Fundamentals of Signal Processing

Collection Editor:
Minh N. Do

Fundamentals of Signal Processing

Collection Editor:
Minh N. Do

Authors:

Richard Baraniuk
Hyeokho Choi
Minh N. Do
Catherine Elder
Benjamin Fite
Anders Gjendemsjø
Michael Haag
Don Johnson
Douglas L. Jones
Stephen Kruzick
Robert Nowak
Ricardo Radaelli-Sanchez
Justin Romberg
Phil Schniter
Clayton Scott
Ivan Selesnick
Melissa Selik

Online:
< http://cnx.org/content/col10360/1.4/ >

C O N N E X I O N S

Rice University, Houston, Texas

Collection structure revised: November 26, 2012
PDF generated: May 20, 2013
For copyright and attribution information for the modules contained in this collection, see p. 218.

Table of Contents

Introduction to Fundamentals of Signal Processing[1]

What is Digital Signal Processing?

To understand what is **Digital Signal Processing (DSP)** let's examine what does each of its words mean. "**Signal**" is any physical quantity that carries information. "**Processing**" is a series of steps or operations to achieve a particular end. It is easy to see that **Signal Processing** is used everywhere to extract information from signals or to convert information-carrying signals from one form to another. For example, our brain and ears take input speech signals, and then process and convert them into meaningful words. Finally, the word "**Digital**" in Digital Signal Processing means that the process is done by computers, microprocessors, or logic circuits.

The field DSP has expanded significantly over that last few decades as a result of rapid developments in computer technology and integrated-circuit fabrication. Consequently, DSP has played an increasingly important role in a wide range of disciplines in science and technology. Research and development in DSP are driving advancements in many high-tech areas including telecommunications, multimedia, medical and scientific imaging, and human-computer interaction.

To illustrate the digital revolution and the impact of DSP, consider the development of digital cameras. Traditional film cameras mainly rely on physical properties of the optical lens, where higher quality requires bigger and larger system, to obtain good images. When digital cameras were first introduced, their quality were inferior compared to film cameras. But as microprocessors become more powerful, more sophisticated DSP algorithms have been developed for digital cameras to correct optical defects and improve the final image quality. Thanks to these developments, the quality of consumer-grade digital cameras has now surpassed the equivalence in film cameras. As further developments for digital cameras attached to cell phones (cameraphones), where due to small size requirements of the lenses, these cameras rely on DSP power to provide good images. Essentially, digital camera technology uses computational power to overcome physical limitations. We can find the similar trend happens in many other applications of DSP such as digital communications, digital imaging, digital television, and so on.

In summary, DSP has foundations on Mathematics, Physics, and Computer Science, and can provide the key enabling technology in numerous applications.

Overview of Key Concepts in Digital Signal Processing

The two main characters in DSP are **signals** and **systems**. A **signal** is defined as any physical quantity that varies with one or more independent variables such as time (one-dimensional signal), or space (2-D or 3-D signal). Signals exist in several types. In the real-world, most of signals are **continuous-time** or **analog signals** that have values continuously at every value of time. To be processed by a computer, a continuous-time signal has to be first **sampled** in time into a **discrete-time signal** so that its values at a discrete set of time instants can be stored in computer memory locations. Furthermore, in order to be

[1] This content is available online at <http://cnx.org/content/m13673/1.1/>.

processed by logic circuits, these signal values have to be **quantized** in to a set of discrete values, and the final result is called a **digital signal**. When the quantization effect is ignored, the terms discrete-time signal and digital signal can be used interchangeability.

In signal processing, a **system** is defined as a process whose input and output are signals. An important class of systems is the class of **linear time-invariant** (or **shift-invariant**) **systems**. These systems have a remarkable property is that each of them can be completely characterized by an **impulse response function** (sometimes is also called as **point spread function**), and the system is defined by a **convolution** (also referred to as a **filtering**) operation. Thus, a linear time-invariant system is equivalent to a (linear) **filter**. Linear time-invariant systems are classified into two types, those that have **finite-duration impulse response (FIR)** and those that have an **infinite-duration impulse response (IIR)**.

A signal can be viewed as a **vector** in a **vector space**. Thus, **linear algebra** provides a powerful framework to study signals and linear systems. In particular, given a vector space, each signal can be represented (or expanded) as a **linear combination of elementary signals**. The most important **signal expansions** are provided by the **Fourier transforms**. The Fourier transforms, as with general transforms, are often used effectively to transform a problem from one domain to another domain where it is much easier to solve or analyze. The two domains of a Fourier transform have physical meaning and are called the **time domain** and the **frequency domain**.

Sampling, or the conversion of **continuous-domain real-life signals** to **discrete numbers** that can be processed by computers, is the essential bridge between the analog and the digital worlds. It is important to understand the connections between signals and systems in the real world and inside a computer. These connections are convenient to analyze in the frequency domain. Moreover, many signals and systems are specified by their **frequency characteristics**.

Because any **linear time-invariant system** can be characterized as a **filter**, the design of such systems boils down to the design the associated filters. Typically, in the **filter design** process, we determine the coefficients of an FIR or IIR filter that closely approximates the desired **frequency response** specifications. Together with Fourier transforms, the **z-transform** provides an effective tool to analyze and design digital filters.

In many applications, signals are conveniently described via **statistical models** as **random signals**. It is remarkable that optimum linear filters (in the sense of **minimum mean-square error**), so called **Wiener filters**, can be determined using only **second-order statistics** (**autocorrelation** and **crosscorrelation** functions) of a **stationary process**. When these statistics cannot be specified beforehand or change over time, we can employ **adaptive filters**, where the filter coefficients are adapted to the signal statistics. The most popular algorithm to adaptively adjust the filter coefficients is the **least-mean square (LMS)** algorithm.

Chapter 1

Foundations

1.1 Signals Represent Information[1]

Whether analog or digital, information is represented by the fundamental quantity in electrical engineering: the **signal**. Stated in mathematical terms, **a signal is merely a function**. Analog signals are continuous-valued; digital signals are discrete-valued. The independent variable of the signal could be time (speech, for example), space (images), or the integers (denoting the sequencing of letters and numbers in the football score).

1.1.1 Analog Signals

Analog signals are usually signals defined over continuous independent variable(s). Speech[2] is produced by your vocal cords exciting acoustic resonances in your vocal tract. The result is pressure waves propagating in the air, and the speech signal thus corresponds to a function having independent variables of space and time and a value corresponding to air pressure: $s(x,t)$ (Here we use vector notation x to denote spatial coordinates). When you record someone talking, you are evaluating the speech signal at a particular spatial location, x_0 say. An example of the resulting waveform $s(x_0,t)$ is shown in this figure (Figure 1.1: Speech Example).

[1] This content is available online at <http://cnx.org/content/m0001/2.27/>.
[2] "Modeling the Speech Signal" <http://cnx.org/content/m0049/latest/>

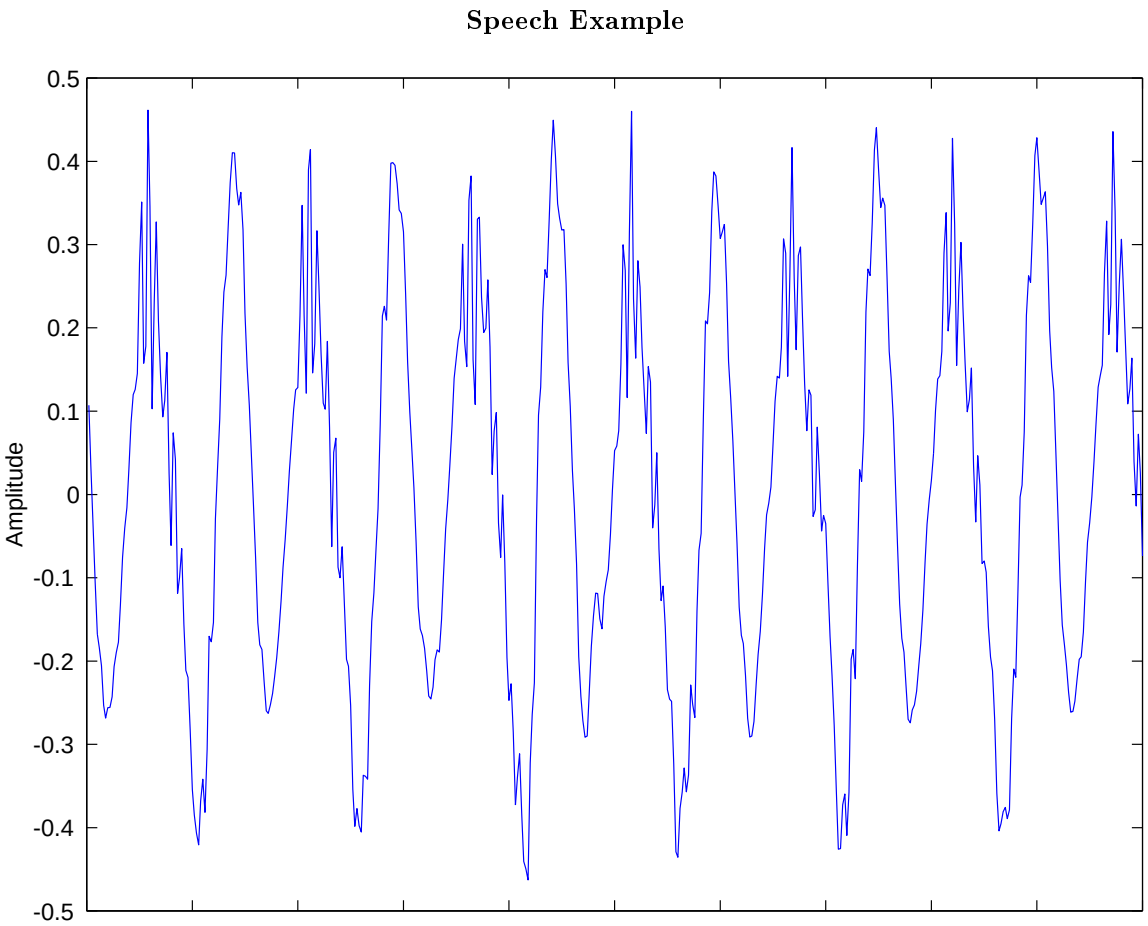
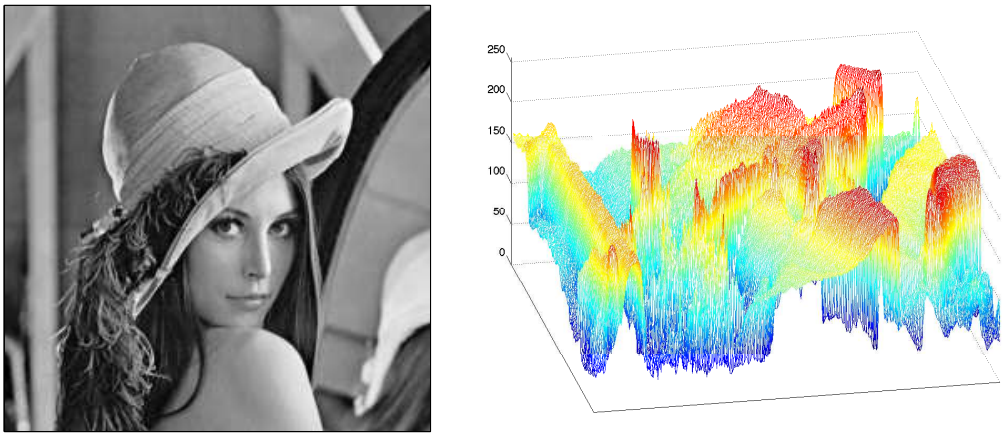


Figure 1.1: A speech signal's amplitude relates to tiny air pressure variations. Shown is a recording of the vowel "e" (as in "speech").

Photographs are static, and are continuous-valued signals defined over space. Black-and-white images have only one value at each point in space, which amounts to its optical reflection properties. In Figure 1.2 (Lena), an image is shown, demonstrating that it (and all other images as well) are functions of two independent spatial variables.

Lena



(a) (b)

Figure 1.2: On the left is the classic **Lena** image, which is used ubiquitously as a test image. It contains straight and curved lines, complicated texture, and a face. On the right is a perspective display of the Lena image as a signal: a function of two spatial variables. The colors merely help show what signal values are about the same size. In this image, signal values range between 0 and 255; why is that?

Color images have values that express how reflectivity depends on the optical spectrum. Painters long ago found that mixing together combinations of the so-called primary colors–red, yellow and blue–can produce very realistic color images. Thus, images today are usually thought of as having three values at every point in space, but a different set of colors is used: How much of red, **green** and blue is present. Mathematically, color pictures are multivalued–vector-valued–signals: $s(x) = (r(x), g(x), b(x))^T$.

Interesting cases abound where the analog signal depends not on a continuous variable, such as time, but on a discrete variable. For example, temperature readings taken every hour have continuous–analog–values, but the signal's independent variable is (essentially) the integers.

1.1.2 Digital Signals

The word "digital" means discrete-valued and implies the signal has an integer-valued independent variable. Digital information includes numbers and symbols (characters typed on the keyboard, for example). Computers rely on the digital representation of information to manipulate and transform information. Symbols do not have a numeric value, and each is represented by a unique number. The ASCII character code has the upper- and lowercase characters, the numbers, punctuation marks, and various other symbols represented by a seven-bit integer. For example, the ASCII code represents the letter **a** as the number 97 and the letter **A** as 65. Table 1.1: ASCII Table shows the international convention on associating characters with integers.

ASCII Table

00	nul	01	soh	02	stx	03	etx	04	eot	05	enq	06	ack	07	bel
08	bs	09	ht	0A	nl	0B	vt	0C	np	0D	cr	0E	so	0F	si
10	dle	11	dc1	12	dc2	13	dc3	14	dc4	15	nak	16	syn	17	etb
18	car	19	em	1A	sub	1B	esc	1C	fs	1D	gs	1E	rs	1F	us
20	sp	21	!	22	"	23	#	24	$	25	%	26	&	27	’
28	(	29	)	2A	*	2B	+	2C	,	2D	-	2E	.	2F	/
30	0	31	1	32	2	33	3	34	4	35	5	36	6	37	7
38	8	39	9	3A	:	3B	;	3C	<	3D	=	3E	>	3F	?
40	@	41	A	42	B	43	C	44	D	45	E	46	F	47	G
48	H	49	I	4A	J	4B	K	4C	L	4D	M	4E	N	4F	0
50	P	51	Q	52	R	53	S	54	T	55	U	56	V	57	W
58	X	59	Y	5A	Z	5B	[	5C	\	5D	]	5E	^	5F	_
60	’	61	a	62	b	63	c	64	d	65	e	66	f	67	g
68	h	69	i	6A	j	6B	k	6C	l	6D	m	6E	n	6F	o
70	p	71	q	72	r	73	s	74	t	75	u	76	v	77	w
78	x	79	y	7A	z	7B	{	7C	|	7D	}	7E	~	7F	del

Table 1.1: The ASCII translation table shows how standard keyboard characters are represented by integers. In pairs of columns, this table displays first the so-called 7-bit code (how many characters in a seven-bit code?), then the character the number represents. The numeric codes are represented in hexadecimal (base-16) notation. Mnemonic characters correspond to control characters, some of which may be familiar (like **cr** for carriage return) and some not (**bel** means a "bell").

1.2 Introduction to Systems[3]

Signals are manipulated by systems. Mathematically, we represent what a system does by the notation $y(t) = S(x(t))$, with x representing the input signal and y the output signal.

Definition of a system

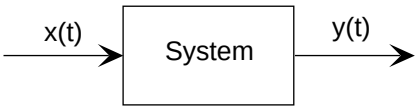


Figure 1.3: The system depicted has input $x(t)$ and output $y(t)$. Mathematically, systems operate on function(s) to produce other function(s). In many ways, systems are like functions, rules that yield a value for the dependent variable (our output signal) for each value of its independent variable (its input signal). The notation $y(t) = S(x(t))$ corresponds to this block diagram. We term $S(\cdot)$ the input-output relation for the system.

[3]This content is available online at <http://cnx.org/content/m0005/2.19/>.

This notation mimics the mathematical symbology of a function: A system's input is analogous to an independent variable and its output the dependent variable. For the mathematically inclined, a system is a **functional**: a function of a function (signals are functions).

Simple systems can be connected together–one system's output becomes another's input–to accomplish some overall design. Interconnection topologies can be quite complicated, but usually consist of weaves of three basic interconnection forms.

1.2.1 Cascade Interconnection

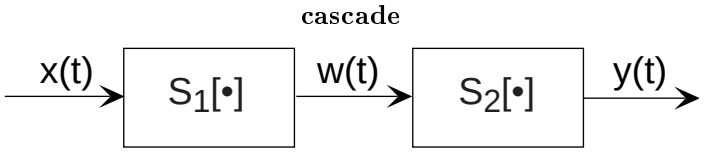


Figure 1.4: The most rudimentary ways of interconnecting systems are shown in the figures in this section. This is the cascade configuration.

The simplest form is when one system's output is connected only to another's input. Mathematically, $w(t) = S_1(x(t))$, and $y(t) = S_2(w(t))$, with the information contained in $x(t)$ processed by the first, then the second system. In some cases, the ordering of the systems matter, in others it does not. For example, in the fundamental model of communication [4] the ordering most certainly matters.

1.2.2 Parallel Interconnection

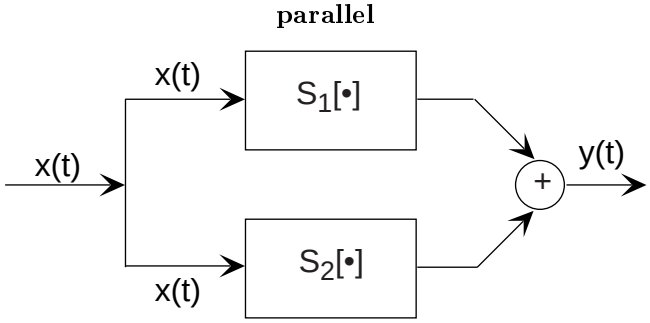


Figure 1.5: The parallel configuration.

A signal $x(t)$ is routed to two (or more) systems, with this signal appearing as the input to all systems simultaneously and with equal strength. Block diagrams have the convention that signals going to more

[4] "Structure of Communication Systems", Figure 1: Fundamental model of communication <http://cnx.org/content/m0002/latest/#commsys>

than one system are not split into pieces along the way. Two or more systems operate on $x(t)$ and their outputs are added together to create the output $y(t)$. Thus, $y(t) = S_1(x(t)) + S_2(x(t))$, and the information in $x(t)$ is processed separately by both systems.

1.2.3 Feedback Interconnection

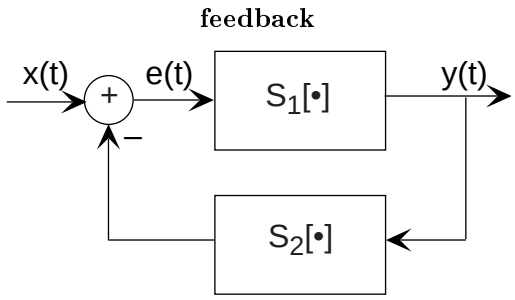


Figure 1.6: The feedback configuration.

The subtlest interconnection configuration has a system's output also contributing to its input. Engineers would say the output is "fed back" to the input through system 2, hence the terminology. The mathematical statement of the feedback interconnection (Figure 1.6: feedback) is that the feed-forward system produces the output: $y(t) = S_1(e(t))$. The input $e(t)$ equals the input signal minus the output of some other system's output to $y(t)$: $e(t) = x(t) - S_2(y(t))$. Feedback systems are omnipresent in control problems, with the error signal used to adjust the output to achieve some condition defined by the input (controlling) signal. For example, in a car's cruise control system, $x(t)$ is a constant representing what speed you want, and $y(t)$ is the car's speed as measured by a speedometer. In this application, system 2 is the identity system (output equals input).

1.3 Discrete-Time Signals and Systems[5]

Mathematically, analog signals are functions having as their independent variables continuous quantities, such as space and time. Discrete-time signals are functions defined on the integers; they are sequences. As with analog signals, we seek ways of decomposing discrete-time signals into simpler components. Because this approach leads to a better understanding of signal structure, we can exploit that structure to represent information (create ways of representing information with signals) and to extract information (retrieve the information thus represented). For symbolic-valued signals, the approach is different: We develop a common representation of all symbolic-valued signals so that we can embody the information they contain in a unified way. From an information representation perspective, the most important issue becomes, for both real-valued and symbolic-valued signals, efficiency: what is the most parsimonious and compact way to represent information so that it can be extracted later.

1.3.1 Real- and Complex-valued Signals

A discrete-time signal is represented symbolically as $s(n)$, where $n = \{\ldots, -1, 0, 1, \ldots\}$.

[5]This content is available online at <http://cnx.org/content/m10342/2.16/>.

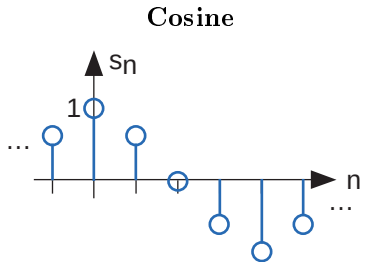


Figure 1.7: The discrete-time cosine signal is plotted as a stem plot. Can you find the formula for this signal?

We usually draw discrete-time signals as stem plots to emphasize the fact they are functions defined only on the integers. We can delay a discrete-time signal by an integer just as with analog ones. A signal delayed by m samples has the expression $s(n-m)$.

1.3.2 Complex Exponentials

The most important signal is, of course, the **complex exponential sequence**.

$$s(n) = e^{i2\pi fn} \tag{1.1}$$

Note that the frequency variable f is dimensionless and that adding an integer to the frequency of the discrete-time complex exponential has no effect on the signal's value.

$$\begin{aligned} e^{i2\pi(f+m)n} &= e^{i2\pi fn}e^{i2\pi mn} \\ &= e^{i2\pi fn} \end{aligned} \tag{1.2}$$

This derivation follows because the complex exponential evaluated at an integer multiple of 2π equals one. Thus, we need only consider frequency to have a value in some unit-length interval.

1.3.3 Sinusoids

Discrete-time sinusoids have the obvious form $s(n) = A\cos(2\pi fn + \phi)$. As opposed to analog complex exponentials and sinusoids that can have their frequencies be any real value, frequencies of their discrete-time counterparts yield unique waveforms **only** when f lies in the interval $\left(-\frac{1}{2}, \frac{1}{2}\right]$. This choice of frequency interval is arbitrary; we can also choose the frequency to lie in the interval $[0, 1)$. How to choose a unit-length interval for a sinusoid's frequency will become evident later.

1.3.4 Unit Sample

The second-most important discrete-time signal is the **unit sample**, which is defined to be

$$\delta(n) = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{otherwise} \end{cases} \tag{1.3}$$

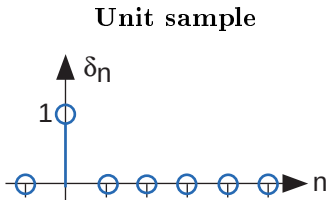


Figure 1.8: The unit sample.

Examination of a discrete-time signal's plot, like that of the cosine signal shown in Figure 1.7 (Cosine), reveals that all signals consist of a sequence of delayed and scaled unit samples. Because the value of a sequence at each integer m is denoted by $s(m)$ and the unit sample delayed to occur at m is written $\delta(n-m)$, we can decompose **any** signal as a sum of unit samples delayed to the appropriate location and scaled by the signal value.

$$s(n) = \sum_{m=-\infty}^{\infty} s(m)\,\delta(n-m) \tag{1.4}$$

This kind of decomposition is unique to discrete-time signals, and will prove useful subsequently.

1.3.5 Unit Step

The **unit step** in discrete-time is well-defined at the origin, as opposed to the situation with analog signals.

$$u(n) = \begin{cases} 1 & \text{if } n \geq 0 \\ 0 & \text{if } n < 0 \end{cases} \tag{1.5}$$

1.3.6 Symbolic Signals

An interesting aspect of discrete-time signals is that their values do not need to be real numbers. We do have real-valued discrete-time signals like the sinusoid, but we also have signals that denote the sequence of characters typed on the keyboard. Such characters certainly aren't real numbers, and as a collection of possible signal values, they have little mathematical structure other than that they are members of a set. More formally, each element of the **symbolic-valued** signal $s(n)$ takes on one of the values $\{a_1, \ldots, a_K\}$ which comprise the **alphabet** A. This technical terminology does not mean we restrict symbols to being members of the English or Greek alphabet. They could represent keyboard characters, bytes (8-bit quantities), integers that convey daily temperature. Whether controlled by software or not, discrete-time systems are ultimately constructed from digital circuits, which consist **entirely** of analog circuit elements. Furthermore, the transmission and reception of discrete-time signals, like e-mail, is accomplished with analog signals and systems. Understanding how discrete-time and analog signals and systems intertwine is perhaps the main goal of this course.

1.3.7 Discrete-Time Systems

Discrete-time systems can act on discrete-time signals in ways similar to those found in analog signals and systems. Because of the role of software in discrete-time systems, many more different systems can be envisioned and "constructed" with programs than can be with analog signals. In fact, a special class of analog signals can be converted into discrete-time signals, processed with software, and converted back into

an analog signal, all without the incursion of error. For such signals, systems can be easily produced in software, with equivalent analog realizations difficult, if not impossible, to design.

1.4 Linear Time-Invariant Systems[6]

A discrete-time signal $s(n)$ is **delayed** by n_0 samples when we write $s(n-n_0)$, with $n_0 > 0$. Choosing n_0 to be negative advances the signal along the integers. As opposed to analog delays[7], discrete-time delays can **only** be integer valued. In the frequency domain, delaying a signal corresponds to a linear phase shift of the signal's discrete-time Fourier transform: $s(n-n_0) \leftrightarrow e^{-(i2\pi f n_0)} S\left(e^{i2\pi f}\right)$.

Linear discrete-time systems have the superposition property.

Superposition

$$S(a_1 x_1(n) + a_2 x_2(n)) = a_1 S(x_1(n)) + a_2 S(x_2(n)) \tag{1.6}$$

A discrete-time system is called **shift-invariant** (analogous to time-invariant analog systems) if delaying the input delays the corresponding output.

Shift-Invariant

$$\text{If} \quad S(x(n)) = y(n), \text{ Then} \quad S(x(n-n_0)) = y(n-n_0) \tag{1.7}$$

We use the term shift-invariant to emphasize that delays can only have integer values in discrete-time, while in analog signals, delays can be arbitrarily valued.

We want to concentrate on systems that are both linear and shift-invariant. It will be these that allow us the full power of frequency-domain analysis and implementations. Because we have no physical constraints in "constructing" such systems, we need only a mathematical specification. In analog systems, the differential equation specifies the input-output relationship in the time-domain. The corresponding discrete-time specification is the **difference equation**.

The Difference Equation

$$y(n) = a_1 y(n-1) + \cdots + a_p y(n-p) + b_0 x(n) + b_1 x(n-1) + \cdots + b_q x(n-q) \tag{1.8}$$

Here, the output signal $y(n)$ is related to its **past** values $y(n-l)$, $l = \{1, \ldots, p\}$, and to the current and past values of the input signal $x(n)$. The system's characteristics are determined by the choices for the number of coefficients p and q and the coefficients' values $\{a_1, \ldots, a_p\}$ and $\{b_0, b_1, \ldots, b_q\}$.

> ASIDE: There is an asymmetry in the coefficients: where is a_0 ? This coefficient would multiply the $y(n)$ term in the difference equation (1.8: The Difference Equation). We have essentially divided the equation by it, which does not change the input-output relationship. We have thus created the convention that a_0 is always one.

As opposed to differential equations, which only provide an **implicit** description of a system (we must somehow solve the differential equation), difference equations provide an **explicit** way of computing the output for any input. We simply express the difference equation by a program that calculates each output from the previous output values, and the current and previous inputs.

1.5 Discrete Time Convolution[8]

1.5.1 Introduction

Convolution, one of the most important concepts in electrical engineering, can be used to determine the output a system produces for a given input signal. It can be shown that a linear time invariant system is

[6] This content is available online at <http://cnx.org/content/m0508/2.7/>.
[7] "Simple Systems": Section Delay <http://cnx.org/content/m0006/latest/#delay>
[8] This content is available online at <http://cnx.org/content/m10087/2.27/>.