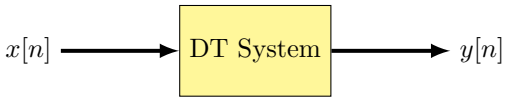
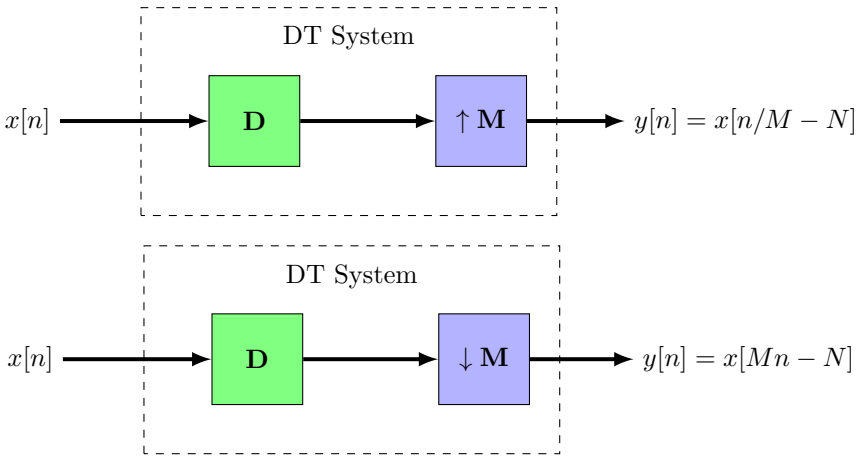
Chapter 2

Discrete-Time Systems

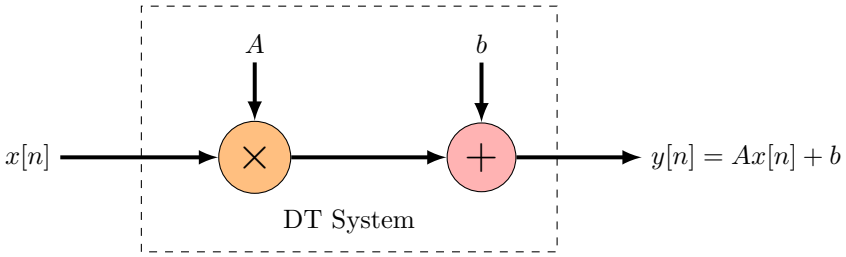
Similar to the continuous case, a discrete-time system essentially transforms an input DT signal into an output DT signal and can be modeled as an operator that maps an input sequence to an output sequence.



An affine transformation on the independent variable is essentially a system consisting of a delay block and either an upsampler or downsampler in the real-time case.



Similarly, an amplitude transformation $y[n] = Ax[n] + b$ can be modeled as a system which inherently has a multiplier and a bias b:



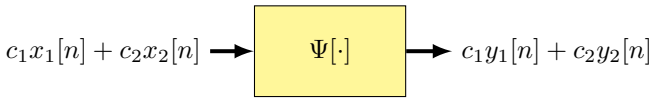
This chapter will cover one-dimensional SISO discrete-time systems and their properties as well as explore the discrete equivalent of LTI systems called *linear shift-invariant (LSI) systems*. Some references still refer to these systems as "LTI systems" due to the indices n representing selected instances of time, while other resources use the term "LSI systems" to emphasize that samples are being manipulated.

This text will use the term LSI systems since DSP is often a precursor to digital image processing, which uses the term "LSI system" as each pixel of an image signal represents a discrete (spatial) sample of two dimensions rather than a discrete (temporal) sample of one dimension.

2.1 Properties of Discrete-Time Systems

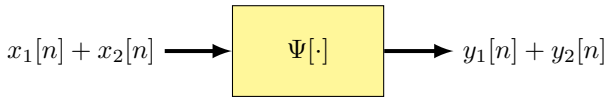
2.1.1 Linear vs Nonlinear

A DT system is said to be *linear* if it follows the *superposition principle*, as depicted in the block diagram below with constants c_1, c_2, input signal addends $x_1[n], x_2[n]$, and output signal addends $y_1[n], y_2[n]$:

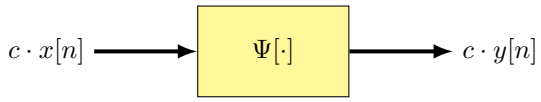


In practice, if the superposition principle is met, then a sum of N input addends will generate a sum of N output addends.

The superposition principle can be broken down into two properties: additivity and scalability. A DT system has *additive property* if for $y_1[n] = \Psi[x_1[n]]$ and $y_2[n] = \Psi[x_2[n]]$, the system-generated output from the summed inputs is the sum of the outputs $\Psi[x_1[n] + x_2[n]] = y_1[n] + y_2[n]$. That is:



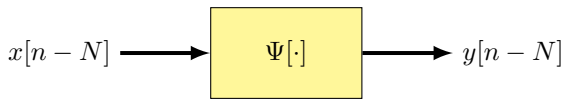
A DT system has *scaling property* if for some constant c, the system-generated output for a scaled input is an appropriately scaled output $\Psi[c \cdot x[n]] = c \cdot y[n]$. That is:



If a system has both scaling and additive properties, then the superposition principle is met, and the system is linear. If the superposition principle is not met, then the system is said to be *nonlinear*.

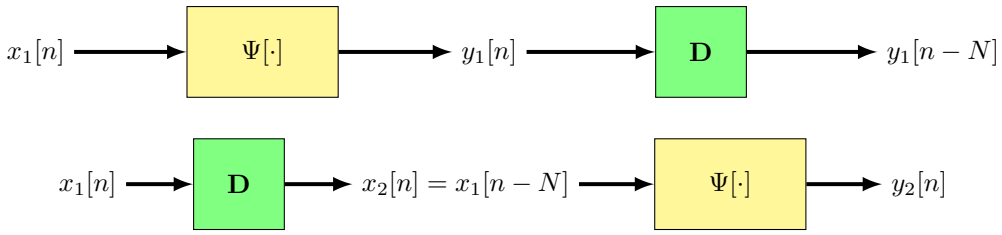
2.1.2 Shift-Invariant vs Shift-Variant

A DT system is said to be *shift-invariant* if a shift in the input signal results in a corresponding shift in the output signal such that $\Psi[x[n - N]] = y[n - N]$. That is:



Some resources refer to this property for DT signals as "time-invariant" as each sample represents an instance in time; here, we will use the more general term "shift-invariant" to highlight the sampled nature of the signals.

A method to check for shift-invariance is to consider the following two block diagrams, with the "D"-block representing a shift of N samples:



From the outputs above, if $y_2[n] = y_1[n-N]$, then the DT system is shift-invariant. Otherwise, if the outputs are not equal, then the DT system is *shift-variant*, in which $\Psi[x[n-N], N] = y[n-N]$ results in an additional dependence on N.

DT systems that are both linear and shift-invariant are called linear shift-invariant (LSI) systems.

2.1.3 Dynamic vs Memoryless

A DT system is said to be *memoryless* or *static* if the output $y[n]$ at time n depends only on the input $x[n]$ at time n. The only LSI DT system that is also memoryless has the form $y[n] = ax[n]$ for a is a constant.

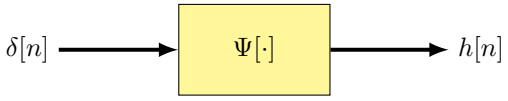
Otherwise, a DT system whose output additionally depends on past and/or future values of the input is a *dynamic* system; that is, the output at n depends on the input at $n - N$ for any $N \neq 0$.

2.1.4 Causal vs Noncausal

Causal systems in discrete-time are DT systems whose output at time n only depend on the past and present values of the input at time $n - N$ for $N > 0$ and time n, respectively. Otherwise, a DT system that anticipates a future value of the input at time $n + N$ before generating an output at time n is said to be *noncausal*.

Essentially, noncausal systems see the future, which is impractical for real-time processing but can be done offline. Because of this, while causality is important in the continuous-time case, the ability to handle offline processing opens opportunities for working with noncausal signals and systems. This is crucial later when considering both the unilateral and bilateral z-transforms.

One way to determine causality is by using an impulse signal $\delta[n]$ as input and observing its output $h[n]$, also called the *impulse response*. If $h[n] = 0$ for all $n < 0$, then the DT system is causal; otherwise it is noncausal. That is, if the impulse response is causal, then the system is causal.



2.1.5 BIBO Stable vs Unstable

A DT signal $x[n]$ is said to be *bounded* if there exists some constant C such that

$$|x[n]| \leq C, \text{ for } \forall n. \tag{2.1}$$

A DT system with a *bounded input, bounded output* (BIBO, for short) is said to be *BIBO stable*; that is, every bounded input signal results in a bounded output signal. Otherwise, if a DT system produces an unbounded output for a bounded input, it is said to be *unstable*.

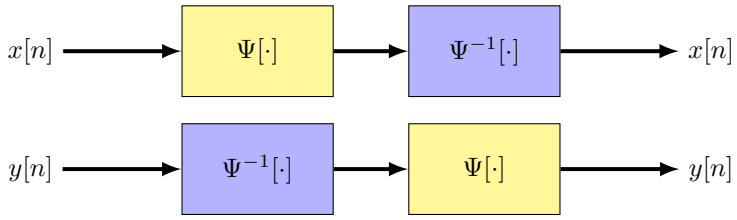
For LSI systems, we can particularly look at the impulse response $h[n]$. An LSI system is BIBO stable if and only if $h[n]$ is *absolutely summable* such that

$$\sum_{n=-\infty}^{+\infty} |h[n]| = C, \text{ for } C \text{ is finite.} \tag{2.2}$$

2.1.6 Invertible vs Non-Invertible

A DT system is said to be *invertible* if it generates unique output signals for every unique input signal. That is, an invertible system has a one-to-one mapping between inputs and outputs. If a DT system has a many-to-one mapping between inputs and outputs, then it is said to be *non-invertible*.

Alternatively, an invertible system has an inverse system $\Psi^{-1}[\cdot]$ that maps outputs back to the inputs of the forward system $\Psi[\cdot]$ – it may be shifted but the shape of the waveform is maintained.



While a DT system can be invertible, it does not mean it is implementable. For this very reason, when inverting LSI systems, generally we are also interested in the properties of the inverse system; that is, if the original system is BIBO stable, we would want the inverse system to be BIBO stable as well. Depending on whether DT systems are implemented in real-time or offline, causality may or may not be of importance for the inverse system.

Example 2.1.1. Fully describe the system characterized by the difference equation

$$y[n] + 2y[n-1] = 3x[n] + nx[n-1].$$

SOLUTION

First, assume $cx[\cdot] \to cy[\cdot]$ for scalability such that

$$(cy[n]) + 2(cy[n-1]) = 3(cx[n]) + n(cx[n-1]).$$

The constant c can be factored such that

$$\begin{aligned} cy[n] + 2cy[n-1] &= 3cx[n] + ncx[n-1] \\ \implies c \cdot (y[n] + 2y[n-1] &= 3x[n] + nx[n-1]). \end{aligned}$$

Therefore, the system is scalable. Checking for additivity, consider the following sum:

$$\begin{aligned} y_1[n] + 2y_1[n-1] &= 3x_1[n] + nx_1[n-1] \\ + \quad y_2[n] + 2y_2[n-1] &= 3x_2[n] + nx_2[n-1] \\ \hline (y_1[n] + y_2[n]) + 2(y_1[n-1] + y_2[n-1]) &= 3(x_1[n] + x_2[n]) + n(x_1[n-1] + x_2[n-1]) \end{aligned}$$

The system is also additive. Thus the system is linear.

Now we check for shift invariance. First, we feed an input and delay the output:

$$\begin{aligned} y_1[n] + 2y_1[n-1] &= 3x_1[n] + nx_1[n-1] \\ y_1[n-N] + 2y_1[(n-N)-1] &= 3x_1[n-N] + (n-N)x_1[(n-N)-1] \end{aligned}$$

If instead we delay the input first and then feed it into the system, we get

$$\begin{aligned} x_2[n] &= x_1[n-N] \\ y_2[n] + 2y_2[n-1] &= 3x_2[n] + nx_2[n-1] = 3x_1[n-N] + nx_1[n-N-1] \end{aligned}$$

Note that $y_1[n-N] + 2y_1[(n-N)-1] \neq y_2[n] + 2y_2[n-1]$ due to the linear term n. Therefore, the system is shift-variant.

Checking for causality, first we input an impulse sequence to attain the impulse response:

$$\begin{aligned} h[n] + 2h[n-1] &= 3\delta[n] + n\delta[n-1] \\ \implies h[n] + 2h[n-1] &= 3\delta[n] + 1\delta[n-1] \end{aligned}$$

This can be simplified to

$$h[n] + 2h[n-1] = C_n$$

where

$$C_n = \begin{cases} 3, & n = 0 \\ 1, & n = 1 \\ 0, & otherwise \end{cases}$$

We can rearrange the recursive equation to be

$$h[n] = -2h[n-1] + C_n,$$

which shows that the impulse response is dependent on its past values (in addition to the present constant value C_n) and not future values. Therefore the system is causal.

As seen in the difference equation, $y[n]$ depends on past values of $x[n]$ as well as $y[n]$ (in addition to present values of $x[n]$). Therefore, the system is dynamic.

Due to the term $nx[n-1]$, the output is not guaranteed to be bounded, regardless of the input signal. As $n \to \infty$, the term also approaches infinity and thus the output approaches infinity. Therefore, the system is not BIBO stable.

Lastly, to check for invertibility, we solve for $x[n]$. First define $y[m] + 2y[m-1] = 3x[m] + mx[m-1]$. Then

$$n = m - 1 \implies m = n + 1.$$

Substituting this relationship in, we get

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

The system is thus invertible.

The DT system can be described as

- linear,
- shift-variant,
- causal,
- unstable,
- dynamic,
- and invertible.

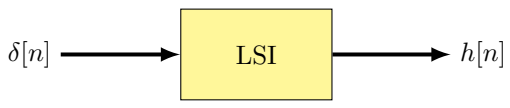
2.2 LSI Systems

By now, we know that an *LSI system* is a DT system that is both linear and shift-invariant. LSI systems are particularly useful for their predictable nature. As long as we know the system response to a few select input signals, we can accurately predict the output for all input signals.

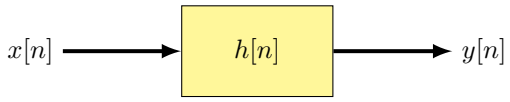
The option to perform either real-time processing or offline processing is what makes DT LSI systems different from CT LTI systems. When modeling CT real systems using LTI systems, one would need to choose an LTI model that is also dynamic, causal, and stable. However, when working with DT LSI systems, one would choose an LSI system that is at least dynamic and stable, with the additional causality condition given for real-time processing applications and not necessarily for offline processing applications.

2.2.1 LSI System Response to Discretized Singularity Signals

As prefaced before, the *impulse response* $h[n]$ of a DT system is the system response to an inputted impulse signal $\delta[n]$, given zero initial conditions. The impulse response of an LSI system is depicted in the following block diagram:



For LSI systems, the impulse response plays an important role. An LSI system can be characterized by its impulse response such that any output signal can be predicted by performing an operation called *linear convolution* between the input signal and the impulse response. The block diagram of an LSI system can be depicted as:



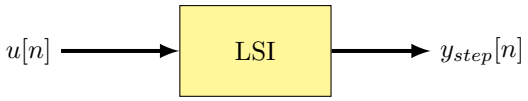
Linear convolution is an operation defined by the *convolution sum* in the discrete-time case:

$$x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] \tag{2.3}$$

$$= \sum_{k=-\infty}^{+\infty} x[n-k]h[k]. \tag{2.4}$$

Here, we specify the more direct term "linear convolution". This is to avoid any confusion when circular convolution is covered in a later chapter.

The *step response* $y_{step}[n]$ of a DT system is the system response to an inputted unit step signal $u[n]$, given zero initial conditions. The step response of an LSI system is depicted in the following block diagram:



While not as prevalent in DSP applications, the step response appears in primarily digital control systems, where we are interested in how well a system "tracks" a step input.

Interestingly, just as $u[n]$ is the cumulative sum of $\delta[n]$, the step response $y_{step}[n]$ is the cumulative sum of the impulse response $h[n]$. That is,

$$h[n] = y_{step}[n] - y_{step}[n-1], \tag{2.5}$$

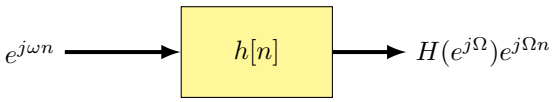
$$y_{step}[n] = \sum_{k=-\infty}^{n} h[k]. \tag{2.6}$$

2.2.2 LSI System Response to DT Exponential and Sinusoidal Signals

Using the convolution sum, we can generalize the LSI system response to a DT complex exponential signal $e^{j\Omega n}$.

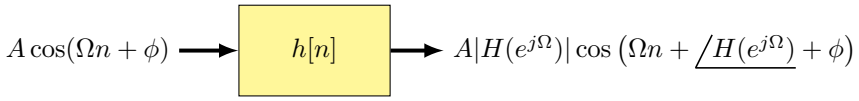
$$\begin{aligned} x[n] * h[n] &= \sum_{k=-\infty}^{+\infty} x[n-k]h[k] \\ \implies e^{j\Omega n} * h[n] &= \sum_{k=-\infty}^{+\infty} e^{j\Omega(n-k)}h[n] \\ &= e^{j\Omega n} \underbrace{\sum_{k=-\infty}^{+\infty} h[k]e^{-j\Omega k}}_{H(e^{j\Omega})}. \end{aligned}$$

Therefore, the LSI system response to a DT complex exponential can be depicted as:



Here, $H(e^{j\Omega})$ is called the *discrete-time frequency response* of the system. Note that $H(e^{j\Omega})$ is independent of time n, is a complex function of $j\Omega$, and is defined for an everlasting complex exponential.

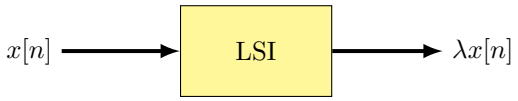
Since sinusoids form the real and imaginary parts of a complex exponential, the LSI sinusoidal response can also be depicted:



In a later chapter, the relationship between the DT impulse reponse and the DT frequency response will be explored. There exists some operator in which the impulse response can be mapped to the frequency response and vice versa.

2.2.3 Eigensequences of LSI Systems

An *eigensequence* of a LSI system is some function $x[n]$ such that the system response is $\lambda x[n]$ for some scalar λ (independent of n). That is,



For LSI systems characterized by $h[n]$, we can use the convolution sum to determine if a sequence is an eigensequence of the system.

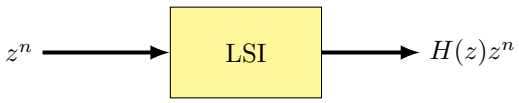
Previously, we saw that for a DT complex exponential input $e^{j\Omega n}$, we get a scaled output $H(e^{j\Omega})e^{j\Omega n}$. Therefore, the complex exponential is an eigensequence of the system. In fact, this particular eigensequence yields the DT frequency response function $H(e^{j\Omega})$, which is crucial for discrete-time Fourier transforms in a later chapter.

Example 2.2.1. Determine if z^n is an eigensequence of LSI systems, for z is a complex variable.

SOLUTION

$$z^n * h[n] = \sum_{k=-\infty}^{+\infty} z^{n-k} h[k] = z^n \underbrace{\sum_{k=-\infty}^{+\infty} h[k] z^{-k}}_{H(z)}$$

Since $H(z)$ is a constant for a given z and is scaling z^n, the function z^n is an eigensequence of LSI systems. In fact, it yields the discrete-time transfer function $H(z)$, which is crucial for z-transforms in a later chapter.



■

Example 2.2.2. Determine if $(1/2)^n u[n]$ is an eigensequence of LSI systems.

SOLUTION

$$(1/2)^n u[n] * h[n] = \sum_{k=-\infty}^{+\infty} (1/2)^{n-k} u[n-k] h[k] = (1/2)^n \sum_{k=-\infty}^{n} h[n] (1/2)^{-k}$$

Since the sum is dependent on time n, it is not a constant. Therefore $(1/2)^n u[n]$ is not an eigensequence of LSI systems.

■

2.2.4 Linear Constant-Coefficient Difference Equations (LCCDE)

LSI systems can be characterized by a class of difference equations called *linear constant-coefficient difference equations* (LCCDE):

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k]. \tag{2.7}$$

Difference equations can be isolated by their parts: the autoregressive (AR) and the moving average (MA) terms.

The *autoregressive* (AR) term is given by

$$\sum_{k=0}^{N} a_k y[n-k] = x[n], \tag{2.8}$$

whereas the *moving average* (MA) term is given by

$$y[n] = \sum_{k=0}^{M} b_k x[n-k]. \tag{2.9}$$

Systems that are described by AR are called autoregressive systems, whereas systems characterized by MA are called moving average systems. Systems that are both AR and MA (i.e., following the general LCCDE) are called autoregressive and moving average (ARMA) systems.