Linear Time-Invariant (LTI) Systems

Digital Signal Processing

January 31, 2023

Linear Systems

Definition

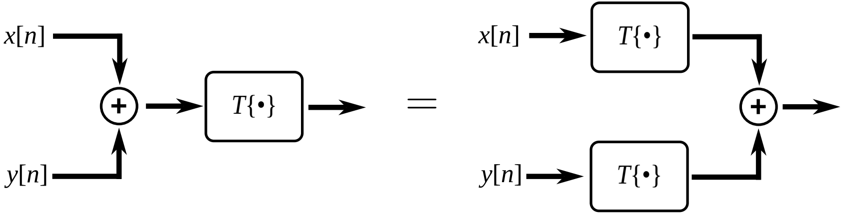
A **linear system** is a system T that satisfies:

1. Additivity: $T\{x[n] + y[n]\} = T\{x[n]\} + T\{y[n]\}$,
2. Scaling: $T\{ax[n]\} = aT\{x[n]\}$,

for all signals $x[n], y[n]$, and all scalar constants, a.

Linearity Property in Diagrams

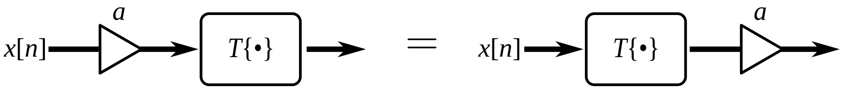
Additivity:



$$T\{x[n] + y[n]\} = T\{x[n]\} + T\{y[n]\}$$

Linearity Property in Diagrams

Scaling:



$$T\{ax[n]\} = aT\{x[n]\}$$

Linear Systems (again)

An *equivalent* definition of linearity combines additivity and scaling into one rule:

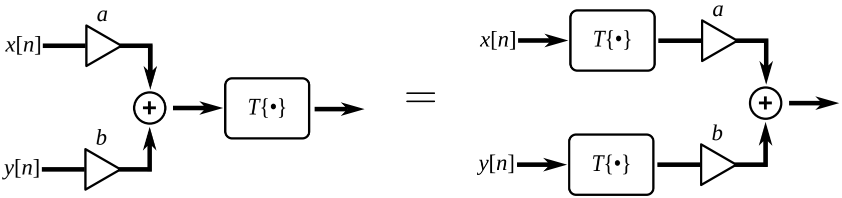
Definition

A **linear system** is a system T that satisfies:

$$T\{ax[n] + by[n]\} = aT\{x[n]\} + bT\{y[n]\},$$

for all signals $x[n], y[n]$, and all scalar constants, a, b.

Linearity Property in Diagrams (again)



$$T\{ax[n] + by[n]\} = aT\{x[n]\} + bT\{y[n]\}$$

Examples

Are the following linear systems or non-linear systems?

- $T\{x[n]\} = 2x[n]$ Linear
- $T\{x[n]\} = x[n-1]$ Linear
- $T\{x[n]\} = x[n]^2$ Non-linear
- $T\{x[n]\} = nx[n]$ Linear
- $T\{x[n]\} = x[2n]$ Linear
- $T\{x[n]\} = x[n] + 1$ Non-linear

Time-Invariant Systems

Definition

A system, T, is called **time-invariant**, or **shift-invariant**, if it satisfies

$$y[n] = T\{x[n]\} \Rightarrow y[n-N] = T\{x[n-N]\},$$

for all signals $x[n]$ and all shifts $N \in \mathbb{Z}$.

Examples

Are the following time-invariant or time-variant systems?

- $T\{x[n]\} = 2x[n]$ Time-invariant
- $T\{x[n]\} = x[n-1]$ Time-invariant
- $T\{x[n]\} = x[n]^2$ Time-invariant
- $T\{x[n]\} = nx[n]$ Time-variant
- $T\{x[n]\} = x[2n]$ Time-variant
- $T\{x[n]\} = x[n] + 1$ Time-invariant

Linear Time-Invariant (LTI) Systems

Definition

A **linear time-invariant (LTI) system** is one that is both linear and time-invariant.

Examples

Are the following LTI or not LTI systems?

- $T\{x[n]\} = 2x[n]$ LTI
- $T\{x[n]\} = x[n-1]$ LTI
- $T\{x[n]\} = x[n]^2$ not LTI
- $T\{x[n]\} = nx[n]$ not LTI
- $T\{x[n]\} = x[2n]$ not LTI
- $T\{x[n]\} = x[n] + 1$ not LTI

LTI Fun Fact

The **only** way to get an LTI system is by composing time shifts and scalings by constants.

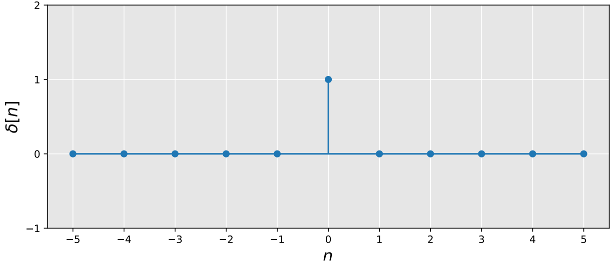
In other words, any LTI system, T, can be written as

$$T\{x[n]\} = \sum_{m=-\infty}^{\infty} a_m x[n-m],$$

for some scalar constants, a_m.

Impulse Response

Recall our **unit sample function** or **impulse**:



$$\delta[n] = \begin{cases} 1, & n = 0, \\ 0, & n \neq 0. \end{cases}$$

Impulse Response

Definition

The **impulse response** of a system, T, is the output it produces when given the unit impulse function as input. This is denoted:

$$h[n] = T\{\delta[n]\}.$$

Impulse Response

Recall any sequence, $x[n]$, can be written as a sum of scaled, shifted impulses:

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

This is the principle of **superposition**.

Impulse Response for an LTI System

Given an LTI, T:

$$T\{x[n]\} = T\left\{\sum_{k=-\infty}^{\infty} x[k]\delta[n-k]\right\} \qquad \text{superposition for } x[n]$$

$$= \sum_{k=-\infty}^{\infty} T\left\{x[k]\delta[n-k]\right\} \qquad \text{additivity property}$$

$$= \sum_{k=-\infty}^{\infty} x[k]T\left\{\delta[n-k]\right\} \qquad \text{scaling property}$$

$$= \sum_{k=-\infty}^{\infty} x[k]h[n-k] \qquad \text{definition of impulse response}$$

Convolution

Definition

The convolution of two sequence, $x[n]$, $h[n]$, is given by

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

With this notation, any LTI system, T, with impulse response, h, can be computed as

$$T\{x[n]\} = x[n] * h[n].$$

Properties of Convolution

Commutativity:

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

Properties of Convolution

Associativity:

$$(x[n] * h_1[n]) * h_2[n] = x[n] * (h_1[n] * h_2[n])$$

This means that we can apply $h_1[n]$ to $x[n]$ followed by $h_2[n]$, or we can convolve the impulse responses $h_2[n] * h_1[n]$ and then apply the resulting system to $x[n]$.

Properties of Convolution

Linearity:

$$(ax[n]) * h[n] = a(x[n] * h[n])$$

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

Properties of Convolution

Time-Invariance / Shift-Invariance:

Let $D\{x[n]\} = x[n-N]$ be an ideal delay by N. Then

$$D\{x[n] * h[n]\} = D\{x[n]\} * h[n]$$

This means that we can convolve $x[n]$ and $h[n]$ and then shift the result, or we can shift $x[n]$ and then convolve it with $h[n]$.

Equivalence of LTI Systems and Convolutions

Theorem

A system, $T\{\}$, is linear and time-invariant if and only if it can be written as a convolution,

$$T\{x[n]\} = (x * h)[n],$$

for some signal, h.

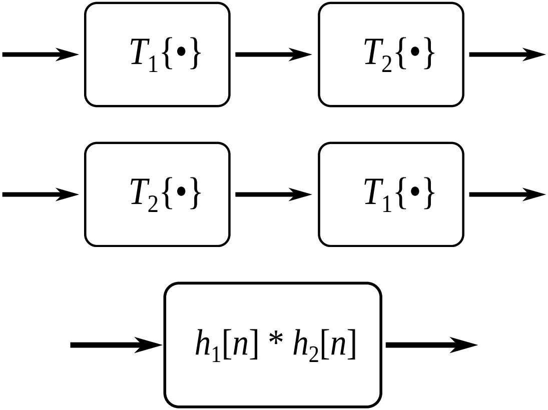
Commutativity of LTI Systems

Let T_1 and T_2 be LTI systems, with impulse responses h_1, h_2, respectively.

$$\begin{aligned}
T_2\{T_1\{x[n]\}\} &= (x[n] * h_1[n]) * h_2[n] \\
&= x[n] * (h_1[n] * h_2[n]) && \text{associativity of } * \\
&= x[n] * (h_2[n] * h_1[n]) && \text{commutativity of } * \\
&= (x[n] * h_2[n]) * h_1[n] && \text{associativity again} \\
&= T_1\{T_2\{x[n]\}\}
\end{aligned}$$

Commutativity of LTI Systems

If T_1 and T_2 are LTI systems, the following are equivalent:



Stability

Definition

A signal, $x[n]$, is **bounded** if $|x[n]| \leq B$ for some $B < \infty$ and for all $n \in \mathbb{Z}$

Definition

A system, $T\{\cdot\}$, is said to be **bounded-input, bounded-output (BIBO) stable** if for every bounded input $x[n]$, the resulting output $T\{x[n]\}$ is also bounded.

BIBO Stability of LTI Systems

Theorem

An LTI system is BIBO stable if and only if its impulse response, $h[n]$*, is absolutely summable:*

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

Causality

Definition

A system is said to be **causal** if, for any $n_0 \in \mathbb{Z}$, $T\{x[n_0]\}$ depends only on previous values of $x[n]$, for $n \leq n_0$

A causal system cannot "look into the future."

If $x[n] = y[n]$ for all $n < n_0$, then $T\{x[n]\} = T\{y[n]\}$ for all $n < n_0$.

Causality of LTI Systems

Theorem

An LTI system is causal if and only if its impulse response function, $h[n]$, satisfies $h[n] = 0$ for all $n < 0$.

Sketchy proof.
Our LTI system output evaluated for some n_0 is:

$$(h * x)[n_0] = \sum_{k=-\infty}^{\infty} h[k]x[n_0 - k]$$

This will avoid using $x[n]$ for $n > n_0$ if and only if $h[k] = 0$ when $n_0 - k > n_0$. That is, when $k < 0$.

Working with Finite-Length Signals

The convolution equation deals with signals, $x[n], h[n]$, that are defined for **infinite time:** $-\infty < n < \infty$:

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k], \qquad \text{for } n \in \mathbb{Z}.$$

Of course, on a computer we can only store signals that are **finite sequences**, that is, arrays with index $n \in [0, L-1]$.

Padding

For a finite-length signal, $x[n]$, defined for $n \in [0, L-1]$, we can extend it to all $n \in \mathbb{Z}$ by **padding**.

Multiple ways to pad:

- Pad with zeros: $x[n] = 0$ for $n < 0$ and $n \geq L$
- Periodic padding: $x[n] = x[n \bmod L]$ for $n \in \mathbb{Z}$
- many more ...

Convolution with Zero Padding

Zero padding means we can truncate the k and n indices in our convolution equation to be between $[0, L-1]$:

$$x[n] * h[n] = \sum_{k=0}^{L-1} x[k]h[n-k], \qquad \text{for } n \in [0, L-1].$$

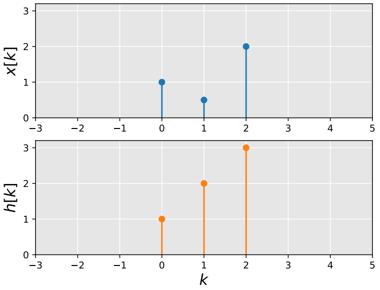
Does this work? No! $n - k$ can be negative.

Instead, truncate k at n:

$$x[n] * h[n] = \sum_{k=0}^{n} x[k]h[n-k], \qquad \text{for } n \in [0, L-1].$$

Convolution Example

Computing $y[n] = x[n] * h[n] = \sum_{k=0}^{n} x[k]h[n-k]$



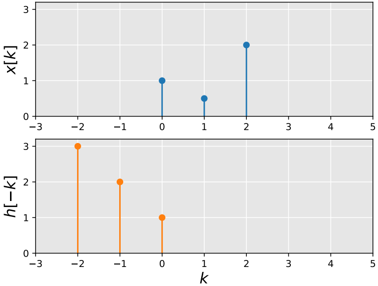
$$x[n] = (0.5, 1.5, 2.5)$$

$$h[n] = (1.0, 2.0, 3.0)$$

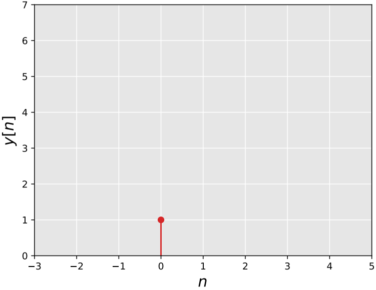
Convolution Example

Computing $y[n] = x[n] * h[n] = \sum_{k=0}^{n} x[k]h[n-k]$

For $n = 0$, flip h about 0 to get $h[-k]$.



$\Rightarrow$

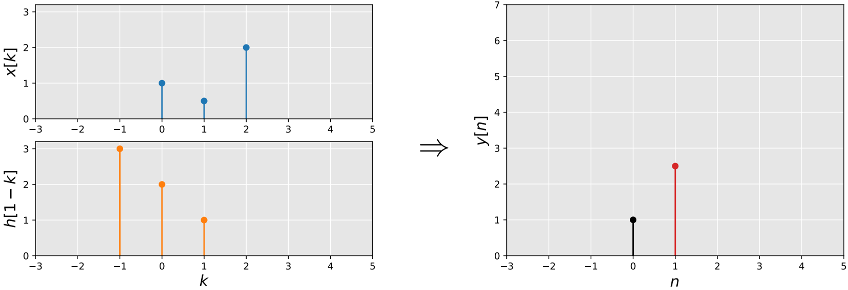


$$y[0] = x[0] \times h[0]$$
$$= 1.0 \times 1.0 = 1.0$$

Convolution Example

Computing $y[n] = x[n] * h[n] = \sum_{k=0}^{n} x[k]h[n-k]$

For $n = 1$, shift h right by one to get $h[1-k]$.



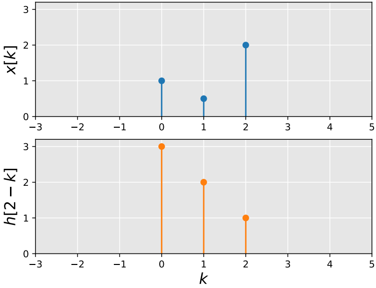
$\Rightarrow$

$$y[1] = x[0]h[1] + x[1]h[0]$$
$$= 1.0 \times 2.0 + 0.5 \times 1.0 = 2.5$$

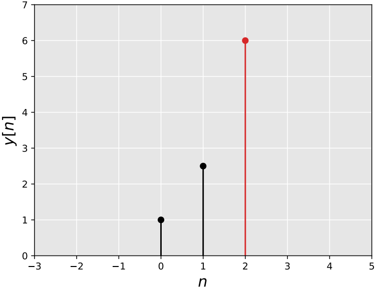
Convolution Example

Computing $y[n] = x[n] * h[n] = \sum_{k=0}^{n} x[k]h[n-k]$

For $n = 2$, shift h right again to get $h[2-k]$.



$\Rightarrow$

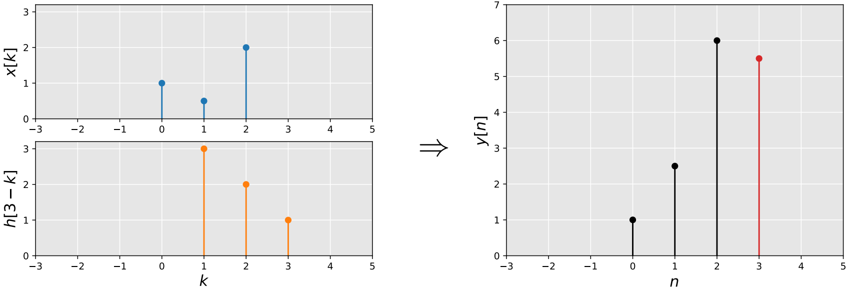


$$
\begin{aligned}
y[2] &= x[0]h[2] + x[1]h[1] + x[2]h[0] \\
&= 1.0 \times 3.0 + 0.5 \times 2.0 + 2.0 \times 1.0 = 6.0
\end{aligned}
$$

Convolution Example

Computing $y[n] = x[n] * h[n] = \sum_{k=0}^{n} x[k]h[n-k]$

For $n = 3$, shift h right again to get $h[3-k]$.

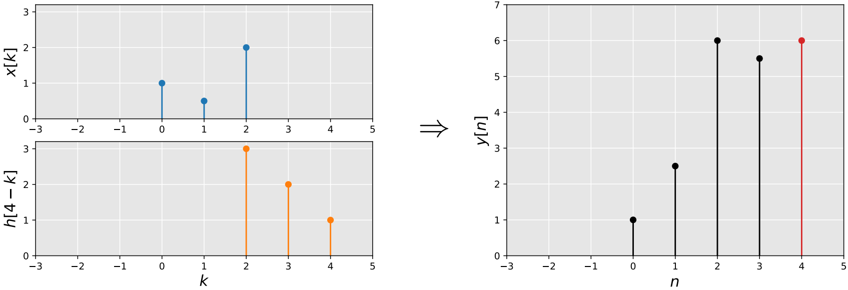


$$
\begin{aligned}
y[3] &= x[1]h[2] + x[2]h[1] \\
&= 0.5 \times 3.0 + 2.0 \times 2.0 = 5.5
\end{aligned}
$$

Convolution Example

Computing $y[n] = x[n] * h[n] = \sum_{k=0}^{n} x[k]h[n-k]$

For $n = 4$, shift h right again to get $h[4-k]$.



$$y[4] = x[2]h[2]$$
$$= 2.0 \times 3.0 = 6.0$$

Output Length

Fact

The convolution of two L-length signals will have length $2L-1$.

$$x[n] * h[n] = \sum_{k=0}^{n} x[k]h[n-k] \qquad \text{for } n \in [0, 2L-2]$$

So, we need to pad $h[n]$ with zeros on the right, from $n = [L, 2L-2]$.

Differing Length Inputs

Fact

If $x[n]$ has length L_x and $h[n]$ has length L_h, then $x[n] * h[n]$ has length $L_x + L_h - 1$.

Need to pad $h[n]$ with zeros to the right, for $n = [L_h, L_x + L_h - 2]$.

Note: It's cheaper to have the longer length signal on the right! (less padding) Because of commutativity, we can always swap $x[n] * h[n] = h[n] * x[n]$.