

MUSTAPHA STAMBOULI UNIVERSITY OF MASCARA

FACULTY OF SCIENCE AND TECHNOLOGY



COURSE HANDBOOK

ADVANCED SIGNAL PROCESSING

This course is intended for first-year Master's students

Field: Telecommunications, Electronics.

Specializations: -Networks and Telecommunications.

-Electronics and Embedded System.

Prepared by:

Dr. Mokhtar Besseghier

Algeria 2025

Advanced Signal Processing Course

Target Audience:

Students: First-year Master's students in "Telecommunications"

- Specialization: - Networks and Telecommunications
- Electronics and Embedded System.

Learning this course requires prior mastery of the following fundamental concepts:

- **Basic Signal Processing:** Fourier transforms, convolution, filtering
- **Advanced Mathematics:** Linear algebra, complex analysis, statistics
- **Probability and Random Processes:** Random variables, stochastic processes
- **Fourier Analysis:** FT, DFT, FFT and their applications
- **Linear Systems:** Transfer function, impulse response

Learning Objectives:

This course aims to deepen knowledge in signal processing and introduce advanced techniques used in modern telecommunications systems.

1. Advanced Digital Filtering

- Master FIR and IIR filters
- Understand Z-transform and its applications
- Analyze linear time-invariant digital systems

2. Random Signals and Stochastic Processes

- Master statistical analysis of signals
- Understand stochastic processes and their characterization
- Analyze power spectral density

3. Parametric Spectral Analysis

- Deepen spectral estimation methods
- Master adaptive algorithms (LMS, RLS)
- Understand AR, MA, ARMA models

4. Time-Frequency and Time-Scale Analysis

- Master time-frequency transforms (STFT, wavelets)
- Understand multiresolution analysis
- Analyze non-stationary signals

Course Outline:

Chapter 1: Overview of Digital Filters (FIR and IIR)

Chapter 2: Random Signals and Stochastic Processes

Chapter 3: Parametric Spectral Analysis and Adaptive Digital Filtering

Chapter 4: Time-Frequency and Time-Scale Analysis

Learning Outcomes

Upon completion of this course, students will be able to:

Technical Skills

- **Design** and **implement** optimal FIR and IIR digital filters
- **Analyze** complex random signals and stochastic processes
- **Apply** parametric and non-parametric spectral estimation methods
- **Utilize** time-frequency transforms for non-stationary signal analysis

Practical Skills

- **Develop** adaptive algorithms for various applications
- **Select** appropriate analysis methods based on signal characteristics
- **Implement** MATLAB solutions for signal processing problems
- **Interpret** spectral and time-frequency analysis results

Transferable Skills

- **Solve** complex signal processing problems
- **Optimize** processing system performance
- **Evaluate** the quality and reliability of conducted analyses
- **Adapt** to new techniques and technologies in the field

Teaching Methods

- **Lectures:** Theoretical concepts and mathematical developments
- **Tutorials:** Applied exercises and problem solving
- **Laboratory Sessions:** MATLAB implementation and real signal analysis
- **Projects:** Complete applications on industrial case studies

Tools and Software

- **MATLAB/Simulink** for simulation and prototyping
- **Signal Processing Toolbox** for specialized functions
- **Wavelet Toolbox** for time-frequency analysis
- **Python** (NumPy, SciPy, PyWavelets) as open-source alternative

Key Features

- **Comprehensive Coverage:** From basic filtering to advanced time-frequency analysis
- **Hands-on Learning:** Extensive MATLAB implementations
- **Industry Relevance:** Real-world telecommunications applications
- **Research Preparation:** Advanced techniques for graduate research
- **Adaptive Learning:** Personalized support for different learning paces

Table of Contents :

Chapter 1: Overview of Digital Filters (FIR and IIR)

I.1. The Z-Transform	1
I.1.1. Definition	1
I.1.2. The Region of Convergence (ROC)	1
I.1.3. Z-Transform of Standard Discrete Signals	3
I.1.4. Properties of the Z-Transform	4
I.1.5. Representation of a Signal by Its Poles and Zeros	6
I.1.6. Inverse Z-Transform	7
I.2. Digital Systems	9
I.2.1. Definition	9
I.2.2. Time-Invariant Linear System (TILS)	10
I.2.3. Recursive Systems – Non-Recursive Systems	11
I.3. Finite Impulse Response Filters (FIR)	12
I.3.1. Properties	12
I.3.2. Origin of the Name FIR	12
I.3.3. Frequency Response	12
I.3.4. Designing an FIR Filter	13
I.3.5. Implementation of FIR Filters	15
I.4. Infinite Impulse Response Filter (IIR)	16
I.4.1. Properties	16
I.4.2. Origin of the Name IIR	16
I.4.3. Frequency Response	17
I.4.4. Designing an IIR Filter	17
I.4.5. Implementation of IIR Filters	20
I.5. Recap on Analog Filters	21
I.5.1. Butterworth Filters	21
I.5.2. Chebyshev Filters	23

Chapter 2: Random Signals and Stochastic Processes

II.1. Introduction to Random Signals	27
II.1.1. Random Variables	27
II.1.2. Probability Density Functions (PDFs)	27
II.1.3. Cumulative Distribution Function (CDF)	28

Table of Contents :

II.1.4. Moments and Statistical Characteristics	28
II.1.5. Common Probability Distributions	29
II.1.6. Joint Probability Distribution of Two RVs	31
II.2. Random Signals	32
II.2.1. Deterministic Signals	32
II.2.2. Random Signals	33
II.2.3. Statistical Moments	34
II.2.4. Stationarity	37
II.2.5. Temporal Moments	38
II.2.6. Ergodicity	39
II.2.7. Relationship Between Stationarity and Ergodicity	40
II.3. Power Spectral Density	41
II.4. Spectral Estimation (Periodogram and Correlogram)	42
II.5. Correlation Functions and Matrices	44
II.6. Remarkable Stochastic Processes	44
II.6.1. Gaussian Random Process	44
II.6.2. White Noise	46
II.6.3. Band-Limited White Noise	46
II.6.4. Markovian Stochastic Process	47

Chapter 3: Parametric Spectral Analysis and Adaptive Digital Filtering

III.1. Stochastic Models	50
III.1.1. Auto-Regressive Models (AR)	50
III.1.2. Moving Average Models (MA)	51
III.1.3. Auto-Regressive Moving Average Models (ARMA)	51
III.1.4. Complex Sinusoids in Noise Models	52
III.2. Spectrum Estimation Based on AR Models	53
III.2.1. Yule-Walker Method	53
III.2.2. Levinson-Durbin Algorithm	57

Table of Contents :

III.2.3. Pisarenko Harmonic Decomposition Method	59
III.2.4. Multiple Signal Classification (MUSIC)	61
III.3. Spectrum Estimation Based on ARMA Models	62
III.4. Adaptive Filtering	63
III.4.1. Least Mean Squares (LMS) Algorithm	63
III.4.2. Recursive Least Squares (RLS) Algorithm	66

Chapter 4: Time-Frequency and Time-Scale Analysis

IV.1. Limitations of Fourier Analysis	71
IV.1.1. The Problem with Classical Fourier Transform	71
IV.1.2. Key Limitations of Fourier Analysis	72
IV.2. Time-Frequency Duality and Uncertainty Principle	73
IV.2.1. The Fundamental Trade-off	73
IV.2.2. Mathematical Formulation	73
IV.2.3. Heisenberg-Gabor Uncertainty Principle	73
IV.3. Short-Time Fourier Transform (STFT) and Spectrogram	75
IV.3.1. Gabor Transform	75
IV.3.2. Generalized Short-Time Fourier Transform	78
IV.3.3. Discrete Short-Time Fourier Transform	80
IV.3.4. The Spectrogram	81
IV.3.5. Window Length Trade-offs	81
IV.4. Wavelet Transform	82
IV.4.1. Motivation and Advantages	82
IV.4.2. Continuous Wavelet Transform (CWT)	82

Table of Contents :

IV.4.3. Inverse Continuous Wavelet Transform	85
IV.4.4. Popular Wavelet Families	85
IV.4.5. Discrete Wavelet Transform (DWT)	87
IV.4.6. Multi-Resolution Analysis (MRA)	87
IV.4.7. Wavelet Properties and Selection	90
IV.5. STFT vs. Wavelet Transform Comprehensive Comparison	91
IV.5.1. Resolution Characteristics Analysis	91
IV.5.2. Practical Applications and Use Cases	91
IV.6. Wigner-Ville Transform	92
IV.6.1. Mathematical Foundation and Motivation	92
IV.6.2. Continuous Wigner-Ville Distribution	92
IV.6.3. Fundamental Properties	93
IV.6.4. Cross-Term Interference Problem	93
IV.6.5. Applications and Practical Considerations	94
IV.7. Time-Scale Analysis	95
IV.7.1. Motivation and Theoretical Foundation	95
IV.7.2. Scale vs. Frequency Relationship	95
IV.7.3. Multiscale Signal Decomposition	96

Appendix

MATLAB Examples and Code Listings

I.1. The Z-Transform

The Z-transform is indeed a crucial tool in signal processing and systems theory. It allows the conversion of discrete signals and difference equations in the time domain into algebraic equations in the Z-transform domain. In this way, it simplifies the analysis and design of discrete systems and facilitates the manipulation of difference equations, much like the Laplace transform does for continuous systems. The Z-transform plays a crucial role in various fields, including digital filtering, image processing, digital communication, and many others. It enables efficient analysis and design of discrete systems.

I.1.1. Definition

The Z-transform, $X(Z)$, of the sequence $x(n)$ is a complex function defined by the following relationship:

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

Notation:

$$x(n) \xrightarrow{Z} X(Z)$$

where:

$$X(Z) = Z\{x(n)\}$$

This transformation in Z is called the bilateral Z-transform, as opposed to the unilateral Z-transform, which is defined by:

$$X_l(Z) = \sum_{n=0}^{+\infty} x(n) Z^{-n}$$

The variable Z is a complex number and can be expressed in polar form as:

$$Z = r e^{j\omega}$$

I.1.2. The Region of Convergence (ROC)

For any arbitrary signal $x(n)$, $X(Z)$ may not exist for certain values of Z. The set of values of the complex variable Z for which $X(Z)$ exists is called the Region of Convergence (ROC).

To find the ROC of $X(Z)$, the Cauchy criterion is used:

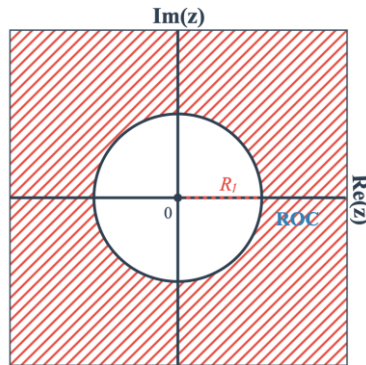
$$\sum_{n=0}^{+\infty} U_n \text{ converges if } \lim_{n \rightarrow \infty} |U_n|^{1/n} < 1$$

This criterion is applied to the Z-transform to determine the ROC.

$$\begin{aligned} X(Z) &= \sum_{n=-\infty}^{+\infty} x(n) Z^{-n} \\ &= \sum_{n=0}^{+\infty} x(n) Z^{-n} + \sum_{n=-\infty}^{-1} x(n) Z^{-n} \\ &= X_1(Z) + X_2(Z) \end{aligned}$$

$X(Z)$ converges if both $X_1(Z)$ and $X_2(Z)$ converge.

- For the convergence of $X_1(Z)$: $X_1(Z)$ converges if $\lim_{n \rightarrow \infty} |x(n)Z^{-n}|^{1/n} < 1$
 where: $\lim_{n \rightarrow \infty} |x(n)|^{1/n} |Z|^{-1} < 1 \Rightarrow |Z| > \lim_{n \rightarrow \infty} |x(n)|^{1/n} = R_1$
 So: $X_1(Z)$ converges if $|Z| > R_1$.

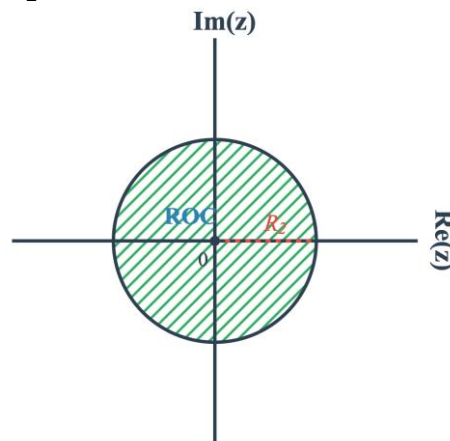


- For the convergence of $X_2(Z)$:

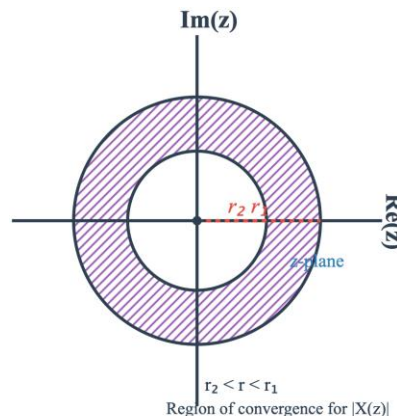
$$X_2(Z) = \sum_{n=-\infty}^{-1} x(n)Z^{-n} = \sum_{n=1}^{+\infty} x(-n)Z^n = \sum_{n=0}^{+\infty} x(-n)Z^n - x(0)$$

$X_2(Z)$ converges if $\lim_{n \rightarrow \infty} |x(-n)Z^n|^{1/n} < 1 \Rightarrow |Z| < \frac{1}{\lim_{n \rightarrow \infty} |x(n)|^{1/n}} = R_2$

where $X_2(Z)$ converges if $|Z| < R_2$.



The ROC of $X(Z) = X_1(Z) + X_2(Z)$ is the intersection of the two regions of convergence of $X_1(Z)$ and $X_2(Z)$, meaning:



Example I.1: Let's consider the sequence:

$$x(n) = a^n u(n) \quad a \text{ is real}$$

The Z-transform of $x(n)$ is:

$$X(Z) = \sum_{n=-\infty}^{+\infty} a^n u(n) Z^{-n} = \sum_{n=0}^{+\infty} (aZ^{-1})^n$$

$X(Z)$ converges if $\lim_{n \rightarrow \infty} |(aZ^{-1})^n|^{1/n} < 1 \Rightarrow |aZ^{-1}| < 1 \Rightarrow |Z| > |a|$

In this case :

$$X(Z) = \frac{1}{1 - aZ^{-1}} = \frac{Z}{Z - a}$$

Example I.2 : Let's consider the sequence:

$$x(n) = -a^n u(-n - 1) \quad a \text{ is real}$$

The Z-transform of $x(n)$ is:

$$\begin{aligned} X(Z) &= \sum_{n=-\infty}^{+\infty} -a^n u(-n - 1) Z^{-n} = - \sum_{n=-\infty}^{-1} a^n Z^{-n} \\ &= - \sum_{n=1}^{+\infty} (a^{-1} Z)^n = 1 - \sum_{n=0}^{+\infty} (a^{-1} Z)^n \end{aligned}$$

$X(Z)$ converges if $\lim_{n \rightarrow \infty} |(a^{-1} Z)^n|^{1/n} < 1 \Rightarrow |a^{-1} Z| < 1 \Rightarrow |Z| < |a|$

In this case :

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

1.1.3. Z-Transform of Standard Discrete Signals

To define the Z-transforms of standard discrete signals, we have:

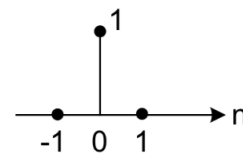
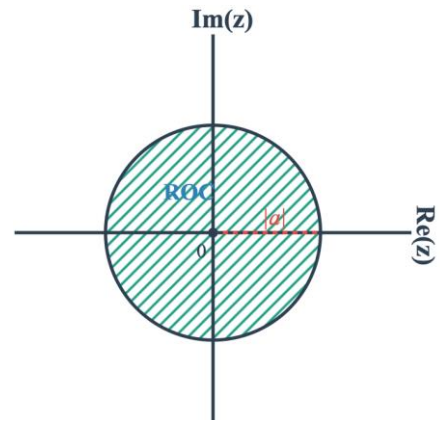
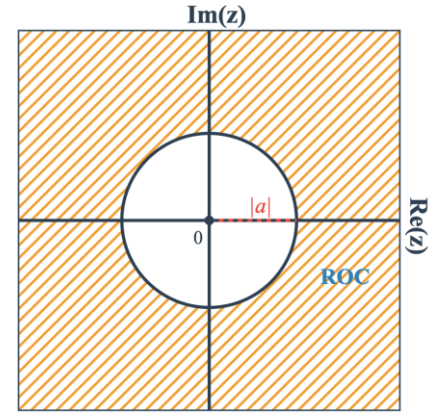
$$\begin{aligned} \sum_{n=0}^{N-1} a^n &= \frac{1 - a^N}{1 - a} \quad \text{ou } a \neq 1 \\ \sum_{n=0}^{+\infty} a^n &= \frac{1}{1 - a} \quad \text{ou } |a| < 1 \end{aligned}$$

- Unit Impulse (Dirac):**

The Z-transform of a unit impulse signal, often denoted as $\delta(n)$, is 1 for all values of Z within the entire complex plane. This means that the ROC for the Z-transform of a unit impulse signal is the entire complex plane. In mathematical terms:

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

$$X(Z) = \sum_{n=-\infty}^{+\infty} \delta(n) Z^{-n} = Z^{-0} = 1$$



• **Unit Step:**

The Z-transform of a unit step signal, often denoted as $u(n)$, is given by: $Z\{u(n)\} = 1 / (1 - Z^{-1})$, ROC: $|Z| > 1$ (outside the unit circle in the Z-plane). In mathematical terms:

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

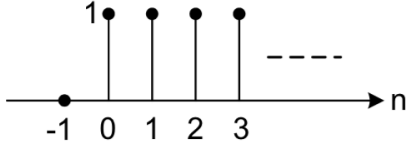
$$X(Z) = \sum_{n=-\infty}^{+\infty} u(n) Z^{-n} = \sum_{n=0}^{+\infty} Z^{-n} = \frac{Z}{Z-1}$$


Table I.1: Some common Z-transforms

$x[n]$	$X(z)$	ROC
$\delta[n]$	1	All z
$u[n]$	$\frac{1}{1 - z^{-1}}, \frac{z}{z - 1}$	$ z > 1$
$-u[-n - 1]$	$\frac{1}{1 - z^{-1}}, \frac{z}{z - 1}$	$ z < 1$
$\delta[n - m]$	z^{-m}	All z except 0 if $(m > 0)$ or ∞ if $(m < 0)$
$a^n u[n]$	$\frac{1}{1 - az^{-1}}, \frac{z}{z - a}$	$ z > a $
$-a^n u[-n - 1]$	$\frac{1}{1 - az^{-1}}, \frac{z}{z - a}$	$ z < a $
$na^n u[n]$	$\frac{az^{-1}}{(1 - az^{-1})^2}, \frac{az}{(z - a)^2}$	$ z > a $
$-na^n u[-n - 1]$	$\frac{az^{-1}}{(1 - az^{-1})^2}, \frac{az}{(z - a)^2}$	$ z < a $
$(n + 1)a^n u[n]$	$\frac{1}{(1 - az^{-1})^2} \cdot \left[\frac{z}{z - a} \right]^2$	$ z > a $
$\sin(an)u(n)$	$\frac{z \sin(a)}{z^2 - 2z \cos(a) + 1}$	$ z > 1$
$\cos(an)u(n)$	$\frac{z[z - \cos(a)]}{z^2 - 2z \cos(a) + 1}$	$ z > 1$
$a^n \sin(bn)u(n)$	$\frac{[a \sin(b)]z}{z^2 - [2a \cos(b)]z + a^2}$	$ z > a $
$a^n \cos(bn)u(n)$	$\frac{z[z - a \cos(b)]}{z^2 - [2a \cos(b)]z + a^2}$	$ z > a $
$\begin{cases} a^n & 0 \leq n \leq N - 1 \\ 0 & \text{otherwise} \end{cases}$	$\frac{1 - a^N z^{-N}}{1 - az^{-1}}$	$ z > 0$

I.1.4. Properties of the Z-Transform:

The basic properties of the Z-transform are stated as follows:

- **Linearity:**

If the input is of the form $ax(n) + by(n) \Rightarrow$ then the output will be $aX(Z) + bY(Z)$.

Example I.3: Consider the following sequence:

$$x(n) = 3u(n) + 4\delta(n), n = 1, 2, 3, \dots$$

$$X(Z) = Z\{3u(n) + 4\delta(n)\} = 3Z\{u(n)\} + 4Z\{\delta(n)\}$$

$$X(Z) = 3 \frac{Z}{Z-1} + 4 = \frac{7Z-4}{Z-1}$$

- **Time Delay (Advance):**

If the input is of the form $x(n-k) \Rightarrow$ then the output will be $Z\{x(n-k)\} = Z^{-k}X(Z)$.

Example I.4: Consider the following sequence:

$$x(n) = 3u(n-2)$$

$$X(Z) = Z\{3u(n-2)\} = 3Z^{-2}Z\{u(n)\}$$

$$X(Z) = 3Z^{-2} \frac{Z}{Z-1} = \frac{3}{Z(Z-1)}$$

- **Multiplication by an Exponential:**

If the input is of the form $a^{-n}x(n) \Rightarrow$ then the output will be $Z\{a^{-n}x(n)\} = X(aZ)$.

$$\sum_{n=-\infty}^{+\infty} a^{-n}x(n)Z^{-n} = \sum_{n=-\infty}^{+\infty} x(n)(aZ)^{-n} = X(aZ)$$

Example I.5: Consider the following sequence:

$$x(n) = e^{-\alpha nT}u(n), n = 0, 1, 2, \dots$$

$$X(Z) = \sum_{n=-\infty}^{+\infty} e^{-\alpha nT}u(n)Z^{-n} = \sum_{n=0}^{+\infty} e^{-\alpha nT}Z^{-n} = \sum_{n=0}^{+\infty} (e^{\alpha T}Z)^{-n}$$

$$X(Z) = \frac{e^{\alpha T}Z}{e^{\alpha T}Z - 1} = \frac{Z}{Z - e^{-\alpha T}}$$

- **Multiplication by n:**

If the input is of the form $nx(n) \Rightarrow$ then the output will be $Z\{nx(n)\} = -Z \frac{dX(Z)}{dZ}$.

Example I.6: Consider the following sequence:

$$x(n) = nu(n), n = 0, 1, 2, \dots$$

$$X(Z) = -Z \frac{d}{dZ} \left(\frac{Z}{Z-1} \right) = \frac{Z}{(Z-1)^2}$$

- **Time Reversal:**

If the input is of the form $x(-n) \Rightarrow$ then the output will be $Z\{x(-n)\} = X(1/Z)$.

Example I.7: Consider the following sequence:

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

$$X(Z) = \sum_{n=-\infty}^{+\infty} u(-n)Z^{-n} = \sum_{n=-\infty}^0 Z^{-n} = \sum_{n=0}^{+\infty} Z^n = \frac{1}{1-Z}$$

- **Convolution:**

If the input is of the form $x(n) * y(n) \Rightarrow$ then the output will be $Z\{x(n) * y(n)\} = X(Z).Y(Z)$

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

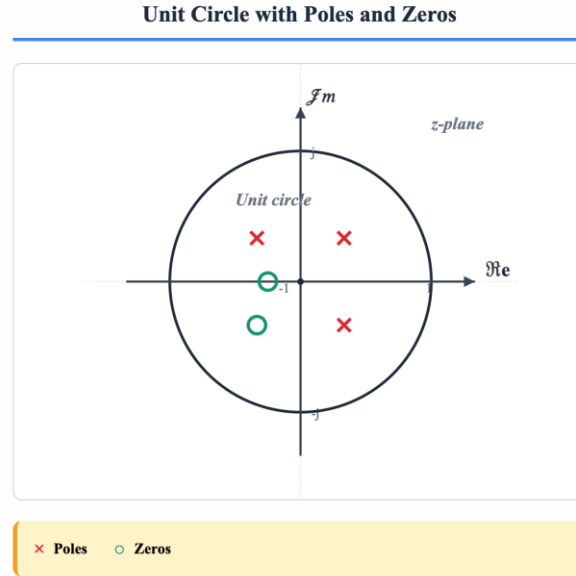
$$Z\{x(n) * y(n)\} = \sum_{k=-\infty}^{+\infty} x(k)Z^{-k}Y(Z) = X(Z).Y(Z)$$

Table I.2: Properties of the z-Transform

Property	Sequence	Transform	Roc
	$x[n]$	$X(z)$	R
	$x_1[n]$	$X_1(z)$	R_1
	$x_2[n]$	$X_2(z)$	R_2
Linearity	$a_1x_1[n] + a_2x_2[n]$	$a_1X_1(z) + a_2X_2(z)$	$R' \supseteq R_1 \cap R_2$
Time shifting	$x[n - n_0]$	$z^{n_0} X(z)$	$R' \supseteq R \cap \{0 < z < \infty\}$
Multiplication by z_0^n	$z_0^n x[n]$	$X(z/z_0)$	$R' = z_0 R$
Multiplication by $e^{j\Omega_0 n}$	$e^{j\Omega_0 n} x[n]$	$X(e^{-j\Omega_0} z)$	$R' = R$
Time reversal	$x[-n]$	$X(1/z)$	$R' = 1/R$
Multiplication by n	$nx[n]$	$-z \frac{dX(z)}{dz}$	$R' = R$
Accumulation	$\sum_{k=-\infty}^n x[k]$	$\frac{1}{1-z^{-1}} X(z)$	$R' \supseteq R \cap \{ z > 1\}$
Convolution	$x_1[n] * x_2[n]$	$X_1(z).X_2(z)$	$R' \supseteq R_1 \cap R_2$

I.1.5. Representation of a Signal by Its Poles and Zeros:

The Z-transform provides, from the poles and zeros of $X(Z)$. This representation is particularly useful for studying the stability of digital filters. In this representation, the locations of the poles and zeros in the Z-plane provide insights into the behavior of the signal and the filter associated with it. The distribution of poles and zeros can reveal information about stability, frequency response, and other important characteristics of the signal processing system. This representation simplifies the analysis and design of digital filters and signal processing systems. Where Poles are marked by "x," and zeros by "o", as shown below.



Example: Consider the following sequence:

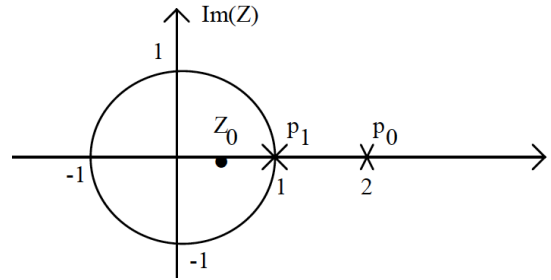
$$x(n) = 2^{n+2}u(n) - u(n)$$

$$X(Z) = Z\{2^{n+2}u(n)\} - Z\{u(n)\}$$

$$X(Z) = \frac{4Z}{Z-2} - \frac{Z}{Z-1} = \frac{3Z^2 - 2Z}{(Z-1)(Z-2)}$$

So, $X(Z)$ contains two zeros $Z_0 = 0$,

$Z_1 = \frac{2}{3}$, and two poles, $p_0 = 2$, $p_1 = 1$



I.1.6. Inverse Z-transform:

The expression for the inverse Z-transform is obtained using Cauchy's theorem, which is defined by the following integral:

$$\frac{1}{2\pi j} \oint Z^{k-1} dZ = \delta(k)$$

By choosing a contour C within the region of convergence of $X(z)$ and applying Cauchy's theorem, we can express the inverse Z-transform:

$$x(k) = \frac{1}{2\pi j} \oint_C X(Z) Z^{k-1} dZ$$

Calculating this integral can be challenging. However, there are several methods to compute $x(k)$, such as partial fraction decomposition, inverse Laplace transform, and more....

- **Fractional decomposition:** This method is the most commonly used and allows us to break down $X(Z)$ into a sum of terms for which the inverse is known. We express $X(Z)$ in the following form:

$$X(Z) = \frac{\mathcal{N}(Z)}{\mathcal{D}(Z)} = k \frac{(Z - Z_0)(Z - Z_1) \dots (Z - Z_{M-1})}{(Z - p_0)(Z - p_1) \dots (Z - p_{N-1})}$$

Assuming $N \geq M$, and all the poles p_n are simple (meaning they have a multiplicity of 1), then:

$$\frac{X(Z)}{Z} = \frac{c_0}{Z} + \frac{c_1}{Z - p_0} + \frac{c_2}{Z - p_1} + \cdots + \frac{c_N}{Z - p_{N-1}} = \frac{c_0}{Z} + \sum_{n=0}^{N-1} \frac{c_{n+1}}{Z - p_n}$$

where:

$$c_0 = X(Z)|_{Z=0} \quad c_{n+1} = (Z - p_n) \frac{X(Z)}{Z} \Big|_{Z=p_n}$$

Therefore, we obtain:

$$X(Z) = c_0 + c_1 \frac{Z}{Z - p_0} + c_2 \frac{Z}{Z - p_1} + \cdots + c_N \frac{Z}{Z - p_{N-1}} = c_0 + \sum_{n=0}^{N-1} c_{n+1} \frac{Z}{Z - p_n}$$

Example 1.8: Consider the following sequence:

$$X(Z) = \frac{Z + 10}{Z^2 + 3Z + 2}$$

$$X(Z) = \frac{Z + 10}{(Z + 1)(Z + 2)}$$

$$X(Z) = c_0 + c_1 \frac{Z}{Z + 1} + c_2 \frac{Z}{Z + 2}$$

$$c_0 = \frac{10}{2} = 5$$

$$c_1 = (Z + 1) \frac{X(Z)}{Z} \Big|_{Z=-1} = (Z + 1) \frac{Z + 10}{Z(Z + 1)(Z + 2)} \Big|_{Z=-1}$$

$$c_1 = \frac{Z + 10}{Z(Z + 2)} \Big|_{Z=-1} = -9$$

$$c_2 = (Z + 2) \frac{X(Z)}{Z} \Big|_{Z=-2} = (Z + 2) \frac{Z + 10}{Z(Z + 1)(Z + 2)} \Big|_{Z=-2}$$

$$c_2 = \frac{Z + 10}{Z(Z + 1)} \Big|_{Z=-2} = 4$$

So :

$$X(Z) = 5 - \frac{9Z}{Z + 1} + \frac{4Z}{Z + 2}$$

By using the table: $x(n) = 5\delta(n) - 9(-1)^n u(n) + (-2)^{n+2} u(n)$

Case for multiple roots: For a function $X(Z)$ with multiple roots:

$$X(Z) = \frac{\mathcal{N}(Z)}{(Z - p_0)^r \prod_{n=r}^{N-1} (Z - p_n)} = \sum_{n=0}^{r-1} b_n \frac{Z}{(Z - p_0)^{r-n}} + \sum_{n=r}^{N-1} c_n \frac{Z}{(Z - p_n)}$$

The coefficients for multiple roots are calculated by:

$$b_n = \frac{1}{n!} \frac{d^n}{dz^n} (Z - p_0)^r X(Z) \Big|_{Z=p_0} \quad n = 0, 1, 2, \dots, r-1$$

Example I.9: Consider the following sequence:

$$X(Z) = \frac{Z}{(Z-1)(Z-0.5)^2}$$

$$X(Z) = b_0 \frac{Z}{(Z-0.5)^2} + b_1 \frac{Z}{(Z-0.5)} + c_2 \frac{Z}{(Z-1)}$$

$$c_2 = (Z-1) \frac{X(Z)}{Z} \Big|_{Z=1} = \frac{1}{(Z-0.5)^2} \Big|_{Z=1} = 4$$

$$b_0 = \frac{1}{0!} \frac{d^0}{dz^0} (Z-0.5)^2 X(Z) \Big|_{Z=0.5} = \frac{Z}{(Z-1)} \Big|_{Z=0.5} = -1$$

$$b_1 = \frac{1}{1!} \frac{d^1}{dz^1} (Z-0.5)^2 X(Z) \Big|_{Z=0.5} = -\frac{1}{(Z-1)^2} \Big|_{Z=0.5} = -4$$

$$X(Z) = \frac{4Z}{(Z-1)} - \frac{4Z}{(Z-0.5)} - \frac{Z}{(Z-0.5)^2}$$

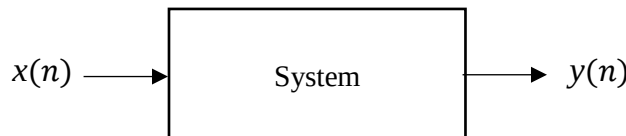
By using the table: $x(n) = 4u(n) - 4(0.5)^n u(n) - 2n(0.5)^n u(n)$

I.2. Digital Systems:

I.2.1. Definition

A digital system or filter, denoted as $\mathcal{P}$, represents the mathematical modeling of a physical process that relates an input signal $x(n)$ to an output signal $y(n)$.

$$y(n) = \mathcal{P}\{x(n)\}$$



- A system is **linear** if the principle of superposition applies. This means that:

$$\mathcal{P}\{\alpha x_1(n) + \beta x_2(n)\} = \alpha \mathcal{P}\{x_1(n)\} + \beta \mathcal{P}\{x_2(n)\}$$

Here are examples of two systems you mentioned:

$\Rightarrow$ *Linear Amplifier with Gain K*: The system can be represented as follows: $\mathcal{P}\{x(n)\} = Kx(n)$

This system is linear because it follows the principle of superposition.

⇒ *Quadratic Operator*: The system can be represented as follows: $\mathcal{P}\{x(n)\} = x^2(n)$

This system is non-linear because it does not follow the principle of superposition.

- A system is considered **time-invariant** if, for any integer time shift n_0 applied to the input, the output is also delayed by n_0 , as expressed by the equation:

$$\mathcal{P}\{x(n - n_0)\} = y(n - n_0)$$

Example I.10: Suppose $y(n) = \cos(x(n))$. Determine if the system is time-varying or time-invariant. If we apply a time shift of n_0 to the input signal $x(n)$ to create $x(n - n_0)$, and the output becomes $y(n - n_0)$, which is a time shift of n_0 to the output signal $y(n)$, this demonstrates that the system is time-invariant.

- The combination of a time-invariant system and a linear system results in a *Time-Invariant Linear System (TILS)*, which is also referred to as a *Time-Invariant System (TIS)*. So:
Time-Invariant System + Linear System = Time-Invariant Linear System (TILS).

1.2.2. Time-Invariant Linear System (TILS)

Considering a linear time-invariant system characterized by the function $h(n)$. The relationship between the input $x(n)$ and the output $y(n)$ is defined as:

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

Where $h(n)$ is known as the **impulse response** and represents the system's response to the unit impulse, defined as:

$$y_{\delta}(n) = h(n) * \delta(n) = \sum_{k=-\infty}^{+\infty} h(k)\delta(n - k) = h(n)$$

⇒ A system is said to be **causal** if:

$$h(n) = 0 \text{ pour } n < 0$$

This means that the system's response at any given time n is dependent only on the current and past values of the input and does not depend on future values.

⇒ A system is said to be **stable** if:

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

This condition ensures that the sum of the absolute values of the impulse response, taken over all possible time instants from negative infinity to positive infinity, remains finite. Stability is

important because it guarantees that the system's output will not grow unbounded in response to a bounded input.

Example I.11: Let $h(n)$ be the impulse response of a digital filter:

Where $h(n) = a^n u(n)$. Determine if the system is stable or not for $a = 1/2$ and $a = 2$.

$$\sum_{n=-\infty}^{+\infty} |h(n)| = \sum_{n=0}^{+\infty} \left(\frac{1}{2}\right)^n = \frac{1}{1 - \frac{1}{2}} = 2 < \infty$$

So, for $a = 1/2$, the filter is stable.

$$\sum_{n=-\infty}^{+\infty} |h(n)| = \sum_{n=0}^{+\infty} 2^n \text{ diverge}$$

So, for $a = 2$, the filter is unstable.

Example I.12: Analyzing the Stability and Causality of the Following System:

$$H(Z) = \frac{Z}{Z - \frac{1}{2}}$$

Here the stability and causality properties depend on the region of convergence (ROC) of the Z-transform.

1. ROC: $|Z| > 1/2$ (System is stable and causal):

In this case, where the ROC is outside the circle of radius $1/2$, the system is indeed stable. The impulse response is $h(n) = (1/2)^n u(n)$, which indicates causality.

2. ROC: $|Z| < 1/2$ (System is non-stable and anti-causal):

When the ROC is inside the circle of radius $1/2$, the system is considered non-stable. The impulse response is $h(n) = -(1/2)^n u(-n - 1)$, which signifies anti-causality.

I.2.3. Recursive Systems - Non-Recursive Systems:

A linear time-invariant system is a system whose behavior over time can be described by a difference equation of the form:

$$\begin{aligned} y(n) + a_1 y(n-1) + a_2 y(n-2) + \dots + a_M y(n-M) &= b_0 x(n) + b_1 x(n-1) + \dots + b_N x(n-N) \end{aligned}$$

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

After applying the Z-transform, we obtain:

$$\left(1 + \sum_{n=1}^{M-1} a_n Z^{-n}\right) Y(Z) = X(Z) \sum_{n=0}^{N-1} b_n Z^{-n}$$

$$H(Z) = \frac{Y(Z)}{X(Z)} = \frac{\sum_{n=0}^{N-1} b_n Z^{-n}}{1 + \sum_{n=1}^{M-1} a_n Z^{-n}}$$

Where $H(Z)$ is the transfer function of the system or filter. The constants b_n and a_n are the filter coefficients, and the filter's order is the maximum of N and M .

⇒ When $N = 1$, the filter is a Finite Impulse Response (FIR) filter (Non-Recursive).

⇒ Otherwise, the filter is an Infinite Impulse Response (IIR) filter (Recursive).

I.3. Finite Impulse Response Filters (FIR)

FIR filters are characterized by having a finite-duration response to an impulse input. In other words, the output of an FIR filter depends only on a finite number of past input samples. They are digital filters defined by a polynomial transfer function in Z^{-1} :

$$H(z) = \sum_{n=0}^{N-1} h(n)Z^{-n}$$

I.3.1. Properties:

⇒ Default stability (no denominator) makes them always stable.

⇒ Linear phase response for realizable filters.

⇒ May require excessive computation time, leading to potential computational complexity.

I.3.2. Origin of the Name FIR (Finite Impulse Response):

Its difference equation is defined as:

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

If we replace the input $x(n)$ with an impulse $\delta(n)$, the output becomes: $y(n) = h(n)$:

$$h(n) = \sum_{i=0}^N b_i \delta(n-i) = b_0 \delta(n) + b_1 \delta(n-1) + b_2 \delta(n-2) \dots \dots + b_{N-1} \delta(n-M+1)$$

Therefore:

$$h(n) = \begin{cases} b_n & 0 \leq n \leq N-1 \\ 0 & \text{Otherwise} \end{cases}$$

This definition of $h(n)$ with finite non-zero values for b_n within the range 0 to $N-1$ gives rise to the name "Finite Impulse Response (FIR)."

I.3.3. Frequency Response:

The Z-transform of $h(n)$ is given by:

$$H(Z) = \sum_{n=0}^{N-1} h(n)Z^{-n}$$

The frequency response of the filter is obtained by replacing Z by $e^{j\omega}$:

$$H(Z)|_{Z=e^{j\omega}} = \sum_{n=0}^{N-1} h(n)e^{-j\omega n}$$

I.3.4. Designing an FIR Filter:

Designing a Finite Impulse Response (FIR) filter involves calculating the coefficients b_n of the filter's polynomials. The challenge in FIR filter design is to find the N coefficients of $h(n)$, where $n = 0, 1, \dots, N - 1$, based on a specified desired frequency response, $H_d(f)$. This frequency response is typically described using a dB template:

- ⇒ Passband (PB) from 0 to f_p
- ⇒ Stopband (or cutoff band, SB) from f_a to $f_e/2$
- ⇒ Transition zone width $\Delta f = |f_a - f_p|$
- ⇒ Cutoff frequency $f_c = \frac{f_a + f_p}{2}$
- ⇒ Passband ripple amplitude :
- ⇒ $R_p = -20 \log\left(\frac{1-\delta_1}{1+\delta_1}\right) > 0 (\approx 0)$
- ⇒ Stopband ripple amplitude :
- ⇒ $A_s = -20 \log\left(\frac{\delta_2}{1+\delta_1}\right) > 0 (\gg 1)$

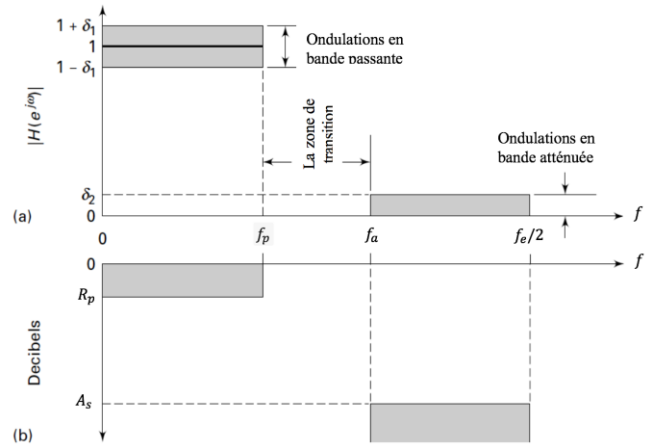


Figure I.1: The Gabarit of a low-pass filter.

For the design of an FIR filter, there are three main methods:

1. The Windowing Method:

- ⇒ In the windowing method, you start with an ideal frequency response and then use a window function to truncate it, resulting in a finite-length impulse response.
- ⇒ This method is relatively simple and widely used for designing low-pass and band-pass FIR filters.
- ⇒ Popular window functions include the Hamming, Hanning, and Kaiser windows.

2. The Frequency Sampling Method:

- ⇒ The frequency sampling method directly determines the filter coefficients by specifying the desired frequency response at discrete frequency points.
- ⇒ It's especially useful when you have precise control over the filter's frequency characteristics.
- ⇒ This method is commonly used for designing FIR filters with specific frequency responses.

3. Optimization Methods:

- ⇒ Optimization methods involve mathematical optimization techniques to find the filter coefficients that best meet the desired specifications while minimizing some error criterion.
- ⇒ Examples of optimization methods include least squares, linear programming, and gradient descent algorithms.
- ⇒ These methods are powerful when you need to meet complex design criteria, but they can be computationally intensive.

Each of these methods has its advantages and is chosen based on the specific requirements and constraints of the filter design task. In this course, we will take an example of the Windowing Method.

- **Windowing Method:**

In this method, we begin by specifying $H_d(f)$ to determine the corresponding desired impulse response $h_d(n)$.

$$h_d(n) = \frac{1}{f_e} \int_{-f_e/2}^{f_e/2} H(f) e^{j2\pi f n T_e} df$$

For an ideal low-pass digital filter with a cutoff frequency of $f_c = \frac{f_a + f_p}{2}$

$$\begin{aligned} h_d(n) &= \frac{1}{f_e} \int_{-f_c}^{f_c} e^{j2\pi f n T_e} df = \frac{1}{j2\pi n f_e T_e} e^{j2\pi f n T_e} \Big|_{-f_c}^{f_c} = \frac{1}{j2\pi n} (e^{j2\pi f_c n T_e} - e^{-j2\pi f_c n T_e}) \\ &= \frac{\sin(2\pi f_c n T_e)}{\pi n} = 2T_e f_c \frac{\sin(2\pi f_c n T_e)}{2\pi f_c n T_e} = 2 \frac{f_c}{f_e} \text{sinc}\left(2\pi \frac{f_c}{f_e} n\right) \end{aligned}$$

This filter is not implementable because its impulse response is infinite and non-causal. If we limit the range of n to N , it is equivalent to multiplying $h_d(n)$ by a rectangular window of the same width, defined as:

$$W_{\text{Rect}}(n) = \begin{cases} 1 & \text{for } |n| \leq \frac{N-1}{2} \\ 0 & \text{Otherwise} \end{cases}$$

So:

$$h(n) = h_d(n) \cdot W(n)$$

This truncation (multiplication of $h_d(n)$ by the rectangular window of the same width, equivalent to convolving $H_d(f)$ with the Fourier transform of the rectangular window) introduces ripples in both the passband and stopband. This effect can be minimized by using non-rectangular windows (see Lab n°1). The following table presents some properties of the windows:

Window	Function for $ n \leq \frac{N-1}{2}$	$As_{\min}$	$\Delta f / f_e$
Rectangular	$w(t) = \text{rect}(t/T)$	21 dB	$0.9/N$
Hanning	$w(t) = 0.5[1 + \cos(2\pi t/T)]$	44 dB	$3.1/N$
Hamming	$w(t) = 0.54 + 0.46\cos(2\pi t/T)$	53 dB	$3.3/N$
Blackman	$w(t) = 0.42 + 0.5\cos(2\pi t/T) + 0.08\cos(4\pi t/T)$	74 dB	$5.5/N$

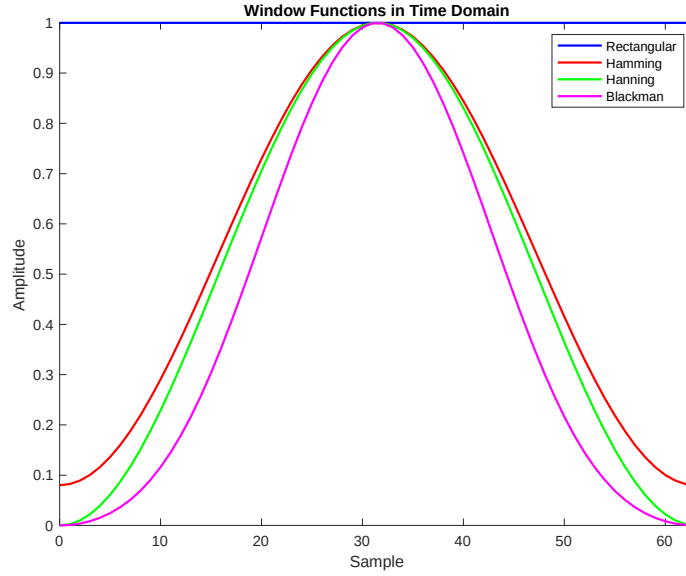


Figure I.2: Windows Functions.

Example I.13: We want to determine the digital low-pass FIR filter $h_d(n)$ using the windowing method based on the following specifications:

$$\begin{aligned} 0.99 \leq |H(\omega)| \leq 1.01 & \quad 0 \leq |\omega| \leq 1.19\pi \\ |H(\omega)| \leq 0.01 & \quad 0.21\pi \leq |\omega| \leq \pi \end{aligned}$$

The attenuation in the stopband is $Aa = -20\log(0.01) = 40 \text{ dB}$. This suggests the use of a Hanning window.

$$\Delta\omega = \omega_a - \omega_p = 0.21\pi - 0.19\pi = 0.02\pi$$

For a Hanning window (refer to the table) : $\frac{\Delta f}{f_e} = \frac{3.1}{N} \Rightarrow \frac{\Delta\omega}{2\pi} = \frac{3.1}{N} \Rightarrow N = \frac{3.1 \cdot 2\pi}{\Delta\omega} = 310$

The cutoff frequency $\omega_c = \frac{\omega_a + \omega_p}{2} = 0.2\pi$.

$$h_d(n) = 0.2 \text{ sinc}(0.2\pi n)$$

I.3.5. Implementation of FIR Filters

Once the filter coefficients, $h(n)$, are calculated, the next step is to realize this function using basic digital circuits (adders, flip-flops, multipliers, etc.). FIR filters are well-suited for applications where linear-phase characteristics and stability are critical, such as audio processing, equalization, and channel equalization in communication systems.

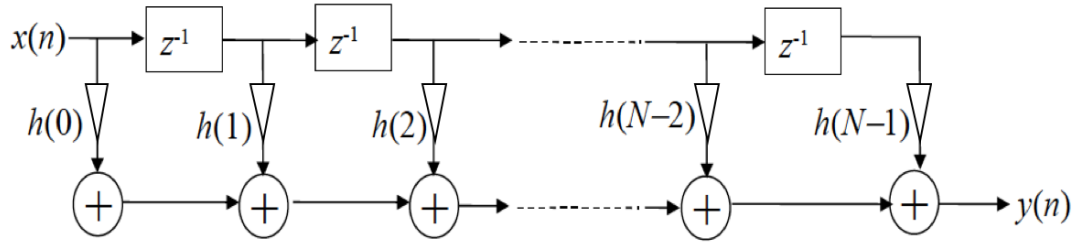


Figure I.3: Canonical form of a FIR filter.

I.4. Infinite Impulse Response Filter (IIR)

Infinite Impulse Response (IIR) filters have an indefinite response to an impulse input, relying on both past and future input samples. They are characterized by a rational polynomial transfer function in Z^{-1} and often incorporate feedback, making them recursive in nature :

$$H(Z) = \frac{\sum_{n=0}^{N-1} b_n Z^{-n}}{1 + \sum_{n=1}^{M-1} a_n Z^{-n}}$$

Introducing poles in $H(Z)$ significantly reduces the number of coefficients compared to a Finite Impulse Response (FIR) filter.

I.4.1. Properties:

- ⇒ Very efficient in terms of computation time, but...
- ⇒ Generally, provide a nonlinear phase response.
- ⇒ Their stability must be verified during the design process.

I.4.2. Origin of the name IIR (Infinite Impulse Response):

Its difference equation is defined as:

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

If we replace the input $x(n]$ with an impulse $\delta(n]$, the output is:

$$\begin{aligned} y(n) = h(n) &= \sum_{i=0}^N b_i \delta(n-i) - \sum_{i=1}^M b_i y(n-i) \\ &= b_0 \delta(n) + b_1 \delta(n-1) + b_2 \delta(n-2) \dots \dots + b_{M-1} \delta(n-N+1) - a_1 y(n-1) \\ &\quad - a_2 y(n-2) \dots \dots - a_{N-1} y(n-M+1) \end{aligned}$$

Therefore:

$$\begin{aligned}
 h(0) &= b_0 \\
 h(1) &= b_1 - a_1 y(0) \\
 h(2) &= b_2 - a_1 y(1) - a_2 y(0) \\
 &\vdots \\
 h(n) &= \begin{cases} b_n - \sum_{i=1}^{N-1} a_i y(n-i) & \text{if } n \leq N-1 \\ -\sum_{i=1}^{N-1} a_i y(n-i) & \text{Otherwise} \end{cases}
 \end{aligned}$$

Unlike FIR filters, the response is not simply equal to the successive values of its coefficients, and it is of infinite duration.

I.4.3. Frequency Response:

The Z-transform of $h(n)$ is given by:

$$H(Z) = \frac{\sum_{n=0}^{N-1} b_n Z^{-n}}{1 + \sum_{n=1}^{M-1} a_n Z^{-n}}$$

The frequency response of the filter is obtained by replacing Z by $e^{j\omega}$:

$$H(Z)|_{Z=e^{j\omega}} = \frac{\sum_{n=0}^{N-1} b_n e^{-j\omega n}}{1 + \sum_{n=1}^{M-1} a_n e^{-j\omega n}}$$

I.4.4. Designing an FIR Filter:

Designing an Infinite Impulse Response (IIR) filter involves determining the coefficients of the filter's numerator b_n and denominator a_n polynomials. Unlike Finite Impulse Response (FIR) filters, IIR filters have feedback loops, which introduce poles and zeros in the frequency response.

The synthesis of an Infinite Impulse Response (IIR) filter involves several steps to design a filter with the desired frequency response and characteristics. Here are the general steps in synthesizing an IIR filter:

1. **Specification of Filter Requirements:** Determine the filter's desired frequency response characteristics, such as cutoff frequency, passband ripple, stopband attenuation, and filter order. These specifications define the filter's behavior and performance.
2. **Choose a Design Method:** Select one of the three main design methods for IIR filters:
 - *Impulse Invariance Method:*
 - ⇒ Starts with an analog filter design.
 - ⇒ Samples the continuous-time impulse response of the analog filter.
 - ⇒ Uses the sampled impulse response as coefficients for the digital IIR filter.
 - ⇒ Directly maps analog filter characteristics to a digital filter.
 - *Poles and Zeros Method:*

- ⇒ Designs the filter based on the locations of its poles and zeros in the complex plane.
 - ⇒ Specifies desired filter characteristics by placing poles (affecting frequency response) and zeros (affecting frequency and phase response).
 - ⇒ Offers precise control over the filter's response characteristics.
 - **Bilinear Transformation Method:**
 - ⇒ Maps the analog filter's transfer function from the s-domain (continuous-time) to the z-domain (discrete-time).
 - ⇒ Utilizes a bilinear mapping to preserve lower-frequency response while warping higher-frequency response.
 - ⇒ Ensures filter stability but may introduce distortion, especially at higher frequencies.
3. **Design the Analog Prototype:** If you choose the Impulse Invariance or Bilinear Transformation methods, design an analog prototype filter (analog filter) that satisfies the specified requirements. This prototype will later be transformed into a digital filter.
 4. **Analog-to-Digital Transformation:** Apply the chosen method to convert the analog prototype filter's transfer function into a digital transfer function suitable for digital signal processing. This step typically involves mapping analog poles and zeros into the digital Z-plane.
 5. **Optimize the Design:** Adjust the filter parameters, such as pole locations or zero locations, to fine-tune the digital filter's performance and meet the desired specifications. Iterative optimization may be required.
 6. **Implement the Digital Filter:** Implement the designed digital filter using practical numerical methods or available software tools. This step involves obtaining the filter coefficients, which define the filter's behavior.
 7. **Evaluate and Test:** Simulate the digital filter's response to different input signals and verify that it meets the specified requirements. Adjust the design if necessary.
 8. **Realization:** Implement the digital filter in hardware or software for real-world applications, such as audio processing, communication systems, or control systems.

The designing methods offer various approaches for designing IIR filters, each with its own advantages and trade-offs depending on the specific design requirements and constraints. In this course, we will take an example of the Bilinear Transformation Method.

- **Bilinear Transformation Method:**

This method involves finding $H(s)$ for an equivalent analog filter and then converting it into $H(Z)$ using the bilinear transformation technique. The process can be summarized as follows:

$$H(Z) = H(s) \Big|_{s=k \frac{Z-1}{Z+1}}$$

Where: $k = 2f_s$. The bilinear transformation introduces a frequency warping effect, which is described by the following equations:

$$f_D = \frac{f_s}{\pi} \arctan\left(\frac{\pi f_A}{f_s}\right)$$

$$f_A = \frac{f_s}{\pi} \tan\left(\frac{\pi f_D}{f_s}\right)$$

Here, f_A represents the analog frequency, f_D is the digital frequency, and f_s stands for the sampling frequency. This transformation results in a distortion of frequency values.

The synthesis of an IIR filter using the Bilinear Transformation Method involves the following steps:

1. Define the desired characteristics of the digital filter (sampling frequency, cutoff frequency, etc.).
2. Calculate the analog frequencies f_A corresponding to the digital frequencies.
3. Determine the specifications of the analog filter and the normalized transfer function $H_N(s)$ of order N (e.g., Butterworth, Chebyshev, etc.).
4. Apply the prototype low-pass filter transformation to obtain high-pass, band-pass, or band-stop filters as needed.
5. Perform the bilinear transformation on the filter obtained in step 4.
6. Calculate the transfer function based on the coefficients obtained.
7. Visualize the impulse response and implement the filter as necessary.

Example 1.14: We want to design a first-order digital Chebyshev high-pass filter with a cutoff frequency of 3 kHz and a ripple of 1 dB. The passband uses a sampling frequency of 8,000 Hz.

1. Calculated the corresponding analog frequency:

$$f_{c_A} = \frac{f_s}{\pi} \tan\left(\frac{\pi f_{c_D}}{f_s}\right) = \frac{8}{\pi} \tan\left(\frac{3\pi}{8}\right) = 6.1477 \text{ kHz}$$

2. Designed the normalized analog transfer function $H_N(s)$ for a first-order Chebyshev filter $N = 1$ with $r = 1 \text{ dB}$:

$$H_N(s) = \frac{1.9652}{s + 1.9652}$$

3. Applied the transformation from prototype (low-pass) to high-pass:

$$H(s) = \frac{1.9652}{\frac{\omega_c}{s} + 1.9652} = \frac{1.9652 s}{3.8627 \times 10^4 + 1.9652 s}$$

4. Applied the bilinear transformation to obtain the digital transfer function $H(Z)$:

$$H(Z) = \frac{1.9652 s}{3.8627 \times 10^4 + 1.9652 s} \bigg|_{s=16000 \frac{Z-1}{Z+1}} = \frac{1.9652.16000 \frac{Z-1}{Z+1}}{3.8627 \times 10^4 + 1.9652.16000 \frac{Z-1}{Z+1}}$$

$$H(Z) = \frac{3.144 \frac{Z-1}{Z+1}}{3.8627 + 3.144 \frac{Z-1}{Z+1}} = \frac{3.144(Z-1)}{3.8627(Z+1) + 3.144(Z-1)} = \frac{3.144Z - 3.144}{7Z - 0.7187}$$

5. Simplified the digital transfer function:

$$H(Z) = \frac{0.449 - 0.449 Z^{-1}}{1 + 0.1027 Z^{-1}}$$

6. the coefficients of the difference equation for the digital filter:

⇒ Numerator coefficients (b_0, b_1): [0.449, -0.449]

⇒ Denominator coefficients (a_0, a_1): [1, 0.1027]

I.4.5. The implementation of IIR Filters

The realization of IIR (Infinite Impulse Response) filters involves the use of basic digital circuits like adders, flip-flops, multipliers, and logic elements to perform the operations required by the filter's transfer function. These circuits are configured to implement the difference equations associated with the transfer function, allowing them to generate the desired filter response. IIR filters, characterized by their recursive nature and feedback loops, often demand more complex logic components for implementation when compared to FIR filters. However, they are valuable for specific filtering tasks due to their capability to offer selective frequency responses.

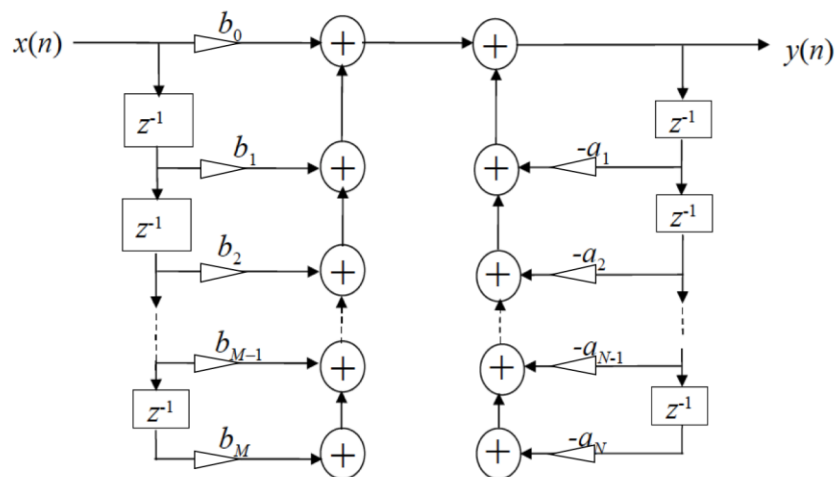


Figure I.4: Canonical form of a IIR filter.

I.5. Recap on Analog Filters

I.5.1. Butterworth Filters:

The Butterworth is a class of filters that provides a maximally flat response in the passband. The pole locations for an N th-order Butterworth filter are equally spaced around a circle with radius equal to the filter cutoff frequency.

Key Properties:

- **Maximally flat passband:** No ripple in the frequency range of interest
- **Monotonic response:** Smooth rolloff without oscillations
- **Symmetric pole distribution:** Poles equally spaced on a circle
- **Predictable phase response:** Relatively linear phase characteristics

Mathematical Foundation

Frequency Response

The magnitude response of an $N - th$ order Butterworth low-pass filter is mathematically expressed as:

$$|H(\omega)| = \frac{1}{\sqrt{1 + \left(\frac{\omega}{\omega_c}\right)^{2N}}}$$

Where:

- $|H(\omega)|$: magnitude response at frequency ω
- ω_c : -3 dB cutoff frequency (rad/s)
- N : filter order (positive integer)
- ω : angular frequency (rad/s)

Critical Frequency Points:

- At $\omega = 0$: $|H(0)| = 1$ (0 dB) - DC gain
- At $\omega = \omega_c$: $|H(\omega_c)| = 1/\sqrt{2} \approx 0.707$ (-3 dB) - cutoff point
- At $\omega \rightarrow \infty$: $|H(\omega)| \rightarrow 0$ - infinite attenuation

Pole Distribution and Transfer Function

The poles of a normalized Butterworth filter ($\omega_c = 1$ rad/s) are located at:

$$p_k = e^{j(2k+N-1)\pi/(2N)} \quad \text{for } k = 0, 1, 2, \dots, N-1$$

These poles are **equally distributed** around a circle of radius ω_c in the s-plane, with the constraint that all poles must lie in the left half-plane for stability.

Example I.15: 4th-order Butterworth Filter Poles

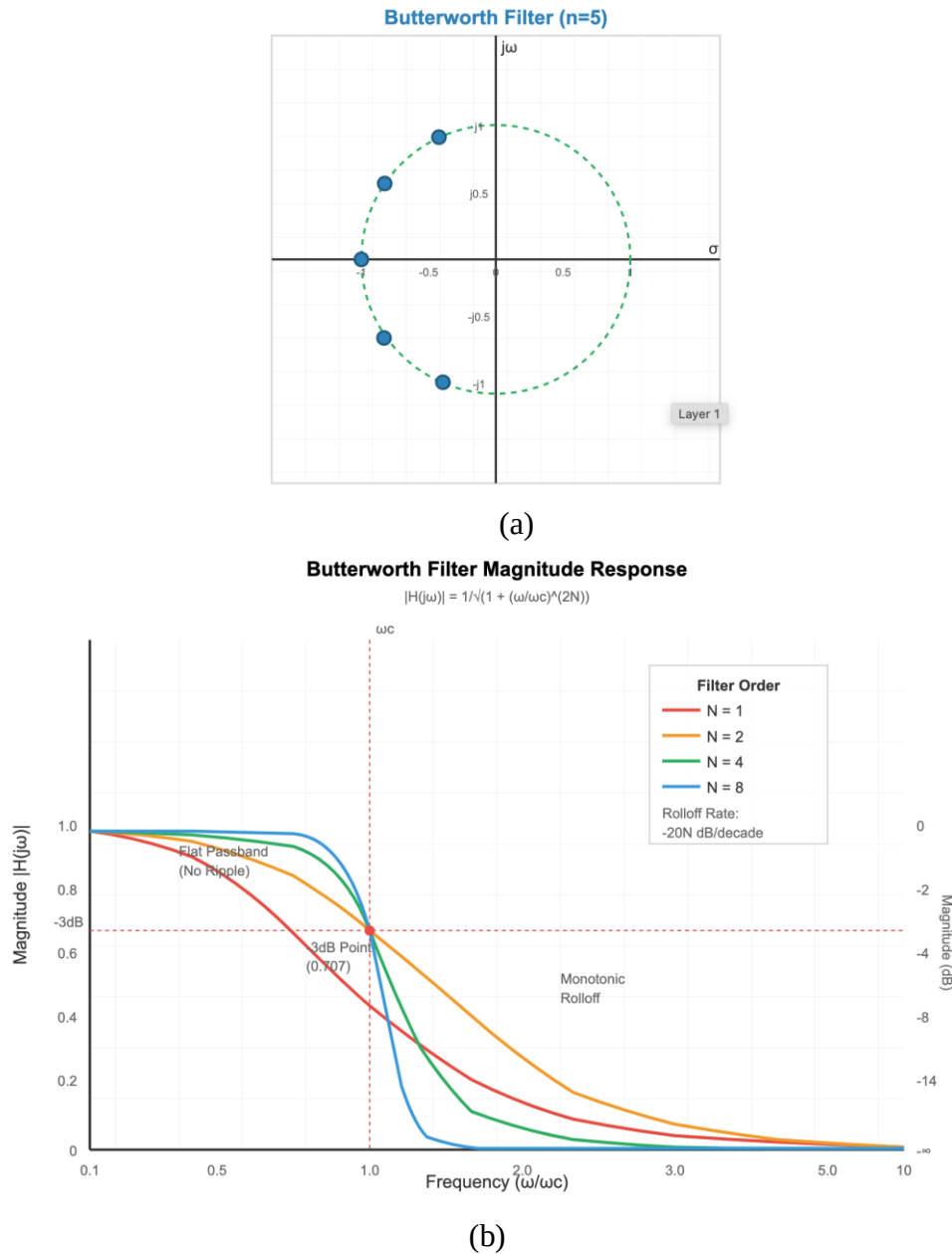


Figure I.5: 4th-order Butterworth Filter Poles.

Butterworth filter. (a) Pole locations for an $N = 5$ Butterworth low-pass filter with a cutoff frequency equal to the radius of the circle and poles spaced equally about the circle. (b) Magnitude response of a Butterworth low-pass analog filter of order N .

Filter Order Determination

The order N is determined by the transition band (ω_a, ω_p) and the allowable ripples in the passband (R_p) and stopband (A_s):

$$N \geq \frac{\log\left(\left(10^{\frac{A_s}{10}} - 1\right) / \left(10^{\frac{R_p}{10}} - 1\right)\right)}{2 \log(\omega_s / \omega_p)}$$

Where:

- ω_p passband edge frequency
- ω_s stopband edge frequency
- R_p maximum passband ripple (dB)
- A_s minimum stopband attenuation (dB)

The normalized transfer function $H_N(s)$ can be expressed in terms of its poles:

$$H_N(s) = \frac{1}{P(s)}$$

Ordre N	$P(s)$
1	$1 + s$
2	$1 + \sqrt{2}s + s^2$
3	$(1 + s)(1 + s + s^2)$
4	$(1 + 0.7653s + s^2)(1 + 1.8477s + s^2)$

Example I.16: Determining Butterworth filter order

Let us determine the filter order N needed to achieve a Butterworth filter with a cutoff frequency $f_c = 1$ MHz and 50-dB attenuation at 2.7 MHz.

First, note that an attenuation of -50 dB is equivalent to a gain magnitude of $10^{-50/20} = 3.16 \times 10^{-3}$. We can use this in the attenuation equation as follows:

$$3.16 \times 10^{-3} = \frac{1}{\sqrt{1 + (2.7)^{2N}}} \rightarrow N = 5.8$$

So, to meet the magnitude specification, we need an $N = 6$ Butterworth filter.

I.5.2. Tchebychev Filters:

We can improve the stopband attenuation significantly by allowing some finite ripple in the passband. The Chebyshev type-1 filter has some ripple in the passband, and a sharper cutoff in the transition band than does the Butterworth filter. As we will see, the price one pays for using the Chebyshev design compared to using the Butterworth is more group delay variation and more overshoot in the transient response.

Advantages:

- ✓ *Sharper transition band* - faster rolloff between passband and stopband
- ✓ *Better stopband attenuation* for the same filter order
- ✓ *Smaller transition bandwidth* for given specifications
- ✓ *Lower implementation cost* (fewer components for same performance)

Disadvantages:

- ✓ *Passband ripple* - non-uniform gain in the desired frequency range
- ✓ *Increased group delay variation* - potential signal distortion
- ✓ *Greater overshoot* in step response
- ✓ *More complex design* - additional ripple parameter to consider

Mathematical Formulation

Frequency Response

The magnitude response of an $N - th$ order Chebyshev Type-I filter is given by:

$$|H(\omega)| = \frac{1}{\sqrt{1 + \varepsilon^2 T_N^2\left(\frac{\omega}{\omega_c}\right)}}$$

Where:

- ε : ripple factor (determines passband ripple amplitude)
- $T_N(x)$: N -th order Chebyshev polynomial of the first kind
- ω_c : cutoff frequency (at passband edge)
- $T_N(x)$ is the Chebyshev polynomial of order N , defined as follows:

$$\begin{cases} T_0(x) = 1 \\ T_1(x) = x \\ \vdots \\ T_{N+1}(x) = 2xT_N(x) - T_{N-1}(x) \end{cases}$$

Or in closed form:

$$T_N(x) = \begin{cases} \cos(N \arccos x) & \text{pour } |x| \leq 1 \quad (\text{passband}) \\ \cosh(N \operatorname{arccosh} x) & \text{pour } |x| > 1 \quad (\text{stopband}) \end{cases}$$

Ripple Parameter Relationships

The relationship between passband ripple (r in dB) and the ripple factor ε is:

$$\varepsilon = \sqrt{10^{\frac{r}{10}} - 1}$$

Pole Distribution

Unlike Butterworth filters, Chebyshev poles are distributed on an **ellipse** in the s -plane:

Real part: $\sigma_k = -\omega_c \cdot \sinh(\beta) \cdot \sin((2k+1)\pi/(2N))$

Imaginary part: $\omega_k = \omega_c \cdot \cosh(\beta) \cdot \cos((2k+1)\pi/(2N))$

where: $\beta = (1/N) \cdot \sinh^{-1}(1/\varepsilon)$

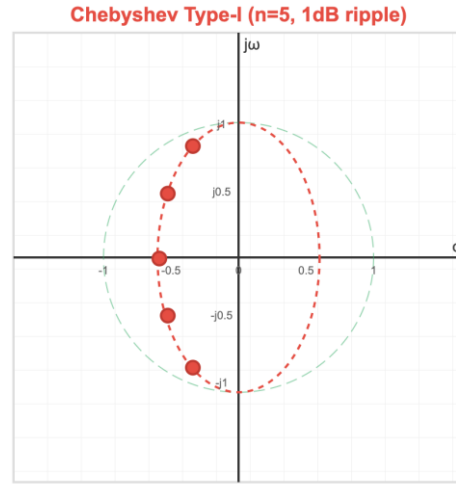


Figure I.6: 5th-order Chebyshev Filter Poles.

we see the pole map of an $N = 5$ Chebyshev filter with a cutoff frequency $\omega_c = 1$ rad/s and 0.25-dB passband ripple. The poles are all inside the unit circle on an ellipse, closer to the $j\omega$ axis than Butterworth filter poles.

The order N of this filter is determined by the transition zone (ω_a, ω_p) and the allowed ripples in the passband (A_p) and stopband (A_s):

$$N \geq \frac{\operatorname{arccosh}\left(\sqrt{10^{\frac{A_s}{10}}} - 1 / \sqrt{10^{\frac{R_p}{10}}} - 1\right)}{\operatorname{arccosh}(\omega_s / \omega_p)}$$

Here, Chebyshev type-1 low-pass filter magnitude, order $N = 4$, cutoff frequency $\omega_c = 1$ rad/s, and passband ripple 0.25 dB.

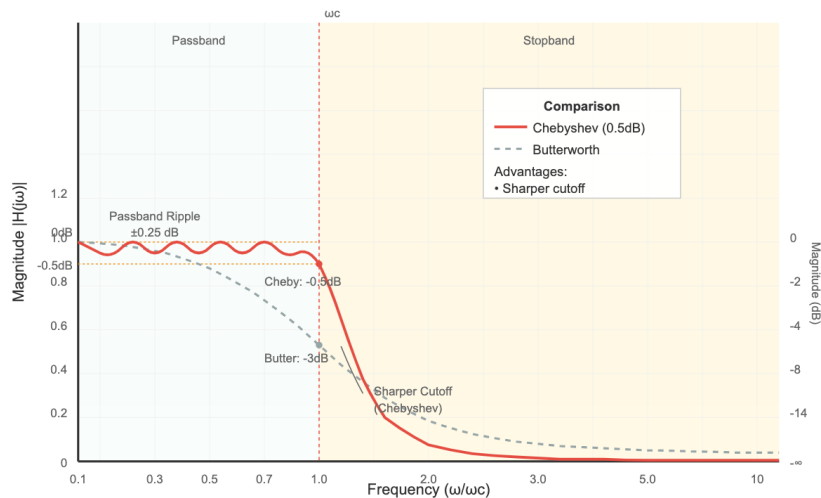


Figure I.7 Magnitude Chebyshev Filter.

As previously mentioned, these are the normalized transfer functions $H_N(s)$. In practice, two values for the ripple parameter are commonly used: $r = 0.5 \text{ dB}$ and 1 dB .

- Table of normalized polynomials for $r = 0.5 \text{ dB} = 1.059$ ou $\varepsilon = 0.3493$.

Ordre N	$H_N(s)$
1	$\frac{2.8628}{s + 2.8628}$
2	$\frac{1.4314}{s^2 + 1.4256 s + 1.5162}$
3	$\frac{0.7157}{s^3 + 1.2529 s^2 + 1.5349 s + 0.7157}$
4	$\frac{0.3579}{s^4 + 1.1974 s^3 + 1.7169 s^2 + 1.0255 s + 0.3791}$

- Table of normalized polynomials for $r = 1.0 \text{ dB} = 1.122$ ou $\varepsilon = 0.5089$.

Ordre N	$H_N(s)$
1	$\frac{1.9652}{s + 1.9652}$
2	$\frac{0.9826}{s^2 + 1.0977 s + 1.1025}$
3	$\frac{0.4913}{s^3 + 0.9883 s^2 + 1.2384 s + 0.4913}$
4	$\frac{0.2456}{s^4 + 0.9528 s^3 + 1.4539 s^2 + 0.7426 s + 0.2756}$

The denormalized transfer function $H(s)$ (low-pass, high-pass, band-pass, and band-stop) can be obtained from the normalized transfer function $H_N(s)$ through frequency transformations as follows:

Filter type	Transformation of s
low-pass	$\frac{s}{\omega_c}$
high-pass	$\frac{\omega_c}{s}$
band-pass	$s = \frac{s^2 + \omega_0^2}{s\varpi}$
band-stop	$s = \frac{s\varpi}{s^2 + \omega_0^2}$

where: $\omega_0 = \sqrt{\omega_l \omega_h}$, $\varpi = \omega_h - \omega_l$

II.1. Introduction to Random Signals

II.1.1. Random Variables

A random variable (RV) can be classified into discrete or continuous based on the nature of its possible values, which are unrelated to time.

- **Discrete Random Variable:** This type of RV can take on a countable number of distinct values. For instance, the number of students passing a module is a discrete RV.
- **Continuous Random Variable:** In contrast, a continuous random variable can assume any value within a given range. For example, the speed of vehicles on a highway is a continuous RV.

Example II.1: Consider rolling a standard six-faced die:

Possible values: $x_i = \{1, 2, 3, 4, 5, 6\}$

Probability of each value: $(x_i) = 1/6$

Total probability: $\sum P(x_i) = 1$

II.1.2. Probability Density Functions (PDFs)

The Probability Density Function (PDF) is a fundamental tool in probability and statistics used to describe the distribution of values for a continuous random variable.

The PDF $p_X(x)$ for a random variable X assigns a probability to each interval of X 's values. It is defined in such a way that the probability that X takes a value in an interval $[a, b]$ is given by the area under the curve of the function $p_X(x)$ over that interval.

The calculation of the probability that X belongs to the interval $[a, b]$ for a random variable X is determined by integrating the Probability Density Function (PDF) $p_X(x)$ over that interval. Mathematically, it is expressed as:

$$P(X \in [a, b]) = \int_a^b p_X(x) dx$$

Properties:

- The PDF is always non-negative: $p_X(x) \geq 0$ for all x .
- The total area under the curve of the PDF over the entire range of possible values is equal to 1:

$$\begin{aligned} \int_{-\infty}^{+\infty} p_X(x) dx &= 1 && \text{for CRV} \\ \sum_k p_X(k) &= 1 && \text{for DRV} \end{aligned}$$

Interpretation: The PDF provides a graphical representation of the probability of finding the random variable in a given interval. The higher the PDF in a certain interval, the greater the probability of

observing the random variable in that interval.

II.1.3. Cumulative Distribution Function (CDF)

The Cumulative Distribution Function (CDF) $F_X(x)$ of a random variable is the integral of the PDF. It provides the probability that the random variable X is less than or equal to x .

$$F_X(x) = P(X \leq x)$$

The relationship between the PDF $p_X(x)$ and the CDF $F_X(x)$ for a random variable X is given by :

$$F_X(x) = \int_{-\infty}^x p_X(u) du$$

II.1.4. Moments and statistical characteristics

In probability theory and statistics, moments and statistical characteristics are important measures that describe various properties of a probability distribution or a random variable.

In probability theory, the $n - th$ moment of a random variable X is a measure of the "spread" or "shape" of its probability distribution. The moment of order n is defined as:

$$\begin{aligned} E[X^n] &= \int_{-\infty}^{+\infty} x^n p_X(x) dx && \text{for CRV} \\ E[X^n] &= \sum_k x_k^n p_X(k) && \text{for DRV} \end{aligned}$$

These moments provide information about the shape, central tendency, and variability of the distribution. The first and second moments (mean and variance) are fundamental in characterizing a RV :

$\Rightarrow$ Case: $n = 1$ (First Moment): We obtain the expected value or mean of the random variable X :

$$\begin{aligned} E[X] &= \int_{-\infty}^{+\infty} x p_X(x) dx && \text{for CRV} \\ E[X] &= \sum_k x_k p_X(k) && \text{for DRV} \end{aligned}$$

The first moment, which is the mean, provides information about the central tendency or average value of the RV.

$\Rightarrow$ Case: $n = 2$ (Second Moment): We calculate the second moment of the random variable X :

$$\begin{aligned} E[X^2] &= \int_{-\infty}^{+\infty} x^2 p_X(x) dx && \text{for CRV} \\ E[X^2] &= \sum_k x_k^2 p_X(k) && \text{for DRV} \end{aligned}$$

The second moment, particularly, provides a measure of the *total average power* of the random variable, indicating how much the values of the variable typically deviate from the mean.

The PDF plot provides initial insights into our RV, complemented by the computation of various

statistical parameters. Two extensively used parameters are the *mathematical expectation (the mean)* μ_X and the *variance* σ_X^2 , defined by:

$\Rightarrow$ for continuous RV:

$$\mu_X = E[X] = \int_{-\infty}^{+\infty} x p_X(x) dx$$

$$\sigma_X^2 = Var[X] = \int_{-\infty}^{+\infty} (x - \mu_X)^2 p_X(x) dx$$

$\Rightarrow$ for discrete RV:

$$\mu_X = E[X] = \sum_k x_k p_X(k)$$

$$\sigma_X^2 = Var[X] = \sum_k (x_k - \mu_X)^2 p_X(k)$$

Indeed, the variance σ_X^2 is the square of the *standard deviation* σ_X , which provides a measure of the dispersion or spread of the RV. Mathematically, it is represented as:

$$\sigma_X^2 = Var[X] = E[(X - \mu_X)^2] = E[X^2] - E^2[X]$$

- **Standard deviation** σ_X is the positive square root of the variance, i.e., $\sigma_X = \sqrt{Var[X]}$.
- **Covariance** between two random variables X and Y is given by:

$$\varphi_{XY} = Cov[X, Y] = E[(X - E[X])(Y - E[Y])] = E[XY] - E[X]E[Y]$$

Two variables X and Y , are independent (uncorrelated) if their covariance is zero: $Cov[X, Y] = 0$

- **Autocovariance** φ_{XX} of RV X (equivalent to variance) is given by:

$$\varphi_{XX} = Cov[X, X] = E[X^2] - E^2[X] = \sigma_X^2$$

- **Correlation** R_{XY} between two RV X and Y is given by:

$$R_{XY} = \frac{Cov[X, Y]}{\sqrt{Var[X]}\sqrt{Var[Y]}} = \frac{\varphi_{XY}}{\sigma_X \sigma_Y}$$

- **Autocorrelation** R_{XX} of a RV X is given by:

$$R_{XX} = \frac{\varphi_{XX}}{\sigma_X^2} = 1$$

These measures provide insights into the relationships, variability, and dependencies between random variables, which are crucial in understanding their behavior and properties.

II.1.5. Common Probability Distributions for Random Variables

II.1.5.1. Bernoulli Distribution

For a discrete random variable X with parameters a and b , and associated probabilities $p_a = p$ and $p_b = (1 - p)$ respectively. Here, a and b are the possible values that the random variable X can take.

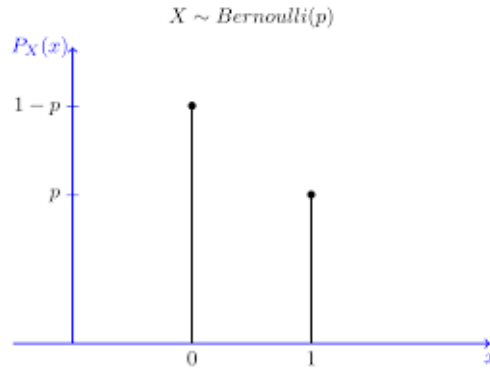


Figure II.1 Bernoulli Distribution.

The Probability Density Function (PDF) can be written as: $p_X(x) = p_a\delta(x - a) + p_b\delta(x - b)$.

The mean or expected value $\mu : E[X] = ap_a + bp_b$

The $n - th$ moment: $E[X^n] = a^n p_a + b^n p_b$.

Example I.18: We're flipping a coin once and our random variable is $X = 1$ if the outcome is *heads* and $X = 0$ if the outcome is *tails*. Then, $X \sim \text{Bernoulli}(1/2)$.

II.1.5.2. Uniform Distribution

The random variable X following a uniform distribution on the interval $[a, b]$, it means that the probability is constant (or equal) for any value of x within the range $[a, b]$. Mathematically, the probability density function $p_X(x)$ for this uniform distribution is defined as:

$$p_X(x) = \begin{cases} \frac{1}{b-a} & a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

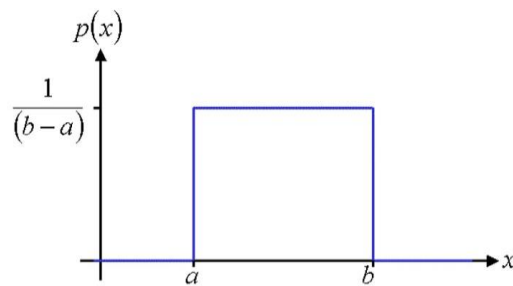


Figure II.2 Uniform Distribution.

The mean or expected value $\mu : E[X] = \int_a^b \frac{x}{b-a} dx = \frac{a+b}{2}$

The second moment: $E[X^2] = \int_a^b \frac{x^2}{b-a} dx = \frac{a^2+ab+b^2}{3}$

Example II.2: rolling a fair six-sided die. Assuming the die is not biased and each face has an equal chance of landing face up, the outcome of rolling the die follows a discrete uniform distribution.

In this case:

- The minimum possible outcome is 1 (rolling a one on the die).
- The maximum possible outcome is 6 (rolling a six on the die).

Probability Density Function (PDF): Each face of the die has an equal probability of 1/6.

II.1.5.3. Normal (Gaussian) Distribution

Is one of the most important and widely used probability distributions in statistics and various fields. It's characterized by its bell-shaped curve, symmetric around the mean μ , and defined by two parameters: the mean μ and the standard deviation σ .

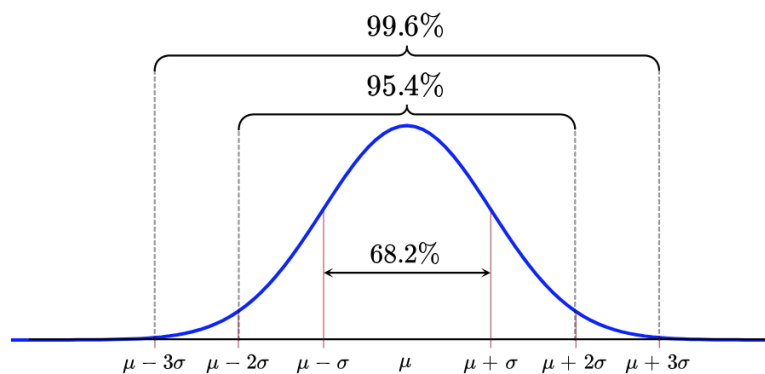


Figure II.2 Gaussian Distribution.

The probability density function (PDF) of a normal distribution is given by:

$$p_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Examples:

- Heights or weights of a population.
- The background noise in communication or electronic systems.
- It's widely used in statistics, physics, finance, engineering, and many other fields due to its many applications and its mathematical tractability.

II.1.6. joint probability distribution of two RVs X and Y

The joint probability density function $p_{XY}(x, y)$ for continuous random variables enables the calculation of the probability that the pair of random variables (X, Y) falls within a specific region Δ in the plane. It represents the probability that X takes a specific value x and Y takes a specific value y simultaneously. The probability that the pair (X, Y) belongs to a specific region Δ in the plane is calculated using the following formula:

$$P((X, Y) \in \Delta) = \iint_{\Delta} p_{XY}(x, y) dx dy$$

The joint probability density function $p_{XY}(x, y)$ satisfies the following properties:

$$\Rightarrow p_{XY}(x, y) \geq 0$$

$$\Rightarrow \iint_{\mathbb{R}^2} p_{XY}(x, y) dx dy = 1$$

For discrete random variables, they are characterized by their sets of possible discrete values: $\{a_1, a_2, \dots, a_k, \dots, a_L\}$ for X and $\{b_1, b_2, \dots, b_k, \dots, b_L\}$ for Y:

$$p_{XY}(a_k, b_k) = P(X = a_k, Y = b_k)$$

The probabilities $p_{XY}(a_k, b_k)$ must satisfy:

$$\Rightarrow p_{XY}(a_k, b_k) \geq 0$$

$$\Rightarrow \sum_k \sum_k p_{XY}(a_k, b_k) = 1$$

Independence of Two Random Variables X and Y: Two random variables X and Y are independent if and only if knowing the value of X provides no information about the value of Y, and vice versa. Mathematically, this is expressed as:

$$p_{XY}(x, y) = p_X(x) p_Y(y) \quad \text{for CRV}$$

$$P(X = a_k, Y = b_k) = P(X = a_k) \cdot P(Y = b_k) \quad \text{for DRV}$$

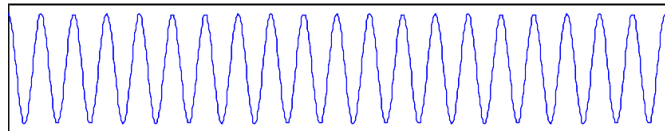
II.2. Random Signals

II.2.1. Deterministic signals:

Deterministic signals are signals whose behavior is entirely determined and predictable based on time and frequency. They are entirely specified by mathematical equations, functions, or constants.

For example, a sinusoidal signal is a deterministic signal because it can be described by an equation of the form:

$$x(t) = A \sin(2\pi f t + \phi)$$



where:

$$\Rightarrow x(t) \text{ is the value of the signal at time } t,$$

$$\Rightarrow A \text{ is the amplitude of the signal,}$$

$\Rightarrow f$ is the frequency of the signal,

$\Rightarrow \phi$ is the phase of the signal.

Each value of t gives a specific value for $x(t)$ using this equation.

II.2.2. Random Signals

A random signal is a *stochastic process*, present an evolution of a random variable with respect to the time. For the measured signal $x(t)$, each observed value in $x(t)$ is a realization of the random variable $X(t)$. For one realization of an event $\omega \in \Omega$, the function $X(t, \omega)$ is called a *trajectory* of the random signal X .

A *random process* can be viewed in two different ways, depending on what is considered as the variable:

- **Trajectory (or Realization):** When considering a specific random event ω_i (a particular "trial" or scenario),
- **Random Variable:** If we fix a specific instant t_i .

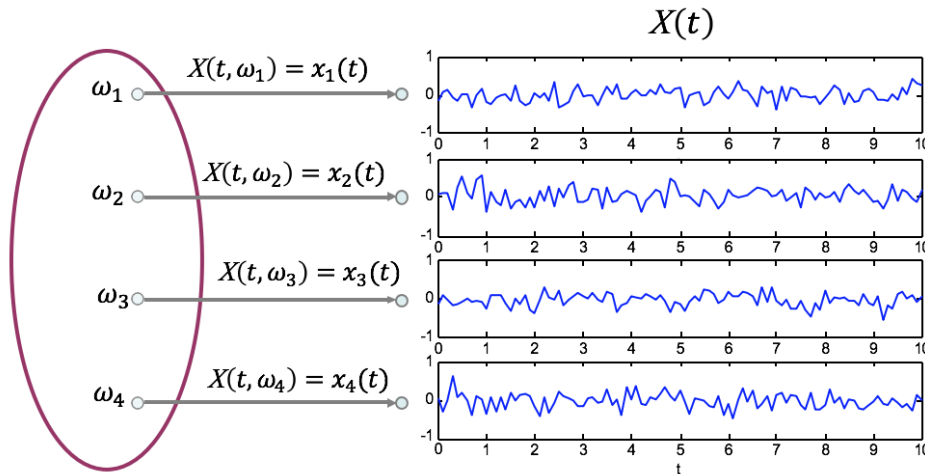


Figure II.4 Random Process Realizations.

Examples :

$\Rightarrow$ White noise is a classic example of a random process.

$\Rightarrow$ Sinusoid with Random Phase: $X(t) = A \sin(2\pi f_0 t + \Phi)$, Φ is a random variable.

$\Rightarrow$ Sinusoid with Random Amplitude: $X(t) = \mathbf{A} \sin(2\pi f_0 t + \Phi)$, $\mathbf{A}$ is a random variable.

- **Continuous-Time Random Process (CT):** A random process is said to be CT if $t \in \mathbb{R}$ (real).
- **Discrete-Time Random Process (DT):** A random process is said to be DT if $t \in \mathbb{Z}$ (integers).

Unlike deterministic signals, a random signal is described using probability theory and typically involves statistical properties, such as mean, variance, distribution, and correlation functions.

II.2.3. Statistical Moments

In the context of random signals, statistical moments are essential tools to analyze and characterize the properties of the signal. Here, we'll discuss how statistical moments are applied to random signals:

II.2.3.1. First-Order Statistical Moment of a Random Process $X(t)$ (Mean)

The mean of a random signal $X(t)$ is a fundamental statistical moment. It provides the average value or central tendency of the signal over a specific time interval. The mean of a continuous random signal $X(t)$ is calculated as:

$$\mu_X(t) = E[X(t)] = \int_{-\infty}^{+\infty} x p_{X(t)}(x) dx$$

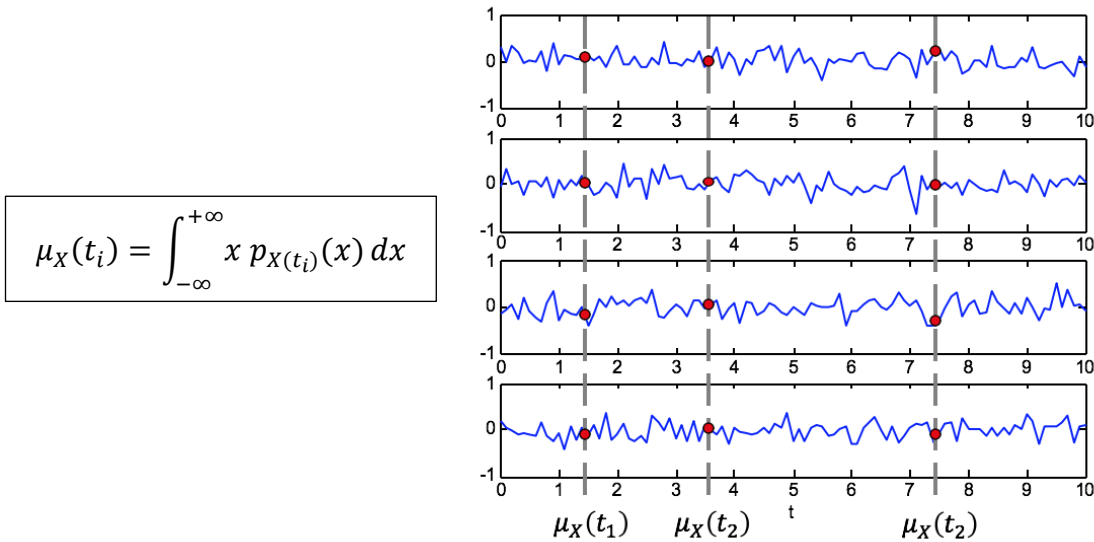


Figure II.5 First-Order Statistical Moment.

For discrete signals, the mean is calculated similarly, replacing the integral with a sum.

Example II.2: Let's consider the stochastic process $X(t)$ defined by: $X(t) = a + bt$, where a and b are random variables (RVs) with known probabilities. We'll calculate the mean (average) of this stochastic process.

$$\mu_X(t) = E[a + bt] = \mu_a + \mu_b t$$

And the variance by:

$$\sigma_X^2(t) = Var[X(t)] = E[(x(t) - \mu_X(t))^2] = E[X^2(t)] - \mu_X^2(t)$$

Example II.3:

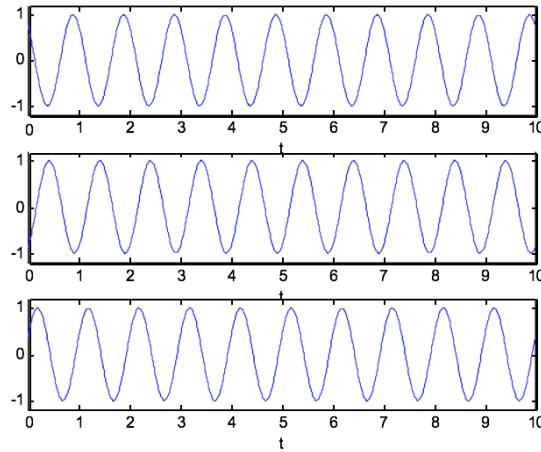
- Let $X(t)$ be a random process described by:

$$X(t) = a \cos(\omega t + \varphi)$$

where A is a constant and φ is a random variable uniformly distributed in the interval $[-\pi, \pi]$. The probability density function (PDF) of φ is given by:

$$p_{X(t)}(\varphi) = \begin{cases} \frac{1}{2\pi} & \varphi \in [-\pi, \pi] \\ 0 & \text{otherwise} \end{cases}$$

Here example of three specific realizations (épreuves) of the random process



We'll calculate the mean (average) of this stochastic process as:

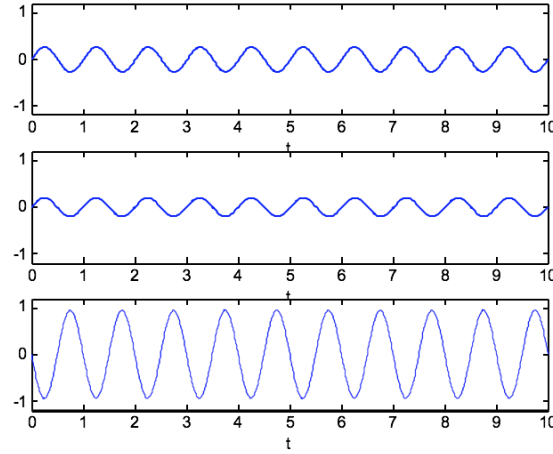
$$\mu_X(t) = E[a \cos(\omega t + \varphi)] = \int_{-\pi}^{\pi} a \cos(\omega t + \varphi) p_{X(t)}(\varphi) d\varphi = \frac{a}{2\pi} \int_{-\pi}^{\pi} \cos(\omega t + \varphi) d\varphi = 0$$

And the variance by:

$$\begin{aligned} \sigma_X^2(t) &= Var[X(t)] = E[(x(t) - \mu_X(t))^2] = E[X^2(t)] - \mu_X^2(t) \\ &= \int_{-\pi}^{\pi} a^2 \cos^2(\omega t + \varphi) p_{X(t)}(\varphi) d\varphi = \frac{a^2}{2} \end{aligned}$$

- Now, suppose that φ is deterministic (i.e., non-random), and a is a random variable with a probability density function $p_{X(t)}(a)$, having a mean μ_a and variance σ_a^2 .

Here example of three specific realizations (épreuves) of the random process



We'll calculate the mean (average) of this stochastic process as:

$$\mu_X(t) = E[a \cos(\omega t + \varphi)] = \mu_a \cos(\omega t + \varphi)$$

And the variance by:

$$\sigma_X^2(t) = \sigma_a^2 \cos^2(\omega t + \varphi)$$

II.2.3.2. Second-Order Statistical Moment of a Random Process $X(t)$

Also known as the variance of the process, is an important statistical measure. It corresponds to the mean of the product of two instances of the signal $X(t_1)$ and $X(t_2)$.

For a random process $X(t)$, the autocorrelation function $R_X(t_1, t_2)$ provides the correlation between the values of the signal X at two different instants t_1 and t_2 . It is defined as:

$$R_X(t_1, t_2) = E[X(t_1)X(t_2)] = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x_1 x_2 p_{X(t_1)X(t_2)}(x_1, x_2) dx_1 dx_2$$

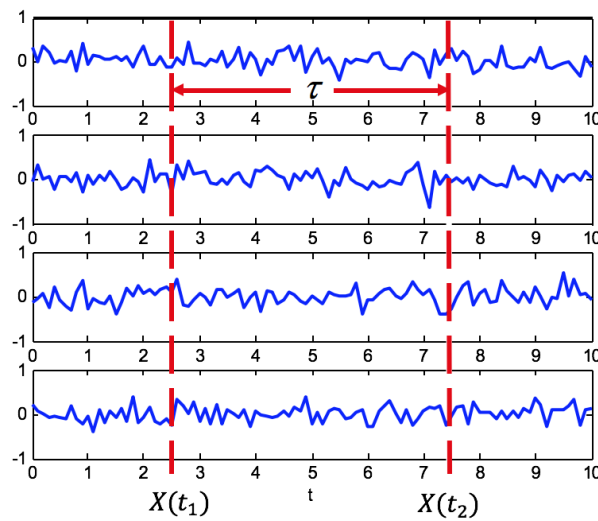


Figure II.6 Second-Order Statistical Moment.

The variance σ_X^2 , which is a second-order moment, is a special case of the autocorrelation function when $t_1 = t_2$, i.e.

The *autocovariance function* $C_X(t_1, t_2)$ of a random process $X(t)$ is defined as:

$$C_X(t_1, t_2) = E[(X(t_1) - \mu_X(t_1))(X(t_2) - \mu_X(t_2))] = R_X(t_1, t_2) - \mu_X(t_1)\mu_X(t_2)$$

The correlation of two random processes $X(t)$ and $Y(t)$ defined with random variables X and Y at time instants t_1 and t_2 respectively, is called the *cross-correlation function* and defined as:

$$R_{XY}(t_1, t_2) = E[X(t_1)Y(t_2)]$$

Let's proceed with the calculation of the autocorrelation for the given example II.3.

$$\begin{aligned} R_X(t_1, t_2) &= E[a \cos(\omega t_1 + \varphi) a \cos(\omega t_2 + \varphi)] \\ &= E\left[\frac{a^2}{2}(\cos(\omega(t_1 + t_2) + 2\varphi) + \cos(\omega(t_1 - t_2)))\right] \\ &= \frac{a^2}{2} \cos(\omega(t_1 - t_2)) \end{aligned}$$

This result shows that the autocorrelation function $R_X(t_1, t_2)$ is a periodic function depending only on the difference $t_1 - t_2$, which aligns with our expectation for a *stationary process*.

II.2.4. Stationarity

II.2.4.1. Strict Sense Stationarity

Strict Sense Stationarity (SSS) of a random process $X(t)$ refers to the property where its statistical properties remain invariant when the time origin is shifted. In other words, the process $X(t)$ is considered strictly stationary if:

A random process $X(t)$ is said to be strictly stationary (SSS) if its statistical properties remain invariant when we change the time origin:

$$p_{X(t_1), \dots, X(t_k)}(x_1, \dots, x_k) = p_{X(t_1+c), \dots, X(t_k+c)}(x_1, \dots, x_k)$$

for all $x_1, \dots, x_k$ and at any c .

II.2.4.2. Wide Sense Stationarity

A random process $X(t)$ is considered wide-sense stationary (WSS) if its mean $\mu_X(t)$ is independent of time t and its autocorrelation function $R_X(t_1, t_2)$ depends only on the time difference $\tau = t_2 - t_1$.

Mathematically, for a WSS process:

1. *Constant Mean* $\mu_X(t)$:

$$\mu_X(t) = E[X(t)] = \int_{-\infty}^{+\infty} x p_{X(t)}(x) dx = \mu_X$$

2. *Autocorrelation* $R_X(t_1, t_2)$:

$$R_X(t_1, t_2) = E[x(t)x(t + \tau)] = R_X(\tau)$$

By setting $\tau = 0$, we have $E[x^2(t)] = R_X(0)$. This implies that the average power of a Wide Sense Stationary (WSS) process is constant and independent of time, and its value is $R_X(0)$.

It's important to note that a Strict Sense Stationary (SSS) process is inherently Wide Sense Stationary (WSS), but the reverse is not necessarily true.

Let's revisit Example II.3: The mean $\mu_X(t) = \mu_X = 0$ is a constant, and the autocorrelation $R_X(t_1, t_2) = \cos(\omega(t_1 - t_2)) = \frac{a^2}{2} \cos(\omega\tau)$ depends only on the time difference τ . This is a characteristic of a wide-sense stationary (WSS) process.

II.2.5. Temporal Moments

Temporal moments characterize the behavior of a random process over time. These moments provide valuable information about the signal's shape, symmetry, and other characteristics.

II.2.5.1. First-Order Temporal Moment of a Random Process $X(t)$

Also known as the *temporal mean*, is calculated for a specific realization $X(t, \omega_i)$ of a random process $X(t)$, is defined as follows:

$$\begin{aligned} \mu_X(T) = \langle X(t) \rangle &= \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} X(t, \omega_i) dt && \text{for CT} \\ \mu_X(N) = \langle X(t) \rangle &= \frac{1}{N+1} \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} X(n, \omega_i) && \text{for DT} \end{aligned}$$

T and N is the duration over which we calculate the mean.

II.2.5.2. Second-Order Temporal Moment of a Random Process $X(t)$

The temporal mean of the autocorrelation function for a given elementary function $x(t)x(t)$ can be defined in both continuous-time (CT) and discrete-time (DT) settings.

$$R_X(\tau, T) = \langle X(t)X(t + \tau) \rangle = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} X(t, \omega_i)X(t + \tau, \omega_i) dt \quad \text{for CT}$$

$$R_X(\tau, N) = \langle X(n)X(n + \tau) \rangle = \frac{1}{N+1} \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} X(n, \omega_i)X(n + \tau, \omega_i) \quad \text{for DT}$$

$\langle \cdot \rangle$ denotes the expectation or average.

II.2.6. Ergodicity

A random process $X(t)$ is considered *ergodic in mean* if its temporal average $\mu_X(T)$ converges to its statistical mean μ_X as the observation interval T tends towards infinity.

$$\lim_{T \rightarrow \infty} \mu_X(T) = \mu_X$$

In addition to the mean convergence, for a random process to be *ergodic in mean and variance*, it is required that the variance of its temporal mean, $\text{var}[\mu_X(T)]$, tends towards zero as the observation interval T tends towards infinity. Mathematically, this is represented as:

$$\lim_{T \rightarrow \infty} \text{var}[\mu_X(T)] = 0$$

A random process $X(t)$ is considered *ergodic in autocorrelation* if its temporal autocorrelation function $R_X(\tau, T)$, tends towards its autocorrelation function $R_X(\tau)$, as the observation duration T tends towards infinity. Mathematically, this is expressed as:

$$\lim_{T \rightarrow \infty} R_X(\tau, T) = R_X(\tau)$$

In addition to the convergence of the autocorrelation function, for a random process to be *ergodic in autocorrelation and variance*, it is required that the variance of its temporal autocorrelation function, $\text{var}[R_X(\tau, T)]$, tends towards zero as the observation duration T tends towards infinity. Mathematically, this is represented as:

$$\lim_{T \rightarrow \infty} \text{var}[R_X(\tau, T)] = 0$$

II.2.7. Relationship between Stationarity and Ergodicity

The *Birkhoff Ergodic Theorem* states that if a random signal is both stationary and ergodic, then the temporal and statistical moments are equal:

$$\lim_{T \rightarrow \infty} \mu_X(T) = \mu_X$$

$$\lim_{T \rightarrow \infty} R_X(\tau, T) = R_X(\tau)$$

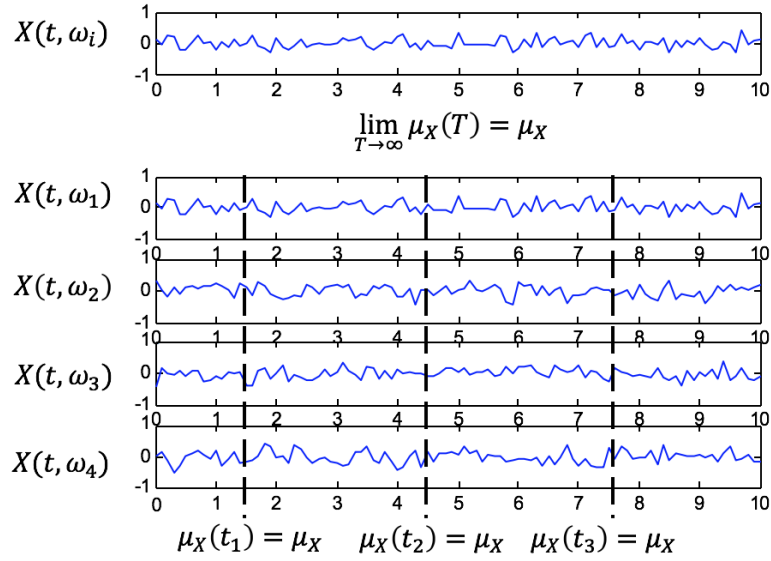


Figure II.7 Relationship between Stationarity and Ergodicity.

Example II.4: Consider a signal $X(t) = a \cos(\omega t + \varphi)$ for all t , where a is a constant and φ is a uniformly distributed random variable in the interval $[-\pi, \pi]$.

Mean:

$$\mu_X(t) = \mu_X = 0$$

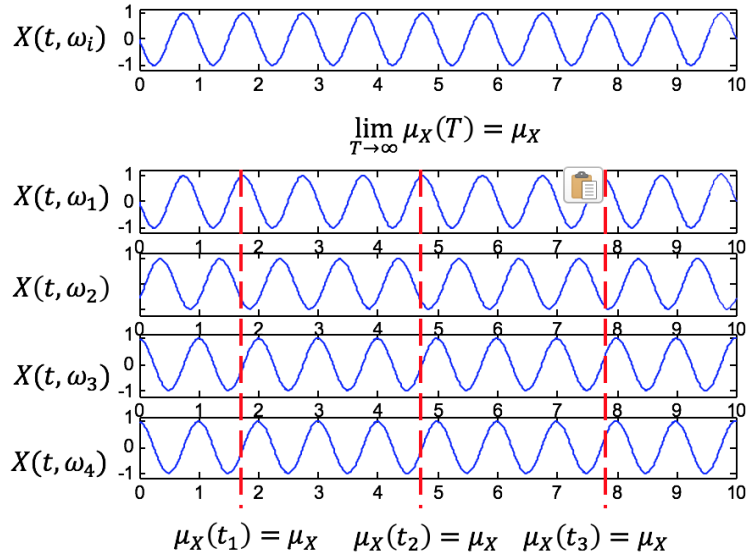
Auto-correlation:

$$R_X(t_1, t_2) = R_X(\tau) = \frac{a^2}{2} \cos(\omega \tau)$$

Since the process $X(t)$ is wide-sense stationary, we may question whether it is *ergodic in mean*.

$$\mu_X(T) = \langle X(t) \rangle = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} X(t, \omega_i) dt = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} a \cos(\omega t + \varphi) dt = 0$$

As T tends to infinity, the temporal mean $\mu_X(T)$ tends towards its statistical mean $\mu_X = 0$. Therefore, the process $X(t)$ exhibits *mean ergodicity*.



$$\begin{aligned}
 R_X(\tau, T) &= \langle X(t)X(t+\tau) \rangle = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} X(t, \omega)X(t+\tau, \omega_i) dt = \\
 &= \frac{a^2}{2T} \int_{-\frac{T}{2}}^{\frac{T}{2}} (\cos(\omega(2t+\tau) + 2\varphi) + \cos(\omega\tau)) dt \\
 &= \frac{a^2}{2T} \int_{-\frac{T}{2}}^{\frac{T}{2}} \cos(\omega\tau) dt = \frac{a^2}{2T} \cos(\omega\tau) \int_{-\frac{T}{2}}^{\frac{T}{2}} dt = \frac{a^2}{2} \cos(\omega\tau)
 \end{aligned}$$

Hence, as T approaches infinity, $\lim_{T \rightarrow \infty} R_X(\tau, T) = R_X(\tau)$. The random process $X(t)$ exhibits mean ergodicity, and its autocorrelation function is also ergodic.

II.3. Power Spectral Density:

The Wiener-Khinchin theorem establishes that the power spectral density of a random signal is the Fourier transform of its autocorrelation function $R_X(\tau)$:

$$S_X(f) = \int_{-\infty}^{+\infty} R_X(\tau) e^{-j2\pi f\tau} d\tau \quad \text{for CT}$$

$$S_X(f) = \sum_{\tau=-\infty}^{+\infty} R_X(\tau) e^{-j2\pi f\tau} \quad \text{for DT}$$

The power of a stationary and ergodic random signal can be obtained by integrating the power spectral density over the entire frequency spectrum, as follows:

$$\begin{aligned}
 P &= \lim_{T \rightarrow +\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} X(t, \omega) X(t, \omega) dt \\
 &= R_X(0) \\
 &= E[|x(t)|^2] \\
 &= \int_{-\infty}^{+\infty} S_X(f) df
 \end{aligned}$$

Let's continue with the previous example: $X(t) = a \cos(\omega t + \varphi)$

$$R_X(\tau) = \frac{a^2}{2} \cos(\omega \tau)$$

$$\begin{aligned}
 S_X(f) &= \frac{a^2}{2} \int_{-\infty}^{+\infty} \cos(\omega \tau) e^{-j2\pi f \tau} d\tau \\
 &= \frac{a^2}{4} [\delta(f - f_0) + \delta(f + f_0)]
 \end{aligned}$$

The power P :

$$\begin{aligned}
 P &= \int_{-\infty}^{+\infty} S_X(f) df \\
 &= \frac{a^2}{2}
 \end{aligned}$$

II.4. Spectral Estimation (Periodogram and Correlogram)

II.4.1. Deterministic Signals

Let $x(n)$ be a sequence of deterministic real data, characterized by finite energy, as is often the case in various domains. Mathematically, the condition for finite energy is expressed as:

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

The Discrete Fourier Transform (DFT) of this sequence is defined as follows:

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

An important property of the DFT is that it can be used to define the *Energy Spectral Density*, denoted $S_x(f)$, which is simply the square of the magnitude of the DFT $|X(f)|^2$.

$$S_x(f) = |X(f)|^2$$

Now, let's consider the auto-correlation of $x(n)$, denoted $R_x(k)$, which measures the correlation between the signal $x(n)$ at different shifts k . Mathematically, it is defined as:

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

It's worth noting that the power spectral density of the signal, $S_x(f)$, can be expressed in terms of the DFT of the autocorrelation, $R_x(k)$, as follows:

$$\sum_{k=-\infty}^{+\infty} R_x(k) e^{-j2\pi f k} = S_x(f)$$

This relationship demonstrates that the spectral information of the signal, characterized by $S_x(f)$, can be obtained from the autocorrelation of the signal. Spectral estimation plays a significant role in understanding the frequency components and characteristics of deterministic signals.

II.4.2. Random Signals

II.4.2.1. Periodogram

Let's consider $x(n)$ as a stationary random signal with finite average power, where $E[x(n)] = 0$. The periodogram is a powerful tool used for spectral estimation in signal processing. The spectral estimator, known as the periodogram, is defined as:

$$S_x(f) = \lim_{N \rightarrow +\infty} E \left[\frac{1}{N+1} \left| \sum_{n=-N/2}^{N/2} x(n) e^{-j2\pi f n} \right|^2 \right]$$

Here, $S_x(f)$ represents the spectral density at frequency f . For a finite random sequence $x(n)$, the periodogram is practically estimated using:

$$\hat{S}_x(f) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x(n) e^{-j2\pi f n} \right|^2 = \frac{1}{N} |X(f)|^2$$

II.4.2.2. Correlogram

The correlogram is another spectral estimation technique, primarily used for random signals. The method involves computing an estimate of the autocorrelation function $R_x(k)$ and then taking its Fourier transform to obtain the spectral estimate.

The spectral estimator, based on the correlogram, is given by:

$$\hat{S}_x(f) = \sum_{k=-N+1}^{N-1} \hat{R}_x(k) e^{-j2\pi f k} = S_x(f)$$

The biased autocorrelation function, denoted as $\hat{R}_x(k)$, is calculated using the formula:

$$\hat{R}_x(k) = \frac{1}{N} \sum_{n=k}^{N-1} x(n) x(n-k) \quad 0 \leq k \leq N-1$$

In this biased calculation, the summation is performed from k to $N-1$, ensuring that the lag k is taken into account while calculating the autocorrelation.

II.5. Correlation Functions and Matrices

Let $x(k)$ be a complex, N -dimensional vector signal in a Signal Space of Length (SSL):

$$\mathbf{x}(n) = [x(k) \ x(k-1) \ \dots \ x(k-i) \ \dots \ x(k-N+1)]^T$$

The mean of this signal is a vector $\mu_x(k)$ of the same dimension, given by:

$$\mu_x(n) = E[\mathbf{x}(n)]$$

$$= [\mu_x(k) \ \mu_x(k-1) \ \dots \ \mu_x(k-i) \ \dots \ \mu_x(k-N+1)]^T$$

where $\mu_x(k) = E[x(k)]$. The correlation matrix is defined as:

$$\mathbf{R}_x = E[\mathbf{x}(k)\mathbf{x}^H(k)] = \begin{bmatrix} R_x(0) & R_x(1) & \dots & R_x(N-1) \\ R_x(-1) & R_x(0) & \ddots & R_x(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ R_x(-N+1) & R_x(-N+2) & \dots & R_x(0) \end{bmatrix}$$

where $*$ represents the conjugation operator, H represents the conjugate transpose, and $R_x(i) = E[x(k)x^*(k-i)]$. The element $R_x(0)$ on the main diagonal is always real and positive. The autocorrelation matrix plays a crucial role in signal processing. This matrix possesses the property of Hermitian symmetry in the complex case $\mathbf{R}_x = \mathbf{R}_x^H$ and regular symmetry in the real case $\mathbf{R}_x = \mathbf{R}_x^T$.

II.6. Remarkable Stochastic Processes

In the realm of stochastic processes, there exist several notable types, each with distinct characteristics and applications. Understanding these processes is fundamental in various fields, including statistics, finance, physics, and engineering. Here, we briefly introduce a few prominent stochastic processes in the field of telecommunications:

II.6.1. Gaussian Random Process

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution. Gaussian processes are a key concept in Bayesian optimization, machine learning, and spatial statistics.

Consider a random process $X(t)$ and define k random variables $X(t_1), X(t_2), \dots, X(t_k)$ corresponding to k measurement instants $t_1, t_2, \dots, t_k$. We form a random vector $\mathbf{X}$ ($k \times 1$ matrix)

defined as:

$$\mathbf{X} = \begin{bmatrix} X(t_1) \\ X(t_2) \\ \vdots \\ X(t_k) \end{bmatrix}$$

Let $\mathbf{x}$ also be an n-dimensional vector (a $k \times 1$ matrix) defined as:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix}$$

This allows us to denote $\{\mathbf{X} \leq \mathbf{x}\}$ as the event $\{X(t_1) \leq x_1, X(t_2) \leq x_2, \dots, X(t_k) \leq x_k\}$. $X(t)$ is, by definition, a Gaussian (or normal) process if X exhibits a jointly Gaussian multivariable density for any finite set of values $\{t_i\}$ and for any k . The multivariable density of a Gaussian process is given by:

$$p_X(\mathbf{x}) = \frac{1}{(2\pi)^{k/2} |\det \mathbf{C}|^{k/2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right]$$

Here, T represents matrix transposition, $\boldsymbol{\mu}$ is the mean vector, and $\mathbf{C}$ is the covariance matrix defined as:

$$\boldsymbol{\mu} = E[\mathbf{X}] = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_k \end{bmatrix} = \begin{bmatrix} E[X(t_1)] \\ E[X(t_2)] \\ \vdots \\ E[X(t_k)] \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1k} \\ C_{21} & C_{22} & \ddots & C_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ C_{k1} & C_{k2} & \dots & C_{kk} \end{bmatrix}$$

In this representation:

$$C_{ij} = C_X(t_i, t_j) = R_X(t_i, t_j) - \mu_X(t_i)\mu_X(t_j)$$

where $R_X(t_i, t_j)$ represents the autocovariance of (t_i) and $X(t_j)$, and $|\det \mathbf{C}|$ is the determinant of the matrix $\mathbf{C}$.

The key properties of a Gaussian process are as follows:

1. A Gaussian process is fully characterized by its mean values $\mu_X(t_i) = E[X(t_i)]$ and its autocorrelation functions $R_X(t_i, t_j) = E[x(t_i)x(t_j)]$ for $i, j = 1, \dots, k$.

2. If the random variables $X(t_i)$, $i = 1, \dots, k$ are uncorrelated (i.e., $C_{ij} = 0, i \neq j$), then $X(t_i)$ are independent.
3. If a Gaussian process $X(t)$ is wide-sense stationary, then it is also strict-sense stationary.
4. Applying a Gaussian process at the input of a linear system results in a Gaussian random process at the output.

II.6.2. White Noise

A white noise process $B(t)$ is a Stationary Stochastic Process (SSP), typically centered, with an autocorrelation function that is zero everywhere except at 0. Its Power Spectral Density (PSD) $S_B(f)$ is constant over the entire frequency axis.

The autocorrelation function of white noise is succinctly expressed as:

$$R_B(\tau) = \sigma_B^2 \delta(\tau)$$

where $\delta(\tau)$ is the Dirac delta function, and σ_B^2 denotes the variance. The characteristic PSD of a white noise process is represented by:

$$S_B(f) = \sigma_B^2$$

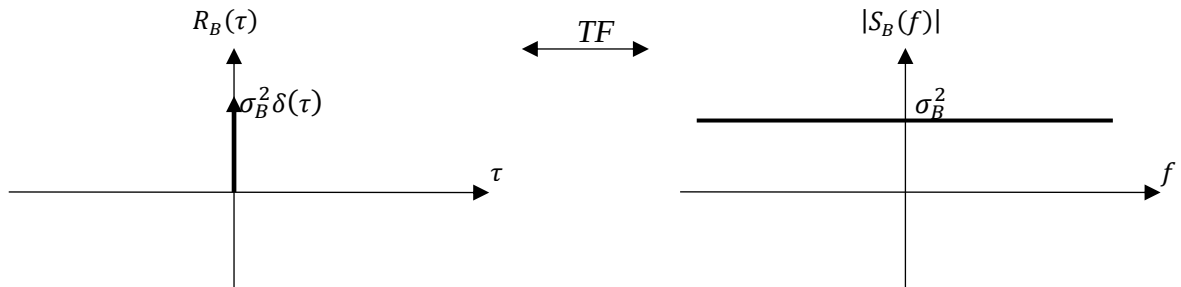


Figure II.8 Autocorrelation function and PSD of white noise.

II.6.3. Band-Limited White Noise

A random process is called "band-limited white noise" if it meets a specific rule:

$$S_B(f) = \begin{cases} \sigma_B^2 & |f| \leq f_B \\ 0 & \text{ailleurs} \end{cases}$$

This Leads to:

$$R_B(\tau) = \frac{1}{2\pi} \int_{-f_B}^{f_B} \sigma_B^2 e^{j2\pi f \tau} df = 2\sigma_B^2 \frac{\sin 2\pi f_B \tau}{2\pi f_B \tau}$$

The following figure represents the power spectral density and the autocorrelation function of band-

limited white noise.

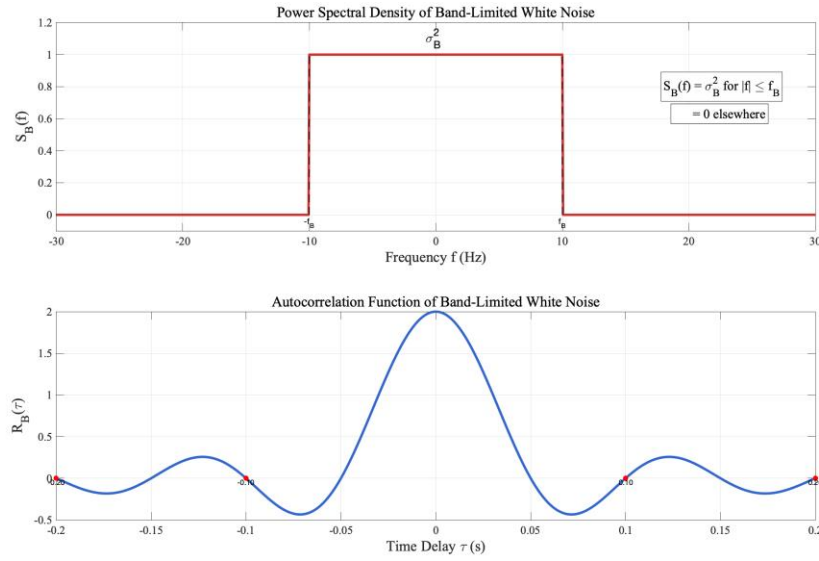


Figure II.9 Autocorrelation function and PSD of band-limited white noise.

II.6.4. Markovian Stochastic Process

A stochastic process $X(t)$ is termed Markovian if it satisfies two essential conditions, emphasizing its "memoryless" nature:

1. the state of the process at a given time t , X_t depends only on its immediately preceding state X_{t-1} .

$$p_X(X_t = j / X_1 = i_1, X_1 = i_1, \dots, X_{t-1} = i_{t-1}) = p_X(X_t = j / X_{t-1} = i_{t-1})$$

2. regardless of the time period t , the probability of transitioning from state i to state j remains unchanged.

$$\forall t, 1 < t \leq N, p_X(X_t = j / X_{t-1} = i) = C$$

A Markov process can be summarized by a transition matrix T where each element $T(i, j)$ represents the transition probability from state i to state j .

$$T(i, j) = p_X(X_t = j / X_{t-1} = i)$$

A Markov process can also be effectively represented by a finite automaton:

1. *State Representation*: Each state within the Markov process corresponds to a state in the automaton. The set of states in the automaton mirrors the possible states the process can occupy.
2. *Transitions and Probabilities*: Transitions from one state i to another state j in the automaton are labeled with the respective transition probability $T(i, j)$.

Example II.5:

Consider a telecommunication network that can be modeled as a Markov process. Let's focus on a simplified scenario where the system can be in two states: Busy (state 1) or Idle (state 2). This could represent the state of a communication channel or a server in a telecommunication system.

1. State Definitions:

⇒ State 1 (Busy): The channel or server is occupied, handling a communication or a task.

⇒ State 2 (Idle): The channel or server is available, waiting for a new communication or task.

2. Probabilities and Transitions:

⇒ Probability of staying Busy: $P_{Busy} = \alpha$

⇒ Probability of transitioning from Busy to Idle: $P_{Busy-Idle} = 1 - \alpha$

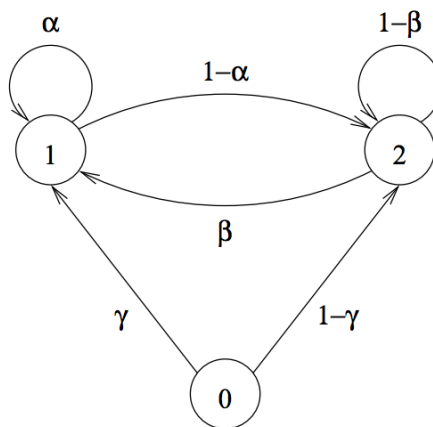
⇒ Probability of staying Idle: $P_{Idle} = 1 - \beta$

⇒ Probability of transitioning from Idle to Busy: $P_{Idle-Busy} = \beta$

We can define a transition matrix for this two-state Markov chain:

$$T = \begin{bmatrix} \alpha & 1 - \alpha \\ 1 - \beta & \beta \end{bmatrix}$$

In this scenario, the transition probabilities represent how the channel's or server's state changes over time, which is fundamental in understanding and optimizing the performance of telecommunication networks.



In a given state, the probability of being Busy is denoted as $P_{Busy}(0) = \gamma$ and the probability of the system being Idle in that state is $P_{Idle}(0) = 1 - \gamma$. These probabilities represent the initial conditions for the system being in the Busy and Idle states.

Example II.6:

For the given numerical values ($\gamma = 0.3, \alpha = 0.6, \beta = 0.2$), we have the following results:

- $\Rightarrow$ Initial probability of being Busy ($P_{Busy}(0) = \gamma = 0.3$)
- $\Rightarrow$ Transition probability from Busy to Idle $P_{Busy-Idle} = 1 - \alpha = 0.6$
- $\Rightarrow$ Initial probability of being Idle $P_{Idle}(0) = 1 - \gamma = 0.7$
- $\Rightarrow$ Transition probability from Idle to Busy $P_{Idle-Busy} = \beta = 0.2$

Let's calculate the probabilities of being Busy and Idle at the next state ($\dot{P}_{Busy}$ and $\dot{P}_{Idle}$).

- Probability of being Busy ($\dot{P}_{Busy}$) at the next state:

$$\dot{P}_{Busy} = P_{Busy}(0) P_{Busy} + P_{Idle}(0) P_{Idle-Busy}$$

$$\dot{P}_{Busy} = \gamma \cdot \alpha + (1 - \gamma) \cdot \beta$$

$$\dot{P}_{Busy} = 0.3 \times 0.6 + 0.7 \times 0.2 = 0.32$$

- Probability of being Idle ($\dot{P}_{Idle}$) at the next state:

$$\dot{P}_{Idle} = P_{Idle}(0) P_{Idle} + P_{Busy}(0) P_{Busy-Idle}$$

$$\dot{P}_{Idle} = (1 - \gamma) \cdot (1 - \beta) + \gamma \cdot (1 - \alpha)$$

$$\dot{P}_{Idle} = 0.7 \times 0.8 + 0.3 \times 0.4 = 0.68$$

In the previous chapter, we discussed nonparametric spectral estimation methods like periodogram and correlogram, which do not assume a specific signal model. In this chapter, we delve into parametric methods that assume a particular model, yielding more accurate estimates when the assumed model matches the actual signal generation process. Nevertheless, nonparametric approaches remain valuable in cases of limited or no prior signal information or when the signal's nature is not clearly understood or precisely defined.

III.1. Stochastic Models

In the field of signal processing and time series analysis, stochastic models are mathematical representations that describe the random behavior of a process. Here's an overview of the mentioned stochastic models:

III.1.1. Auto-Regressive Models (AR)

These models describe a variable as a linear combination of its past values. An $AR(p)$ model uses the past p observations to predict the current value.

A signal $x(n)$ as a realization of an autoregressive process of order p . The signal is represented by the equation:

$$x(n) = \sum_{k=1}^p a_k x(n-k) + v(n)$$

where a_k for $k = 1, \dots, p$ are the coefficients of the model, and $v(n)$ is a white noise. This representation can also be written as:

$$x(n) - \sum_{k=1}^p a_k x(n-k) = v(n)$$

After applying the Z-transform, the representation becomes:

$$\left(1 - \sum_{k=1}^p a_k Z^{-k}\right) X(Z) = V(Z)$$

we have:

$$H_{AR}(Z).X(Z) = V(Z)$$

where $H_{AR}(Z) = 1 - \sum_{k=1}^p a_k Z^{-k}$.

Generating Autoregressive Signal from White Noise: For given white noise $\{v(n)\}$, the autoregressive signal $\{x(n)\}$ can be generated using the relation:

$$X(Z) = \frac{V(Z)}{H_{AR}(Z)} = H_G(Z).V(Z)$$

Where $H_G(Z)$, the inverse of the FIR filter $H_{AR}(Z)$, is a "generator" filter with an infinite impulse response (IIR).

⇒ *Producing White Noise from Autoregressive Process*: For given the autoregressive process $\{x(n)\}$, the filter $H_{AR}(Z)$ can be used to produce white noise $\{v(n)\}$; thus, $H_{AR}(Z)$ acts as an "analyzer" filter with an infinite impulse response (IIR).

⇒ *Zeros Poles, and Stability*:

- The filter $H_{AR}(Z) = 1 - \sum_{k=1}^p a_k Z^{-k}$ is an *all-zero filter* because it only has zeros (exactly $p - 1$ zeros). This filter is implicitly stable, meaning that the response to a finite energy input is also of finite energy.
- On the other hand, the filter $H_G(Z) = 1/H_{AR}(Z)$ is an *all-pole filter*. The poles of $H_G(Z)$ are, obviously, the zeros of $H_{AR}(Z)$. This filter is stable if and only if all the poles of $H_G(Z)$ are located inside the unit circle in the z -plane.

III.1.2.Moving Average Models (MA)

Moving Average models describe a variable as a linear combination of previous random error terms. An $MA(q)$ model uses the past q errors to predict the current value. The generator filter for the MA signal $\{x(n)\}$ is an all-zero filter, impacted by white noise. The signal at discrete time n is given by:

$$x(n) = \sum_{k=0}^q b_k v(n - k)$$

Here, $v(n)$ is zero-mean white noise with variance σ_v^2 , and b_k are the coefficients of the filter.

This can also be represented in the z -domain as:

$$X(Z) = H_{MA}(Z).V(Z)$$

where $H_{MA}(Z)$ is the transfer function of the MA filter.

- *Whitening an MA Signal*: Conversely, to "whiten" an MA signal $u\{x(n)\}$, the relation can be sought as:

$$V(Z) = \frac{1}{H_{MA}(Z)}.X(Z) = H_B(Z).X(Z)$$

the filter $H_B(Z)$ is an all-pole filter.

III.1.3.Auto-Regressive Moving Average Models (ARMA)

ARMA models combine aspects of both AR and MA models. They describe the variable based on its

past values and past random error terms. An ARMA process $\{x(n)\}$ satisfies the equation:

$$\sum_{k=0}^p a_k x(n-k) = \sum_{k=0}^q b_k v(n-k)$$

where a_k and b_k are the coefficients of the AR and MA parts respectively, and $v(n)$ is noise.

$$H_{AR}(Z).X(Z) = H_{MA}(Z).V(Z)$$

This leads to the ratio of $X(Z)$ to $V(Z)$ being:

$$\frac{X(Z)}{V(Z)} = \frac{H_{MA}(Z)}{H_{AR}(Z)}$$

In the ARMA filter, the zeros are determined by $H_{MA}(Z)$, and the poles are determined by $H_{AR}(Z)$. Since the AR and MA orders are p and q , respectively, the ARMA model is referred to as *ARMA* (p, q).

III.1.4. Complex sinusoids in noise Models

The model of complex sinusoids in noise describes a signal composed of a combination of complex sinusoidal components corrupted by the presence of noise. This model is fundamental in signal processing and communication systems, especially when dealing with various types of signals embedded in noisy environments. The general representation of this model can be expressed as follows:

$$x(n) = \sum_{k=1}^p A_k e^{j2\pi f_k n} + v(n)$$

where:

- $x(n)$ represents the signal at discrete time n .
- A_k denotes the amplitude of the $k - th$ complex sinusoidal component.
- f_k denotes the frequency of the $k - th$ complex sinusoidal component.
- $v(n)$ is a sample of a noise process, which is not necessarily white.

This model is widely used in various domains such as telecommunications, audio processing, signal analysis, and more, enabling the study and processing of signals in the presence of noise.

The AR model is computationally advantageous over MA and ARMA models because its coefficients can be directly solved using a system of linear equations known as the Yule-Walker equations. This simplifies coefficient estimation compared to the more complex methods required for MA and ARMA models.

III.2. Spectrum Estimation Based on AR Models

An AR model (or process) is fully characterized by:

- The variance of the white noise $\{v(n)\}$ that excites the filter, denoted as σ_v^2 .
- The AR coefficients: a_k for $k = 1, \dots, p$ (where $a_0 = 1$).

The model is described by the equation:

$$X(Z) \left[1 - \sum_{k=1}^p a_k Z^{-k} \right] = V(Z)$$

Let the transfer function $H(Z)$ be defined as the ratio of the output $X(Z)$ to the input $V(Z)$. Then

$$H(Z) = \frac{X(Z)}{V(Z)} = \frac{1}{1 - \sum_{k=1}^p a_k Z^{-k}}$$

By setting $Z = e^{j2\pi f T_e}$ in $H(Z)$. Then

$$H(f) = \frac{1}{1 - \sum_{k=1}^p a_k e^{-j2\pi f k T_e}}$$

if $x(n)$ is a wide-sense stationary random sequence with a power spectral density $S_x(f)$, then

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

III.2.1. Yule-Walker method

we use the minimum mean-square error criterion to estimate these coefficients based on the observed $x(n)$. Let $\hat{x}(n)$ be an approximation of $x(n)$ by using a linear combination of its past p values as given by

$$\hat{x}(n) = \sum_{k=1}^p a_k x(n-k)$$

The estimation error is given by

$$e(n) = x(n) - \hat{x}(n) = x(n) - \sum_{k=1}^p a_k x(n-k)$$

$$x(n) - \sum_{k=1}^p a_k x(n-k) = e(n)$$

The mean-square error is given by

$$E[|e(n)|^2] = E \left[\left| x(n) - \sum_{k=1}^p a_k x(n-k) \right|^2 \right]$$

Let us use the orthogonal principle method in minimizing the mean-square estimation.

$$\begin{aligned} \min E[|e(n)|^2] &= \min_{a_k \text{ for } k=1, \dots, M} E \left[\left| x(n) - \sum_{k=1}^p a_k x(n-k) \right|^2 \right] \\ \min_{\{a_1, \dots, a_M\}} E \left[\left| x(n) - \sum_{k=1}^p a_k x(n-k) \right|^2 \right] &\Rightarrow \frac{\partial}{\partial \{a_1, \dots, a_M\}} E \left[\left| x(n) - \sum_{k=1}^p a_k x(n-k) \right|^2 \right] = 0 \\ E \left[\left| x(n) - \sum_{k=1}^p a_k x(n-k) \right|^2 \right] &= E[|x(n) - \mathbf{a}\mathbf{x}(n)|^2] \end{aligned}$$

where : $\mathbf{a} = [a_1 \ \dots \ a_p]$ and $\mathbf{x}(n) = [x(n-1) \ \dots \ x(n-p)]^T$.

$$E[|x(n) - \mathbf{a}\mathbf{x}(n)|^2] = E[x(n)x^*(n) - \mathbf{a}\mathbf{x}(n)x^*(n) - x(n)\mathbf{x}^H(n)\mathbf{a}^H + \mathbf{a}\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{a}^H]$$

$$\frac{\partial}{\partial \mathbf{a}} E[|x(n) - \mathbf{a}\mathbf{x}(n)|^2] = E[-2\mathbf{x}(n)x^*(n) + 2\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{a}^H]$$

$$= E[\mathbf{x}(n)\mathbf{x}^T(n)]\mathbf{a}^H = E[\mathbf{x}(n)x^*(n)]$$

$$\begin{bmatrix} R_x(0) & R_x(1) & \dots & R_x(p-1) \\ R_x(1) & R_x(0) & \ddots & R_x(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ R_x(p-1) & R_x(p-2) & \dots & R_x(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} R_x(1) \\ R_x(2) \\ \vdots \\ R_x(p) \end{bmatrix}$$

where :

$$\mathbf{R}_x \mathbf{a} = \mathbf{r}_x$$

Then the optimum $\hat{\mathbf{a}}$ are given by

$$\hat{\mathbf{a}} = \mathbf{R}_x^{-1} \mathbf{r}_x$$

Where the optimum $\hat{a}_k$ are given by

$$\sum_{k=1}^p \hat{a}_k R_x(i-k) = R_x(i) \quad i = 1, \dots, p$$

This equation is called the normal equation (also called the Yule–Walker equation) solution of the AR coefficients. The noise variance is found from

$$\hat{\sigma}_v^2 = R_x(0) + \sum_{k=1}^p a_k R_x(k)$$

The PSD estimate is obtained when $\hat{\mathbf{a}}$ and $\hat{\sigma}_v^2$ are substituted in $S_x(f)$. This approach for estimating the AR parameters is known in the literature as the autocorrelation method.

Example III.1: For $p = 1$. Then

$$\hat{a}_1 = \frac{R_x(1)}{R_x(0)}$$

$$\hat{\sigma}_v^2 = R_x(0) + \hat{a}_1 R_x(1)$$

For $p = 2$. Then $\hat{a}_1$ and $\hat{a}_2$ are solved from

$$\begin{bmatrix} R_x(0) & R_x(1) \\ R_x(1) & R_x(0) \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \end{bmatrix} = \begin{bmatrix} R_x(1) \\ R_x(2) \end{bmatrix}$$

$$\hat{\sigma}_v^2 = R_x(0) + \hat{a}_1 R_x(1) + \hat{a}_2 R_x(2)$$

Example III.2:

Comparison of AR and FFT spectral estimations for complex exponential AR modeled data. Consider a 4th-order AR data sequence $x(n), n = 1, \dots, 512$, generated from

$$x(n) - \sum_{k=1}^4 a_k x(n-k) = v(n)$$

where the AR model coefficients are given by

$$\mathbf{a} = [a_1 \ a_2 \ a_3 \ a_4]^T = [-0.3760 \ -0.6487 \ -0.0966 \ -0.1764]^T$$

and the input sequence $v(n)$ is a zero-mean white sequence of unit variance.

where the complex exponential frequencies and amplitudes are taken as $f_1 = 0.33 = -f_3$, $f_2 = 0.21 = -f_4$

and as $A_1 = A_3 = 0.7$, $A_2 = A_4 = 0.5$ respectively.

Now, suppose we model the observed sequence $x(n)$ of length 512 as a 4th-order AR sequence. The time-averaged 4×4 correlation matrix $\mathbf{R}_x$ is given by

$$\mathbf{R}_x = \begin{bmatrix} 1.5960 & -0.3649 & -0.7606 & 0.3773 \\ -0.3649 & 1.5960 & -0.3649 & -0.7606 \\ -0.7606 & -0.3649 & 1.5960 & -0.3649 \\ 0.3773 & -0.7606 & -0.3649 & 1.5960 \end{bmatrix}$$

and the time-averaged correlation vector $\mathbf{r}_x$ is given by

$$\mathbf{r}_x = [-0.3649 \ -0.7606 \ 0.3773 \ 0.1457]^T$$

The normal equation solution yields the estimated AR coefficients

$$\hat{\mathbf{a}} = \mathbf{R}_x^{-1} \mathbf{r}_x$$

where

$$\mathbf{R}_x^{-1} = \begin{bmatrix} 0.9685 & 0.3955 & 0.5726 & 0.0904 \\ 0.3955 & 1.1215 & 0.5758 & 0.5726 \\ 0.5726 & 0.5758 & 1.1215 & 0.3955 \\ 0.0904 & 0.5726 & 0.3955 & 0.9685 \end{bmatrix}$$

Then

$$\hat{\mathbf{a}} = [-0.4250 \quad -0.6966 \quad -0.1661 \quad -0.1782]^T$$

As expected, the roots of the denominator of the AR transfer function with the true coefficients $\mathbf{a}$ yield the frequencies of $f_1 = 0.33 = -f_3$, $f_2 = 0.21 = -f_4$, while the roots of the estimated coefficients $\hat{\mathbf{a}}$ yield the estimated frequencies of $\hat{f}_1 = 0.32 = -\hat{f}_3$, $\hat{f}_2 = 0.2215 = -\hat{f}_4$.

We observe that the estimated frequencies closely match the true frequencies. In the figure below, we present the true AR spectral density estimator.

$$S_x(f) = \frac{\sigma_v^2}{|1 - a_1 e^{-j2\pi f} - a_2 e^{-j4\pi f} - a_3 e^{-j6\pi f} - a_4 e^{-j8\pi f}|^2}, \quad 0 \leq f < 1$$

As well as the estimated AR spectral density

$$\hat{S}_x(f) = \frac{\sigma_v^2}{|1 - \hat{a}_1 e^{-j2\pi \hat{f}} - \hat{a}_2 e^{-j4\pi \hat{f}} - \hat{a}_3 e^{-j6\pi \hat{f}} - \hat{a}_4 e^{-j8\pi \hat{f}}|^2}, \quad 0 \leq f < 1$$

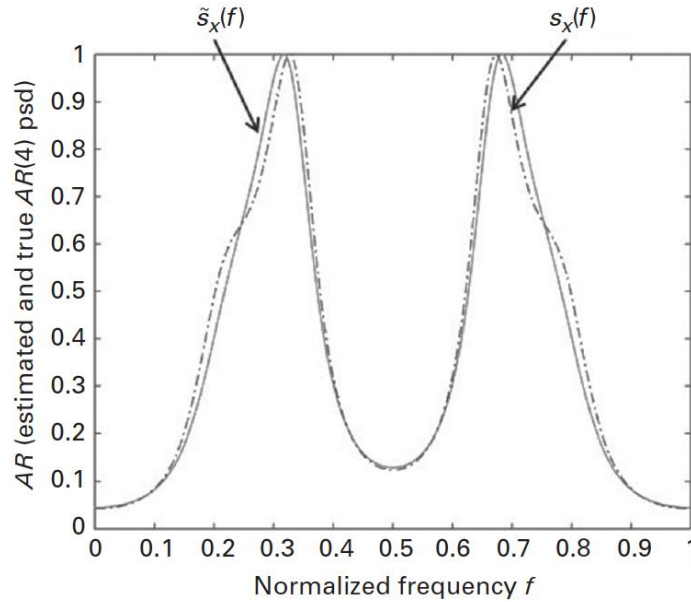


Figure III.1 Fourth-order AR and estimated AR spectral densities.

We observe the estimated AR spectral density approximates closely the true AR spectral density. In the second figure below, we plot the true AR spectral density $S_x(f)$ as well as the absolute value of the FFT of the zero padded observed data, given by

$$\hat{S}_x(k/2048) = \text{abs}\left(\text{fft}(x, \text{zeros}(1536, 1))\right), \quad k = 0, \dots, 2047$$

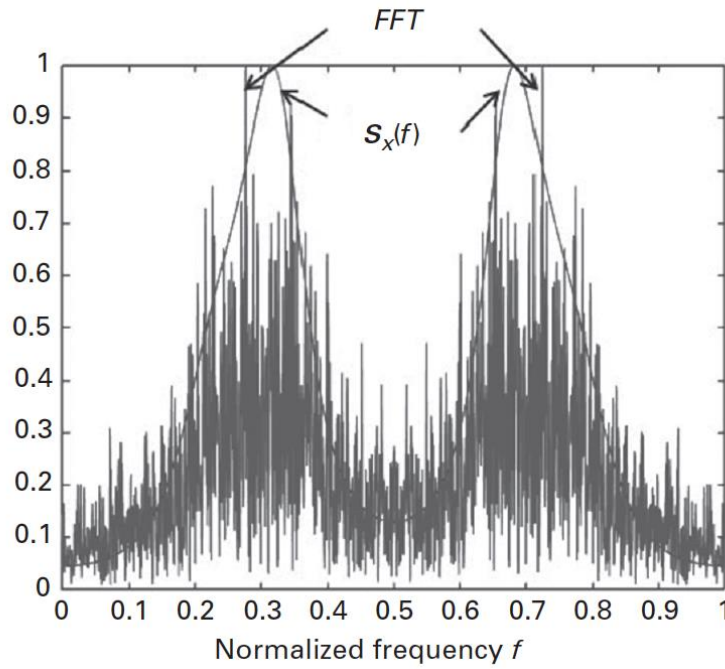


Figure III.2 Fourth-order AR spectral density compared to FFT magnitude spectrum.

where $\mathbf{x} = [x_1 \ \dots \ x_{512}]^T$ is the 512 point observed data vector. The padding of 1,536 number of zeros in the FFT does not improve its resolution, but only yields additional observation values in the frequency domain. As can be seen, the FFT estimated spectral density has great variations and appears to be less desirable than the estimated AR spectral density as compared to the true AR spectral density. Furthermore, while the AR model can yield quite good estimates of the frequencies of the complex exponentials as shown above, the FFT approach yields no information in this direction. For AR model generated data, the parametric AR spectral density approach, in particular when the order of the AR model is known or can be estimated correctly, generally yields a better approximation than the non-parametric FFT spectral density approach.

III.2.2. Levinson-Durbin algorithm

The Yule-Walker equations provide a relationship between the parameters of an AR model and its covariance coefficients. Solving these equations through matrix inversion is computationally expensive.

$$\hat{\mathbf{a}} = \mathbf{R}_x^{-1} \mathbf{r}_x$$

The Levinson-Durbin algorithm is an iterative algorithm that yields the same solution as the matrix inversion method, meaning it calculates the coefficients for rank " k " based on those from rank " $k - 1$ ". The algorithm follows a specific notation, where the order " k " is iterated from $k = 1, \dots, p$.

The Levinson procedure is as follows:

Input:

- Autocovariance sequence $R_x(k)$, $k = 0, 1, \dots, p$ for a given time series.

Output:

AR coefficients a_k , $k = 0, 1, \dots, p$ for an $AR(p)$ model.

Procedure:

Initialization:

- $m_1 = \frac{R_x(1)}{R_x(0)}$ (first reflection coefficient)
- $a_1 = m_1$ (first AR coefficient)
- $P_1 = R_x(0)[1 - |m_k|^2]$ (variance of the input sequence)

For $k = 2$ **to** p :

Update Reflection Coefficient:

- $m_k = \frac{R_x(k) + \sum_{i=1}^{k-1} a_i R_x(k-i)}{P_{k-1}}$

Update AR Coefficients:

- $a_k = m_k$ (k - th AR coefficient)

End For

For $i = 1$ **to** $k - 1$:

Update a_i using the Durbin recursion:

- $a_i(k) = a_i(k-1) + m_k a_{k-i}(k-1)$.

Update Prediction Error:

- $P_k = P_{k-1}[1 - |m_k|^2]$

End For

Output:

The AR coefficients a_k , $k = 0, 1, \dots, p$ are the desired parameters of the $AR(p)$ model.

Example III.3:

The autocorrelation matrix of the output from a second-order IIR system, with a zero-mean Gaussian noise input, is given by

$$\mathbf{R}_x = \begin{bmatrix} 5.5611 & 1.3291 & 0.8333 \\ 1.3291 & 5.5611 & 1.3291 \\ 0.8333 & 1.3291 & 5.5611 \end{bmatrix}$$

- $m_1 = \frac{1.3291}{5.5611} = 0.239$ (first reflection coefficient)
- $a_1 = 0.239$ (first AR coefficient)
- $P_1 = 5.5611[1 - |0.239|^2] = 5.2434$ (variance of the input sequence)

For $k = 2$:

- $m_2 = \frac{R_x(2) + a_1 R_x(1)}{P_1} = \frac{0.8333 + 0.239 \times 1.3291}{5.2434} = 0.2195$
- $a_2 = m_2 = 0.2195$

End For

For $i = 1$

- $a_1(2) = a_1(1) + m_2 a_1(1) = 0.239 + 0.2195 \times 0.239 = 0.2915$
- $P_2 = P_1[1 - |m_2|^2] = 5.2434 [1 - |0.2195|^2] = 4.991$

The resulting AR coefficients for the AR(2) model are $a_1(2) \approx 0.2915$ and $a_2 \approx 0.2195$.

III.2.3. Pisarenko Harmonic Decomposition Method

Let the observed data represent p complex sinusoids in noise, i.e.,

$$x(n) = \sum_{k=1}^p A_k e^{j2\pi f_k n} + v(n)$$

The PSD of the process is a sum of the continuous spectrum of the noise and a set of impulses with area $|A_k|^2$ at the frequencies f_k , or

$$S_x(f) = \sum_{k=1}^p |A_k|^2 \delta(f - f_k) + S_v(f)$$

Pisarenko studied the model of $x(n)$ shown above and found that the frequencies of the sinusoids can be obtained from the eigenvector corresponding to the smallest eigenvalue of the autocorrelation matrix. His method, known as Pisarenko harmonic decomposition (PHD), led to important insights and stimulated further work which resulted in many new procedures known today as “signal and noise subspace” methods.

When the noise $v(n)$ is zero mean white with variance σ_v^2 , the autocorrelation of $x(n)$ can be written as

$$R_x(n) = \sum_{k=1}^p |A_k|^2 e^{j2\pi f_k n} + \sigma_v^2 \delta(n)$$

The autocorrelation matrix can be represented by

$$\mathbf{R}_x = \sum_{k=1}^p |A_k|^2 \mathbf{e}_k \mathbf{e}_k^H + \sigma_v^2 \mathbf{I} = \mathbf{E} \mathbf{A} \mathbf{E}^H + \sigma_v^2 \mathbf{I} = \mathbf{R}_s + \sigma_v^2 \mathbf{I}$$

where: $\mathbf{e}_k = [1 \quad e^{j2\pi f_k} \quad \dots \quad e^{j2\pi f_k(M-1)}]^T$, $\mathbf{E} = [\mathbf{e}_1 \quad \mathbf{e}_2 \quad \dots \quad \mathbf{e}_p]$ and $\mathbf{I}$ is the identity matrix.
 $\mathbf{A} = \text{diag}(|A_1|^2 \quad |A_2|^2 \quad \dots \quad |A_p|^2)$.

The eigenvalue decomposition (EVD) of $\mathbf{R}_s$ can thus be written as

$$\mathbf{R}_s = \sum_{k=1}^p \lambda_k \mathbf{u}_k \mathbf{u}_k^H + 0 \sum_{k=p+1}^M \mathbf{u}_k \mathbf{u}_k^H$$

Where:

- The $\lambda_k, k = 1, 2, \dots, p$, are the nonzero eigenvalues of $\mathbf{R}_s$.
- Let the eigenvalues of $\mathbf{R}_s$ be arranged in decreasing order so that $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_M$ where $\mathbf{u}_k$ be the eigenvector corresponding to λ_k .
- The space spanned by the eigenvectors $\mathbf{u}_k, k = 1, 2, \dots, p$, is called the signal subspace.
- The space spanned by $\mathbf{u}_k, k = p+1, p+2, \dots, M$, is the noise subspace.

Since the set of eigenvectors are orthonormal, that is

$$\mathbf{u}_k^H \mathbf{u}_m = \begin{cases} 1 & \text{if } k = m \\ 0 & \text{if } k \neq m \end{cases}$$

the two subspaces are orthogonal. In other words, if $\mathbf{s}$ is in the signal subspace, and $\mathbf{z}$ is in the noise subspace, then $\mathbf{s}^H \mathbf{z} = \mathbf{0}$.

The eigenvalue decomposition (EVD) of $\mathbf{R}_x$ follows from that of $\mathbf{R}_s$:

$$\begin{aligned} \mathbf{R}_x &= \sum_{k=1}^p \lambda_k \mathbf{u}_k \mathbf{u}_k^H + 0 \sum_{k=p+1}^M \mathbf{u}_k \mathbf{u}_k^H + \sigma_v^2 \mathbf{I} \\ &= \sum_{k=1}^p \lambda_k \mathbf{u}_k \mathbf{u}_k^H + 0 \sum_{k=p+1}^M \mathbf{u}_k \mathbf{u}_k^H + \sigma_v^2 \left(\sum_{k=1}^p \mathbf{u}_k \mathbf{u}_k^H + \sum_{k=p+1}^M \mathbf{u}_k \mathbf{u}_k^H \right) \\ &= \sum_{k=1}^p (\lambda_k + \sigma_v^2) \mathbf{u}_k \mathbf{u}_k^H + \sigma_v^2 \sum_{k=p+1}^M \mathbf{u}_k \mathbf{u}_k^H \end{aligned}$$

Now suppose that the matrix $\mathbf{R}_x$ is $(p+1) \times (p+1)$. Pisarenko observed that the noise variance corresponds to the smallest eigenvalue of $\mathbf{R}_x$ and that the frequencies of the complex sinusoids can be estimated by using the orthogonality of the signal and noise subspaces, that is,

$$\mathbf{e}_k^H \mathbf{u}_{p+1} = 0 \quad \text{where } k = 1, 2, \dots, p$$

We can estimate the f_k 's by forming the pseudospectrum

$$\hat{S}_{\text{PHD}}(f) = \frac{1}{|\mathbf{e}^H(f) \mathbf{u}_{p+1}|^2}$$

The PSD estimate in $\hat{S}_{\text{PHD}}(f)$ does not include information about the power of the noise and the complex sinusoids. The powers, however, can easily be obtained by using $S_x(f)$. First note that $S_v(f) = \sigma_v^2$, and $\hat{\sigma}_v^2 = \lambda_{p+1}$. Second, the frequencies f_k are determined from the pseudospectrum $\hat{S}_{\text{PHD}}(f)$, so it remains to find the powers of the complex sinusoids $\mathbf{A}_k = |A_k|^2$. This can readily be accomplished by using the set of p linear equations

$$\begin{bmatrix} |\hat{\mathbf{e}}_1^H \mathbf{u}_1|^2 & |\hat{\mathbf{e}}_2^H \mathbf{u}_1|^2 & \dots & |\hat{\mathbf{e}}_p^H \mathbf{u}_1|^2 \\ |\hat{\mathbf{e}}_1^H \mathbf{u}_2|^2 & |\hat{\mathbf{e}}_2^H \mathbf{u}_2|^2 & \ddots & |\hat{\mathbf{e}}_p^H \mathbf{u}_2|^2 \\ \vdots & \vdots & \ddots & \vdots \\ |\hat{\mathbf{e}}_1^H \mathbf{u}_p|^2 & |\hat{\mathbf{e}}_2^H \mathbf{u}_p|^2 & \dots & |\hat{\mathbf{e}}_p^H \mathbf{u}_p|^2 \end{bmatrix} \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_p \end{bmatrix} = \begin{bmatrix} \lambda_1 - \hat{\sigma}_v^2 \\ \lambda_2 - \hat{\sigma}_v^2 \\ \vdots \\ \lambda_p - \hat{\sigma}_v^2 \end{bmatrix}$$

$$\mathbf{U}\mathbf{\Lambda} = \boldsymbol{\psi}$$

$$\mathbf{\Lambda} = \mathbf{U}^{-1}\boldsymbol{\psi}$$

where

$$\hat{\mathbf{e}}_k = [1 \quad e^{j2\pi\hat{f}_k} \quad \dots \quad e^{j2\pi\hat{f}_k(p-1)}]^T$$

In summary, Pisarenko's method consists of four steps:

1. Estimate the $(p + 1) \times (p + 1)$ autocorrelation matrix $\mathbf{R}_x$ (provided it is known that the number of complex sinusoids is p).
2. Evaluate the minimum eigenvalue λ_{p+1} and the eigenvectors of $\mathbf{R}_x$.
3. Set the white noise power to $\hat{\sigma}_v^2 = \lambda_{p+1}$, estimate the frequencies of the complex sinusoids from the peak locations of $\hat{S}_{\text{PHD}}(f)$, and compute their powers $\mathbf{\Lambda}$.
4. Substitute the estimated parameters in $S_x(f)$.

Pisarenko's method is not used frequently in practice because its performance is much poorer than the performance of some other signal and noise subspace-based methods developed later.

III.2.4. Multiple Signal Classification (MUSIC)

A procedure very similar to Pisarenko's is the Multiple Signal Classification (MUSIC) method, which was proposed in the late 1970's by Schmidt. Suppose again that the process $x(n)$ is described by p complex sinusoids in white noise. If we form an $M \times M$ autocorrelation matrix $\mathbf{R}_x$, find its eigenvalues and eigenvectors and rank them as before, then as mentioned in the previous subsection, its p eigenvectors corresponding to the p largest eigenvalues span the signal subspace, and the remaining eigenvectors, the noise subspace. According to MUSIC, we estimate the noise variance from the $M - p$ smallest eigenvalues of $\mathbf{R}_x$.

$$\hat{\sigma}_v^2 = \frac{1}{M - p} \sum_{k=p+1}^M \lambda_k$$

and the frequencies from the peak locations of the pseudospectrum

$$\hat{S}_{\text{MUSIC}}(f) = \frac{1}{\sum_{k=p+1}^M |\mathbf{e}^H(f) \mathbf{u}_k|^2}$$

It should be noted that there are other ways of estimating the f_k 's. Finally the powers of the complex sinusoids are determined from $\mathbf{\Lambda}$, and all the estimated parameters substituted in :

$$S_x(f) = \sum_{k=1}^p |A_k|^2 \delta(f - f_k) + S_v(f)$$

MUSIC has better performance than Pisarenko's method because of the introduced averaging via the extra noise eigenvectors.

III.3. Spectrum Estimation Based on ARMA Models

The PSD of a process that is represented by the ARMA model is given by

$$S_{ARMA}(f) = \sigma_v^2 \frac{|\sum_{n=0}^{q-1} b_n e^{-j2\pi f n}|^2}{|1 + \sum_{n=1}^{p-1} a_n e^{-j2\pi f n}|^2}$$

The ML estimates of the ARMA coefficients are difficult to obtain, so we usually resort to methods that yield suboptimal estimates. For example, we can first estimate the AR coefficients based on the equation,

$$\begin{bmatrix} R_x(q) & R_x(q-1) & \dots & R_x(q-p+1) \\ R_x(q+1) & R_x(q) & \ddots & R_x(q-p+2) \\ \vdots & \vdots & \ddots & \vdots \\ R_x(q+p-1) & R_x(q+p-2) & \dots & R_x(q) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} R_x(q+1) \\ R_x(q+2) \\ \vdots \\ R_x(q+p) \end{bmatrix}$$

We can find the least squares estimates of $\mathbf{a}$ by

$$\hat{\mathbf{a}} = \mathbf{R}_x^\dagger \mathbf{r}_x$$

This procedure is known as the least-squares modified Yule-Walker equation method. Once the AR coefficients are estimated, we can filter the observed data

$$y(n) = x(n) + \sum_{k=1}^p \hat{a}_k x(n-k)$$

and obtain a sequence that is approximately modeled by an $MA(q)$ model. From the data $y(n)$ we can estimate the MA PSD by

$$S_{MA}(f) = \sum_{n=-q}^q R_y(k) e^{-j2\pi f n}$$

and obtain the PSD estimate of the data $x(n)$

$$S_{ARMA}(f) = \sigma_v^2 \frac{S_{MA}(f)}{|1 + \sum_{n=1}^{p-1} \hat{a}_n e^{-j2\pi f n}|^2}$$

The ARMA model has an advantage over the AR and MA models because it can better fit spectra with nulls and peaks. Its disadvantage is that it is more difficult to estimate its parameters than the parameters of the AR and MA models.

III.4. Adaptive filtering

Adaptive filtering is a technique used in signal processing to adjust the characteristics of a filter in response to changing signal conditions. This adaptability is achieved by modifying the filter's coefficients based on a predefined criterion as soon as new signal data becomes available. These modifications should track the changes in the systems within their environment as quickly as possible. Adaptive filtering is typically associated with real-time operation.

Two common adaptive filtering algorithms are the Least Mean Squares (LMS) algorithm and the Recursive Least Squares (RLS) algorithm. These algorithms are widely used in various applications like noise cancellation, equalization, prediction, and more.

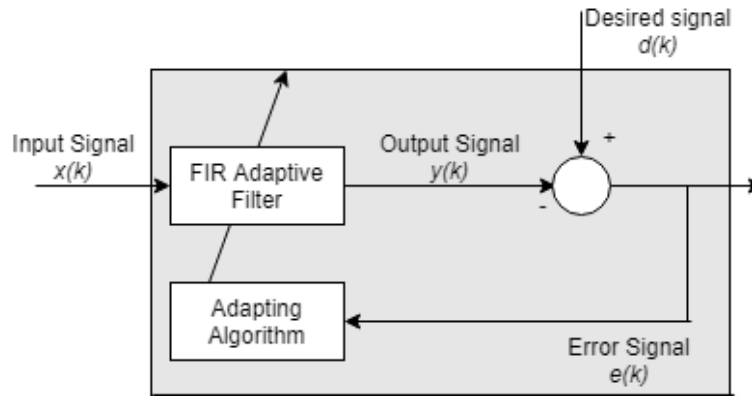


Figure III.3 Adaptive filtering principle.

III.4.1. Least Mean Squares (LMS) Algorithm

The Least Mean Squares (LMS) algorithm, also known as stochastic gradient descent, provides two advantages. Firstly, it avoids the need to know the exact statistics of the signal (e.g., inter-correlation and auto-correlation), which are rarely available in practice. Secondly, this method has a tracking mechanism that allows it to follow variations in the signal statistics.

Let's consider a FIR filter with the impulse response $\{h(i), 0 \leq i \leq M - 1\}$. Let $x(k)$ be the input sequence of the filter, and $y(k)$ be the corresponding output, where:

$$y(k) = \sum_{i=0}^{M-1} h(i)x(k-i) = \mathbf{h}^T \mathbf{x}(k) = \mathbf{x}^T(k) \mathbf{h}$$

where :

$$\mathbf{h} = [h(0), h(1), h(2), \dots, h(M-1)]^T$$

$$\mathbf{x}(k) = [x(k), x(k-1), \dots, x(k-M+1)]^T$$

Suppose we also have a desired sequence $d(k)$ with which we can compare the filter's output. Then,

we can form the error sequence $e(k)$ by taking the difference between $d(k)$ and $y(k)$, as follows:

$$e(k) = d(k) - y(k), \quad k = 0, \dots, N$$

The coefficients of the FIR filter will be selected to minimize the mean squared error. Therefore, we have:

$$\begin{aligned} j(\mathbf{h}) &= E[e^2(k)] = E\left[(d(k) - \mathbf{h}^T \mathbf{x}(k))^2\right] \\ &= E[d^2(k)] - 2\mathbf{h}^T E[d(k)\mathbf{x}(k)] + \mathbf{h}^T E[\mathbf{x}(k)\mathbf{x}^T(k)]\mathbf{h} \\ &= E[d^2(k)] - 2\mathbf{h}^T \mathbf{r}_{dx} + \mathbf{h}^T \mathbf{R}_x \mathbf{h} \end{aligned}$$

where

$$\mathbf{r}_{dx} = E[d(k)\mathbf{x}(k)]$$

$$\mathbf{R}_x = E[\mathbf{x}(k)\mathbf{x}^T(k)]$$

Taking the derivative of the function $j(\mathbf{h})$ with respect to $\mathbf{h}^T$, we obtain:

$$\frac{\partial j(\mathbf{h})}{\partial \mathbf{h}^T} = 0 \quad 0 \leq i \leq M-1$$

Therefore,

$$\mathbf{r}_{dx} = \mathbf{R}_x \mathbf{h}$$

This is a linear equation that produces the optimal filter coefficients $\mathbf{h}$. To directly solve the linear equation, we need to compute the auto-correlation sequence $\mathbf{R}_x$ of the input signal and the cross-correlation sequence $\mathbf{r}_{dx}$ between the desired sequence $d(k)$ and the input sequence $x(k)$.

The LMS algorithm provides an iterative method to determine the optimal filter coefficients $\mathbf{h}$ without computing the correlation sequences $\mathbf{R}_x$ and $\mathbf{r}_{dx}$. The algorithm is a recursive gradient method (steepest-descent) that finds the minimum of $j(\mathbf{h})$, thus yielding the set of optimal filter coefficients $\mathbf{h}$.

We begin by choosing initial values for $\mathbf{h}$, denoted as $\mathbf{h}_0$. For example, we can start with $\{\mathbf{h}_0 = \mathbf{0}_{M \times 1}\}$. Then, after each new input sequence $x(k)$ enters the adaptive FIR filter, we compute the corresponding output $y(k)$, form the error signal $e(k) = d(k) - y(k)$, and update the filter coefficients according to the equation:

$$\mathbf{h}_{k+1} = \mathbf{h}_k + \mu \mathbf{x}(k) e(k), \quad k = 0, 1, \dots$$

Where μ is called the *step size* and controls the convergence rate of the algorithm towards the optimal solution. A large value of μ leads to large adjustments and hence quick convergence, while a small value of μ results in slower convergence. However, for a very large value of μ , the algorithm becomes

unstable, and to ensure stability, μ should be chosen such that:

$$0 < \mu < \frac{1}{10MP_x}$$

where M is the length of the adaptive FIR filter, and P_x is the input signal power.

The algorithm can then be expressed in the following procedure:

1. *Initial Values:*

a. $\mathbf{h}_0 = \mathbf{0}_{M \times 1}$

2. **For** $k = 0, 1, \dots$, Total number of iterations $- 1$ **repeat:**

b. $y(k) = \mathbf{h}_n^T \mathbf{x}(k)$

c. $e(k) = d(k) - y(k)$

d. $\mathbf{h}_{k+1} = \mathbf{h}_k + \mu \mathbf{x}(k) e(k)$

Example III.4:

Let's consider a simple case where we have an FIR filter with three coefficients $M = 3$ and a desired sequence $d(k)$ that we want to approximate using the LMS algorithm.

1. *Initialization:*

- Set $M = 3$ (number of filter coefficients)
- Set $\mu = 0.01$ (step size)
- Set $\mathbf{h}_0 = [0 \ 0 \ 0]^T$ (initial filter coefficients)

2. *Input Sequence:* Let's assume we have an input sequence $x(k)$ as follows:

$$x(k) = [1 \ 0.5 \ -0.2] \text{ for } k = 0, 1, 2, \dots$$

3. *Desired Sequence:* Let's assume a desired sequence $d(k)$ that we want to approximate:

$$d(k) = [1 \ 0 \ -0.5] \text{ for } k = 0, 1, 2, \dots$$

4. *Applying LMS Algorithm:*

For each k :

- Compute the filter output $y(k) = \mathbf{h}_n^T \mathbf{x}(k)$
- Compute the error signal $e(k) = d(k) - y(k)$
- Update filter coefficients using LMS rule: $\mathbf{h}_{k+1} = \mathbf{h}_k + \mu \mathbf{x}(k) e(k)$

Repeat this for a certain number of iterations or until convergence.

III.4.2. Recursive Least Squares (RLS) Algorithm

Unlike LMS which aims to minimize the mean squared error, the Recursive Least Squares (RLS) algorithm recursively finds filter coefficients that minimize the least squares cost function. While RLS offers faster convergence compared to LMS, it comes with higher computational complexity.

The cost function for this algorithm is given by:

$$j(n) = \sum_{k=0}^n \lambda^{n-k} e^2(n) = \sum_{k=0}^n \lambda^{n-k} (d(k) - \mathbf{h}_n^T \mathbf{x}(k))^2$$

Where:

$$\mathbf{h}_n = [h_n(0), h_n(1), h_n(2), \dots, h_n(M-1)]^T$$

$$\mathbf{x}(k) = [x(k), x(k-1), x(k-2), \dots, x(k-M+1)]^T$$

$\mathbf{h}_n$ is the FIR filter at time n and $0 < \lambda < 1$ is called the "forgetting factor" as it gives less weight to samples from the distant past.

The problem is to determine the coefficient vector $\mathbf{h}_n$ that minimizes $j(n)$.

$$\begin{aligned} j(n) &= \sum_{k=0}^n \lambda^{n-k} (d(k) - \mathbf{h}_n^T \mathbf{x}(k))^2 \\ &= \sum_{k=0}^n \lambda^{n-k} d^2(k) - 2\mathbf{h}_n^T \sum_{k=0}^n \lambda^{n-k} \mathbf{x}(k) d(k) + \sum_{k=0}^n \lambda^{n-k} \mathbf{h}_n^T \mathbf{x}(k) \mathbf{x}^T(k) \mathbf{h}_n \\ &= \sum_{k=0}^n \lambda^{n-k} d^2(k) - 2\mathbf{h}_n^T \bar{\mathbf{r}}_{dx}(n) + \mathbf{h}_n^T \bar{\mathbf{R}}_x(n) \mathbf{h}_n \end{aligned}$$

Where :

$$\bar{\mathbf{r}}_{dx}(n) = \sum_{k=0}^n \lambda^{n-k} \mathbf{x}(k) d(k)$$

$$\bar{\mathbf{R}}_x(n) = \sum_{k=0}^n \lambda^{n-k} \mathbf{x}(k) \mathbf{x}^T(k)$$

Taking the derivative of $j(n)$ with respect to $\mathbf{h}_n^T$:

$$\frac{\partial j(n)}{\partial \mathbf{h}_n^T} = 0$$

Therefore:

$$\bar{\mathbf{r}}_{dx}(n) = \bar{\mathbf{R}}_x(n) \mathbf{h}_n$$

$$\mathbf{h}_n = \bar{\mathbf{R}}_x^{-1}(n) \bar{\mathbf{r}}_{dx}(n)$$

$\bar{\mathbf{R}}_x(n)$ and $\bar{\mathbf{r}}_{dx}(n)$ can be computed recursively:

$$\bar{\mathbf{R}}_x(n) = \lambda \bar{\mathbf{R}}_x(n-1) + \mathbf{x}(n) \mathbf{x}^T(n)$$

$$\bar{\mathbf{r}}_{dx}(n) = \lambda \bar{\mathbf{r}}_{dx}(n-1) + \mathbf{x}(n) d(n)$$

Using the matrix inversion lemma:

Let $\mathbf{A}$ and $\mathbf{B}$ be two positive definite matrices of size $M \times M$ satisfying :

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^T$$

Where $\mathbf{D}$ is an invertible $N \times N$ matrix and $\mathbf{C}$ is an $M \times N$ matrix. Then:

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}(\mathbf{D} + \mathbf{C}^T\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{B}$$

We can use this lemma to compute $\bar{\mathbf{R}}_x^{-1}(n)$ by setting:

$$\mathbf{A} = \bar{\mathbf{R}}_x(n)$$

$$\mathbf{B}^{-1} = \lambda \bar{\mathbf{R}}_x(n-1)$$

$$\mathbf{C} = \mathbf{x}(n)$$

$$\mathbf{D} = 1$$

This gives us the recursive equation for the inverse of the autocorrelation matrix:

$$\bar{\mathbf{R}}_x^{-1}(n) = \lambda^{-1} \bar{\mathbf{R}}_x^{-1}(n-1) - \frac{\lambda^{-2} \bar{\mathbf{R}}_x^{-1}(n-1) \mathbf{x}(n) \mathbf{x}^T(n) \bar{\mathbf{R}}_x^{-1}(n-1)}{1 + \lambda^{-1} \mathbf{x}(n) \bar{\mathbf{R}}_x^{-1}(n-1) \mathbf{x}^T(n)}$$

where :

$$\mathbf{Q}_x(n) = \bar{\mathbf{R}}_x^{-1}(n)$$

$$\mathbf{K}_x(n) = \frac{\lambda^{-1} \mathbf{Q}_x(n-1) \mathbf{x}(n)}{1 + \lambda^{-1} \mathbf{x}(n) \mathbf{Q}_x(n-1) \mathbf{x}^T(n)}$$

Then the recursive equation for $\mathbf{Q}_x(n)$ can be written as:

$$\mathbf{Q}_x(n) = \lambda^{-1} \mathbf{Q}_x(n-1) - \lambda^{-1} \mathbf{K}_x(n) \mathbf{x}^T(n) \mathbf{Q}_x(n-1)$$

This equation is known as the Riccati equation. The vector $\mathbf{K}_x(n)$ is called the Kalman gain, which can be rearranged as:

$$\mathbf{K}_x(n) = \mathbf{Q}_x(n) \mathbf{x}(n) = \bar{\mathbf{R}}_x^{-1}(n) \mathbf{x}(n)$$

Now, we need to calculate the filter coefficients recursively. We have:

$$\begin{aligned} \mathbf{h}_n &= \bar{\mathbf{R}}_x^{-1}(n) \bar{\mathbf{r}}_{dx}(n) = \mathbf{Q}_x(n) \bar{\mathbf{r}}_{dx}(n) \\ &= \lambda \mathbf{Q}_x(n) \bar{\mathbf{r}}_{dx}(n-1) + \mathbf{Q}_x(n) \mathbf{x}(n) d(n) \end{aligned}$$

By replacing $\mathbf{Q}_x(n)$ in the first term of the previous expression with the Riccati equation, we obtain:

$$\begin{aligned} \mathbf{h}_n &= \lambda (\lambda^{-1} \mathbf{Q}_x(n-1) - \lambda^{-1} \mathbf{K}_x(n) \mathbf{x}^T(n) \mathbf{Q}_x(n-1)) \bar{\mathbf{r}}_{dx}(n-1) + \mathbf{Q}_x(n) \mathbf{x}(n) d(n) \\ &= \mathbf{Q}_x(n-1) \bar{\mathbf{r}}_{dx}(n-1) - \mathbf{K}_x(n) \mathbf{x}^T(n) \mathbf{Q}_x(n-1) \bar{\mathbf{r}}_{dx}(n-1) + \mathbf{Q}_x(n) \mathbf{x}(n) d(n) \\ \mathbf{h}_n &= \mathbf{h}_{n-1} - \mathbf{K}_x(n) \mathbf{x}^T(n) \mathbf{h}_{n-1} + \mathbf{K}_x(n) d(n) \\ \mathbf{h}_n &= \mathbf{h}_{n-1} + \mathbf{K}_x(n) (d(n) - \mathbf{x}^T(n) \mathbf{h}_{n-1}) \end{aligned}$$

Finally, we have:

$$\mathbf{h}_n = \mathbf{h}_{n-1} + \mathbf{K}_x(n) e(n)$$

Where:

$e(n) = d(n) - \mathbf{x}^T(n)\mathbf{h}_{n-1}$ is the a priori error, which is different from the posteriori error:

$$e(n) = d(n) - \mathbf{x}^T(n)\mathbf{h}_n$$

The algorithm can then be expressed in the following procedure:

1. Initial Values :

- $\mathbf{h}_0 = \mathbf{0}_{M \times 1}$
- $\mathbf{Q}_x(n) = \delta^{-1} \mathbf{I}_{M \times M} : \delta = \text{constante positive.}$

2. For $n = 1, 2, \dots$, Total number of iterations – 1 repeat:

$$\mathbf{K}_x(n) = \frac{\lambda^{-1} \mathbf{Q}_x(n-1) \mathbf{x}(n)}{1 + \lambda^{-1} \mathbf{x}(n) \mathbf{Q}_x(n-1) \mathbf{x}^T(n)}$$

$$e(n) = d(n) - \mathbf{x}^T(n) \mathbf{h}_{n-1}$$

$$\mathbf{h}_n = \mathbf{h}_{n-1} + \mathbf{K}_x(n) e(n)$$

$$\mathbf{Q}_x(n) = \lambda^{-1} \mathbf{Q}_x(n-1) - \lambda^{-1} \mathbf{K}_x(n) \mathbf{x}^T(n) \mathbf{Q}_x(n-1)$$

Example III.5: Let's consider a simple case where we have an FIR filter with three coefficients $M = 3$ and a desired sequence $d(n)$ that we want to approximate using the RLS algorithm.

1. Initialization:

- Set $M = 3$ (number of filter coefficients)
- Set $\lambda = 0.98$ (forgetting factor)
- Set $\mathbf{h}_0 = [0 \ 0 \ 0]^T$ (initial filter coefficients)
- Set $\delta = 0.1$ (small positive constant)
- Set $\mathbf{Q}_x(0) = 10\mathbf{I}$ (initial inverse correlation matrix)

2. Input Sequence: Let's assume we have an input sequence $\mathbf{x}(n)$ as follows: $\mathbf{x}(n) = [1 \ 0.5 \ -0.2]$ for $n = 0, 1, 2, \dots$

3. Desired Sequence: Let's assume a desired sequence $d(n)$ that we want to approximate: $d(n) = [1 \ 0 \ -0.5]$ for $n = 0, 1, 2, \dots$

4. Applying RLS Algorithm: For each n :

- Compute the Kalman gain $\mathbf{K}_x(n)$
- Calculate the a priori error $e(n)$
- Update filter coefficients $\mathbf{h}_n$
- Update inverse correlation matrix $\mathbf{Q}_x(n)$

MATLAB Example III.1: RLS Algorithm MATLAB Implementation

```
% RLS Algorithm Implementation
clear all;
close all;

% Initialize parameters
M = 3;           % filter length
lambda = 0.98;   % forgetting factor
delta = 0.1;     % small positive constant
iterations = 3;  % number of iterations

% Initialize vectors and matrices
h = zeros(M, 1); % initial filter coefficients
Q = (1/delta) * eye(M); % initial inverse correlation matrix

% Input and desired sequences
x = [1; 0.5; -0.2]; % input vector
d = [1 0 -0.5];     % desired sequence

% Arrays to store results
h_history = zeros(M, iterations+1);
h_history(:,1) = h;

% Main RLS loop
for n = 1:iterations
    % 1. Compute Kalman gain
    % K(n) = [lambda^(-1)Q(n-1)x(n)] / [1 + lambda^(-1)x^T(n)Q(n-1)x(n)]
    K = (Q * x) / (lambda + x' * Q * x);

    % 2. Calculate a priori error
    % e(n) = d(n) - x^T(n)h(n-1)
    e = d(n) - x' * h;

    % 3. Update filter coefficients
    % h(n) = h(n-1) + K(n)e(n)
    h = h + K * e;

    % 4. Update inverse correlation matrix
    % Q(n) = lambda^(-1)Q(n-1) - lambda^(-1)K(n)x^T(n)Q(n-1)
    Q = (1/lambda) * Q - (1/lambda) * K * x' * Q;

    % Store filter coefficients history
    h_history(:,n+1) = h;

    % Display results for current iteration
    fprintf('\nIteration %d:\n', n)
    fprintf('Kalman gain K_ %d:\n', n)
    disp(K)
    fprintf('A priori error e_ %d: %f\n', n, e)
    fprintf('Updated filter coefficients h_ %d:\n', n)
    disp(h)
    fprintf('Updated inverse correlation matrix Q_ %d:\n', n)
    disp(Q)
end
```

Example III.6: Consider a first-order autoregressive process AR^1 , solution to the equation:

$$x(n) + a_1x(n-1) = w(n).$$

- 1) Write the Yule-Walker equations.
- 2) Derive the expressions for the covariances as a function of a_1 and the noise variance σ^2 .
- 3) Derive the expressions for a_1 and σ^2 as functions of the covariances.

Solution:

1) Yule-Walker Equations

The generating filter is of first order, so the Yule-Walker equations are written for $p = 1$:

$$\begin{cases} R(0) + a_1 R(1) = \sigma^2 \\ R(1) + a_1 R(0) = 0 \end{cases}$$

For $k > 1$: $R(k) = -a_1 R(k-1)$

This recursive relationship shows that:

- All higher-order covariances can be computed from $R(0)$ and $R(1)$
- The covariances decay exponentially for a stationary process

2) Covariances as Functions of a_1 and σ^2

Using the Yule-Walker equations where $R(1) = -a_1 R(0)$, we find:

$$R(0) = \sigma^2 / (1 - a_1^2) \text{ and therefore: } R(1) = (-a_1 \sigma^2) / (1 - a_1^2)$$

✓ Detailed Derivation:

From the second Yule-Walker equation: $R(1) = -a_1 R(0)$

Substituting into the first equation: $R(0) + a_1(-a_1 R(0)) = \sigma^2$
 $R(0) - a_1^2 R(0) = \sigma^2$
 $R(0)(1 - a_1^2) = \sigma^2$
 $R(0) = \sigma^2 / (1 - a_1^2)$

Then: $R(1) = -a_1 R(0) = -a_1 \sigma^2 / (1 - a_1^2)$

✓ Physical Interpretation:

- $R(0) = \sigma^2 / (1 - a_1^2)$ is the total variance of the process
- For $|a_1| < 1$, the process is stationary and the variance is finite
- The factor $1/(1 - a_1^2)$ amplifies the noise variance due to the feedback effect

3) Parameters a_1 and σ^2 as Functions of Covariances

Similarly: $a_1 = -R(1)/R(0)$ and $\sigma^2 = R(0) - R^2(1)/R(0)$

IV.1. Limitations of Fourier Analysis

IV.1.1. The Problem with Classical Fourier Transform

The classical Fourier Transform (FT) provides a complete frequency domain representation of a signal but suffers from fundamental limitations when analyzing non-stationary signals.

Example IV.1: Stationary vs. Non-stationary Signals

Consider two signals:

- **Signal A (Stationary):** A superposition of three sinusoids existing simultaneously throughout the entire duration
- **Signal B (Non-stationary):** The same three sinusoidal components occurring sequentially in time

Although both signals have identical Fourier spectra, Signal B loses critical temporal information, we cannot determine when each frequency component occurs.

MATLAB Example IV.1: *Demonstrating Fourier Transform Limitations*

```
% Create stationary and non-stationary signals
fs = 1000; % Sampling frequency
t = 0:1/fs:3-1/fs; % 3 seconds
N = length(t);

% Signal A: Stationary (all frequencies present at all times)
f1 = 10; f2 = 25; f3 = 50;
signalA = sin(2*pi*f1*t) + sin(2*pi*f2*t) + sin(2*pi*f3*t);

% Signal B: Non-stationary (frequencies appear sequentially)
t1 = t(1:N/3);
t2 = t(N/3+1:2*N/3);
t3 = t(2*N/3+1:end);
signalB = [sin(2*pi*f1*t1), sin(2*pi*f2*t2), sin(2*pi*f3*t3)];

% Compute FFTs
fftA = fft(signalA);
fftB = fft(signalB);
freq = (0:N-1)*(fs/N);

% Plot results
figure;
subplot(2,2,1); plot(t, signalA); title('Signal A (Stationary)');
xlabel('Time (s)'); ylabel('Amplitude');
subplot(2,2,2); plot(freq(1:N/2), abs(fftA(1:N/2))); axis([0 fs/10 0 max(abs(fftA))]);
title('FFT of Signal A'); xlabel('Frequency (Hz)'); ylabel('Magnitude');

subplot(2,2,3); plot(t, signalB); title('Signal B (Non-stationary)');
xlabel('Time (s)'); ylabel('Amplitude');
subplot(2,2,4); plot(freq(1:N/2), abs(fftB(1:N/2))); axis([0 fs/10 0 max(abs(fftB))]);
title('FFT of Signal B'); xlabel('Frequency (Hz)'); ylabel('Magnitude');

% The FFTs are nearly identical despite different time structures!
```

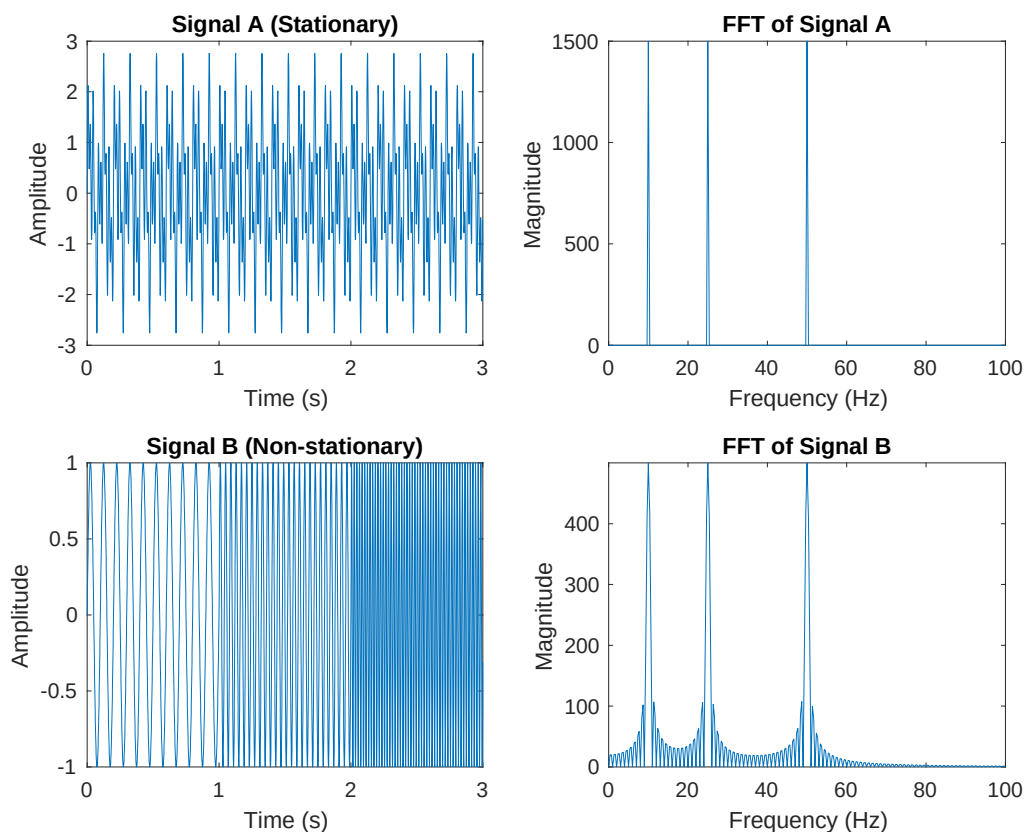


Figure IV.1: *Fourier Transform Limitations*

This example clearly shows the fundamental limitation of Fourier analysis. Even though Signal B has a rich time-varying structure with different frequencies appearing at different times, its FFT spectrum is nearly identical to the stationary Signal A. This demonstrates why we need time-frequency analysis methods.

IV.1.2. Key Limitations of Fourier Analysis

1. *Loss of Temporal Localization:* Standard spectral analysis averages over all time, losing information about when specific frequency components occur.
2. *Inadequacy for Non-stationary Signals:* Fourier analysis assumes stationarity, where each frequency component exists at all times. This assumption fails for:
 - Biological signals (ECG, EEG, EMG)
 - Speech and audio signals
 - Transient phenomena
 - Time-varying systems
3. *Unidirectional Mapping:* Fourier analysis provides either time→frequency or frequency→time mapping, but never a simultaneous time-frequency representation.
4. *No Temporal Resolution:* Cannot answer the fundamental question: “What frequencies are present at what times?”

IV.2. Time-Frequency Duality and Uncertainty Principle

IV.2.1. The Fundamental Trade-off

For any signal analysis technique, there exists a fundamental trade-off between time and frequency resolution. This is mathematically formalized through the **Heisenberg-Gabor Uncertainty Principle**.

Physical Intuition: To measure frequency accurately, we need to observe many oscillations, which requires a long observation window. Conversely, to pinpoint an event in time, we need a short observation window, which limits our frequency resolution. This trade-off is fundamental and unavoidable.

IV.2.2. Mathematical Formulation

For a finite-energy signal $g(t)$ with energy E :

✓ *Time Duration:*

$$\Delta_t^2 = \frac{1}{E} \int_{-\infty}^{\infty} (t - t_0)^2 |g(t)|^2 dt$$

✓ *Frequency Bandwidth:*

$$\Delta_f^2 = \frac{1}{E} \int_{-\infty}^{\infty} (f - f_0)^2 |G(f)|^2 df$$

where:

- $E = \int_{-\infty}^{\infty} |g(t)|^2 dt$ is the signal energy
- $t_0 = \frac{1}{E} \int_{-\infty}^{\infty} t |g(t)|^2 dt$ is the temporal centroid
- $f_0 = \frac{1}{E} \int_{-\infty}^{\infty} f |G(f)|^2 df$ is the spectral centroid

IV.2.3. Heisenberg-Gabor Uncertainty Principle

The fundamental uncertainty relation states:

$$\Delta_t \Delta_f \geq \frac{1}{4\pi}$$

Key Properties:

- ✓ *Lower Bound:* No function can achieve better joint localization than this limit.
- ✓ *Optimal Function:* The Gaussian function achieves equality, providing optimal time-frequency localization.
- ✓ *Trade-off:* Improving time resolution necessarily degrades frequency resolution and vice versa.

MATLAB Example IV.2: Demonstrating the Uncertainty Principle

```
% Demonstrate uncertainty principle with different window widths
fs = 1000;
t = -2:1/fs:2;
f = -fs/2:fs/length(t):fs/2-fs/length(t);

% Create Gaussian windows with different widths
sigma1 = 0.1; % Narrow window
sigma2 = 0.5; % Wide window

g1 = exp(-t.^2/(2*sigma1^2));
g2 = exp(-t.^2/(2*sigma2^2));

% Compute FFTs
G1 = fftshift(fft(fftshift(g1)));
G2 = fftshift(fft(fftshift(g2)));

% Calculate time and frequency spreads
dt1 = sqrt(sum(t.^2 .* abs(g1).^2) / sum(abs(g1).^2));
dt2 = sqrt(sum(t.^2 .* abs(g2).^2) / sum(abs(g2).^2));

df1 = sqrt(sum(f.^2 .* abs(G1).^2) / sum(abs(G1).^2));
df2 = sqrt(sum(f.^2 .* abs(G2).^2) / sum(abs(G2).^2));

% Calculate uncertainty products
uncertainty1 = dt1 * df1;
uncertainty2 = dt2 * df2;

fprintf('Narrow window:  $\Delta t = %.3f$ ,  $\Delta f = %.3f$ ,  $\Delta t \Delta f = %.3f$ \n', dt1, df1, uncertainty1);
fprintf('Wide window:  $\Delta t = %.3f$ ,  $\Delta f = %.3f$ ,  $\Delta t \Delta f = %.3f$ \n', dt2, df2, uncertainty2);
fprintf('Theoretical minimum:  $\Delta t \Delta f \geq %.3f$ \n', 1/(4*pi));

% Plot results
figure;
subplot(2,2,1); plot(t, g1); title(sprintf('Narrow Window ( $\sigma=%.1f$ )', sigma1));
xlabel('Time (s)'); ylabel('Amplitude'); grid on;
subplot(2,2,2); plot(f, abs(G1)); axis([-fs/10 fs/10 0 max(abs(G1))]);
title('Frequency Response');
xlabel('Frequency (Hz)'); ylabel('Magnitude'); grid on;

subplot(2,2,3); plot(t, g2); title(sprintf('Wide Window ( $\sigma=%.1f$ )', sigma2));
xlabel('Time (s)'); ylabel('Amplitude'); grid on;
subplot(2,2,4); plot(f, abs(G2)); axis([-fs/10 fs/10 0 max(abs(G2))]);
title('Frequency Response');
xlabel('Frequency (Hz)'); ylabel('Magnitude'); grid on;
```

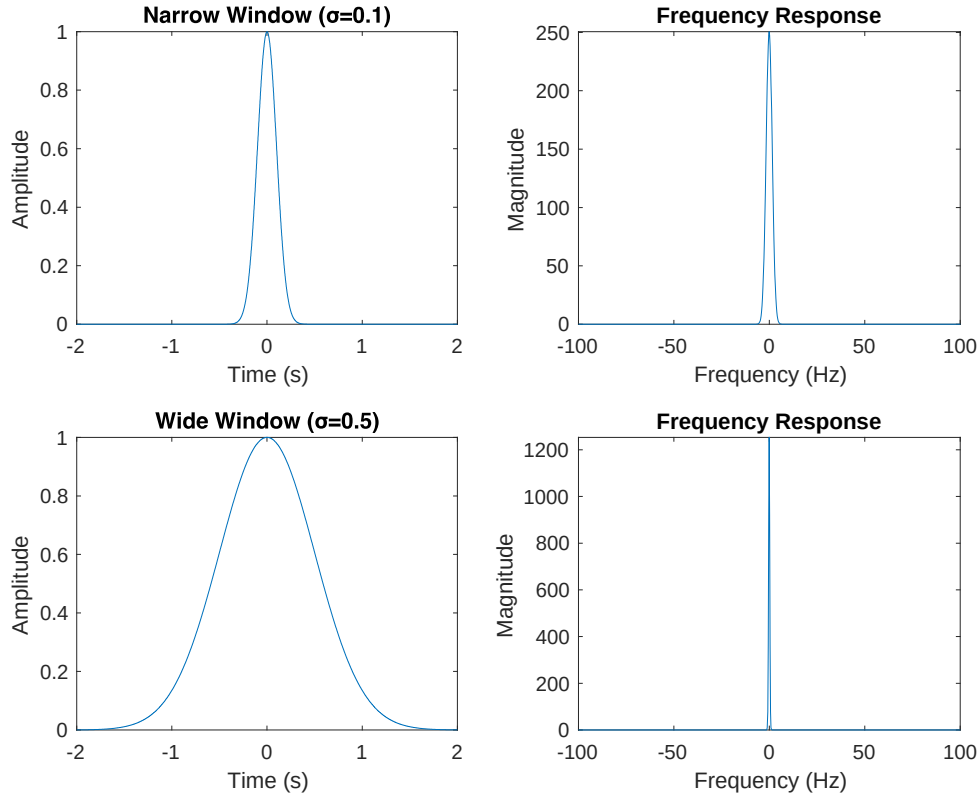


Figure IV.2: Heisenberg-Gabor Uncertainty Principle.

- *Narrow window:* $\Delta t = 0.071, \Delta f = 1.132, \Delta t \Delta f = 0.080$
- *Wide window:* $\Delta t = 0.354, \Delta f = 0.257, \Delta t \Delta f = 0.091$
- *Theoretical minimum:* $\Delta t \Delta f \geq \frac{1}{4\pi} = 0.080$

This example demonstrates the uncertainty principle in action. The narrow Gaussian window provides better time localization but poorer frequency resolution. The wide window provides better frequency resolution but poorer time localization. Both achieve the theoretical minimum uncertainty product, confirming that Gaussians are optimal.

IV.3. Short-Time Fourier Transform (STFT) and Spectrogram

IV.3.1. Gabor Transform

The Gabor transform uses a Gaussian window to achieve optimal time-frequency localization. Dennis Gabor introduced this concept in 1946, combining the optimal properties of Gaussian functions with Fourier analysis.

✓ *Gaussian Window:*

$$g(t) = Ae^{-Bt^2}$$

where $A, B > 0$.

✓ *Gabor Transform:*

$$X_G(t_0, f) = \int_{-\infty}^{\infty} x(t)g(t - t_0)e^{-j2\pi ft} dt$$

$$= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} x(t)e^{-\frac{(t-t_0)^2}{2\sigma^2}} e^{-j2\pi ft} dt$$

✓ *Elementary Gabor Functions:* The basis functions $y(t) = g(t - t_0)e^{j2\pi ft}$ are amplitude-modulated sinusoids:

⇒ Real part: $\cos(2\pi ft)$ modulated by Gaussian envelope.

⇒ Imaginary part: $\sin(2\pi ft)$ modulated by Gaussian envelope.

MATLAB Example IV.3: Gabor Elementary Functions

```
% Create Gabor elementary functions
t = -2:0.01:2;
t0 = 0; % Time center
f = 2; % Frequency
sigma = 0.3; % Gaussian width

% Gaussian envelope
envelope = exp(-(t-t0).^2/(2*sigma^2));

% Complex Gabor function
gabor_complex = envelope .* exp(-1j*2*pi*f*t);

% Real and imaginary parts
gabor_real = real(gabor_complex);
gabor_imag = imag(gabor_complex);

% Plot Gabor functions
figure;
subplot(3,1,1); plot(t, envelope, 'k--', 'LineWidth', 2);
title('Gaussian Envelope'); xlabel('Time (s)'); ylabel('Amplitude'); grid on;

subplot(3,1,2); plot(t, gabor_real, 'b', 'LineWidth', 1.5);
title('Real Part of Gabor Function (Even)');
xlabel('Time (s)'); ylabel('Amplitude'); grid on;

subplot(3,1,3); plot(t, gabor_imag, 'r', 'LineWidth', 1.5);
title('Imaginary Part of Gabor Function (Odd)');
xlabel('Time (s)'); ylabel('Amplitude'); grid on;

% Show effect of frequency changes
figure;
frequencies = [1, 2, 4, 8];
for i = 1:4
    subplot(2,2,i);
    gabor = envelope .* cos(2*pi*frequencies(i)*t);
    plot(t, gabor, 'LineWidth', 1.5);
    title(sprintf('Gabor Function, f = %d Hz', frequencies(i)));
    xlabel('Time (s)'); ylabel('Amplitude'); grid on;
end
```

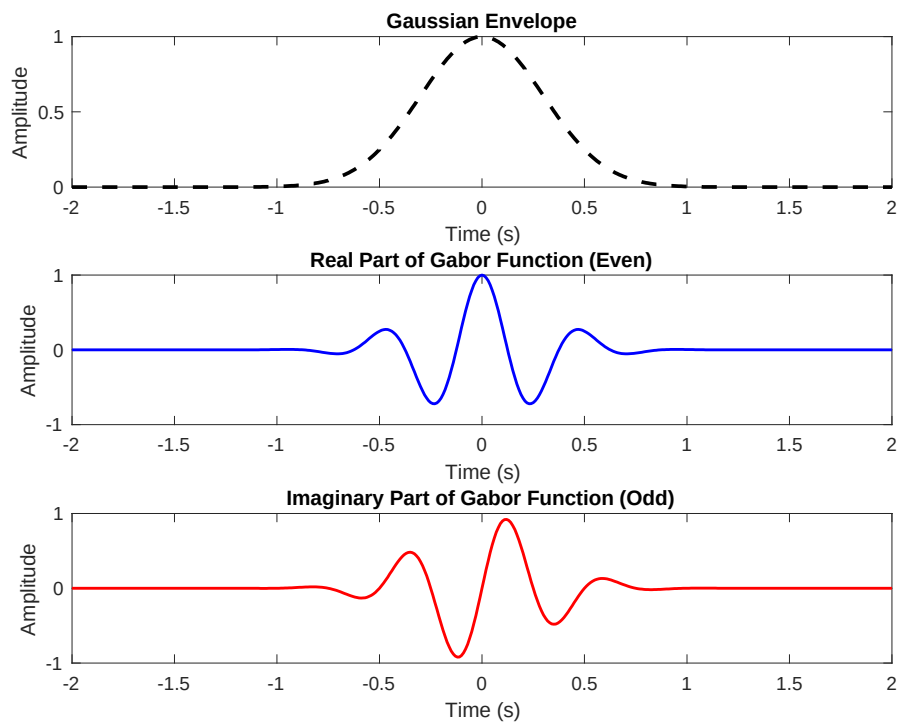


Figure IV.3: *Gabor Function.*

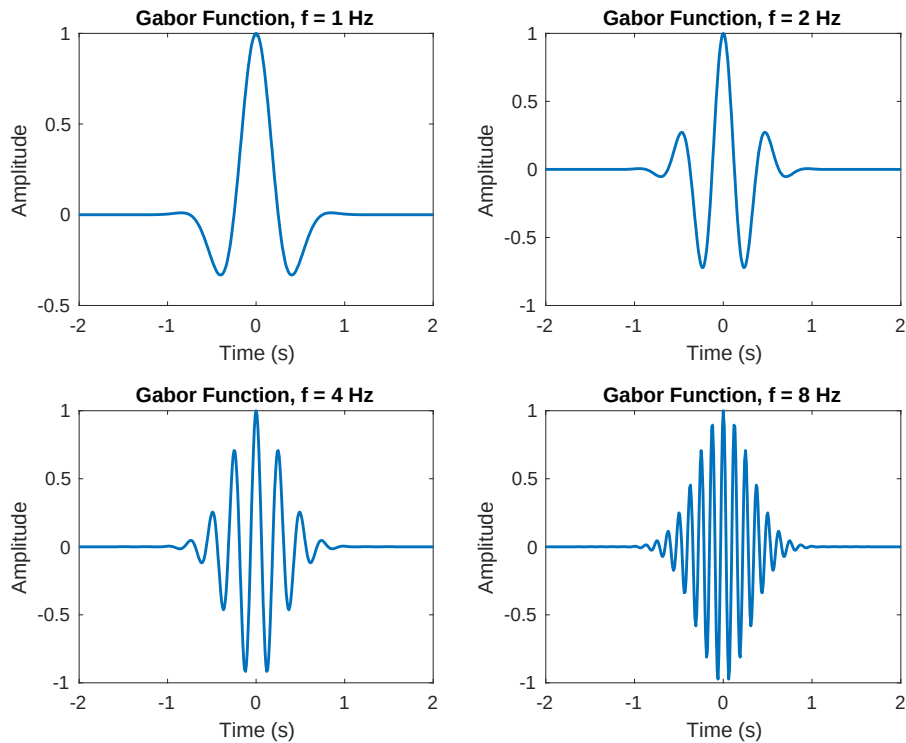


Figure IV.4: *Gabor Function for different frequencies.*

Gabor functions are fundamental building blocks for time-frequency analysis. They represent localized oscillations, sinusoids that exist only for a finite duration. The Gaussian envelope ensures optimal time-frequency localization while the exponential term captures the frequency content.

IV.3.2. Generalized Short-Time Fourier Transform

The STFT generalizes the Gabor transform by allowing arbitrary window functions. This flexibility allows adaptation to specific signal characteristics.

Definition:

$$X_{STFT}(\tau, f) = \int_{-\infty}^{\infty} x(t)w(t - \tau)e^{-j2\pi ft} dt$$

where:

- $w(t - \tau)$ is a sliding analysis window.
- τ is the time localization parameter.
- f is the frequency parameter.

Window Design Considerations:

1. *Main lobe width:* Determines frequency resolution
2. *Side lobe levels:* Affects spectral leakage
3. *Roll-off rate:* Controls transition band characteristics
4. *Computational efficiency:* Some windows allow fast algorithms

Common Window Functions:

Window	Formula	Properties
<i>Rectangular</i>	$w(t) = \text{rect}(t/T)$	Narrowest main lobe, highest side lobes
<i>Hamming</i>	$w(t) = 0.54 + 0.46\cos(2\pi t/T)$	Good compromise, widely used
<i>Hanning</i>	$w(t) = 0.5[1 + \cos(2\pi t/T)]$	Lower side lobes than Hamming
<i>Blackman</i>	$w(t) = 0.42 + 0.5\cos(2\pi t/T) + 0.08\cos(4\pi t/T)$	Very low side lobes
<i>Gaussian</i>	$w(t) = e^{-t^2/2\sigma^2}$	Optimal time-frequency localization

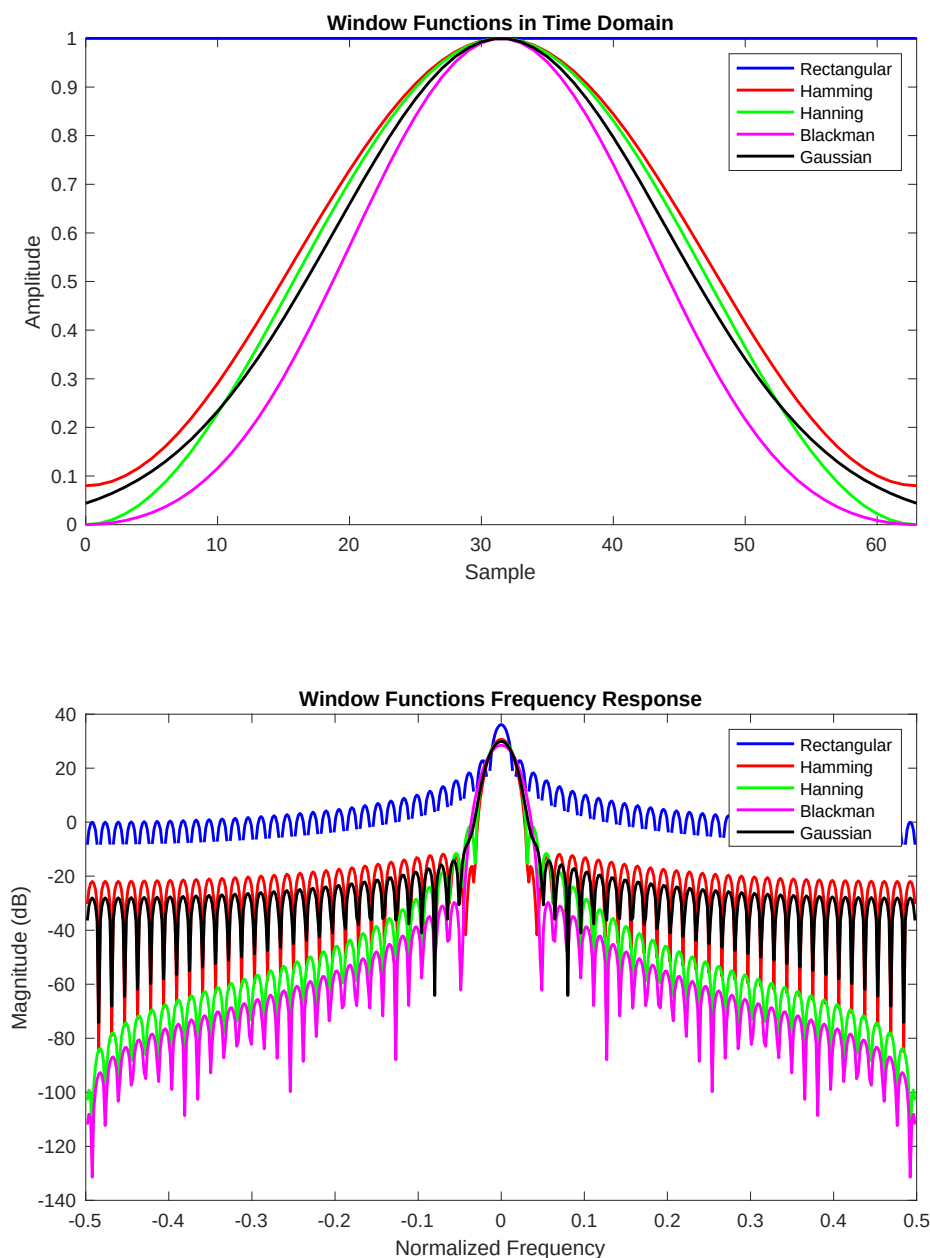


Figure IV.5: Window Function Comparison.

This figure demonstrates the trade-offs between different window functions:

- **Rectangular:** Narrowest main lobe (best frequency resolution) but highest side lobes (worst spectral leakage)
- **Hamming/Hanning:** Good balance between main lobe width and side lobe suppression
- **Blackman:** Excellent side lobe suppression at the cost of wider main lobe
- **Gaussian:** Optimal time-frequency localization properties

IV.3.3. Discrete Short-Time Fourier Transform

For discrete signals $x(n)$ and window $w(n)$ of length N :

✓ *Forward Transform:*

$$X_{STFT}(m, k) = \sum_{n=0}^{N-1} x(n)w(n-m)e^{-j2\pi kn/N}$$

✓ *Inverse Transform:*

$$x(n) = \frac{1}{N} \sum_{m=0}^{N-1} \sum_{k=0}^{N-1} X_{STFT}(m, k)w(n-m)e^{j2\pi kn/N}$$

Implementation Considerations:

1. *Hop size:* Overlap between adjacent windows (typically 50-75%)
2. *Zero-padding:* Improves frequency resolution display
3. *Windowing:* Applied to each frame before FFT
4. *Boundary effects:* Handling signal edges appropriately

IV.3.4. The Spectrogram

The spectrogram provides a visual representation of signal energy distribution in time and frequency:

$$\text{Spectrogram}\{x(t)\} = |X_{STFT}(\tau, f)|^2$$

Interpretation: The spectrogram shows:

- *Horizontal axis:* Time
- *Vertical axis:* Frequency
- *Color/intensity:* Energy density at each time-frequency point

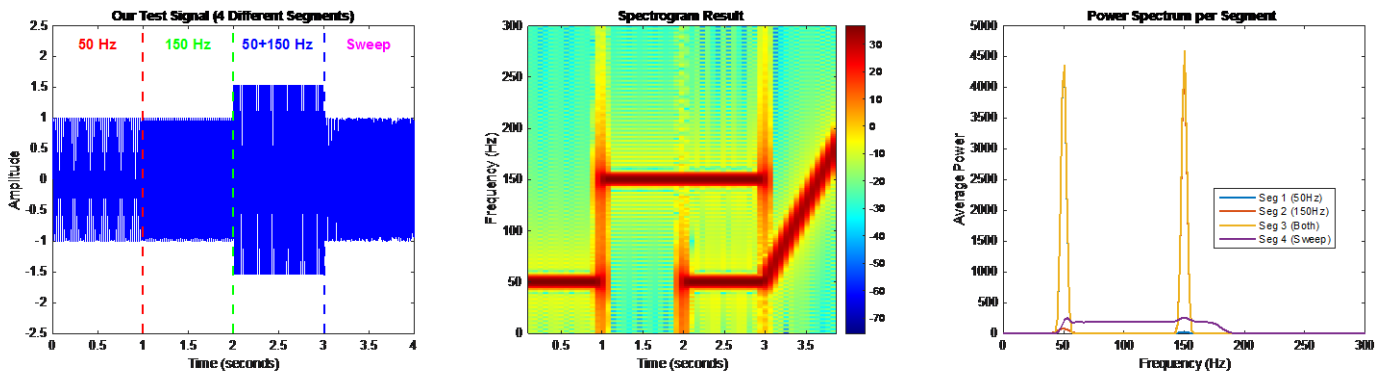


Figure IV.6: Spectrogram of 4 different segments.

Resolution Properties:

- *Long Window*: Good frequency resolution, poor time resolution
- *Short Window*: Good time resolution, poor frequency resolution
- *Fixed Resolution*: Unlike wavelets, STFT provides constant resolution across all frequencies

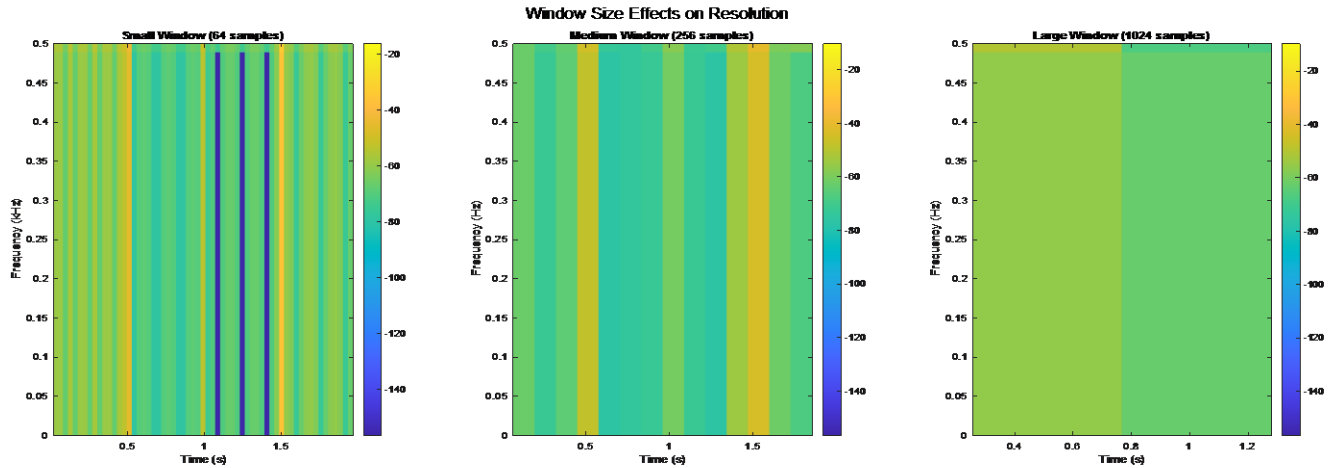


Figure IV.7: window size effects on resolution

IV.3.5. Window Length Trade-offs

The choice of window length fundamentally determines the analysis characteristics:

a) **Long Windows (T_w large):**

- High frequency resolution $\Delta f \propto 1/T_w$
- Poor time localization $\Delta t \propto T_w$
- Suitable for stationary or slowly varying signals
- Better for harmonic analysis

b) **Short Windows (T_w small):**

- High time resolution $\Delta t \propto T_w$
- Poor frequency resolution $\Delta f \propto 1/T_w$
- Suitable for rapidly changing transients
- Better for onset detection

c) **Optimal Window Selection Guidelines:**

- *Signal characteristics*: Consider stationarity, transient content, frequency content
- *Analysis goals*: Harmonic analysis vs. transient detection
- *Computational constraints*: Longer windows require more computation
- *Display requirements*: Consider final visualization needs

Rule of Thumb: For quasi-stationary signals, choose window length to be 10-20% of the signal duration. For transient analysis, use shorter windows (2-5% of signal duration).

IV.4. Wavelet Transform

IV.4.1. Motivation and Advantages

The wavelet transform addresses the fundamental limitation of STFT's fixed resolution by providing adaptive time-frequency resolution. The key insight is that different frequency ranges benefit from different analysis window sizes.

Multi-Resolution Concept:

- *High frequencies*: Need good time resolution (short windows) to capture rapid changes
- *Low frequencies*: Need good frequency resolution (long windows) to separate closely spaced components

Real-World Motivation: Consider analyzing music:

- *Drum hits*: High frequency, short duration → need good time resolution
- *Bass notes*: Low frequency, longer duration → need good frequency resolution - Wavelets naturally provide this adaptive resolution

The wavelet transform provides:

1. *Adaptive Resolution*: Window size varies inversely with frequency
2. *Multi-resolution Analysis*: Natural decomposition at multiple scales
3. *Optimal for Transients*: Excellent for analyzing sudden changes
4. *Flexible Basis Functions*: Various mother wavelets for different signal types
5. *Computational Efficiency*: Fast algorithms available ($O(N)$ complexity)

IV.4.2. Continuous Wavelet Transform (CWT)

✓ *Definition*:

$$W(a, b) = \int_{-\infty}^{\infty} f(t) \psi_{a,b}^*(t) dt$$

✓ *Wavelet Function*:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}} \psi\left(\frac{t-b}{a}\right)$$

Parameters:

- *Scale parameter a* : Controls frequency content (a small → high frequency, a large → low frequency)
- *Translation parameter b* : Controls time localization (sliding window center)
- *Normalization*: $\frac{1}{\sqrt{|a|}}$ ensures energy conservation across scales

Physical Interpretation:

- When a is small, the wavelet is compressed, capturing high-frequency, short-duration events
- When a is large, the wavelet is stretched, capturing low-frequency, long-duration events
- The parameter b slides the wavelet across time to analyze different temporal regions.

MATLAB Example IV.4: Understanding Wavelet Scaling and Translation

```
% Demonstrate wavelet scaling and translation effects
t = -4:0.01:4;
basic_wavelet = (1 - t.^2) .* exp(-t.^2/2); % Mexican Hat wavelet
% Different scales
scales = [0.5, 1, 2, 4];
translations = [-2, -1, 0, 1, 2];
% Show scaling effect
figure;
for i = 1:length(scales)
    a = scales(i);
    scaled_wavelet = (1/sqrt(a)) * basic_wavelet;
    compressed_t = t / a; % Effective time axis compression

    subplot(2, 2, i);
    plot(t, scaled_wavelet, 'LineWidth', 2);
    title(sprintf('Scale a = %.1f', a));
    xlabel('Time'); ylabel('Amplitude');
    grid on; axis([-4, 4, -1, 1.5]);

    % Add frequency indication
    if a < 1
        freq_info = 'High Frequency';
    elseif a == 1
        freq_info = 'Base Frequency';
    else
        freq_info = 'Low Frequency';
    end
    text(0.5, 0.6, freq_info, 'FontSize', 10, 'Color', 'red');
end
% Show translation effect
figure;
base_scale = 1;
for i = 1:length(translations)
    b = translations(i);
    translated_wavelet = (1/sqrt(base_scale)) * basic_wavelet;

    subplot(2, 3, i);
    plot(t + b, translated_wavelet, 'LineWidth', 2);
    title(sprintf('Translation b = %.1f', b));
    xlabel('Time'); ylabel('Amplitude');
    grid on; axis([-6, 6, -1, 1]);

    % Mark the center
    line([b, b], [-1, 1], 'Color', 'red', 'LineStyle', '--');
    text(b+0.2, 0.6, 'Center', 'FontSize', 8, 'Color', 'red');
end
% Show combined scaling and translation
subplot(2, 3, 6);
a_combined = 2; b_combined = 1;
combined_wavelet = (1/sqrt(a_combined)) * ...
    ((1 - ((t-b_combined)/a_combined).^2) .* exp(-((t-b_combined)/a_combined).^2/2));
plot(t, combined_wavelet, 'LineWidth', 2);
title(sprintf('Combined: a=%.1f, b=%.1f', a_combined, b_combined));
xlabel('Time'); ylabel('Amplitude');
grid on; axis([-6, 6, -1, 1]);
```

```
line([b_combined, b_combined], [-1, 1], 'Color', 'red', 'LineStyle', '--');
```

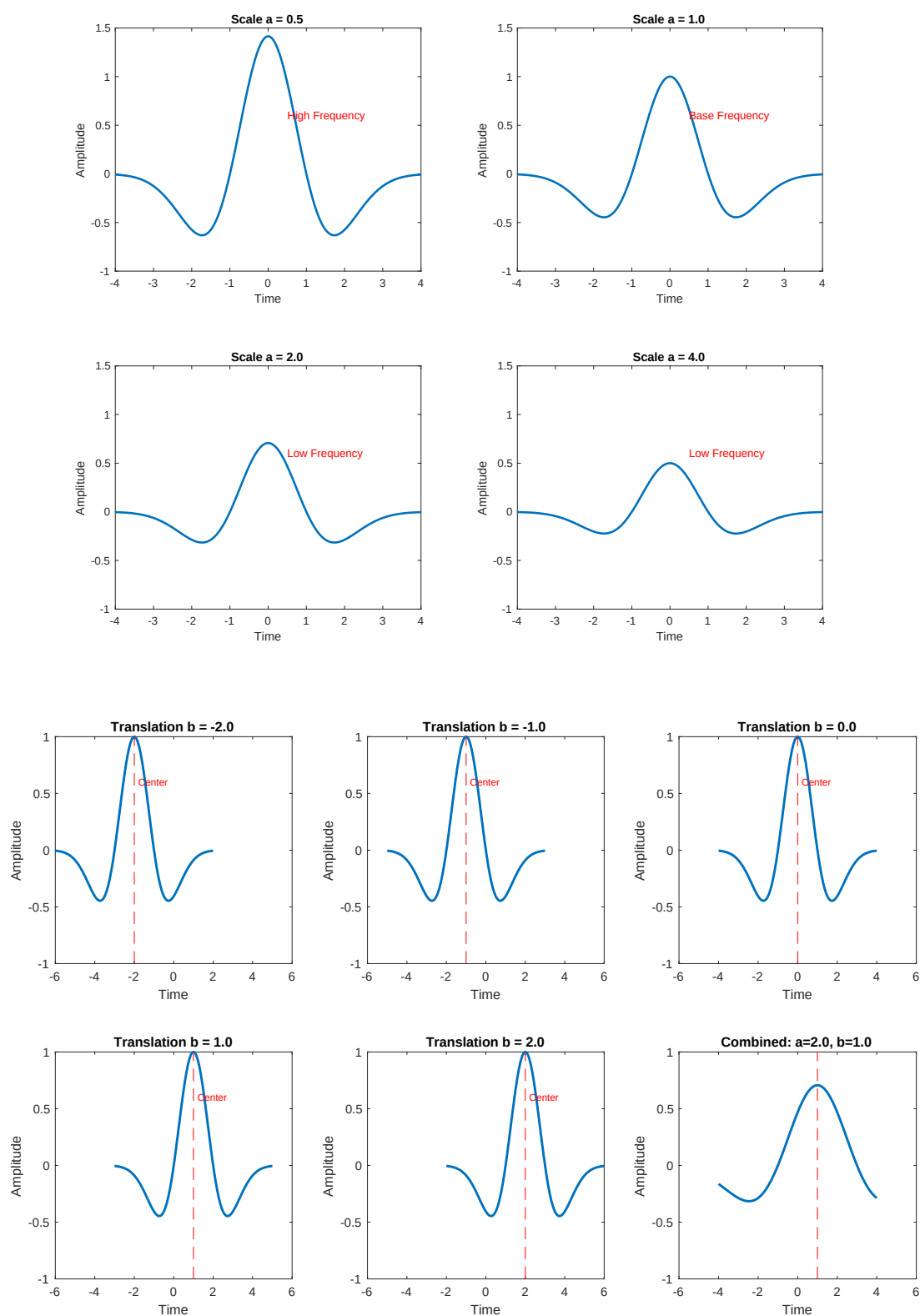


Figure IV.8: Wavelet translation effects.

This example illustrates the fundamental wavelet operations:

- **Scaling:** Compresses or stretches the wavelet, affecting its frequency selectivity

- *Translation*: Slides the wavelet in time for temporal localization
- *Energy Conservation*: The $1/\sqrt{a}$ normalization ensures consistent energy across scales

IV.4.3. Inverse Continuous Wavelet Transform

✓ *Reconstruction Formula*:

$$f(t) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W(a, b) \psi_{a,b}(t) \frac{da db}{a^2}$$

Admissibility Condition: $C_\psi = \int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < \infty$

Physical Meaning:

- The admissibility condition ensures perfect reconstruction is possible
- It requires $\int_{-\infty}^{\infty} \psi(t) dt = 0$ (zero mean)
- This means wavelets must be oscillatory (not purely positive or negative)
- The condition also requires $\Psi(0) = 0$ (no DC component)

The inverse transform reconstructs the original signal by weighted superposition of wavelets at all scales and positions. The $1/a^2$ factor in the integral measure accounts for the logarithmic scaling of the transform.

IV.4.4. Popular Wavelet Families

Different wavelets are suited for different applications based on their mathematical properties and shape characteristics.

IV.4.4.1. Haar Wavelet (Simplest)

$$\psi(t) = \begin{cases} 1 & 0 \leq t < 0.5 \\ -1 & 0.5 \leq t < 1 \\ 0 & \text{elsewhere} \end{cases}$$

Properties:

- Orthogonal, compact support
- Discontinuous (poor for smooth signals)
- Excellent for edge detection - Minimal computational requirements

IV.4.4.2. Morlet Wavelet (Complex)

$$\psi(t) = \frac{1}{\pi^{1/4}} e^{j\omega_0 t} e^{-t^2/2}$$

Properties:

- Complex-valued (provides phase information)
- Good time-frequency localization
- Excellent for oscillatory signal analysis
- Popular in geophysics and neuroscience

IV.4.4.3. Daubechies Wavelets (db_N)

Properties:

- Orthogonal with compact support
- N vanishing moments (good for polynomial signals)
- Increasing N provides better frequency selectivity
- Foundation for discrete wavelet transform

IV.4.4.4. Mexican Hat (Ricker)

$$\psi(t) = \frac{2}{\sqrt{3}\pi^{1/4}} (1 - t^2)e^{-t^2/2}$$

Properties:

- Real-valued, symmetric
- Good for detecting peaks and discontinuities
- Similar to second derivative of Gaussian
- Popular in edge detection

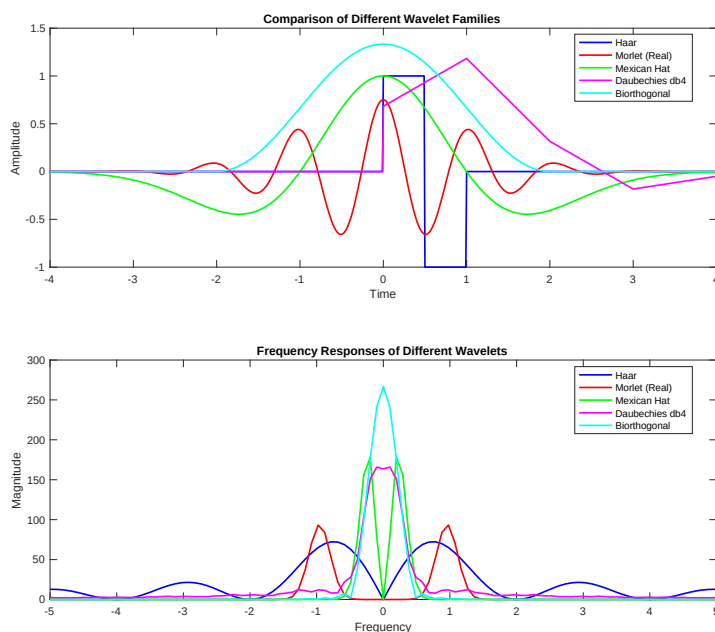


Figure IV.9: Wavelet Family Comparison

Each wavelet family has distinct characteristics making them suitable for different applications:

- **Haar:** Simple, fast, good for discontinuous signals

- **Morlet:** Complex, excellent for harmonic analysis, provides phase information
- **Mexican Hat:** Good for feature detection, similar to Laplacian of Gaussian
- **Daubechies:** Versatile, orthogonal, good for general signal processing
- **Biorthogonal:** Linear phase, good for image processing and reconstruction

IV.4.5. Discrete Wavelet Transform (DWT)

For computational efficiency and practical implementation, the continuous parameters are discretized using a dyadic grid.

Dyadic Discretization:

- *Scales:* $a = 2^{-m}$, $m \in \mathbb{Z}$ (powers of 2)
- *Translations:* $b = k2^{-m}$, $k \in \mathbb{Z}$ (dyadic grid)

Discrete Wavelet:

$$\psi_{m,k}(t) = 2^{m/2} \psi(2^m t - k)$$

Advantages of Dyadic Sampling:

1. *Computational Efficiency:* Enables fast algorithms
2. *Critical Sampling:* No redundancy in representation
3. *Orthogonality:* Allows perfect reconstruction
4. *Multi-resolution Structure:* Natural hierarchical decomposition

IV.4.6. Multi-Resolution Analysis (MRA)

Multi-Resolution Analysis provides the mathematical framework for understanding how DWT decomposes signals into different frequency bands at different resolutions.

Core Concept: A signal can be represented as a combination of:

- *Approximation:* Low-frequency, coarse features (overall shape)
- *Details:* High-frequency, fine features (rapid variations)

Mathematical Framework: The DWT decomposes a signal through iterative filtering and downsampling:

Analysis (Decomposition):

- ✓ *Detail Coefficients:* (High-pass filtering)

$$W_D(j, k) = \sum_n x(n) h_1(n - 2k)$$

✓ *Approximation Coefficients: (Low-pass filtering)*

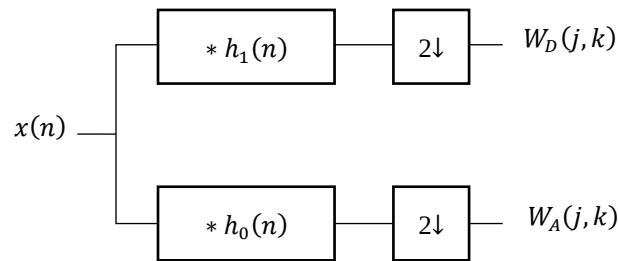
$$W_A(j, k) = \sum_n x(n) h_0(n - 2k)$$

✓ *Synthesis (Reconstruction):*

$$x(n) = \sum_k W_A(j_0, k) \phi_{j_0, k}(n) + \sum_{j=j_0}^{J-1} \sum_k W_D(j, k) \psi_{j, k}(n)$$

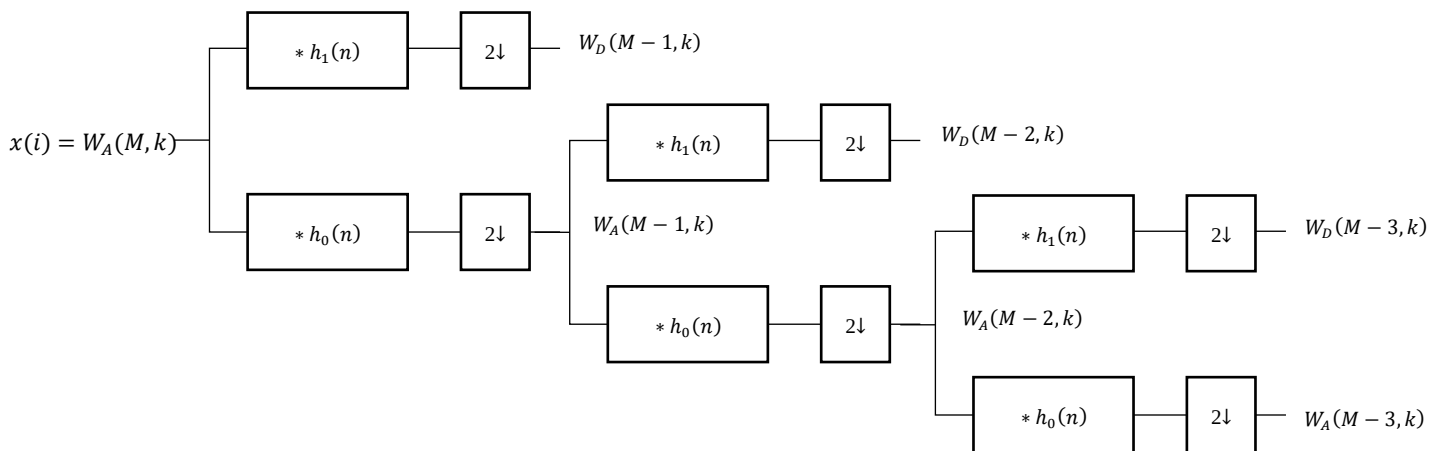
where:

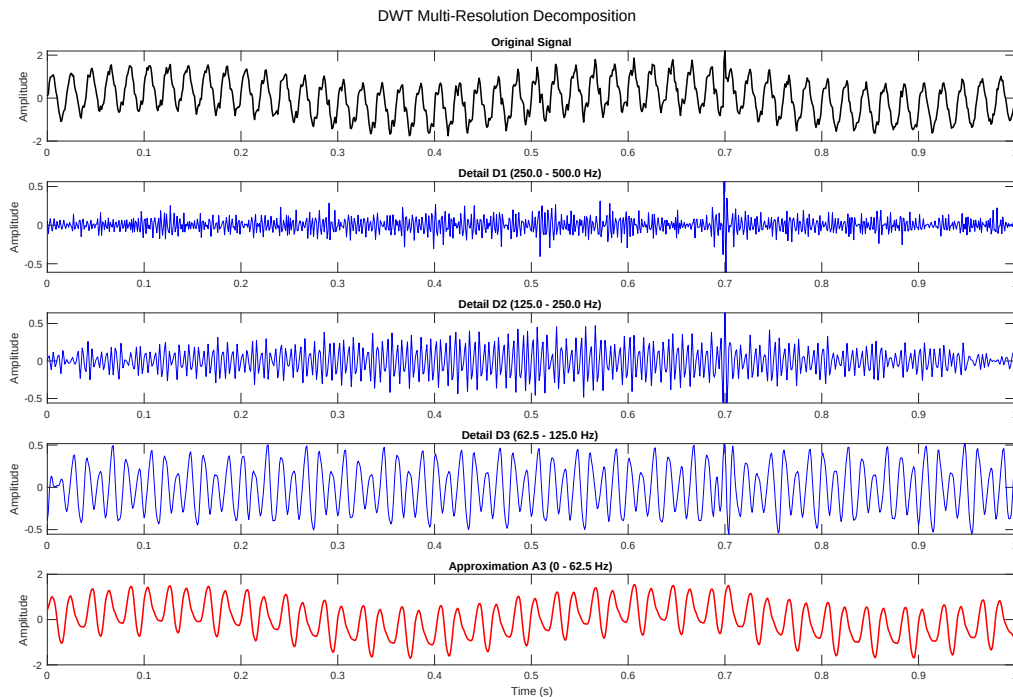
- $\phi_{j, k}(n)$ are scaling functions (approximation basis).
- $\psi_{j, k}(n)$ are wavelet functions (detail basis).
- j_0 is the coarsest resolution level.
- J is the finest resolution level.



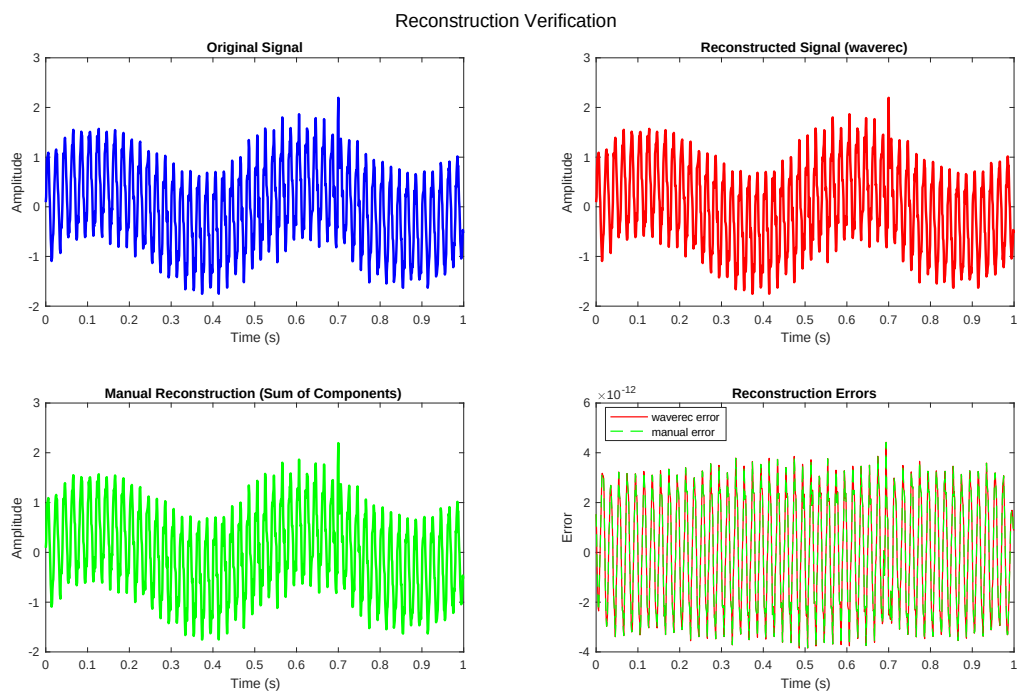
MATLAB Example IV.5:

Multi-Resolution Analysis Demonstration: example of level 3 decomposition of a signal is shown in the following figure:

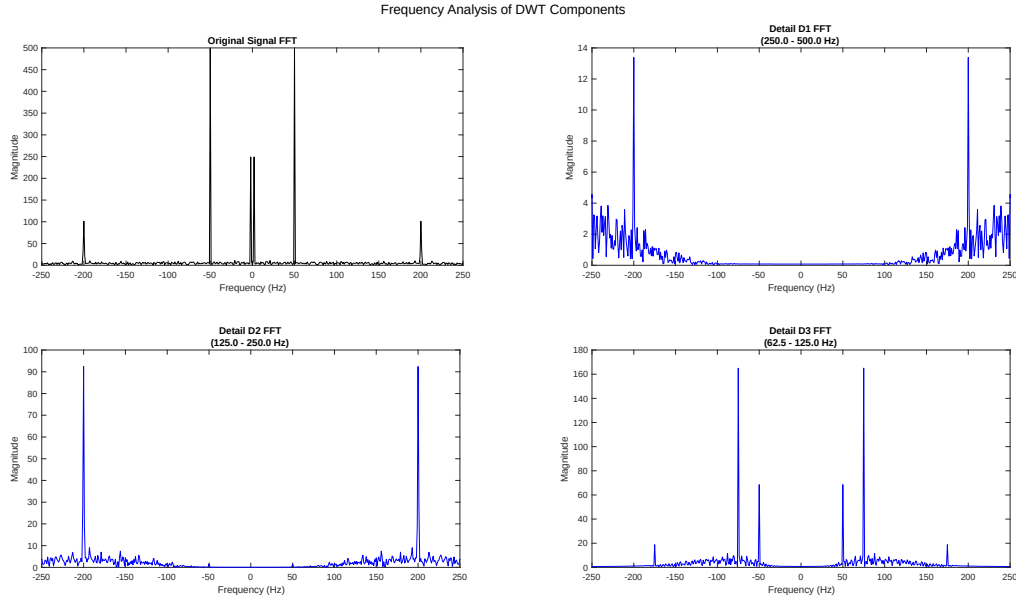




(a)



(b)



(c)

Figure IV.10: (a) DWT Multi Resolution Decomposition, (b) Reconstruction Verification, (c) Frequency Analysis of DWT Components

This comprehensive example demonstrates several key MRA concepts:

1. **Hierarchical Decomposition:** Each detail level captures a specific frequency band
2. **Energy Distribution:** Shows how signal energy is distributed across scales
3. **Perfect Reconstruction:** Demonstrates the reversibility of the DWT
4. **Frequency Localization:** Each detail level has well-defined frequency bounds

IV.4.7. Wavelet Properties and Selection

Choosing the appropriate wavelet for a specific application requires understanding various mathematical properties and their practical implications.

1. **Orthogonality:** $\langle \psi_{j,k}, \psi_{j',k'} \rangle = \delta_{j,j'} \delta_{k,k'}$
 - Ensures unique representation
 - Enables efficient computation
 - Guarantees energy conservation
2. **Compact Support:** Wavelet has finite duration
 - Better time localization
 - Finite impulse response filters
 - Boundary effects are localized
3. **Vanishing Moments:** $\int_{-\infty}^{\infty} t^n \psi(t) dt = 0$ for $n = 0, 1, \dots, N - 1$

- Number of vanishing moments = degree of polynomials that wavelets can represent exactly
- Higher moments $\rightarrow$ better approximation of smooth functions
- Related to the number of zeros at $\omega = 0$ in frequency domain
- 4. **Regularity (Smoothness):** Differentiability properties
 - Smoother wavelets $\rightarrow$ better frequency selectivity
 - Important for signal reconstruction quality
 - Trade-off with compact support
- 5. **Symmetry:** Even or odd symmetry properties
 - Linear phase filters
 - Important for image processing
 - Reduces boundary artifacts

IV.5. STFT vs. Wavelet Transform Comprehensive Comparison

IV.5.1. Resolution Characteristics Analysis

The fundamental difference between STFT and wavelets lies in their resolution properties, which determine their suitability for different signal analysis tasks.

Time-Frequency Resolution Comparison:

Aspect	STFT	Wavelet Transform
Time Resolution	Constant: $\Delta t = \text{constant}$	Variable: $\Delta t \propto a$
Frequency Resolution	Constant: $\Delta f = \text{constant}$	Variable: $\Delta f \propto 1/a$
Analysis Window	Fixed size for all frequencies	Adaptive: narrow for HF, wide for LF
Frequency Coverage	Linear frequency sampling	Logarithmic frequency sampling
Computational Complexity	$O(N \log N)$ per frame	$O(N)$ for complete decomposition
Memory Requirements	$O(MN)$ for M frames	$O(N)$
Perfect Reconstruction	Possible with overlap	Inherent with orthogonal wavelets

IV.5.2. Practical Applications and Use Cases

STFT Advantages and Applications:

1. **Linear Frequency Analysis:**
 - Musical instrument analysis
 - Speech processing (formant tracking)
 - Vibration analysis
 - Real-time spectral analysis
2. **Uniform Resolution Requirements:**
 - Harmonic analysis

- Filter bank design
- Doppler shift analysis
- 3. **Implementation Benefits:**
 - FFT-based algorithms
 - Real-time processing capability
 - Intuitive frequency interpretation
 - Established windowing techniques

Wavelet Advantages and Applications:

1. **Multi-scale Analysis:**
 - Image compression (JPEG 2000)
 - Biomedical signal analysis (ECG, EEG)
 - Geophysical data processing
 - Financial time series analysis
2. **Transient Detection:**
 - Fault diagnosis in machinery
 - Earthquake analysis
 - Power quality monitoring
 - Speech onset detection
3. **Adaptive Resolution Benefits:**
 - Better frequency resolution at low frequencies
 - Better time resolution at high frequencies
 - Natural multi-resolution decomposition
 - Efficient sparse representations

IV.6. Wigner-Ville Transform

IV.6.1. Mathematical Foundation and Motivation

The Wigner-Ville Distribution (WVD) represents a fundamentally different approach to time-frequency analysis. Unlike STFT and wavelets, which use basis function decompositions, the WVD is based on the concept of local autocorrelation and provides the highest possible time-frequency resolution theoretically achievable.

Physical Motivation: The WVD was originally developed in quantum mechanics by Eugene Wigner (1932) to describe quantum states in phase space. Ville (1948) later adapted it for signal processing, recognizing its potential for time-frequency analysis.

Key Innovation: Rather than windowing the signal or using basis functions, the WVD directly estimates the instantaneous power spectral density by analyzing local autocorrelation properties.

IV.6.2. Continuous Wigner-Ville Distribution

✓ *Definition:*

$$W_x(t, f) = \int_{-\infty}^{\infty} x\left(t + \frac{\tau}{2}\right) x^*\left(t - \frac{\tau}{2}\right) e^{-j2\pi f\tau} d\tau$$

✓ *Alternative Formulation:*

$$W_x(t, f) = 2 \int_{-\infty}^{\infty} R_x(t, \tau) e^{-j2\pi f\tau} d\tau$$

where $R_x(t, \tau) = x\left(t + \frac{\tau}{2}\right) x^*\left(t - \frac{\tau}{2}\right)$ is the *instantaneous autocorrelation function*.

Physical Interpretation:

- The WVD measures the “local correlation” of the signal around time t
- It represents the instantaneous power spectral density
- The τ parameter captures the “memory” or correlation length
- Unlike STFT/wavelets, no external basis functions are imposed

IV.6.3. Fundamental Properties

1. Real-Valued: $W_x(t, f) \in \mathbb{R}$ for all real signals

2. Time Marginal:

$$\int_{-\infty}^{\infty} W_x(t, f) df = |x(t)|^2$$

The integration over frequency yields the instantaneous power.

3. Frequency Marginal:

$$\int_{-\infty}^{\infty} W_x(t, f) dt = |X(f)|^2$$

The integration over time yields the power spectral density.

4. Energy Conservation:

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W_x(t, f) dt df = \int_{-\infty}^{\infty} |x(t)|^2 dt = E_x$$

5. Translation Properties:

- Time shift: $W_{x(t-t_0)}(t, f) = W_x(t - t_0, f)$
- Frequency shift: $W_{x(t)e^{j2\pi f_0 t}}(t, f) = W_x(t, f - f_0)$

IV.6.4. Cross-Term Interference Problem

The Fundamental Limitation: The WVD suffers from cross-term interference when analyzing multi-component signals. For a signal with two components:

$$x(t) = x_1(t) + x_2(t)$$

The WVD becomes:

$$W_x(t, f) = W_{x_1}(t, f) + W_{x_2}(t, f) + W_{x_1, x_2}(t, f) + W_{x_2, x_1}(t, f)$$

The cross-terms $W_{x_1, x_2}(t, f)$ and $W_{x_2, x_1}(t, f)$ create **interference patterns** that can obscure the true time-frequency content.

Cross-Term Characteristics:

- Oscillatory interference patterns
- Located between true components
- Can have negative values
- Magnitude proportional to $\sqrt{|x_1(t)||x_2(t)|}$

IV.6.5.Applications and Practical Considerations

Advantages of WVD:

1. *Optimal Resolution:* No uncertainty principle limitations
2. *Real-valued:* Always produces real distributions
3. *Marginal Properties:* Perfect time/frequency marginals
4. *Instantaneous Frequency:* Direct estimation possible

Applications:

1. *Radar and Sonar:* Doppler analysis
2. *Biomedical:* Non-stationary physiological signals
3. *Machine Fault Detection:* Transient analysis
4. *Seismic Analysis:* Earthquake signal processing
5. *Speech Processing:* Voice analysis and synthesis

Limitations:

1. *Cross-term Interference:* Major problem for multi-component signals
2. *Computational Complexity:* $O(N^2)$ for discrete implementation
3. *Negative Values:* Can be difficult to interpret

4. *Boundary Effects*: Requires careful handling at signal edges

IV.7. Time-Scale Analysis

IV.7.1. Motivation and Theoretical Foundation

Time-scale analysis represents a paradigm shift from traditional frequency-based approaches to scale-based signal decomposition. This approach is particularly powerful for analyzing signals with self-similar or fractal properties, where the concept of “scale” is more natural than “frequency.”

Fundamental Concept: Instead of asking “what frequencies are present at what times,” time-scale analysis asks “what scales are present at what times.” This distinction becomes crucial for signals that exhibit:

- Self-similarity across different scales
- Fractal characteristics
- Power-law relationships
- Multiscale phenomena

Mathematical Foundation: The relationship between frequency and scale in continuous wavelet transform:

$$\text{Scale } a \leftrightarrow \text{Frequency } f_c/a$$

where f_c is the center frequency of the mother wavelet.

IV.7.2. Scale vs. Frequency Relationship

Physical Interpretation:

- *Small scales* ($a < 1$): Compressed wavelets $\rightarrow$ high frequencies $\rightarrow$ fine details
- *Large scales* ($a > 1$): Stretched wavelets $\rightarrow$ low frequencies $\rightarrow$ coarse features

Logarithmic Scale Sampling: Unlike STFT’s linear frequency sampling, wavelets naturally provide logarithmic scale sampling:

$$a_j = a_0 \cdot s^j, \quad j = 0, 1, 2, \dots$$

where $s > 1$ is the scale step (typically $s = 2$).

Equivalent Frequency Sampling:

$$f_j = \frac{f_c}{a_0 \cdot s^j} = f_{\max} \cdot s^{-j}$$

This results in constant relative bandwidth (constant Q-factor):

$$\frac{\Delta f}{f} = \text{constant}$$

IV.7.3. Multiscale Signal Decomposition

Time-scale analysis naturally leads to multiscale decomposition, where signals are analyzed at multiple resolution levels simultaneously.

Hierarchical Decomposition Structure:

Original Signal

```

├─ Scale 1 (Highest frequency details)
├─ Scale 2 (Medium-high frequency details)
├─ Scale 3 (Medium frequency details)
├─ ...
└─ Scale J (Lowest frequency approximation)
    
```

Mathematical Framework:

$$x(t) = \sum_{j=J_0}^J \sum_k d_{j,k} \psi_{j,k}(t) + \sum_k a_{J,k} \phi_{J,k}(t)$$

where:

- $d_{j,k}$ are detail coefficients at scale j
- $a_{J,k}$ are approximation coefficients at coarsest scale J
- $\psi_{j,k}(t)$ are wavelet functions
- $\phi_{J,k}(t)$ are scaling functions

MATLAB Example IV.5 Program:

```
% Demonstrate Multi-Resolution Analysis with DWT - CORRECTED VERSION
% Create a test signal with multiple frequency components
fs = 1000;
t = 0:1/fs:1-1/fs;
N = length(t);

% Multi-component test signal:
% Low-frequency trend + medium frequency oscillation + high-frequency noise + transient
trend = 0.5 * sin(2*pi*2*t);           % Low frequency (2 Hz)
oscillation = sin(2*pi*50*t);          % Medium frequency (50 Hz)
noise = 0.3 * sin(2*pi*200*t) .* exp(-5*(t-0.5).^2); % High freq transient
impulse = zeros(size(t));
impulse(abs(t - 0.7) < 0.001) = 2;     % Sharp transient

test_signal = trend + oscillation + noise + impulse;

% Add some random noise
test_signal = test_signal + 0.1 * randn(size(t));

% Perform multi-level DWT decomposition
wavelet_name = 'db6'; % Daubechies 6
levels = 3;

% Decompose signal
[C, L] = wavedec(test_signal, levels, wavelet_name);

% Extract approximation and detail coefficients
A = cell(levels, 1); % Approximations
D = cell(levels, 1); % Details

for j = 1:levels
    A{j} = appcoef(C, L, wavelet_name, j);
    D{j} = detcoef(C, L, j);
end

% CORRECTED: Reconstruct individual components properly
A_recon = cell(levels, 1);
D_recon = cell(levels, 1);

% Method 1: Direct reconstruction using wrcoef function (RECOMMENDED)
for j = 1:levels
    % Reconstruct approximation at level j
    A_recon{j} = wrcoef('a', C, L, wavelet_name, j);

    % Reconstruct detail at level j
    D_recon{j} = wrcoef('d', C, L, wavelet_name, j);
end

% Alternative Method 2: Manual reconstruction (for educational purposes)
% Uncomment this section if you want to see the manual approach
%{
for j = 1:levels
    % For approximation reconstruction
    C_temp = zeros(size(C));
    % Only keep the approximation coefficients at level j
    if j == levels
        % For the final level, use the approximation coefficients
```

Appendix

```
C_temp(1:L(1)) = A{j};
else
    % For other levels, we need to reconstruct from that level down
    % This is more complex and requires iterative reconstruction
    temp_C = C;
    temp_L = L;
    % Zero out all detail coefficients above level j
    for k = 1:j-1
        start_idx = sum(L(1:k)) + 1;
        end_idx = sum(L(1:k+1));
        temp_C(start_idx:end_idx) = 0;
    end
    A_recon{j} = waverec(temp_C, temp_L, wavelet_name);
end

% For detail reconstruction
C_temp = zeros(size(C));
start_idx = sum(L(1:j)) + 1;
end_idx = sum(L(1:j+1));
C_temp(start_idx:end_idx) = D{j};
D_recon{j} = waverec(C_temp, L, wavelet_name);
end
%}

% Plot original signal and decomposition
figure('Position', [100, 100, 1200, 900]);

% Original signal
subplot(levels+2, 1, 1);
plot(t, test_signal, 'k', 'LineWidth', 1.5);
title('Original Signal');
ylabel('Amplitude'); grid on;
xlim([0, 1]);

% Plot details from finest to coarsest
for j = 1:levels
    subplot(levels+2, 1, j+1);

    % All reconstructed signals have the same length as original
    plot(t, D_recon{j}, 'b', 'LineWidth', 1);

    % Calculate frequency band
    freq_low = fs / (2^(j+1));
    freq_high = fs / (2^j);

    title(sprintf('Detail D%d (%.1f - %.1f Hz)', j, freq_low, freq_high));
    ylabel('Amplitude'); grid on;
    xlim([0, 1]);
end

% Final approximation
subplot(levels+2, 1, levels+2);
plot(t, A_recon{levels}, 'r', 'LineWidth', 1.5);
freq_approx = fs / (2^(levels+1));
title(sprintf('Approximation A%d (0 - %.1f Hz)', levels, freq_approx));
xlabel('Time (s)'); ylabel('Amplitude'); grid on;
xlim([0, 1]);

sgtitle('DWT Multi-Resolution Decomposition');
```

Appendix

% Analyze frequency content of each level

```
figure('Position', [200, 200, 1400, 800]);
frequencies = fftshift(fft(test_signal));
freq_axis = (-N/2:N/2-1) * (fs/N);

subplot(2,3,1);
plot(freq_axis, abs(frequencies), 'k', 'LineWidth', 1);
title('Original Signal FFT');
xlabel('Frequency (Hz)'); ylabel('Magnitude');
xlim([-250, 250]); grid on;
```

% Plot FFT of detail coefficients

```
for j = 1:min(5, levels)
    subplot(2,2,j+1);
    detail_fft = fftshift(fft(D_recon{j}));
    plot(freq_axis, abs(detail_fft), 'b', 'LineWidth', 1);

    freq_low = fs / (2^(j+1));
    freq_high = fs / (2^j);
    title(sprintf('Detail D%d FFT\n(%.1f - %.1f Hz)', j, freq_low, freq_high));
    xlabel('Frequency (Hz)'); ylabel('Magnitude');
    xlim([-250, 250]); grid on;
```

% Highlight the theoretical frequency band

```
hold on;
y_lim = ylim;
fill([freq_low freq_high freq_high freq_low], ...
     [y_lim(1) y_lim(1) y_lim(2) y_lim(2)], ...
     'red', 'FaceAlpha', 0.2, 'EdgeColor', 'none');
fill([-freq_high -freq_low -freq_low -freq_high], ...
     [y_lim(1) y_lim(1) y_lim(2) y_lim(2)], ...
     'red', 'FaceAlpha', 0.2, 'EdgeColor', 'none');
end
```

sgtitle('Frequency Analysis of DWT Components');

% Energy distribution analysis

```
energy_original = sum(test_signal.^2);
energy_details = zeros(levels, 1);
energy_approx = sum(A_recon{levels}.^2);
```

```
for j = 1:levels
    energy_details(j) = sum(D_recon{j}.^2);
end
```

% Display energy distribution

```
fprintf('\nEnergy Distribution Analysis:\n');
fprintf('%-20s %-15s %-15s\n', 'Component', 'Energy', 'Percentage');
fprintf('%-20s %-15s %-15s\n', repmat('-', 1, 20), repmat('-', 1, 15), repmat('-', 1, 15));
```

```
for j = 1:levels
    energy_percent = 100 * energy_details(j) / energy_original;
    freq_low = fs / (2^(j+1));
    freq_high = fs / (2^j);
    fprintf('%-20s %-15.3f %-15.1f%%\n', sprintf('Detail D%d (%.1f-%.1f Hz)', j, freq_low, freq_high), ...
           energy_details(j), energy_percent);
end
```

```
energy_percent_approx = 100 * energy_approx / energy_original;
freq_approx = fs / (2^(levels+1));
```

Appendix

```
fprintf('%-20s %-15.3f %-15.1f%%\n', sprintf('Approximation A%d (0-%.1f Hz)', levels, freq_approx), ...
    energy_approx, energy_percent_approx);

total_energy = energy_approx + sum(energy_details);
fprintf('%-20s %-15.3f %-15.1f%%\n', 'Total Reconstructed', total_energy, 100*total_energy/energy_original);

% Demonstrate perfect reconstruction
reconstructed = waverec(C, L, wavelet_name);
reconstruction_error = max(abs(test_signal - reconstructed));
fprintf('\nPerfect Reconstruction Check:\n');
fprintf('Maximum reconstruction error: %.2e\n', reconstruction_error);
if reconstruction_error < 1e-12
    fprintf('✓ Perfect reconstruction achieved!\n');
else
    fprintf('✗ Reconstruction error detected!\n');
end

% Verify that sum of all components equals original signal
manual_reconstruction = A_recon{levels};
for j = 1:levels
    manual_reconstruction = manual_reconstruction + D_recon{j};
end

manual_error = max(abs(test_signal - manual_reconstruction));
fprintf('Manual reconstruction error: %.2e\n', manual_error);
if manual_error < 1e-12
    fprintf('✓ Manual reconstruction also perfect!\n');
else
    fprintf('✗ Manual reconstruction error detected!\n');
end

% Plot reconstruction verification
figure('Position', [300, 300, 1000, 600]);

subplot(2,2,1);
plot(t, test_signal, 'b', 'LineWidth', 2);
title('Original Signal');
xlabel('Time (s)'); ylabel('Amplitude');
grid on;

subplot(2,2,2);
plot(t, reconstructed, 'r', 'LineWidth', 2);
title('Reconstructed Signal (waverec)');
xlabel('Time (s)'); ylabel('Amplitude');
grid on;

subplot(2,2,3);
plot(t, manual_reconstruction, 'g', 'LineWidth', 2);
title('Manual Reconstruction (Sum of Components)');
xlabel('Time (s)'); ylabel('Amplitude');
grid on;

subplot(2,2,4);
plot(t, test_signal - reconstructed, 'r', 'LineWidth', 1);
hold on;
plot(t, test_signal - manual_reconstruction, 'g--', 'LineWidth', 1);
title('Reconstruction Errors');
xlabel('Time (s)'); ylabel('Error');
legend('waverec error', 'manual error', 'Location', 'best');
grid on;
```

Appendix

```
sgtitle('Reconstruction Verification');

fprintf('\nExplanation of the correction:\n');
fprintf('- Used wrcoef() function for proper component reconstruction\n');
fprintf('- wrcoef("a", C, L, wavelet, j) reconstructs approximation at level j\n');
fprintf('- wrcoef("d", C, L, wavelet, j) reconstructs detail at level j\n');
fprintf('- This ensures all reconstructed components have the same length as original\n');
fprintf('- Alternative manual method is more complex and educational only\n');
```

Books

- Cooley, J. W., & Tukey, J. W. (1965). *An Algorithm for the Machine Calculation of Complex Fourier Series*. *Mathematics of Computation*, 19(90), 297-301.
- Cohen, L. (1995). *Time-Frequency Analysis*. Prentice Hall.
- Daubechies, I. (1992). *Ten Lectures on Wavelets*. SIAM.
- Durbin, J. (1960). The Fitting of Time-Series Models. *Review of the International Statistical Institute*, 28(3), 233-244.
- Flandrin, P. (1999). *Time-Frequency/Time-Scale Analysis*. Academic Press.
- Gabor, D. (1946). Theory of Communication. Part 1: The Analysis of Information. *Journal of the Institution of Electrical Engineers*, 93(26), 429-441.
- Godard, D. N. (1974). Channel Equalization Using a Kalman Filter for Fast Data Transmission. *IBM Journal of Research and Development*, 18(3), 267-273.
- Hayes, M. H. (1996). *Statistical Digital Signal Processing and Modeling*. John Wiley & Sons.
- Haykin, S. (2013). *Adaptive Filter Theory* (5th ed.). Prentice Hall.
- IEEE Std 1057-2017. *IEEE Standard for Digitizing Waveform Recorders*.
- IEEE Std 181-2011. *IEEE Standard on Transitions, Pulses, and Related Waveforms*.
- Ingle, V. K., & Proakis, J. G. (2016). *Digital Signal Processing Using MATLAB* (3rd ed.). Cengage Learning.
- Jackson, L. B. (1996). *Digital Filters and Signal Processing* (3rd ed.). Kluwer Academic Publishers.
- Kay, S. M. (1988). *Modern Spectral Estimation: Theory and Application*. Prentice Hall.
- Kay, S. M. (1993). *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall.
- Levinson, N. (1947). The Wiener RMS (Root Mean Square) Error Criterion in Filter Design and Prediction. *Journal of Mathematics and Physics*, 25(4), 261-278.
- Mallat, S. (2008). *A Wavelet Tour of Signal Processing* (3rd ed.). Academic Press.
- McClellan, J. H., Schafer, R. W., & Yoder, M. A. (2015). *Signal Processing First*. Pearson.
- Mitra, S. K. (2017). *Digital Signal Processing: A Computer-Based Approach* (4th ed.). McGraw-Hill.
- Oppenheim, A. V., & Schafer, R. W. (2010). *Discrete-Time Signal Processing* (3rd ed.). Prentice Hall.
- Papoulis, A., & Pillai, S. U. (2002). *Probability, Random Variables and Stochastic Processes* (4th ed.). McGraw-Hill.
- Parks, T. W., & McClellan, J. H. (1972). Chebyshev Approximation for Nonrecursive Digital Filters with Linear Phase. *IEEE Transactions on Circuit Theory*, 19(2), 189-194.
- Pisarenko, V. F. (1973). The Retrieval of Harmonics from a Covariance Function. *Geophysical Journal International*, 33(3), 347-366.

Bibliography

- Proakis, J. G., & Manolakis, D. G. (2021). *Digital Signal Processing: Principles, Algorithms, and Applications* (5th ed.). Pearson.
- Rabiner, L. R., & Gold, B. (1975). *Theory and Application of Digital Signal Processing*. Prentice Hall.
- Schmidt, R. O. (1986). Multiple Emitter Location and Signal Parameter Estimation. *IEEE Transactions on Antennas and Propagation*, 34(3), 276-280.
- Stark, H., & Woods, J. W. (2001). *Probability and Random Processes with Applications to Signal Processing* (3rd ed.). Prentice Hall.
- Vetterli, M., & Kovačević, J. (1995). *Wavelets and Subband Coding*. Prentice Hall.
- Ville, J. (1948). Théorie et Applications de la Notion de Signal Analytique. *Câbles et Transmission*, 2A(1), 61-74.
- Widrow, B., & Hoff, M. E. (1960). Adaptive Switching Circuits. *IRE WESCON Convention Record*, 4, 96-104.
- Wigner, E. (1932). On the Quantum Correction for Thermodynamic Equilibrium. *Physical Review*, 40(5), 749-759.

Online Resources

- GNU Octave Documentation. *Signal Processing Packages*. Available at: <https://octave.org/>
- IEEE Signal Processing Society. *Conference Proceedings and Journals*. Available at: <https://signalprocessingsociety.org/>
- MathWorks Documentation. *MATLAB Signal Processing Toolbox User's Guide*. Available at: <https://www.mathworks.com/>
- SciPy Documentation. *Python Scientific Computing Library for Signal Processing*. Available at: <https://scipy.org/>

Journals

- IEEE Signal Processing Magazine*
- IEEE Transactions on Audio, Speech, and Language Processing*
- IEEE Transactions on Signal Processing*