1}(\Omega) - T_{r-2}(\Omega), \qquad \ge 2,$$

with $T_0(\Omega) = 1$ and $T_1(\Omega) = \Omega$.

A typical magnitude response of an analog lowpass Type 1 Chebyshev filter is shown below:

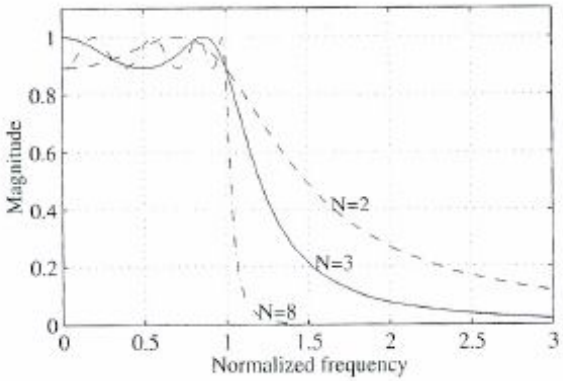


Figure A.4: Typical Type 1 Chebyshev lowpass filter responses with 1-dB passband ripple.

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

The order N of the transfer function is determined by the attenuation specification in the stopband, for instance at the stopband edge frequency:

$$|H_a(j\Omega_p)|^2 = \frac{1}{1+\epsilon^2 T_N^2(\Omega_s/\Omega_p)}, \tag{10.01}$$

$$= \frac{1}{1+\epsilon^2\left\{\cosh[N\cos^{-1}(\Omega_s/\Omega_p)]\right\}^2} = \frac{1}{A^2} \tag{10.02}$$

Solving the above equation we have

$$N = \frac{\cosh^{-1}(\sqrt{A^2-1}/\epsilon)}{\cosh^{-1}(\Omega_s/\Omega_p)} = \frac{\cosh^{-1}(1/k_1)}{\cosh^{-1}(1/k)}.$$

Once again, the transfer function $H_a(s)$ is all-pole and there are simple formulas for computing the poles.

10.02.4 Chebyshev filters of type II

The square-magnitude response of the analog lowpass Type II Chebyshev filter of order N is

$$|H_a(j\Omega_p)|^2 = \frac{1}{1+\epsilon^2\left[\dfrac{T_N^2(\Omega_s/\Omega_p)}{T_N^2(\Omega_s/\Omega)}\right]},$$

Typical responses are shown in the following figure:

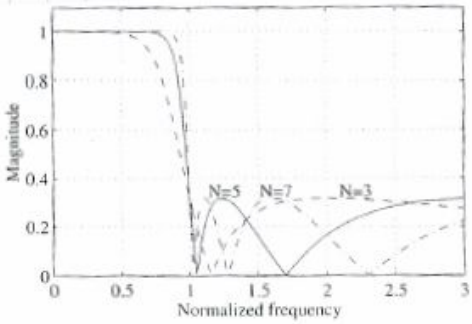


Figure A.5: Typical Type 2 Chebyshev lowpass filter responses with 10-dB minimum stopband attenuation.

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

The transfer function of Type 2 Chebyshev filters is no longer all-pole and includes both poles and zeros. There are formulas available for computing the order, as well as the locations of the zeros and poles. For the same attenuation specifications and order, Chebyshev filters of Type I and Type II have a transition band that is smaller than that of Butterworth filters.

10.02.5 Elliptic filters

The *elliptic filters* exhibit an equiripple magnitude response in both the passband and the stopband, as depicted in the following figure:

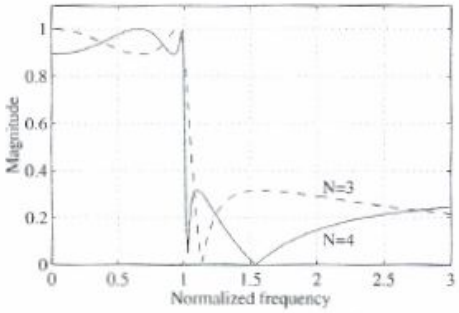


Figure A.6: Typical elliptic lowpass filter responses with 1-dB passband ripple and 10-dB minimum stopband attenuation.

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

The transfer function of an elliptic filter satisfies a given set of filter specifications, including passband edge frequency, stopband edge frequency, passband ripple, and minimum stopband attenuation, with the lowest filter order N.
The square-magnitude response of a lowpass elliptic filter is given by:

$$|H_a(j\Omega_p)|^2 = \frac{1}{1+\epsilon^2 R_N^2(\Omega/\Omega_p)}$$

where $R_N(\Omega)$ is a rational function of order N satisfying the property $R_N(1/\Omega) = 1/R_N(\Omega)$, with the

roots of its numerator lying in the interval $0 < \Omega < 1$ and the roots of the denominator lying in the interval $1 < \Omega < \infty$.

The transfer function includes poles and zeros, and there are formulas available for computing the order N from the specifications and determining the positions of poles and zeros.

We have explored the design of Butterworth, Chebyshev, and Elliptic lowpass filters. The design of highpass, bandpass, and bandstop filters can be achieved through straightforward spectral transformations of the frequency variable. The design process entails: i. Development of specifications for a prototype analog lowpass filter based on the desired highpass, bandpass, or bandstop filter. ii. Design of the prototype lowpass filter. iii. Transformation of the prototype lowpass filter to the desired filter by applying the inverse of the frequency transformation used to determine the specification of the lowpass filter.

Let s denote the Laplace transform variable of the prototype analog lowpass transfer function $H_{\mathrm{LP}}(s)$, and let $\hat{s}$ denote the Laplace transform variable of the desired highpass, bandpass, or bandstop filter $H_{\mathrm{D}}(\hat{s})$. The mapping from the s-domain to the $\hat{s}$-domain is represented by an invertible transformation:

$$s = F(\hat{s}).$$

The relationship between $H_{\mathrm{LP}}(s)$ and $H_{\mathrm{D}}(\hat{s})$ is defined as follows:

$$H_{\mathrm{D}}(\hat{s}) = H_{\mathrm{LP}}(s)|_{s=F(\hat{s})}\,. \tag{10.03}$$

$$H_{\mathrm{LP}}(s) = H_{\mathrm{D}}(\hat{s})|_{\hat{s}=F^{-1}(s)}\,. \tag{10.04}$$

10.02.6 Analog highpass filter design

An analog lowpass transfer function $H_{\mathrm{LP}}(s)$ with passband edge frequency Ω_p can be transformed into an analog highpass transfer function $H_{\mathrm{HP}}(\hat{s})$ with passband edge frequency $\hat{\Omega}_p$ using the spectral transformation

$$s = \frac{\Omega_p \hat{\Omega}_p}{\hat{s}}.$$

On the imaginary axis, we have

$$\Omega = -\frac{\Omega_p \hat{\Omega}_p}{\hat{\Omega}}$$

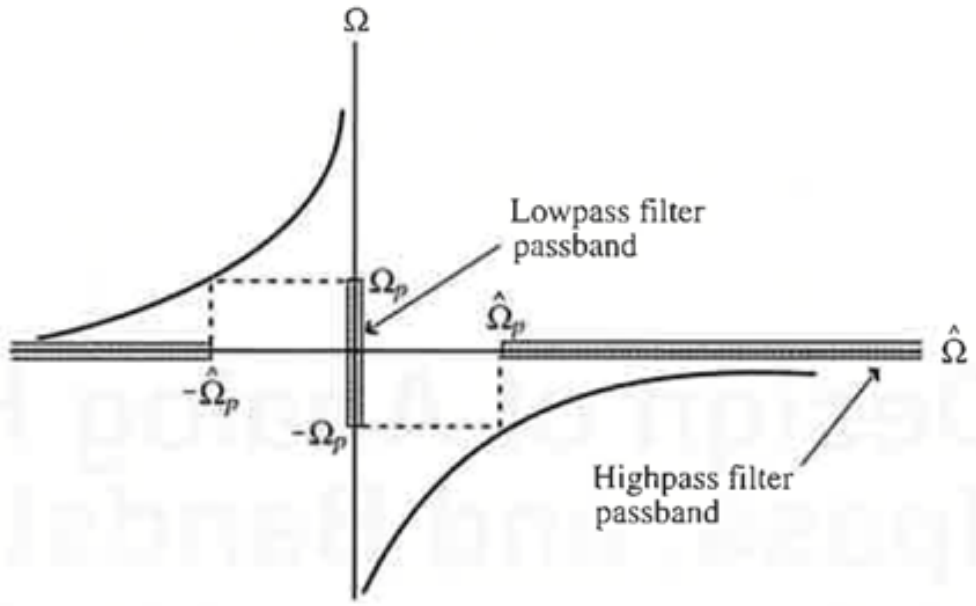


Figure B.1: The lowpass-to-highpass mapping of Eq. (2.2).

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

Note that this mapping transform the band $0 \leq \Omega \leq \Omega_p$ into the range $-\infty < \hat{\Omega} \leq \hat{\Omega}_p$, and the range $\Omega_p \leq \Omega < \infty$ into the range $-\hat{\Omega}_p \leq \hat{\Omega} < 0$; and similar for the negative frequencies. Thus, the mapping ensures that the gain values of $H_{\rm LP}(j\Omega)$ of the prototype filter in the passband $|\Omega| \leq \Omega_p$ will appear in the passband of the highpass filter $H_{\rm HP}(j\hat{\Omega})$, $\hat{\Omega} \geq \Omega_p$. Likewise, the gain values of $H_{\rm LP}(j\Omega)$ in the stopband $|\Omega| > \Omega_p$ will appear in the stopband of the highpass filter $H_{\rm HP}(j\hat{\Omega})$, $0 \leq \hat{\Omega} < \hat{\Omega}_p$.

10.02.7 Analog bandpass filter design

An analog lowpass transfer function $H_{\rm LP}(s)$ with passband edge frequency Ω_p can be transformed into an analog bandpass transfer function $H_{\rm BP}(\hat{s})$ with passband edge frequencies $\hat{\Omega}_{p1}$ and $\hat{\Omega}_{p2}$ ($\hat{\Omega}_{p1} < \hat{\Omega}_{p2}$) using the spectral transformation

$$s = \Omega_p \frac{\hat{s}^2 + \hat{\Omega}_0^2}{\hat{s}(\hat{\Omega}_{p2} - \hat{\Omega}_{p1})},$$

with $\hat{\Omega}_0$ called the *passband center frequency* of the passband filter, satisfying:

$$\hat{\Omega}_0^2 = \hat{\Omega}_{p2} \cdot \hat{\Omega}_{p1} = \hat{\Omega}_{s2} \cdot \hat{\Omega}_{s1}$$

Thus, the two passband edge frequencies exhibit geometric symmetry with respect to the center frequency $\hat{\Omega}_0$. On the imaginary axis, we have

$$\Omega = -\Omega_p \frac{\hat{\Omega}_0^2 - \hat{\Omega}^2}{\hat{\Omega} B_w}$$

where $B_w = \hat{\Omega}_{p2} - \hat{\Omega}_{p1}$ is the passband bandwidth of the bandpass filter. Note that $\Omega = 0$ is mapped onto the frequency $\hat{\Omega} = \hat{\Omega}_0$.

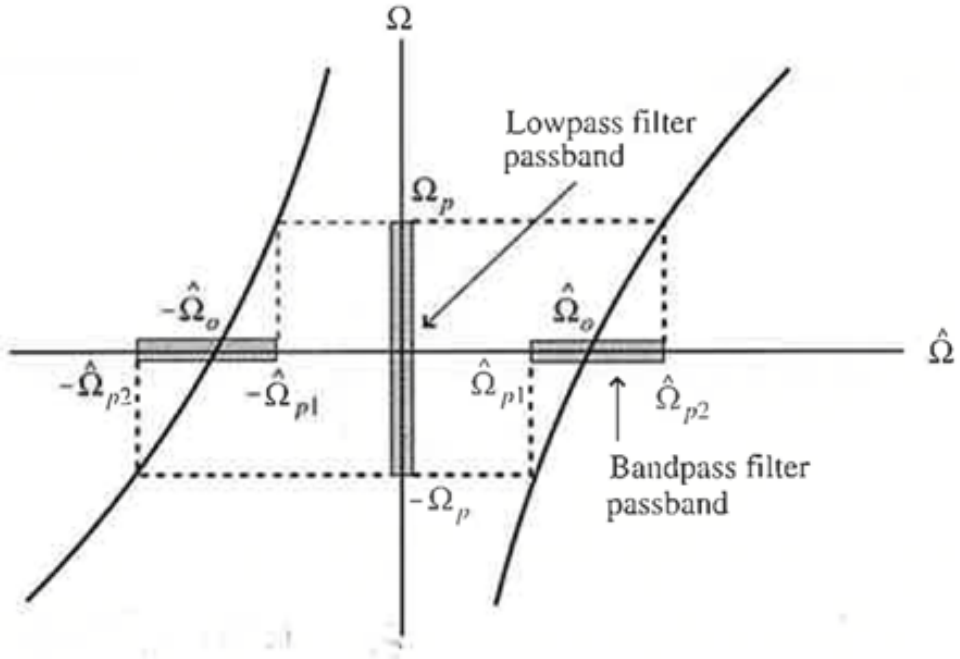


Figure B.3: The lowpass-to-bandpass mapping of Eq. (2.4).

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

10.02.8 Analog bandstop filter design

An analog lowpass transfer function $H_{\mathrm{LP}}(s)$ with passband edge frequency Ω_p can be transformed into an bandstop transfer function $H_{\mathrm{BS}}(\hat{s})$ with stopband edge frequencies $\hat{\Omega}_{s1}$ and $\hat{\Omega}_{s2}$ ($\hat{\Omega}_{s1} < \hat{\Omega}_{s2}$) using the spectral transformation:

$$s = \Omega_s \frac{\hat{s}(\hat{\Omega}_{s2} - \hat{\Omega}_{s1})}{\hat{s}^2 + \hat{\Omega}_0^2},$$

where $\hat{\Omega}_0$ is referred to as the *stopband center frequency* of the stopband filter, satisfying:

$$\hat{\Omega}_0^2 = \hat{\Omega}_{p2} \cdot \hat{\Omega}_{p1} = \hat{\Omega}_{s2} \cdot \hat{\Omega}_{s1}$$

Once again, the two stopband edge frequencies exhibit geometric symmetry with respect to the center frequency $\hat{\Omega}_0$. On the imaginary axis, we have

$$\Omega = \Omega_s \frac{\hat{\Omega} B_w}{\hat{\Omega}_0^2 - \hat{\Omega}^2}$$

where $B_w = \hat{\Omega}_{s2} - \hat{\Omega}_{s1}$ represents the width of the stopband of the stopband filter.

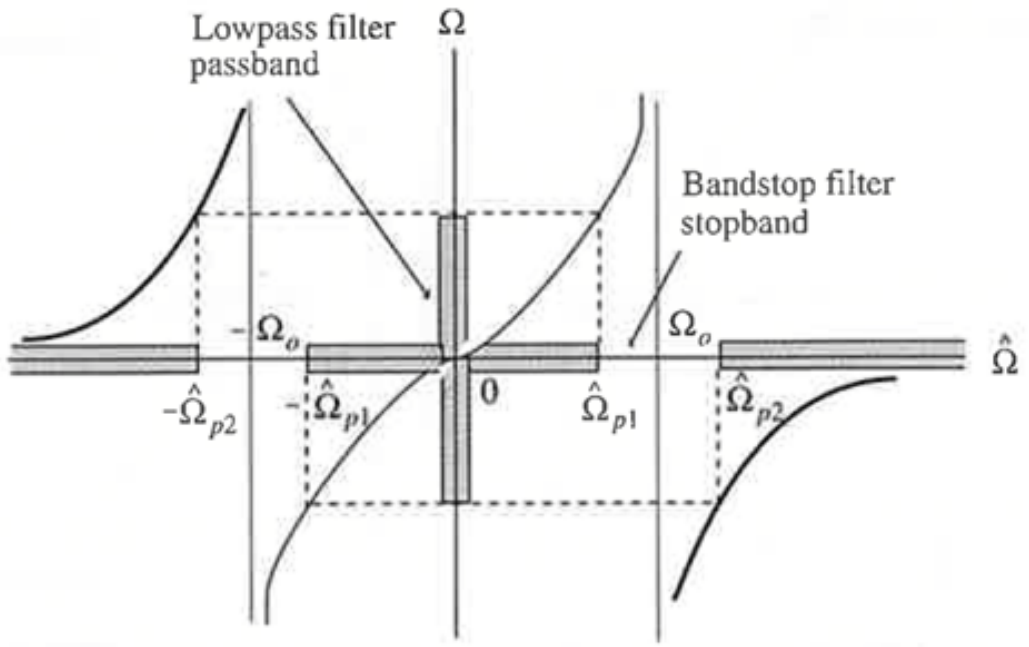


Figure B.5: The lowpass-to-bandstop mapping of Eq. (2.7).

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

In summary, we are knowledgeable about Butterworth, Chebyshev, and Elliptic filters, and there exist formulas that enable us to easily design them. It is natural to leverage this knowledge for designing equivalent digital filters.

10.03 The bilinear transformation

A number of transformations have been proposed to convert an analog transfer function $H_a(s)$ into a digital transfer function $G(z)$, transferring the essential properties of the analog transfer function in the s-domain to the digital transfer function in the z-domain. Of these, the *bilinear transformation* is the most commonly used in the design of IIR filters. The transformation is a one-to-one mapping: it maps a single point in the s-plane to a unique point in the z-plane and vice versa.
A bilinear transformation that maps a point in the z-plane to a point in the s-plane is given by

$$s = \frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right).$$

The relation between $G(z)$ and $H_a(s)$ is

$$G(z) = H_a(s)\big|_{s=\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}.$$

The bilinear transformation is derived by applying the trapezoidal numerical integration with step size T to the differential equation representation of $H_a(s)$, obtaining the difference equation representation of $G(z)$.

Let us examine the above transformation. The inverse transformation is:

$$z = \frac{1 + \frac{T}{2}s}{1 - \frac{T}{2}s}.$$

For a specific point $s = \sigma_0 + j\Omega_0$ in the s-plane, we have

$$z = \frac{1 + \frac{T}{2}(\sigma_0 + j\Omega_0)}{1 - \frac{T}{2}(\sigma_0 + j\Omega_0)} = \frac{(1 + \frac{T}{2}\sigma_0) + j\frac{T}{2}\Omega_0}{(1 - \frac{T}{2}\sigma_0) - j\frac{T}{2}\Omega_0}.$$

Thus,

$$|z|^2 = \frac{(1 + \frac{T}{2}\sigma_0)^2 + (\frac{T}{2}\Omega_0)^2}{(1 - \frac{T}{2}\sigma_0)^2 + (\frac{T}{2}\Omega_0)^2}.$$

It follows that

$$|z|^2 \begin{cases} < 1, & \text{for } \sigma_0 < 0, \\ = 1, & \text{for } \sigma_0 = 0, \\ > 1, & \text{for } \sigma_0 > 0. \end{cases}$$

A point on the $j\Omega$-axis in the s-plane ($\sigma_0 = 0$) is mapped onto a point on the unit circle in the z-plane.
A point on the left s-semiplane ($\sigma_0 < 0$) is mapped onto a point inside the unit circle in the z-plane.
A point on the right s-semiplane ($\sigma_0 > 0$) is mapped onto a point outside the unit circle in the z-plane.
Any point in the s-plane is mapped onto a unique point in the z-plane as illustrated by the following figure:

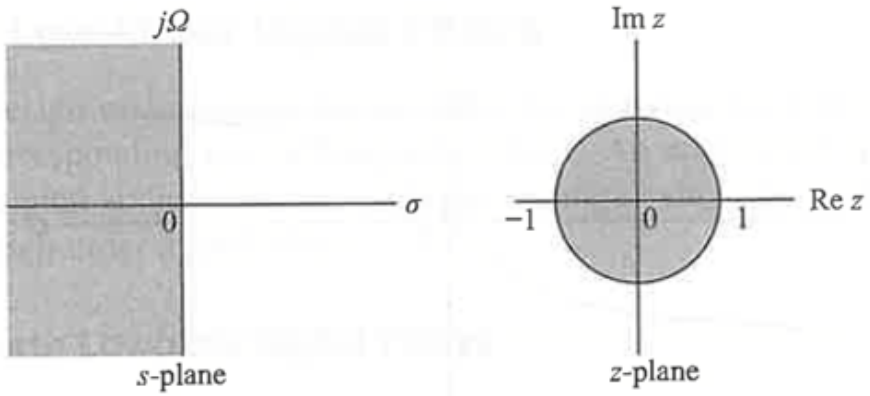


Figure 9.2: The bilinear transformation mapping.

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

Particularly interesting is the relation between the imaginary axis in the s-plan ($s = j\Omega$) and the unit circle in the z-plane ($z = e^{j\omega}$):

$$j\Omega = \frac{2}{T}\left(\frac{1 + e^{-j\omega}}{1 + e^{j\omega}}\right) = j\frac{2}{T}\tan(\frac{\omega}{2}),$$
$$\Omega = \frac{2}{T}\tan(\frac{\omega}{2}).$$

Alternatively,

$$\omega = 2\tan^{-1}\left(\frac{\Omega T}{2}\right).$$

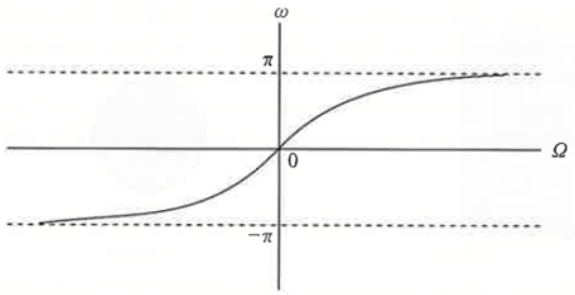


Figure 9.3: Mapping of the angular analog frequencies Ω to the angular digital frequencies ω via the bilinear transformation of Eq. (9.14).

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

This relation is highly nonlinear. The range of frequencies $-\infty < \Omega < +\infty$ in the analog domain is warped into the range $-\pi < \omega < +\pi$ in the digital domain. This introduces a distortion in the frequency axis called *frequency warping*.

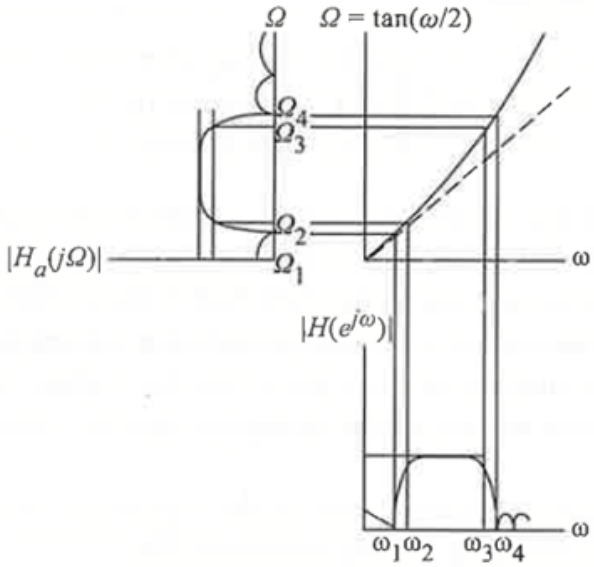


Figure 9.4: Illustration of the frequency warping effect on the magnitude response.

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

To develop a digital filter meeting a specified magnitude response:

1. we must first prewarp the critical band-edge frequencies (ω_p and ω_s) to find their analog equivalents (Ω_p and Ω_s) using

$$\Omega = \frac{2}{T}\tan(\frac{\omega}{2}).$$

2. We must design the analog prototype using the prewarped critical frequencies, and then
3. Transform $H_a(s)$ using the bilinear transformation $s = \dots$ to obtain $G(z)$.

The bilinear transformation preserves the form of the magnitude response of an analog filter only when the specifications require a piecewise constant magnitude. Thus, a lowpass analog filter is transformed into a lowpass digital filter, a highpass filter into a highpass filter, a bandpass filter into a bandpass filter, and a bandstop filter into a bandstop filter. However, the phase response of the analog filter is not preserved after transformation.
It should be noted that the parameter T in reality has no effect on the expression for $G(z)$. For convenience, we can set $T = 2$ in the bilinear transformation to simplify the design procedure. The bilinear transformation reduces to:

$$
\begin{aligned}
s &= \frac{1 - z^{-1}}{1 + z^{-1}}, \\
z &= \frac{1 + s}{1 - s}, \\
\Omega &= \tan(\frac{\omega}{2}), \\
\omega &= 2 \tan^{-1} \Omega.
\end{aligned}
$$

10.03.1 Design of Highpass, bandpass, and bandstop IIR digital filters

For designing lowpass digital filters we will always follow the procedure outlined earlier. However, for highpass, bandpass, and bandstop IIR digital filters, two general approaches can be followed.
The first consists of the following steps:

1. Prewarp the specified digital frequency specifications of the desired digital filter $G_D(z)$ to arrive at the frequency specifications of an analog filter $H_D(s)$ of the same type.
2. Convert the frequency specifications of $H_D(s)$ into those of a prototype analog lowpass filter $H_{\mathrm{LP}}(s)$ using the frequency transformation we have seen for analog filters.
3. Design the analog lowpass filter $H_{\mathrm{LP}}(s)$.
4. Convert the transfer function $H_{\mathrm{LP}}(s)$ into $H_D(s)$ using the analog frequency transformation.
5. Transform $H_D(s)$ with the bilinear transformation to arrive at the desired digital filter $G_D(z)$.

The second approach consists of the following steps:

1. Prewarp the specified digital frequency specification of the desired digital filter $G_D(z)$ to arrive at the frequency specifications of an analog filter $H_D(s)$ of the same type.
2. Convert the frequency specification of $H_D(s)$ into those of a prototype analog lowpass filter $H_{\mathrm{LP}}(s)$ using the frequency transformation we have seen for analog filters.
3. Design the analog lowpass filter $H_{\mathrm{LP}}(s)$.
4. Convert the transfer function $H_{\mathrm{LP}}(s)$ with the bilinear transformation to arrive at a digital lowpass transfer function $G_{\mathrm{LP}}(z)$.

5. Transform $G_{\mathrm{LP}}(z)$ into the desired digital filter $G_D(z)$ using an appropriate spectral transformation in the digital domain.

The difference between the two methods lies in the domain where the transformation between the lowpass and the desired highpass, bandpass, or bandstop filter is applied. The transformation can be performed in the analog domain using the formulas we have seen earlier, or it can also be performed in the digital domain.

The transformation in the digital domain can be executed with the following formulas, which convert a lowpass digital filter with a cutoff frequency ω_c into a highpass filter, a bandpass filter, and also a bandstop filter:

Table 9.1: Spectral transformations of a lowpass filter with a cutoff frequency ω_c.

Filter type	Spectral Transformation	Design Parameters
Lowpass	$z^{-1} = \frac{\hat{z}^{-1} - \lambda}{1 - \lambda \hat{z}^{-1}}$	$\lambda = \frac{\sin\left(\frac{\omega_c - \hat{\omega}_c}{2}\right)}{\sin\left(\frac{\omega_c + \hat{\omega}_c}{2}\right)}$ $\hat{\omega}_c$ = desired cutoff frequency
Highpass	$z^{-1} = -\frac{\hat{z}^{-1} + \lambda}{1 + \lambda \hat{z}^{-1}}$	$\lambda = -\frac{\cos\left(\frac{\omega_c + \hat{\omega}_c}{2}\right)}{\cos\left(\frac{\omega_c - \hat{\omega}_c}{2}\right)}$ $\hat{\omega}_c$ = desired cutoff frequency
Bandpass	$z^{-1} = -\frac{\hat{z}^{-2} - \frac{2\lambda\rho}{\rho+1}\hat{z}^{-1} + \frac{\rho-1}{\rho+1}}{\frac{\rho-1}{\rho+1}\hat{z}^{-2} - \frac{2\lambda\rho}{\rho+1}\hat{z}^{-1} + 1}$	$\lambda = \frac{\cos\left(\frac{\hat{\omega}_{c2} + \hat{\omega}_{c1}}{2}\right)}{\cos\left(\frac{\hat{\omega}_{c2} - \hat{\omega}_{c1}}{2}\right)}$ $\rho = \cot\left(\frac{\hat{\omega}_{c2} - \hat{\omega}_{c1}}{2}\right)\tan\left(\frac{\omega_c}{2}\right)$ $\hat{\omega}_{c2}, \hat{\omega}_{c1}$ = desired upper and lower cutoff frequencies
Bandstop	$z^{-1} = \frac{\hat{z}^{-2} - \frac{2\lambda}{1+\rho}\hat{z}^{-1} + \frac{1-\rho}{1+\rho}}{\frac{1-\rho}{1+\rho}\hat{z}^{-2} - \frac{2\lambda}{1+\rho}\hat{z}^{-1} + 1}$	$\lambda = \frac{\cos\left(\frac{\hat{\omega}_{c2} + \hat{\omega}_{c1}}{2}\right)}{\cos\left(\frac{\hat{\omega}_{c2} - \hat{\omega}_{c1}}{2}\right)}$ $\rho = \tan\left(\frac{\hat{\omega}_{c2} - \hat{\omega}_{c1}}{2}\right)\tan\left(\frac{\omega_c}{2}\right)$ $\hat{\omega}_{c2}, \hat{\omega}_{c1}$ = desired upper and lower cutoff frequencies

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

10.03.2 IIR Filter design with Matlab

Matlab and Octave provide a set of functions that implement the design of IIR filters by transforming analog prototypes with the Bilinear transform. The functions are `butter`, `cheby1`, `cheby2`, `ellip`, which design Butterworth, Chebyshev type I and type II, and Elliptic filters, respectively. These functions can return the coefficients of the numerator and denominator polynomials of the transfer function, or the poles and zeros, along with the multiplicative constant factor of the filter.

```
[b, a] = butter(N, Wn, options)

[z, p, k] = butter(N, Wn, options)
```

For the Butterworth filter, the parameters include the filter order and the 3 dB cut-off frequency normalized with respect to the Nyquist frequency (i.e., the angular frequency π corresponds to the normalized frequency `1`; all previously introduced angular frequencies need to be divided by π in Matlab)
For the Chebyshev filter of type I, we use the following syntax:

```
[b, a] = cheby1(N, Rp, Wp, options)

[z, p, k] = cheby1(N, Rp, Wp, options)
```

where the parameters include the filter order `N`, the maximum attenuation in the passband `Rp` in dB, and the passband edge frequency (or the `Rp` dB cut-off frequency) `Wp`.
For the Chebyshev filter of type II we have:

```
[b, a] = cheby2(N, Rs, Ws, options)

[z, p, k] = cheby2(N, Rs, Ws, options)
```

with parameters the filter order `N`, the minimum attenuation in the stopband `Rs` in dB, and the stopband edge frequency `Ws`.
For the Elliptic filter

```
[b, a] = ellip(N, Rp, Rs, Wp, options)

[z, p, k] = ellip(N, Rp, Rs, Wp, options)
```

with parameters the filter order `N`, the maximum attenuation in the passband `Rp` in dB, the minimum attenuation in the stopband `Rs` in dB, and the passband edge frequency `Wp`.
By default, all these functions design a lowpass filter. To design a highpass filter we use the option `'high'`. To design a passband filter `Wn`, `Wp`, or `Ws` is an array with the two edge frequencies. To design a bandstop filter, `Wn`, `Wp`, or `Ws` is an array with the two edge frequencies and the option `'stop'` is used.
Examples:

```
[b,a] = butter(5, 0.4) % lowpass filter
[b,a] = cheby1(4, 1, [0.4, 0.7]) % bandpass filter
[b,a] = cheby2(6, 60, 0.8, 'high') % highpass filter
[b,a] = ellip(3, 1, 60, [0.4, 0.7], 'stop') %bandstop filter
```

Given the specifications in terms of passband and stopband edge frequencies and of attenuation in the passband and stopband, the values to be passed to these functions can be obtained with the following functions:

```
[N, Wn] = buttord(Wp, Ws, Rp, Rs)
[N, Wp] = cheby1ord(Wp, Ws, Rp, Rs)
[N, Ws] = cheby2ord(Wp, Ws, Rp, Rs)
[N, Wp] = ellipord(Wp, Ws, Rp, Rs)
```

10.03.3 Direct methods for IIR filter design

There exist design methods that directly search for the IIR filter coefficients of the digital filter that meet our specifications. These methods typically apply optimization techniques on a computer.
Let's denote $G(e^{j\omega})$ as the frequency response of the filter we design, and $D(e^{j\omega})$ as the desired frequency response we want to approximate (usually given in the form of a piecewise linear function). The objective of the design is to iteratively compute the filter coefficients such that the error between $G(e^{j\omega})$ and $D(e^{j\omega})$ is minimized. Typically, a weighted version of this error is minimized:

$$\epsilon(\omega) = W(e^{j\omega})[G(e^{j\omega}) - D(e^{j\omega})],$$

where $W(e^{j\omega})$ is a positive weight function chosen by the designer.
A commonly employed method is to apply the *minmax criterion* and search for the coefficients that minimize the maximum error in a certain set of frequency bands R (with R typically including all the passbands and stopbands),

$$\epsilon = \max_{\omega \in R} |\epsilon(\omega)|.$$

A second optimization criterion aims to minimize the following integral:

$$\epsilon = \int_{\omega \in R} \left|W(e^{j\omega})[G(e^{j\omega}) - D(e^{j\omega})]\right|^p \mathrm{d}\omega,$$

where often $p = 2$ (*least square criterion*). As p approaches $+\infty$, this criterion tends toward the minimax criterion. The integral is often approximated by the sum:

$$\epsilon = \sum_{i=1}^{k} \left|W(e^{j\omega_i})[G(e^{j\omega_i}) - D(e^{j\omega_i})]\right|^p,$$

which is evaluated at a dense grid of points ω_i.
In practice, we iteratively look for the coefficient vector $[b, a]$ that minimizes one of these criteria.
Two examples of Matlab functions that optimize these criteria are `yulewalk`, which applies the least square method, and `iirlpnorm`, which minimizes the p-th norm.

10.03.4 Design method for IIR filters with arbitrary magnitude and phase

There are situations in which you have to design an IIR filter with an arbitrary magnitude and phase. For this purpose, we can apply a simple approach called *frequency-domain least-square*. The input

to the algorithm is a large number of magnitude and phase values (typically thousands) at arbitrary frequencies between 0 Hz and half the sampling rate, or at arbitrary normalized frequencies between 0 and π. The approach is quite flexible and can be used to design filters with poles and zeros, only zeros (an FIR filter), or only poles.

Let us consider a causal IIR filter with the following transfer function

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \dots b_M z^{-M}}{1 + a_1 z^{-1} + \dots + a_N z^{-N}}$$

and time-domain difference equation

$$y(n) = -a_1 y(n-1) - \dots - a_N y(n-N) + b_0 x(n) + b_1 x(n-1) + \dots + b_M x(n-M).$$

In matrix form we have

$$y(n) = \begin{bmatrix} -y(n-1) & \dots & -y(n-N) & x(n) & \dots & x(n-M) \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ \vdots \\ a_N \\ b_0 \\ \vdots \\ b_M \end{bmatrix}.$$

Let us assume that at angular frequency ω_1 the desired magnitude response is A_1 and the phase response is ϕ_1. Remember the physical meaning of the magnitude and phase spectrum: If we apply to the filter a sinusoid at normalized angular frequency ω_1,

$$x_1(n) = \cos(\omega_1 n),$$

the resulting output is the same sinusoid amplified by A_1 and shifted in phase by ϕ_1:

$$y_1(n) = A_1 \cos(\omega_1 n + \phi_1).$$

Knowing the expression of the input and the output signal, we can easily estimate $y_1(0), y_1(-1), \dots, y_1(-N)$ and $x(0), x(-1), \dots, x(-M)$ and we can obtain one equation in $M + N + 1$ unknowns:

$$y_1(0) = \begin{bmatrix} -y_1(-1) & \dots & -y_1(-N) & x_1(n) & \dots & x_1(-M) \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ \vdots \\ a_N \\ b_0 \\ \vdots \\ b_M \end{bmatrix}$$

We can now repeat this procedure for all K angular frequencies ω_i for which we know the desired

magnitude and phase response, where $K \gg M+N+1$ in general. We obtain the system of equations:

$$\begin{bmatrix} y_1(n) \\ y_2(n) \\ \vdots \\ y_K(n) \end{bmatrix} = \begin{bmatrix} -y_1(-1) & \dots & -y_1(-N) & x_1(n) & \dots & x_1(-M) \\ -y_2(-1) & \dots & -y_2(-N) & x_2(n) & \dots & x_2(-M) \\ \vdots & & \vdots & \vdots & & \vdots \\ -y_K(-1) & \dots & -y_K(-N) & x_K(n) & \dots & x_K(-M) \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ \vdots \\ a_N \\ b_0 \\ \vdots \\ b_M \end{bmatrix},$$

which we can write shortly as

$$\mathbf{y} = \mathbf{X} \cdot \mathbf{h},$$

with

$$\mathbf{y} = \begin{bmatrix} y_1(n) & y_2(n) & \dots & y_K(n) \end{bmatrix}^T,$$

$$\mathbf{X} = \begin{bmatrix} -y_1(-1) & \dots & -y_1(-N) & x_1(n) & \dots & x_1(-M) \\ -y_2(-1) & \dots & -y_2(-N) & x_2(n) & \dots & x_2(-M) \\ \vdots & & \vdots & \vdots & & \vdots \\ -y_K(-1) & \dots & -y_K(-N) & x_K(n) & \dots & x_K(-M) \end{bmatrix},$$

$$\mathbf{h} = \begin{bmatrix} a_1 & \dots & a_N & b_0 & \dots & b_M \end{bmatrix}^T.$$

The equation system is overdetermined since it has more equations than unknowns but it can be solved with the least-square approach using the pseudo-inverse:

$$\mathbf{h} \simeq (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}.$$

The entire design method can be summarized in the following steps:

1. Select the transfer function numerator and denominator orders M and N, respectively.
2. Define K separate input cosine sequences $x_1(n)$ for $n = -M, \dots, 0$.
3. Compute the K separate output cosine sequences $y_1(n)$ for $n = -N, \dots, 0$.
4. Fill the $\mathbf{X}$ matrix with the input and output cosine sequences.
5. Fill the vector $\mathbf{y}$ with the output cosine values.
6. Compute the pseudo-inverse. The resulting vector $\mathbf{h}$ contains the filter coefficients.

10.04 FIR filter design

The FIR filter design is always performed in the digital domain without exploiting the analog domain.
In the case of FIR filters, linear phase filters can always be designed.
A straightforward and effective method for designing FIR lowpass, highpass, bandpass, bandstop filters, as well as filters with a generic amplitude response, is the *window method*.

Let $H_d(e^{j\omega})$ denote the desired frequency response function. Our objective is to find a finite-duration impulse response sequence $\{h_t(n)\}$ of length $2M+1$ whose DTFT $H_t(e^{j\omega})$ approximates the desired DTFT $H_d(e^{j\omega})$ in some sense. A common criterion is to minimize the integral-squared error:

$$\Phi_R = \frac{1}{2\pi}\int_{-\pi}^{\pi} |H_t(e^{j\omega}) - H_d(e^{j\omega})|^2 \mathrm{d}\omega.$$

Using the Parseval relation

$$\begin{aligned}\Phi_R &= \sum_{n=-\infty}^{+\infty} |h_t(n) - h_d(n)|^2 \\ &= \sum_{n=-M}^{+M} |h_t(n) - h_d(n)|^2 + \sum_{n=-\infty}^{-M-1} h_d^2(n) + \sum_{n=M+1}^{+\infty} h_d^2(n)\end{aligned}$$

It is evident that the integral-squared error is minimized when $h_t(n) = h_d(n)$ for all $-M \leq n \leq M$. In other words, the best finite-length approximation to the ideal infinite-length impulse response in terms of mean-square error is obtained by truncation.

A causal FIR filter with impulse response $h(n)$ can be derived from $h_t(n)$ by delaying the latter by M samples,

$$h(n) = h_t(n - M).$$

$h(n)$ has the same magnitude response as $h_t(n)$ while the phase response has a linear phase shift of ωM radians.

Let us consider the ideal filter which implements our specifications. For simplicity, let us refer to a lowpass ideal filter,

$$H_{LP}(e^{j\omega}) = \begin{cases} 1 & |\omega| \leq \omega_c \\ 0 & \omega_c < |\omega| \leq \pi. \end{cases}$$

We know the corresponding impulse response

$$h_{LP}(n) = \frac{\sin(\omega_c n)}{\pi n} \qquad \infty < n < +\infty$$

which is doubly infinite, not absolutely summable, and not realizable. However, the impulse response tends to 0 as $n \to \pm\infty$.

If we set to zero all the coefficients outside the interval $-M \leq n \leq M$, we arrive at an approximation of the impulse response with finite length $2M+1$, which is non-causal. By delaying this impulse response by M samples, we obtain the coefficients of a causal FIR lowpass filter that approximates our specifications. The filter obtained in this way exhibits an oscillatory behavior (a ripple) in the magnitude response, commonly referred to as the *Gibbs phenomenon*

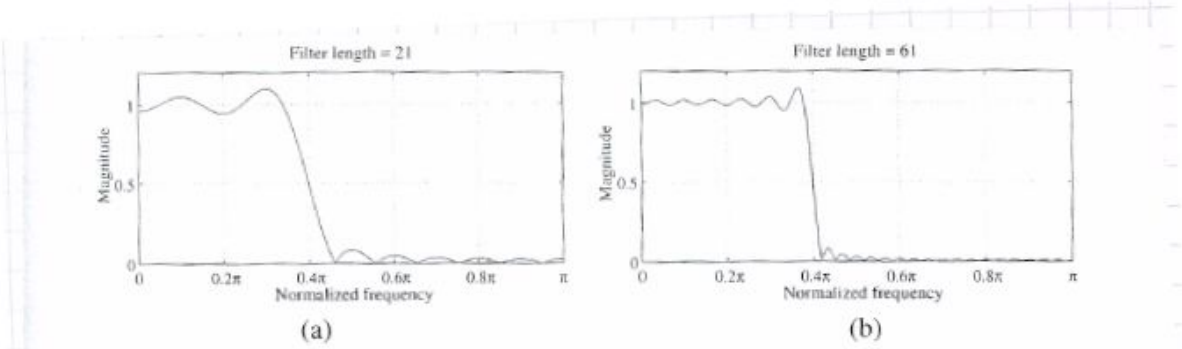


Figure 10.2: Magnitude responses of lowpass filters designed using the truncated impulse response of Eq. (10.15): (a) length $N = 21$ and (b) length $N = 61$.

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

The ripple is evident on both sides of the cutoff frequency ω_c (which is equal to 0.4π in this case). By increasing the filter length, the number of oscillations increases and the amplitude of the oscillations reduces. Nevertheless, the amplitudes of the two most relevant ripples, on the two sides of the cutoff frequency ω_c, remain unaltered.

The Gibbs phenomenon can be explained by considering the truncation operation as a multiplication operation between the impulse response $h_{LP}(n)$ and a rectangular window function $w(n)$, and by examining the effect in the frequency domain:

$$\widetilde{h}_{LP}(n) = h_{LP}(n) \cdot w(n)$$

$$\widetilde{H}_{LP}(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{+\pi} H_{LP}(e^{j\phi}) W(e^{j(\omega-\phi)}) \mathrm{d}\phi.$$

The spectrum of the truncated impulse response is the convolution between the spectrum of the lowpass ideal filter and that of the rectangular function:

$$W(e^{j\omega}) = \sum_{n=-M}^{M} e^{-j\omega n} = \frac{\sin([2M+1]\frac{\omega}{2})}{\sin(\frac{\omega}{2})}.$$

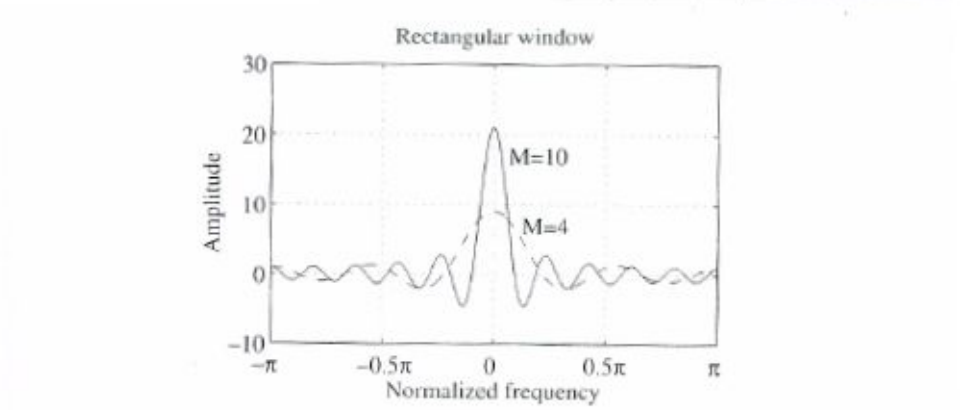


Figure 10.5: Frequency responses of the rectangular window for $M = 4$ and $M = 10$.

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

If M is small (indicating a low-complexity filter), we observe a large main lobe and significant oscillations at all frequencies. As M increases, the main lobe becomes narrower, and the significant oscillations concentrate around it, while the area under each lobe remains constant. Understanding the convolution between this spectrum and that of an ideal lowpass filter helps elucidate the behavior of the resulting spectrum $\widetilde{H}(e^{j\omega})$ and the behavior of the Gibbs oscillations.

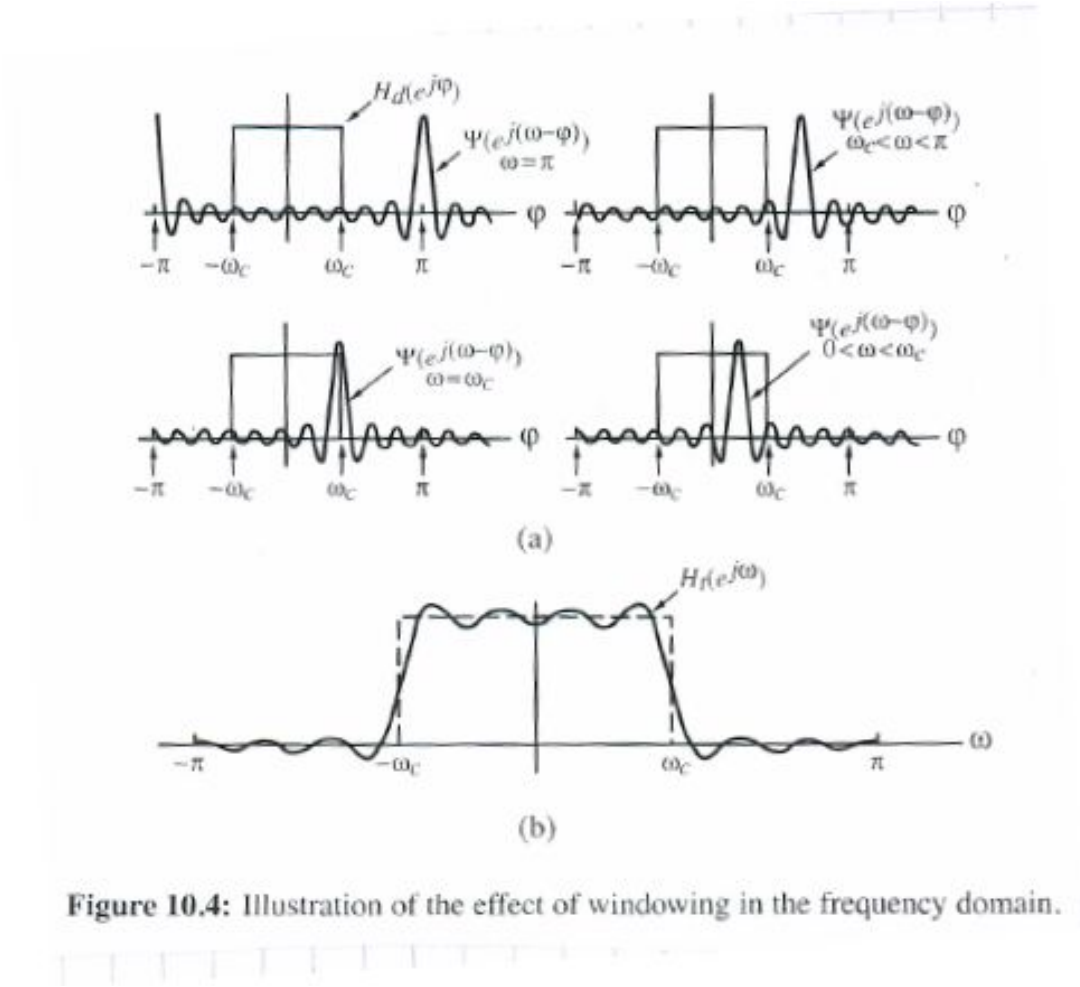


Figure 10.4: Illustration of the effect of windowing in the frequency domain.

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

The rectangular window exhibits a steep transition to zero outside the interval $-M \leq n \leq M$, which is the cause of the Gibbs phenomenon in the magnitude response. To mitigate the Gibbs phenomenon, other window functions with smoother transitions to zero can be used. Some commonly used window functions include:
Bartlett or triangular window:

$$w(n) = 1 - \frac{|n|}{M+1},$$

Hann window:

$$w(n) = \frac{1}{2}\left[1 - \cos(\frac{\pi n}{M})\right],$$

Hamming window:

$$w(n) = 0.54 - 0.46\cos(\frac{\pi n}{M}),$$

Blackman window:

$$w(n) = 0.42 - 0.5\cos(\frac{\pi n}{M}) + 0.08\cos(\frac{2\pi n}{M}).$$

These windows have a wider main lobe compared to the rectangular window, but significantly attenuate secondary lobes.

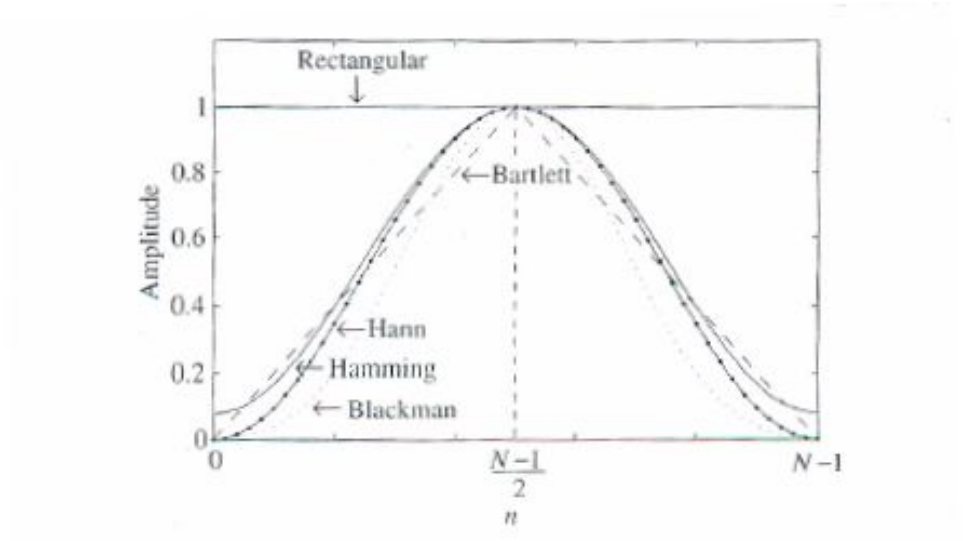


Figure 10.6: Plots of the fixed windows shown with solid lines for clarity.

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

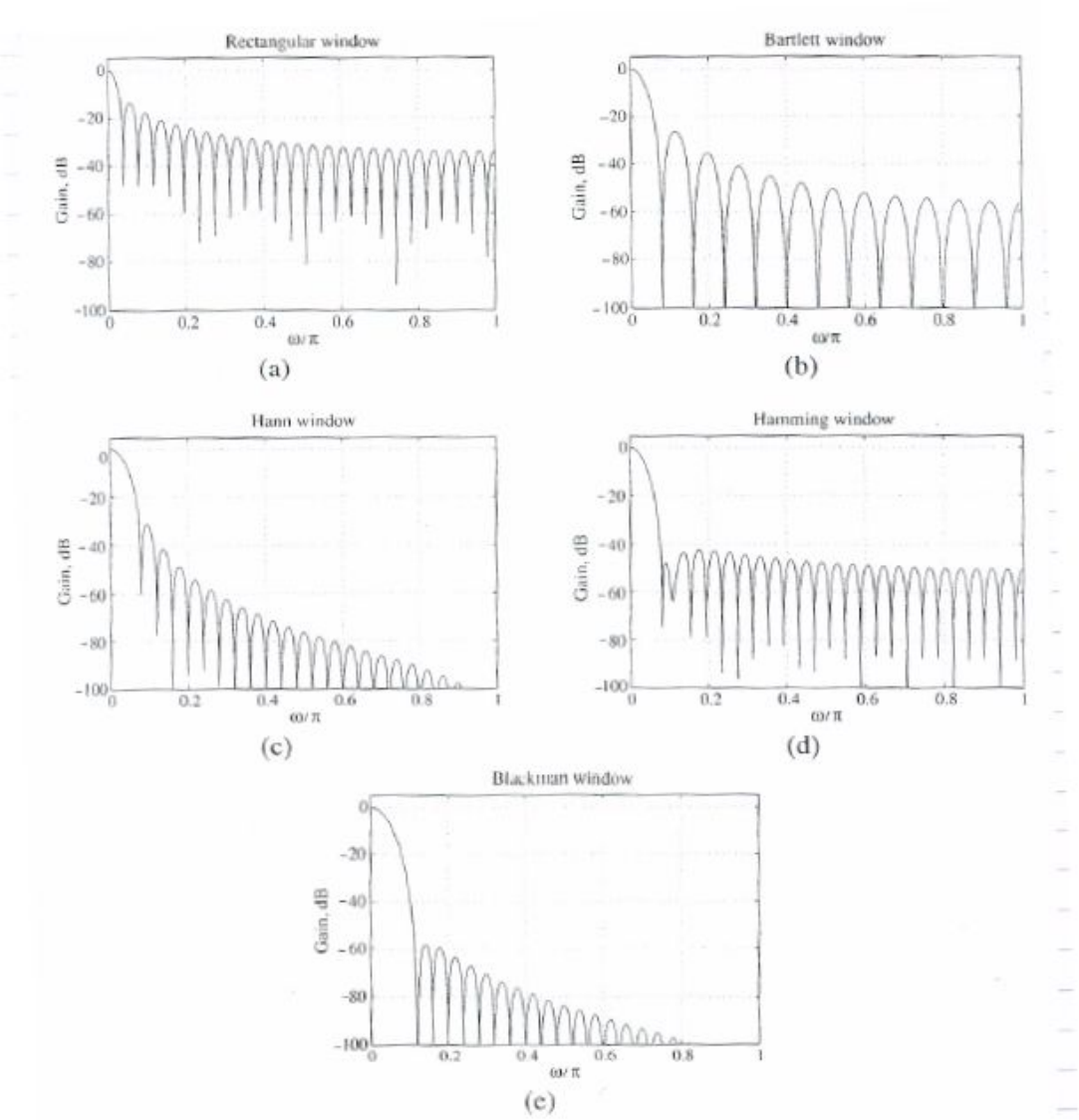


Figure 10.7: Gain responses of the fixed window functions.

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

The width of the transition band of the designed filter primarily depends on the width of the main lobe of the window function (it is smaller than that width). The amplitude of ripple in the passband and stopband depends on the amplitude of the secondary lobes. The following table provides the width of the main lobe, the attenuation of the secondary lobe, the minimum attenuation in the stopband, and the width of the transition band for a filter designed with the window functions mentioned above.

Table 10.2: Properties of some fixed window functions.

Type of Window	Main Lobe Width Δ_{ML}	Relative Sidelobe Level $A_{s\ell}$	Minimum Stopband Attenuation	Transition Bandwidth $\Delta\omega$
Rectangular	$4\pi/(2M+1)$	13.3 dB	20.9 dB	$0.92\pi/M$
Bartlett	$4\pi/(M+1)$	26.5 dB	See text	See text
Hann	$8\pi/(2M+1)$	31.5 dB	43.9 dB	$3.11\pi/M$
Hamming	$8\pi/(2M+1)$	42.7 dB	54.5 dB	$3.32\pi/M$
Blackman	$12\pi/(2M+1)$	58.1 dB	75.3 dB	$5.56\pi/M$

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

The following figure illustrates the relationship between the magnitude response of a lowpass filter and the spectrum of the window function used in the design.

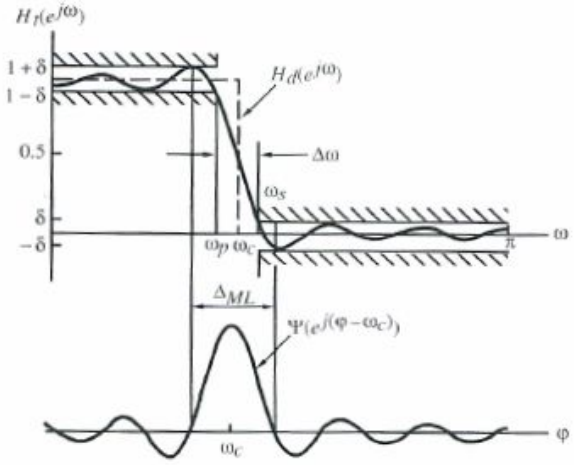


Figure 10.8: Relations among the frequency responses of an ideal lowpass filter, a typical window, and the windowed filter.

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

The ripple, denoted a δ, in the magnitude response of a filter designed with the window functions we have discussed has a fixed amplitude, which depends on the specific chosen window. Other window functions have been developed to control the ripple δ using another parameter β. By varying β, it is possible to reduce the amplitude of the secondary lobes at the expense of a wider main lobe. One of the most commonly used windows is the *Kaiser window*. Another notable window is the *Chebyshev window*, which offers equiripple sidelobes.

In Matlab, the window functions of Bartlett, Hann, Hamming, Blackman, Kaiser, and Chebyshev are generated with the following functions:

```
w = bartlett(N)
w = hann(N)
w = hamming(N)
w = blackman(N)
w = kaiser(N,beta)
w = chebywin(N, r)
```

(where r directly represents the sidelobe attenuation)
Matlab provides two functions, `fir1` and `fir2`, for the FIR filter design with the window functions[1].
`fir1` designs lowpass, highpass, bandpass, and bandstop FIR filters,

```
b = fir1(n, wn [, ftype] [,window])
```

Here, `n` is the filter order (the length minus 1). If not specified, the Hamming window is used. For a lowpass or highpass filter, `wn` is the normalized 6 dB cut-off frequency (the cut-off frequency of the ideal filter). For a bandpass filter or a bandstop filter, `wn` is an array with the two 6 dB cut-off frequencies. The optional parameter `ftype` is `'high'` for an highpass filter, `'stop'` for a bandstop filter. You can use any window function we have discussed before.
For the Kaiser window, given some filter specifications, the function `kaiserord` computes the corresponding parameters of the function `fir1`.

```
[n, wn, beta, ftype] = kaiserord(f, a, dev [, fs])
```

Here, `f` is a vector with the edges of the passbands and the stopbands (the first band goes from 0 to `f(1)`, the last one goes from from `f(L)` to 1) and `a` is a vector that specifies the desired amplitude values in these bands (0 or 1). The length of `f` must be twice the length of `a` minus 2. `f` and `a` define the desired spectrum as a piecewise constant function. The frequencies of `f` are normalized frequencies, or frequencies in Hz if we specify the sampling frequency `fs`. `dev` is a vector with the same size as `a` which defines the deviation, i.e., maximum ripple, in each band. The values returned by the `kaiserord` function can be passed to `fir1` as follows:

```
b = fir1(n, wn, ftype, kaiser(n+1, beta))
```

For example, to design a lowpass filter with passband from 0 to 1 kHz and stopband from 1.5 kHz and 4 kHz, using a sampling frequency of 8 kHz, a passband ripple of 5%, and a minimum attenuation in the stopband of 40 dB, you can use the following MATLAB code:

```
fsamp = 8000;
fcuts = [1000, 1500];
mags = [1, 0];
devs = [0.05, 0.01];
[n, Wn, beta, ftype] = kaiserord(fcuts, mags, devs, fsamp);
h = fir1(n, Wn, ftype, kaiser(n+1, beta));
freqz(h);
```

[1] These functions design linear phase filters. Choose with care the even or odd length of the filter.

For a passband filter, you can use:

```
fsamp = 8000;
fcuts = [1000, 1300, 2210, 2410];
mags = [0, 1, 0];
devs = [0.01, 0.05, 0.01];
[n, Wn, beta, ftype] = kaiserord(fcuts, mags, devs, fsamp);
n = n + rem(n,2); % to be sure to have an even length filter
h = fir2(n, Wn, ftype, kaiser(n+1, beta));
freqz(h);
```

The function `fir2` is also capable of designing FIR filters using the window method, but with an arbitrary piecewise linear magnitude response.
The following commands generate a row vector of length `n`+1 containing the coefficients of an FIR filter of order `n`, which approximate the magnitude response defined by vectors `f` and `m`.

```
n = 50,
f = [0, 0.4, 0.5, 1];
m = [1, 1, 0, 0];
b = fir2(n, f, m);
```

Here, `f` is a vector of normalized frequencies ranging from 0 to 1 (with 1 representing the Nyquist frequency), and `m` is a vector containing the magnitude response values at the frequencies specified in `f`. The filter aims to approximate the magnitude response illustrated in the following plot:

```
plot(f,m)
```

Computer-based design of IIR filters
For FIR filters, various design techniques have been introduced that iteratively optimize some error criterion on the frequency response. One commonly used criterion is the minimax criterion:

$$\epsilon = \max_{\omega \in R} \left| W(e^{j\omega})[G(e^{j\omega}) - D(e^{j\omega})] \right|$$

or the least-square criterion:

$$\epsilon = \int_{\omega \in R} \left| W(e^{j\omega})[G(e^{j\omega}) - D(e^{j\omega})] \right|^2 \mathrm{d}\omega$$

where R is the union of the passband and stopband frequency ranges, $G(e^{j\omega})$ is the designed frequency response, $D(e^{j\omega})$ is the desired frequency response, and $W(e^{j\omega})$ is a weight function used to assign different importance to various frequency ranges.
Matlab implements two functions, `firls` and `firpm`, which allow for designing filters with an arbitrary piecewise linear magnitude response, including transition bands. `firls` designs the filter with the least-square criterion. `firpm` minimizes the minmax criterion and applies the Parks-McClellan algorithm.This algorithm is widely used for FIR filter design and results in linear phase equiripple FIR filters, meaning filters with constant amplitude of passband or stopband ripple. The Parks-McClellan algorithm applies

the Remez theorem, which states that a filter of length $N = 2L + 1$ (for N odd) or of length $N = 2L$ (for N even) is optimal in the minmax sense if and only if in R there exist $L + 2$ extreme frequencies (maxima or minima) $\omega_0 \leq \omega_1 \leq \cdots \leq \omega_{L+1}$ such that

$$\epsilon(\omega_i) = -\epsilon(\omega_{i+1}) \text{ with } |\epsilon(\omega_i)| = \epsilon \quad \forall i,$$

where $\epsilon(\omega) = W(\omega)[A(\omega) - D(\omega)]$ represents the weighted error of the magnitude response.
The Parks-McClellan algorithm iteratively searches for these extreme frequencies in order to optimize the filter design.
`firls` and `firpm` have the same syntax.
Examples:
A lowpass filter of order 20, unit magnitude response between 0 and 0.4, zero magnitude response between 0.5 and 1 is

```
n=20;
f=[0, 0.4, 0.5, 1];
a=[1, 1, 0, 0];
b=firpm(n, f, a);
```

This is an equiripple filter. From 0.4 to 0.5 we have the transition band in which `firpm` does no optimization. With `b=firls(n, f, a);` we design a similar FIR filter with the least-square approach. Let us compare the two filters as follows:

```
[ H, W ]= freqz(b);
[ HH, W ]= freqz(bb);
plot(W/pi,abs(H),W/pi,abs(HH), '--');
```

The filter designed with `firpm` shows an equiripple behavior. `firls`, on the other hand, provides a lower error in most parts of the passband and stopband, but a larger error is obtained at frequencies $f = 0.4$ and $f = 0.5$.
With `firls` and `firpm` also passband, highpass, and bandstop filters can be designed. An example of passband filter is the following,

```
f = [0, 0.3, 0.4, 0.7, 0.8, 1];
a = [0, 0, 1, 1, 0, 0];
```

There are 5 bands:
$[0, 0.3]$ and $[0.8, 1]$ are two stopbands. $[0.4, 0.7]$ is the passband. $[0.3, 0.4]$ and $[0.7, 0.8]$ are two transition bands.
Examples of highpass and stopband filters are the following:

```
f = [0, 0.7, 0.8, 1];
a = [0, 0, 1, 1];
```

```
f = [0, 0.3, 0.4, 0.7, 0.8, 1];
a = [1, 1, 0, 0, 1, 1];
```

But we can also have any number of passbands and stopbands, and also passbands with different boundary values.
While minimizing the error, `firls` and `firpm` also allow giving different emphasis to the various bands using a weight vector.
For instance,

```
n=20;
f=[0, 0.4, 0.5, 1];
a=[1, 1, 0, 0];
w=[1, 10],
b=firpm(n, f, a, w);
```

In this example, `w` introduces a weight 10 times larger in the stopband compared to the passband. Therefore, in the stopband, we will have a ripple 10 times lower than in the passband. Each passband and stopband must be assigned a weight in the `w` vector.

For more information read:

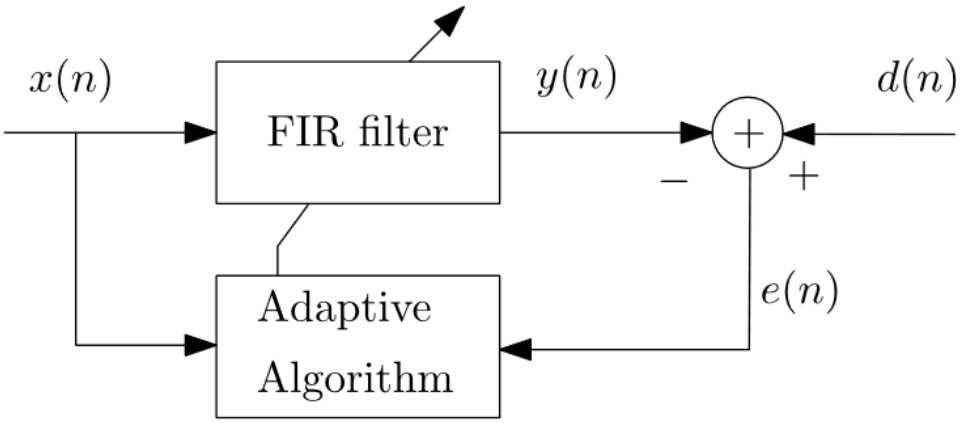
S. K. Mitra, ”Digital Signal Processing: a computer based approach,” 4th edition, McGraw-Hill, 2011
Chapter 9.1-9.5, pp. 489-509, 9.7, pp. 515-516,
Appendix A.1-A.4, pp. 863-871 Appendix B, pp. 863-871 Chapter 10.2, pp. 531-544, 10.3, pp. 556-562

11 Adaptive filters

11.01 Introduction

In classical applications of signal processing, the coefficients of the digital filter are designed to obtain a filter with the desired characteristics. For example, we design the coefficients to realize a low-pass filter, which lets pass only the low frequencies of the signal, or we design an equalization filter for some telecom or audio channel. The equalization filter emphasizes some frequencies and attenuates others. To design the filter, we must know the characteristics that the filter must have and that are required by the specific application. Given the specifics of the filter, we design the filter coefficients with a suitable algorithm with the end goal of obtaining a digital filter that satisfies those specifics. The designed coefficients are established once and for all and are then kept constant during the filter's use.

On the contrary, in adaptive filters, the coefficients vary in time; they are adapted in such a way that the filter adapts to an unknown environment or even to a time-varying environment. The adopted optimization criterion typically minimizes a certain error signal, the error between the current filter output $y(n)$ and a desired signal $d(n)$.



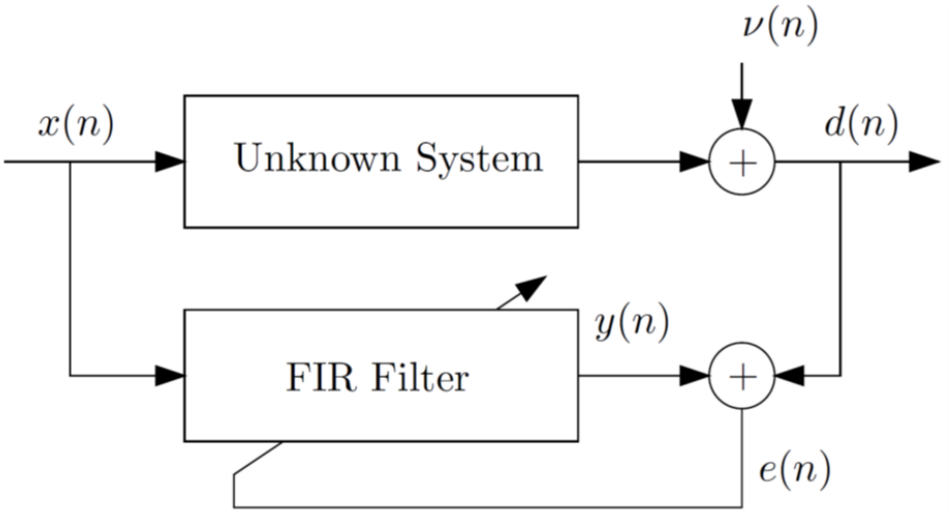
An adaptive algorithm modifies the filter coefficients based on the input signal $x(n)$ and the error signal $e(n)$. The adaptive filter aims to replicate the desired signal $d(n)$, which must be correlated with the input signal. Adaptive filters are typically FIR filters, although adaptive IIR filters have also been studied and proposed in the literature, albeit with the significant challenge of ensuring filter stability. Note that since the coefficients vary according to the input signal, even when adapting a linear filter structure, whether FIR or IIR, the resulting filter is non-linear: the superposition principle does not hold here.

Let us first examine some applications of adaptive filters.

11.02 Applications of adaptive filters

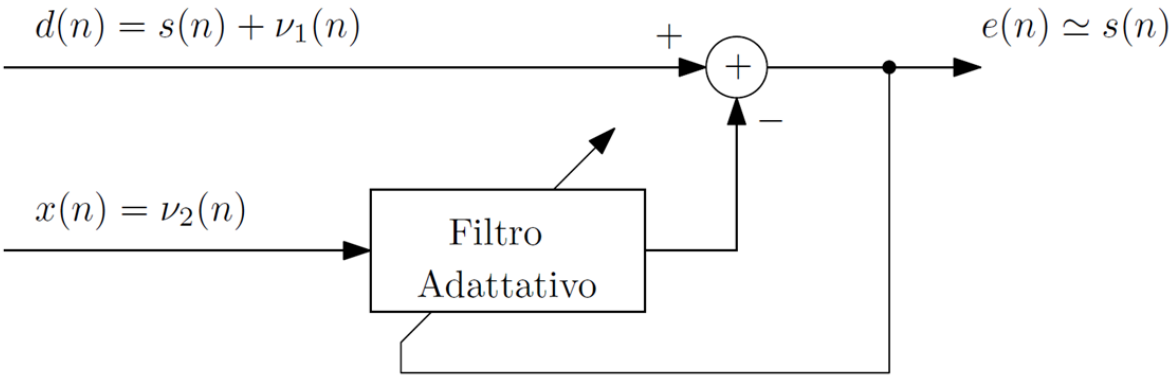
11.02.1 System identification

The primary application of adaptive filters is system identification, where the adaptive filter is employed to model an unknown system. It represents the core application of adaptive filtering, as in all other scenarios, the adaptive filter is utilized to identify an unknown system. The unknown system we aim to identify must meet certain hypotheses. For identifying an unknown system with an FIR filter, the system in question must be both linear and FIR.



11.02.2 Adaptive noise cancellation

Adaptive noise cancellation is employed when extracting a weak signal from a highly noisy environment is necessary. This method constructs a signal that effectively cancels out the noise instant by instant. To achieve noise cancellation, we require another noise signal that is correlated with the noise but not with the useful signal.

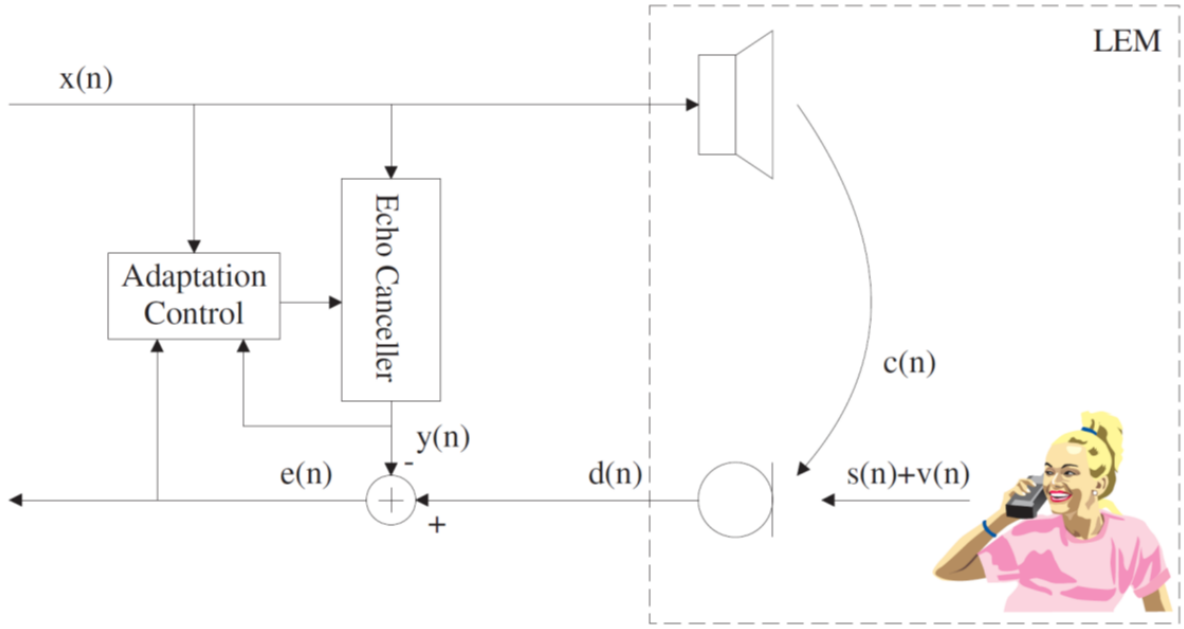


For example, this technique is used to cancel the noise picked up by the microphone of fighter pilots, helicopter pilots, and tank pilots. These environments are very noisy, and although the microphone is concealed within the pilot's mask, it still picks up a significant amount of noise that could mask the voice. A second microphone located outside the pilot's mask is utilized. This microphone captures the noise

present in the cockpit but not the pilot's voice. It provides the signal ν_2 in the figure above, which serves as the input for the adaptive filter, while the desired signal is the combination of the noise ν_1 inside the mask (correlated with ν_2) and the voice $s(n)$. The adaptive filter can model the noise ν_1 but not the voice. Therefore, the error signal represents the clear voice of the pilot. In recent years, a similar technique has also been employed in cellular phones, with one microphone capturing the speaker's voice and a second (or more) microphone(s) directed outward to capture only the surrounding noise.

11.02.3 Acoustic echo cancellation

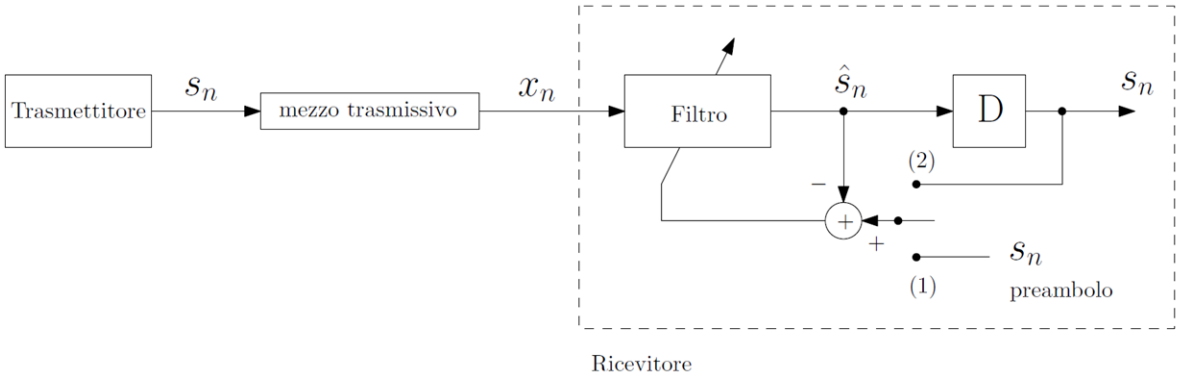
In hands-free phones, as well as in modern compact smartphones, headsets, and audio accessories, the voice emitted by the loudspeaker is captured by the microphone and returned to the far-end speaker. This results in the speaker perceiving the echo of their own voice. This echo is generally not bothersome if the delay is minimal (such as in classical analog telephone lines) because we are accustomed to hearing the echo of our own voice. However, if the delay becomes lengthy, on the order of hundreds of milliseconds as in modern telecom systems, the echo becomes highly disruptive, requiring suitable solutions to either eliminate or at least reduce it. An acoustic echo canceller is an adaptive filter designed to estimate the impulse response of the loudspeaker-enclosure-microphone system (LEM system) in such a manner that it can cancel the echo through subtraction.



11.02.4 Channel equalization

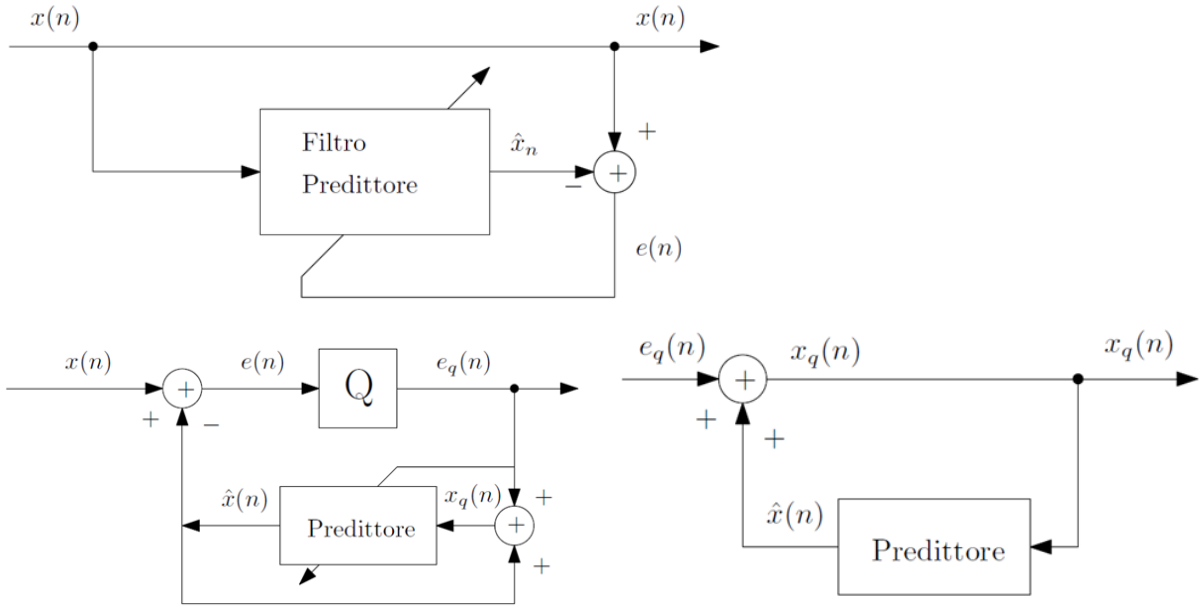
In a digital transmission system, information is encoded using appropriate waveforms, which are then transmitted through a transmission medium. These waveforms, as they pass through the transmission medium, are subject to distortion in an unknown manner, often varying with time. The resulting output signal is utilized to determine which symbol has been transmitted. An adaptive equalizer is frequently employed to mitigate or eliminate the effects of these distortions. Typically, each data packet is preceded

by a known header, which can be utilized as the desired signal to obtain an initial estimate of the channel equalizer. Following the header, the equalizer can be adapted using the decoded symbols.



11.02.5 Predictive coding

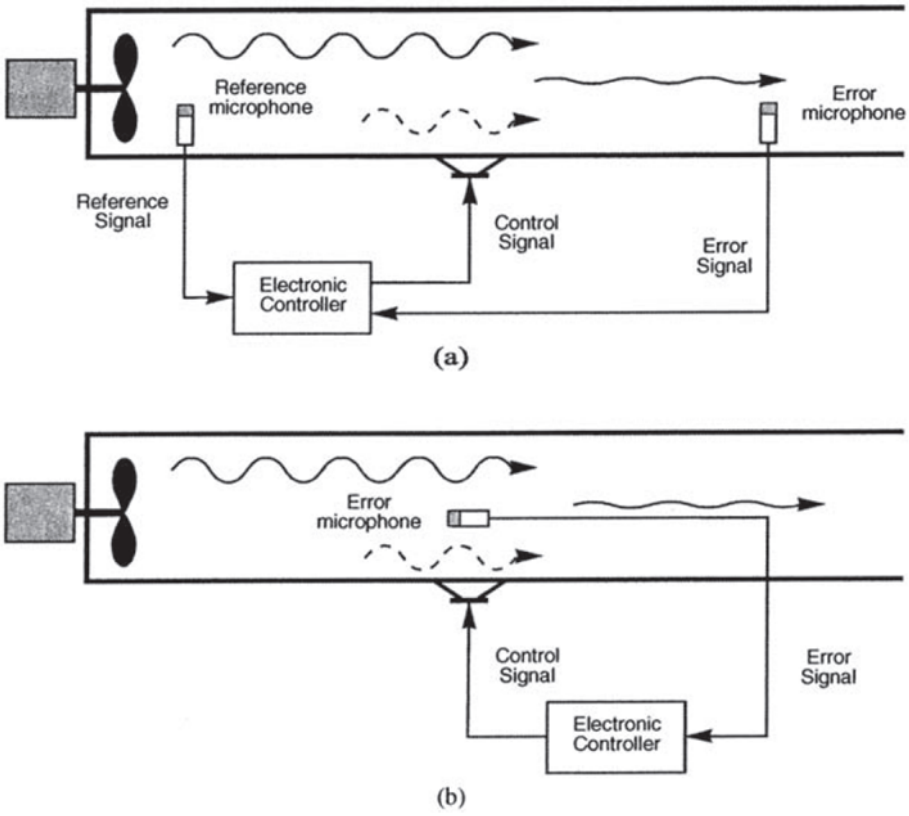
A predictor is a filter that estimates the current signal sample based on past samples. Prediction filters are widely used in voice coding and waveform coding in general, as well as in video coding. Voice signals are highly correlated. By using a prediction filter, we can eliminate the correlation between samples, resulting in a prediction residual that is coded and transmitted. The residual typically has a lower amplitude than the original signal and generally requires fewer bits for transmission. The prediction filter could have fixed coefficients, in which case we refer to it as Differential Pulse Code Modulation (DPCM) coding, utilized, for example, in DECT telephony. More often, however, the predictor is estimated sample by sample using an adaptive filter, leading to Adaptive Differential Pulse Code Modulation (ADPCM) coding.



Note how in the figures, the prediction is carried out starting from the reconstructed signal $x_q(n)$. This ensures that the same operations are performed in both transmission and reception.

11.02.6 Active noise control

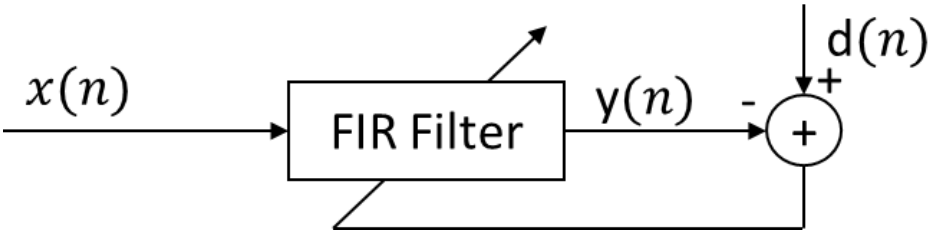
Another application is active noise control or vibration control. The controller is, once again, an adaptive filter that adapts based on an input signal (the reference signal) and/or an error signal (the difference between the primary noise and the secondary noise).
In feedback control, the input signal is generated by the error signal itself. Here lies a significant difference from all previous cases. It's important to note that there is always an acoustic path between the controller output and the error microphone, which has its own transfer function. This path includes a filter, the secondary path, connecting the output of the controller to the error microphone. This filter complicates the treatment.
We will now study several adaptive filters, focusing on a system identification scenario.



11.03 Optimal Wiener filter

How can we mathematically obtain the filters that identify an unknown system, predict a signal, cancel echo, or equalize a channel? The first person to pose this question was Norbert Wiener. This occurred during the middle of World War II, while Wiener was working for MIT. His particular problem was the aiming of an anti-aircraft cannon. The objective was to fire the shots in such a way as to minimize the error between the point of explosion and the target's position. Norbert Wiener developed a theory known as the Wiener optimal filter, which forms the foundation of all adaptive filtering.

Let us consider the scheme of our adaptive filter, with input $x(n)$, output $y(n)$ and desired signal $d(n)$



The theory of the optimal Wiener filter applies to signals $x(n)$ and $d(n)$ which are stationary and zero mean. As optimization criterion it applies the Minimum Mean Square Error (MMSE) criterion, i.e., it estimates the constant coefficients filter that minimizes

$$J = E[e^2(n)] = E\left[(d(n) - y(n))^2\right]$$

Assume the filter to be an FIR filter of length N and consider its vector form:

$$\begin{aligned}
\mathbf{x}(n) &= [x(n), x(n-1), \ldots, x(n-N+1)]^T \\
\mathbf{h} &= [h(0), h(1), \ldots, h(N-1)]^T \\
y(n) &= \mathbf{x}^T(n) \cdot \mathbf{h} = \mathbf{h}^T \cdot \mathbf{x}(n) \\
e(n) &= d(n) - y(n) \\
e^2(n) &= d^2(n) - 2d(n)y(n) + y^2(n) \\
&= d^2(n) - 2\mathbf{h}^T\mathbf{x}(n)d(n) + \mathbf{h}^T\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{h} \\
E[e^2(n)] &= E[d^2(n)] - 2\mathbf{h}^T E[\mathbf{x}(n)d(n)] + \mathbf{h}^T E[\mathbf{x}(n)\mathbf{x}^T(n)]\mathbf{h}
\end{aligned}$$

Let

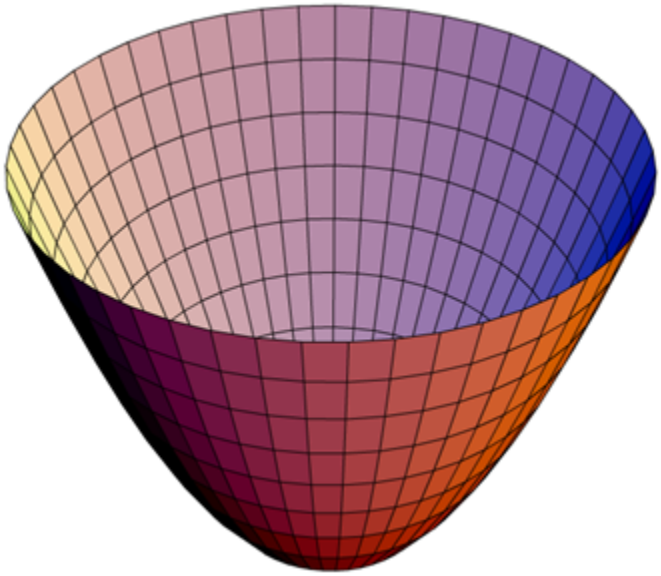
$$\begin{aligned}
\mathbf{R}_{xx} &= E[\mathbf{x}(n)\mathbf{x}^T(n)] \\
&= E\begin{bmatrix}
x(n)x(n) & x(n)x(n-1) & \ldots & x(n)x(n-N+1) \\
x(n-1)x(n) & x(n-1)x(n-1) & \ldots & x(n-1)x(n-N+1) \\
\vdots & \ldots & & \ldots \\
x(n-N+1)x(n) & x(n-N+1)x(n-1) & \ldots & x(n-N+1)x(n-N+1)
\end{bmatrix}
\end{aligned}$$

be the **autocorrelation matrix** of the signal $x(n)$ (an $N \times N$ matrix), and

$$\begin{aligned}
\mathbf{R}_{xd} &= E[\mathbf{x}(n)d(n)] \\
&= E\begin{bmatrix}
x(n)d(n) \\
x(n-1)d(n) \\
\vdots \\
x(n-N+1)d(n)
\end{bmatrix}
\end{aligned}$$

be the **cross-correlation vector**.

Assuming that the autocorrelation matrix and the cross-correlation vector are known, we need to minimize $E[e^2(n)]$ with respect to $\mathbf{h}$. It's important to note that $E[e^2(n)]$ represents a quadratic function of the coefficients, meaning it forms a paraboloid, and the minimum of this paraboloid represents our solution.



Let us set to zero the gradient of $E[e^2(n)]$:

$$\nabla_{\mathbf{h}} E[e^2(n)] = \left[\begin{array}{cccc} \frac{\partial E[e^2(n)]}{\partial h(0)} & \frac{\partial E[e^2(n)]}{\partial h(1)} & \cdots & \frac{\partial E[e^2(n)]}{\partial h(N-1)} \end{array} \right] = 0$$

We get

$$\nabla_{\mathbf{h}} E[e^2(n)] = -2\mathbf{R}_{xd} + 2\mathbf{R}_{xx}\mathbf{h}_{\text{opt}} = 0$$

i.e.,

$$\boxed{\mathbf{h}_{\text{opt}} = \mathbf{R}_{xx}^{-1}\mathbf{R}_{xd},}$$

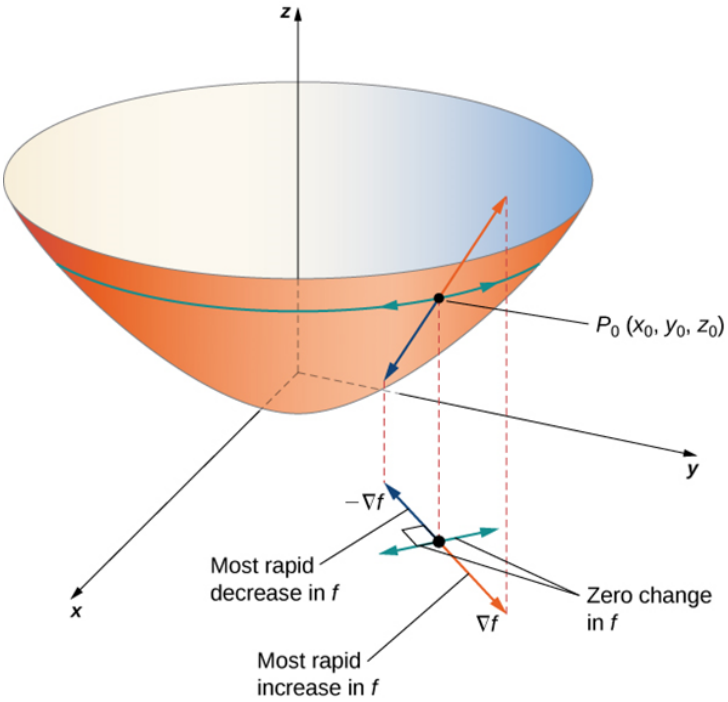
which is the *Wiener-Hopf equation*.
According to the Wiener-Hopf equation, if we know the signal statistics, i.e., the autocorrelation matrix and the cross-correlation matrix, we can compute the Wiener optimal filter for our process. In reality, the signal statistics are not known a priori in general, and often the aleatory processes are not even stationary. It becomes necessary to estimate the signal statistics as we observe the signal. Two approaches are possible:

1. Block algorithms, and
2. Recursive algorithms.

In block algorithms, the signals are divided into data windows that are sufficiently long to obtain a good estimate of $\mathbf{R}_{xx}$ and $\mathbf{R}_{xd}$, but at the same time sufficiently short to assume the signals are stationary. In each window, $\mathbf{R}_{xx}$ and $\mathbf{R}_{xd}$ are computed, and the optimal filter is estimated.
In recursive algorithms, the estimate of the filter is updated sample by sample based on the input signal, the error signal, and the previous estimate of the filter.
In the following, we will consider only recursive algorithms.

11.04 Least mean square (LMS) adaptive filter of Widrow and Hoff

The Least mean square (LMS) adaptive algorithm is the most famous and simplest adaptive algorithm. Sample by sample, it allows obtaining an estimate of the optimal filter, which for stationary systems is proven to converge to the Wiener optimal filter. Remember that $E[e^2(n)]$ forms a paraboloid in the coefficients, and the optimal filter corresponds to the minimum of this paraboloid. The LMS algorithm adjusts the coefficients in the direction of the maximum slope to reach the minimum within a certain number of steps. In cases where the signals are not stationary, the paraboloid will change shape and shift. Nonetheless, the LMS algorithm will track the minimum of the paraboloid, always moving in the direction of maximum slope.



We move in the opposite direction of the gradient.

$$\mathbf{h}_{n+1} = \mathbf{h}_n - \frac{\mu}{2}\nabla_{\mathbf{h}} E[e^2(n)]$$

where μ is a small step-size. We have to estimate $\nabla_{\mathbf{h}} E[e^2(n)]$, which still depends on the signal statistics. Widrow and Hoff developed the LMS algorithm considering a drastic but effective approximation:

$$\nabla_{\mathbf{h}} E[e^2(n)] \simeq \nabla_{\mathbf{h}} e^2(n) = 2e(n)\nabla_{\mathbf{h}} e(n)$$

where

$$e(n) = d(n) - \mathbf{h}^T\mathbf{x}(n)$$

Since $\nabla_{\mathbf{h}} d(n) = 0$ and $\nabla_{\mathbf{h}} \mathbf{h}^T\mathbf{x}(n) = \mathbf{x}(n)$, $\nabla_{\mathbf{h}} e(n) = -\mathbf{x}(n)$. Thus, we obtain

$$\boxed{\begin{aligned} e(n) &= d(n) - \mathbf{h}_n^T\mathbf{x}(n), \\ \mathbf{h}_{n+1} &= \mathbf{h}_n + \mu e(n)\mathbf{x}(n), \end{aligned}}$$

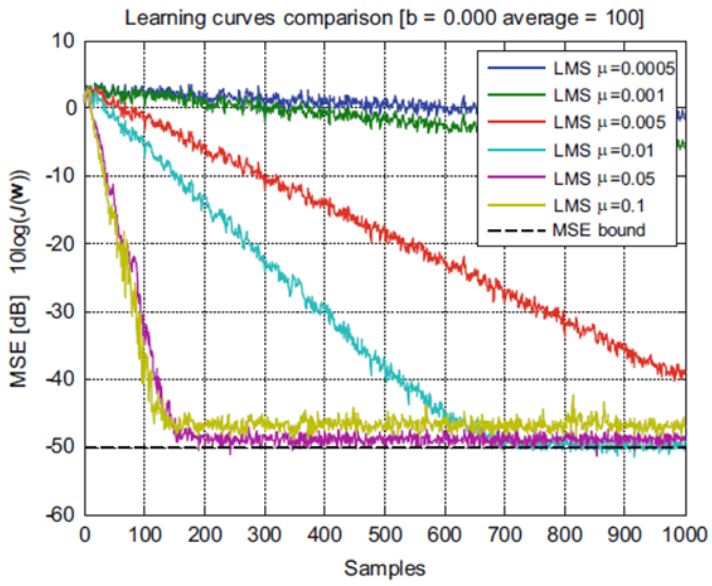
The algorithm requires $2N$ multiplications and $2N$ additions, thus having a very low computational cost. In the case of stationary signals, it can be proven that the algorithm converges to the optimal filter provided that $0 < \mu < 2/\lambda_{\max}$, where $\lambda_{\max}$ is the largest eigenvalue of $\mathbf{R}xx$. Since $\lambda\max \leq \text{Trace}(\mathbf{R}_{xx})$, a sufficient condition for convergence is

$$0 < \mu < \frac{2}{\sum_{k=0}^{N-1} E[x^2(n-k)]}$$

which is easier to approximate. Without convergence, the algorithm diverges to $\pm\infty$ within a few samples.

Let's consider the learning curves of the mean square error, i.e., the ensemble average over a certain number of runs of $E[e^2(n)]$ (obtained by averaging over a certain interval), for different step-sizes μ. The larger μ is, the faster the convergence speed, but also the larger the oscillations around the optimal solution.

For a white Gaussian noise input signal, we have:



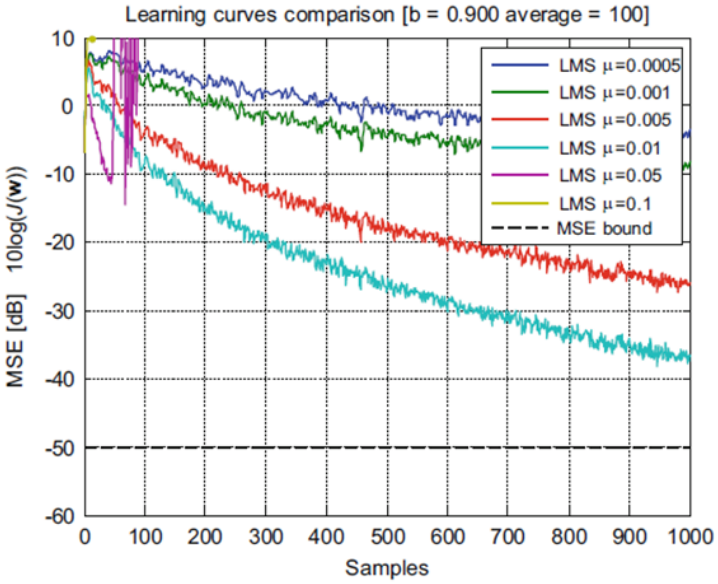
[Figure taken from A. Uncini "Fundamental of adaptive signal processing", Springer, 2015]

A drawback of the LMS algorithm is its slow convergence speed in the case of correlated input signals. The greater the signal correlation, the slower the convergence speed.

With a narrowband moving average colored process:

$$x(n) = bx(n-1) + u(n)$$

with $u(n)$ a white Gaussian noise signal, we have



[Figure taken from A. Uncini "Fundamental of adaptive signal processing", Springer, 2015]

The speed of convergence can be improved by decorrelating the signal. This concept will become evident as we explore the RLS and APA algorithms.

Several other algorithms have been developed based on adaptation rules similar to those of the LMS algorithm.

11.05 Sign-error adaptive filter

The Sign-error adaptive filter is derived from a manipulation of the LMS algorithm:

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \mu \, \text{sign}[e(n)]\mathbf{x}(n),$$

We continue to move in the direction of the gradient, but we lose the ability to control the amplitude of the movements with the error.

In reality, the algorithm optimizes the mean absolute error:

$$J = E[|e(n)|].$$

Generally, it has a lower convergence speed than LMS for the same steady-state MSE, and for the same convergence speed, it has a larger steady-state MSE. Its main advantage lies in its computational efficiency. If $\mu = 2^{-K}$, then adaptation with fixed-point arithmetic requires only additions and shifts, making it suitable for hardware implementation.

The algorithm is also always numerically stable; it never diverges to $\pm\infty$.

11.06 Normalized LMS (NLMS) adaptive filter

The NLMS (Normalized Least Mean Square) is the most famous and widely used adaptive filter.
It was initially derived by manipulating the adaptation rule of the LMS algorithm.
Consider the LMS adaptation rule:

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \mu e(n)\mathbf{x}(n),$$
$$e(n) = d(n) - \mathbf{h}^T\mathbf{x}(n),$$

If the amplitude of the input signal $x(n)$ reduces by a factor of 2, then $e(n)$ also reduces by the same factor, while $\Delta\mathbf{h}_n = \mathbf{h}_{n+1} - \mathbf{h}_n$, and the convergence speed decreases by a factor of 4. To address this issue, a modification to the adaptation rule has been proposed:

$$\boxed{\mathbf{h}_{n+1} = \mathbf{h}_n + \frac{\mu e(n)}{\mathbf{x}^T(n)\cdot\mathbf{x}(n) + \delta}\mathbf{x}(n),}$$

where δ is a small positive constant used to avoid division by 0. This modification ensures that $\Delta\mathbf{h}_n$ becomes independent of the signal's amplitude, providing additional benefits. It can be proven that in the case of stationary signals, the NLMS filter converges to the Wiener optimal filter for any step-size satisfying $0 < \mu < 2$. Ensuring this convergence condition is extremely simple.
The NLMS algorithm:

$$\boxed{\begin{aligned} e(n) &= d(n) - \mathbf{h}_n^T\mathbf{x}(n), \\ \mathbf{h}_{n+1} &= \mathbf{h}_n + \frac{\mu e(n)}{\mathbf{x}^T(n)\cdot\mathbf{x}(n) + \delta}\mathbf{x}(n), \end{aligned}}$$

Even though the NLMS algorithm was initially derived from the LMS algorithm, it was later determined to be the exact solution of the following optimization criterion:

> Minimize the Euclidean norm of the coefficient variation $\Delta\mathbf{h}_n = \mathbf{h}_{n+1} - \mathbf{h}_n$ imposing the a posteriori error $e_{n+1}(n) = d(n) - \mathbf{h}_{n+1}^T\mathbf{x}(n)$ to be zero, i.e.,
>
> $$\min\{\Delta\mathbf{h}_n^T\Delta\mathbf{h}_n\} \text{ s.t. } e_{n+1}(n) = d(n) - \mathbf{h}_{n+1}^T\mathbf{x}(n) = 0.$$

We will revisit this criterion when discussing the Affine Projection algorithms.
The computational cost of the NLMS algorithm is similar to that of the LMS algorithm ($2N$ multiplications and $2N$ additions), and its convergence speed is also comparable. Like the LMS algorithm, it suffers from poor convergence speed with correlated signals.
If the slow convergence speed is a concern, one option is to utilize one of the RLS algorithms.

11.07 Recursive Least Square (RLS) adaptive filter

There is a full family of RLS algorithms. We will explore the parent algorithm within this family. In this algorithmic family, sample by sample, the following cost function is optimized:

$$J(n) = \sum_{k=-\infty}^{n} \lambda^{n-k} e^2(k) = \sum_{k=-\infty}^{n} \lambda^{n-k} \left[d(n) - \mathbf{h}_n^T\mathbf{x}(k)\right]^2.$$

These algorithms directly approximate the stochastic cost function $E[e^2(n)]$ with a deterministic expression. Note that the terms $e^2(n)$ are weighted by an exponential window function λ^{n-k}, where λ, known as the *forgetting factor*, is chosen such that $0 < \lambda \lesssim 1$. Thus, the most recent samples are weighted more heavily than the older ones.
By properly choosing λ, we are able to track time-varying systems as well.
Let us minimize $J(n)$ as a function of $\mathbf{h}_n$. Proceeding similarly to the optimal Wiener filter, we impose

$$\nabla_{\mathbf{h}_n} J(n) = 0.$$

$$\begin{aligned}
J(n) &= \sum_{k=-\infty}^{n} \lambda^{n-k} \left[d^2(k) - 2\mathbf{h}_n^T \mathbf{x}(k) + \mathbf{h}_n^T \mathbf{x}(k)\mathbf{x}^T(k)\mathbf{h}_n \right] \\
\nabla_{\mathbf{h}_n} J(n) &= 2\mathbf{R}_{xx}(n) \cdot \mathbf{h}_n - 2\mathbf{R}_{xd}(n), \qquad \text{where} \\
\mathbf{R}_{xx}(n) &= \sum_{k=-\infty}^{n} \lambda^{n-k} \mathbf{x}(k)\mathbf{x}^T(k), \\
\mathbf{R}_{xd}(n) &= \sum_{k=-\infty}^{n} \lambda^{n-k} \mathbf{x}(k) d(k), \\
\mathbf{x}(k) &= [x(k), x(k-1), \ldots, x(k-N+1)]^T.
\end{aligned}$$

Then we have

$$\boxed{\mathbf{h}_n = \mathbf{R}_{xx}^{-1}(n) \cdot \mathbf{R}_{xd}(n)}$$

which is analogous to the Wiener-Hopf equation.
We can estimate $\mathbf{h}_n$ by updating sample by sample:

$$\begin{aligned}
\mathbf{R}_{xx}(n) &= \sum_{k=-\infty}^{n} \lambda^{(n-k)} \mathbf{x}(k)\mathbf{x}^T(k) \\
&= \lambda \cdot \mathbf{R}_{xx}(n-1) + \mathbf{x}(n)\mathbf{x}^T(n), \\
\mathbf{R}_{xd}(n) &= \lambda \cdot \mathbf{R}_{xd}(n-1) + \mathbf{x}(n) d(n).
\end{aligned}$$

However, the matrix inversion $\mathbf{R}_{xx}^{-1}(n)$ in $\mathbf{h}_n = \mathbf{R}_{xx}^{-1}(n) \cdot \mathbf{R}_{xd}(n)$ is computationally expensive, requiring $O(N^3)$ operations. To mitigate this, we can reduce the number of computations by directly updating $\mathbf{R}_{xx}^{-1}(n)$ using the *Matrix Inversion Lemma*:

> If A is $N \times N$ and invertible, B is $N \times M$, C is $M \times M$, D is $M \times N$, then:
>
> $$[A + B \cdot C \cdot D]^{-1} = A^{-1} - A^{-1}B \left[D \cdot A^{-1} \cdot B + C \right]^{-1} DA^{-1}$$

From $\mathbf{R}_{xx}(n) = \lambda \cdot \mathbf{R}_{xx}(n-1) + \mathbf{x}(n)\mathbf{x}^T(n)$, setting $A = \lambda \cdot \mathbf{R}_{xx}(n-1)$, $B = \mathbf{x}(n)$, $C = I$, $D = \mathbf{x}^T(n)$, we have

$$\mathbf{R}_{xx}^{-1}(n) = \lambda^{-1}\mathbf{R}_{xx}^{-1}(n-1) - \lambda^{-1}\mathbf{R}_{xx}^{-1}(n-1)\mathbf{x}(n) \left[\mathbf{x}^T(n)\lambda^{-1}\mathbf{R}_{xx}^{-1}(n-1)\mathbf{x}(n) + 1 \right]^{-1} \mathbf{x}^T(n)\lambda^{-1}\mathbf{R}_{xx}^{-1}(n-1)$$

$$\mathbf{R}_{xx}^{-1}(n) = \frac{1}{\lambda}\mathbf{R}_{xx}^{-1}(n-1) - \frac{\frac{1}{\lambda^2}\mathbf{R}_{xx}^{-1}(n-1)\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{R}_{xx}^{-1}(n-1)}{1 + \frac{1}{\lambda}\mathbf{x}^T(n)\mathbf{R}_{xx}^{-1}(n-1)\mathbf{x}(n)}$$

Moreover,

$$\begin{aligned}\mathbf{h}_n &= \mathbf{R}_{xx}^{-1}(n) \cdot \mathbf{R}_{xd}(n) \\ &= \mathbf{R}_{xx}^{-1}(n) \cdot [\lambda \mathbf{R}_{xd}(n-1) + \mathbf{x}(n)d(n)] \\ &= \mathbf{R}_{xx}^{-1}(n) \cdot [\lambda \mathbf{R}_{xx}(n-1)\mathbf{h}_{n-1} + \mathbf{x}(n)d(n)] \\ &= \mathbf{R}_{xx}^{-1}(n) \cdot [(\mathbf{R}_{xx}(n) - \mathbf{x}(n)\mathbf{x}^T(n))\mathbf{h}_{n-1} + \mathbf{x}(n)d(n)] \\ &= \mathbf{h}_{n-1} + \mathbf{R}_{xx}^{-1}(n)\mathbf{x}(n)(d(n) - \mathbf{x}^T(n)\mathbf{h}_{n-1})\end{aligned}$$

Setting

$$\begin{aligned}e(n) &= d(n) - \mathbf{x}^T(n)\mathbf{h}_{n-1}, \text{ the a priori error, and} \\ \mathbf{k}(n) &= \mathbf{R}_{xx}^{-1}(n)\mathbf{x}(n), \text{ the } \textit{Kalman Gain Vector}\text{, we get} \\ \mathbf{h}_n &= \mathbf{h}_{n-1} + e(n)\mathbf{k}(n)\end{aligned}$$

Multiplying the expression of $\mathbf{R}_{xx}^{-1}(n)$ by $\mathbf{x}(n)$, we can find an update rule for $\mathbf{k}(n)$. Moreover, by collecting $\mathbf{k}(n)$ in the expression of $\mathbf{R}_{xx}^{-1}(n)$, we can simplify it. Therefore, we obtain the RLS algorithm as shown in the following table:

$$\begin{aligned}e(n) &= d(n) - \mathbf{x}^T(n)\mathbf{h}_{n-1} \\ k(n) &= \frac{\mathbf{R}_{xx}^{-1}(n-1)\mathbf{x}(n)}{\lambda + \mathbf{x}^T(n)\mathbf{R}_{xx}^{-1}(n-1)\mathbf{x}(n)} \\ \mathbf{h}_n &= \mathbf{h}_{n-1} + e(n)\mathbf{k}(n) \\ \mathbf{R}_{xx}^{-1}(n) &= \frac{1}{\lambda}[\mathbf{R}_{xx}^{-1}(n-1) - \mathbf{k}(n)\mathbf{x}^T(n)\mathbf{R}_{xx}^{-1}(n-1)]\end{aligned}$$

The algorithm has initial conditions $\mathbf{h}_{-1} = 0$ and $\mathbf{R}_{xx}(-1) = c\mathbf{I}_N$, where c is a small positive constant and $\mathbf{I}_N$ is the identity matrix.

The RLS algorithm has a computational cost of order N^2 operations.

It has a high convergence speed, but its tracking ability is worse than LMS due to the exponential windowing.

If we compare the update rules of LMS and RLS, we see that

$$\begin{aligned}\Delta\mathbf{h}_n &= \mathbf{h}_{n+1} - \mathbf{h}_n = \mu\mathbf{x}(n)e(n) && \text{for LMS} \\ \Delta\mathbf{h}_n &= \mathbf{h}_{n+1} - \mathbf{h}_n = \mathbf{R}_{xx}^{-1}(n)\mathbf{x}(n)e(n) && \text{for RLS}\end{aligned}$$

The effect of the inverse autocorrelation matrix is to decorrelate the input signal. Thanks to this decorrelation, the RLS algorithm has a higher convergence speed than the LMS algorithm.

The inverse autocorrelation matrix is symmetric and positive definite. If, in the implementation, due to error propagation, $\mathbf{R}_{xx}^{-1}(n)$ loses its symmetry, the algorithm becomes unstable. Many algorithms have been proposed to solve this problem.

Eventually, it was proven that ensuring the symmetry of $\mathbf{R}_{xx}^{-1}(n)$ is sufficient to guarantee the algorithm's stability.

The main limitation of the original RLS algorithm is its high computational cost, which is of order N^2. In the 1990s, many *Fast-RLS* algorithms with computational complexity of order N were proposed. The first ones were numerically unstable, but later many fast and stable RLS algorithms were developed.

Nevertheless, the computational complexity remains high compared to the LMS algorithm. It involves at least $8N \div 10N$ multiplications and often requires order N divisions.

11.07.1 Affine projection algorithms (APA)

The affine projection algorithms (APA) are a family of adaptive filters that offer a high convergence speed (although slower than RLS) with computational complexity only slightly higher than LMS. They are designed to minimize the Euclidean norm of the coefficient variation $\Delta\mathbf{h}_n = \mathbf{h}_{n+1} - \mathbf{h}_n$, while ensuring the last P a posteriori errors are zero. This optimization criterion is expressed as:

$$\min \Delta\mathbf{h}_n^T \Delta\mathbf{h}_n \quad \text{subject to}$$

$$e_{n+1}(n) = d(n) - \mathbf{h}_{n+1}^T \cdot \mathbf{x}(n) = 0$$

$$e_{n+1}(n-1) = d(n-1) - \mathbf{h}_{n+1}^T \cdot \mathbf{x}(n-1) = 0$$

$$\vdots$$

$$e_{n+1}(n-P+1) = d(n-P+1) - \mathbf{h}_{n+1}^T \cdot \mathbf{x}(n-P+1) = 0$$

For $P = 1$, the NLMS algorithm is obtained. As P increases, the convergence speed also increases. Often, even for small P, a high convergence speed is achieved.
To solve this optimization problem, the *Lagrange multiplier method* is employed, where:

$$J = \Delta\mathbf{h}_n^T \cdot \Delta\mathbf{h}_n + \sum_{j=0}^{P-1} \lambda_j \left[d(n-j) - \mathbf{h}_{n+1}^T \cdot \mathbf{x}(n-j) \right] \tag{11.01}$$

Here, λ_j represents the Lagrange multipliers.
By imposing $\nabla_{\mathbf{h}_{n+1}} J = 0$, we obtain:

$$2\Delta\mathbf{h}_n = \sum_{j=0}^{P-1} \lambda_j \mathbf{x}(n-j) = \mathbf{G}(n) \cdot \Lambda \quad \text{with}$$

$$\mathbf{G}(n) = [\mathbf{x}(n), \mathbf{x}(n-1), \ldots, \mathbf{x}(n-P+1)]$$

$$\Lambda = [\lambda_0, \lambda_1, \ldots, \lambda_{P-1}]^T$$

Pre-multiplication by $\mathbf{G}^T(n)$ yields:

$$\Lambda = [\mathbf{G}^T(n)\mathbf{G}(n)]^{-1} \mathbf{G}^T(n) 2\Delta\mathbf{h}_n$$

Utilizing the relation

$$\mathbf{x}^T(n-j)\Delta\mathbf{h}_n = \mathbf{x}^T(n-j)\mathbf{h}_{n+1} - \mathbf{x}^T(n-j)\mathbf{h}_n = d(n-j) - \mathbf{x}^T(n-j)\mathbf{h}_n$$

we obtain

$$\mathbf{G}^T(n)\Delta\mathbf{h}_n = \begin{bmatrix} \mathbf{x}^T(n)\Delta\mathbf{h}_n \\ \vdots \\ \mathbf{x}^T(n-P+1)\Delta\mathbf{h}_n \end{bmatrix} = \begin{bmatrix} d(n) - \mathbf{x}^T(n)\mathbf{h}_n \\ \vdots \\ d(n-P+1) - \mathbf{x}^T(n-P+1)\mathbf{h}_n \end{bmatrix} = \mathbf{e}(n),$$

which is the a priori error vector.
Finally, the update rule for $\mathbf{h}_{n+1}$ is given by:

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \frac{1}{2}\mathbf{G}(n) \cdot \Lambda = \mathbf{h}_n + \mathbf{G}(n)[\mathbf{G}^T(n)\mathbf{G}(n)]^{-1}\mathbf{e}(n)$$

In practice, a step-size and a small regularization factor are often introduced, leading to the following updated rule:

$$\boxed{\mathbf{h}_{n+1} = \mathbf{h}_n + \mu\mathbf{G}(n)[\mathbf{G}^T(n)\mathbf{G}(n) + \delta\mathbf{I}]^{-1}\mathbf{e}(n)}$$

The algorithm exhibits a computational complexity of order $P \cdot N$, although fast algorithms have been developed to reduce it to order N.

11.08 Decorrelation NLMS (DNLMS) algorithm

This adaptation algorithm represents an evolution of the NLMS algorithm, enhancing the convergence speed by decorrelating the input data vector. With a first-order decorrelation, the DLMS algorithm yields:

$$\boxed{\begin{aligned} e(n) &= d(n) - \mathbf{h}_n^T \cdot \mathbf{x}(n) \\ c(n) &= \frac{\mathbf{x}^T(n)\mathbf{x}(n-1)}{\mathbf{x}^T(n-1)\mathbf{x}(n-1)} \\ \mathbf{z}(n) &= \mathbf{x}(n) - c(n)\mathbf{x}(n-1) \\ \mathbf{h}_{n+1} &= \mathbf{h}_n + \frac{\mu}{\mathbf{z}^T(n) \cdot \mathbf{x}(n) + \delta} e(n)\mathbf{z}(n) \end{aligned}}$$

Although different, it exhibits similar convergence behavior to that of a second-order APA.

11.09 Perfect periodic sequences

Note that all the algorithms we have studied—NLMS, RLS, APA, DNLMS—exhibit different convergence speeds in the presence of colored aleatory inputs. However, they all reach the same maximum convergence speed for a white Gaussian input. When identifying a system and given the option to choose the input signal, the common approach is to select a white Gaussian input and apply the NLMS algorithm. Nevertheless, in the early '90s, German researcher Dr. Christiane Antweiler proved that the signal

optimizing the convergence speed of the NLMS algorithm is not white noise, but rather a deterministic perfect periodic sequence (PPS). A PPS is a periodic sequence $p(n)$ of period N that satisfies the following condition:

$$\sum_{i=0}^{N-1} p(n)p(n+i) = \begin{cases} \neq 0 \text{ if } i = 0 \\ = 0 \text{ otherwise} \end{cases}$$

There exist infinitely many PPSs. Some are formed by a fundamental period composed only of $\{-1, +1\}$ with a leading 0. The simplest way to generate a PPS is to consider the Inverse Discrete Fourier Transform (IDFT) of a constant magnitude spectrum with random phase satisfying the conjugate symmetry. For the same residual Mean Squared Error (MSE), the NLMS algorithm with a PPS input consistently achieves a faster convergence speed than the same algorithm with a white Gaussian input. Moreover, in the absence of measurement noise, we can identify the system with only N samples using $\mu = 1$. The computational complexity remains the same as that of the NLMS algorithm—$2N$ multiplications and additions.

12 Multirate Digital Signal Processing

12.01 Introduction

The digital signal processing systems discussed so far belong to the class of single-rate systems, as the sampling rate remains constant at the input, output, and all nodes. There are numerous applications where a signal sampled at one rate needs to be converted to a different rate. For instance, various sampling rates are utilized in digital audio: 32 kHz for broadcasting, 44.1 kHz for CDs, and 48 kHz for digital audio tapes (DATs). Frequently, there's a need to transition between these sample rates. Another example is pitch control in audio recordings. In analog tapes, this is achieved by adjusting the tape recorder speed. However, in digital processing systems, it necessitates altering the sample rate.

To alter the sampling rate of a digital signal, multirate digital signal processing systems employ two fundamental devices: a *downsampler* and an *upsampler*. Systems that operate with discrete-time signals having unequal sampling rates at different points are referred to as *multirate* systems.

12.02 Basic sampling rate alteration devices

First, let's examine the characteristics of the upsampler and downsampler in the time domain, and later, we'll explore them in the frequency domain.

12.02.1 Time domain characterization

An upsampler with an upsampling factor L, where $L \in \mathbb{N}$, generates an output sequence $x_u(n)$ with a sampling rate that is L times higher than that of the input sequence $x(n)$. The upsampling operation is executed by inserting $L-1$ zero-value samples between two consecutive samples of the input sequence $x(n)$, following the relation:

$$x_u(n) = \begin{cases} x(n/L), & n = 0, \pm L, \pm 2L, \dots, \\ 0, & \text{elsewhere.} \end{cases}$$

An illustration of the upsampling process is shown in the following figure:

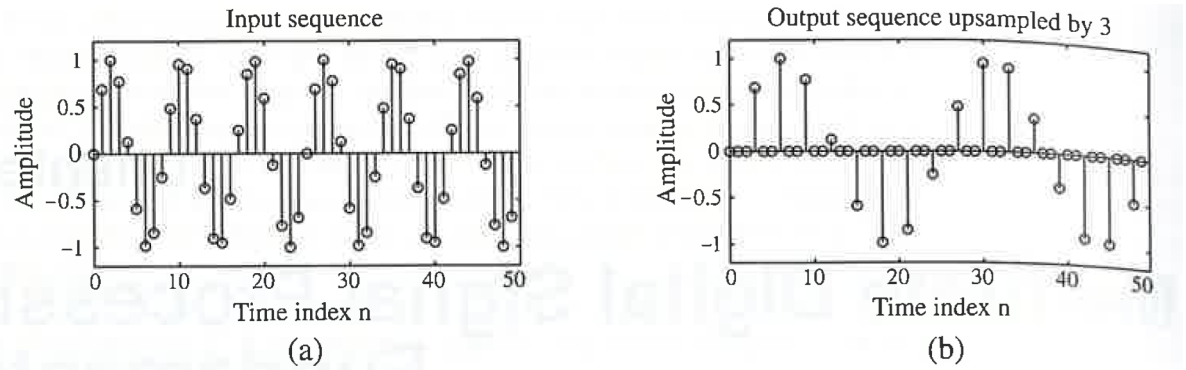


Figure 13.1: Illustration of the up-sampling process.

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

The block diagram representation of the upsampler, also known as a *sampling rate expander*, is shown in the following figure:

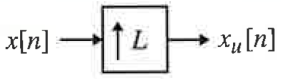


Figure 13.2: Block diagram representation of an up-sampler.

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

In practice, the zero-valued samples inserted by the upsampler are interpolated using some form of filtering process to eliminate any unnecessary spectral components from the higher-rate sequence. This process, known as *interpolation*, will be discussed later.

A downsampler with a downsampling factor M, where $M \in \mathbb{N}$, generates an output sequence $y(n)$ with a sampling rate that is $\frac{1}{M}$ of that of the input sequence $x(n)$. The downsampling operation is executed by retaining every Mth sample of the input sequence and discarding the remaining $M-1$ samples in-between. Therefore, the output sequence is generated according to the relation:

$$y(n) = x(nM).$$

All input samples with indexes equal to an integer multiple of M are retained in the output, while the others are discarded.

The downsampling process is illustrated in the following figure:

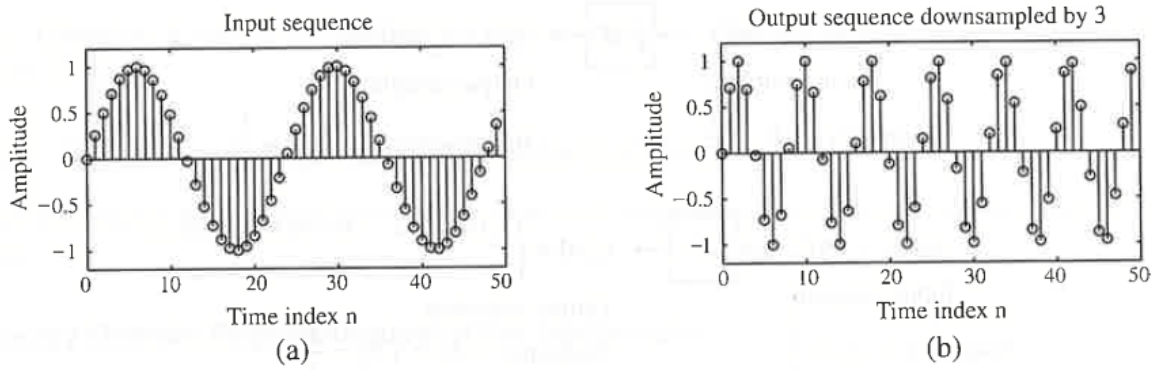


Figure 13.3: Illustration of the down-sampling process.

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

The block diagram representation of the *downsampler*, also known as a *sampling rate compressor*, is shown below:

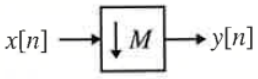


Figure 13.4: Block diagram representation of a down-sampler.

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

The sampling periods are typically not depicted in the block diagrams of the upsampler and downsampler. However, the following figure illustrates how the sampling rate changes in both cases:

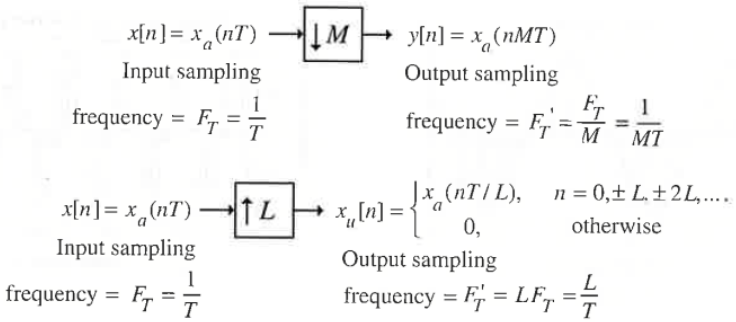


Figure 13.5: The sampling rate alteration building blocks with sampling rates explicitly shown.

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

The upsampler and downsampler building blocks are frequently used together in multirate signal processing applications. One example of using both types of devices is to achieve a sampling rate change by a rational number rather than an integer value.

12.02.2 Frequency domain characterization

We consider the case of the upsampler and derive the relations between the input and output spectra of a factor of 2 upsampler:

$$x_u(n) = \begin{cases} x(n/2), & n = 0, \pm 2, \pm 4, \dots, \\ 0, & \text{elsewhere.} \end{cases}$$

The z-transform is

$$X_u(z) = \sum_{n=-\infty}^{+\infty} x_u(z) z^{-n} = \sum_{\substack{n=-\infty \\ n \text{ even}}}^{+\infty} x(n/2) z^{-n} =$$

$$(n = 2m) = \sum_{m=-\infty}^{+\infty} x(m) z^{-2m} = X(z^2)$$

In a similar manner, we can show that for a factor of L upsampler, we have:

$$X_u(z) = X(z^L).$$

Let's examine what happens on the unit circle, i.e., for $z = e^{j\omega}$. Then, $X_u(e^{j\omega}) = X(e^{j\omega L})$. As ω ranges from 0 to 2π, $e^{j\omega L}$ will traverse around the unit circle L times, implying that the X-axis of the frequency response is compressed by a factor of L.
For instance, the following figure depicts, for simplicity, a real frequency response $X(e^{j\omega})$ with an asymmetric spectrum (hence $x(n) \in \mathbb{C}$), and the resulting upsampled spectrum $X_u(e^{j\omega})$ when $L = 2$:

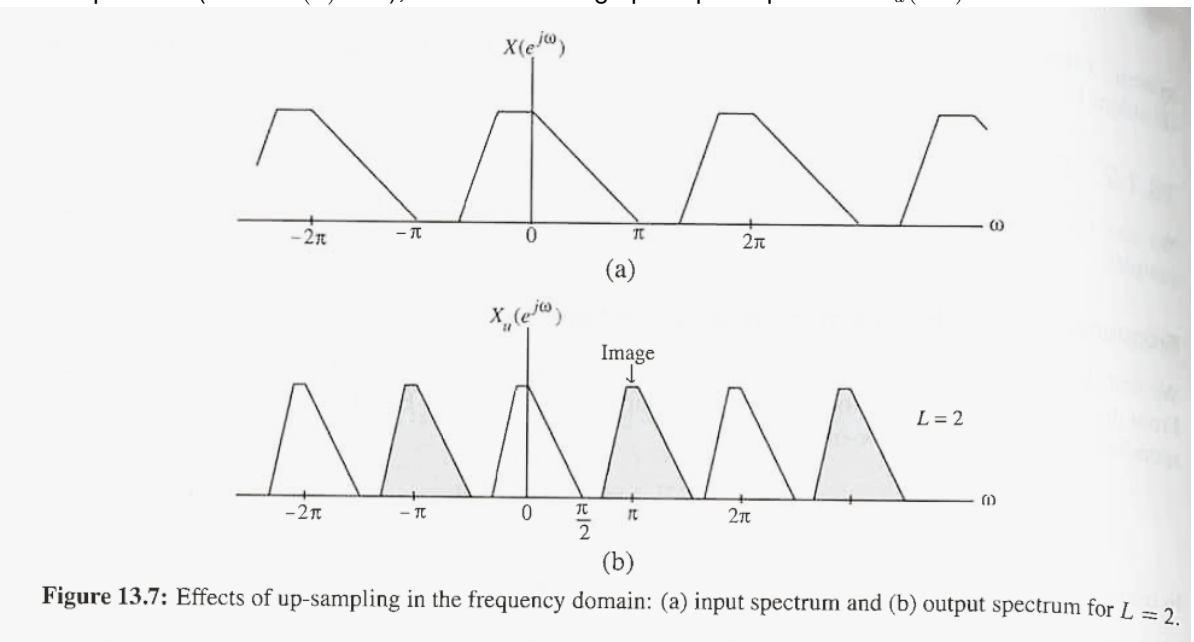


Figure 13.7: Effects of up-sampling in the frequency domain: (a) input spectrum and (b) output spectrum for $L = 2$.

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

We notice that the X-axis has been compressed by a factor of 2, and within the range $[0, 2\pi]$, we now observe two repetitions of $X(e^{j\omega})$. This process is termed imaging because we obtain additional images of the input spectrum. With a factor of L upsampling, we will obtain $L - 1$ additional images of this spectrum. Even if the original spectrum is bandlimited and lowpass, the upsampled spectrum

will not resemble a lowpass spectrum anymore due to the zero-valued samples inserted between the non-zero samples. We can eliminate the images of the original spectrum by employing a lowpass filter with a bandwidth lower than π/L. The effect of such filtering will be to "fill in" the zero-valued samples of $x_u(n)$ with interpolated sample values.
Let us now consider the down-sampler. We apply the z-transform to the input-output relationship, yielding:

$$Y(z) = \sum_{n=-\infty}^{+\infty} x(Mn)z^{-n}$$

This expression cannot be directly expressed in terms of $X(e^{j\omega})$. To address this, we introduce an intermediate signal:

$$x_{\text{int}}(n) = \begin{cases} x(n), & n = 0, \pm M, \pm 2M, \dots \\ 0 & \text{otherwise.} \end{cases}$$

Then, we have:

$$Y(z) = \sum_{n=-\infty}^{+\infty} x(Mn)z^{-n} = \sum_{n=-\infty}^{+\infty} x_{\text{int}}(Mn)z^{-n} =$$
$$(k = Mn) = \sum_{k=-\infty}^{+\infty} x_{\text{int}}(k)z^{-k/M} = X_{\text{int}}(z^{1/M}).$$

$x_{\text{int}}(n)$ can be related to $x(n)$, since

$$x_{\text{int}}(n) = c(n)x(n)$$

where

$$c(n) = \begin{cases} 1, & n = 0, \pm M, \pm 2M, \dots \\ 0 & \text{otherwise,} \end{cases}$$

and

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

Then, we have:

$$X_{\text{int}}(z) = \sum_{n=-\infty}^{+\infty} c(n)x(n)z^{-n} = \frac{1}{M} \sum_{n=-\infty}^{+\infty} \sum_{k=0}^{M-1} e^{j\frac{2\pi}{M}kn} x(n)z^{-n}$$
$$= \frac{1}{M} \sum_{k=0}^{M-1} \sum_{n=-\infty}^{+\infty} x(n)(e^{-j\frac{2\pi}{M}k}z)^{-n} = \frac{1}{M} \sum_{k=0}^{M-1} X(e^{-j\frac{2\pi}{M}k}z)$$

and

$$Y(z) = \frac{1}{M} \sum_{k=0}^{M-1} X(e^{-j\frac{2\pi}{M}k}z^{1/M}).$$

To understand the implications of this relation, let's consider a factor of 2 downsampler and the resulting output spectrum:

$$Y(e^{j\omega}) = \frac{1}{2} \sum_{k=0}^{1} X(e^{-j\frac{2\pi}{2}k}e^{j\omega/2}) = \frac{1}{2} \left[X(e^{j\omega/2}) + X(e^{j(\omega-2\pi)/2}) \right]$$

$X(e^{j\omega/2})$ represents the spectrum of the original signal, which has been stretched on the X-axis by a factor of 2 and is now periodic with a period of 4π. $X(e^{j(\omega-2\pi)/2})$ is the same spectrum but time-shifted by 2π. In general, these two spectra overlap, resulting in an aliasing error caused by this overlap.

The situation is illustrated in the following figure:

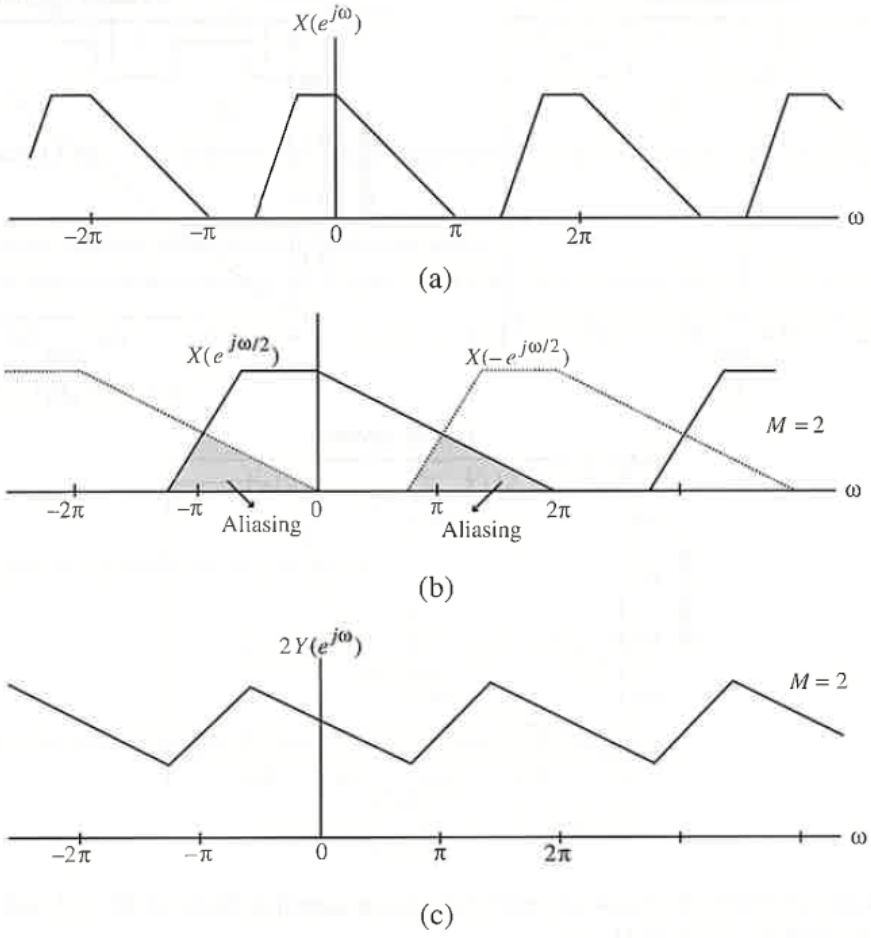


Figure 13.10: Illustration of the aliasing effect in the frequency domain caused by down-sampling.

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

The absence of overlap, and hence no aliasing, occurs only if $X(e^{j\omega}) = 0$ for $|\omega| > \pi/2$.

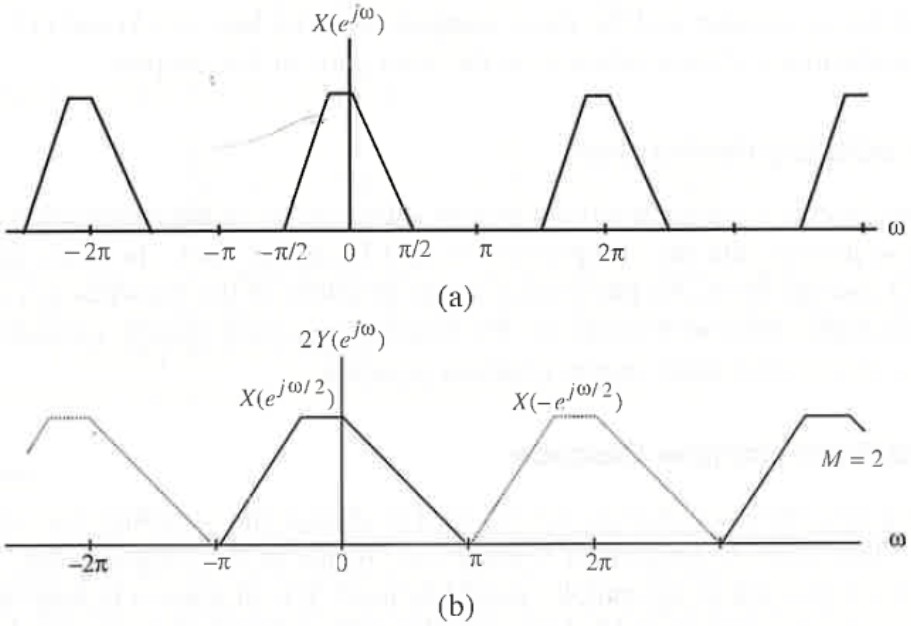


Figure 13.11: Effect of down-sampling in the frequency domain illustrating absence of aliasing.

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

For the general case, the situation remains the same. With a factor of M downsampler,

$$Y(e^{j\omega}) = \frac{1}{M}\sum_{k=0}^{M} X(e^{\frac{j(\omega-2\pi k)}{M}}).$$

The spectrum $Y(e^{j\omega})$ consists of M uniformly shifted and stretched versions of $X(e^{j\omega})$, each stretched by a factor of M.
Aliasing is absent if and only if $X(e^{j\omega}) = 0$ for $|\omega| > \pi/M$.

12.02.3 Cascade equivalences

Complex multirate systems are formed by an interconnection of basic sampling rate alteration devices and the components of an LTI digital filter. In many applications, these devices appear in cascade form. Interchanging the positions of the branches in a cascade can often lead to a computationally efficient realization.
We will now consider some identities that we will exploit later on.

Up-sampler and down-sampler cascade

The basic sampling rate alteration devices can only up-sample or down-sample a signal by an integer factor. To implement a fractional change in sampling rate, a cascade of an upsampler and downsampler should be used. It is interesting to determine under which conditions the cascade of a factor M downsampler and a factor L upsampler are interchangeable, with no change in the input-output relation.

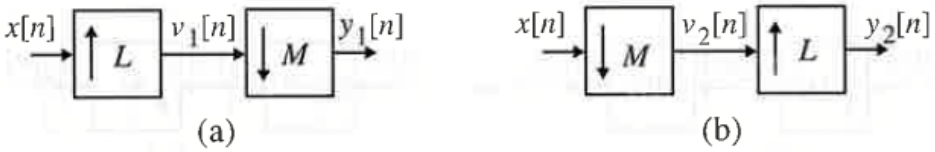


Figure 13.13: Two different cascade arrangements of a down-sampler and an up-sampler.

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

Consider first the case of the upsampler followed by the downsampler, denoted as case (a) in the figure:

$$
\begin{aligned}
V_1(z) &= X(z^L) \\
Y_1(z) &= \frac{1}{M}\sum_{k=0}^{M-1} V_1(z^{1/M} e^{-j\frac{2\pi}{M}k}) = \\
&= \frac{1}{M}\sum_{k=0}^{M-1} X(z^{L/M} e^{-j\frac{2\pi}{M}kL})
\end{aligned}
$$

In the case of the cascade of the downsampler with the upsampler,

$$
\begin{aligned}
V_2(z) &= \frac{1}{M}\sum_{k=0}^{M-1} X(z^{1/M} e^{-j\frac{2\pi}{M}k}) \\
Y_1(z) &= V_2(z^L) = \\
&= \frac{1}{M}\sum_{k=0}^{M-1} X(z^{L/M} e^{-j\frac{2\pi}{M}k})
\end{aligned}
$$

It follows that $Y_1(z) = Y_2(z)$ if

$$
\sum_{k=0}^{M-1} X(z^{L/M} e^{-j\frac{2\pi}{M}kL}) = \sum_{k=0}^{M-1} X(z^{L/M} e^{-j\frac{2\pi}{M}k}).
$$

The equality is satisfied if and only if M and L are *relatively prime*, meaning they do not have common factors greater than 1. Under this condition, $e^{-j\frac{2\pi}{M}kL}$ and $e^{-j\frac{2\pi}{M}k}$ take the same set of values for $k = 0, 1, \ldots, M-1$.

Noble identities

Two other simple cascade equivalence relations are depicted in the following figure:

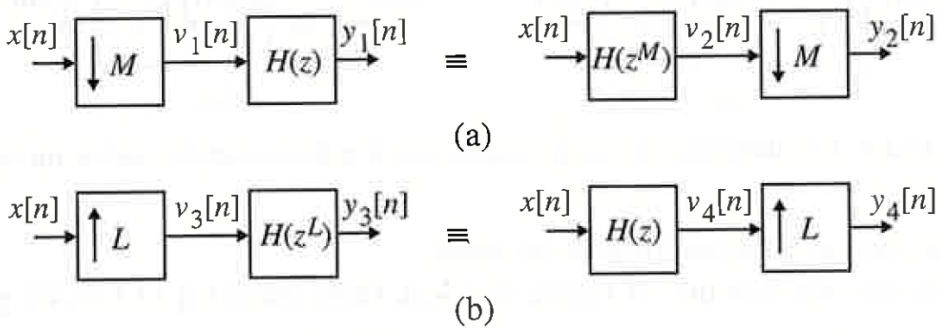


Figure 13.14: Cascade equivalences: (a) equivalence #1 and (b) equivalence #2.

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

These relations are commonly used in multirate networks to rearrange sampling rate alteration devices (downsamplers and upsamplers) into the most convenient positions.

12.03 Multirate structures for sampling rate conversion

The process of reducing the sample rate of a sequence is commonly known as *decimation*, and the corresponding system is called a *decimator*. Similarly, the process of increasing the sample rate is called *interpolation*, and the corresponding system is called an *interpolator*.

We know from the sampling theorem that the sampling rate of a critically sampled signal with a spectrum occupying the full Nyquist range cannot be reduced any further, as this would introduce aliasing. Therefore, the bandwidth of a critically sampled signal must first be reduced by lowpass filtering before the sampling rate is further reduced by the downsampler.

On the other hand, the zero-valued samples introduced by the upsampler must be interpolated for an effective sampling rate increase. We must remove the repetitions of the spectrum, the images, generated by the upsampler. Again, this operation can be performed with lowpass filtering.

Since a fractional sampling rate alteration can be obtained by concatenating an interpolator and a decimator, filters are also needed to design such multirate systems.

Here, we derive the input-output relations for the multirate structures used for sampling rate conversion.

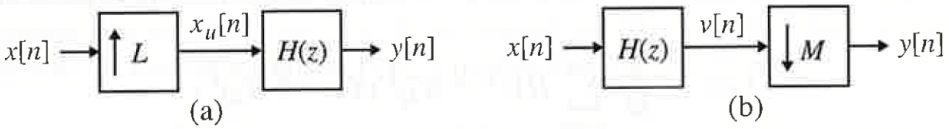


Figure 13.15: Sampling rate alteration systems for integer-valued conversion factors: (a) interpolator and (b) decimator.

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

We have observed that upsampling by a factor of L leads to the periodic repetition of the signal spectrum. Hence, the basic interpolator structure for integer-valued sampling rate increase comprises an

upsampler followed by a lowpass filter $H(z)$ with a cutoff frequency of π/L. $H(z)$ is referred to as the *interpolation filter*, and it is employed to eliminate the $L-1$ undesired images in the upsampled signal $x_u(n)$.

Since downsampling by a factor of M may lead to aliasing, the fundamental decimator structure comprises a lowpass filter $H(z)$ with a stop-band starting at π/M, followed by the downsampler. In this context, the lowpass filter is referred to as the *decimation filter* and is utilized to prevent aliasing by constraining the spectrum of the input signal to $|\omega| < \pi/M$ before downsampling.

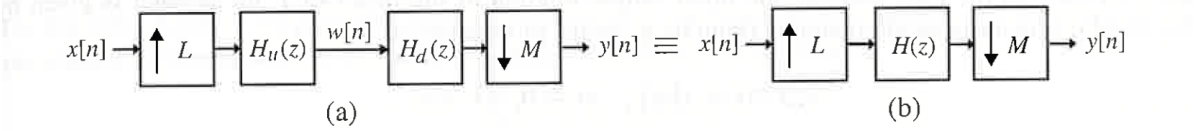


Figure 13.16: (a) General scheme for increasing the sampling rate by L/M and (b) an efficient implementation.

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

A fractional change in the sampling rate by a rational factor can be achieved by cascading an interpolator and a decimator. The interpolator must precede the decimator to ensure that the baseband of the intermediate signal is greater than that of the original signal and to prevent the loss of components. Both the interpolator and the decimator involve the use of a lowpass filter, denoted as $H_u(z)$ and $H_d(z)$, respectively, operating at the same sampling rate. These two lowpass filters can be replaced by a single lowpass filter $H(z)$, with the passband set to the lowest of the two, i.e., with a stopband starting at $\pi/\max(L,M)$.

Let us first develop the input-output relationship of the decimator. Let $h(n)$ be the impulse response of the decimation filter $H(z)$. Then the corresponding output is

$$v(n) = \sum_{r=-\infty}^{+\infty} h(n-r)x(r)$$

The output of the decimator is instead

$$y(n) = v(Mn)$$

which, combined with the previous one, gives

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

Note that if the decimation filter is implemented in FIR form, we only need to compute its output once every M input samples, thereby reducing the computational complexity by a factor of M. However, with an IIR filter, we need to compute all output samples, and we do not achieve such a significant reduction in computation. Therefore, FIR filters are often preferred over IIR filters in multirate systems.

Next, we develop the input-output relation of the interpolator. In the time domain, the input-output relation of the upsampler with a factor of L is given by:

$$x_u(Lm) = x(m), \qquad m = 0, \pm 1, \pm 2, \ldots$$

and is 0 elsewhere.
Substituting this expression into the input-output relation of the interpolation filter:

$$y(n) = \sum_{r=-\infty}^{\infty} h(n-r)x_u(r)$$

which, with a change of variable, becomes:

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

Again, using an FIR interpolation filter, we can reduce the number of operations to be performed for each sample by a factor of L, because only one every L sample of $h(n)$ is involved in the relationship. However, with IIR filters, we do not have such a significant computational advantage.
Eventually, for the fractional-rate sampling rate converter, the input-output relationship can be obtained in a similar manner and is given by:

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

12.04 Multistage Design of Decimator and Interpolator

The decimator and interpolator structures we have seen are single-stage structures. Indeed, the basic scheme for implementation involves a single lowpass filter and a single sampling rate alteration device. However, if the interpolation factor L can be expressed as a product of two integers L_1 and L_2, the factor of L interpolation can be realized in two stages. Similarly, if the decimation factor M is the product of two integers M_1 and M_2, the factor of M decimation can also be realized in two stages:

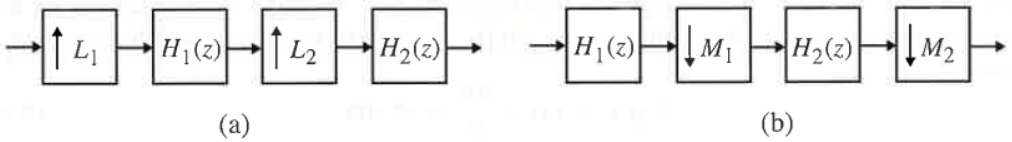


Figure 13.22: Two-stage implementation of sampling rate alteration systems: (a) interpolator and (b) decimator.

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

The design can involve more than two stages, depending on the number of factors used to express L and M. In general, computation efficiency is significantly improved by designing the sampling rate alteration system as a cascade of several stages.
For example, consider a decimator for the reduction of the sampling rate from 12 kHz to 400 Hz, with a decimation factor of 30:

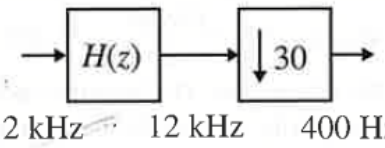


Figure 13.23: Block-diagram representation of the single-stage factor-of-30 decimator.

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

The specifications for the decimation filter are as follows: passband edge $F_p = 180$ Hz, stopband edge $F_s = 200$ Hz, passband ripple $\delta_p = 0.002$, and stopband ripple $\delta_s = 0.001$. Thus, $\omega_p = \frac{180}{12000} 2\pi$. If $H(z)$ is an equiripple linear-phase FIR filter, the order N_H (determined using `firpmord`) is

$$N_H = 1827$$

The number of multiplications per second (ignoring the filter symmetry) is

$$(N_H + 1) \times \frac{F_T}{30} = 1828 \times \frac{12\,000}{30} = 730\,800$$

Let us now consider the implementation of $H(z)$ as a cascade realization of two systems. The first system is a comb filter $F(z^{15})$ that realizes the lowpass filter we want with the requested transition band but also generates some upper bands. The other filter $I(z)$ is a pure lowpass filter that selects the lowest band but considers a larger transition band, since it takes into account the spectral gaps between the passbands of $F(z^{15})$. Since the cascade of the two systems has a passband ripple that is the sum in dB of the ripples of the two systems, to compensate we will require both $F(z)$ and $I(z)$ to have a passband ripple $\delta_p = 0.001$ (rather than $\delta_p = 0.002$). On the other hand, the stopband of $I(z)F(z^{15})$ is at least as good as $I(z)$ and $F(z)$, and therefore we will require $\delta_s = 0.001$.
We design $F(z)$ with the Parks-McClellan algorithm considering $F_p = 15 \times 180$ Hz and $F_s = 15 \times 200$ Hz. The transition band is 15 times larger than before, and the filter requires an order $N_f = 130$.
We also design $I(z)$ with the Parks-McClellan algorithm considering $F_p = 180$ Hz and $F_s = \frac{12\,000}{15} - 200 = 600$ Hz. The transition band is much larger than the 20 Hz of the first filter, and the order $N_I = 93$.

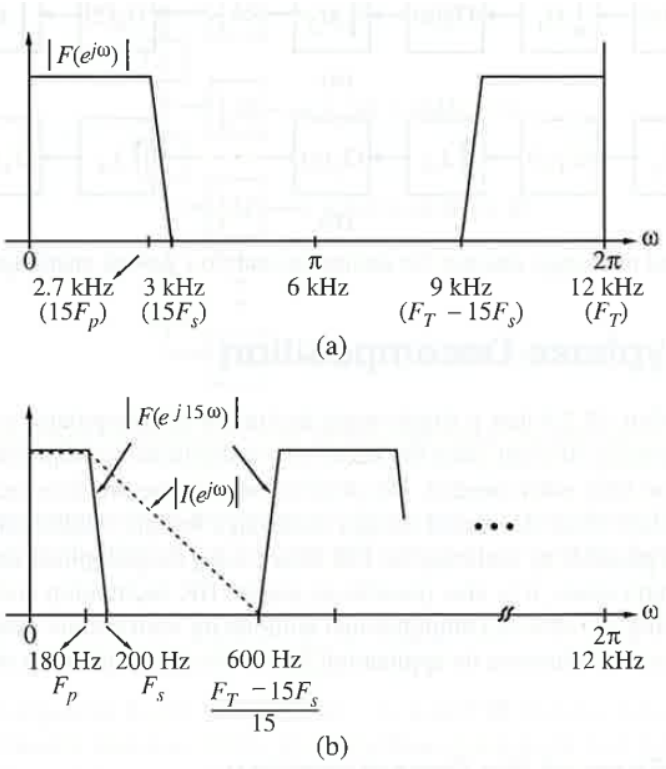


Figure 13.24: Decimation filter design based on the IFIR approach (frequency response plots not shown to scale).

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

We can now drastically reduce the computational complexity by making use of the cascade equivalence and performing multistage decimation.

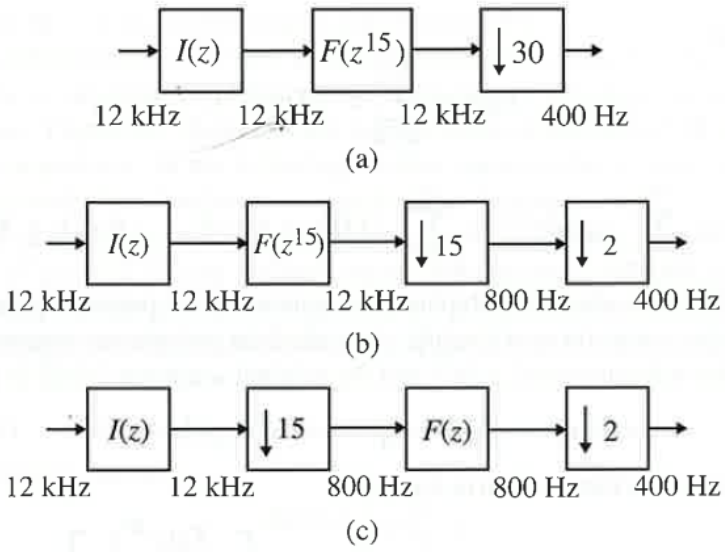


Figure 13.25: The steps in the two-stage realization of the decimator structure.

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

The implementation of $F(z)$ at sampling rate 800 Hz followed by a down-sampling by a factor 2 requires

$$131 \times \frac{800}{2} = 52\,400 \text{ multiplications per second.}$$

The implementation of $I(z)$ at sampling rate 12 kHz followed by a down-sampling by a factor 15 requires

$$94 \times \frac{12000}{15} = 75\,200 \text{ multiplications per second.}$$

The total complexity of this implementation becomes

$$52\,400 + 75\,200 = 127\,600 \text{ multiplications per second}$$

i.e. 5.7 time smaller than the direct single-stage implementation.

12.05 The polyphase decomposition

We have seen that a single-stage decimator and interpolator employing FIR lowpass filters can be computationally efficient, since the multiplications to compute the output can be carried out only when needed. We have also seen the computational requirements can be further decreased using a multi-stage design. We can further reduce the computational complexity by exploiting the polyphase decomposition of FIR filters.

To illustrate this approach, let's consider, for simplicity, a causal FIR transfer function $H(z)$ of length 9:

$$\begin{aligned} H(z) &= h(0) + h(1)z^{-1} + h(2)z^{-2} + h(3)z^{-3} + h(4)z^{-4} \\ &+ h(5)z^{-5} + h(6)z^{-6} + h(7)z^{-7} + h(8)z^{-8} \end{aligned}$$

This transfer function can expressed as the sum of two terms:

$$\begin{aligned} H(z) &= \left(h(0) + h(2)z^{-2} + h(4)z^{-4} + h(6)z^{-6} + h(8)z^{-8}\right) \\ &+ \left(h(1)z^{-1} + h(3)z^{-3} + h(5)z^{-5} + h(7)z^{-7}\right) = \\ &= \left(h(0) + h(2)z^{-2} + h(4)z^{-4} + h(6)z^{-6} + h(8)z^{-8}\right) \\ &+ z^{-1}\left(h(1) + h(3)z^{-2} + h(5)z^{-4} + h(7)z^{-6}\right) \end{aligned}$$

and by using the notation

$$\begin{aligned} E_0(z) &= h(0) + h(2)z^{-1} + h(4)z^{-2} + h(6)z^{-3} + h(8)z^{-4} \\ E_1(z) &= h(1) + h(3)z^{-1} + h(5)z^{-2} + h(7)z^{-3} \end{aligned}$$

we can rewrite

$$H(z) = E_0(z^2) + z^{-1}E_1(z^2).$$

In a similar manner, selecting every third term, we can express it in the form:

$$H(z) = E_0(z^3) + z^{-1}E_1(z^3) + z^{-2}E_1(z^3)$$

where now

$$\begin{aligned} E_0(z) &= h(0) + h(3)z^{-1} + h(6)z^{-2} \\ E_1(z) &= h(1) + h(4)z^{-1} + h(7)z^{-2} \\ E_2(z) &= h(2) + h(5)z^{-1} + h(8)z^{-2}. \end{aligned}$$

The decomposition of $H(z)$ in this form is called *polyphase decomposition*. In case of M branches, we have

$$H(z) = \sum_{k=0}^{M-1} z^{-k}E_k(z^M)$$

where the polyphase component $E_k(z)$ is an FIR transfer function.
We will not consider the case of IIR filters, but I would like to mention the fact that polyphase decomposition is applicable also to IIR filters, and in that case, the polyphase component $E_k(z)$ is IIR. Moreover, for certain types of stable lowpass IIR transfer functions with cutoff at π/M, the polyphase decomposition takes the form:

$$H(z) = \sum_{k=0}^{M-1} z^{-k}A_k(z^M)$$

with $A_k(z)$ being a stable allpass transfer function.
Let us consider the M-branch polyphase decomposition of $H(z)$. The following figure provides the direct realization and the transposed realization of

$$H(z) = \sum_{k=0}^{M-1} z^{-k}E_k(z^M)$$

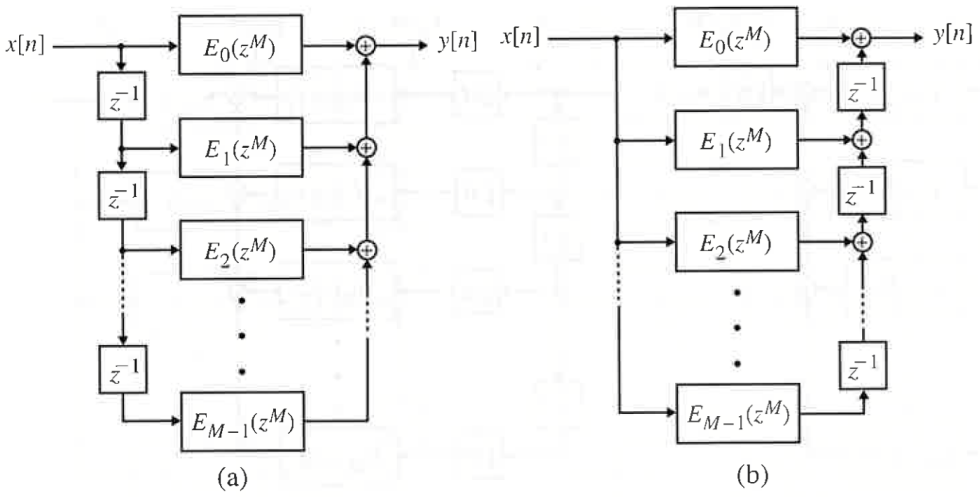


Figure 13.28: (a) Direct realization of an FIR filter based on a Type I polyphase decomposition and (b) its transpose.

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

The first polyphase decomposition is also called *Type I polyphase decomposition*. Very often, the transposed structure is represented in an alternative manner considering

$$R_l(z^M) = E_{M-1-l}(z^M).$$

The corresponding polyphase decomposition is thus given by

$$H(z) = \sum_{l=0}^{M-1} z^{-(M-1-l)} R_l(z^M).$$

This decomposition is called *Type II polyphase decomposition*.

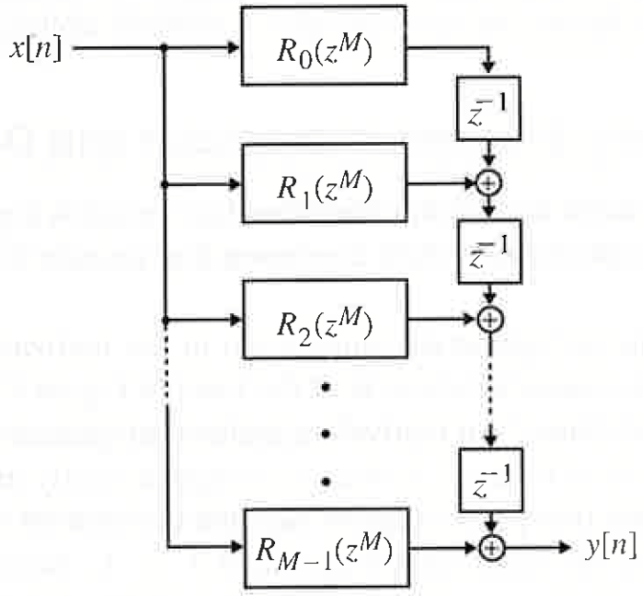


Figure 13.29: Realization of an FIR filter based on a Type II polyphase decomposition.

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

Computationally efficient decimator and interpolator structures employing lowpass filters can be derived by applying a polyphase decomposition to the corresponding transfer functions.

Consider first the use of the polyphase decomposition in the realization of the decimation filter. Using the first cascade equivalence, we can obtain a computationally more efficient realization.

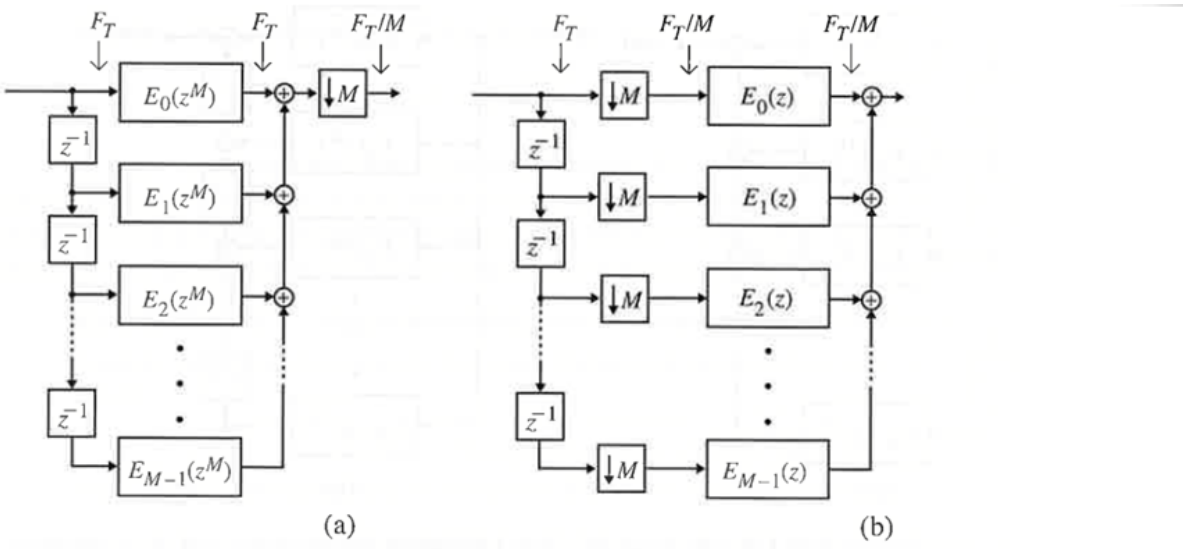


Figure 13.30: (a) Decimator implementation based on a Type I polyphase decomposition and (b) computationally efficient decimator structure. In the figures, the sampling rates of pertinent sequences are indicated by an arrow.

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

Assume first the use of a standard decimation filter $H(z)$, with length N. Since the decimator output $y(n)$ is obtained by down-sampling the filter output $v(n)$ by a factor M, we need to compute $v(n)$ only for $n = \ldots, -2M, -M, 0, M, 2M, \ldots$, the computational requirement is of N multiplications and $N-1$ addition per output sample. However, as n increases, the values memorized in the tapped delay line change. Thus, it is necessary to perform all operations in one input sampling period (i.e., at the highest rate) and in the following $M-1$ periods the arithmetic unit will remain idle.

Now consider the polyphase decomposition structure. If the length of the subfilter $E_k(z)$ is N_k, then $N = \sum_{k=0}^{M-1} N_k$. The overall computational cost is of $\sum_{k=0}^{M-1} N_k = N$ multiplications and $\sum_{k=0}^{M-1}(N_k - 1) + (M-1) = N-1$ additions per decimator output sample. The computational cost hasn't changed, but now all operations are performed at the lowest rate, with the arithmetic unit operative all instants of the output sampling period (which is M times that of the input sampling period).

Similar savings can be obtained in the case of the interpolator structure employing polyphase decomposition for the realization of a computationally efficient interpolator. In this case, it is convenient to use the second cascade equivalence:

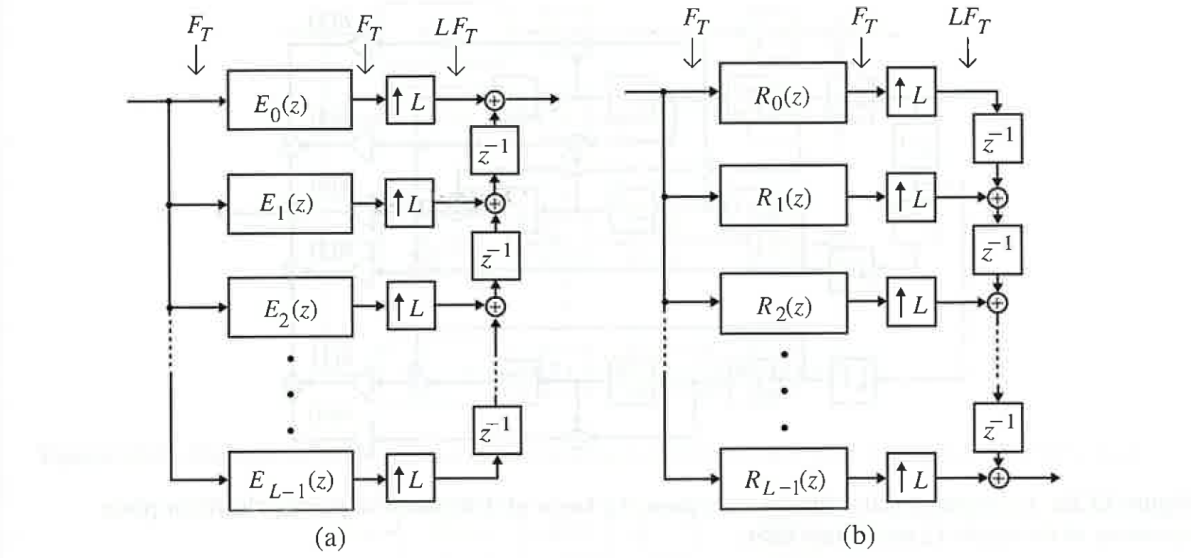


Figure 13.31: Computationally efficient interpolator structures: (a) Type I polyphase decomposition and (b) Type II polyphase decomposition. In the figures, the sampling rates of pertinent sequences are indicated by an arrow.

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

For more information read:

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

Chapter 13.1, pp. 740-749

Chapter 13.2, pp. 750-752,

Chapter 13.3, pp. 758-759,

Chapter 13.4, pp. 762-767,

Contents

00 A brief overview of the course **1**
00.01 The course organization 1

01 Signals and signal processing **3**
01.01 Characterization and classification of signals 3
01.02 Digital signal processing: pros and cons 6

02 Discrete-time signals in the time domain **9**
02.01 Time domain representation 9
02.02 Operations on sequences 10
02.03 Classification of sequences 12
02.04 Basic sequences 16

03 Discrete-time signals in the frequency domain **21**
03.01 The Fourier series 21
03.02 The Continuous-Time Fourier Transform 22
03.03 The Discrete-Time Fourier Transform 27
03.04 Properties of DTFT 33
03.05 The sampling theorem 38

04 Discrete-time systems **44**
04.01 Examples of simple systems 44
04.02 Classification of discrete-time systems 47
04.03 Impulse response and convolution sum 50
04.04 Frequency response 58

05 The Z-Transform **62**
05.01 Definition of the Z-Transform 62
05.02 The inverse Z-Transform 67
05.03 Properties of the Z-Transform 71
05.04 The Transfer Function 74

06 The Discrete Fourier Transform **78**
06.01 Definition of the Discrete Fourier Transform (DFT) 78
06.02 The relation between DFT, DTFT, and Z Transform 79

06.03 Properties of the DFT . . . 82
06.04 The Fast Fourier Transform (FFT) . . . 87
06.05 Linear convolution and circular convolution . . . 98
06.06 Frequency analysis with DTFT and DFT . . . 112
06.07 The Short-Time Fourier Transform - STFT . . . 118
06.08 The Discrete Cosine Transform - DCT . . . 123

07 Discrete-time LTI systems in the frequency domain 127
07.01 Ideal filters . . . 127
07.02 Phase delay and Group delay . . . 128
07.03 Zero-phase filters . . . 132
07.04 Linear phase FIR filters . . . 133
07.05 Geometric interpretation of frequency response computation . . . 139
07.06 Simple digital filters . . . 141
07.07 Comb filters . . . 148
07.08 All-pass filters . . . 149
07.09 Minimum-phase and maximum-phase transfer functions . . . 150
07.10 Inverse system . . . 152
07.11 Deconvolution . . . 152
07.12 Amplitude equalizer and phase equalizer . . . 153
07.13 Stability test for IIR filters . . . 155
07.13.1 The stability triangle . . . 155
07.13.2 Schur-Cohn stability test . . . 156

08 Digital filter structures 160
08.01 Basic building blocks . . . 160
08.02 FIR filter structures . . . 161
08.02.1 Direct form realization . . . 161
08.02.2 Cascade realization . . . 162
08.02.3 Frequency-sampling structure . . . 163
08.02.4 Lattice structure . . . 165
08.03 IIR filter structures . . . 170
08.03.1 Direct form realization of IIR filters . . . 170
08.03.2 Transposition principle . . . 172
08.03.3 Cascade realization . . . 175
08.03.4 Parallel form realization . . . 175
08.03.5 Lattice ladder realization . . . 178

09 Finite precision arithmetic effects 186
09.01 Finite precision arithmetic . . . 186
09.02 Filter coefficients quantization . . . 186
09.03 A/D conversion noise . . . 188
09.04 Uncorrelated noise due to rounding or truncation in multiplications . . . 191

09.05 Overflow in additions . . . 192
09.06 Dynamic range scaling . . . 193
09.06.1 An absolute bound . . . 193
09.06.2 $\mathcal{L}_\infty$-bound . . . 194
09.06.3 $\mathcal{L}_2$-bound . . . 194
09.06.4 A more general scaling rule . . . 194
09.06.5 Scaling the Cascade form IIR filter structure . . . 195
09.07 Limit cycles . . . 197

10 Digital filter design **200**
10.01 Digital filter design specifications . . . 200
10.02 IIR filter design . . . 202
10.02.1 The analog domain or continuous-time domain . . . 202
10.02.2 Butterworth filters . . . 204
10.02.3 Chebyshev filters of type I . . . 205
10.02.4 Chebyshev filters of type II . . . 206
10.02.5 Elliptic filters . . . 207
10.02.6 Analog highpass filter design . . . 208
10.02.7 Analog bandpass filter design . . . 209
10.02.8 Analog bandstop filter design . . . 210
10.03 The bilinear transformation . . . 211
10.03.1 Design of Highpass, bandpass, and bandstop IIR digital filters . . . 214
10.03.2 IIR Filter design with Matlab . . . 216
10.03.3 Direct methods for IIR filter design . . . 217
10.03.4 Design method for IIR filters with arbitrary magnitude and phase . . . 217
10.04 FIR filter design . . . 219

11 Adaptive filters **230**
11.01 Introduction . . . 230
11.02 Applications of adaptive filters . . . 231
11.02.1 System identification . . . 231
11.02.2 Adaptive noise cancellation . . . 231
11.02.3 Acoustic echo cancellation . . . 232
11.02.4 Channel equalization . . . 232
11.02.5 Predictive coding . . . 233
11.02.6 Active noise control . . . 234
11.03 Optimal Wiener filter . . . 234
11.04 Least mean square (LMS) adaptive filter of Widrow and Hoff . . . 237
11.05 Sign-error adaptive filter . . . 239
11.06 Normalized LMS (NLMS) adaptive filter . . . 240
11.07 Recursive Least Square (RLS) adaptive filter . . . 240
11.07.1 Affine projection algorithms (APA) . . . 243

11.08 Decorrelation NLMS (DNLMS) algorithm . . . 244
11.09 Perfect periodic sequences . . . 244

12 Multirate Digital Signal Processing **246**
12.01 Introduction . . . 246
12.02 Basic sampling rate alteration devices . . . 246
12.02.1 Time domain characterization . . . 246
12.02.2 Frequency domain characterization . . . 249
12.02.3 Cascade equivalences . . . 252
12.03 Multirate structures for sampling rate conversion . . . 254
12.04 Multistage Design of Decimator and Interpolator . . . 256
12.05 The polyphase decomposition . . . 258