Lectures 2 & 3: Introduction to Systems Notes

Konstantinos Chatzilygeroudis
costashatz@upatras.gr

3 March 2026

Contents

1 Systems

1.1 What is a System?

Definition 1.1. A **system** is a rule (or transformation) that maps an input signal to an output signal.

For continuous-time (CT) signals:

$$x(t) \longrightarrow y(t)$$

For discrete-time (DT) signals:

$$x[n] \longrightarrow y[n]$$

Using operator notation:

$$y = \mathcal{T}\{x\}$$

Here, $\mathcal{T}$ denotes the transformation (operator) that acts on the signal.

Interpretation

- A system is a *signal transformation*.
- It processes an input signal and produces an output signal.
- Mathematically, it is an *operator acting on functions*.

Signals are functions of time (or index), and systems are operators that act on those functions.

Examples of Systems

1. Amplifier

$$y(t) = 3x(t)$$

The output is a scaled version of the input.

2. Echo System

$$y(t) = x(t) + 0.5\,x(t-1)$$

The output consists of the original signal plus a delayed and attenuated copy.

3. Moving Average (Discrete-Time)

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

The output is the average of the current and previous two samples.

1.2 System Interconnections

In practice, complex systems are built by interconnecting simpler subsystems. The three fundamental interconnections are:

- Cascade (Series)
- Parallel
- Feedback

Cascade (Series) Connection

Two systems are connected in cascade when the output of the first is the input of the second:

$$x \longrightarrow \mathcal{T}_1 \longrightarrow v \longrightarrow \mathcal{T}_2 \longrightarrow y$$

Mathematically,

$$v = \mathcal{T}_1\{x\}$$
$$y = \mathcal{T}_2\{v\}$$

Therefore,

$$\boxed{y = \mathcal{T}_2\{\mathcal{T}_1\{x\}\}}$$

In general, the order of systems matters.

Parallel Connection

Two systems are connected in parallel when the same input is processed by both systems and their outputs are combined:

$$y_1 = \mathcal{T}_1\{x\} \qquad y_2 = \mathcal{T}_2\{x\}$$

$$\boxed{y = \mathcal{T}_1\{x\} + \mathcal{T}_2\{x\}}$$

Feedback Connection

In a feedback system, the output is processed and fed back to the input.

Let

$$u = x - v$$

Forward system:

$$y = \mathcal{T}_1\{u\}$$

Feedback path:

$$v = \mathcal{T}_2\{y\}$$

Thus,

$$\boxed{y = \mathcal{T}_1\{x - \mathcal{T}_2\{y\}\}}$$

Feedback plays a central role in control systems, recursive filters, and dynamical systems.

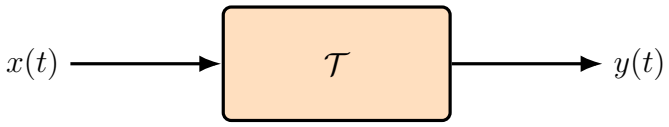
1.3 Graphical Representation of Systems

Systems and their interconnections are commonly represented using **block diagrams**. These diagrams visualize how signals flow through system components and how subsystems are interconnected.

Block diagrams are not merely illustrations — they encode structural information that allows us to derive mathematical relationships.

Single System Block

A system can be represented as a block that transforms the input into the output:

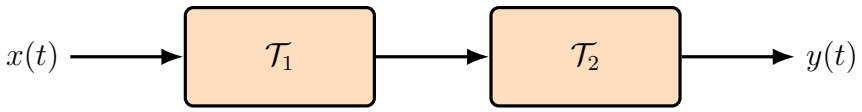


This represents the mapping:

$$y = \mathcal{T}\{x\}.$$

Cascade (Series) Connection

When systems are connected in series, the output of one system becomes the input of the next:

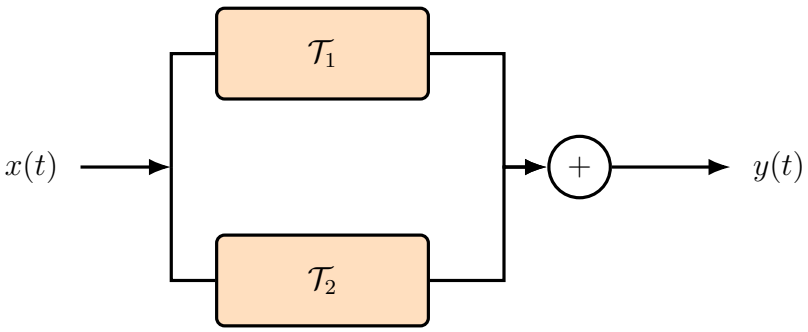


This corresponds to:

$$y = \mathcal{T}_2\{\mathcal{T}_1\{x\}\}.$$

Parallel Connection

In a parallel configuration, the same input is processed by multiple systems, and their outputs are combined at a summing junction:

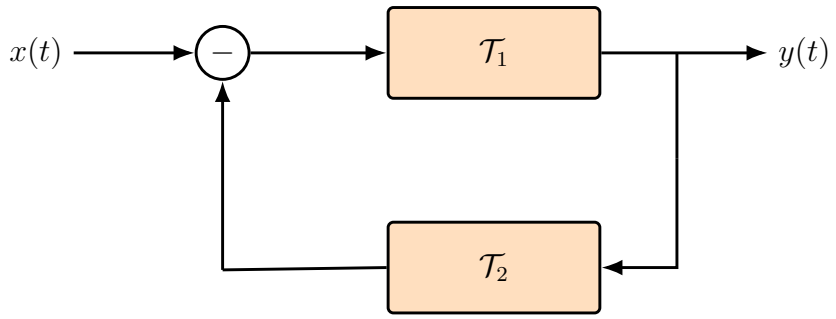


This represents:

$$y = \mathcal{T}_1\{x\} + \mathcal{T}_2\{x\}.$$

Feedback Connection

In feedback systems, the output is processed and fed back to modify the input through a summing junction:



This corresponds to:

$$y = \mathcal{T}_1\{x - \mathcal{T}_2\{y\}\}\,.$$

Why Block Diagrams Matter

Block diagrams:

- Clarify signal flow.
- Make complex systems easier to understand.
- Allow algebraic manipulation (block diagram reduction).
- Reveal structural properties (feedback, recursion, parallel paths).

1.4 Why Study Systems?

Real-world signal processing and control systems are rarely simple. Instead, they are built from simpler blocks interconnected in structured ways.

Understanding systems allows us to:

- Analyze complex behavior
- Simplify system structures
- Predict stability and response
- Design filters and controllers

Key Idea:

If we understand simple systems well, we can understand complex systems.

1.5 Taxonomy of Systems

A system maps an input signal to an output signal:

$$y = \mathcal{T}\{x\}.$$

Beyond this basic definition, systems are classified according to *how they behave* and *what structural assumptions we can make*. These classifications determine which mathematical tools can be used for analysis and design.

Common classification axes include:

- Deterministic vs. stochastic
- Single-/multi-input and single-/multi-output (SISO/MIMO)
- Static (memoryless) vs. dynamic (with memory)
- Causal vs. anti-causal vs. non-causal
- Linear vs. non-linear
- Time-invariant vs. time-varying

1.5.1 Deterministic vs. Stochastic Systems

Definition 1.2. A system is **deterministic** if, for a given input, the output is completely determined.

$$y = \mathcal{T}\{x\} \qquad \text{(no randomness involved)}$$

Definition 1.3. A system is **stochastic** (random) if the output is not fully predictable even for a fixed input.

Such systems can be written as:

$$y = \mathcal{T}\{x, \omega\},$$

where ω represents randomness (e.g., noise, uncertain parameters, random disturbances).

Examples

- Deterministic:
$$y[n] = 3x[n]$$
- Stochastic:
$$y[n] = x[n] + w[n],$$
where $w[n]$ is a noise process.

Deterministic systems are the primary focus of classical systems theory, while stochastic systems are studied in statistical signal processing and control.

1.5.2 SISO vs. MIMO Systems

Definition 1.4. A **SISO** (Single-Input Single-Output) system has one input signal and one output signal.

$$x \longrightarrow y$$

Definition 1.5. A **MIMO** (Multiple-Input Multiple-Output) system has multiple input signals and multiple output signals.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix} \quad \longrightarrow \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_P \end{bmatrix}$$

Examples

- SISO: audio filter (one waveform in, one waveform out)
- MIMO: stereo audio (2-in/2-out), multi-antenna communication systems

For linear systems, MIMO systems are often represented using matrices of operators.

1.5.3 Static (Memoryless) vs. Dynamic Systems

Definition 1.6. A system is **static** (or memoryless) if the output at time t (or index n) depends only on the input at that same time.

$$y(t) = f(x(t)), \qquad y[n] = f(x[n])$$

Definition 1.7. A system is **dynamic** (has memory) if the output depends on input and/or output values at times different from the present time.

$$y(t) \text{ depends on } x(\tau) \text{ or } y(\tau) \;\; \tau \neq t,$$
$$y[n] \text{ depends on } x[k] \text{ or } y[k] \;\; k \neq n.$$

Examples

- Memoryless:
$$y[n] = (x[n])^2$$
- With memory:
$$y[n] = x[n] + 0.5x[n-1]$$

Systems involving delays, integrals, derivatives, or sums over time necessarily have memory.

1.5.4 Causal, Anti-Causal, and Non-Causal Systems

Definition 1.8. A system is **causal** if the output at time t_0 depends only on input and/or output values for $t \leq t_0$.

Continuous-time:

$$y(t_0) \text{ depends only on } x(t) \text{ or } y(t),\ t \leq t_0.$$

Discrete-time:

$$y[n_0] \text{ depends only on } x[n] \text{ or } y[n],\ n \leq n_0.$$

Definition 1.9. A system is **anti-causal** if the output depends only on future inputs and/or outputs.

Definition 1.10. A system is **non-causal** if the output depends on both past and future inputs and/or outputs.

Examples

- Causal:
$$y[n] = x[n] + x[n-1]$$
- Anti-causal:
$$y[n] = x[n+1]$$
- Non-causal:
$$y[n] = x[n+1] + x[n-1]$$

Physical real-time systems must be causal. Offline processing (e.g., image processing) may use non-causal systems.

1.5.5 Linear vs. Non-Linear Systems

Definition 1.11. A system is **linear** if it satisfies the superposition principle:

$$\mathcal{T}\left\{\sum_i a_i x_i\right\} = \sum_i a_i \mathcal{T}\{x_i\}.$$

Equivalently, linearity consists of:

- Additivity
- Homogeneity (scaling)

Examples

- Linear:
$$y[n] = 2x[n] - x[n-1]$$
- Non-linear:
$$y[n] = (x[n])^2$$
$$y[n] = |x[n]|$$

Practical tip: Use the superposition test to check linearity.

1.5.6 Time-Invariant vs. Time-Varying Systems

Definition 1.12. A system is **time-invariant** if shifting the input by t_0 shifts the output by the same amount.

Continuous-time:

$$x(t) \to y(t) \quad \Rightarrow \quad x(t - t_0) \to y(t - t_0).$$

Discrete-time:

$$x[n - n_0] \to y[n - n_0].$$

Definition 1.13. If this property does not hold, the system is **time-varying**.

Examples

- Time-invariant:
$$y(t) = x(t - 1)$$
- Time-varying:
$$y(t) = t\, x(t)$$

Time invariance is crucial because it allows the use of convolution, Fourier transforms, and powerful structural results.

1.6 Systems Defined by Differential and Difference Equations

Many important systems in engineering and physics are not defined by an explicit input-output formula, but instead by an equation that relates the input and output signals.

Continuous-Time Systems: Differential Equations

Physical continuous-time systems are often described by **linear differential equations** relating the input $x(t)$ and the output $y(t)$:

$$a_N \frac{d^N y(t)}{dt^N} + a_{N-1} \frac{d^{N-1} y(t)}{dt^{N-1}} + \cdots + a_1 \frac{dy(t)}{dt} + a_0 y(t) = b_M \frac{d^M x(t)}{dt^M} + \cdots + b_1 \frac{dx(t)}{dt} + b_0 x(t).$$

Compactly, this can be written as:

$$\sum_{k=0}^{N} a_k \frac{d^k y(t)}{dt^k} = \sum_{k=0}^{M} b_k \frac{d^k x(t)}{dt^k}.$$

Interpretation

- The output depends on the input and its derivatives.
- The output may also depend on its own derivatives.
- The highest derivative of $y(t)$ determines the **order** of the system.

If the coefficients a_k and b_k are constant, the system is linear and time-invariant (LTI).

Example: RC Circuit A simple RC low-pass filter satisfies:

$$RC\frac{dy(t)}{dt} + y(t) = x(t).$$

This is a first-order linear differential equation. The system order is 1 because the highest derivative of $y(t)$ is first order.

Discrete-Time Systems: Difference Equations

Discrete-time systems are commonly described by **difference equations**, which relate shifted versions of the signals.

A general linear difference equation has the form:

$$a_N y[n-N]+a_{N-1}y[n-N+1]+\cdots+a_1 y[n-1]+a_0 y[n] = b_M x[n-M]+\cdots+b_1 x[n-1]+b_0 x[n].$$

Compactly:

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k].$$

Interpretation

- The output depends on current and past inputs.
- The output may depend on past outputs.
- The largest delay of $y[n]$ determines the **order** of the system.

If a_k and b_k are constants, the system is linear and time-invariant.

Example: Recursive Filter

$$y[n] = x[n] + 0.5\, y[n-1].$$

This is a first-order recursive (IIR) system, since the output depends on a previous output value.

Remark: Why These Representations Matter

Differential and difference equations:

- Naturally describe physical systems.
- Reveal system order and internal memory.
- Lead to transfer functions via Laplace and Z transforms.
- Provide a bridge to convolution representations for LTI systems.

In fact, every linear time-invariant system described by such an equation can also be represented using an impulse response and convolution (we will see this later).

1.7 Stability

One of the most fundamental properties of a system is **stability**. Intuitively, a stable system does not produce arbitrarily large outputs from finite inputs.

Bounded Signals

Definition 1.14. A signal is **bounded** if there exists a finite constant $M < \infty$ such that

$$|x(t)| \leq M \quad \forall t \qquad \text{(continuous-time)}$$

or

$$|x[n]| \leq M \quad \forall n \qquad \text{(discrete-time)}.$$

That is, the signal amplitude never exceeds some finite bound.

BIBO Stability

Definition 1.15. A system is **Bounded-Input Bounded-Output (BIBO) stable** if every bounded input produces a bounded output.

Formally,

$$\text{if } |x| \leq M_x < \infty \quad \Rightarrow \quad |y| \leq M_y < \infty.$$

Interpretation

- A finite input should not produce an infinite output.
- Physically, the system does not "blow up."
- Stability ensures predictable and safe behavior.

Examples

Example 1: Stable Static System

$$y[n] = 0.5\, x[n].$$

If $|x[n]| \leq M$, then

$$|y[n]| = 0.5|x[n]| \leq 0.5M.$$

Thus, the system is BIBO stable.

Example 2: Unstable Time-Varying System

$$y[n] = n\, x[n].$$

If $x[n] = 1$ (bounded), then

$$y[n] = n,$$

which grows without bound as $n \to \infty$. The system is therefore unstable.

Example 3: Stable Recursive System

$$y[n] = x[n] + 0.5\, y[n-1].$$

The feedback coefficient 0.5 has magnitude less than 1, so past outputs decay over time. The system is stable.

Example 4: Unstable Recursion

$$y[n] = x[n] + 1.5\, y[n-1].$$

Here, the feedback coefficient has magnitude greater than 1. Even with bounded input, the output grows exponentially. The system is unstable.

Why One Recursion Converges and the Other Diverges

Consider the first-order recursive system

$$y[n] = x[n] + a\, y[n-1],$$

with constant $a \in \mathbb{R}$ (or $a \in \mathbb{C}$). We show that the long-term behavior is governed by $|a|$.

Step 1: Expand the recursion (iteration). Substitute $y[n-1]$ repeatedly:

$$\begin{aligned} y[n] &= x[n] + ay[n-1] \\ &= x[n] + a\big(x[n-1] + ay[n-2]\big) \\ &= x[n] + ax[n-1] + a^2 y[n-2] \\ &\;\;\vdots \\ &= \sum_{k=0}^{n} a^k\, x[n-k] \;+\; a^{n+1} y[-1]. \end{aligned}$$

(Here $y[-1]$ denotes the initial condition. For a general initial time n_0, the last term becomes $a^{n-n_0+1} y[n_0-1]$.)

Step 2: Use bounded input. Assume the input is bounded:

$$|x[n]| \le M \quad \forall n.$$

Then

$$\left| \sum_{k=0}^{n} a^k\, x[n-k] \right| \le \sum_{k=0}^{n} |a|^k\, |x[n-k]| \le M \sum_{k=0}^{n} |a|^k.$$

Case A: $|a| < 1$ (convergent / stable). If $|a| < 1$, the geometric series is bounded:

$$\sum_{k=0}^{n} |a|^k \leq \sum_{k=0}^{\infty} |a|^k = \frac{1}{1-|a|}.$$

Hence,

$$\left| \sum_{k=0}^{n} a^k \, x[n-k] \right| \leq \frac{M}{1-|a|}.$$

Also, the initial-condition term decays to zero:

$$|a^{n+1} y[-1]| = |a|^{n+1} |y[-1]| \xrightarrow[n\to\infty]{} 0.$$

Therefore,

$$|y[n]| \leq \frac{M}{1-|a|} + |a|^{n+1} |y[-1]|$$

is bounded for all n. This proves BIBO stability for $|a| < 1$. In particular, for $a = 0.5$,

$$y[n] = x[n] + 0.5\, y[n-1]$$

is stable.

Case B: $|a| > 1$ (divergent / unstable). If $|a| > 1$, we can show unbounded output for a bounded input by choosing

$$x[n] \equiv 1 \quad \text{(bounded)}.$$

With zero initial condition $y[-1] = 0$, the expansion becomes

$$y[n] = \sum_{k=0}^{n} a^k = \frac{a^{n+1} - 1}{a - 1}.$$

If $a > 1$, then $a^{n+1} \to \infty$, so $y[n] \to \infty$ (unbounded). If $a < -1$, then $|a^{n+1}| \to \infty$ and $y[n]$ grows in magnitude while alternating sign, so $|y[n]| \to \infty$ (also unbounded). Thus the system is not BIBO stable for $|a| > 1$.

In particular, for $a = 1.5$,

$$y[n] = x[n] + 1.5\, y[n-1]$$

is unstable.

Remark: Boundary case $|a| = 1$. When $|a| = 1$, the geometric sum does not decay. For example, if $a = 1$ and $x[n] \equiv 1$,

$$y[n] = \sum_{k=0}^{n} 1 = n + 1,$$

which is unbounded, hence unstable. (Similar counterexamples exist for $a = -1$.)

Conclusion: The recursion $y[n] = x[n] + ay[n-1]$ is BIBO stable if and only if $|a| < 1$.

Physical Intuition

- Stable systems dissipate or limit energy.
- Unstable systems amplify energy without bound.

In continuous-time systems, instability often corresponds to exponentially growing solutions. In discrete-time systems, instability corresponds to exponential growth across time steps.

Engineering Perspective

Stability is essential in:

- Control systems
- Communication systems
- Signal processing filters

An unstable system may lead to oscillations, saturation, or physical damage in real-world applications.

1.8 Worked Examples: Classifying Systems

We now classify several systems according to the taxonomy introduced earlier. For each system, we examine memory, causality, linearity, time invariance, and stability (when applicable).

Example 1

$$y[n] = 3x[n]$$

Memory Memoryless: the output depends only on $x[n]$.

Causality Causal: no future inputs are used.

Linearity Linear:

$$\mathcal{T}\{ax_1 + bx_2\} = 3(ax_1 + bx_2) = a(3x_1) + b(3x_2).$$

Time Invariance Time-invariant: shifting the input shifts the output identically.

Stability BIBO stable:

$$|y[n]| = 3|x[n]| \leq 3M.$$

Conclusion: Linear, time-invariant, memoryless, causal, stable (LTI).

Example 2

$$y[n] = x[n] + x[n-1]$$

Memory Has memory: depends on a past input.

Causality Causal: depends only on present and past values.

Linearity Linear (sum of shifted inputs).

Time Invariance Time-invariant: a shift in input produces the same shift in output.

Stability BIBO stable:

$$|y[n]| \leq |x[n]| + |x[n-1]| \leq 2M.$$

Conclusion: Linear, time-invariant, causal, stable (dynamic LTI).

Example 3

$$y[n] = n\,x[n]$$

Memory Memoryless.

Causality Causal.

Linearity Linear:

$$\mathcal{T}\{ax_1 + bx_2\} = n(ax_1 + bx_2) = a(nx_1) + b(nx_2).$$

Time Invariance Time-varying:

$$x[n-n_0] \to nx[n-n_0] \neq (n-n_0)x[n-n_0].$$

Stability Unstable: if $x[n] = 1$, then $y[n] = n$ (unbounded).

Conclusion: Linear, time-varying, causal, unstable.

Example 4

$$y[n] = (x[n])^2$$

Memory Memoryless.

Causality Causal.

Linearity Nonlinear:

$$(x_1 + x_2)^2 \neq x_1^2 + x_2^2.$$

Time Invariance Time-invariant (no explicit dependence on n).

Stability BIBO stable:

$$|y[n]| = |x[n]|^2 \leq M^2.$$

Conclusion: Nonlinear, time-invariant, causal, stable.

Example 5

$$y[n] = x[n+1]$$

Memory Has memory (depends on another time index).

Causality Anti-causal (depends on future input).

Linearity Linear.

Time Invariance Time-invariant (pure shift).

Stability BIBO stable:

$$|y[n]| = |x[n+1]| \leq M.$$

Conclusion: Linear, time-invariant, anti-causal, stable.

Example 6

$$y[n] = x[n] + 0.5\, y[n-1]$$

Memory Dynamic (recursive).

Causality Causal.

Linearity Linear (can verify via superposition).

Time Invariance Time-invariant (coefficients constant).

Stability Stable since $|0.5| < 1$ (see recursion proof).

Conclusion: Linear, time-invariant, causal, stable (recursive LTI).

Summary Table

System	Linear	TI	Causal	Memory	Stable
$3x[n]$	Yes	Yes	Yes	No	Yes
$x[n] + x[n-1]$	Yes	Yes	Yes	Yes	Yes
$nx[n]$	Yes	No	Yes	No	No
$(x[n])^2$	No	Yes	Yes	No	Yes
$x[n+1]$	Yes	Yes	No	Yes	Yes
$x[n] + 0.5y[n-1]$	Yes	Yes	Yes	Yes	Yes

2 Linear Time-Invariant (LTI) Systems

Linear Time-Invariant (LTI) systems form the most important class of systems in signals and systems theory. They combine two fundamental properties: linearity and time invariance.

2.1 Definition of an LTI System

Definition 2.1. A system is **Linear Time-Invariant (LTI)** if it satisfies:

1. **Linearity** (Superposition)
2. **Time Invariance**

Linearity

For any signals x_1, x_2 and scalars a, b:

$$\mathcal{T}\{ax_1 + bx_2\} = a\,\mathcal{T}\{x_1\} + b\,\mathcal{T}\{x_2\}.$$

Linearity consists of:

- Additivity
- Homogeneity (scaling)

Time Invariance

If

$$x(t) \to y(t),$$

then

$$x(t - t_0) \to y(t - t_0).$$

Similarly for discrete-time:

$$x[n - n_0] \to y[n - n_0].$$

The system behavior does not depend on when the input is applied.

2.2 Intuition Behind LTI Systems

Linearity

- Scaling the input scales the output.
- Adding inputs produces the sum of outputs.

Time Invariance

- The system behaves the same at all times.
- There is no explicit dependence on time.

Engineering Interpretation: LTI systems are predictable, composable, and mathematically tractable.

Many physical systems are approximately LTI:

- Electrical circuits
- Mechanical systems
- Communication channels

2.3 Total Response of Linear Systems

For linear systems described by differential or difference equations, the total response can be decomposed into two components:

$$\textbf{Total Response} = \textbf{Zero-Input Response} + \textbf{Zero-State Response}.$$

This decomposition follows from linearity.

2.4 Zero-Input (Natural) Response

The **zero-input response** is obtained by setting the input to zero:

$$x(t) = 0 \qquad \text{or} \qquad x[n] = 0.$$

We then solve the corresponding homogeneous equation.

Example (RC Circuit) Consider the first-order differential equation:

$$RC\,\frac{dy(t)}{dt} + y(t) = 0.$$

The solution is:

$$y(t) = Ae^{-t/RC}.$$

The constant A depends on the initial condition.

Interpretation The zero-input response is determined entirely by stored energy (initial conditions), such as:

- Initial capacitor voltage
- Initial inductor current

It is also called the **natural response**.

2.5 Zero-State (Forced) Response

The **zero-state response** is obtained by setting all initial conditions to zero and solving the system equation using only the input.

Example (First-Order Discrete-Time System)

$$y[n] = x[n] + 0.5\, y[n-1].$$

If

$$y[-1] = 0,$$

the output depends only on the input signal.

Interpretation The zero-state response is the part of the output caused directly by the input signal. It is also called the **forced response**.

Why the Decomposition Matters

The separation

$$y[n] = y_{\mathrm{ZI}}[n] + y_{\mathrm{ZS}}[n]$$

is possible because the system is linear.
For LTI systems:

- The zero-input response is determined by the system poles.
- The zero-state response will lead us to the concept of **impulse response** and **convolution**.

In fact:

The zero-state response of an LTI system can be written as a convolution.

This fundamental result is developed next.

2.6 Impulse Response

Definition 2.2. The **impulse response** of a system is the output produced when the input is an impulse:

Continuous-time:

$$x(t) = \delta(t) \quad \Rightarrow \quad y(t) = h(t)$$

Discrete-time:

$$x[n] = \delta[n] \quad \Rightarrow \quad y[n] = h[n].$$

Important:

- Initial conditions are assumed zero.
- The impulse response is a zero-state response.

Why the Impulse Response Is So Important

For **LTI systems**:

The impulse response completely characterizes the system.

Once $h(t)$ or $h[n]$ is known, the output for any input can be computed (using convolution, which we derive next).

2.7 Worked Examples: Computing the Impulse Response

We now compute impulse responses for several systems.

Example 1: Memoryless System

$$y[n] = 3x[n]$$

Apply impulse input:

$$x[n] = \delta[n].$$

Then

$$h[n] = y[n] = 3\delta[n].$$

Observation Memoryless systems have impulse responses that are scaled impulses.

Example 2: FIR System

$$y[n] = x[n] + 2x[n-1].$$

Apply impulse:

$$x[n] = \delta[n].$$

Then

$$h[n] = \delta[n] + 2\delta[n-1].$$

Interpretation The impulse response directly reveals the system coefficients.

Example 3: First-Order Recursive System

$$y[n] = x[n] + a\,y[n-1], \qquad |a| < 1.$$

We compute the impulse response by setting:

$$x[n] = \delta[n], \qquad y[-1] = 0 \quad \text{(zero initial condition)}.$$

We now compute $h[n]$ step-by-step.

Step 1: Evaluate at $n = 0$

$$h[0] = \delta[0] + a\,h[-1].$$

Since $\delta[0] = 1$ and $h[-1] = 0$:

$$h[0] = 1.$$

Step 2: Evaluate at $n = 1$

$$h[1] = \delta[1] + a\,h[0].$$

Since $\delta[1] = 0$:

$$h[1] = a.$$

Step 3: Evaluate at $n = 2$

$$h[2] = \delta[2] + a\,h[1].$$

$$h[2] = a^2.$$

Continuing the recursion We observe the pattern:

$$h[n] = a^n, \quad n \geq 0.$$

For $n < 0$, causality implies:

$$h[n] = 0.$$

Thus,

$$\boxed{h[n] = a^n u[n]}$$

where $u[n]$ is the unit step.

Alternative Derivation (Closed-Form Solution)

We can also derive this formally.

For $n > 0$, the impulse $\delta[n] = 0$, so the equation becomes homogeneous:

$$h[n] = a\, h[n-1].$$

This is a first-order difference equation with solution:

$$h[n] = Ca^n.$$

Using the initial condition $h[0] = 1$:

$$C = 1.$$

Thus,

$$h[n] = a^n u[n].$$

Observation Recursive systems produce exponentially decaying impulse responses (if $|a| < 1$).

Example 4: Continuous-Time First-Order System

Consider the RC circuit:

$$RC\,\frac{dy(t)}{dt} + y(t) = x(t).$$

Set:

$$x(t) = \delta(t).$$

We solve:

$$RC\,\frac{dh(t)}{dt} + h(t) = \delta(t).$$

For $t > 0$, the equation becomes homogeneous:

$$RC\,\frac{dh(t)}{dt} + h(t) = 0.$$

Solution:

$$h(t) = Ae^{-t/RC}, \quad t > 0.$$

To determine A, integrate the differential equation around $t = 0$:

$$\int_{-\epsilon}^{+\epsilon} \left(RC\,\frac{dh}{dt} + h(t) \right) dt = \int_{-\epsilon}^{+\epsilon} \delta(t)\, dt = 1.$$

The derivative term dominates, giving:

$$RC\big(h(0^+) - h(0^-)\big) = 1.$$

Assuming zero initial condition:

$$h(0^-) = 0,$$

we obtain:

$$h(0^+) = \frac{1}{RC}.$$

Thus,

$$\boxed{h(t) = \frac{1}{RC} e^{-t/RC} u(t)}$$

Interpretation The impulse response is an exponentially decaying function. The decay rate is determined by the system time constant.

Key Observations

- Memoryless systems $\rightarrow$ scaled impulse.
- FIR systems $\rightarrow$ finite combination of shifted impulses.
- Stable recursive systems $\rightarrow$ decaying exponentials.
- Continuous-time first-order systems $\rightarrow$ exponential impulse response.

Next Step: How does the impulse response allow us to compute the output for any input?

This leads directly to **convolution**.

2.8 Deriving Convolution from LTI Systems

We now derive one of the most fundamental results in signals and systems:

The output of an LTI system is the convolution of the input with the impulse response.

We first derive the continuous-time result and then give the discrete-time analogue.

Continuous-Time Derivation

Suppose we have an LTI system:

$$y(t) = \mathcal{T}\{x(t)\}.$$

Step 1: Decompose the input into impulses Any well-behaved continuous-time signal can be written as a weighted superposition of shifted impulses:

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

This representation expresses $x(t)$ as an infinite sum of shifted impulses, each weighted by $x(\tau)$.

Step 2: Define the impulse response By definition,

$$\delta(t) \;\longrightarrow\; h(t) = \mathcal{T}\{\delta(t)\}.$$

Step 3: Use time invariance If

$$\delta(t) \to h(t),$$

then shifting the input by τ shifts the output by the same amount:

$$\delta(t-\tau) \;\longrightarrow\; h(t-\tau).$$

Step 4: Use linearity (superposition) Since

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

applying the system gives:

$$\begin{aligned} y(t) &= \mathcal{T}\left\{\int_{-\infty}^{\infty} x(\tau)\,\delta(t-\tau)\,d\tau\right\} \\ &= \int_{-\infty}^{\infty} x(\tau)\,\mathcal{T}\{\delta(t-\tau)\}\,d\tau. \end{aligned}$$

Using Step 3:

$$\mathcal{T}\{\delta(t-\tau)\} = h(t-\tau).$$

Therefore,

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

This is the **convolution integral**.

We write:

$$y(t) = (x * h)(t).$$

Discrete-Time Derivation

The discrete-time argument is completely analogous.

Any discrete-time signal can be written as:

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\,\delta[n-k].$$

By definition,

$$\delta[n] \to h[n].$$

By time invariance,

$$\delta[n-k] \to h[n-k].$$

By linearity,

$$\begin{aligned} y[n] &= \mathcal{T}\left\{\sum_{k=-\infty}^{\infty} x[k]\,\delta[n-k]\right\} \\ &= \sum_{k=-\infty}^{\infty} x[k]\,\mathcal{T}\{\delta[n-k]\} \\ &= \sum_{k=-\infty}^{\infty} x[k]\,h[n-k]. \end{aligned}$$

Thus,

$$\boxed{y[n] = (x * h)[n] = \sum_{k=-\infty}^{\infty} x[k]\,h[n-k]}$$

2.9 How to Compute Convolution (Mechanics)

To compute

$$y(t) = \int x(\tau)\,h(t-\tau)\,d\tau,$$

we follow four steps:

1. **Flip** one signal:
$$h(\tau) \to h(-\tau)$$
2. **Shift** it by t:
$$h(-\tau) \to h(t-\tau)$$
3. **Multiply** with $x(\tau)$
4. **Integrate** over τ

For discrete-time systems, replace integration with summation.

Geometric Interpretation

Convolution measures the *overlap* between:

- the input signal, and
- a flipped and shifted version of the impulse response.

As t changes, the overlap changes — and this produces the output.

Key Insight

An LTI system is completely characterized by its impulse response.

Linearity + time invariance $\Longrightarrow$ Convolution representation.

2.10 Stability of LTI Systems

For general systems, BIBO stability is defined as:

$$\text{bounded input} \;\Rightarrow\; \text{bounded output.}$$

For LTI systems, stability admits a very precise and powerful characterization.

Continuous-Time Case

For an LTI system,

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

Theorem 2.1 (BIBO Stability Condition — CT)**.** A continuous-time LTI system is BIBO stable if and only if its impulse response is absolutely integrable:

$$\boxed{\int_{-\infty}^{\infty} |h(t)|\, dt < \infty.}$$

Proof (Sufficiency) Assume the input is bounded:

$$|x(t)| \le M.$$

Then

$$\begin{aligned} |y(t)| &= \left| \int_{-\infty}^{\infty} x(\tau)\, h(t-\tau)\, d\tau \right| \\ &\le \int_{-\infty}^{\infty} |x(\tau)|\, |h(t-\tau)|\, d\tau \\ &\le M \int_{-\infty}^{\infty} |h(t-\tau)|\, d\tau. \end{aligned}$$

Since shifting does not change the integral:

$$\int_{-\infty}^{\infty} |h(t-\tau)|\, d\tau = \int_{-\infty}^{\infty} |h(\lambda)|\, d\lambda.$$

If this integral is finite, say equal to H, then:

$$|y(t)| \leq MH,$$

which is bounded. Therefore, the system is stable.

Discrete-Time Case

For discrete-time LTI systems:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]\, h[n-k].$$

Theorem 2.2 (BIBO Stability Condition — DT)**.** A discrete-time LTI system is BIBO stable if and only if its impulse response is absolutely summable:

$$\boxed{\sum_{n=-\infty}^{\infty} |h[n]| < \infty.}$$

Proof (Sufficiency) Assume

$$|x[n]| \leq M.$$

Then

$$\begin{aligned} |y[n]| &= \left| \sum_{k=-\infty}^{\infty} x[k]\, h[n-k] \right| \\ &\leq \sum_{k=-\infty}^{\infty} |x[k]|\, |h[n-k]| \\ &\leq M \sum_{k=-\infty}^{\infty} |h[n-k]|. \end{aligned}$$

Since summation is invariant under shifts:

$$\sum_{k=-\infty}^{\infty} |h[n-k]| = \sum_{m=-\infty}^{\infty} |h[m]|.$$

If this sum is finite, say equal to H, then:

$$|y[n]| \leq MH,$$

which is bounded.

Examples

Example 1: Stable First-Order System

$$h[n] = a^n u[n], \quad |a| < 1.$$

Then:

$$\sum_{n=0}^{\infty} |a|^n = \frac{1}{1-|a|} < \infty.$$

Thus the system is stable.

Example 2: Unstable First-Order System

$$h[n] = a^n u[n], \quad |a| > 1.$$

Then

$$\sum_{n=0}^{\infty} |a|^n = \infty.$$

Thus the system is unstable.

Physical Interpretation

- If $h(t)$ decays sufficiently fast, the system is stable.
- If $h(t)$ does not decay (or grows), the system is unstable.

The impulse response represents how the system reacts to a unit impulse. If that reaction has finite total magnitude, the system cannot amplify bounded inputs into unbounded outputs.

Key Result:

Stability of an LTI system is completely determined by its impulse response.

2.11 Connection to Differential and Difference Equations

We have seen that LTI systems can be described in two equivalent ways:

1. By a differential or difference equation.
2. By an impulse response and convolution.

We now show how these viewpoints are connected.

Continuous-Time Case

Consider a linear constant-coefficient differential equation:

$$\sum_{k=0}^{N} a_k \frac{d^k y(t)}{dt^k} = \sum_{k=0}^{M} b_k \frac{d^k x(t)}{dt^k}.$$

Assume zero initial conditions (zero-state response).

Step 1: Apply impulse input To find the impulse response, set:

$$x(t) = \delta(t).$$

Then $h(t)$ satisfies:

$$\sum_{k=0}^{N} a_k \frac{d^k h(t)}{dt^k} = \sum_{k=0}^{M} b_k \frac{d^k \delta(t)}{dt^k}.$$

Thus, the impulse response is the solution of the system equation with impulse input and zero initial conditions.

Step 2: General input For an arbitrary input $x(t)$, we previously derived:

$$y(t) = (x * h)(t).$$

Therefore:

Solving the differential equation $\iff$ Convolving the input with the impulse response.

Discrete-Time Case

Consider the linear constant-coefficient difference equation:

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k].$$

With zero initial conditions, the impulse response is obtained by setting:

$$x[n] = \delta[n].$$

Thus,

$$\sum_{k=0}^{N} a_k h[n-k] = \sum_{k=0}^{M} b_k \delta[n-k].$$

Once $h[n]$ is determined, the response to any input is:

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

Key Insight

For LTI systems with constant coefficients:

- The differential/difference equation describes the *local dynamics*.
- The impulse response describes the *global behavior*.
- Convolution gives the complete input-output relationship.

Example: First-Order Discrete-Time System

$$y[n] = x[n] + a\, y[n-1].$$

We previously found:

$$h[n] = a^n u[n].$$

Thus, the solution for any input is:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] a^{n-k} u[n-k].$$

This convolution expression is equivalent to solving the difference equation recursively.

3 Convolution

Convolution is the fundamental operation that describes the input–output relationship of LTI systems. Beyond systems theory, it is also a central operation in signal processing, probability, differential equations, and machine learning.

3.1 Convolution in Continuous and Discrete Time

Discrete-Time Convolution

$$y[n] = (x * h)[n] = \sum_{k=-\infty}^{\infty} x[k]\, h[n-k].$$

Continuous-Time Convolution

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

Common Structure

Both definitions follow the same structure:

- Multiply one signal with a shifted version of the other.
- Sum (DT) or integrate (CT).
- The output depends on the overlap between the two signals.

3.2 Equivalent Forms of Convolution

Discrete-Time

Starting from:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]\, h[n-k],$$

let $m = n - k$. Then:

$$y[n] = \sum_{m=-\infty}^{\infty} h[m]\, x[n-m].$$

Continuous-Time

Starting from:

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

let $\lambda = t - \tau$. Then:

$$y(t) = \int_{-\infty}^{\infty} h(\lambda)\, x(t-\lambda)\, d\lambda.$$

Key Insight

- Either signal can be viewed as the shifted one.
- Convolution is symmetric in x and h.

In fact,

$$\boxed{x * h = h * x}$$

3.3 How to Compute Convolution

Continuous-Time Convolution Mechanics

To compute

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

we follow four steps:

1. **Flip** one signal:
$$h(\tau) \to h(-\tau)$$
2. **Shift** it by t:
$$h(-\tau) \to h(t-\tau)$$
3. **Multiply** with $x(\tau)$
4. **Integrate** over τ

Geometric Interpretation

Convolution measures the *overlapping area* between $x(\tau)$ and a shifted, flipped version of $h(\tau)$.

As t varies, the overlap changes — producing $y(t)$.

Discrete-Time Convolution Mechanics

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]\, h[n-k].$$

To compute $y[n]$:

1. Flip: $h[k] \to h[-k]$
2. Shift: $h[-k] \to h[n-k]$
3. Multiply with $x[k]$
4. Sum over all k

Interpretation

$y[n]$ is a sliding weighted sum of the input. The weights are given by the impulse response.

Length of Convolution (Discrete-Time)

Assume:

$$x[n] \neq 0 \quad \text{for } n = n_x^{\min}, \ldots, n_x^{\max}$$

$$h[n] \neq 0 \quad \text{for } n = n_h^{\min}, \ldots, n_h^{\max}$$

Define lengths:

$$L_x = n_x^{\max} - n_x^{\min} + 1, \qquad L_h = n_h^{\max} - n_h^{\min} + 1.$$

The convolution $y[n] = (x * h)[n]$ is nonzero for:

$$n_y^{\min} = n_x^{\min} + n_h^{\min}, \qquad n_y^{\max} = n_x^{\max} + n_h^{\max}.$$

Therefore,

$$\boxed{L_y = L_x + L_h - 1.}$$

Interpretation

Convolution increases signal length. Each new shift produces additional overlap.

Conceptual Summary

- Convolution combines two signals through weighted overlap.
- It is the natural operation of LTI systems.
- It is symmetric.
- For finite-length signals, output length grows.

3.4 When Does Convolution Not Make Sense?

Convolution is defined by:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]\, h[n-k] \quad \text{(DT)}$$

$$y(t) = \int_{-\infty}^{\infty} x(\tau)\, h(t-\tau)\, d\tau \quad \text{(CT)}$$

However, these expressions only make sense if the sum or integral *converges*.

Discrete-Time Case

The convolution sum must converge for every n.

Convolution may fail to exist if:

- The series diverges.
- The terms do not decay sufficiently.
- The series is not absolutely summable.

Sufficient Condition If both signals are absolutely summable:

$$\sum_{n=-\infty}^{\infty} |x[n]| < \infty, \qquad \sum_{n=-\infty}^{\infty} |h[n]| < \infty,$$

then their convolution exists and is absolutely summable.

More generally, if one signal is absolutely summable and the other is bounded, convolution also exists.

Continuous-Time Case

The convolution integral must converge for every t.

Convolution may fail if:

- The integral diverges.
- The signals are not absolutely integrable.

Sufficient Condition If

$$\int_{-\infty}^{\infty} |x(t)|\,dt < \infty, \qquad \int_{-\infty}^{\infty} |h(t)|\,dt < \infty,$$

then the convolution exists and is absolutely integrable.

More generally, if one signal is absolutely integrable and the other is bounded, the convolution exists.

Examples Where Convolution Fails

Discrete-Time Example

Let

$$x[n] = 1, \qquad h[n] = 1.$$

Then

$$y[n] = \sum_{k=-\infty}^{\infty} 1.$$

This sum diverges to infinity.

Convolution does not exist.

Continuous-Time Example

Let

$$x(t) = 1, \qquad h(t) = 1.$$

Then

$$y(t) = \int_{-\infty}^{\infty} 1\,d\tau,$$

which diverges.

The convolution integral does not exist.

Important Clarification

Convolution may fail in the classical (Lebesgue/Riemann) sense, even though we can sometimes interpret it using **generalized functions (distributions)**.

For example:

$$x(t) = 1, \qquad h(t) = u(t)$$

produces

$$y(t) = \int_{-\infty}^{t} 1 \, d\tau,$$

which diverges as $t \to \infty$, but is finite for each fixed t. Thus convolution may exist pointwise but not be bounded.

Connection to Stability

For LTI systems:

$$y[n] = x[n] * h[n].$$

If $h[n]$ is not absolutely summable, even bounded inputs may produce unbounded outputs.

Thus:

Absolute summability / integrability of h $\Longleftrightarrow$ BIBO stability.

Summary

Convolution makes sense when:

- The sum (DT) or integral (CT) converges.
- Signals decay sufficiently fast.
- At least one signal is absolutely integrable/summable and the other is bounded.

Otherwise, convolution may diverge and the operation is not well-defined in the classical sense.

3.5 Worked Examples

This subsection contains worked convolution examples in both discrete time (DT) and continuous time (CT). In the DT examples we also compute the output support (min/max indices) and the resulting length.

Discrete-Time Example 1 (Finite-length sequences, explicit computation)

Let

$$x[n] = \begin{cases} 1, & n = 0 \\ 2, & n = 1 \\ 1, & n = 2 \\ 0, & \text{otherwise} \end{cases} \qquad h[n] = \begin{cases} 1, & n = 0 \\ -1, & n = 1 \\ 0, & \text{otherwise.} \end{cases}$$

Step 1: Support and length Here,

$$n_x^{\min} = 0,\ n_x^{\max} = 2 \ \Rightarrow\ L_x = 2 - 0 + 1 = 3,$$
$$n_h^{\min} = 0,\ n_h^{\max} = 1 \ \Rightarrow\ L_h = 1 - 0 + 1 = 2.$$

Thus the convolution support is:

$$n_y^{\min} = n_x^{\min} + n_h^{\min} = 0, \qquad n_y^{\max} = n_x^{\max} + n_h^{\max} = 3,$$

and the output length is:

$$L_y = L_x + L_h - 1 = 3 + 2 - 1 = 4.$$

So we only need to compute $y[n]$ for $n = 0, 1, 2, 3$.

Step 2: Convolution sum

$$y[n] = (x * h)[n] = \sum_{k=-\infty}^{\infty} x[k]\, h[n-k].$$

Since $x[k]$ is nonzero only for $k \in \{0, 1, 2\}$ and $h[m]$ only for $m \in \{0, 1\}$, only a few terms contribute.

Compute each output sample

$$y[0] = \sum_k x[k]h[0-k] = x[0]h[0] = 1 \cdot 1 = 1.$$

$$y[1] = \sum_k x[k]h[1-k] = x[0]h[1] + x[1]h[0] = 1 \cdot (-1) + 2 \cdot 1 = 1.$$

$$y[2] = \sum_k x[k]h[2-k] = x[1]h[1] + x[2]h[0] = 2 \cdot (-1) + 1 \cdot 1 = -1.$$

$$y[3] = \sum_k x[k]h[3-k] = x[2]h[1] = 1 \cdot (-1) = -1.$$

Result

$$y[n] = \begin{cases} 1, & n = 0 \\ 1, & n = 1 \\ -1, & n = 2 \\ -1, & n = 3 \\ 0, & \text{otherwise.} \end{cases}$$

Discrete-Time Example 2 (Nonzero starting indices, support & length + computation)

Let

$$x[n] = \begin{cases} 1, & n = -1 \\ -1, & n = 0 \\ 2, & n = 1 \\ 0, & \text{otherwise} \end{cases} \qquad h[n] = \begin{cases} 2, & n = 0 \\ 1, & n = 1 \\ 1, & n = 2 \\ 0, & \text{otherwise.} \end{cases}$$

Step 1: Support and length

$$n_x^{\min} = -1,\ n_x^{\max} = 1 \Rightarrow L_x = 1 - (-1) + 1 = 3,$$

$$n_h^{\min} = 0,\ n_h^{\max} = 2 \Rightarrow L_h = 2 - 0 + 1 = 3.$$

Output support:

$$n_y^{\min} = -1 + 0 = -1, \qquad n_y^{\max} = 1 + 2 = 3,$$

Output length:

$$L_y = 3 + 3 - 1 = 5.$$

So compute $y[n]$ for $n = -1, 0, 1, 2, 3$.

Step 2: Convolution sum

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k],$$

where $x[k]$ is nonzero only for $k \in \{-1, 0, 1\}$.

Compute each output sample For $n = -1$:

$$y[-1] = x[-1]h[0] = 1 \cdot 2 = 2.$$

For $n = 0$:

$$y[0] = x[-1]h[1] + x[0]h[0] = 1 \cdot 1 + (-1) \cdot 2 = -1.$$

For $n = 1$:

$$y[1] = x[-1]h[2] + x[0]h[1] + x[1]h[0] = 1 \cdot 1 + (-1) \cdot 1 + 2 \cdot 2 = 4.$$

For $n = 2$:

$$y[2] = x[0]h[2] + x[1]h[1] = (-1) \cdot 1 + 2 \cdot 1 = 1.$$

For $n = 3$:

$$y[3] = x[1]h[2] = 2 \cdot 1 = 2.$$

Result

$$y[n] = \begin{cases} 2, & n = -1 \\ -1, & n = 0 \\ 4, & n = 1 \\ 1, & n = 2 \\ 2, & n = 3 \\ 0, & \text{otherwise.} \end{cases}$$

Continuous-Time Example 1 (Rectangular pulses → triangular output)

Let

$$x(t) = u(t) - u(t-1), \qquad h(t) = u(t) - u(t-1),$$

i.e., both are unit-height rectangular pulses of duration 1 on $[0, 1]$.

The convolution is:

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

Key idea (overlap length) Since $x(\tau) = 1$ when $\tau \in [0, 1]$ and $h(t-\tau) = 1$ when $t - \tau \in [0, 1]$, i.e. $\tau \in [t-1, t]$, the integrand equals 1 exactly over the overlap interval:

$$\tau \in [0, 1] \cap [t-1, t].$$

Therefore, $y(t)$ equals the length of this overlap.

Determine overlap by cases

- If $t < 0$: no overlap $\Rightarrow y(t) = 0$.
- If $0 \le t \le 1$: overlap is $[0, t]$ (length t) $\Rightarrow y(t) = t$.
- If $1 \le t \le 2$: overlap is $[t-1, 1]$ (length $2 - t$) $\Rightarrow y(t) = 2 - t$.
- If $t > 2$: no overlap $\Rightarrow y(t) = 0$.

Result

$$\boxed{y(t) = \begin{cases} 0, & t < 0, \\ t, & 0 \le t \le 1, \\ 2 - t, & 1 \le t \le 2, \\ 0, & t > 2. \end{cases}}$$

This is a triangular pulse on $[0, 2]$.

Continuous-Time Example 2 (Exponential input through an integrator)

Let

$$x(t) = e^{-t}u(t), \qquad h(t) = u(t).$$

Here $h(t) = u(t)$ corresponds to an ideal integrator in the sense that

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

Since $x(\tau) = 0$ for $\tau < 0$, we get for $t \geq 0$:

$$y(t) = \int_{0}^{t} e^{-\tau}\, d\tau = \left[-e^{-\tau}\right]_{0}^{t} = 1 - e^{-t}.$$

For $t < 0$, the upper limit is negative and $x(\tau) = 0$ over $(-\infty, t]$, so $y(t) = 0$.

Result

$$\boxed{y(t) = (1 - e^{-t})u(t).}$$

Remarks

- In DT, computing the output support first $(n_y^{\min}, n_y^{\max})$ tells you exactly which samples to compute.
- In CT, piecewise convolution is often easiest via overlap geometry and determining the correct integration limits.

3.6 Properties of Convolution

We assume throughout that all sums and integrals converge absolutely, so that reordering of sums and change of variables are justified.

Definitions

Continuous-time:

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

Discrete-time:

$$(x * h)[n] = \sum_{k=-\infty}^{\infty} x[k]\, h[n-k].$$

1. Commutativity

Theorem 3.1.

$$x * h = h * x.$$

Proof (CT)

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

Let $\lambda = t - \tau$. Then $\tau = t - \lambda$ and $d\tau = -d\lambda$. Reversing limits removes the minus sign:

$$\begin{aligned}(x * h)(t) &= \int_{-\infty}^{\infty} h(\lambda)\, x(t-\lambda)\, d\lambda \\ &= (h * x)(t).\end{aligned}$$

Proof (DT)

$$(x * h)[n] = \sum_{k=-\infty}^{\infty} x[k]\, h[n-k].$$

Let $m = n - k$. Then $k = n - m$:

$$(x * h)[n] = \sum_{m=-\infty}^{\infty} h[m]\, x[n-m] = (h * x)[n].$$

2. Associativity

Theorem 3.2.

$$(x * h) * g = x * (h * g).$$

Proof (DT)

$$\begin{aligned}((x * h) * g)[n] &= \sum_{m} (x * h)[m]\, g[n-m] \\ &= \sum_{m} \left(\sum_{k} x[k]\, h[m-k] \right) g[n-m].\end{aligned}$$

Rearranging summations (absolute convergence assumed):

$$= \sum_{k} x[k] \sum_{m} h[m-k]\, g[n-m].$$

Let $\ell = m - k$. Then $m = \ell + k$:

$$= \sum_{k} x[k] \sum_{\ell} h[\ell]\, g[n-k-\ell].$$

But the inner sum is $(h * g)[n-k]$, so:

$$= \sum_{k} x[k]\, (h * g)[n-k] = (x * (h * g))[n].$$

The CT proof follows identically by replacing sums with integrals.

3. Distributivity

Theorem 3.3.

$$x * (h + g) = x * h + x * g.$$

Proof (CT)

$$\begin{aligned}(x * (h + g))(t) &= \int x(\tau)\,(h + g)(t - \tau)\,d\tau \\ &= \int x(\tau)h(t - \tau)\,d\tau + \int x(\tau)g(t - \tau)\,d\tau \\ &= (x * h)(t) + (x * g)(t).\end{aligned}$$

The DT case is identical using sums.

4. Identity Element

Theorem 3.4.

$$x * \delta = x.$$

Proof (CT)

$$(x * \delta)(t) = \int x(\tau)\,\delta(t - \tau)\,d\tau.$$

Using the sifting property of the delta function:

$$= x(t).$$

Proof (DT)

$$(x * \delta)[n] = \sum_k x[k]\delta[n - k].$$

Only the term $k = n$ survives:

$$= x[n].$$

Thus δ is the identity element of convolution.

5. Differentiation Property (Continuous-Time)

Theorem 3.5.

$$\frac{d}{dt}(x * h) = \left(\frac{dx}{dt}\right) * h = x * \left(\frac{dh}{dt}\right).$$

Proof Start from

$$y(t) = \int x(\tau)h(t-\tau)\,d\tau.$$

Differentiate under the integral (justified by regularity assumptions):

$$\begin{aligned}\frac{d}{dt}y(t) &= \int x(\tau)\frac{d}{dt}h(t-\tau)\,d\tau \\ &= \int x(\tau)h'(t-\tau)\,d\tau \\ &= (x * h')(t).\end{aligned}$$

Using commutativity:

$$(x * h')(t) = (x' * h)(t).$$

6. Difference Property (Discrete-Time)

Define the first-difference operator:

$$\Delta x[n] = x[n] - x[n-1].$$

Theorem 3.6.

$$\Delta(x * h) = (\Delta x) * h = x * (\Delta h).$$

Proof

$$\begin{aligned}\Delta(x * h)[n] &= (x * h)[n] - (x * h)[n-1] \\ &= \sum_k x[k]h[n-k] - \sum_k x[k]h[n-1-k].\end{aligned}$$

Combine:

$$= \sum_k x[k]\big(h[n-k] - h[n-1-k]\big).$$

But

$$h[n-k] - h[n-1-k] = \Delta h[n-k].$$

Thus:

$$= \sum_k x[k]\Delta h[n-k] = (x * \Delta h)[n].$$

Using commutativity:

$$= (\Delta x) * h.$$

Summary of Properties

- Commutativity: order does not matter.
- Associativity: grouping does not matter.
- Distributivity: convolution distributes over addition.
- Identity: δ is the neutral element.
- Differentiation/difference distributes over convolution.

These algebraic properties make convolution behave like multiplication in many ways — a fact that becomes fully clear in the transform domain.

Property	**Continuous-Time (CT)**	**Discrete-Time (DT)**
Definition	$(x * h)(t) = \int_{-\infty}^{\infty} x(\tau)\, h(t-\tau)\, d\tau$	$(x * h)[n] = \sum_{k=-\infty}^{\infty} x[k]\, h[n-k]$
Commutativity	$(x * h)(t) = (h * x)(t)$	$(x * h)[n] = (h * x)[n]$
Associativity	$((x * h) * g)(t) = (x * (h * g))(t)$	$((x * h) * g)[n] = (x * (h * g))[n]$
Distributivity	$(x*(h+g))(t) = (x*h)(t)+(x*g)(t)$	$(x*(h+g))[n] = (x*h)[n]+(x*g)[n]$
Identity Element	$(x * \delta)(t) = x(t)$	$(x * \delta)[n] = x[n]$
Differentiation / Difference	$\frac{d}{dt}(x * h) = \left(\frac{dx}{dt}\right) * h = x * \left(\frac{dh}{dt}\right)$	Let $\Delta x[n] = x[n] - x[n-1]$, $\Delta(x * h) = (\Delta x) * h = x * (\Delta h)$
Support (finite signals)	If $x(t)$ supported on $[a, b]$, $h(t)$ on $[c, d]$: support of $y(t) \subseteq [a + c,\ b + d]$	If $x[n]$ nonzero on $[n_x^{\min}, n_x^{\max}]$, $h[n]$ on $[n_h^{\min}, n_h^{\max}]$: $n_y^{\min} = n_x^{\min} + n_h^{\min}$, $n_y^{\max} = n_x^{\max} + n_h^{\max}$
Length (finite DT case)	—	$L_y = L_x + L_h - 1$
Stability Condition (LTI)	$\int_{-\infty}^{\infty} \lvert h(t)\rvert\, dt < \infty$	$\sum_{n=-\infty}^{\infty} \lvert h[n]\rvert < \infty$

4 Correlation

Correlation measures the similarity between two signals as one is shifted relative to the other. It plays a central role in detection, synchronization, estimation, and matched filtering.

4.1 Cross-Correlation

Discrete-Time

$$R_{xy}[m] = \sum_{n=-\infty}^{\infty} x[n]\, y^*[n+m].$$

Continuous-Time

$$R_{xy}(\tau) = \int_{-\infty}^{\infty} x(t)\, y^*(t+\tau)\, dt.$$

Interpretation

- Slide one signal relative to the other.
- Multiply (with complex conjugation).
- Sum (DT) or integrate (CT).

A large magnitude of R_{xy} indicates strong similarity at that particular shift, τ, m (also called *lag*). For real-valued signals, $y^* = y$. Correlation is essentially an inner product between one signal and a shifted version of another.

4.2 When Does Cross-Correlation Not Exist?

Discrete-Time Case

For each shift m, the series

$$\sum_{n=-\infty}^{\infty} x[n]\, y^*[n+m]$$

must converge.
Cross-correlation may fail to exist if:

- The series diverges.
- The signals do not decay sufficiently.
- Neither signal is absolutely summable.

Sufficient Condition If both signals are absolutely summable:

$$\sum_{n=-\infty}^{\infty} |x[n]| < \infty, \qquad \sum_{n=-\infty}^{\infty} |y[n]| < \infty,$$

then $R_{xy}[m]$ exists for all m.

More generally, if one signal is absolutely summable and the other is bounded, correlation exists.

Continuous-Time Case

The integral

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

must converge for each τ.

Sufficient condition:

$$\int_{-\infty}^{\infty} |x(t)|\, dt < \infty, \qquad \int_{-\infty}^{\infty} |y(t)|\, dt < \infty.$$

Example Where Correlation Fails

Discrete-time:

$$x[n] = 1, \qquad y[n] = 1.$$

Then

$$R_{xy}[m] = \sum_{n=-\infty}^{\infty} 1,$$

which diverges.

Continuous-time:

$$x(t) = 1, \qquad y(t) = 1,$$

$$R_{xy}(\tau) = \int_{-\infty}^{\infty} 1\, dt,$$

which diverges.

Thus cross-correlation does not exist in the classical sense.

Energy vs Power Signals

Cross-correlation is always well-defined for *"finite-energy signals (energy signals)"*. For infinite-energy signals (e.g., sinusoids), classical correlation may diverge and must be defined using time averages (power signals).

4.3 When Is Cross-Correlation Nonzero?

Cross-correlation measures overlap between a signal and a shifted version of another signal:

$$R_{xy}[m] = \sum_n x[n]\, y^*[n+m].$$

It is nonzero when the two signals have nonzero overlap after shifting.

Finite-Length Signals (DT Case)

Assume:

$$x[n] \neq 0 \quad \text{for } n_x^{\min} \leq n \leq n_x^{\max}$$

$$y[n] \neq 0 \quad \text{for } n_y^{\min} \leq n \leq n_y^{\max}.$$

Then $R_{xy}[m]$ is nonzero only when the supports overlap after shifting.
The valid range of m is:

$$m = n_y^{\min} - n_x^{\max} \quad \ldots \quad n_y^{\max} - n_x^{\min}.$$

Outside this range:

$$R_{xy}[m] = 0.$$

Interpretation If the signals do not overlap after shifting, their product is zero everywhere, so the sum is zero.

Orthogonality Condition

If

$$\sum_n x[n]\, y^*[n] = 0,$$

the signals are said to be **orthogonal**.
More generally:

$$R_{xy}[m] = 0 \quad \forall m$$

means the signals are orthogonal under all shifts.

Energy Interpretation

At each shift, correlation computes an inner product:

$$R_{xy}[m] = \langle x[n], y[n+m] \rangle.$$

Thus:

- Large magnitude $\rightarrow$ strong similarity.
- Zero $\rightarrow$ orthogonality (no similarity).

Continuous-Time Analogue

For finite-support signals:

$$R_{xy}(\tau) \neq 0$$

only when the supports of $x(t)$ and $y(t+\tau)$ overlap.

Important Special Case: Autocorrelation

For any finite-energy signal:

$$R_{xx}(0) = \sum_n |x[n]|^2 > 0 \quad (\text{unless } x \equiv 0).$$

Thus:

Autocorrelation is always maximal at zero shift (for nonzero finite-energy signals).

4.4 Worked Discrete-Time Example

Let

$$x[n] = \{1, 2, 1\}, \quad n = 0, 1, 2$$
$$y[n] = \{0, 1, 2, 1\}, \quad n = 0, 1, 2, 3.$$

Step 1: Determine valid range of m

$$n_x^{\min} = 0,\ n_x^{\max} = 2, \qquad n_y^{\min} = 0,\ n_y^{\max} = 3.$$

Cross-correlation is nonzero for

$$m = n_y^{\min} - n_x^{\max} \ \ldots \ n_y^{\max} - n_x^{\min}.$$

Thus,

$$m = 0 - 2 \ldots 3 - 0 \quad \Rightarrow \quad m = -2, -1, 0, 1, 2, 3.$$

Length:

$$L_R = L_x + L_y - 1 = 3 + 4 - 1 = 6.$$

Step 2: Compute $R_{xy}[m]$ Using

$$R_{xy}[m] = \sum_n x[n]y[n+m].$$

Compute each shift:

$$R_{xy}[-2] = 1 \cdot y[-2] + 2 \cdot y[-1] + 1 \cdot y[0] = 0.$$

$$R_{xy}[-1] = 1 \cdot y[-1] + 2 \cdot y[0] + 1 \cdot y[1] = 1.$$

$$R_{xy}[0] = 1 \cdot 0 + 2 \cdot 1 + 1 \cdot 2 = 4.$$

$$R_{xy}[1] = 1 \cdot 1 + 2 \cdot 2 + 1 \cdot 1 = 6.$$

$$R_{xy}[2] = 1 \cdot 2 + 2 \cdot 1 + 1 \cdot 0 = 4.$$

$$R_{xy}[3] = 1 \cdot 1 = 1.$$

Thus:

$$R_{xy}[m] = \{0, 1, 4, 6, 4, 1\}.$$

The maximum occurs at $m = 1$, indicating strongest similarity when y is shifted left by 1.

4.5 Autocorrelation

Discrete-Time

$$R_{xx}[m] = \sum_{n=-\infty}^{\infty} x[n]\, x^*[n+m].$$

Continuous-Time

$$R_{xx}(\tau) = \int_{-\infty}^{\infty} x(t)\, x^*(t+\tau)\, dt.$$

Energy at Zero Shift

$$R_{xx}[0] = \sum_{n=-\infty}^{\infty} |x[n]|^2,$$

$$R_{xx}(0) = \int_{-\infty}^{\infty} |x(t)|^2 dt.$$

Thus, autocorrelation at zero equals signal energy.

4.6 Properties of Correlation

We assume throughout that the relevant sums/integrals converge (typically guaranteed for finite-energy signals, or under absolute summability/integrability assumptions).

Definitions

Discrete-time cross-correlation:

$$R_{xy}[m] = \sum_{n=-\infty}^{\infty} x[n]\, y^*[n+m].$$

Continuous-time cross-correlation:

$$R_{xy}(\tau) = \int_{-\infty}^{\infty} x(t)\, y^*(t+\tau)\, dt.$$

Autocorrelation is the special case $y = x$.

1. Symmetry Relations

Theorem 4.1 (Cross-correlation symmetry)**.**

$$R_{xy}[m] = R_{yx}^*[-m], \qquad R_{xy}(\tau) = R_{yx}^*(-\tau).$$

Proof (DT) Start from

$$R_{yx}[-m] = \sum_n y[n]\, x^*[n-m].$$

Take complex conjugate:

$$R_{yx}^*[-m] = \sum_n y^*[n]\, x[n-m].$$

Reindex with $k = n - m$ (i.e., $n = k + m$):

$$R_{yx}^*[-m] = \sum_k y^*[k+m]\, x[k] = \sum_k x[k]\, y^*[k+m] = R_{xy}[m].$$

Proof (CT) Similarly,

$$R_{yx}(-\tau) = \int y(t)\, x^*(t-\tau)\, dt.$$

Take conjugate:

$$R_{yx}^*(-\tau) = \int y^*(t)\, x(t-\tau)\, dt.$$

Let $\lambda = t - \tau$ ($t = \lambda + \tau$):

$$R_{yx}^*(-\tau) = \int y^*(\lambda+\tau)\, x(\lambda)\, d\lambda = \int x(\lambda)\, y^*(\lambda+\tau)\, d\lambda = R_{xy}(\tau).$$

Theorem 4.2 (Autocorrelation (Hermitian symmetry))**.**

$$R_{xx}[m] = R_{xx}^*[-m], \qquad R_{xx}(\tau) = R_{xx}^*(-\tau).$$

Remark If x is real-valued, then R_{xx} is real and even:

$$R_{xx}[m] = R_{xx}[-m], \qquad R_{xx}(\tau) = R_{xx}(-\tau).$$

2. Value at Zero Shift (Energy)

Theorem 4.3. For finite-energy signals,

$$R_{xx}[0] = \sum_{n=-\infty}^{\infty} |x[n]|^2 \geq 0, \qquad R_{xx}(0) = \int_{-\infty}^{\infty} |x(t)|^2 \, dt \geq 0.$$

Proof Set $m = 0$ (DT) or $\tau = 0$ (CT) in the definition:

$$R_{xx}[0] = \sum_n x[n]x^*[n] = \sum_n |x[n]|^2,$$

$$R_{xx}(0) = \int x(t)x^*(t) \, dt = \int |x(t)|^2 \, dt.$$

3. Linearity / Conjugate-Linearity

Theorem 4.4. Correlation is linear in its first argument and conjugate-linear in its second:

$$R_{ax_1+bx_2,\, y} = a\, R_{x_1 y} + b\, R_{x_2 y},$$
$$R_{x,\, ay_1+by_2} = a^* R_{xy_1} + b^* R_{xy_2}.$$

Proof (DT)

$$\begin{aligned} R_{ax_1+bx_2,\, y}[m] &= \sum_n (ax_1[n] + bx_2[n])\, y^*[n+m] \\ &= a \sum_n x_1[n] y^*[n+m] + b \sum_n x_2[n] y^*[n+m] \\ &= a R_{x_1 y}[m] + b R_{x_2 y}[m]. \end{aligned}$$

For the second argument:

$$\begin{aligned} R_{x,\, ay_1+by_2}[m] &= \sum_n x[n]\, (ay_1[n+m] + by_2[n+m])^* \\ &= \sum_n x[n]\, (a^* y_1^*[n+m] + b^* y_2^*[n+m]) \\ &= a^* R_{xy_1}[m] + b^* R_{xy_2}[m]. \end{aligned}$$

The CT proof is identical, replacing sums with integrals.

4. Time-Shift Properties

Theorem 4.5 (DT shift)**.** If $x_1[n] = x[n - n_0]$ and $y_1[n] = y[n - n_1]$, then

$$R_{x_1y_1}[m] = R_{xy}[m + (n_0 - n_1)].$$

Proof (DT)

$$R_{x_1y_1}[m] = \sum_n x[n - n_0]\, y^*[n + m - n_1].$$

Let $k = n - n_0$ ($n = k + n_0$):

$$R_{x_1y_1}[m] = \sum_k x[k]\, y^*[k + (m + n_0 - n_1)] = R_{xy}[m + (n_0 - n_1)].$$

Theorem 4.6 (CT shift)**.** If $x_1(t) = x(t - t_0)$ and $y_1(t) = y(t - t_1)$, then

$$R_{x_1y_1}(\tau) = R_{xy}(\tau + (t_0 - t_1)).$$

5. Connection to Convolution

Define the time-reversed conjugate:

$$\tilde{y}[n] = y^*[-n], \qquad \tilde{y}(t) = y^*(-t).$$

Theorem 4.7.

$$R_{xy}[m] = (x * \tilde{y})[m], \qquad R_{xy}(\tau) = (x * \tilde{y})(\tau).$$

Proof (DT)

$$(x * \tilde{y})[m] = \sum_n x[n]\tilde{y}[m - n] = \sum_n x[n]\, y^*[n - m] \quad (\text{since } \tilde{y}[k] = y^*[-k]).$$

Now change variable $n \mapsto n + m$:

$$(x * \tilde{y})[m] = \sum_n x[n + m]\, y^*[n].$$

Using commutativity of multiplication and renaming the dummy index gives the standard correlation form; equivalently one may start from $R_{xy}[m] = \sum_n x[n]y^*[n+m]$ and rewrite $y^*[n + m] = \tilde{y}[m - (-n)]$ to match convolution directly. Hence $R_{xy}[m] = (x * \tilde{y})[m]$.

Proof (CT)

$$(x * \tilde{y})(\tau) = \int x(t)\tilde{y}(\tau - t)\, dt = \int x(t)\, y^*(t + \tau)\, dt = R_{xy}(\tau).$$

6. Cauchy–Schwarz Bound (Useful Inequality)

Theorem 4.8. For finite-energy signals,

$$|R_{xy}[m]| \le \|x\|_2 \, \|y\|_2, \qquad |R_{xy}(\tau)| \le \|x\|_2 \, \|y\|_2,$$

where

$$\|x\|_2^2 = \sum_n |x[n]|^2 \quad \text{or} \quad \|x\|_2^2 = \int |x(t)|^2 dt.$$

Proof (DT) For fixed m, define $y_m[n] = y[n+m]$. Then

$$R_{xy}[m] = \sum_n x[n] \, y_m^*[n] = \langle x, y_m \rangle.$$

By Cauchy–Schwarz,

$$|R_{xy}[m]| \le \|x\|_2 \, \|y_m\|_2.$$

Shifting does not change the ℓ_2 norm, so $\|y_m\|_2 = \|y\|_2$.

The CT proof is identical in L^2.

Summary Table of Correlation Properties

Property	**Continuous-Time (CT)**	**Discrete-Time (DT)**
Definition	$R_{xy}(\tau) = \int_{-\infty}^{\infty} x(t) \, y^*(t+\tau) \, dt$	$R_{xy}[m] = \sum_{n=-\infty}^{\infty} x[n] \, y^*[n+m]$
Cross symmetry	$R_{xy}(\tau) = R_{yx}^*(-\tau)$	$R_{xy}[m] = R_{yx}^*[-m]$
Auto symmetry	$R_{xx}(\tau) = R_{xx}^*(-\tau)$	$R_{xx}[m] = R_{xx}^*[-m]$
Energy at zero	$R_{xx}(0) = \int \|x(t)\|^2 dt$	$R_{xx}[0] = \sum \|x[n]\|^2$
Linearity	$R_{ax_1+bx_2,\, y} = aR_{x_1y} + bR_{x_2y}$	$R_{ax_1+bx_2,\, y}[m] = aR_{x_1y}[m] + bR_{x_2y}[m]$
Conjugate-linearity	$R_{x,\, ay_1+by_2} = a^* R_{xy_1} + b^* R_{xy_2}$	$R_{x,\, ay_1+by_2}[m] = a^* R_{xy_1}[m] + b^* R_{xy_2}[m]$
Time shift	If $x(t-t_0), y(t-t_1)$: $R(\tau) = R_{xy}(\tau + t_0 - t_1)$	If $x[n-n_0], y[n-n_1]$: $R[m] = R_{xy}[m + n_0 - n_1]$
Connection to convolution	$R_{xy}(\tau) = (x * \tilde{y})(\tau)$, $\tilde{y}(t) = y^*(-t)$	$R_{xy}[m] = (x * \tilde{y})[m]$, $\tilde{y}[n] = y^*[-n]$
Cauchy–Schwarz bound	$\|R_{xy}(\tau)\| \le \|x\|_2 \|y\|_2$	$\|R_{xy}[m]\| \le \|x\|_2 \|y\|_2$

4.7 Correlation as an Inner Product and Projection

Correlation can be interpreted geometrically as an inner product between signals in a vector space.

Discrete-Time View

Consider finite-energy sequences. Define the inner product:

$$\langle x, y \rangle = \sum_{n=-\infty}^{\infty} x[n]\, y^*[n].$$

Then cross-correlation can be written as:

$$R_{xy}[m] = \sum_n x[n]\, y^*[n+m] = \langle x, y_m \rangle,$$

where

$$y_m[n] = y[n+m]$$

is a shifted version of y.
Thus:

$$\boxed{R_{xy}[m] = \langle x, y_m \rangle}$$

Correlation is simply the inner product between x and a shifted version of y.

Continuous-Time View

Define the L^2 inner product:

$$\langle x, y \rangle = \int_{-\infty}^{\infty} x(t)\, y^*(t)\, dt.$$

Then:

$$R_{xy}(\tau) = \int x(t)\, y^*(t+\tau)\, dt = \langle x, y_\tau \rangle,$$

with

$$y_\tau(t) = y(t+\tau).$$

Again, correlation is an inner product with a shifted signal.

Projection Interpretation

In a Hilbert space (finite-energy signals), the projection of x onto y_m is:

$$\text{proj}_{y_m}(x) = \frac{\langle x, y_m \rangle}{\|y_m\|^2}\, y_m.$$

The scalar coefficient of the projection is:

$$\frac{\langle x, y_m \rangle}{\|y_m\|}.$$

Thus:

$$R_{xy}[m] = \langle x, y_m \rangle$$

measures how much of x lies in the direction of y_m.

Geometric Picture

Think of signals as vectors in a high-dimensional space:

- If x and y_m are aligned, correlation is large.
- If they are orthogonal, correlation is zero.
- If they are opposite, correlation is negative (real case).

Thus:

$$R_{xy}[m] = \|x\| \, \|y\| \cos(\theta_m),$$

where θ_m is the angle between x and y_m.

4.8 Normalized Cross-Correlation

The magnitude of the cross-correlation

$$R_{xy}[m] = \sum_{n=-\infty}^{\infty} x[n] \, y^*[n+m]$$

depends on the energy of the signals. If one signal is scaled by a constant, the correlation scales as well.

To remove this amplitude dependence, we define the **normalized cross-correlation**:

$$\boxed{\rho_{xy}[m] = \frac{R_{xy}[m]}{\sqrt{R_{xx}[0] \, R_{yy}[0]}}}$$

where

$$R_{xx}[0] = \sum_n |x[n]|^2, \qquad R_{yy}[0] = \sum_n |y[n]|^2.$$

Continuous-time version:

$$\rho_{xy}(\tau) = \frac{R_{xy}(\tau)}{\sqrt{R_{xx}(0) \, R_{yy}(0)}}$$

with

$$R_{xx}(0) = \int |x(t)|^2 dt, \qquad R_{yy}(0) = \int |y(t)|^2 dt.$$

Interpretation

Recall that correlation is an inner product:

$$R_{xy}[m] = \langle x, y_m \rangle,$$

where $y_m[n] = y[n+m]$.
Define the ℓ_2 norms:

$$\|x\|_2 = \sqrt{R_{xx}[0]}, \qquad \|y\|_2 = \sqrt{R_{yy}[0]}.$$

Then normalized correlation becomes:

$$\rho_{xy}[m] = \frac{\langle x, y_m \rangle}{\|x\|_2 \|y\|_2}.$$

This is exactly the cosine of the angle between the two vectors:

$$\rho_{xy}[m] = \cos(\theta_m).$$

Key Property (Cauchy–Schwarz)

From the Cauchy–Schwarz inequality:

$$|\langle x, y_m \rangle| \leq \|x\|_2 \|y_m\|_2.$$

Since shifting does not change energy,

$$\|y_m\|_2 = \|y\|_2,$$

we obtain:

$$\boxed{|\rho_{xy}[m]| \leq 1.}$$

Equality holds if and only if

$$y[n+m] = Cx[n]$$

for some constant C.

Special Case: Autocorrelation

For $x = y$:

$$\rho_{xx}[m] = \frac{R_{xx}[m]}{R_{xx}[0]}.$$

Thus:

$$\rho_{xx}[0] = 1.$$

Normalized autocorrelation measures similarity relative to total energy.

Why Is Normalization Important?

Without normalization:

- Larger-amplitude signals produce larger correlation values.
- Comparison across signals with different energy is misleading.

With normalization:

- Values lie in $[-1, 1]$ (real case).
- 1 indicates perfect alignment.
- 0 indicates orthogonality.
- -1 indicates perfect inverse alignment.

5 Matched Filter

The matched filter is one of the most important applications of convolution and correlation in engineering. It provides an optimal solution to a fundamental problem: *detecting a known signal embedded in noise*. Rather than treating convolution and correlation as purely mathematical operations, the matched filter shows how they arise naturally when we ask:

"How can we reliably detect a known waveform inside a noisy measurement?"

Surprisingly, the answer is deeply geometric: detection reduces to measuring similarity, which in turn reduces to computing an inner product. The matched filter implements this idea using an LTI system whose impulse response is tailored ("matched") to the signal we want to detect.

In this section, we will:

- Formulate the detection problem,
- Interpret correlation geometrically,
- Derive the matched filter mathematically,
- Show why it maximizes signal-to-noise ratio,
- Connect theory to practical implementation.

5.1 Detection Problem

Suppose we observe a noisy signal:

$$r(t) = s(t - \tau_0) + w(t),$$

where:

- $s(t)$: known template signal,
- τ_0: unknown delay,
- $w(t)$: additive noise.

Goal: Estimate the delay τ_0.

This is a fundamental problem in:

- Radar
- Sonar
- Wireless communications
- Synchronization
- Pattern recognition

5.2 Key Idea: Detection as Similarity Search

If we slide the template across the received signal and measure similarity at each shift:

$$\text{Maximum similarity} \quad \Rightarrow \quad \text{correct delay.}$$

Thus detection becomes a similarity maximization problem.

5.3 Similarity via Correlation

We measure similarity using cross-correlation:

$$R_{rs}(\tau) = \int_{-\infty}^{\infty} r(t)\, s^*(t - \tau)\, dt.$$

This computes the inner product between:

$$r(t) \quad \text{and} \quad s(t - \tau).$$

We estimate:

$$\hat{\tau} = \arg\max_{\tau} |R_{rs}(\tau)|.$$

5.4 Matched Filter as an LTI System

From the convolution–correlation relation:

$$R_{rs}(\tau) = (r * \tilde{s})(\tau), \qquad \tilde{s}(t) = s^*(-t).$$

Thus correlation can be implemented as an LTI system with impulse response:

$$\boxed{h(t) = s^*(-t)}$$

and output:

$$y(t) = (r * h)(t).$$

This system is called the **matched filter**.
It is "matched" because the impulse response is matched to the signal shape.

5.5 Geometric Interpretation (Discrete-Time)

In discrete time:

$$y[n] = \sum_k r[k]\, s^*[k-n].$$

Define the shifted template:

$$s_n[k] = s[k-n].$$

Then:

$$y[n] = \langle r, s_n \rangle.$$

Thus:

Matched filtering = projection onto shifted template.

Maximum output occurs when the signals are maximally aligned.

5.6 Signal vs Noise Behavior

At the correct delay τ_0:

$$r(t) = s(t-\tau_0) + w(t).$$

The matched filter output becomes:

$$y(\tau_0) = \int s(t-\tau_0)s^*(t-\tau_0)\,dt + \int w(t)s^*(t-\tau_0)\,dt.$$

Thus:

$$y(\tau_0) = \|s\|^2 + \text{noise term}.$$

- The signal adds **coherently** (constructively).
- Noise adds **incoherently**.

This produces a peak at the correct delay.

5.7 Why Is It Optimal? (SNR Argument)

Assume:

- $w(t)$ is white noise,
- We use a linear filter $h(t)$,
- We sample the output at time τ_0.

The output is:

$$y(\tau_0) = \int r(t)h(\tau_0 - t)\,dt.$$

Signal component:

$$\text{Signal} = \langle s, h_{\tau_0} \rangle.$$

Noise variance at output:

$$\sigma_w^2 \int |h(t)|^2 dt.$$

The output SNR is:

$$\text{SNR} = \frac{|\langle s, h \rangle|^2}{\sigma_w^2 \|h\|^2}.$$

By the Cauchy–Schwarz inequality:

$$|\langle s, h \rangle|^2 \leq \|s\|^2 \|h\|^2.$$

Equality holds if and only if:

$$h(t) = Cs^*(-t).$$

Thus:

The matched filter maximizes output SNR in white noise.

5.8 Discrete-Time Version

Observation:

$$r[n] = s[n - n_0] + w[n].$$

Matched filter:

$$h[n] = s^*[-n].$$

Output:

$$y[n] = (r * h)[n].$$

Estimate:

$$\hat{n}_0 = \arg\max_n |y[n]|.$$

5.9 Practical Implementation

Using NumPy (Discrete-Time)

```
import numpy as np

# template signal
s = np.array([...])

# received signal
r = np.array([...])

# matched filter (time-reversed conjugate)
h = np.conj(s[::-1])

# convolution
y = np.convolve(r, h, mode='full')

# estimate delay
n_hat = np.argmax(np.abs(y))
```

Using SciPy

```
from scipy.signal import correlate

y = correlate(r, s, mode='full')
n_hat = np.argmax(np.abs(y))
```

Note: `correlate()` directly computes correlation, so manual reversal is not needed.

5.10 Matched Filter Example: Detecting a Delayed Cosine Burst

We now implement a matched filter in discrete time to detect a known finite-duration template embedded in white noise.

Problem Setup

We observe a noisy signal

$$r[n] = s[n - n_0] + w[n],$$

where:

- $s[n]$ is a known finite-duration cosine burst (the template),
- n_0 is an unknown delay,
- $w[n]$ is additive white noise.

The matched filter is defined by

$$h[n] = s^*[-n].$$

For real-valued $s[n]$, this becomes a simple time-reversal:

$$h[n] = s[-n].$$

The matched-filter output is

$$y[n] = (r * h)[n],$$

and the delay estimate is obtained by the location of the peak:

$$\hat{n}_0 \approx \arg\max_n y[n] - (L_s - 1),$$

where L_s is the template length (due to the `full` convolution index offset).

Python Code (Signal Generation + Matched Filter)

```
import numpy as np
import matplotlib.pyplot as plt

rng = np.random.default_rng(0)

# Parameters
N = 400
n = np.arange(N)

# Finite-duration cosine burst (template)
Ls = 120
n_s = np.arange(Ls)
f0 = 0.08  # normalized frequency
s = np.cos(2*np.pi*f0*n_s)

# Insert delayed burst into noise
delay = 180
r = np.zeros(N)
r[delay:delay+Ls] += s

sigma = 1.
r_noisy = r + sigma * rng.standard_normal(N)
```

```python

# Matched filter (time-reversed template)
h = s[::-1]

# Convolution (matched filter output)
y = np.convolve(r_noisy, h, mode='full')
n_y = np.arange(len(y))

# Estimate delay
peak_index = np.argmax(y)
delay_hat = peak_index - (Ls - 1)

# --- Plot 1: noisy signal ---
plt.figure()
plt.plot(n, r_noisy, label="Noisy signal")
plt.plot(n, r - 4., label="Clean signal (shifted down for visibility)")
plt.title("Noisy signal with delayed cosine burst")
plt.xlabel("n")
plt.ylabel("Amplitude")
plt.legend()
plt.grid(True)
plt.savefig("figs/matched_filter_signal.pdf")
plt.show()

# --- Plot 2: matched filter output ---
plt.figure()
plt.plot(n_y, y)
plt.title(f"Matched Filter Output (peak at {peak_index}, delay \u2248 {
    delay_hat})")
plt.xlabel("n")
plt.ylabel("Output")
plt.grid(True)
plt.savefig("figs/matched_filter.pdf")
plt.show()
```

Interpretation

Figure 1 shows the received noisy signal containing a delayed cosine burst. Figure 2 shows the matched filter output, which exhibits a clear peak at the delay location.

In this run, the matched filter peak occurs at

$$n_{\text{peak}} = 299, \qquad \hat{n}_0 = n_{\text{peak}} - (L_s - 1) = 299 - 119 = 180,$$

which exactly matches the true delay.

Implementation Notes

- Using `mode='full'` yields an output of length $N + L_s - 1$. The delay estimate must account for the convolution offset $(L_s - 1)$.
- If the template is complex, use $h =$ `np.conj(s[::-1])`.
- For large signals, correlation/convolution is often computed efficiently using FFTs.

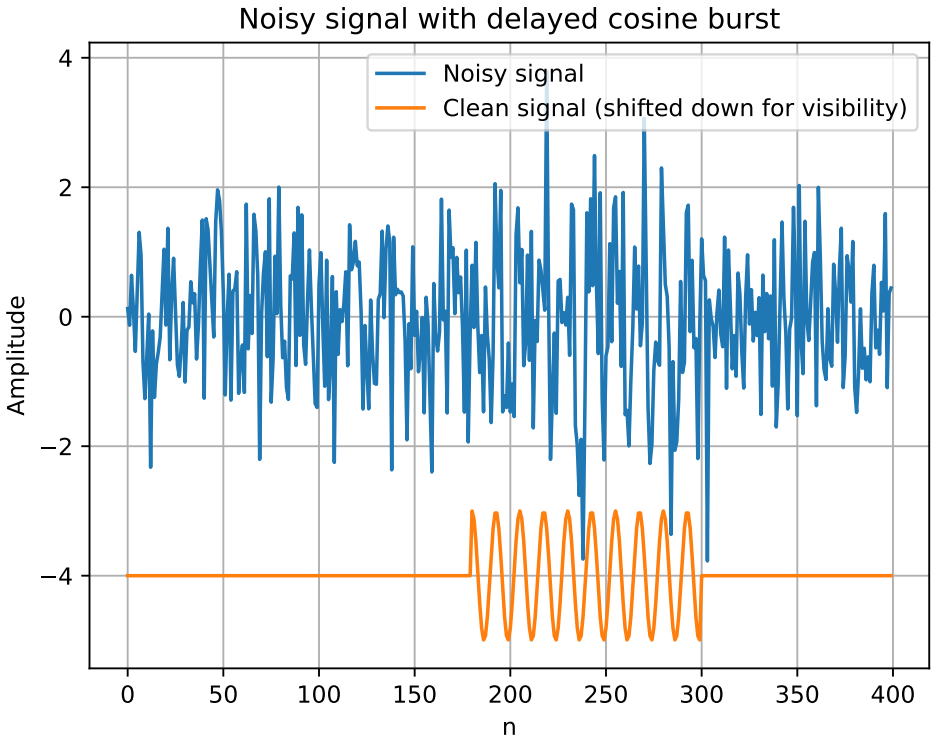


Figure 1: Noisy observation containing a delayed cosine burst (clean signal shown shifted down for visibility).

How to Implement the Matched Filter: Time Reversal vs Indexing

1. Ideal matched filter definition

In general, the matched filter is defined as

$$h[n] = s^*[-n]$$

(ignoring complex conjugation for simplicity).
Thus, conceptually, the matched filter is simply a time-reversed version of the signal.

2. The issue with finite-length discrete signals

In practice, the template $s[n]$ is only defined for a finite set of indices:

$$s[n], \quad n = 0, 1, 2, \ldots, L_s - 1.$$

If we directly form $s[-n]$, then:

$$\begin{aligned} n = 0 &\Rightarrow s[0] \quad \text{(valid)} \\ n = 1 &\Rightarrow s[-1] \quad \text{(not defined)} \\ n = 2 &\Rightarrow s[-2] \quad \text{(not defined)} \end{aligned}$$

Thus, $s[-n]$ shifts the signal into negative indices, which are typically not used in implementations.

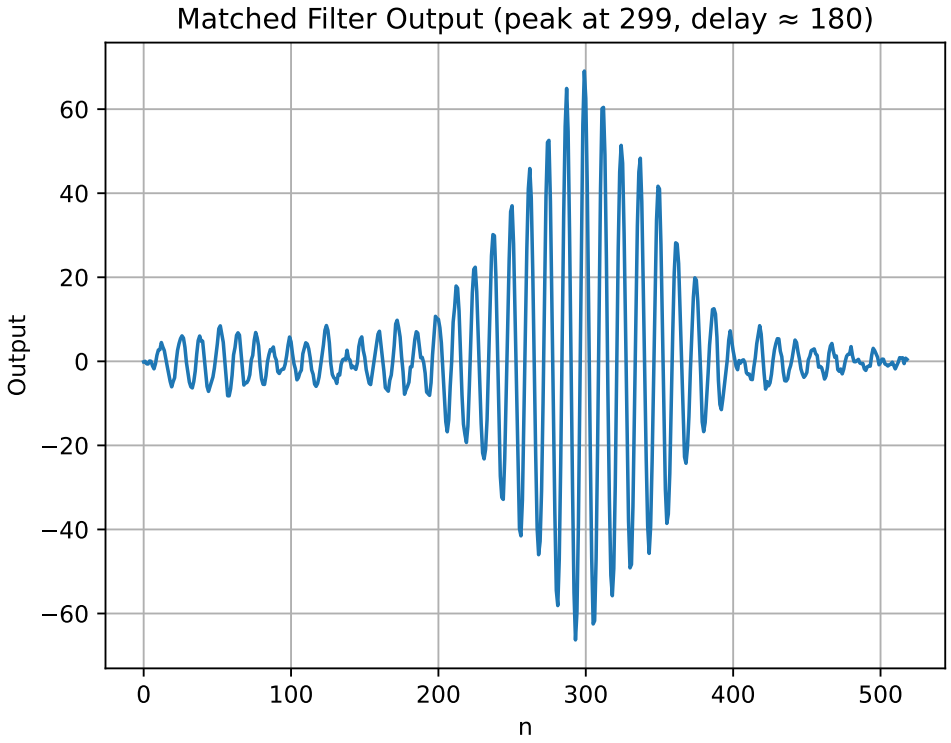


Figure 2: Matched filter output $y[n] = (r * h)[n]$. The peak location provides the delay estimate.

3. Practical discrete-time reversal

To reverse the sequence while keeping indices nonnegative, we define:

$$h[n] = s[L_s - 1 - n], \quad n = 0, \ldots, L_s - 1.$$

Now:

$$\begin{aligned} h[0] &= s[L_s - 1], \\ h[1] &= s[L_s - 2], \\ &\vdots \\ h[L_s - 1] &= s[0]. \end{aligned}$$

This is exactly the time-reversed sequence, but reindexed so that it remains supported on

$$0 \le n \le L_s - 1.$$

4. Relation to the ideal definition

Define the ideal reversed sequence

$$\tilde{h}[n] = s[-n].$$

Then the implementable version satisfies

$$h[n] = \tilde{h}[n - (L_s - 1)].$$

Thus,

$$s[L_s - 1 - n]$$

is simply a shifted version of

$$s[-n].$$

The shift by $(L_s - 1)$ ensures the reversed sequence starts at index 0 instead of $-(L_s - 1)$.

5. Consequence for delay estimation

Because the reversed signal is shifted by $(L_s - 1)$ samples, the convolution output peak appears at

$$n_{\text{peak}} = n_0 + (L_s - 1),$$

rather than at n_0.
Therefore, the delay estimate is

$$\hat{n}_0 = \arg\max_n y[n] - (L_s - 1).$$

Finally, the key takeaway:

- The ideal matched filter is $h[n] = s[-n]$.
- For finite-length discrete sequences, we use

  $$h[n] = s[L_s - 1 - n]$$

  to keep indices nonnegative.
- This introduces a shift of $(L_s - 1)$ samples.
- The delay estimate compensates for this shift.

5.11 Summary

- Detection reduces to correlation.
- Correlation can be implemented as convolution.
- The matched filter is:

  $$h(t) = s^*(-t).$$

- It maximizes output SNR in white noise.
- It is geometrically a projection.

6 Practical Hints: Convolution and Correlation in NumPy/SciPy

This section shows how to implement convolution/correlation in practice using `numpy` and `scipy.signal`. The key ideas from the math remain the same:

- Convolution: $y = x * h$ (LTI filtering).
- Correlation: R_{xy} measures similarity as a function of shift.
- Pay attention to *indexing*, *output length*, and *mode*.

6.1 Convolution in NumPy: `np.convolve`

Definition and Output Length (DT)

For finite-length sequences x (length N) and h (length L):

$$y[n] = (x * h)[n] = \sum_{k=-\infty}^{\infty} x[k]\, h[n-k], \qquad L_y = N + L - 1.$$

`np.convolve(x, h, mode='full')` returns exactly this full-length output.

Modes: `full`, `same`, `valid`

Let $N =$ `len(x)`, $L =$ `len(h)`:

- `mode='full'`: length $N + L - 1$ (all overlaps).
- `mode='same'`: length N (centered crop of full).
- `mode='valid'`: length $N - L + 1$ if $N \geq L$ (only full overlap).

```python
import numpy as np

x = np.array([1, 2, 1])      # length N=3
h = np.array([1, -1])        # length L=2

y_full  = np.convolve(x, h, mode="full")   # length 4
y_same  = np.convolve(x, h, mode="same")   # length 3
y_valid = np.convolve(x, h, mode="valid")  # length 2

print("full :", y_full)
print("same :", y_same)
print("valid:", y_valid)
```

Rule of thumb Use `full` when you care about correct indexing and shifts (e.g., delay estimation). Use `same` when you want an output aligned to the input length (common in filtering pipelines). Use `valid` when you only want samples unaffected by boundary effects.

6.2 Example 1: Simple Low-Pass Filter via Moving Average

A moving-average filter of length L is[1]:

$$h[n] = \frac{1}{L}, \quad n = 0, 1, \ldots, L-1.$$

Filtering is convolution:

$$y[n] = (x * h)[n].$$

```python
import numpy as np
import matplotlib.pyplot as plt

# Signal: noisy sinusoid
N = 400
n = np.arange(N)
x = np.sin(2*np.pi*0.07*n) + 0.6*np.random.randn(N)

# Moving-average low-pass FIR
L = 21
h = np.ones(L) / L

# Filter output (same-length)
y = np.convolve(x, h, mode="same")

plt.figure()
plt.plot(n, x, label="x[n] (noisy)")
plt.plot(n, y, label="y[n] (moving-average)")
plt.title("FIR low-pass via np.convolve")
plt.xlabel("n"); plt.ylabel("Amplitude")
plt.grid(True); plt.legend()
plt.show()
```

Interpretation This is exactly the "sliding weighted sum" view of convolution:

$$y[n] = \sum_{k=0}^{L-1} \frac{1}{L} x[n-k].$$

(With boundary behavior controlled by `mode`.)

6.3 Correlation in SciPy: `scipy.signal.correlate`

SciPy provides correlation directly:

$$R_{xy}[m] = \sum_{n} x[n]\, y^*[n+m].$$

Modes and Output Length

For finite-length signals:

- `mode='full'`: length $N + L - 1$ (all shifts).

[1] This is a simple FIR filter with uniform coefficients, which we will cover later in the course.

- `mode='same'`: length N.
- `mode='valid'`: length $N - L + 1$ (if $N \geq L$).

```python
import numpy as np
from scipy.signal import correlate, correlation_lags

x = np.array([1, 2, 1])
y = np.array([0, 1, 2, 1])

R = correlate(x, y, mode="full")  # R_xy
lags = correlation_lags(len(x), len(y), mode="full")

print("lags:", lags)
print("R   :", R)
```

Important SciPy returns the correlation values and also provides the correct *lag axis* via `correlation_lags`. Always plot/interpret correlation as a function of lag.

6.4 Example 2: Autocorrelation and Energy

From the theory:

$$R_{xx}[0] = \sum_n |x[n]|^2 = \|x\|_2^2$$

(i.e., autocorrelation at zero lag equals energy for finite-energy signals).

```python
import numpy as np
from scipy.signal import correlate, correlation_lags

rng = np.random.default_rng(0)
x = rng.standard_normal(200)

Rxx = correlate(x, x, mode="full")
lags = correlation_lags(len(x), len(x), mode="full")

# pick the value at zero lag
Rxx0 = Rxx[lags == 0][0]
energy = np.sum(np.abs(x)**2)

print("Rxx[0] =", Rxx0)
print("Energy =", energy)
```

What you should observe

$$R_{xx}[0] \approx \sum |x[n]|^2$$

(up to floating-point rounding).

6.5 Example 3: Delay Estimation via Cross-Correlation

If

$$r[n] \approx s[n - n_0] + w[n],$$

then $R_{rs}[m]$ peaks near the delay n_0 (depending on the precise definition/sign convention). Using SciPy, the cleanest approach is:

```python
import numpy as np
from scipy.signal import correlate, correlation_lags

rng = np.random.default_rng(0)

# template
Ls = 80
n_s = np.arange(Ls)
s = np.cos(2*np.pi*0.08*n_s)

# received = delayed template + noise
N = 300
delay = 140
r = np.zeros(N)
r[delay:delay+Ls] += s
r_noisy = r + 0.8*rng.standard_normal(N)

# cross-correlation
R = correlate(r_noisy, s, mode="full")
lags = correlation_lags(len(r_noisy), len(s), mode="full")

# delay estimate = lag at maximum correlation magnitude
lag_hat = lags[np.argmax(np.abs(R))]
print("true delay:", delay)
print("estimated delay:", lag_hat)
```

Why this is robust Using `correlation_lags` avoids manual "$L-1$" index corrections and keeps the math-to-code mapping explicit: you estimate the delay directly in *lag units*.

6.6 Matched Filter Implementation Choices

You can implement matched filtering either by:

- Convolution with $h[n] = s^*[-n]$ using `np.convolve`, or
- Direct correlation using `scipy.signal.correlate`.

Both are mathematically equivalent because:

$$R_{rs}[m] = (r * s^*[-\cdot])[m].$$

Complex-valued templates If s may be complex, use conjugation:

$$h[n] = s^*[-n] \quad \Rightarrow \quad \texttt{h = np.conj(s[::-1])}.$$

6.7 Practical Notes and Common Pitfalls

- **Indexing/lag conventions:** always compute or track the lag axis explicitly (e.g., `correlation_lags`).
- **Boundary effects:** `same` crops; `valid` removes partial overlaps.
- **Normalization:** sometimes you want normalized correlation

$$\rho_{xy}[m] = \frac{R_{xy}[m]}{\sqrt{R_{xx}[0]\, R_{yy}[0]}}$$

  to compare across signals with different energy.
- **Speed:** for long signals, FFT-based methods are faster (e.g., `scipy.signal.fftconvolve` for convolution).

Takeaway

- Use `np.convolve` for LTI filtering (convolution).
- Use `scipy.signal.correlate` (+ `correlation_lags`) for similarity search (correlation).
- Always be explicit about `mode` and indexing/lag interpretation.