الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي

Academic year 2023-2024

Ziane Achour University of Djelfa

Advanced Physiological Signal Processing Course

Dr. Mohamed DJEMAI
Faculty of science and technology

Table of Contents

Introduction

This course material, designed for first-year Master's students in Biomedical Engineering, focuses on "Advanced Processing of Physiological Signals." It provides the theoretical and practical knowledge necessary to analyze and process various physiological signals, such as electrocardiograms (ECG), electroencephalograms (EEG), and medical images like MRIs and X-rays. These signals are often affected by noise and artifacts, making their interpretation challenging. The course addresses advanced signal processing techniques, including digital filters (FIR and IIR) and the Discrete Fourier Transform (DFT), crucial for enhancing signal quality by reducing noise and extracting relevant diagnostic information.

In addition to these techniques, the course covers the essential characteristics and classification of physiological signals, such as their origin (cardiac, neural) and type (continuous, discrete), which are essential for accurate interpretation and application in medical diagnosis. It also delves into how random or stochastic signals behave in linear systems, focusing on how system properties like stability and impulse response affect signal filtering and system performance. These insights are vital for improving signal processing, noise reduction, and system optimization in biomedical applications.

Through this material, students will develop a deep understanding of filtering concepts, the application of the DFT in signal analysis, and how to design digital filters tailored to biomedical needs. They will also gain practical skills in using software tools for the effective processing of physiological signals.

Chapter 1: Analysis and Synthesis of FIR and IIR Digital Filters

1.1 Signal

A signal is the physical representation of information to be transmitted, serving as the entity that carries this information.

Examples of signals include acoustic waves, such as the current produced by a microphone (speech, music, etc.), biological signals like EEG or ECG, the voltage across a charging capacitor, geophysical signals such as seismic vibrations, financial data like stock market prices, the flow rate of the Seine River, as well as images and videos.

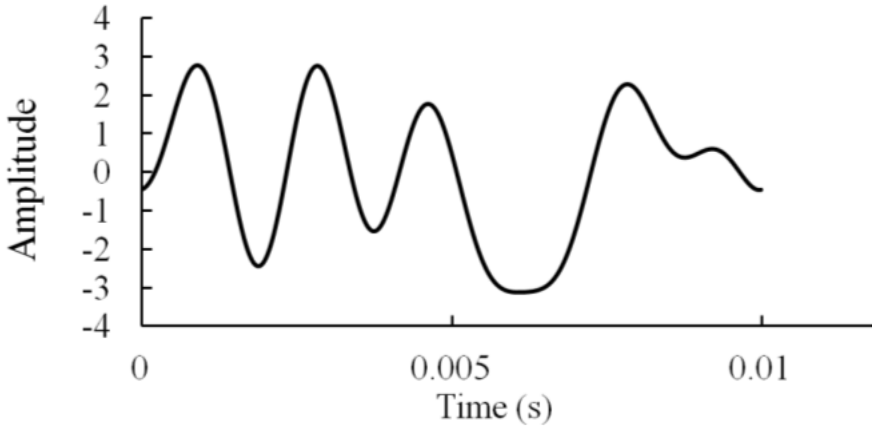


Figure 1.1: Random signal in time space

1.1.1 Dimensional classification

1D (single-dimensional) signal: A function that depends on a single independent variable or parameter, which could be time, space (e.g., an x-coordinate), concentration, or other quantities. It typically represents how a quantity varies along a single axis or parameter.

Examples

- Sound wave: The amplitude (volume) of a sound wave varies over time.
- Stock prices: Stock prices fluctuate over time.
- Electrocardiogram (ECG): An ECG measures the electrical activity of the heart over time.

A 2D (two-dimensional) signal varies across two dimensions. Most commonly these dimensions are spatial (like the x and y positions within an image).

Examples

- Image: The brightness or color intensity of each pixel in an image varies based on its position along the x and y coordinates.
- Geographic map: The elevation on a topographical map varies based on longitude and latitude coordinates.
- Medical imaging (X-ray or MRI scan): Intensity values in a medical scan represent different tissue types and vary across both x and y dimensions.

3D (Three-dimensional) signal: refers to a signal that varies across three dimensions, capturing data along three independent parameters. These parameters are typically spatial, but they could also include time or other variables. In a 3D signal, information is organized in a three-dimensional space, allowing the representation of more complex data structures compared to 1D or 2D signals.

Examples

- Medical Imaging: Signals from MRI or CT scans, where the data represents a 3D model of the human body or an organ.
- 3D Video: A video that captures three-dimensional visual information, often used in virtual reality or stereoscopic displays.
- Geospatial Data: Topographical maps or seismic data that represent the earth's surface or subsurface in three dimensions.

1.1.2 Analog signal (continuous)

A signal that can take an infinite number of values and varies continuously over time.

1.1.3 Digital signal

A digital signal is a signal that varies discretely over time. It is said to be quantized. It consists of a sequence of 0s and 1s, called bits. It is referred to as binary.

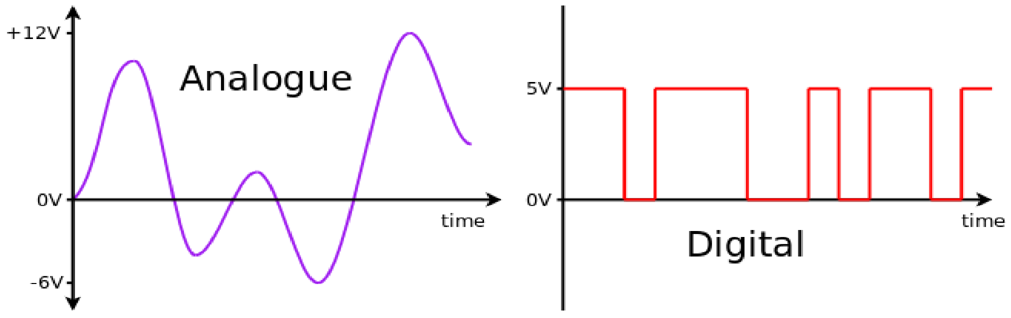


Figure 1.2: Analog signal and digital signal

1.1.4 Digitization of an analog signal

The importance of digital information processing systems is continuously increasing (radio, television, telephone, instrumentation, etc.). This choice is often justified by technical advantages such as high parameter stability, excellent result reproducibility, and enhanced functionalities. Since the external world is inherently "analog," a preliminary analog-to-digital conversion operation is necessary.

Digitization: The process of transforming (or encoding) information into a sequence of bits.

1.1.5 Analog-to-Digital Conversion

The steps of analog-to-digital conversion can be grouped into two main steps: first, sampling the analog signal, and second, quantizing and encoding the sampled values into digital form.

1.1.5.1 Sampling

This step involves slicing the analog signal into small time intervals according to a well-defined period set by a clock. The amplitude (A) of the signal at the clock's pulse is used as the reference for this interval. This value is then encoded. If the signal is sampled at a frequency of 100 Hertz (for example), each time interval, called a period, is equal to $\frac{1}{100} = 0.01$ seconds, or 10 milliseconds. The sampling frequency ($\frac{1}{T_0}$) must be sufficiently high to fully capture the signal.

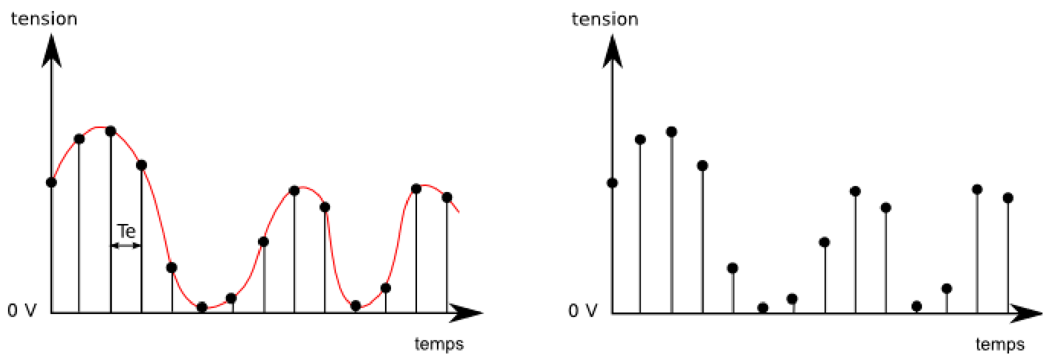


Figure 1.3: Sampled Signal at T_0

1.1.5.2 Quantization and Encoding

After sampling, the values are quantized by mapping them to discrete levels, each representing a range of signal amplitudes. These quantized values are then encoded into a digital format. Encoding involves representing the amplitude of the quantized signal in terms of binary digits, with each binary digit corresponding to a specific weight (e.g., 4, 2, 1 in a 3-bit system). For example, a signal with an amplitude of 7 volts (in decimal) would be converted into a 3-bit binary word (7 in decimal = 0111 in binary).

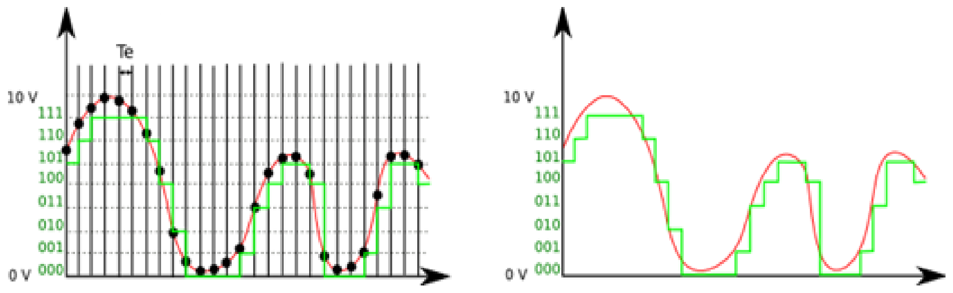


Figure 1.4: Signal encoded on 3 bits

Some basic signals that are useful for studying the properties of signal processing systems are described below.

1.1.6 Discrete-Time Basic Signals

1.1.6.1 Unit Impulse or Kronecker Delta

This is a signal denoted as δ(k) such that:

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

The shifted impulse, denoted as $\delta(k-k)$, is defined as follows:

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

1.1.6.2 Unit Step

This is a signal denoted as $\boldsymbol{u(n)}$ such that:

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

It can be written as:

$$u(n) = \sum_{k=0}^{+\infty} \delta(n-k)$$

1.1.6.3 **Causal Exponential Signal**

This is the signal such that:

$$x(n) = e^{-an}u(n)$$

Depending on whether $a > 0$ or $a < 0$, the exponential signal will converge to 0 or diverge to $+\infty$.

1.1.6.4 Causal Rectangular Signal of Duration N or Pulse

This is the signal such that:

$$rect_N(n) = \begin{cases} 1 & if\ \ 0 \leq n \leq N-1 \\ 0 & if\ \quad \text{otherwise} \end{cases}$$

1.1.7 Properties of Discrete-Time Signals

1.1.7.1 Causality

A signal is said to be causal when:

$$x(n) = 0 \quad \forall\, n < 0$$

1.1.7.2 **Energy**

The energy of a discrete-time signal $\boldsymbol{x(n)}$ is defined as follows:

$$E_x = \sum_{k=-\infty}^{+\infty} |x(n)|^2$$

1.1.7.3 **Average Power**

The average power of a signal $\boldsymbol{x(n)}$ is defined as:

$$P_x = \lim_{N\to\infty} \frac{1}{N} \sum_{k=-\frac{N}{2}}^{+\frac{N}{2}} |x(n)|^2$$

If the energy E_x is finite, then $x(n)$ is a finite-energy signal and $P_x = 0$. If E is infinite, then P_x can be either finite or infinite. If P_x is finite and non-zero, then $x(n)$ is a finite-power signal.

1.1.7.4 Instantaneous Power

Instantaneous power is defined by:

$$\boldsymbol{P(n)} = |\boldsymbol{x(n)}|^2$$

1.1.7.5 Periodicity

A signal $\boldsymbol{x(n)}$ is periodic with period N if and only if $\boldsymbol{x(n+N)} = \boldsymbol{x(n)}$. Otherwise, $\boldsymbol{x(n)}$ is aperiodic.

1.1.7.6 Symmetry

A signal $\boldsymbol{x(n)}$. is symmetric or even if and only if $\boldsymbol{x(-n)} = \boldsymbol{x(n)}$. A signal $\boldsymbol{x(n)}$ is antisymmetric or odd if and only if $\boldsymbol{x(-n)} = -\boldsymbol{x(n)}$. Every signal can be decomposed into the sum of an even signal and an odd signal.

1.1.7.7 Autocorrelation

The autocorrelation of a signal $\boldsymbol{x(n)}$ is defined by:

$$R_{xx}(k) = \sum_{k=-\infty}^{+\infty} x(n)x(n+k) = R_{xx}(-k)$$

We have: $|R_{xx}(k)| \leq: |R_{xx}(0)| = E$

1.1.7.8 Cross-correlation

The cross-correlation of two signals $\boldsymbol{x(n)}$ and $\boldsymbol{y(n)}$ is defined by:

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

1.1.7.9 Convolution

The linear convolution between two signals $\boldsymbol{x(n)}$ and $\boldsymbol{y(n)}$ is defined by:

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

It is noted that:

$$R_{xy}(k) = x(k) * y(-k)$$

1.2 Z-Transform

The Z-transform is a widely used tool for studying digital signal processing systems. It plays a role analogous to that of the Laplace transform in continuous-time systems.

The bilateral Z-transform of a discrete-time signal $x(n)$ is defined by:

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

Where z is a complex variable ($z\epsilon ₵$) defined wherever this series converges. Since discrete signals are most often causal, the Z-transform (referred to as unilateral) is more commonly defined as:

$$Z[x(n)] = X(z) = \sum_{n=0}^{+\infty} x(n)z^{-n}$$

1.2.1 Examples of Z-Transform

Example 1

Let the following discrete-time signal be:

$$x(n) = \delta(n)$$

We have:

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

$$X(z) = z^0 \quad \forall\, z \epsilon ₵$$

$$X(z) = 1 \qquad \forall\, z \epsilon ₵$$

Example 2

Let the following discrete-time signal be:

$$x(n) = \delta(n-k)$$

We have:

$$X(z) = z^{-k} \quad \forall\, z \epsilon ₵$$

Example 3

Let the discrete-time unit step signal $u(n)$ be the following, we have:

$$U(z) = \sum_{n=0}^{+\infty} u(n) z^{-n} = \sum_{n=0}^{+\infty} z^{-n} = \sum_{n=0}^{+\infty} (z^{-1})^n = 1 + z^{-1} + z^{-2} + z^{-3} + z^{-4} + \cdots$$

It is a geometric series.

When n approaches ∞, the geometric series becomes:

$$U(z) = \frac{1}{1 - z^{-1}} = \frac{z}{z-1} \qquad if\ |z^{-1}| < 1$$

Example 4

Let the following discrete-time signal be:

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

$$X(z) = \sum_{n=0}^{+\infty} x(n) z^{-n} = \sum_{n=0}^{+\infty} a^n z^{-n} = \sum_{n=0}^{+\infty} (az^{-1})^n =$$

$$X(z) = \frac{1}{1 - az^{-1}} = \frac{z}{z-a}$$

1.2.2 Properties of the Z-Transform

The Z-Transform of Discrete-Time sequence is given by:

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

The Z-Transform pair can be written as:

$$x(n) \leftrightarrow X(z)$$

The Z-Transform obeys the following properties:

1.2.2.1 Linearity

If $x_1(n) \leftrightarrow X_1(z)$ and $x_2(n) \leftrightarrow X_2(z)$

$\mathrm{Roc} = \mathrm{R}_1$ $\mathrm{Roc} = \mathrm{R}_2$

Then, $ax_1(n) + bx_2(n) \leftrightarrow aX_1(z) + bX_2(z)$ with $\mathrm{Roc} = R_1 \cap R_2$

Proof

$$ZT[ax_1(n) + bx_2(n)] = \sum_{n=-\infty}^{+\infty} [ax_1(n) + bx_2(n)] z^{-n}$$

$$= \sum_{n=-\infty}^{+\infty} ax_1(n) z^{-n} + \sum_{n=-\infty}^{+\infty} bx_2(n) z^{-n}$$

$$= a \sum_{n=-\infty}^{+\infty} x_1(n) z^{-n} + b \sum_{n=-\infty}^{+\infty} x_2(n) z^{-n}$$

$$ZT[ax_1(n) + bx_2(n)] = aX_1(z) + bX_2(z)$$

1.2.2.2 Time shifting

If $x(n) \leftrightarrow X(z);$ $\mathrm{Roc} = \mathrm{R}_1$

Then $x(n-k) \leftrightarrow z^{-k} X(z);$ $\mathrm{Roc} = \mathrm{R}_1$

Proof

$$ZT[x(n-k)] = \sum_{n=-\infty}^{+\infty} x(n-k) z^{-n}$$

Let $n - k = m \Rightarrow n = m + k$. Also, as $n \to +\infty$ then $m \to +\infty$,

$$ZT[x(n-k)] = \sum_{n=-\infty}^{+\infty} x(n-k) z^{-n}$$

$$ZT[x(n-k)] = \sum_{m=-\infty}^{+\infty} x(m)z^{-(m+k)}$$

$$ZT[x(n-k)] = \sum_{m=-\infty}^{+\infty} x(m)z^{-m}z^{-k}$$

$$= z^{-k}X(z)$$

1.2.2.3 Scaling in Z- Domain

$$\text{If} \quad x(n) \leftrightarrow X(z);\ \text{Roc} = R$$

$$\text{Then } a^n x(n) \leftrightarrow X\left(\frac{z}{a}\right);\ \text{Roc} = |a|R$$

Proof

$$ZT[a^n x(n)] = \sum_{m=-\infty}^{+\infty} a^n x(n) z^{-n} = \sum_{n=-\infty}^{+\infty} x(n)(\frac{Z}{a})^{-n} = X(\frac{Z}{a})$$

If ROC is $\alpha < |z| < \beta$, then the new ROC will be $|a|\alpha < |z| < |a|\beta$

1.2.2.4 Time reversal

$$\text{If} \quad x(n) \leftrightarrow X(z);\ \text{Roc} = R$$

$$\text{Then } x(-n) \leftrightarrow X\left(\frac{1}{z}\right);\ \text{Roc} = \frac{1}{R}$$

Proof

$ZT[x(-n)] = \sum_{n=-\infty}^{+\infty} x(-n)z^{-n}$; let $-n = m \Rightarrow As\ n \rightarrow +\infty,$ then $m \rightarrow -\infty,$

$$ZT[x(-n)] = \sum_{m=-\infty}^{+\infty} x(m)z^{m}$$

$$ZT[x(-n)] = \sum_{m=-\infty}^{+\infty} x(m)(z^{-1})^{-m}$$

$$= X(z^{-1}) = X\left(\frac{1}{z}\right)$$

If ROC is $a < |z| < b$, then the new ROC will be $a < \left|\frac{1}{z}\right| < b \Rightarrow \frac{1}{b} < |z| < \frac{1}{a}$

1.2.2.5 Differentiation in Z- Domain

$$\text{If} \quad x(n) \leftrightarrow X(z);\ \text{Roc} = R$$

$$\text{Then } nx(n) \leftrightarrow -z\frac{d}{dz}X(z);\ \text{Roc} = R$$

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

$$\frac{d}{dz}[X(z)] = \sum_{n=-\infty}^{+\infty} x(n)\frac{d}{dz}(z^{-n})$$

$$\frac{d}{dz}[X(z)] = \sum_{n=-\infty}^{+\infty} x(n)(-n.z^{-n-1})$$

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

$$-z\frac{d}{dz}[X(z)] = \sum_{n=-\infty}^{+\infty} [n.x(n)]z^{-n}$$

$$-z\frac{d}{dz}[X(z)] = ZT[n.x(n)]$$

1.2.2.6 Convolution property

If $x_1(n) \leftrightarrow X_1(z)$; Roc = R_1

$x_2(n) \leftrightarrow X_2(z)$; Roc = R_2

Then $x_1(n) * x_2(n) \leftrightarrow X_1(z) \times X_2(z)$; Roc = $R_1 \cap R_2$

(Convolution in time domain↔ multiplication in Z- Domain)

Proof

Convolution of two signals $x_1(n)$ and $x_2(n)$ is given by:

$$x_1(n) * x_2(n) = \sum_{n=-\infty}^{+\infty} x_1(k)x_2(n-k)$$

The Z-transform of convolution is given by:

$$ZT[x_1(n) * x_2(n)] = \sum_{n=-\infty}^{+\infty}\left[\sum_{k=-\infty}^{+\infty} x_1(k)x_2(n-k)\right]z^{-n}$$

Interchanging the order of summation

$$ZT[x_1(n) * x_2(n)] = \sum_{k=-\infty}^{+\infty} x_1(k)\left[\sum_{n=-\infty}^{+\infty} x_2(n-k)z^{-n}\right]$$

If $x_2(n) \leftrightarrow X_2(z)$;

Then $x_2(n-k) \leftrightarrow z^{-k} X_2(z)$;

$$ZT[x_1(n) * x_2(n)] = \sum_{k=-\infty}^{+\infty} x_1(k)\, X_2(z)z^{-k}$$

$$=X_2(z)\sum_{k=-\infty}^{+\infty} x_1(k)\, z^{-k}$$

$$= X_1(z).X_2(z)$$

1.2.3 Inverse Z-Transform

The general idea of the inverse Z-transform is to decompose a complex function $X(z)$ into simpler components for which the inverse Z-transform is known. Using the linearity property of the Z-transform, the original time-domain signal can be reconstructed by summing the individual time-domain signals that correspond to each of the elementary components in the decomposition. Assuming: $\alpha < |z| < \beta$

$$X(z) = \alpha_1 X_1(z) + \alpha_2 X_2(z) + \cdots + \alpha_L X_L(z)$$

We obtain:

$$x(n) = \alpha_1 x_1(n) + \alpha_2 x_2(n) + \cdots + \alpha_L x_L(n)$$

The class of rational Z-transforms can always be expressed according to this principle. We will then write:

$$X(z) = \alpha_1 \frac{z}{z-p_1} + \alpha_2 \frac{z}{z-p_2} + \cdots$$

We will then have, by applying the inverse Z-transform:

$$x(n) = \alpha_1 p_1^n + \alpha_2 p_2^n + \cdots$$

The table below summarizes the Z-transforms and inverse Z-transforms of the most commonly used functions in signal processing.

Table 1.1: Z-Transforms and Inverse Z-Transforms of Common Functions

$x(n)$	$X(z)$
$\delta(n)$	1
$\delta(n-k)$	z^{-k}
$u(n)$	$\frac{z}{z-1}$
n	$\frac{z}{(z-1)^2}$
a^n	$\frac{z}{z-a}$
e^{-an}	$\frac{z}{z-e^{-a}}$
$sin(\omega_0 n)$	$\frac{z.sin(\omega_0)}{z^2-2zcos(\omega_0)+1}$
$cos(\omega_0 n)$	$\frac{z.\left(z-cos(\omega_0)\right)}{z^2-2zcos(\omega_0)+1}$

1.3 Digital Filtering

A digital filter is a system that processes discrete-time signals by applying mathematical operations to modify or enhance certain aspects of the signal. It operates on sequences of data (e.g., audio or sensor data) and is characterized by its transmittance function or transfer function in the Z-domain, which describes the relationship between the input and output signals.

- Examples of filtering are given below:
  Noise reduction for radio signals, sensor-derived images, or biomedical signals (ECG, EEG, EMG, etc.).
- Modification of certain frequency regions in an audio signal or image.
- Restriction to a predefined frequency band.

1.3.1 Transmittance in Z of a Digital Filter:

Let a system, which takes an input sequence $x(n)$ and produces an output sequence $y(n)$, be represented as:

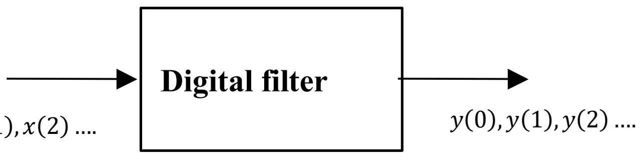


Figure 1.5: Representation of an Input-Output Relationship in a Discrete-Time System

Let $\mathrm{X(z)}$ and $\mathrm{Y(z)}$ be the Z-transforms of the input and output sequences. The transmittance $\mathrm{T(z)}$ of the filter is then defined by:

$$\mathrm{T(z)} = \frac{\mathrm{Y(z)}}{\mathrm{X(z)}}$$

Since the Z-transforms $\mathrm{X(z)}$ and $\mathrm{Y(z)}$ are polynomials containing negative powers of z, the transmittance will be a ratio of two polynomials in negative powers of z.

For example, let's find the transmittance of a digital high-pass filter that responds to a step input in the same way as an analog filter with a time constant τ=10 ms, and therefore a cutoff frequency of $f_c = \frac{1}{2\pi} = 15.9$ Hz:

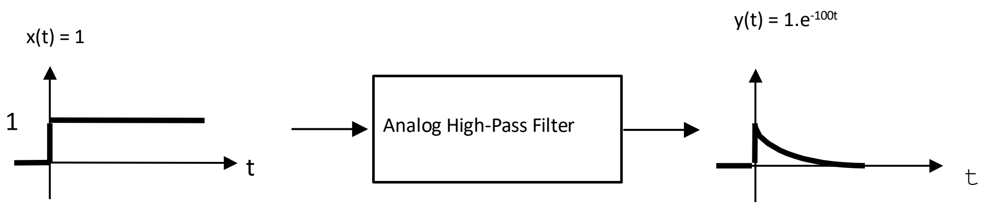


Figure 1.6: Step Response of an Analog High-Pass Filter

The equivalent digital filter would exhibit the following behavior:

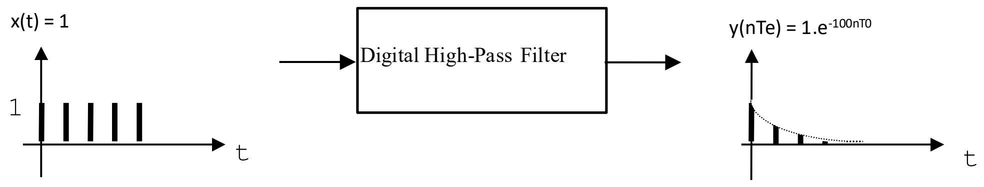


Figure 1.7: Step Response of a Digital High-Pass Filter

If the signal is sampled at $F_0 = 1$ KHz, which corresponds to $T_0 = 1$ ms, then: $X(z) = \frac{z}{z-1}$ and $Y(z) = \frac{z}{z-k}$ with $k = e^{-100T_0} = 0.905$

We can deduce the transmittance of the filter as:

$$T(z) = \frac{Y(z)}{X(z)} = \frac{z-1}{z-0.905}$$

This simple example illustrates how easy it is to find the transmittance of a digital filter that produces a particular output in response to a given input. This technique, used to synthesize sophisticated digital filters, is called the impulse or step response identification method.

1.3.2 Algorithm for calculating $y(n)$

The system thus uses the previous p output samples and the previous q input samples, plus the current input $x(n)$, to calculate the output at time $t = nT_0$.

The algorithm allows us to calculate the value of the output sample yny_nyn based on the previous input and output samples. The most general digital filter can be described by a calculation algorithm of the following form:

$$y(n) = a_1 y(n-1) + a_2 y(n-2) + \cdots + a_p y(n-p) + b_0 x(n) + b_1 x(n-1) + \cdots + b_q x(n-q)$$

Depending on the form of the algorithm, there are two main types of filters, each with its specific properties:

A. Filters where the output depends only on the input and not on previous outputs:

- Their response to an impulse eventually settles to zero after a certain time.
- These are called non-recursive filters or Finite Impulse Response (FIR) filters.
- They have no direct analog equivalent.

Example

Moving average filter $y(n) = \frac{x(n)+x(n-1)+x(n-2)}{3}$.

B. Filters where the output depends on both the current and previous inputs and outputs:

- Their response to an impulse decay to zero only after an infinite amount of time.

- These are called recursive filters or Infinite Impulse Response (IIR) filters.

Example

First-order low-pass filter $y(n) = 0.5.y(n-1) + 0.25.[x(n) + x(n-1)]$.

1.3.3 Transition from the Algorithm to T(z)

The algorithm

$$y(n) = a_1 y(n-1) + a_2 y(n-2) + a_3 y(n-3) + \cdots + a_p y(n-p) + b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + \cdots + a_q x(n-q)$$

By applying the Z-transform, we obtain:

$$Y(z) = a_1 Y(z) z^{-1} + a_2 Y(z) z^{-2} + a_3 Y(z) z^{-3} + \cdots + a_p Y(z) z^{-p} + b_0 X(z) + b_1 X(z) z^{-1} + b_2 X(z) z^{-2} + \cdots + b_q X(z) z^{-q}$$

$$Y(z)\left(1 - a_1 z^{-1} - a_2 z^{-2} - \cdots - a_p z^{-p}\right) = X(z)\left(b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_q z^{-q}\right)$$

Z-transmittance of the filter:

$$\mathrm{T(z)} = \frac{\mathrm{Y(z)}}{\mathrm{X(z)}} = \frac{\mathrm{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_q z^{-q}}}{\mathrm{1 - a_1 z^{-1} - a_2 z^{-2} - \cdots - a_p z^{-p}}}$$

$$= \frac{\sum_{\mathrm{i=0}}^{\mathrm{q}} \mathrm{b_i z^{-i}}}{1 - \sum_{\mathrm{i=1}}^{\mathrm{p}} \mathrm{a_i z^{-i}}}$$

Example

Moving average filter over 4 values

The algorithm

$$Y(n) = \frac{x(n) + x(n-1) + x(n-2) + x(n-3)}{4}$$

$$= 0.25(x(n) + x(n-1) + x(n-2) + x(n-3)).$$

By applying the Z-transform, we obtain:

$$Y(z) = 0.25(\mathrm{X(z)} + \mathrm{X(z)} z^{-1} + \mathrm{X(z)} z^{-2} + \mathrm{X(z)} z^{-3})$$

$$= 0.25(1 + z^{-1} + z^{-2} + z^{-3})\mathrm{X(z)}$$

$$T(z) = \frac{Y(z)}{X(z)} = 0.25(1 + z^{-1} + z^{-2} + z^{-3})$$

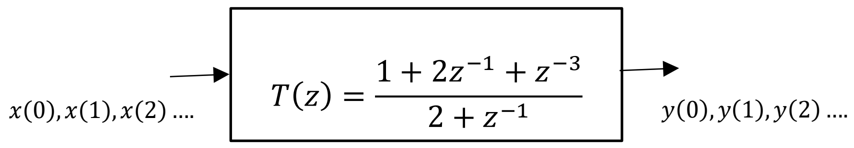
Z-transmittance of the filter:

$$T(z) = \frac{1 + z^{-1} + z^{-2} + z^{-3}}{4}$$

1.3.4 Transition from T(z) to the algorithm

Example

We aim to find the calculation algorithm of the filter characterized by the following transmittance T(z):



The transmittance is the ratio between the Z-transform of the output and the Z-transform of the input:

$$T(z) = \frac{1 + 2z^{-1} + z^{-3}}{2 + z^{-1}} = \frac{Y(z)}{X(z)}$$

$$Y(z)(2 + z^{-1}) = X(z)(1 + 2z^{-1} + z^{-3})$$

$$2Y(z) + Y(z)z^{-1} = X(z) + 2X(z)z^{-1} + X(z)z^{-3}$$

$$Y(z) = -0.5Y(z)z^{-1} + 0.5X(z) + X(z)z^{-1} + 0.5X(z)z^{-3}$$

By using the inverse Z-transform, we obtain:

$$y(n) = -0.5y(n-1) + 0.5x(n) + x(n-1) + 0.5x(n-3)$$

1.3.5 Stability of a digital filter

As with analog filters, it is possible to predict the stability or instability of the corresponding physical system from the transmittance.

To determine if a continuous analog system with transmittance $T(p)$ is stable, the poles are calculated as the values of p that nullify the denominator. The system is stable if the poles are negative or complex with a negative real part. If these poles are plotted in the complex plane, they are all located in the left half-plane.

This stability criterion also applies to the transmittances$\mathrm{T}^*(\mathrm{p})$ of sampled systems. A sampled system with transmittance $\mathrm{T}^*(\mathrm{p})$ is stable if all its poles are negative or complex with a negative real part.

Since most work with sampled systems is done using transmittances in Z, it is useful to examine the position of the poles z_i in the plane for a stable system. We know that z and p are related by the variable change: $z = e^{T_e p}$

A stable system will have poles $p_i = a + jb_i$, with $a_i < 0$.

The corresponding point $z_i = e^{T_e p_i} = e^{T_e(a+jb_i)} = e^{T_e a_i}(cosb_i + jsinb_i)$ is such that the magnitude of the complex number $e^{T_e a_i}$ satisfies $|z_i| < 1$.

We deduce a graphical stability criterion for a sampled system: A sampled system with transmittance $\mathrm{T(z)}$ is stable if all its poles are inside the unit circle.

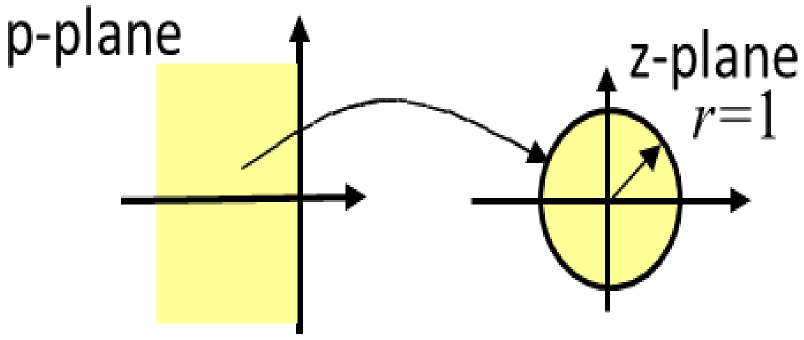


Figure 1.8: Stability criterion of a digital system

1.3.6 Representation of a Digital Filter

A digital filter can be represented using several types of specifications, including:

1.3.6.1 Z-Transfer Function

This is the most common representation. It links the input and output in the Z-plane as $Y(z) = H(z).X(z)$. Going forward, we will assume:

$$H(z) = \frac{N(z)}{D(z)} = \frac{\sum_{i=0}^{N} b_i z^{-i}}{1 + \sum_{i=1}^{N} a_i z^{-i}}$$

Where $N(z)$ is the numerator polynomial of the transfer function, and $D(z)$ is its denominator. Here, N represents the filter's order. If $H(z)$ has poles, the filter is called an IIR filter (Infinite Impulse Response). If $N(z) = 1$, the filter is referred to as an all-pole filter. In the case where $D(z) = 1$, the filter only has zeros, corresponding to FIR filters (Finite Impulse Response).

This type of filter has no equivalent in analog filtering, and we will see that its properties make it a widely used function in digital signal processing.

1.3.6.2 Impulse Response

The impulse response is the inverse Z-transform of $H(z)$.

$$H(z) = \sum_{n=0}^{\infty} h(n) z^{-n}$$

As in analog filtering, the output of a filter $y(nT)$ is the result of the convolution of the input signal, represented in the time domain $x(nT)$, with the impulse response of the filter $h(nT)$. Thus, we have $y(nT) = x(nT) * h(nT)$, or, ignoring the sampling period T:

$$y(n) = x(n) * h(n)$$

$$= \sum_{k=0}^{\infty} x(k) h(n-k)$$

$$= \sum_{k=0}^{\infty} x(n-k) h(k),$$

In the case where $x(n)$ is an impulse $\delta(n)$, we indeed obtain $y(n) = h(n)$. Depending on whether $h(n)$ has infinite or finite support, we will obtain the respective types of filters: IIR (Infinite Impulse Response) and FIR (Finite Impulse Response).

1.3.6.3 **Difference Equation**

An inverse Z-transform of the equation from the first representation leads to the following form:

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

Here, we identify two distinct parts: one that depends on the current and previous values of the input $x(n)$, and another that depends on the previous values of the output $y(n)$. Depending on whether the coefficients a_i are non-zero or zero, we refer to these as recursive filters or non-recursive filters.

1.3.7 Specifications of a Digital Filter

Before a digital filter is designed and implemented, its specifications need to be defined. A filter must allow certain frequencies to pass while attenuating (or even eliminating) others. Therefore, we need to be able to represent these constraints. There are four basic types of filters:

1.3.7.1 Low-pass filters

allow frequencies below a cutoff frequency f_c to pass and block those above it (see Figure 1.9.a).

1.3.7.2 High-pass filters

block frequencies below a cutoff frequency f_c and allow those above it to pass (see Figure 1.9.b).

1.3.7.3 Band-pass filters

allow frequencies around a central frequency f_0(or between f_{c_1} and f_{c_2}) to pass and block others (see Figure 1.9.c).

1.3.7.4 Band-stop filters

block frequencies around a central frequency f_0(or between f_{c_1} and f_{c_2}) and allow others to pass (see Figure 1.9.d).

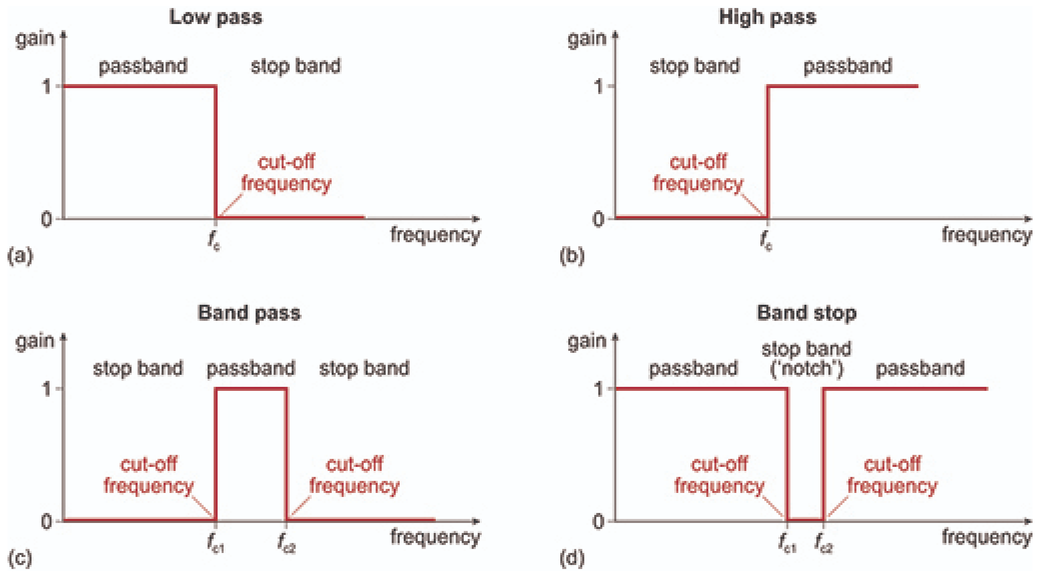


Figure 1.9: Ideal Frequency Responses of the 4 Basic Filters

The filters shown in Figure 1.9 are ideal. In practice, however, filters cannot have discontinuities. The transition between passbands and stopbands occurs through so-called 'transition zones,' where the width of the zone determines the filter's selectivity. Additionally, the passbands and stopbands are not perfectly flat; they exhibit ripples. The amplitude of these ripples is described by the ripple parameter in the passband and the attenuation level in the stopband.

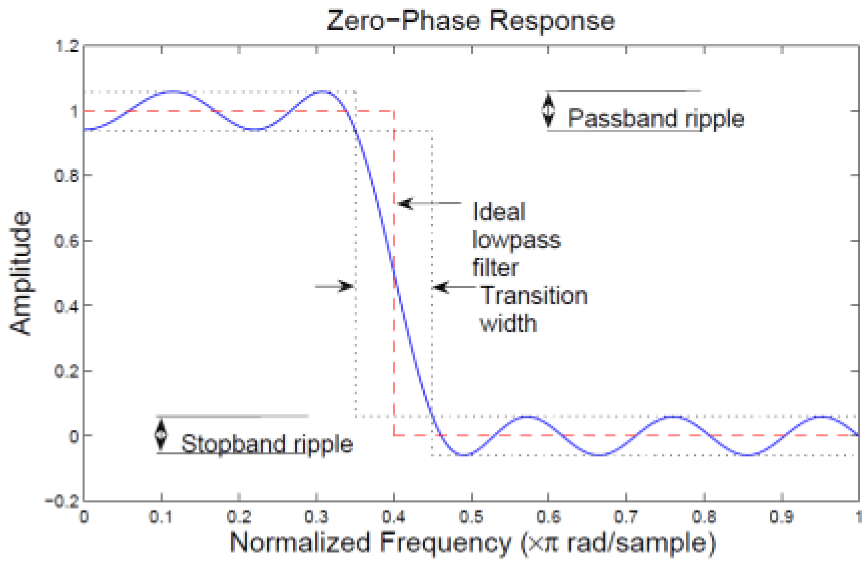


Figure 1.10: Ideal and actual frequency response of lowpass filters

1.4 Classification of Digital Filters

Digital filters can be classified based on several criteria:

- **Impulse Response Length**: This classification distinguishes between two types of filters: IIR (Infinite Impulse Response) and FIR (Finite Impulse Response).
- **Structure or Representation**: This classification differentiates between recursive and non-recursive filters.

It is important to note that, with the exception of specific cases, recursive filters are typically equivalent to IIR filters, while non-recursive filters correspond to FIR filters.

1.4.1 Finite Impulse Response (FIR) Digital Filters

FIR filters cannot be derived from analog filters. However, they are widely used because they have unique properties (linear phase, stability, flexibility). The equations below show the transfer function in the z-domain and the corresponding difference equation for the general form of an FIR filter.

$$H(z) = \sum_{i=0}^{N} h(n) z^{-i}$$

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

It is noted that the coefficients b_i of the filter are also the values of the impulse response $h(k)$, which is therefore limited in time.

$$H(z) = \sum_{i=0}^{N} b_i z^{-i} \Leftrightarrow h(n) = \sum_{i=0}^{N} b_i \delta(n-i)$$

$$h(n) = \begin{cases} b_n & to\, 0 \leq n \leq N \\ 0 & \text{otherwise} \end{cases}$$

1.4.1.1 Characteristics of FIR Filters

The main characteristics of FIR filters are:

- A transition band that will always be wider than that of an IIR filter with the same number of coefficients;
- Synthesis methods allowing any frequency response to be derived;
- Inherent stability ($\sum_{n=0}^{N} |h(n)| < \infty$);
- Greater numerical stability than IIR filters;
- A phase that can be exactly linear, therefore no harmonic distortion in the signal;
- Easier implementation in a digital signal processing system.

1.4.1.2 Structure of FIR Filters

Although there are several practical implementations for FIR filters, the direct form structure and its transposed counterpart are among the most commonly used.

- **Direct Form FIR Filter**

The direct form FIR filter structure calculates the output $y(n)$ as a weighted sum of the current and past input values, where $x(n)$ is the input signal at time n, and the z^{-1} blocks represent unit delays, each shifting the input by one sample. The filter coefficients b_0, b_1, b_0, b_0,…,b_q are applied to the delayed input samples, and the output is obtained by summing the products of these delayed inputs and their corresponding coefficients. The difference equation for the filter is:

$$y(n) = \sum_{i=0}^{q} b_i x(n-i) = b_0 x(n) + b_1 x(n-1) + \cdots + b_{q-1} x(n-(q-1)) + b_q x(n-q)$$

In this form, all designed filter coefficients are typically intended for implementation in the direct form structure. The direct form structure and the associated difference equation are often recommended for fixed-point implementations due to the use of a single accumulator, which helps to mitigate issues related to limited precision in such systems.

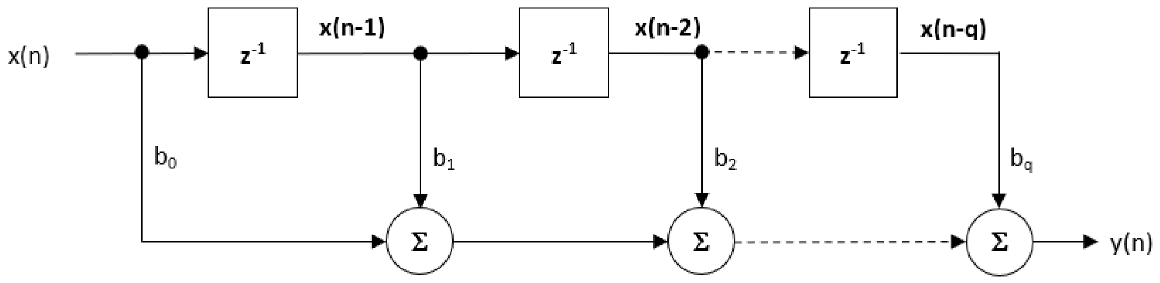


Figure 1.11: Direct Structure

- **Direct Form Transposed FIR Filter**

The Figure 1.12 represents a Direct Form Transposed FIR filter structure, which is commonly recommended due to its superior numerical precision, particularly in floating-point arithmetic. This advantage arises because floating-point addition is performed on numbers of similar magnitude, minimizing the undesirable effects of numerical damping.

Mathematically, the output can be described by the following set of recursive equations:

$$\begin{aligned} y(n) &= b_0x(n) + \omega_1(n-1) \\ \omega_1(n) &= b_1x(n) + \omega_2(n-1) \\ \omega_2(n) &= b_2x(n) + \omega_3(n-1) \\ \vdots &= \vdots \quad + \quad \vdots \\ \omega_q(n) &= b_qx(n) \end{aligned}$$

These equations describe the intermediate variables $\omega_i(n)$, which represent accumulated products of the input signals and the corresponding filter coefficients. As the input signal $x(n)$ propagates through the structure, each $\omega_i(n)$ contributes to the final output signal $y(n)$, resulting in a weighted sum of the current and delayed input values.

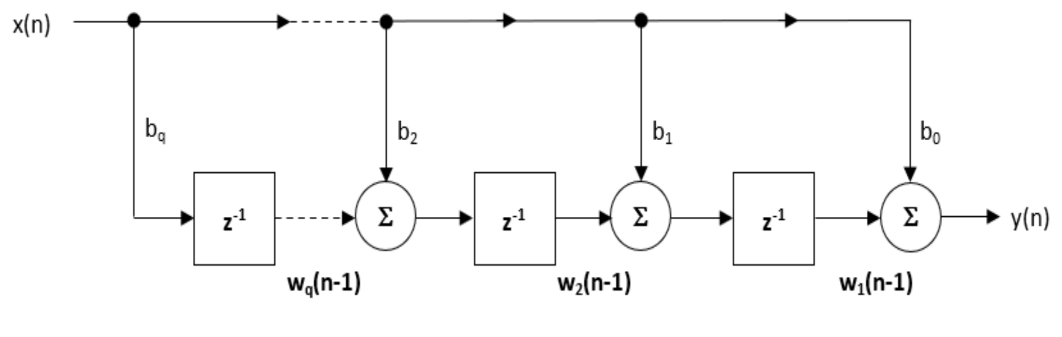


Figure 1.12 : Transposed Structure

1.4.2 Advantages of FIR Filters

- **Linear Phase**: FIR filters can be easily designed to have a linear phase. This means that no phase distortion is introduced into the signal being filtered, as all frequencies are shifted in time by the same amount, thus maintaining their relative harmonic relationships (i.e., constant group and phase delay). This is not the case with IIR filters, which have a non-linear phase characteristic.

- **Stability**: Since FIR filters do not use previous output values to compute their current output (i.e., they have no feedback), they can never become unstable for any type of input signal. This gives them a distinct advantage over IIR filters.
- **Arbitrary Frequency Response**: The ability to design an FIR (Finite Impulse Response) filter with an arbitrary frequency response is a significant advantage for engineers and designers. The Parks-McClellan algorithm is a widely used technique for creating FIR filters that can achieve specific magnitude responses over a defined frequency range.

  This algorithm optimally designs the filter by minimizing the maximum error between the desired and actual frequency responses, allowing for greater flexibility in meeting design specifications.
- **Fixed Point Performance**: The effects of quantization are less severe in FIR filters compared to IIR filters.

1.4.3 Disadvantages of FIR Filters

- **High Computational and Memory Requirement**: FIR filters usually require many more coefficients to achieve a sharp cut-off than their IIR counterparts. Consequently, they require significantly more memory and a higher number of multiply-and-accumulate operations. However, modern microcontroller architectures based on Arm's Cortex-M cores now include DSP hardware support via SIMD (Single Instruction, Multiple Data) that significantly speeds up the filtering operation.
- **Higher Latency**: The higher number of coefficients generally makes a linear phase FIR filter less suitable than an IIR filter for fast, high-throughput applications. This can be problematic for real-time closed-loop control applications, where a linear phase FIR filter may have too much group delay to ensure loop stability.
- **Minimum Phase Filters**: To overcome the inherent N/2 latency (group delay) in a linear filter, one can use a minimum phase filter, where any zeros outside the unit circle are moved to their conjugate reciprocal locations inside the unit circle. The result of this zero flipping is that the magnitude spectrum will be identical to the original filter, and the phase will be nonlinear. Most importantly, the latency will be reduced from N/2 to something much smaller (though non-constant), making it more suitable for real-time control applications. For applications where phase is less important, this may seem ideal. However, the difficulty arises in the numerical accuracy of the root-finding algorithm when dealing with large polynomials. Therefore, orders of 50 or 60 should be considered a maximum when using this approach. Although other methods exist (e.g., the Complex Cepstrum), transforming higher-order linear phase FIR filters to their minimum phase counterparts remains a challenging task.
- **No Analog Equivalent**: Using the Bilinear Transform or matched z-transform (s-z mapping), an analog filter can be easily transformed into an equivalent IIR filter. However, this is not possible for an FIR filter, as it has no analog equivalent.

1.4.4 FIR Filter Design

FIR filters are only realizable in the discrete domain. Consequently, their design methods are not derived from analog filters. There are three main methods of synthesis:

- **The Windowing Method**: This involves applying a window of size N to the equivalent ideal filter.
- **The Frequency Sampling Method**: This method uses the inverse Discrete Fourier Transform (IDFT) from a discrete function that represents the filter and is defined in the frequency domain.
- **Optimization Methods**: These focus on minimizing an error criterion between the actual filter response and the ideal filter. The most commonly used is the Parks and McClellan method, which reformulates the filter design problem as a polynomial approximation.

1.4.4.1 FIR Filter Design by Windowing

We will explain the window method by using an example. Suppose we want to design a lowpass filter with a cutoff frequency of ω_c, i.e., the desired frequency response will be:

$$H_d(\omega) = \begin{cases} 1 & |\omega| < \omega_c \\ 0 & else \end{cases} \ldots\ldots..(*)$$

To find the equivalent time-domain representation, we calculate the inverse discrete-time Fourier transform:

$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{+\pi} H_d(\omega)e^{i\omega n}d\omega \ldots\ldots(**)$$

Substituting Equation $(*)$ into Equation $(**)$, we obtain:

$$h_d(n) = \frac{1}{2\pi}\int_{-\omega_c}^{+\omega_c} H_d(\omega)e^{i\omega n}d\omega = \frac{sin(n\omega_c)}{n\pi} \ldots\ldots.(***)$$

Equation $(***)$ for $\omega_c = \frac{\pi}{4}$ is shown in Figure 1.13:

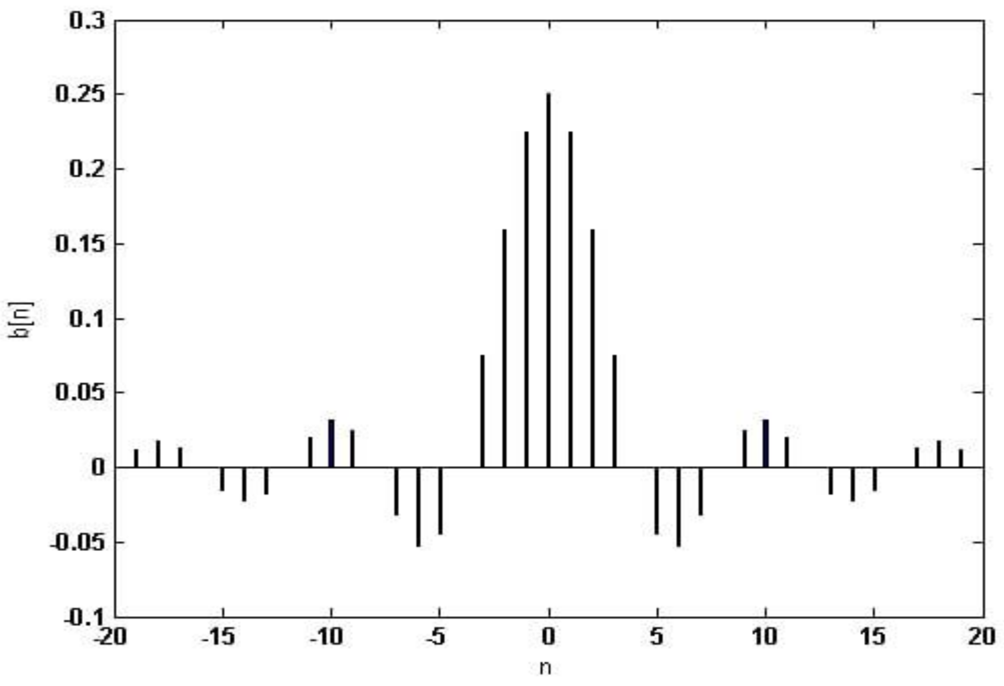


Figure 1.13: Impulse response of an ideal lowpass filter with $\omega_c = \frac{\pi}{4}$

Figure 1.13 shows that $h_d(n)$ needs an infinite number of input samples to perform filtering and that the system is not a causal system.

The obvious solution will be to truncate the impulse response and use, for example, only 21 samples of the input and assume other coefficients to be zero. Intuition suggests that, as the number of samples increases, the truncated impulse response will be closer to the ideal impulse response in Figure 1.13 and therefore the frequency response of the achieved filter will be closer to Equation $(*)$. On the other hand, as we increase the number of samples, more hardware will be required. If we choose to use only 21 taps of the ideal response, there will be three options which are shown in Figures 1.14 to Figures 1.16. The first option is shown in Figure 1.14. This impulse response corresponds to a non-causal system and cannot be used.

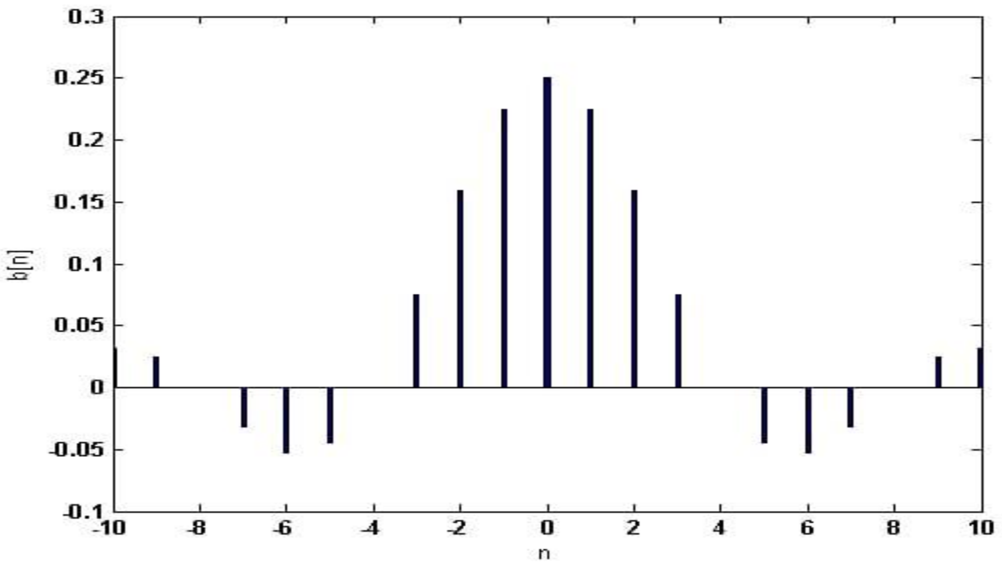


Figure 1.14: Truncated impulse response: linear-phase, but non-causal

The next option is shown in Figure 1.15 which, despite being causal, does not have a linear-phase response (the most important property of an FIR system).

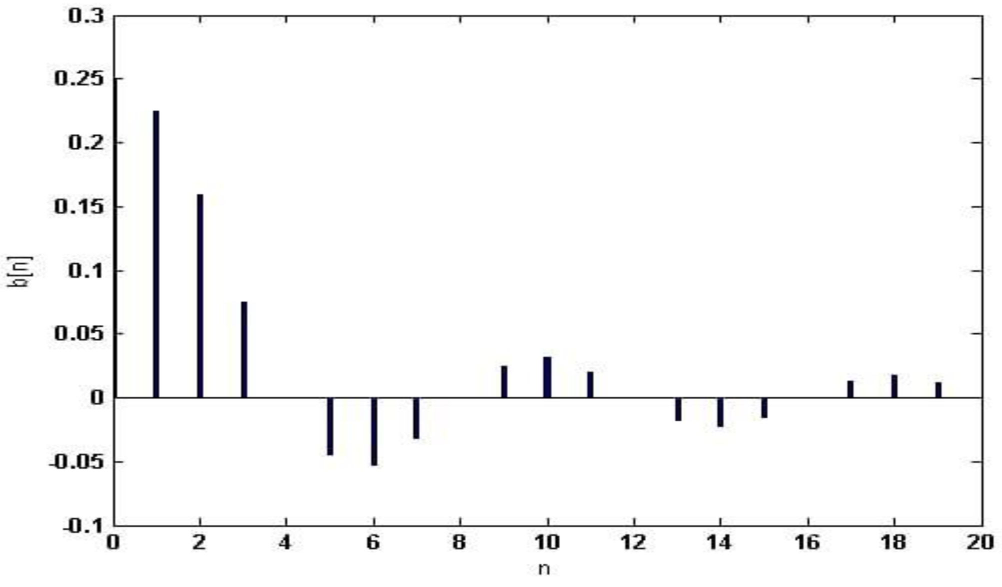


Figure 1.15: Truncated impulse response: causal, but nonlinear-phase

The last option is shown in Figure 1.16. This system is both causal and linear phase. The only drawback to this system is its delay which is $\frac{M-1}{2}$ samples. In other words, in response to an impulse at $n = 0$, the system will not react until almost $n = \frac{M-1}{2}$. This delay may cause problems in some applications.

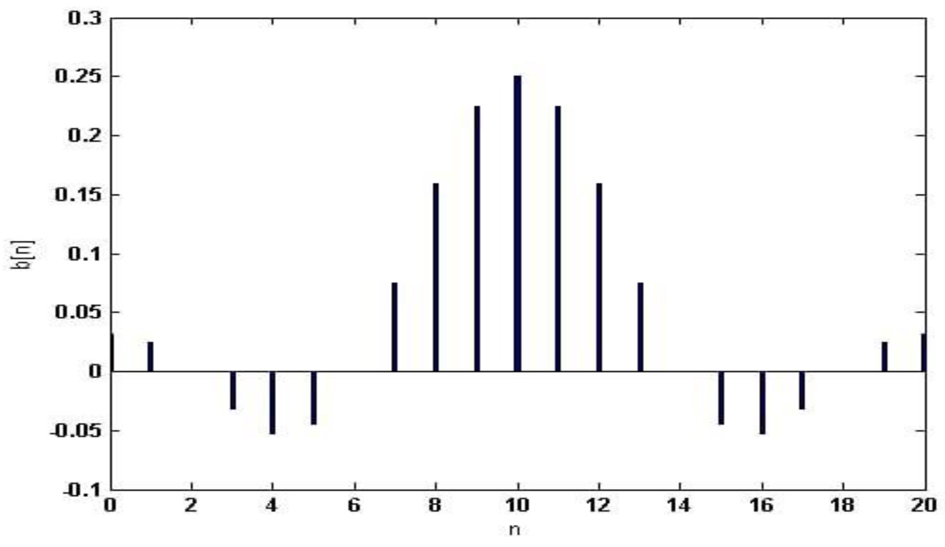


Figure 1.16: Truncated impulse response: causal and linear phase

Truncation of the impulse response is equivalent to multiplying $h_d(\mathrm{n})$ (or its shifted version) by a rectangular window, $\omega(\mathrm{n})$ which is equal to one for $\mathrm{n} = 1, \dots, \mathrm{M} - 1$ and zero otherwise. Therefore, considering the applied shift, we obtain the impulse response of the designed filter:

$$h(\mathrm{n}) = h_d\left[n - \frac{M-1}{2}\right]\omega(\mathrm{n})$$

Clearly the spectrum of the rectangular window will cause the filter response to deviate from the ideal response in Equation (∗). Figure (∗∗) compares the response of the designed filter with that of the ideal one. This figure shows that, unlike the ideal filter, the designed filter has a smoother transition from the passband to the stopband. Moreover, there are some ripples in both the passband and stopband of $H(\omega)$.How can we make the transition band sharper? How can we make the ripples smaller? What other options are there to be used instead of a rectangular window?

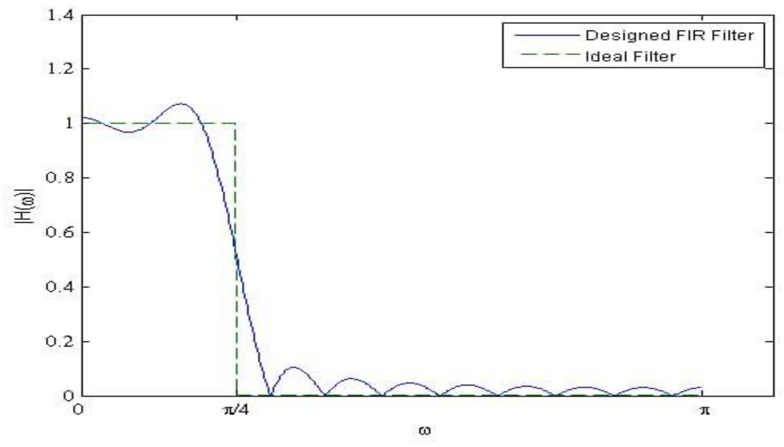


Figure 1.17: Frequency response of the filter designed by a rectangular window

Summary

- To design a digital filter, we need to find the coefficients, a_k and b_k.
- An FIR filter is a special case of Equation $H(z) = \frac{\sum_{k=0}^{M-1} b_k z^{-k}}{\sum_{k=0}^{N-1} a_k z^{-k}}$, where $a_0 = 1$ and $a_k = 0$ for $\mathrm{k} = 1, \dots, \mathrm{N} - 1$.
- Stability and linear-phase response are the two most important advantages of an FIR filter over an IIR filter.
- A linear-phase frequency response corresponds to a constant delay.
- Truncation of the impulse response is equivalent to multiplying $h_d(\mathrm{n})$ by a rectangular window, $\omega(\mathrm{n})$, which is equal to one for $\mathrm{n} = 1, \dots, \mathrm{M} - 1$ and zero otherwise.
- A wider transition band and ripples in the passband and stopband are the most important differences between the ideal filters and those designed by window method.

1.4.4.2 Frequency Sampling Method

The frequency sampling method for filter design is based on the frequency response of an ideal continuous filter $H(f)$, for which the exact mathematical formula may be unknown. As a result, we cannot directly compute the time-domain impulse response $h(n)$ through the inverse Fourier transform of $H(f)$. Instead, we approximate the desired response using a set of sampled frequency values and apply the inverse Discrete Fourier Transform (IDFT) to obtain $h(n)$. This means we "sample" the desired response in the frequency domain, obtaining N points from this frequency response, which correspond to N points of the equivalent time-domain response obtained through the inverse DFT as follows:

We begin by sampling $H(f)$: $H(k) = H(k)|_{\frac{k}{N}}\ k = \frac{-(N-1)}{2}\ to\ \frac{(N-1)}{2}$

then the inverse DFT is applied: $h(n) = \frac{1}{N} \sum_{k=\frac{-(N-1)}{2}}^{k=\frac{(N-1)}{2}} H(k) e^{\frac{2\pi jkn}{N}}$

This synthesis method is very simple and allows the realization of any filter shape (something that cannot be achieved with the previous method). However, this synthesis method only guarantees the frequency points $H(k)$. Between these points, the value of $H(f)$ is not controlled, and oscillations may occur that are not evenly distributed, with the maximum error between the ideal response and the obtained response typically occurring around the transition band. To obtain the frequency response of the final filter, one can apply a DFT to the obtained impulse response $h(n)$ of size N, after adding a large number of zeros. Additionally, due to the use of an inverse DFT on N points, the resulting impulse response $h(n)$ is periodic with a period N, even though the desired ideal impulse response is not of limited duration.

Example: We aim to implement an ideal low-pass filter in digital form with a cutoff frequency $f_c = \frac{f_0}{10}$ and $\Delta f < f_c = \frac{f_0}{16}$. We therefore take N=17, which gives us Δf =0.0588.

We have: $N = 17$, $H(0) = H(-1) = H(1) = 1$ and $H(2) = H(-2) = \cdots = H(8) = H(-8) = 0$

We can deduce the values of the impulse response through the Discrete Fourier Transform (DFT):

$$h(n) = \frac{1}{N}\sum_{k=\frac{-(N-1)}{2}}^{k=\frac{(N-1)}{2}} H(k)e^{\frac{2\pi jkn}{N}} = \frac{1}{N}(1 + e^{\frac{-2\pi jn}{17}} + e^{\frac{2\pi jn}{17}})$$

$$h(n) = \frac{1}{17}(1 + 2\cos(\frac{2\pi n}{17}) \text{ for } -8 \le n \le 8$$

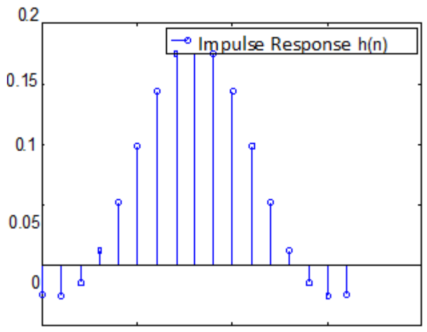
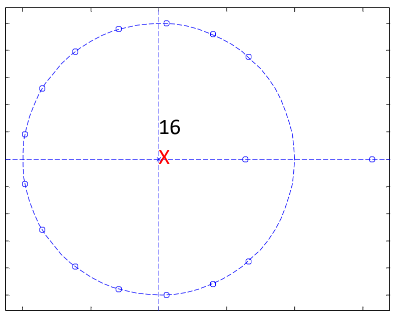


Figure 1.18: Impulse response $h(n)$ of the filter

Finally, to make this filter physically realizable, the impulse response is shifted by 8 samples.



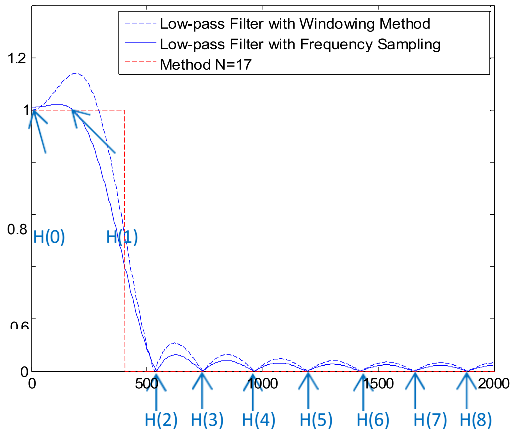


Figure 1.19: Comparison of Filter Responses Using Frequency Sampling and Windowing Methods

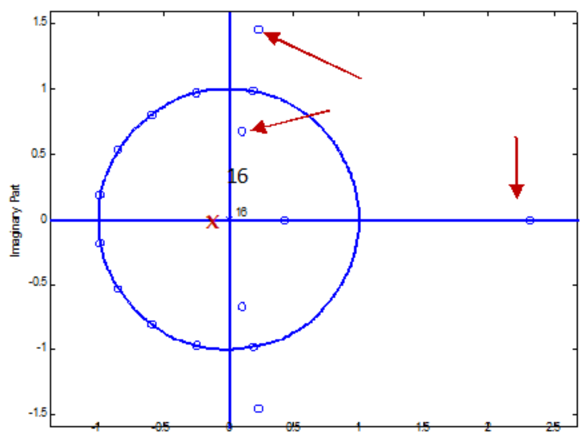
Given that $H^*(k) = H(-k)$ for a real signal $h(n)$, we can generally demonstrate the following:

$$h(n) = \frac{1}{N}(H(0) + 2\sum_{k=1}^{\frac{(N-1)}{2}} H(k)\cos(\frac{2\pi n}{N}) \text{ for } -\frac{N}{2} \le n \le \frac{N}{2}$$

The reduction of oscillations can also be achieved through windowing. Below are the coefficients of the filter $h(n)$, followed by those obtained after applying the Hamming window $h'_N(n)$.

Table 1.2: Filter Coefficients $h(n)$ and Reduced Oscillations Using Hamming Window $h'_N(n)$

n	0	1	2	3	4	5	6	7	8
h(n)	-0,0257	-0,0269	-0,0153	0,0114	0,0514	0,0985	0,1430	0,1749	0,1865
h'ₙ(n)	-0,0020	-0,0031	-0,0033	0,0041	0,0279	0,0705	0,1237	0,1688	0,1865
n	9	10	11	12	13	14	15	16	
h(n)	0,1749	0,1430	0,0985	0,0514	0,01139	-0,0153	-0,0269	-0,0257	
h'ₙ(n)	0,1688	0,1237	0,0705	0,0279	0,0041	-0,0032	-0,0031	-0,0020	



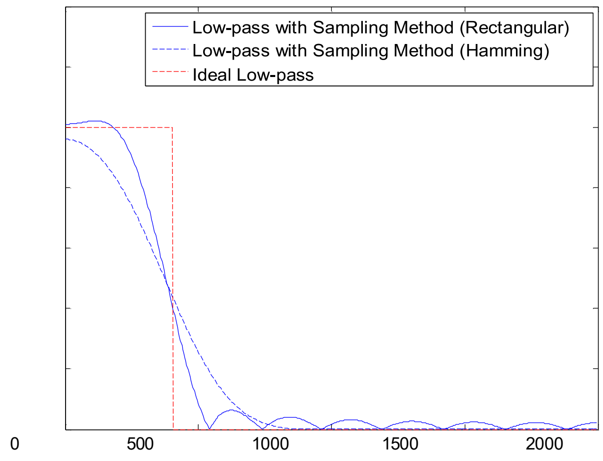


Figure 1.20: Pole-Zero Plot in the Complex Plane

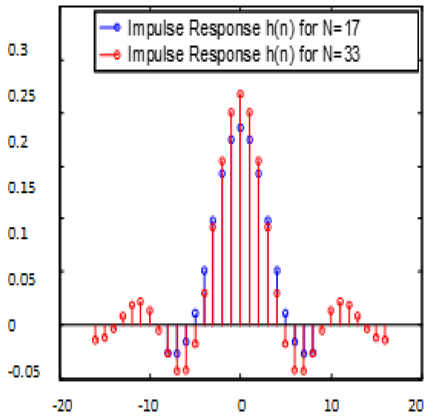
Remarks

When choosing N, ensure that $\frac{f_e}{N}$ is less than Δf. Additionally, the ripples can be somewhat mitigated by smoothing the transitions in the filter's frequency response. To do this, 0.5 will be introduced between 1 and 0. However, this will increase Δf, so N will need to be adjusted accordingly ($2\Delta f < \frac{f_e}{16}$). This will result in a value of N of 33 and Δf =0.0303.

We will set $H(3) = H(-3) = 0.5$ Thus, $H(0) = H(-1) = H(1) = H(2) = H(-2) = 1$

$$h(n) = \frac{1}{33}\left(1 + 2\cos\left(\frac{2\pi n}{33}\right) + 2\cos\left(\frac{4\pi n}{33}\right) + \cos\left(\frac{6\pi n}{33}\right)\right)$$

For $-16 \leq n \leq 16$



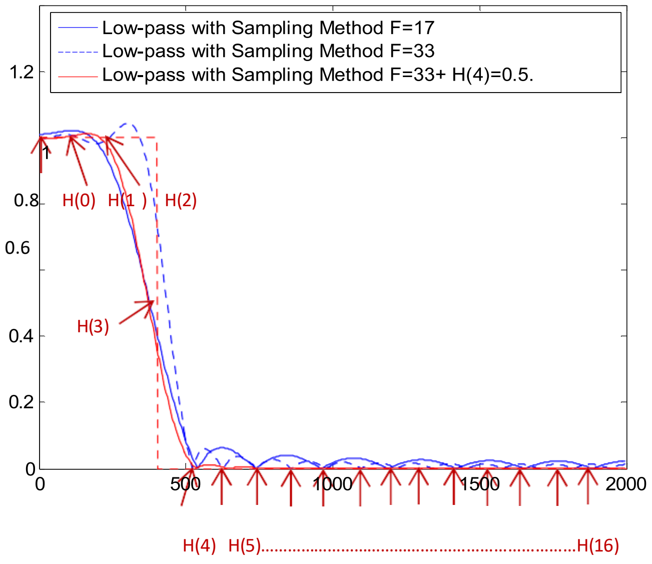


Figure 1.21: Comparison of Impulse Responses h(n) for Different N Values (N=17 and N=33)

1.5 Infinite Impulse Response (IIR) Digital Filters

Analog filters inherently have an infinite impulse response. IIR digital filters behave similarly, except for the effects caused by discretization. This category of filter is also characterized by a transfer function in the z-domain that contains poles, and by a recursive difference equation, meaning the output $\boldsymbol{y(n)}$ depends on both the current inputs and previous outputs. The equations below show the z-domain transfer function and the corresponding difference equation for the general form of an IIR filter. Here, N is referred to as the filter order.

$$H(z) = \frac{N(z)}{D(z)} = \frac{\sum_{i=0}^{N} b_i z^{-i}}{1 + \sum_{i=1}^{N} a_i z^{-i}}$$

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

1.5.1 IIR Filter Topologies

The following equation shows that an IIR filter $H(z)$ can be represented as the product of two structures: one being an FIR filter $N(z)$, and the other an all-pole IIR filter $\frac{1}{D(z)}$.

$$H(z) = \frac{N(z)}{D(z)} = N(z) \times \frac{1}{D(z)} = \sum_{i=0}^{N} b_i z^{-i} \times \frac{1}{1 + \sum_{i=1}^{N} a_i z^{-i}}$$

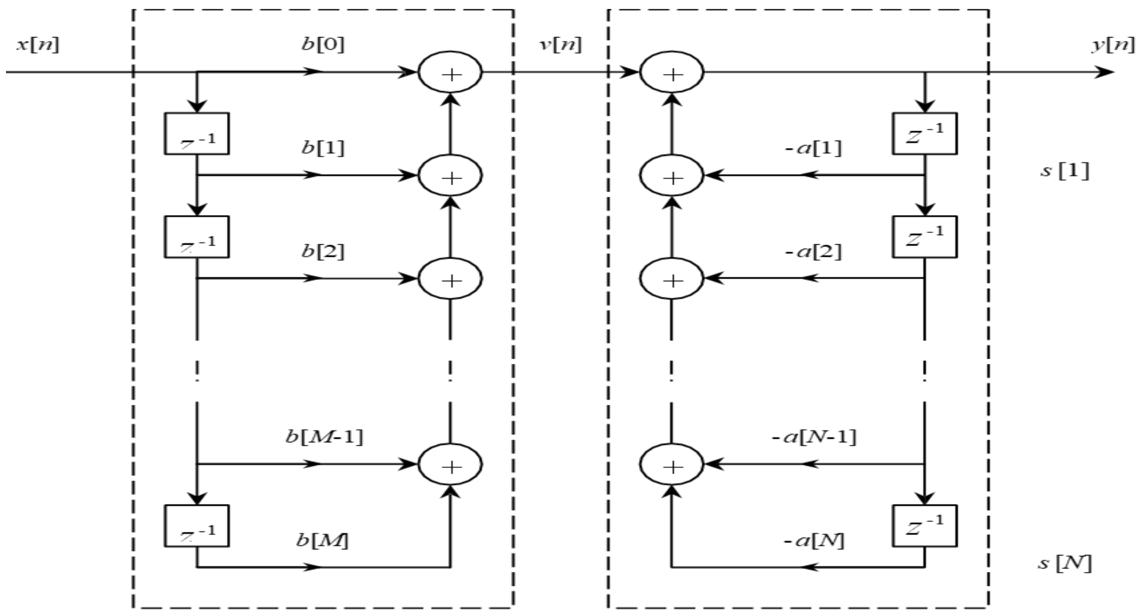
IIR filter topologies can be classified as:

- Direct Type I or Type II
- Transposed

In addition, filters can be either monolithic or built from cascades of smaller (typically second order) sections.

- **Direct Type I**

This is a basic structure for implementing IIR filters, where the filter's transfer function is divided into a numerator (representing the feedforward terms) and a denominator (representing the feedback terms). The input signal is delayed and processed through both the feedforward and feedback sections to generate the output. While it can be used in both fixed-point and floating-point implementations, it may not always provide optimal numerical precision.



- **Direct Type II**

This is an improved variation of the Direct Form Type I structure. The main distinction is that it minimizes the number of delay elements needed by merging the separate delay chains for the numerator and denominator into a single chain. This design can be more efficient, especially in hardware implementations. It is often more stable and provides better numerical precision compared to Type I.

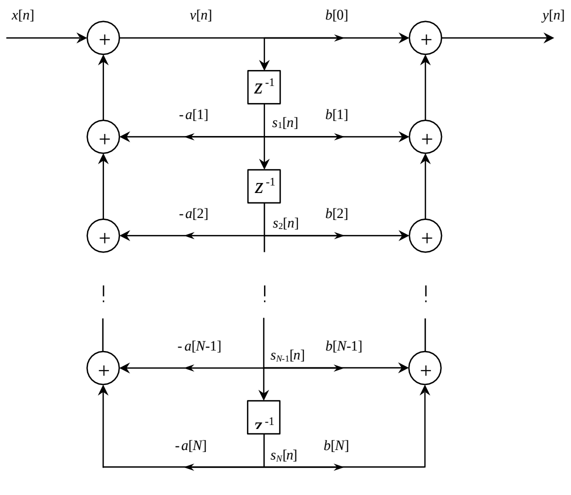


Figure 1.22: Direct Structures of IIR Filters

- **Transposed Type II**

In the transposed form, the multiplication and addition operations are rearranged compared to the direct form structures, with the feedback and feedforward paths swapped. Transposed forms typically offer improved numerical performance, particularly in floating-point implementations, as they reduce rounding errors by performing additions between numbers of similar magnitude. Both Type I and Type II structures have corresponding transposed versions.

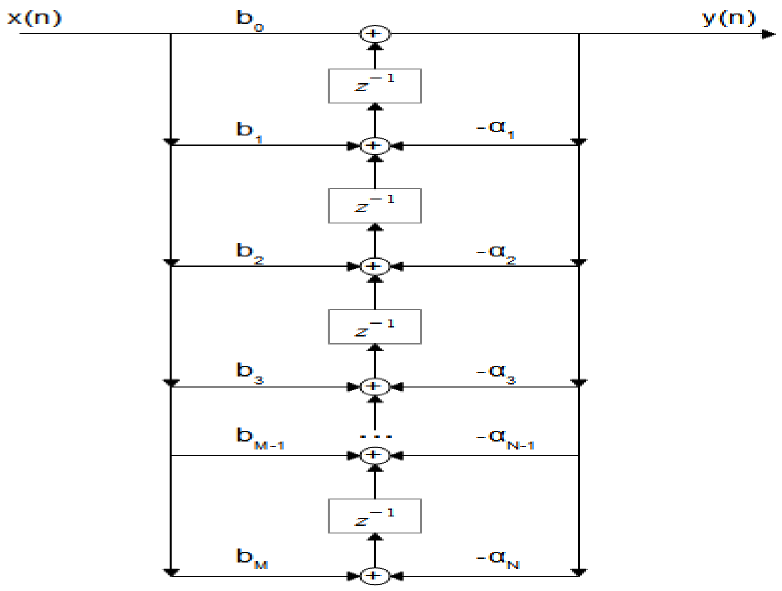


Figure 1.23: Transposed type II of IIR Filters

1.5.2 IIR Filter Design

The design of a digital filter involves finding a function $H(z)$ (or $h(k)$) that meets the specified requirements in the form of a template. This function can be determined using various methods. The most common method is to use analog filter design techniques to obtain a function $H(p)$ that meets the specifications. A function that transforms the P-domain into the Z-domain (i.e., $p = f(z)$) is then used to derive $H(z)$. This transformation must preserve the stability of the analog filter and, as much as possible, maintain the frequency response characteristics of the digital filter. Three types of transformations are commonly used: the impulse invariance method, the Euler transformation, and the bilinear transformation.

- **IIR Filter Design Using Bilinear Transformation**

Principle: An analog filter is provided with a frequency response that meets the required specifications (such as a Butterworth filter).

Objective: To find a digital IIR filter with a frequency response equivalent to that of the analog filter.
Note: The matching of the frequency responses is limited to the useful frequency range of the digital filter, specifically for frequencies between $\frac{f_s}{2}$ and 0.

To obtain the transfer function of the digital filter using bilinear transformation, the variable p in the transfer function of the analog filter is replaced by: $\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$

where T is the sampling period.

Example

Design a digital low-pass Butterworth filter using the bilinear transformation method with the following specifications:

- Analog cutoff frequency: $\omega_c = 1000\ ^{rad}/_{s}$
- Sampling frequency: $F_s = 8000\ Hz$

- Order of the filter: N=1 (first-order Butterworth filter)

Step-by-Step Solution:

Step 1: Define the Analog Filter Transfer Function

The transfer function of a first-order analog Butterworth filter is given by:

$$H(p) = \frac{\omega_c}{p + \omega_c}$$

where ω_c=1000 rad/s is the analog cutoff frequency.

Thus, the analog transfer function becomes:

$$H(p) = \frac{1000}{p + 1000}$$

Step 2: Apply the Bilinear Transformation

The bilinear transformation is given by:

$$\frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}}$$

where $T = \frac{1}{F_s} = 8000 = 0.000125$ seconds is the sampling period.

Substituting this into the analog transfer function $H(p)$, we replace s with the bilinear transformation:

$$H(z) = H(\frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}})$$

First, calculate the term: $p = \frac{2}{0.000125}\frac{1-z^{-1}}{1+z^{-1}}$=16000$\frac{1-z^{-1}}{1+z^{-1}}$

Now, substitute this into the analog transfer function:

$$H(z) = \frac{1000}{16000\frac{1 - z^{-1}}{1 + z^{-1}} + 1000}$$

Simplify the expression:

$$H(z) = \frac{1000(1 + z^{-1})}{16000(1 - z^{-1}) + 1000(1 + z^{-1})}$$

$$H(z) = \frac{1000(1 + z^{-1})}{17000 - 1500z^{-1}}$$

Step 3: Final Digital Filter Transfer Function

Thus, the digital filter's transfer function becomes:

$$H(z) = \frac{1000(1 + z^{-1})}{17000 - 1500z^{-1}}$$

We can simplify this by dividing both the numerator and the denominator by 17000:

$$H(z) = \frac{\frac{1000}{17000}(1 + z^{-1})}{1 - \frac{1500}{17000}z^{-1}}$$

$$H(z) = \frac{0.0588(1 + z^{-1})}{1 - 0.8824z^{-1}}$$

Step 4: Difference Equation Form

The digital filter transfer function can now be converted into the time-domain difference equation using:

$$H(z)\frac{Y(z)}{X(z)} = \frac{0.0588(1 + z^{-1})}{1 - 0.8824z^{-1}}$$

This corresponds to the following difference equation:

$$y(n) - 0.8824y(n-1) = 0.0588 \times x(n) + 0.0588 \times x(n-1)$$

Or equivalently:

$$y(n) = 0.8824y(n-1) + 0.0588 \times x(n) + 0.0588 \times x(n-1)$$

In this example, we designed a digital low-pass Butterworth filter using the bilinear transformation method. We started with an analog Butterworth filter with a cutoff frequency of 1000 rad/s and a sampling frequency of 8000 Hz. After applying the bilinear transformation, we obtained the digital filter's transfer function and the corresponding difference equation. This process can be extended to higher-order filters and other filter types (e.g., high-pass, band-pass, etc.).

1.5.2.1 Advantages of IIR digital filters

- **Low implementation cost**: requires less coefficients and memory than FIR filters in order to satisfy a similar set of specifications, i.e., cut-off frequency and stopband attenuation.
- **Low latency**: suitable for real-time control and very high-speed RF (Radio Frequency) applications by virtue of the low number of coefficients.
- **Analog equivalent**: May be used for mimicking the characteristics of analog filters using p-z plane mapping transforms.

1.5.2.2 Disadvantages of IIR digital filters

- **Non-linear phase characteristics**: The phase charactersitics of an IIR filter are generally nonlinear, especially near the cut-off frequencies. All-pass equalisation filters can be used in order to improve the passband phase characteristics.
- **More detailed analysis:** Requires more scaling and numeric overflow analysis when implemented in fixed point. The Direct form II filter structure is especially sensitive to the effects of quantisation, and requires special care during the design phase.
- **Numerical stability**: Less numerically stable than their FIR (finite impulse response) counterparts, due to the feedback paths.

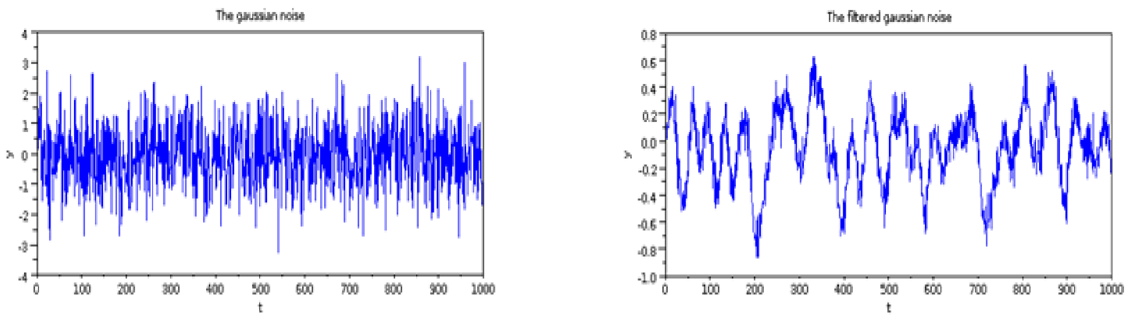


Figure 1.24: Example of filtering on a sound file

Application of a filter for edge detection

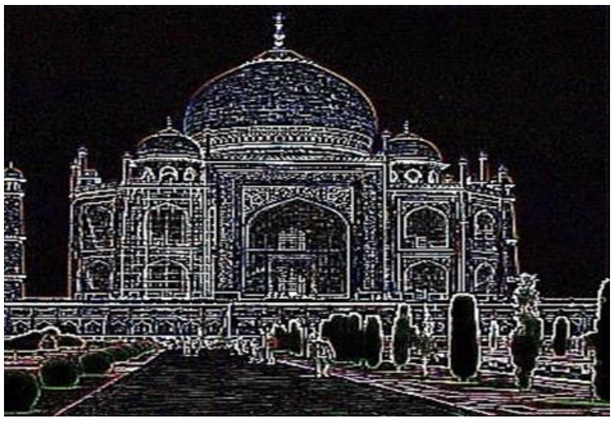


Figure 1.25: Example of filtering on an image file

Chapter 2: Averaging Filter and Median Filter

2.1 Averaging Filter

Physiological signals, such as electrocardiograms (ECG), electroencephalograms (EEG), as well as medical images like MRIs (Magnetic Resonance Imaging) or X-rays, are essential tools for diagnosing and monitoring diseases. However, these signals and images are often affected by various types of noise and artifacts, making their analysis more complex. To enhance the quality of this data and extract clear and reliable information, the application of appropriate filtering techniques is crucial. Among these techniques, the averaging and median filters are particularly common. The averaging filter is a linear filter that replaces each sample or pixel with the average of the neighboring values within a defined window. In equation form, this can be written as :

$$y[i] = \frac{1}{M}\sum_{J=0}^{M-1} x[i+j]$$

Example

$$y(70) = \frac{x(70) + x(71) + x(72) + x(73) + x(74)}{5}$$

Alternatively, the group of points from the input signal can be chosen symmetrically around the output point.

$$y(70) = \frac{x(68) + x(69) + x(70) + x(71) + x(72)}{5}$$

This corresponds to changing the summation in equation (1) from $j = 0$ to $M - 1$ to $\frac{-(M-1)}{2}$ à $\frac{(M-1)}{2}$.

By smoothing rapid variations and reducing random noise, this filter is widely used in the processing of physiological signals and medical images. In the case of physiological signals, it helps to attenuate minor fluctuations that can obscure important information. For medical imaging, the averaging filter reduces background noise, thereby improving the visibility of anatomical structures. However, its application can lead to edge blurring or loss of details, which is a significant limitation when applied to medical images or signals where abrupt transitions are important.

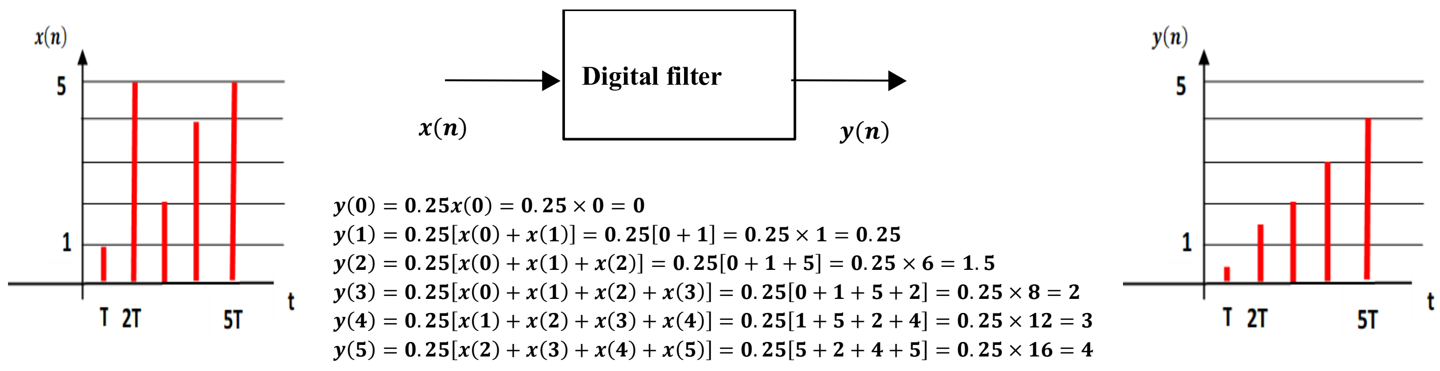
Example

Consider a filter that performs a sliding average on the most recent samples arriving at the input.

The algorithm is written as follows:

$$\begin{aligned} y(n) &= \frac{x(n) + x(n-1) + x(n-2) + x(n-3)}{4} \\ &= 0.25.\,[x(n) + x(n-1) + x(n-2) + x(n-3)] \end{aligned}$$

From the algorithm, it is easy to manually calculate the outputs at times T, $2T$, …to observe the behavior of the filter.



$$y(0) = 0.25x(0) = 0.25 \times 0 = 0$$
$$y(1) = 0.25[x(0) + x(1)] = 0.25[0 + 1] = 0.25 \times 1 = 0.25$$
$$y(2) = 0.25[x(0) + x(1) + x(2)] = 0.25[0 + 1 + 5] = 0.25 \times 6 = 1.5$$
$$y(3) = 0.25[x(0) + x(1) + x(2) + x(3)] = 0.25[0 + 1 + 5 + 2] = 0.25 \times 8 = 2$$
$$y(4) = 0.25[x(1) + x(2) + x(3) + x(4)] = 0.25[1 + 5 + 2 + 4] = 0.25 \times 12 = 3$$
$$y(5) = 0.25[x(2) + x(3) + x(4) + x(5)] = 0.25[5 + 2 + 4 + 5] = 0.25 \times 16 = 4$$

Figure 2.1: Digital filter for averaging four input values

Note: This manual study is only feasible for very simple filters. For more complex filters, software tools are used.

2.1.1 Example Application for a Physiological Signal

An electrocardiogram (ECG) is an electrophysiological signal that describes the electrical behavior of the human heart. The electrocardiographic signal (ECG) requires several measurements from different parts of the patient's body (biomedical instrumentation). However, the presence of electrical equipment introduces disturbances that affect the instrumentation system. The amplitude of the ECG is about 1 mV, and its bandwidth ranges from 0.5 to 100 Hz. Figure 2.2 shows an electrocardiographic signal extracted from the Physionet signal database, along with its components in the frequency domain.

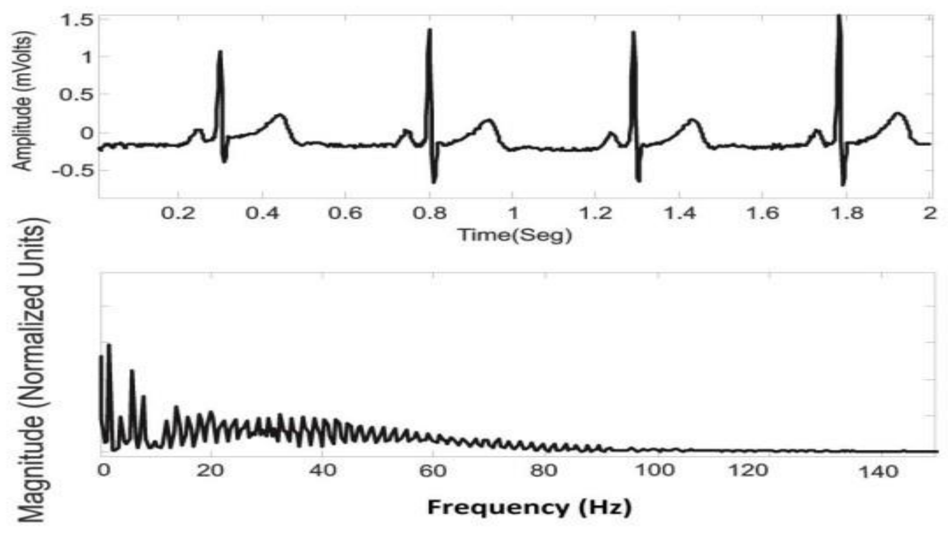


Figure 2.2: Proposed ECG Signal and Its Spectrum

This signal will be modified by adding noise corresponding to the interference generated by nearby electrical lines, which is common in ECG signals once they are acquired. The resulting waveform is shown in Figure 2.3. This new version of the original signal is closer to a real-world scenario where various factors, such as electrical lines, can introduce unwanted data into the time window. Next, the averaging filter can be obtained using a simple MATLAB code.

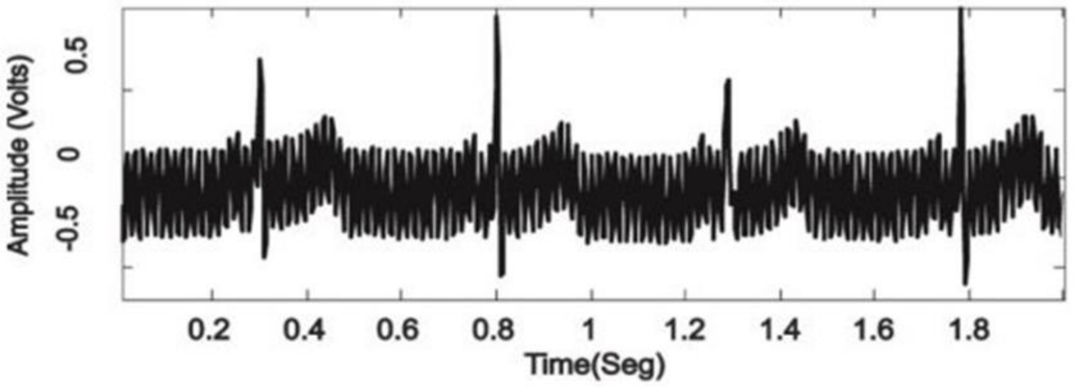


Figure 2.3: ECG Signal Distorted by Electrical Line Noise

As a result, the initially added noise is significantly reduced, as shown in Figure 2.4. This demonstrates that the averaging filter is highly effective as a low-pass filter, removing unwanted components from the studied signal. Additionally, the computational load is not demanding for real-time applications.

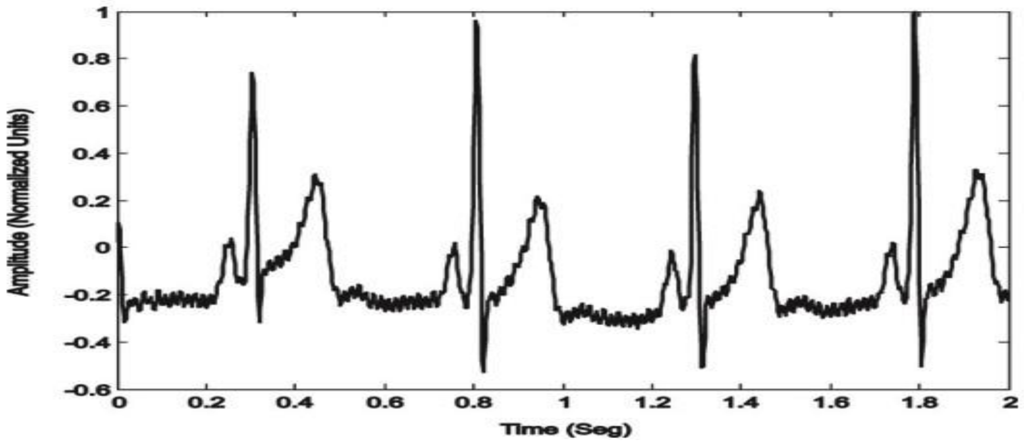
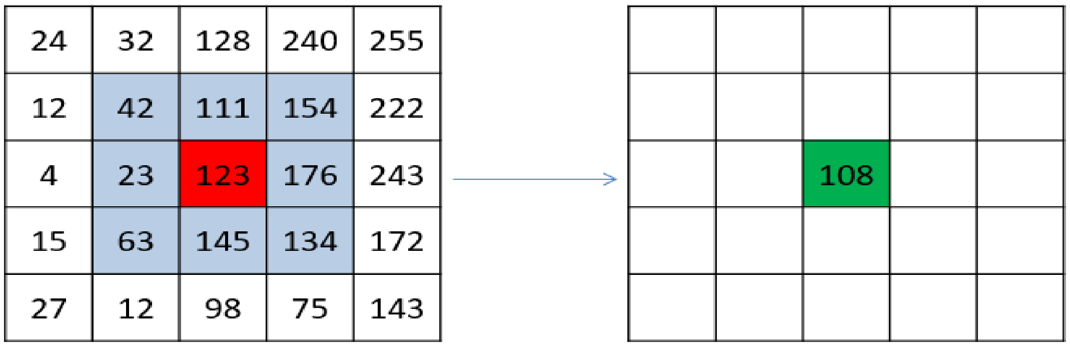


Figure 2.4: Filtered ECG Signal Using an Averaging Filter with M=8

2.1.2 Local Averaging Filter Application in Medical Imaging

The averaging filter is part of the category of local image filters because it computes the new value of a pixel based on the values of neighboring pixels. Specifically, the filtered value of a pixel p is equal to the average of the values of the pixels surrounding p. Generally, the "neighboring pixels of p" are defined as the set of pixels contained within a square of width k centered on p:



24	32	128	240	255
12	42	111	154	222
4	23	123	176	243
15	63	145	134	172
27	12	98	75	143

		108		

Figure 2.5: Example of 3x3 Average Filtering Applied to a Pixel Matrix

With an averaging filter of width 3, to calculate the new value of the red pixel in the original image on the left, we compute the average value of the pixels located within a 3×3 square centered on that pixel. This gives the new value of the pixel in the transformed image (green pixel in the image on the right):

$$\frac{42 + 111 + 154 + 23 + 123 + 176 + 63 + 145 + 134}{9} = 108$$

This operation is repeated for all pixels in the image. The term "sliding window" refers to the square over which the average of the pixels is calculated and which moves across the image.

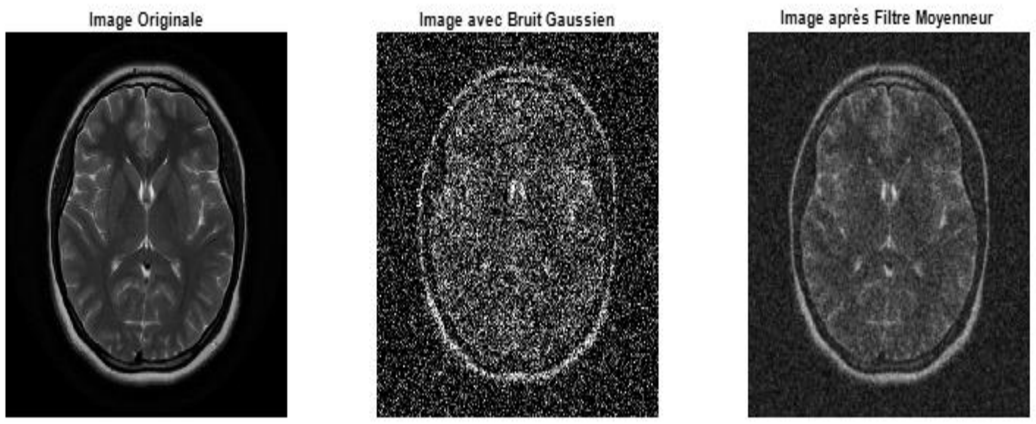


Figure 2.6: Example of Denoising Gaussian Noise Using an Averaging Filter on a Pixel Matrix

MATLAB Implementation

```
clear, clc, close all

 %Load a medical image (for example, a brain scan image)
image = imread('C:\XX\aa.jpeg'); % Replace with your image
image_gray = im2gray(image); % Convert to grayscale if the image is in RGB format
```

```
% Add Gaussian noise to the image
noisy_image = imnoise(image_gray, 'gaussian', 0, 0.1); % Gaussian noise with a variance of 0.01
% Apply an averaging filter with a 5x5 kernel size
kernel_size = 5;
filtered_image = imfilter(noisy_image, fspecial('average', kernel_size));

% Display the images: original, noisy, and filtered
figure;
subplot(1, 3, 1);
imshow(image_gray);
title('Original Image ');
subplot(1, 3, 2);
imshow(noisy_image);
title ('Image with Gaussian Noise');
subplot(1, 3, 3);
imshow(filtered_image);
title ('IImage after Averaging Filter');
```

2.2 Median filter

The median filter, on the other hand, is a nonlinear filter that replaces each sample or pixel with the median of the neighboring values. This type of filtering is particularly effective at removing impulse noise, such as salt-and-pepper artifacts in imaging, or noise spikes in physiological signals. Unlike the averaging filter, the median filter better preserves edges and sharp transitions, which is crucial for diagnostic purposes in medical imaging where fine details are important. In the domain of physiological signals, it is also useful for removing noise spikes while maintaining the essential morphology of the signal.

Thus, the choice between an averaging filter and a median filter depends on the type of noise present and the specific requirements of the analysis. In medical imaging and physiological signal processing, these filters play a crucial role in enhancing data quality, thereby facilitating more accurate and reliable diagnosis.

What Is Median Filtering?

Image noise can be briefly defined as random variations in some of the pixel values of an image. We know filters are used to reduce the amount of noise present in an image, but how does Median filtering work? Let’s use an example 3x3 matrix of pixel values:

$$\begin{bmatrix} 22\ 24\ 27 \\ 31\ 98\ 29 \\ 27\ 22\ 23 \end{bmatrix}$$

Notice the center pixel: the clear outlier in this matrix. Outliers like this can produce what is called salt-and-pepper noise, which produces an image that looks exactly what you might imagine:

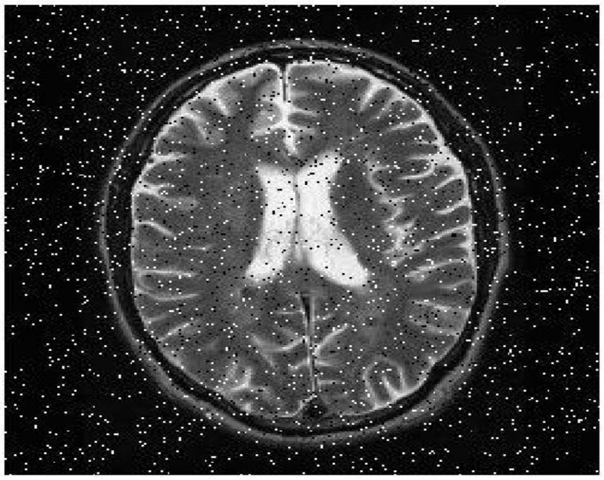
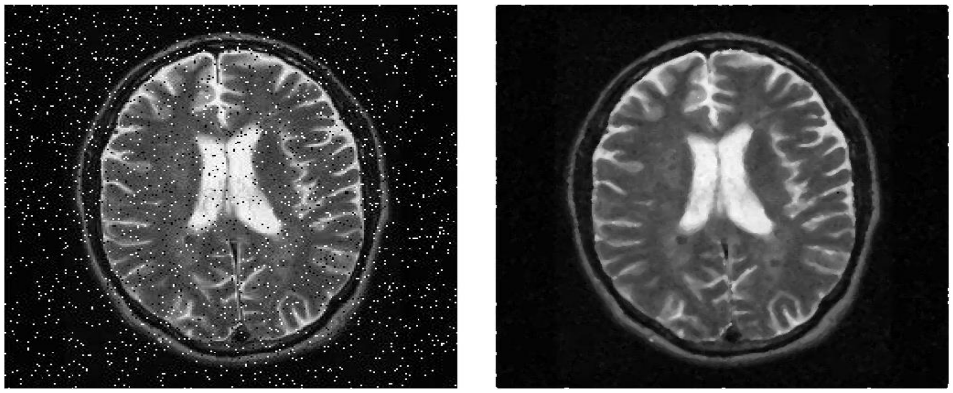


Figure 2.7: Image Degraded by Salt-and-Pepper Noise

This image has a significant amount of salt-and-pepper noise, namely the black and white pixels that appear out of place Median filtering is excellent at reducing this type of noise. The filtering algorithm will scan the entire image, using a small matrix (like the 3x3 depicted above), and recalculate the value of the center pixel by simply taking the median of all of the values inside the matrix.

With the example above, the sorted values are [22, 22, 23, 24, **27**, 27, 29, 31, 98], and median of this set is 27. Let's apply the filter and see how it looks:



Left: original image with noise. **Right:** Image with median filter applied.

Figure 2.8: Example of Denoising Salt-and-Pepper Noise Using median filter

Look at that! Basically all of the salt-and-pepper noise is gone! Now, let's compare this to a Gaussian filter and see if there is a difference:

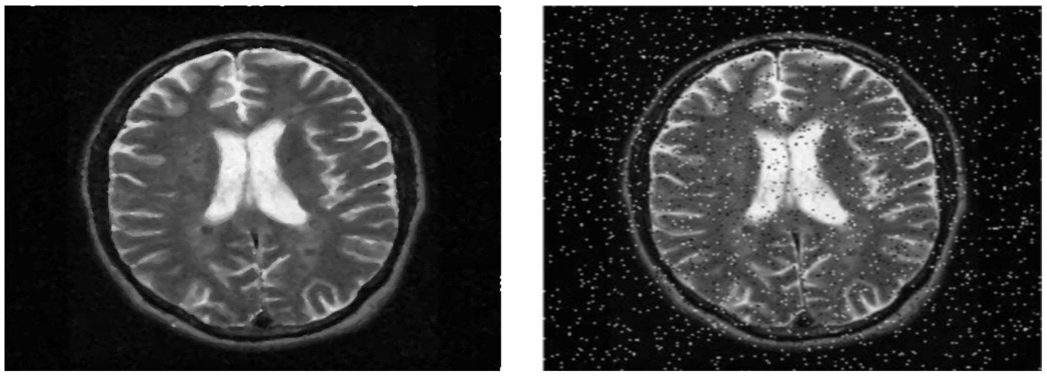


Figure 2.9: Comparison of Median Filtering (Left) and Gaussian Filtering (Right)

As we can see, the Gaussian filter didn’t get rid of any of the salt-and-pepper noise! The neat thing about a median filter is that the center pixel value will be replaced by a value that is present in the surrounding pixels. This differs from Gaussian which will use the weighted average instead, where outliers can heavily skew the average, resulting in almost no noise reduction in this case.

Chapter 3: Discrete Fourier Transform (DFT) and Discrete Cosine Transform (DCT)

3.1 Discrete Fourier Transform (DFT)

When we want to compute the Fourier transform of a function $x(t)$ using a computer, since the computer only has a finite number of words of finite size, we are led to:

- Discretize the time-domain function,
- Truncate the time-domain function,
- Discretize the frequency-domain function.

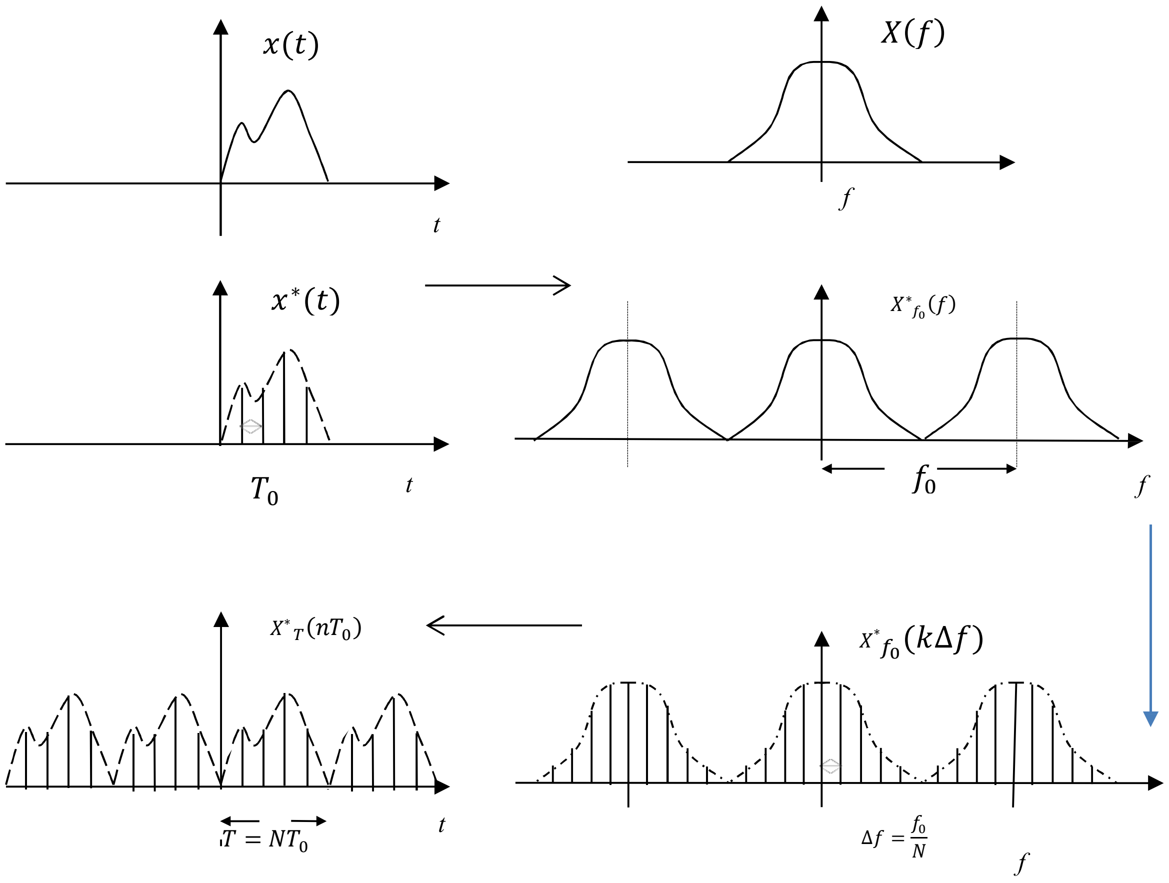


Figure 3.1: Steps for Computing the Fourier Transform on a Computer: Discretization and Truncation

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi f t} dt$$

$$x^*(t) = x(t) \sum_{n=-\infty}^{+\infty} \delta(t - nT_0)$$

$$= \sum_{n=-\infty}^{+\infty} x(nT_0)\delta(t - nT_0)$$

By using the Fourier Transform, we obtain:

$$X^*(f) = \int_{-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} x(nT_0)\delta(t-nT_0)e^{-j2\pi ft}dt$$

$$= \sum_{n=-\infty}^{+\infty} x(nT_0) \int_{-\infty}^{+\infty} \delta(t-nT_0)e^{-j2\pi ft}\,dt$$

$$= \sum_{n=-\infty}^{+\infty} x(nT_0) \int_{-\infty}^{+\infty} \delta(t-nT_0)e^{-j2\pi fnT_0}\,dt$$

$$= \sum_{n=-\infty}^{+\infty} x(nT_0)\, e^{-j2\pi fnT_0} \int_{-\infty}^{+\infty} \delta(t-nT_0)\,dt$$

$$= \sum_{n=-\infty}^{+\infty} x(nT_0)\, e^{-j2\pi fnT_0} \quad \text{where: } \int_{-\infty}^{+\infty} \delta(t-nT_0)\,dt = 1$$

$$X^*(f) = \sum_{n=-\infty}^{+\infty} x(nT_0)\, e^{-j2\pi fnT_0}$$

On the other hand, we have:

$$x^*(t) = x(t)\sum_{n=-\infty}^{+\infty} \delta(t-nT_0)$$

By using the Fourier Transform, we obtain:

$$X^*(f) = X(f) * TF\left[\sum_{n=-\infty}^{+\infty} \delta(t-nT_0)\right]$$

$$TF\left[\sum_{n=-\infty}^{+\infty} \delta(t-nT_0)\right] = TF\left[\sum_{n=-\infty}^{+\infty} C_n e^{j2\pi nf_0t}\right]$$

$$C_n = \frac{1}{T_0}\int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} \delta(t)dt = \frac{1}{T_0} = f_0$$

$$TF\left[\sum_{n=-\infty}^{+\infty} \delta(t-nT_0)\right] = f_0 TF\left[\sum_{n=-\infty}^{+\infty} e^{j2\pi nf_0t}\right]$$

$$= f_0 \int_{-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} e^{j2\pi n f_0 t} \times e^{-j2\pi f t} dt$$

$$= f_0 \int_{-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} e^{-j2\pi(f-nf_0)t} dt$$

$$= f_0 \sum_{n=-\infty}^{+\infty} \int_{-\infty}^{+\infty} e^{-j2\pi(f-nf_0)t} dt$$

$$= f_0 \sum_{n=-\infty}^{+\infty} \delta(f - nf_0)$$

$$X^*(f) = X(f) * TF\left[\sum_{n=-\infty}^{+\infty} \delta(n - nT_0)\right]$$

$$X^*(f) = X(f) * f_0 \sum_{n=-\infty}^{+\infty} \delta(f - nf_0)$$

$$= f_0 \int_{-\infty}^{+\infty} X(\tau) \sum_{n=-\infty}^{+\infty} \delta[(f - nf_0) - \tau] d\tau$$

$$= f_0 \int_{-\infty}^{+\infty} X(\tau) \sum_{n=-\infty}^{+\infty} \delta[\tau - (f - nf_0)] d\tau$$

$$= f_0 \sum_{n=-\infty}^{+\infty} X(f - nf_0)$$

$$X^*(f) = f_0 \sum_{n=-\infty}^{+\infty} X(f - nf_0)$$

$f = k\Delta f$

$f_0 = N\Delta f \rightarrow \Delta f = \frac{f_0}{N} = \frac{1}{NT_0}$

$$X^*(f) = \sum_{n=-\infty}^{+\infty} C_n e^{-j2\pi n T_0 f}$$

$$C_n = \frac{1}{f_0}\int_{-\frac{f_0}{2}}^{\frac{f_0}{2}} X^*(f)e^{j2\pi nT_0 f}\,df$$

$$X^*(f) = \sum_{n=-\infty}^{+\infty} \frac{1}{f_0}\int_{-\frac{f_0}{2}}^{\frac{f_0}{2}} X^*(f)e^{j2\pi nfT_0}df e^{-j2\pi nfT_0}$$

We also have:

$$X^*(f) = \sum_{n=-\infty}^{+\infty} x(nT_0)\, e^{-j2\pi fnT_0}$$

By identification, we obtain:

$$x(nT_0) = \frac{1}{f_0}\int_{-\frac{f_0}{2}}^{\frac{f_0}{2}} X(f)e^{j2\pi nfT_0}df$$

$$X^*(f) = \sum_{n=-\infty}^{+\infty} x(nT_0)\, e^{-j2\pi fnT_0}$$

$$X^*(k\Delta f) = \sum_{n=-\infty}^{+\infty} x(nT_0)\, e^{-j2\pi k\Delta fnT_0}$$

$$= \sum_{n=-\infty}^{+\infty} x(nT_0)\, e^{-j2\pi k\frac{f_0}{N}nT_0}$$

$$= \sum_{n=-\infty}^{+\infty} x(nT_0)\, e^{-j2\pi\frac{nk}{N}}$$

$$X(k) = \sum_{n=-\infty}^{+\infty} x(nT_0)\, e^{-j2\pi\frac{nk}{N}}$$

$$x(nT_0) = \frac{1}{f_0}\int_{-\frac{f_0}{2}}^{\frac{f_0}{2}} X(f)e^{j2\pi nfT_0}df$$

$$x(n) = \frac{1}{f_0}\int_{-\frac{f_0}{2}}^{\frac{f_0}{2}} X(k\Delta f)e^{j2\pi nk\Delta f_0}df$$

$$x(n) = \frac{1}{N}\sum_{n=-\frac{N}{2}}^{\frac{N}{2}} X(k)\, e^{j2\pi\frac{1}{N}nk}$$

$$x(n) = \frac{1}{N}\sum_{n=-\frac{N}{2}}^{\frac{N}{2}} X(k)\, e^{j2\pi\frac{nk}{N}}$$

In relation to the equations derived in the "DFT" section, it is helpful to introduce the following substitution:

$$W_N^{nk} = e^{-j\,\frac{2\pi nk}{N}}$$

The W_N^{nk} element in this substitution is also called the "twiddle factor." With respect to this substitution, we may rewrite the equation for computing the DFT and IDFT into these formats:

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

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

3.1.1 Propriétés de la TFD

To enhance the efficiency of computing the DFT, certain properties of W_N^{nk} are exploited. These properties stem from the graphical representation of the twiddle factor as a rotational vector for each *nk* value. They are described as follows:

3.1.1.1 Periodicity

The sequence $X(k)$ is a periodic sequence with a period of N.

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

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

$$\boxed{e^{-j.2.\pi.n} = 1\forall n}$$

3.1.1.2 Symetry

$$X(-k) = \frac{1}{N}.\sum_{n=0}^{N-1} x(n).e^{-j.\frac{2.\pi}{N}.(-k).n} = X(k) = \frac{1}{N}.\sum_{n=0}^{N-1} X_n.\left(e^{-j.\frac{2\pi}{N}.k.n}\right)^* = X^*(k)$$

where $X^*(k)$ denotes the complex conjugate of $X(k)$.

Example

Consider the sequence:

$x(n)$= [3, −1, 2, −2]

We will calculate the DFT using the matrix form for N=4.

DFT Matrix Form:

The DFT matrix W_N of size N×N is given by:

$$W_N^{nk} = e^{-j\frac{2\pi}{N}kn}$$

For N=4, the matrix W_4 is: $e^{-j\pi}$

$$W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-j\frac{\pi}{2}} & e^{-j\pi} & e^{-j\frac{3\pi}{2}} \\ 1 & e^{-j\pi} & e^{-2j\pi} & e^{-3j\pi} \\ 1 & e^{-j\frac{3\pi}{2}} & e^{-3j\pi} & e^{-j\frac{9\pi}{2}} \end{bmatrix}$$

Substituting values:

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

$$e^{-j\pi} = -1$$

$$e^{-j\frac{3\pi}{2}} = j$$

$$e^{-j2\pi} = 1$$

The DFT matrix becomes:

$$W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & 1 \\ 1 & j & -1 & -j \end{bmatrix}$$

Step 1: Input Sequence $x(n)$:

The input sequence is:

$$x = \begin{bmatrix} 3 \\ -1 \\ 2 \\ -2 \end{bmatrix}$$

Step 2: Perform Matrix Multiplication

Now multiply the DFT matrix W_4 by the input sequence x:

$$X = W_4 x = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & 1 \\ 1 & j & -1 & -j \end{bmatrix} \times \begin{bmatrix} 3 \\ -1 \\ 2 \\ -2 \end{bmatrix}$$

Perform the multiplication row by row:

$X(0)$=1×3+1×(−1)+1×2+1×(−2)=3−1+2−2=2

$X(1)$=1×3+(−j)×(−1)+(−1)×2+j×(−2) =3+j−2−2j=1−j

$X(2)$=1×3+(−1)×(−1)+1×2+(−1)×(−2)=3+1+2+2=8

$X(3)$=1×3+j× (−1) +(−1) ×2+(−j)×(−2)=3−j−2+2j= 1 + j

The DFT of the sequence $x(n)$= [3, −1, 2, −2] is: $X(k)$= [2, 1−j, 8, 1+j]

In this example, we computed the DFT of the sequence $x(n)$= [3, −1, 2, −2] using matrix multiplication. The result $X(k)$=[2,1−j,8,1+j] provides the frequency domain representation of the input sequence, showing how both positive and negative values contribute to the DFT. The result contains both real and imaginary components, representing the magnitude and phase of the frequency components.

3.1.2 Fast Fourier Transform (FFT)

The Fast Fourier Transform is an efficient algorithm for computing the Discrete Fourier Transform (DFT) and its inverse. However, the direct computation of the DFT has a high computational cost, requiring O (N^2) operations for a signal of length N, making it impractical for large datasets. The FFT significantly reduces this computational complexity to O ($NlogN$) by taking advantage of the symmetry and periodicity properties of the twiddle factors (the complex exponential terms in the DFT). This allows for faster and more efficient processing, especially for large data sets.

The most commonly used FFT algorithm is the Cooley-Tukey algorithm, which recursively breaks down a DFT of size NNN into smaller DFTs, usually by splitting the signal into even and odd indexed components. This divide-and-conquer approach enables faster calculations, making FFT crucial in various real-time applications like audio processing, image compression, radar, and more.

FFT revolutionized the field of digital signal processing by enabling practical and efficient analysis of signals in both time and frequency domains.

3.1.2.1 Radix-2 decimation in time FFT description

The basic idea of the FFT is to decompose the DFT of a time-domain sequence of length N into successively smaller DFTs whose calculations require fewer arithmetic operations. This is known as a divide-and-conquer strategy, made possible using the properties described in the previous section. The decomposition into shorter DFTs may be performed by splitting an N-point input data sequence *x(n)* into two *N/2*-point data sequences *a(m)* and *b(m),* corresponding to the even-numbered and odd-numbered samples of *x(n),* respectively, that is:

- *a(m)=x(2m),* that is, samples of *x(n)* for n = *2m*
- *b(m)=x(2m+1),* that is, samples of *x(n)* for n = *2m + 1*

where m is an integer in the range of $0 \leq m < N/2$.

The DFT of $x(n)$ is given by:

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

We split this sum into even and odd indices:

$$\begin{aligned} X(k) &= \sum_{n=0}^{N/2-1} x(2m) \, . \, W_N^{2mk} + \sum_{n=0}^{N/2-1} x(2m+1) \, . \, W_N^{(2m+1)k} \\ &= \sum_{\mathrm{n}=0}^{\mathrm{N}/2-1} \mathrm{x}(2\mathrm{m}) \, . \, \mathrm{W}_\mathrm{N}^{2\mathrm{mk}} + \mathrm{W}_\mathrm{N}^\mathrm{k} \sum_{\mathrm{n}=0}^{\mathrm{N}/2-1} \mathrm{x}(2\mathrm{m}+1) . \, \mathrm{W}_\mathrm{N}^{2\mathrm{mk}} \\ &= \sum_{\mathrm{n}=0}^{\frac{\mathrm{N}}{2}-1} \mathrm{a}(\mathrm{m}) \, . \, \mathrm{W}_{\frac{\mathrm{N}}{2}}^{\mathrm{mk}} + \mathrm{W}_\mathrm{N}^\mathrm{k} \sum_{\mathrm{n}=0}^{\frac{\mathrm{N}}{2}-1} \mathrm{b}(\mathrm{m}) . \, \mathrm{W}_{\frac{\mathrm{N}}{2}}^{\mathrm{mk}} \\ &= A(k) \end{aligned}$$

$$0 \leq \mathrm{k} \leq \mathrm{N}$$

These two summations represent the N/2-point DFTs of the sequences *a(m)* and *b(m),* respectively.

Thus, DFT*[a(m)] = A(k)* for even-numbered samples, and DFT*[b(m)] = B(k)* for odd-numbered samples.

Thanks to the periodicity property of the DFT, the outputs for N/2 ≤ k < N from a DFT of length N/2 are identical to the outputs for $0 \leq k < N/2$.

That is, $A(k+N/2) = A(k)$ and $B(k + N/2) = B(k)$ for $0 \leq k < N/2$.

In addition, the factor $W_N^{k+N/2} = -W_N^K$ thanks to the symmetry property.

Thus, the whole DFT can be calculated as follows:

$$X(k) = A(k) + W_N^k B(k)$$
$$X(k + N/2) = A(k) - W_N^k B(k)$$
$$0 \leq \mathrm{k} \leq \mathrm{N}/2$$

This result, expressing the DFT of length recursively in terms of two DFTs of size N/2, is the core of the radix-2 DIT FFT. Note, that final outputs of X(k) are obtained by a +/- combination of A(k) and B(k) W, which is simply a size 2 DFT. These combinations can be demonstrated by a simply-oriented graph, sometimes called "butterfly" in this context (see Figure 3.2).

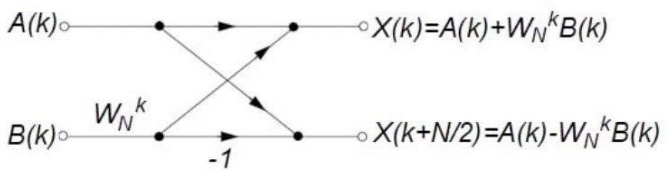


Figure 3.2: Basic butterfly computation in the DIT FFT algorithm

The procedure of computing the discrete series of an *N*-point DFT into two *N/2*-point DFT s may be adopted for computing the series of *N/2*-point DFTs from items of *N/4*-point DFT s. For this purpose, each *N/2*-point sequence should be divided into two sub-sequences of even and odd items, and computing their DFTs consecutively. The decimation of the data sequence can be repeated again and again until the resulting sequence is reduced to one basic DFT.

For illustrative purposes, Figure 3.3 depicts the computation of an *N*= *8*-point DFT. We observe that the computation is performed in three stages (*3 = log28*), beginning with the computations of four 2-point DFTs, then two 4-point DFTs, and finally, one 8-point DFT. Generally, for an *N*-point FFT, the FFT algorithm decomposes the DFT into log2N stages, each of which consists of *N/2* butterfly computations. The combination of the smaller DFTs to form the larger DFT for *N*= *8* is illustrated in Figure 3.4.

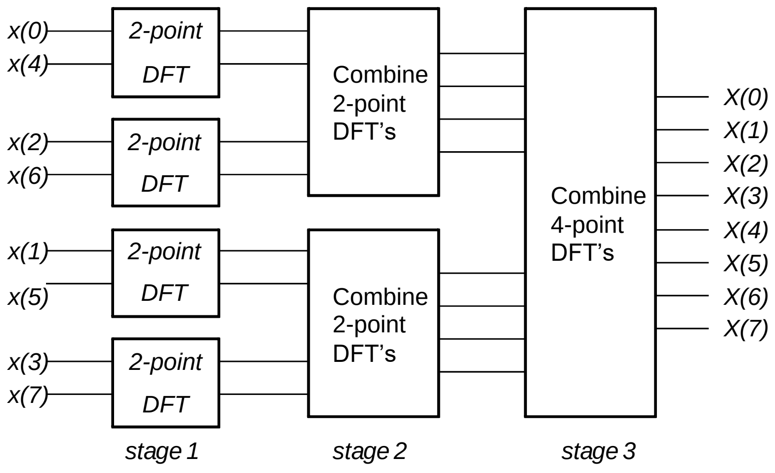


Figure 3.3: Decomposition of an 8-point DFT

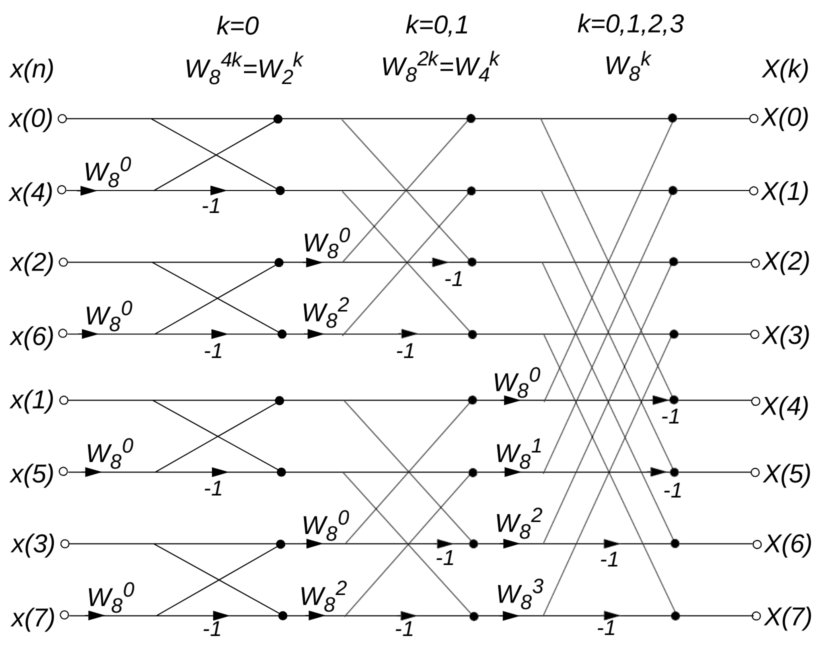

k=0
W8^4k=W2^k
k=0,1
W8^2k=W4^k
k=0,1,2,3
W8^k
x(n)
X(k)
x(0) x(4) x(2) x(6) x(1) x(5) x(3) x(7)
X(0) X(1) X(2) X(3) X(4) X(5) X(6) X(7)
W8^0 W8^1 W8^2 W8^3
-1

Figure 3.4: 8-point radix-2 DIT FFT algorithm data flow

Each dot represents a complex addition and each arrow represents a complex multiplication, as shown in Figure 3.4 The W_N^k factors in Figure 3.4 may be presented as a power of two (W_2) at the first stage, as a power of four (W_4) at the second stage, as a power of eight (W_8) at the third stage, and so on. It is also possible to represent it uniformly as a power of N (W_N), where Nis the size of the input sequence x(n).

3.1.2.2 Radix-2 decimation in time FFT requirements

For effective and optimal decomposition of the input data sequence into even and odd sub-sequences, it is good to have the power-of-two input data samples (.... 64, 128, and so on).

The first step before computing the radix-2 FFT is re-ordering of the input data sequence (see also the left side of Figure 3.4 and Figure 3.5). This means that this algorithm needs a bit-reversed data ordering: that is, the MSBs become LSBs, and vice versa. Table 3.1 shows an example of a bit-reversal with an 8-point input sequence.

Table 3.1: Bit reversal with an 8-point input sequence

Decimal number	0	1	2	3	4	5	6	7
Binary equivalent	000	001	010	011	100	101	110	111
Binary equivalent	000	100	010	110	001	101	011	111
Decimal equivalent	0	4	2	6	1	5	3	7

3.1.2.3 Fast Fourier Transform (FFT) in ECG Signal Analysis:

FFT is widely used to analyze Electrocardiogram (ECG) signals, which measure heart activity. By converting the ECG signal from the time domain to the frequency domain, FFT helps reveal important frequency components related to heart function.

Key Applications:

- **Heart Rate Variability (HRV)**: FFT breaks down RR intervals into frequency bands (low and high), indicating autonomic nervous system activity.
- **Arrhythmia Detection**: Abnormal heart rhythms, like atrial fibrillation, show distinctive high-frequency components.
- **Myocardial Ischemia**: Shifts in specific frequency components can indicate reduced blood flow to the heart.

FFT enhances the ability to diagnose heart conditions by making hidden patterns in ECG data visible.

3.1.2.4 Application of FFT in ECG Signal Analysis using MATLAB

Below is a step-by-step guide to applying FFT to an ECG signal in MATLAB. This example will show how to load an ECG signal, preprocess it, apply FFT, and visualize the frequency spectrum.

A. Load or Simulate ECG Signal

You can either load an actual ECG dataset or simulate one using built-in MATLAB functions.

MATLAB Implementation

```
clear, clc, close all

% Simulate an ECG signal (you can replace this with actual ECG data)
Fs = 500; % Sampling frequency (500 Hz)
t = 0:1/Fs:5-1/Fs; % Time vector for 5 seconds
ecgSignal = ecg(500*5); % Generate ECG signal
Plot the ECG signal in time domain figure;
plot(t, ecgSignal);
title('ECG Signal in Time Domain');
 xlabel('Time (s)');
ylabel('Amplitude');
```

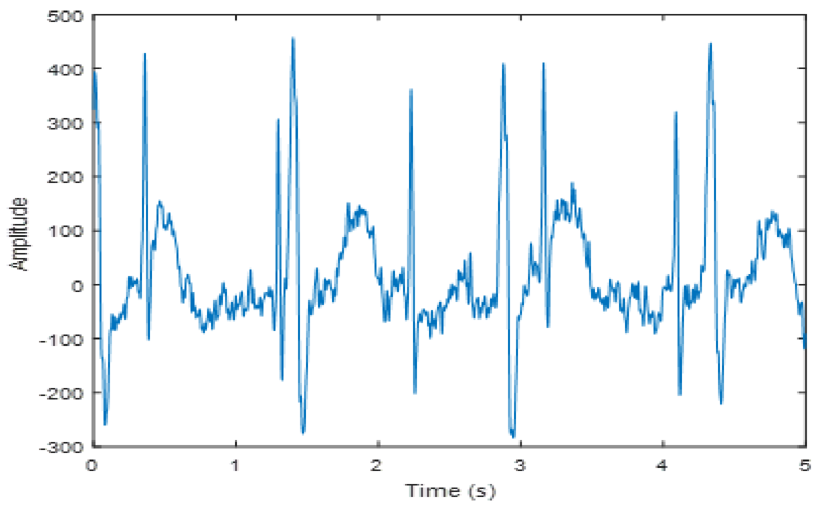


Figure 3.3: ECG signal in time domain

B. Preprocessing the ECG Signal

In practice, you would remove noise such as baseline wander or powerline interference using filters. For simplicity, we will skip this step here.

C. Apply FFT to ECG Signal

MATLAB Implementation

FFT will transform the ECG signal from the time domain to the frequency domain.

```
clear, clc, close all

N = length(ecgSignal); % Number of points in the ECG signal
ecgFFT = fft(ecgSignal); % Compute FFT of the ECG signal

% Compute frequency axis
f = (0:N-1)*(Fs/N); % Frequency vector
P2 = abs(ecgFFT/N); % Two-sided amplitude spectrum
P1 = P2(1:N/2+1); % Single-sided amplitude spectrum
P1(2:end-1) = 2*P1(2:end-1); % Correcting the amplitude

% Plot the FFT result (frequency spectrum)
figure;
plot(f(1:N/2+1), P1);
title('Single-Sided Amplitude Spectrum of ECG Signal');
xlabel('Frequency (Hz)');
ylabel('|P1(f)|');
```

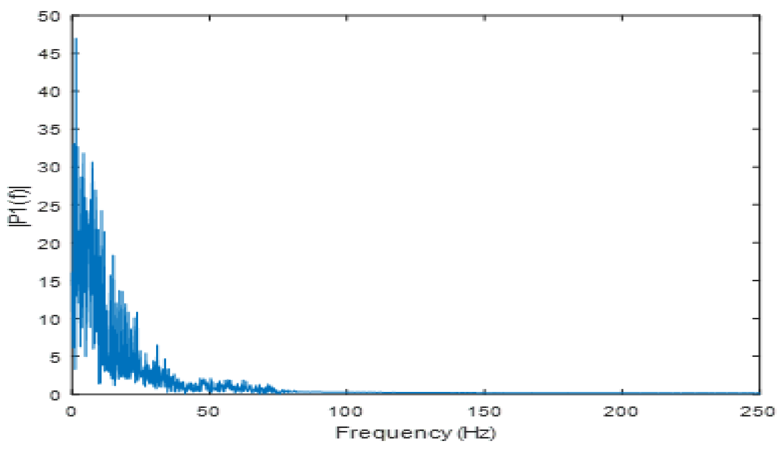


Figure 3.4: Single-sided amplitude spectrum of ECG signal

Interpretation

- The x-axis represents the frequency in Hz.
- The y-axis shows the amplitude of each frequency component.
- You'll observe a dominant peak corresponding to the heart rate (in Hz), and other peaks may represent noise or artifacts.

3.2 Discrete Cosine Transform (DCT)

The DCT is a Fourier-related transform similar to the Discrete Fourier Transform (DFT), but it uses only real numbers (cosine functions) instead of complex exponentials (sines and cosines). The main purpose of the DCT is to express a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies.

For an input signal (or image), the DCT transforms it into a sequence of coefficients that represent the signal's energy distribution over various frequencies. The basic idea is that most of the signal's information is concentrated in the low-frequency components, making it easier to compress.

3.2.1 Mathematical Formulation

The DCT transforms a sequence of real numbers $x(n)$, where $\mathrm{n} = 0,1,2,\dots,\mathrm{N}-1$, into another sequence of real numbers $X(k)$, where $\mathrm{k} = 0,1,2,\dots,\mathrm{N}-1$, which represent the frequency coefficients.

3.2.2 1D DCT Formula:

The most common form is the DCT Type-II, which is typically what is referred to as "the DCT" in most applications. Its formula is:

$$X(k) = \alpha(k) \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi(2n+1)k}{2N}\right), \text{ k=0, 1…, N-1}$$

Where:

- $x(n)$ is the input sequence.
- $X(k)$ is the DCT coefficient for frequency k.
- N is the length of the sequence.
- α(k) is a normalization factor:

$$\alpha(k) = \begin{cases} \sqrt{\dfrac{1}{N}} & if\ \ k = 0 \\ \sqrt{\dfrac{2}{N}} & if\ \ k \neq 0 \end{cases}$$

This normalization ensures that the DCT is orthonormal, which is crucial for certain properties such as energy preservation.

Example

Let's go through a simple example of calculating the DCT of a 4-point sequence.

Input Sequence:

Suppose the input sequence is x=[1,2,3,4].

$$X(k) = \alpha(k) \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi(2n+1)k}{2N}\right), \text{ k=0, 1…, N-1}$$

With N=4, the DCT coefficients $X(k)$ are calculated for k=0,1,2,3

For $k = 0$: $X(0) = \alpha(0) \sum_{n=0}^{N-1} x\,[n] cos\left(\frac{\pi(2n+1)0}{8}\right) = \frac{1}{2}(1 + 2 + 3 + 4) = 5$

For $k = 1$: $X(1) = \alpha(0) \sum_{n=0}^{N-1} x\,[n] cos\left(\frac{\pi(2n+1)1}{8}\right)$

Substituting values, this gives $X(1)$≈−1.148.

Similarly, we calculate $X(2)$ and $X(3)$, leading to the full DCT output

Chapter 4: Concepts of Characteristics and Classification of Physiological Signals

4.1 Characteristics of Physiological Signals

Physiological signals are inherently complex due to their non-stationary nature and high variability. These signals vary based on the organ or system they originate from (e.g., heart, brain, muscles), the physiological state during the recording, and the type of signal being measured. Understanding their characteristics is essential for accurate interpretation and analysis.

4.1.1 Types of Physiological Signals

4.1.1.1 Electrocardiogram (ECG)

Measures the electrical activity of the heart by recording voltage changes generated by the depolarization and repolarization of cardiac tissue. ECG signals are used to diagnose a variety of heart conditions, such as arrhythmias, ischemia, and myocardial infarction. These signals typically range between 0.05–150 Hz, and their characteristic components include the P-wave, QRS complex, and T-wave, which correspond to different phases of the cardiac cycle.

4.1.1.2 Electroencephalogram (EEG)

Captures the brain’s electrical activity by recording the voltage fluctuations produced by the collective firing of neurons. EEG signals are used to study brain function and diagnose neurological disorders such as epilepsy, sleep disorders, and brain injuries. They operate in lower frequency bands (0.1–100 Hz) and are often classified into different frequency ranges: delta, theta, alpha, beta, and gamma waves, each associated with specific brain states.

4.1.1.3 Electromyogram (EMG)

Monitors the electrical activity generated by skeletal muscles during contraction. EMG signals help in diagnosing neuromuscular diseases and assessing muscle function. The amplitude of EMG signals is generally higher compared to EEG and ECG, and they are more susceptible to artifacts due to external movements or noise.

4.2 Characteristics of Physiological Signals

4.2.1 Non-stationary nature

Physiological signals often exhibit non-stationary behavior, meaning their statistical properties, such as mean and variance, change over time. For instance, an ECG waveform might vary due to changes in heart rate, while EEG patterns can shift depending on the subject's level of alertness.

4.2.2 Periodicity

Some physiological signals, such as ECG, display repetitive, cyclical patterns corresponding to biological rhythms (e.g., heartbeats), while others, like EEG, tend to have more irregular patterns with less periodicity. Understanding periodicity is critical for identifying and classifying abnormal patterns, such as arrhythmic heartbeats.

4.2.3 Amplitude and frequency ranges

The amplitude and frequency of physiological signals vary significantly. For example, ECG signals are typically low-frequency (0.05–150 Hz) and have relatively higher amplitude compared to EEG signals, which fall in the range of 0.1–100 Hz and have lower amplitude.

4.2.4 Noise and artifacts

Physiological signals are often contaminated by noise from external sources (e.g., power-line interference) or internal physiological processes (e.g., respiration in ECG). These artifacts can obscure important features and must be filtered out during preprocessing.

4.3 Feature Extraction Methods

Feature extraction is the process of transforming raw physiological signal data into a meaningful set of characteristics that reflect essential information for further analysis or classification. Different approaches are employed depending on whether the analysis is conducted in the time or frequency domain.

4.3.1 Time-Domain Feature Extraction

Time-domain features are derived directly from the signal without transforming it into another domain. These features are particularly useful for capturing the basic statistical properties of physiological signals.

4.3.2 Mean and variance

The mean provides the average amplitude of a signal, while variance indicates the degree of variability over time. These two measures are commonly used to describe the overall behavior of a signal, such as changes in EMG activity during muscle contractions.

4.3.3 Root Mean Square (RMS)

RMS is a standard measure of signal magnitude and is especially useful in EMG analysis, where it reflects the intensity of muscle activity. RMS is calculated by taking the square root of the mean of the squared signal values over a given time window.

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$$

where x_i is the signal value at time i, and N is the total number of samples.

4.3.4 Zero-crossing rate

This feature measures the rate at which a signal crosses the zero-amplitude line. It is especially useful for EMG signals to classify different phases of muscle activity, such as distinguishing between active muscle contraction and relaxation periods.

4.4 Frequency-Domain Feature Extraction

Frequency-domain analysis involves transforming the signal into its frequency components, which is particularly important for analyzing physiological signals like EEG, where different frequency bands are associated with specific brain states.

4.4.1 Discrete Fourier Transform (DFT) and Short-Time Fourier Transform (STFT)

The DFT decomposes a signal into its constituent frequencies, offering insight into its frequency content. STFT extends this by providing time-localized frequency information, making it useful for non-stationary signals such as EEG.

4.4.2 Power Spectral Density (PSD)

PSD describes how the power of a signal is distributed across different frequencies. This measure is particularly important in EEG analysis, where the power within specific frequency bands (e.g., delta, theta, alpha, beta) can provide diagnostic information.

4.4.3 Bandpower

Bandpower refers to the total power of a signal within a specific frequency band. For instance, in EEG analysis, the alpha band (8–13 Hz) is associated with relaxation, and an increase in power in this band may indicate a relaxed mental state.

4.5 Wavelet Transform

The Wavelet Transform (WT) is a powerful tool for analyzing non-stationary signals like ECG and EMG. Unlike the Fourier Transform, which only provides frequency information, WT offers both time and frequency localization, making it highly effective for signals with transient features.

4.5.1 Wavelet coefficients

By decomposing a signal into wavelet coefficients at different scales, time-localized frequency information can be extracted, allowing for a detailed analysis of signal dynamics over time.

4.6 Non-linear Feature Extraction

For more complex physiological signals like EEG, non-linear features such as entropy, fractal dimensions, and Lyapunov exponents are often used. These features capture the chaotic or irregular nature of signals, providing additional insights into the underlying physiological processes.

4.6.1 Detrended Fluctuation Analysis (DFA)

DFA is a specialized technique for analyzing signals that demonstrate self-similarity and for identifying long-term correlations in non-stationary time series. Its straightforward approach and high effectiveness have led to its widespread use in diverse fields such as DNA sequencing, long-term weather data, cloud structure studies, geology, ethnology, economic time series, and solid-state physics. Additionally, it is applied in heart rate variability analysis and EEG signals. Because of the non-linear nature of EEG signals, Fractal Dimension analysis can be utilized to assess the irregularities present in brain activity.

4.7 Classification Approaches

After feature extraction, classifiers are used to distinguish between different physiological states or conditions (e.g., normal vs. abnormal heartbeats). The choice of classifier depends on the complexity and dimensionality of the feature space.

4.7.1 Types of Classifiers

4.7.1.1 Support Vector Machine (SVM)

SVM constructs a hyperplane that maximizes the margin between different classes in the feature space. It is highly effective for handling high-dimensional data and is widely used in physiological signal classification, such as distinguishing between normal and arrhythmic ECG signals.

4.7.1.2 Artificial Neural Networks (ANN)

ANNs consist of interconnected layers of artificial neurons and are capable of learning complex patterns in the data. They are widely used in physiological signal analysis, especially for detecting patterns in noisy and non-linear signals like EEG.

4.7.1.3 K-Nearest Neighbors (KNN)

KNN is a simple yet effective algorithm that classifies data points based on the majority class of their nearest neighbors in the feature space. It is easy to implement and works well for smaller datasets but may struggle with large, high-dimensional data.

4.7.1.4 Random Forest

Random Forest is an ensemble learning method that constructs multiple decision trees and aggregates their predictions. It is robust to noise and performs well on complex datasets with high variability, making it suitable for physiological signals like EMG and EEG.

4.7.2 Training and Testing

4.7.2.1 Training

The classifier is trained using a labeled dataset, where the correct class of each signal is known. The goal is for the classifier to learn the relationship between features and their corresponding labels.

4.7.2.2 Testing

The classifier's performance is evaluated on a separate test set that was not used during training. Performance metrics such as accuracy, sensitivity, specificity, and the area under the receiver operating characteristic (ROC) curve are commonly used.

4.8 Performance Evaluation

Evaluating the performance of classifiers in physiological signal analysis is essential for determining how accurately they can differentiate between various conditions like normal and abnormal heartbeats. This evaluation uses a combination of key metrics and methods, which include accuracy, sensitivity, specificity, cross-validation, and ROC curves.

4.8.1 Accuracy, Sensitivity, and Specificity

4.8.1.1 Accuracy

The ratio of correctly predicted cases (both positive and negative) to the total number of cases.

$$Accurracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **TP (True Positives)**: Correctly classified positive cases (e.g., correctly detecting abnormal heartbeats).
- **TN (True Negatives)**: Correctly classified negative cases (e.g., correctly detecting normal heartbeats).

- **FP (False Positives)**: Incorrectly classified cases as positive.
- **FN (False Negatives)**: Incorrectly classified cases as negative.

Provides an overall measure of the model's performance but may not be sufficient in imbalanced datasets, where normal signals far outnumber abnormal ones.

4.8.1.2 Sensitivity (Recall)

Measures the proportion of actual positives that are correctly identified (e.g., abnormal conditions).

$$Sensitivity\ (Recall) = \frac{TP}{TP + FN}$$

Critical in physiological analysis because missing positive cases (e.g., failing to detect a heart condition) can have severe consequences. High sensitivity ensures that abnormal conditions are detected, minimizing false negatives.

4.8.1.3 Specificity

Measures the proportion of actual negatives that are correctly identified (e.g., normal conditions).

$$Specificity = \frac{TN}{TN + FP}$$

Important to avoid false alarms in physiological analysis (e.g., diagnosing a healthy individual with a condition), which could lead to unnecessary treatments or interventions.

4.8.2 Cross-Validation

Cross-validation is a technique used to assess how well a classifier generalizes to new data by splitting the dataset into training and testing subsets.

4.8.2.1 k-Fold Cross-Validation

The data is divided into k subsets (folds). The classifier is trained on k-1 folds and tested on the remaining fold. This is repeated k times, and the results are averaged. It provides a more accurate estimate of performance, reducing biases from any particular data split. For smaller datasets, it maximizes the use of available data.

4.8.2.2 Leave-One-Out Cross-Validation (LOOCV)

A special case where the model is trained on all but one sample, and tested on the remaining one, repeated for each sample. It offers an unbiased performance estimate but can be computationally expensive for large datasets.

4.8.2.3 Stratified Cross-Validation

Ensures that each fold maintains the same class distribution as the original dataset, particularly important when working with imbalanced data. It ensures that both normal and abnormal cases are well-represented in each fold.

4.8.3 ROC Curves and AUC

4.8.3.1 ROC Curve (Receiver Operating Characteristic)

A plot of the true positive rate (sensitivity) against the false positive rate (1-specificity) across different classification thresholds. It shows the trade-off between sensitivity and specificity. The shape of the ROC curve helps visualize how well a classifier can distinguish between different classes (e.g., normal vs. abnormal signals).

4.8.3.2 AUC (Area Under the Curve)

A single scalar value that summarizes the ROC curve performance.

- AUC = 1.0: Perfect classification.
- AUC = 0.5: No better than random guessing.

Particularly useful in datasets where classes are imbalanced. A higher AUC indicates better discriminatory ability of the classifier, even when one class is more common than the other.

4.9 Application for a Physiological Signal: ECG Analysis

Feature extraction and classification methods are frequently applied in real-world scenarios, such as detecting arrhythmias in ECG signals.

4.9.1 Feature Extraction in ECG

4.9.1.1 R-peak detection

R-peaks are the highest points in the ECG waveform and correspond to individual heartbeats. Detecting these peaks is essential for analyzing heart rate and heart rate variability (HRV).

4.9.1.2 Heart Rate Variability (HRV)

HRV measures the time variation between consecutive R-peaks and is used to assess autonomic nervous system activity. Reduced HRV is associated with conditions like arrhythmia and heart failure.

4.9.1.3 Waveform morphology

The shape, amplitude, and duration of the QRS complex in the ECG signal are analyzed to diagnose heart conditions. For example, a prolonged QRS complex may indicate bundle branch block or ventricular hypertrophy.

4.9.2 Classifier Design for Arrhythmia Detection

Extracted ECG features, such as R-peak intervals and QRS morphology, are used to train classifiers (e.g., SVM or ANN) to distinguish between normal and abnormal heartbeats. The classifier's performance is assessed by its ability to detect arrhythmias, with metrics such as accuracy, precision, and recall being used to evaluate its effectiveness.

4.9.2.1 Real-Time Application

In clinical settings, real-time ECG monitoring continuously extracts features and applies trained classifiers to detect abnormal heart conditions, enabling timely medical interventions. For example, wearable devices equipped with ECG sensors can alert patients or healthcare providers when arrhythmias are detected, leading to early diagnosis and treatment.

4.10 Application of SVM for Auditory Evoked Potentials (AEP) Classification Using MATLAB

In this example, we developed an SVM classifier model using Radial Basis Function (RBF) kernel functions to distinguish between the normal hearing group and the hearing-impaired group, based on Fractal Dimension (FD) features extracted from Auditory Evoked Potentials (AEP) signals recorded from the participants. These signals, detected in the EEG auditory cortex area, are small electrical responses to sound stimuli, generated by the auditory pathway from the inner ear to the cerebral cortex. The AEP signals were recorded via electrodes attached to the scalp, measuring the bioelectric function of the auditory system.

Twenty participants were involved in the experiment, divided equally into a normal hearing group and a hearing-impaired group. The AEP signals were stimulated at four distinct frequencies in both the right and left ears at a fixed sound intensity level of 20 dBHL.

MATLAB Implementation

```
clear, clc, close all

load('C:\XX\svm_fractal_DFA_R500'); % Fractal Dimension values of all subjects (xdata) at a frequency of 500 Hz for the right ear

g1=zeros(50,1);
g2=ones(50,1);
group=[g1;g2];
P=0.3;

[TRAIN,TEST] = crossvalind('HoldOut',group,P);
TrainingSample=xdata(TRAIN,:);
TrainingLabel=group(TRAIN,1);
TestingSample=xdata(TEST,:);
TestingLabel=group(TEST,1);

numfolds=5;
Indices = crossvalind('Kfold', TrainingLabel, numfolds);

%% Training  SVM_RBF
        for i=1:numfolds
          TestingFoldSample=TrainingSample(Indices==i,:);
           TrainingFoldSample=TrainingSample(Indices~=i,:);
           TraingFoldLabeL=TrainingLabel(Indices~=i,:);
         Md = fitcsvm(TrainingFoldSample,TraingFoldLabeL,'KernelFunction', 'RBF');
           OutLabel_Train(Indices==i,1)=predict(Md,TestingFoldSample);
        end
           accfol=sum(grp2idx(OutLabel_Train)==grp2idx(TrainingLabel))/length(TrainingLabel);

%% Testing  Grid_SVM_RBF
         OutLabel_Test=predict(Md,TestingSample);
acc_test_svm_RBF=sum(grp2idx(OutLabel_Test)==grp2idx(TestingLabel))/length((TestingLabel))
```

The classification rates for the RBF kernel SVM classifier are provided in Table 4.1.

Table 4.1: Classification Performance of SVM Model at Different Frequencies for the Right and Left Ear

Fréquence (Hz)	Ear	Accuracy (%)	Sensitivity (%)	specificity (%)
500	**R**	96.67	93.33	100
1000	**R**	90.0	93.33	86.67
2000	**R**	83.33	86.67	80.0
4000	**R**	80.0	93.33	66.67
500	**L**	83.33	73.33	93.33
1000	**L**	86.67	73.33	100
2000	**L**	83.33	86.67	80.0
4000	**L**	90.0	86.67	93.33

The model achieves its best performance at 500 Hz for the right ear, with the highest accuracy (96.67%) and perfect specificity (100%). The 4000 Hz left ear also performs well, showing high accuracy (90.0%), sensitivity (86.67%), and specificity (93.33%). However, the right ear at 4000 Hz has the lowest specificity (66.67%), indicating difficulties in classifying normal hearing individuals, and the left ear exhibits lower sensitivity (73.33%) at 500 Hz and 1000 Hz, suggesting challenges in detecting hearing impairment at these frequencies. Overall, the right ear excels at lower frequencies, while the left ear performs better at higher frequencies.

4.11 Application of VGG19 and SVM for MRI Brain Tumor Classification Using MATLAB

In this example, we developed a hybrid approach by combining the VGG19 and SVM models to classify MRI images into two categories: tumor class and non-tumor class. The dataset used in this study consists of 2,000 MRI images of the human brain, equally divided between these two classes. VGG19, a pre-trained convolutional neural network (CNN) with 19 layers trained on a large dataset (e.g., ImageNet), was employed to extract deep features from the MRI images. These features were then used to construct a feature vector, which was fed into an SVM classifier. The SVM was trained using a linear kernel function to classify the input MRI images into normal (non-tumor) and abnormal (tumor) categories.

MATLAB Implementation

```
clear,clc,close all
net=vgg19;
myFolder=fullfile('C:\XX\'); % Path to the file containing Brain Tumor Dataset
categories={'no','yes'}; %Categories for Brain Tumor Classification

imds=imageDatastore(fullfile(myFolder,categories),'LabelSource','foldernames');
[imdsTrain,imdsValidation]=splitEachLabel(imds,0.7,'randomized');
imsize=net.Layers(1).InputSize;
normTrainingSet=augmentedImageDatastore(imsize,imdsTrain,"ColorPreprocessing","gray2rgb"
);
```

```
normTestingSet=augmentedImageDatastore(imsize,imdsValidation,"ColorPreprocessing","gray2
rgb");
fcFeature='fc8';
trainingFeatures=activations(net,normTrainingSet,fcFeature,'MiniBatchSize',32,'OutputAs','colu
mns' );
tesingValidationFeatures=activations(net,normTestingSet,fcFeature,'MiniBatchSize',32,'OutputA
s','columns' );
trainingLabels=imdsTrain.Labels;
testningLabels=imdsValidation.Labels;

t = templateSVM('KernelFunction','linear');
classifier_vgg=fitcecoc(trainingFeatures,trainingLabels,"Learners",t,ObservationsIn="columns")
;
[label_vgg_Test,Pred_vgg_core]=predict(classifier_vgg,tesingValidationFeatures,"observationsI
n","columns");
acc_vgg_Test=sum((label_vgg_Test==testningLabels)/numel(testningLabels))

cm_VGG = confusionchart(testningLabels,label_validation_vgg)
figure    %Confusion Matrix of the VGG19 Model for Binary Brain MRI Classification
rocObj1 = rocmetrics(testningLabels,Pred_vgg_core,categories);
plot(rocObj1,ClassNames=categories,ShowModelOperatingPoint=false); %ROC Curve for the
VGG19 Model
```

Figure 4.1 presents the confusion matrix for the VGG19 model used in the classification task.

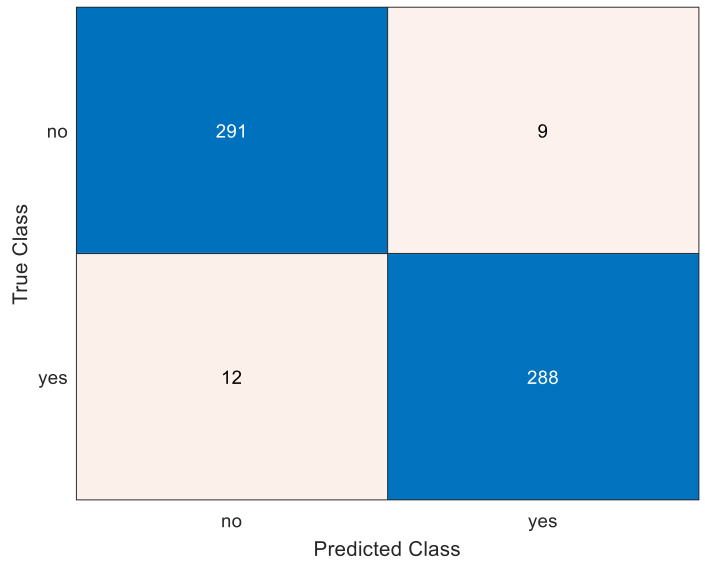


Figure 4.1: Confusion Matrix of the VGG19 Model for Binary Brain MRI Classification

The confusion matrix indicates that the classification model performs exceptionally well, with 291 true negatives (correctly predicted "no") and 288 true positives (correctly predicted "yes"). Misclassifications are minimal, with only 9 false positives (incorrectly predicted "yes") and

12 false negatives (incorrectly predicted "no"). Overall, the model achieves high accuracy (96.50%) and demonstrates strong precision and recall, effectively classifying both categories with few errors. The sensitivity of the model is 96%, indicating its ability to correctly identify positive cases, while its specificity is 97%, reflecting its effectiveness in identifying negative cases.

The performance of VGG19 model using the AUC metric, which measures its effectiveness in differentiating between classes, is shown in figure 4.2

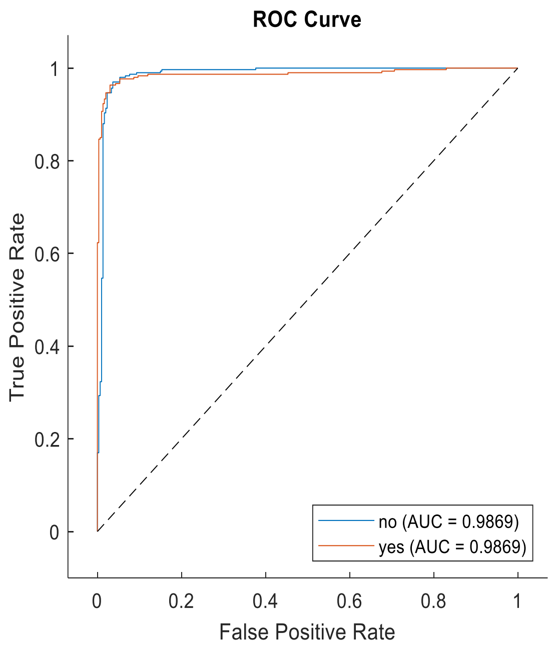


Figure 4.2: ROC Curve for the VGG19 Model

The figure displays two ROC curves for a binary classification task, one for the "no" class and the other for the "yes" class, both with an AUC of 0.9869. This high AUC indicates excellent model performance, signifying that the classifier effectively distinguishes between the two classes with a high true positive rate and a low false positive rate. Overall, the VGG19 model demonstrates remarkable sensitivity and specificity, accurately predicting both classes with minimal errors.

Chapter 5: Response of Random Signals to Linear Systems

5.1 Random signals

A random signal $X(t)$ is a function of random variables indexed by time t, with each $X(t)$ representing the value of the signal at a specific time. For each t, $X(t)$ can take on different values based on a probability distribution. The values of $X(t)$ at different time points can either be uncorrelated, as in the case of white noise, or correlated, as seen in autocorrelated signals, where the value at one time depends on previous values.

Examples of random signals are: speech, audio, ECG, EEG, economic series…

- Speech: Varies in pitch, amplitude, and content unpredictably.
- Audio: Includes music and environmental sounds with complex, often unpredictable patterns.
- ECG (Electrocardiogram): While generally periodic, it contains random variations in intervals and amplitudes.
- EEG (Electroencephalogram): Brain activity signals that are highly complex and non-stationary.

Statistical properties used to describe random signals are:

- The mean $\mu_X = E[X(t)]$, which is the expected value of the signal at time t.
- The variance $\sigma_X^2 = E[(X(t) - \mu_X)^2]$, which describes how much the signal varies around the mean at time t.
- The autocorrelation $R_X(t_1, t_2) = E[X(t_1)X(t_2)]$, measuring the relationship between the values at two different times t_1 and t_2.
- Stationarity, which means the signal's statistical properties (like mean and variance) do not change over time.

Random signals are important in areas such as signal processing, where they are used for noise reduction and data transmission, communications, where they model noise and interference, and control systems, where they represent random disturbances or uncertainties in inputs.

5.2 A Linear Time-Invariant (LTI) system

A Linear Time-Invariant (LTI) system, such as filters, is a system for which both the properties of linearity and time invariance hold. An LTI system can be characterized by its impulse response, denoted as $h(t)$. The impulse response $h(t)$ is the output of the system when an impulse (represented by the Dirac delta function $\delta(t)$ is applied to the input.

For any input signal $x(t)$, the output of the LTI system can be computed by convolving the input signal $x(t)$ with the system's impulse response $h(t)$. The convolution is given by the following integral:

$$y(t) = x(t) * h(t) = \int_{-\infty}^{+\infty} h(\tau)x(t-\tau)\,d\tau$$

In the frequency domain, the input-output relationship becomes much simpler using the system's transfer function $H(f)$:

$$Y(f) = H(f)X(f)$$

Where $X(f)$ and $Y(f)$ are the Fourier transforms of the input and output signals, and $H(f)$ is the transfer function of the system.

5.3 Memoryless Systems

A memoryless system is characterized by an operator L, whose action at time t depends only on the input signal at time t. Such a system does not take into account any previous or future states of the input signal. For a memoryless system, the response to an input signal $x(t)$ is given by:

$$y(t) = L[x(t)]$$

Where:

$y(t)$ is the output,

$x(t)$ is the input,

L represents the system function that relates input to output.

In a memoryless system, there is no integration or differentiation involved in the system's operation, making it instantaneous.

- A **simple linear amplifier** is a perfect example of a memoryless system. For an input $x(t)$, the amplifier gives an output $y(t)$ as:

$$y(t) = Ax(t)$$

Where A is a constant gain. This system has no memory since the output at any time t depends only on the input at time t.

- A **diode** that clips signals based on certain thresholds can also be modeled as a memoryless system but with a nonlinear response:

$$y(t) = \begin{cases} x(t) & if\ |x(t)| < V_{clip} \\ V_{clip} & if\ x(t) > V_{clip} \\ -V_{clip} & if\ x(t) < -V_{clip} \end{cases}$$

Here, V_{clip} is the clipping threshold.

5.4 Systems with Memory

A system is said to have memory when the present output depends on past input values, meaning the past affects the present. In contrast, a causal system is one where the output at any time depends only on current and past inputs, but never on future inputs. In real-world physical systems, the future cannot influence the present, which means these systems are inherently non-anticipatory. This principle of causality ensures that physical systems do not rely on future information for their present state, making them causal by nature.

5.5 Characteristics of Memoryless Systems

5.5.1 Instantaneous Response

The output is an immediate function of the input. There is no delay, and no historical data is stored.

5.5.2 Causality

A memoryless system can be causal (output depends on the present input) or non-causal (if it depends on future inputs), but for most practical physical systems, it's causal.

5.5.3 Linearity

Memoryless systems can be either linear or nonlinear depending on how the input $x(t)$ is transformed into the output.

5.6 Response of Memoryless Systems to Random Signals

When the input to a memoryless system is a random signal $x(t)$, the output $y(t)$ inherits the randomness of the input. However, the statistical properties of the output can still be determined if the input properties are known.

Example

Let's assume that the random input $x(t)$ has a Gaussian distribution with mean μ_x and variance σ_x^2. If the system is a linear memoryless system with a gain A, the output $y(t)$ will also have a Gaussian distribution:

- Mean of output: μ_y=A$\cdot\mu_x$.
- Variance of output: $\sigma_y^2 = A^2\sigma_x^2$.

For a nonlinear memoryless system, the statistical properties of the output are more complex and often require specialized techniques such as moment-generating functions to calculate.

5.7 Power Spectral Density (PSD)

5.7.1 Densities of Energy Spectral Density and Power Spectral Density

The Fourier transform of the autocorrelation function is called the power spectral density (for signals with finite average power) or the energy spectral density (for signals with finite energy). This allows us to analyze how the signal's power or energy is distributed across different frequencies.

The power spectral density (PSD) is given by the following expression:

$$S_x(f) = \int_{-\infty}^{+\infty} R_x(\tau) e^{-j2\pi f\tau} d\tau$$

Where:

$S_x(f)$ is the power spectral density at frequency f,

$R_x(\tau)$ is the autocorrelation function of the signal.

Unit: The unit of PSD is power per frequency unit (e.g., watts per hertz).

Total Power: The total power of the signal can be calculated by integrating the PSD across all frequencies:

$$P = \int_{-\infty}^{+\infty} S_x(f) df$$

This total power represents the average power of the signal.

5.7.2 Properties of Power Spectral Density

5.7.3 Non-negativity

$S_x(f)$ is always non-negative since it represents power.

5.7.4 Symmetry

For real-valued signals, $S_x(f)$ is symmetric around $f = 0$, meaning $S_x(f) = S_x(-f)$.

5.7.4.1 Wiener-Khinchin Theorem

This theorem establishes the relationship between the autocorrelation function and the power spectral density. It allows us to compute the PSD from the autocorrelation.

5.8 Methods to Estimate Power Spectral Density

Estimating the Power Spectral Density (PSD) of a continuous-time signal can be done using two main types of methods: Non-parametric Methods (Model-free) and Parametric Methods (Model-based).

The choice between these methods depends on the signal's characteristics and the data available. Each has advantages, and the best method for a given scenario depends on factors like computational efficiency, resolution needs, and the signal's nature. PSD estimation helps analyze the frequency content and power distribution of signals, which is critical in various signal processing applications.

5.8.1 Non-parametric Methods

These methods do not assume a specific model for the signal and rely directly on the data to estimate the PSD.

5.8.1.1 Periodogram

A widely used and straightforward technique is the periodogram, which estimates the power spectral density (PSD) by calculating the squared magnitude of the Fourier transform of the signal:

$$P_x(f) = \frac{1}{T}|X(f)|^2$$

where T is the observation period, and $X(f)$ is the Fourier transform of the signal. The periodogram is computationally efficient, but it can exhibit high variance, particularly for short observation periods.

Pros:

- Simple and easy to implement.
- Computationally efficient, especially with the Fast Fourier Transform (FFT).

Cons:

- High variance: The periodogram can give noisy PSD estimates, especially for short signals.
- Spectral leakage: If the signal is not periodic within the observation window, the energy leaks into other frequency bins, leading to inaccurate PSD estimates.

5.8.1.2 Welch's Method

Welch's method is a technique used to reduce the variance of the periodogram for estimating the power spectral density (PSD) of a signal. It works by dividing the signal into overlapping segments, applying a window function to each segment to reduce spectral leakage, and then computing and averaging the periodograms for each segment. This results in a smoother and more reliable estimate of the PSD compared to the basic periodogram method.

Pros:

- Lower variance compared to the simple periodogram.
- Better for stationary signals (those whose statistical properties don't change over time).

Cons:

- Reduced frequency resolution due to averaging.
- Can still suffer from spectral leakage, although windowing helps mitigate this.

5.8.1.3 Blackman-Tukey Method

This method uses the autocorrelation function of the signal, which is windowed, and then applies the Fourier transform to the autocorrelation function to estimate the PSD. It's based on the

Wiener-Khinchin theorem, which states that the Fourier transform of the autocorrelation function equals the PSD.

Pros

- flexibility in choosing the window function for the autocorrelation sequence.
- A smoother PSD estimate than the periodogram due to the averaging inherent in the autocorrelation.

Cons

- Tradeoff between resolution and variance: A larger window length improves frequency resolution but increases variance.
- More computationally intensive than the periodogram.

5.8.1.4 Multitaper Method

The multitaper method effectively reduces variance in power spectral density (PSD) estimation by employing multiple orthogonal tapers (or windows) to generate several independent PSD estimates, which are subsequently averaged. Specifically, this process involves applying multiple tapers, such as Slepian sequences, to the signal, estimating the PSD for each tapered version of the signal, and then averaging these estimates to yield a final PSD estimate. This approach is designed to minimize spectral leakage and reduce bias in the results.

Pros:

- Provides a good tradeoff between bias and variance.
- Effective in reducing spectral leakage and variance simultaneously.
- Better suited for signals with complex spectral characteristics.

Cons:

- More computationally expensive due to the use of multiple tapers.
- The choice of tapers and the number of tapers can affect the final estimate.

5.8.2 Parametric Methods

These techniques rely on the premise that the signal can be modeled using a specific structure, such as an autoregressive (AR), moving average (MA), or autoregressive-moving-average (ARMA) process. By estimating the parameters associated with these models, the power spectral density (PSD) can be calculated. This method takes advantage of the established structure of the signal, facilitating a more precise estimation of the PSD in comparison to non-parametric approaches.

5.8.2.1 Autoregressive (AR) Model-Based Methods

These methods assume that the signal can be modeled as an autoregressive (AR) process. The power spectral density (PSD) can then be computed by estimating the parameters of the AR model.

The PSD is given by:

$$S_x(f) = \frac{\sigma^2}{\left|1 - \sum_{k=1}^{p} a_k e^{-j2\pi f k}\right|^2}$$

Where a_k are the AR model coefficients, and σ^2 is the noise variance.

Pros:

- Provides good frequency resolution, especially for signals with sharp spectral peaks.
- Effective for signals with narrowband features.

Cons:

- Poor at modeling broadband signals.
- Requires careful selection of model order ppp, which can be challenging.

5.8.2.2 Moving Average (MA) Model

The MA model assumes the signal can be represented as the output of an all-zero filter driven by white noise. The signal depends on a weighted sum of current and past noise terms:

$$x(t) = \sum_{k=0}^{q} b_k \, \epsilon(t-k)$$

where b_k are the MA coefficients and $\epsilon(t)$ is white noise.

Pros:

- Well-suited for modeling signals with deep spectral nulls.
- Simpler for signals that exhibit rapid changes.

Cons:

- Less commonly used because it can be numerically unstable and doesn't work as well with sharp spectral peaks.
- Sensitive to noise.

5.8.2.3 Autoregressive Moving Average (ARMA) Model

The ARMA model combines both the AR and MA models, assuming the signal is the output of a filter that has both poles and zeros, driven by white noise:

$$x(t) = \sum_{k=1}^{p} a_k \, x(t-k) + \sum_{k=0}^{q} b_k \, \epsilon(t-k)$$

This provides greater flexibility in modeling a wide range of signals.

Pros:

- More flexible than either AR or MA models alone, capable of modeling both narrowband and broadband signals.

Cons:

- More complex parameter estimation due to the combined AR and MA components.
- Selecting both the AR and MA orders can be challenging.

5.8.2.4 Prony's Method

Prony's method fits a sum of complex exponentials to the signal by solving a set of linear equations. This technique is useful for modeling signals with sharp spectral peaks, such as damped sinusoids.

Pros:

- Very accurate for signals with sharp spectral peaks.
- Effective in applications like radar, sonar, and modal analysis.

Cons:

- Highly sensitive to noise, which can lead to instability in the model.
- Computationally intensive due to the matrix operations involved in solving the system of equations.

Each method comes with its own advantages and drawbacks. Non-parametric methods are generally simpler to implement but can experience high variance or lower resolution. In contrast, parametric methods provide improved resolution for certain types of signals but necessitate a precise model and careful selection of parameters. The selection of an appropriate method is influenced by the unique characteristics of the signal and the specific requirements of the analysis.

5.9 Applications of Power Spectral Density

- **Signal Processing**: PSD is widely used to analyze the frequency content of signals, especially in communications and audio processing.
- **Noise Analysis**: Engineers use PSD to understand the frequency content of noise in systems, which helps in designing filters and improving signal quality.

- **Biomedical Applications**: PSD is used in EEG analysis to understand the brain's activity and in ECG analysis for heart rate variability studies.

5.10 Response of LTI Systems to Random Signals

When the input to an LTI system is a random signal, the output is also a random signal. If the input is characterized by its PSD $S_x(f)$, the PSD of the output can be computed using the transfer function $H(f)$:

$$S_y(f) = |H(f)|^2 S_x(f)$$

This equation shows how the system shapes the frequency content of the input signal.

Example

Consider an LTI system characterized by the impulse response $h(t) = e^{-t}u(t),$ where $u(t)$ is the unit step function. A random signal $x(t)$ is defined as a white Gaussian noise process with zero mean and power spectral density $S_x(f) = \frac{N_0}{2}$, where N_0 is a constant.

a) Determine the output of the LTI system $x(t)$ in terms of the input $x(t)$.

b) Calculate the power spectral density $S_y(f)$ of the output signal $y(t)$.

c) Find the variance of the output signal $y(t)$.

Solution

a) The output $y(t)$ of an LTI system to an input $x(t)$ can be computed using the convolution of the input signal $x(t)$ with the system's impulse response $h(t)$:

$$y(t) = x(t) * h(t) = \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)\, d\tau$$

$$y(t) = \int_{-\infty}^{+\infty} x(\tau)e^{-(t-\tau)}u(t-\tau)\, d\tau$$

$$= \int_{-\infty}^{t} x(\tau)e^{-(t-\tau)}\, d\tau$$

b) The relationship between the input and output power spectral densities for an LTI system is given by:

$$S_y(f) = |H(f)|^2 S_x(f)$$

$$H(f) = \mathcal{F}[h(t)]$$

$$= \mathcal{F}[e^{-t}u(t)]$$

$$= \frac{1}{1 + j2\pi f}$$

$$|H(f)|^2 = \left|\frac{1}{1 + j2\pi f}\right|^2$$

$$= \frac{1}{1 + (2\pi f)^2}$$

$$S_y(f) = \frac{1}{1 + (2\pi f)^2} \cdot \frac{N_0}{2}$$

$$= \frac{N_0}{2(1 + (2\pi f)^2)}$$

c) The variance of the output signal $y(t)$ can be determined from its power spectral density. The variance σ_y^2 is given by:

$$\sigma_y^2 = \int_{-\infty}^{+\infty} S_y(f)\, df$$

$$\sigma_y^2 = \int_{-\infty}^{+\infty} \frac{N_0}{2(1 + (2\pi f)^2)} df$$

This integral can be evaluated using a standard result from integral calculus:

$$\int_{-\infty}^{+\infty} \frac{1}{x^2 + a^2} dx = \frac{\pi}{a}$$

$$\sigma_y^2 = \frac{N_0}{8\pi^2} \int_{-\infty}^{+\infty} \frac{1}{\left(\frac{1}{2\pi}\right)^2 + f^2} df$$

$$= \frac{N_0}{8\pi^2} \cdot \frac{\pi}{\frac{1}{2\pi}}$$

$$= \frac{N_0}{4}$$

Example2

An LTI system is characterized by the impulse response $h(t) = e^{-t}u(t),$ where $u(t)$ is the unit step function. The input signal $x(t)$ is a wide-sense stationary (WSS) random process with the following characteristics:

Mean: $E[x(t)] = 5$

Autocorrelation function: $R_x(\tau) = \sigma_x^2 e^{-\alpha|\tau|}$, where $\sigma_x^2 = 4$ and $\alpha = 2$

a) Find the mean $E[y(t)]$ of the output signal $y(t)$.
b) Determine the output autocorrelation function $R_y(\tau)$ of the system.
c) Calculate the output variance σ_y^2.

Solution

a) Mean of the Output Signal $E[y(t)]$

The mean of the output signal $y(t)$ is related to the mean of the input signal $x(t)$ through the impulse response of the system. The general expression for the mean of the output is given by:

$$E[y(t)] = \int_{-\infty}^{+\infty} h(\tau)E[x(t-\tau)]d\tau$$

Since $x(t)$ is a WSS process and its mean $E[x(t)] = 5$ is constant, the equation simplifies to:

$$E[y(t)] = E[x(t)]\int_{0}^{+\infty} e^{-\tau}d\tau$$

The integral evaluates as:

$$\int_{0}^{+\infty} e^{-\tau}d\tau = 1$$

Thus, the mean of the output signal is:

$$E[y(t)] = 5 \times 1 = 5$$

b) The output autocorrelation function $R_y(\tau)$

The output autocorrelation function $R_y(\tau)$ is related to the input autocorrelation function $R_x(\tau)$ through the convolution of the impulse response $h(t)$ with itself. The expression is given by:

The expression is given by:

$$R_y(\tau) = \iint\limits_{-\infty\,-\infty}^{+\infty\,+\infty} h(t_1)h(t_2)\, R_x(t_2 - t_1 + \tau)dt_1dt_2$$

Given that the impulse response is $h(t) = e^{-t}u(t)$, and the input autocorrelation function is $R_x(\tau) = \sigma_x^2 e^{-\alpha|\tau|}$, this becomes:

$$R_y(\tau) = 4 \iint\limits_{0\,0}^{+\infty\,+\infty} e^{-t_1}e^{-t_2}\, e^{-2|t_2-t_1+\tau|}dt_1dt_2$$

Breaking the expression into two cases for $t_2 - t_1 + \tau$ being positive or negative, the integration gives:

$$R_y(\tau) = \frac{4}{2(2+1)} e^{-3|\tau|}$$

$$= \frac{2}{3} e^{-3|\tau|}$$

c) Output Variance σ_y^2

The variance of the output signal is simply the value of the output autocorrelation function evaluated at $\tau = 0$:

$$\sigma_y^2 = R_y(0)$$

$$= \frac{2}{3} e^0$$

$$= \frac{2}{3}$$

5.11 Applications of Random Signal Response in LTI Systems

- **Communication Systems**: In wireless communication, the random noise added during transmission can be analyzed using the PSD, allowing for noise reduction techniques to be applied.
- **Control Systems**: For random disturbances acting on a system, the PSD helps in designing controllers that minimize the impact of these disturbances.

References

References

[1] J.P. MULLER, « Le filtrage numérique », Décembre 2000.

[2] S. Sarpal, «Algorithms, ASN Filter Designer, ASN Filter Designer Functionality Difference between IIR and FIR filters: a practical design guide», April 2020.

[3] A. Kourgli, « Analyse et Filtrage des signaux numeriques : Conception des Filtres Numériques RIF », 2016.

[4] M. BOUTAA, « Cours de Traitement avancé des signaux physiologiques ». 2022

[5] B. Perret, «Traitement et analyse d'images», 2017.

[6] M. Djemai, « Approche Hybride Basée sur les SVM et les AG en vue du Dépistage de la Surdité de Perception utilisant les Potentiels Evoqués Auditifs », PhD Thesis, July 2023.