Chapter 1

Discrete-time signals

In general terms, a discrete-time signal is an *indexed sequence* of complex numbers called *samples*; the number of elements in the sequence can be finite or infinite. When the values in the sequence are a function of an *integer* index $n \in \mathbb{Z}$, the index is usually interpreted as an abstract timing reference and the one-dimensional signal provides a good model to encode successive measurements of a quantity that varies over time. Two-dimensional signals are also common; in this case the samples are indexed by a pair of integer coordinates and the signals are a natural representation for the pixels in a digital image.

In this chapter we will explore a few examples of discrete-time signals, establish an initial taxonomy of signal classes together with their properties, and we will illustrate some of the fundamental operations used in processing.

1.1 Notational Conventions

Digital signal processing is a relatively recent discipline whose broad interdisciplinary reach attracted researchers from vastly different backgrounds. In the absence of an established notational standard, many of the early DSP scholars simply borrowed the symbolism used in their domain of origin which resulted in today's vast and sometimes contradictory range of conventions (one may even say *quirks*) that we encounter today in the signal processing literature. Unfortunately, in spite of their acceptance, not all of these conventions turn out to be at the service of coherence and clarity.

By way of example, consider the notation commonly used for a discrete-time signal, that is, a *sequence* of values indexed by an integer variable; the expression

$$x[n] = \cos(2\pi n/3) \tag{1.1}$$

uses square brackets in the left-hand side to highlight the integer-valued nature of the "time" index, and conventional parentheses for the real-valued argument of the trigonometric function. The enduring success of this notation comes from the fact a discrete-time signal is a countable and ordered set of values, that is, an array, and most programming languages use brackets in their array-indexing syntax. However, when the expression $x[n]$ is used to denote the *entire* signal and not just its value at index n, then we start to run into issues. Of course this abuse of notation is hardly a prerogative of signal processing: in calculus it's customary to denote an entire function by $f(t)$, an expression that should only be used to indicate the value of f when computed in t. But in signal processing the problem is a bit more delicate because of its aforementioned algorithmic nature; for instance:

- finite-length signals are *defined* only for a finite set of index values (e.g., for $0 \leq n < N$); in these cases, we need to always specify the legal domain for the expression $x[n]$ to avoid what, in an algorithmic implementation, would result in an access violation exception. Furthermore, for finite-length signals, a shifted version of the signal notated as $x[n - M]$ is almost always a meaningless expression

- most signal manipulations are *operators* acting on the entire signal. If we consider the Discrete Fourier Transform, for instance, a commonly used notation is

  $$X[k] = \text{DFT}\{x[n]\}$$

  by which we indicate that the frequency-domain data points are a transformation of the entire set of the time-domain data. However it's far from clear what the indexes on the left- and right-hand side of the expression stand for: are they just placeholders or are they related?

- placeholder indexes become extraordinarily confusing when the convolution of two sequences is written as

  $$y[n] = h[n] * x[n]. \tag{1.2}$$

  If we focus on a single value of the output sequence and write out the convolution "algorithmically"

  $$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k], \tag{1.3}$$

  then it's clear that n is the "anchor point" for the computation of the sum. But what is the meaning of the two n's in the right-hand side of (1.2)? Which of the following expressions is correct?

  $$\begin{aligned} y[n-2] &= h[n-2] * x[n-2] \\ y[n-2] &= h[n-1] * x[n-1] \\ y[n-2] &= h[n] * x[n-2] \end{aligned}$$

Of course, in these notes we are not going to try and fix all of these problems by introducing yet another set of notational conventions; after all, the main purpose of a course is to prepare its readers so that they can ultimately tackle the "raw" research literature available on its subject, and part of this training includes the exposure to established practices. We will be rather pragmatic, and follow what we feel is a good compromise between tradition and clarity:

- for us, discrete-time signals are first and foremost elements of a given vector space; we will therefore use boldface characters when indicating the entire signal[1] and normal type with square brackets to specify the value of a signal at a given discrete time:

  $$\mathbf{x} \in \mathbb{C}^N, \quad x[n] = e^{j\frac{2\pi}{N}n};$$

- signal operators will be indicated with calligraphic letters: the notation

  $$\mathbf{y} = \mathcal{H}\mathbf{x}$$

  indicates that $\mathbf{y}$ is the result of applying the operator $\mathcal{H}$ (e.g. a delay, a filter, or a transform) to the input $\mathbf{x}$. For finite-length signals and linear operators, this notation exactly mirrors the fact that the operator can be implemented as a matrix-vector multiplication. Sometimes, to enhance the algorithmic nature of some operators, we may choose to use explicit names and curly braces, as in the expression

  $$\mathbf{X} = \text{DFT}\{\mathbf{x}\}, \quad \mathbf{X}, \mathbf{x} \in \mathbb{C}^N.$$

  The operator's output can be explicitly addressed by a time index:

  $$y[n] = (\mathcal{H}\mathbf{x})[n];$$

  the same convention will be used for the convolution:

  $$\mathbf{y} = \mathbf{h} * \mathbf{x}, \quad \mathbf{y}, \mathbf{h}, \mathbf{x} \in L_2(\mathbb{Z})$$

  and

  $$y[n] = (\mathbf{h} * \mathbf{x})[n];$$

- Discrete-Time Fourier Transforms, which are complex-valued, 2π-periodic functions of a real-valued variable, are also elements of a specific vector space and so the relationship between a square-summable infinite sequence and its DTFT will be

  $$\mathbf{X} = \text{DTFT}\{\mathbf{x}\}, \quad \mathbf{x} \in \ell_2(\mathbb{Z}), \quad \mathbf{X} \in L_2([-\pi, \pi]);$$

[1]The real practical problem that prevented a wider adoption of vector notation is that it's really inconvenient to draw boldface characters by hand; while alternate and more handwriting-friendly vector symbols exist (e.g. $\vec{x}$), they are still "busy-looking" and can be easily mistook for other common symbols (e.g. $\bar{x}$ for a mean or $\hat{x}$ for an estimate). On the other hand, once the use of $\mathbf{x}$ is established to indicate the whole signal, one needs not be too strict with the boldface convention and the meaning of the lone symbol x will be entirely clear.

explicitly, the actual expression for the transform will be notated in a form like this[2]

$$X(\omega) = 1 - 2\cos(\omega).$$

- when dealing with continuous-time signals we will usually stick to the common tradition of using the function notation $x(t)$ with t expressed in seconds, and, for the associated Fourier transform, we will write $X(f)$, with f expressed in Hertz.

1.2 Examples of discrete-time signals

A discrete-time signal **x** can be defined analytically as a function of a "time" index $n \in \mathbb{Z}$; for example[3]:

$$x[n] = (n \mod 11) - 5 \tag{1.4}$$

shown as the "triangular" waveform plotted in Figure 1.1; or

$$x[n] = e^{j\frac{\pi}{20}n} \tag{1.5}$$

which is a complex exponential of period 40 samples, plotted in Figure 1.2. Another example, this time of a random sequence whose samples are uniformly distributed between −1 and 1, is

$$x[n] = \text{the } n\text{-th output of a random source } \mathcal{U}(-1, 1) \tag{1.6}$$

[2]After a long internal struggle, we finally decided to let go of the beloved but ultimately uselessly baroque notation $X(e^{j\omega})$.

[3]Here and in the rest of the notes the modulus operation always returns a positive value, that is, $a \mod N$ is the remainder of the Euclidean division of a by N. E.g., $(6 \mod 5) = (-1 \mod 5) = 1$.

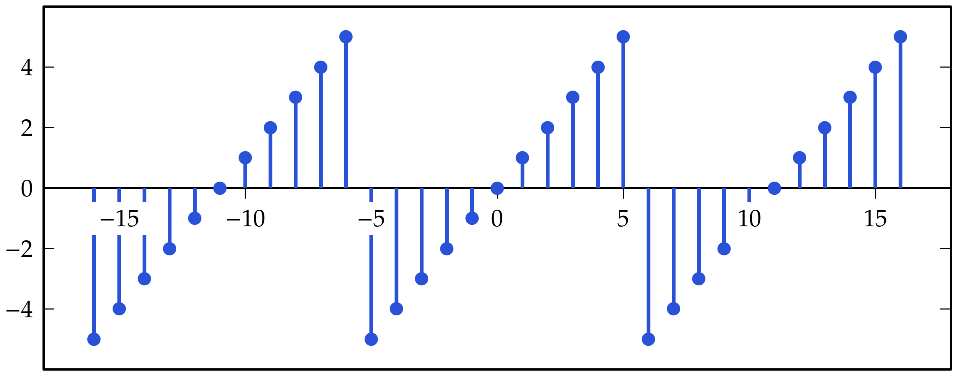

Figure 1.1: Triangular discrete-time wave.

a realization of which is plotted in Figure 1.3. Finally, an example of a sequence drawn from the real world and for which, therefore, no analytical expression exists, is

$$x[n] = \text{The average Dow-Jones index in year } n \tag{1.7}$$

plotted in Figure 1.4 from year 1900 to 2007.

A few notes are in order:

- The sequence index n is best thought of as a measure of *dimensionless time*; while it has no physical unit of measure, it imposes a chronological order on the values of the sequences.
- We consider complex-valued discrete-time signals; while physical signals can be expressed by real quantities, the generality offered by the complex domain is particularly useful in designing systems which *synthesize* signal, such as data communication systems.
- In graphical representations, when we need to emphasize the discrete-time nature of the signal, we resort to stem (or "lollipop") plots as in Figure 1.1; this works well for plots that show a small number of data points. For lager data sets, when the discrete-time domain is implicitly understood, we will often use a function-like representation as in Figure 1.4. In the latter case, each ordinate of the sequence is graphically connected to its neighboring data points, giving the illusion of a smooth line; while this makes the plot easier on the eye, it must be remembered that the signal is defined only over a *discrete* set of abscissas.

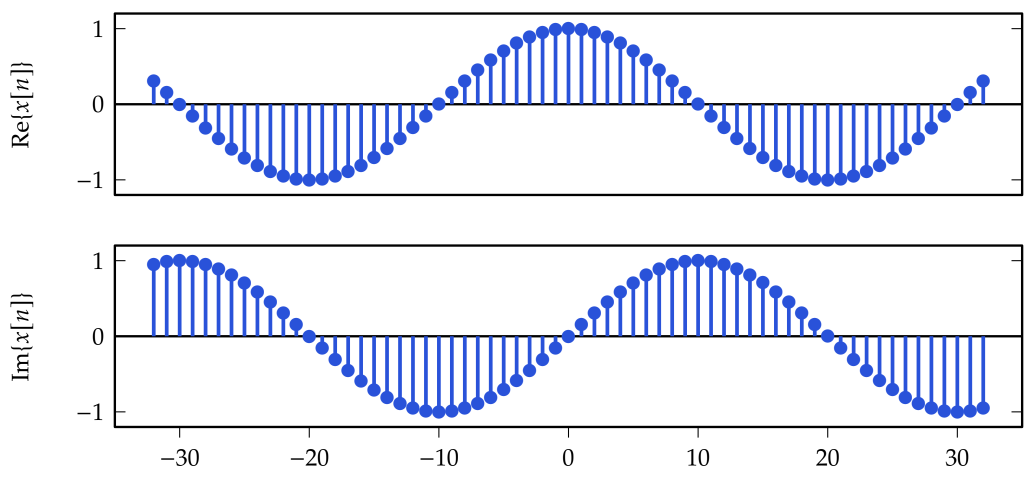

Figure 1.2: *Discrete-time complex exponential $x[n] = e^{j\frac{\pi}{20}n}$ (real and imaginary parts).*

1.3 Types of Discrete-Time Signals

The examples described by (1.4) or (1.5) represent two-sided, infinite sequences. Of course, in the practice of signal processing, it is impossible to deal with an infinite amount of data: for a processing algorithm to execute in a finite amount of time and to use a finite amount of storage, the input must be of finite length; even for algorithms that operate on the fly, i.e. algorithms that produce an output sample for each new input sample, an implicit cap on the input data size is imposed by the necessarily limited life span of the processing device.[4] This problem was clearly swept under the rug in our attempts to plot the signals in Figure 1.1 and 1.2: what the diagrams show, in fact, is only a representative, meaningful portion of the data, and we rely on the original mathematical expressions for a full characterization.

Signals sampled from the real world, for which no analytical description exists, always possess a finite time support because of the finite time spent recording the underlying phenomena; for the Dow Jones index, for instance, the data does not exist for years before 1884, when the index was introduced, and future values are certainly not known. More importantly (and more often), the finiteness of a discrete-time signal is explicitly imposed by design since we are interested in concentrating our processing efforts on a small portion of an otherwise longer signal; in a speech recognition system, for instance, the practice is to cut a speech signal into small segments and try to identify the phonemes associated to each one of them.[5] These finite sets of data are not, strictly speaking, sequences in the mathematical sense, although they can be formally extended into one, as we will see.

Another case is that of *periodic* signals, as in (1.5); even though these are indeed infinite sequences, it is clear that all the relevant information is contained in just a single period. By describing one period (graphically or otherwise), we are, in fact, providing a full description of the entire sequence. This variety of signal types demands a little taxonomic effort.

[4]Or that of the supervising engineer . . .

[5]Note that, in the end, phonemes are pasted together into words and words into sentences; therefore, for a complete speech recognition system, long-range dependencies become important again.

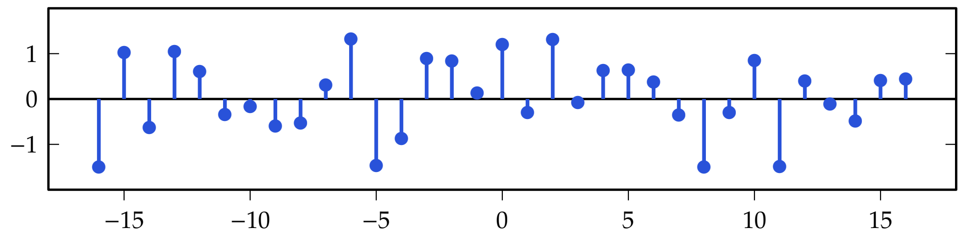

Figure 1.3: *Random discrete-time signal.*

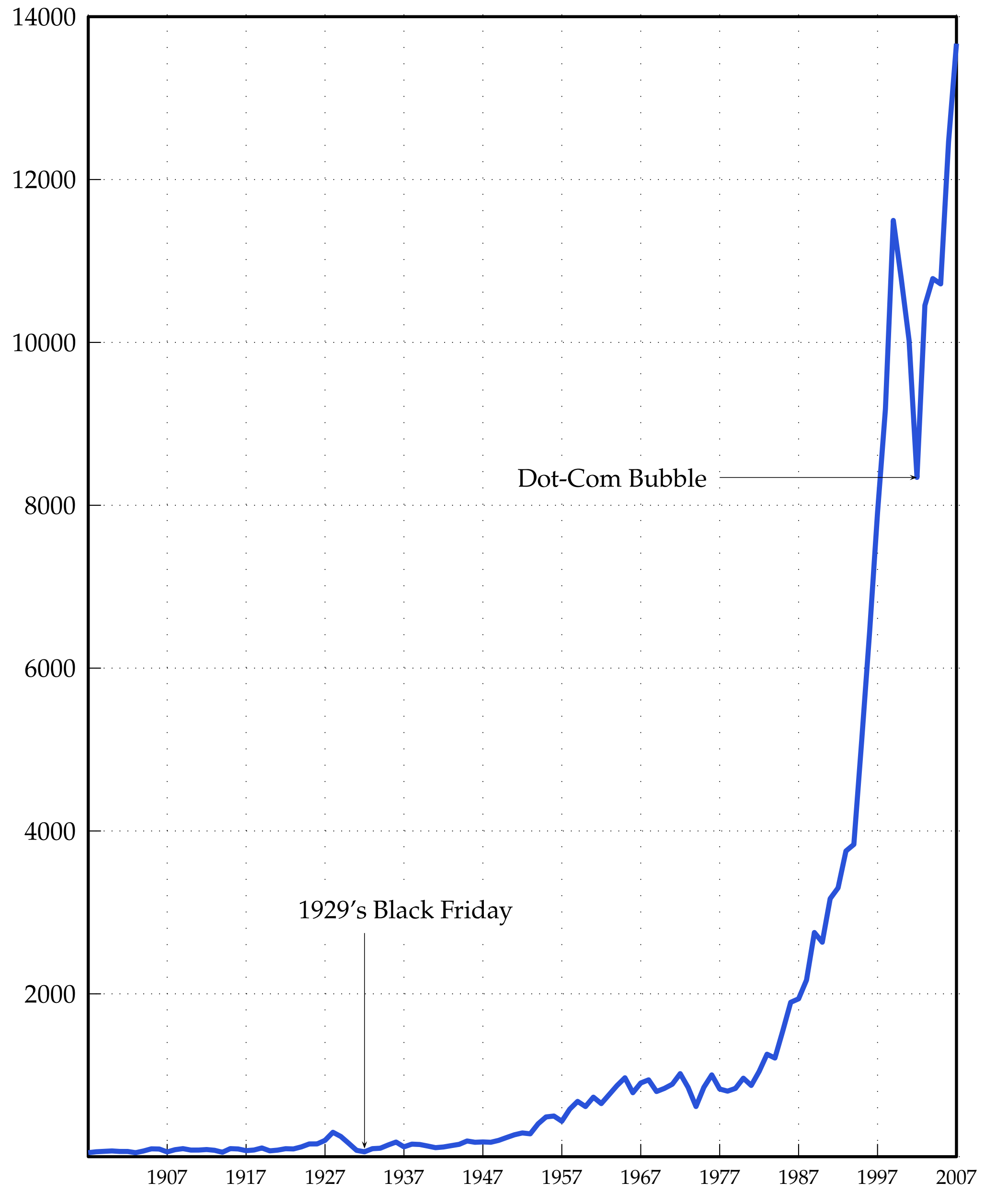

Figure 1.4: *The Dow-Jones industrial index.*

1.3.1 Finite-Length Signals

A finite-length discrete-time signal of length N is simply a collection (or a *tuple*) of N complex values. It is both natural and convenient to consider finite-length signals as Euclidean vectors in the N-dimensional space $\mathbb{C}^N$, and we will explore this viewpoint in detail in the next chapter. The key point for now is that the object that represents an N-point, finite-length signal is the vector

$$\mathbf{x} = \begin{bmatrix} x_0 & x_1 & \dots & x_{N-1} \end{bmatrix}^T;$$

note the transpose operator that shows the convention of using *column* vectors. The individual element of the signal is denoted by the expression

$$x[n], \qquad n = 0, \dots, N-1;$$

the range of the index n is finite and, for values outside of the $[0, N-1]$ interval, the value $x[n]$ is simply not defined — although, as we will see shortly, the natural solution to avoid these "access violation" errors for the index n is to take its value modulo N, which corresponds to implicitly assuming a periodic repetition of the base signal.

As we said, finite-length signals are the only actual entities that we can manipulate in practical signal processing applications; it would be extremely awkward, however, to develop the whole theory of signal processing only in terms of finite-length signals. In fact, it is more convenient, when possible, to develop theoretical proofs using infinite sequences since such general results will hold for all finite-size signals as well.

1.3.2 Infinite-Length Signals

Infinite-length signals are proper sequences in the mathematical sense; although these kind of signals in general lie beyond our processing and storage capabilities, they are useful from a theoretical point of view since they allow us to prove general theorems that apply to all other signal classes. We can define three subclasses as follows.

Aperiodic Signals. The most general type of discrete-time signal is represented by an aperiodic, two-sided complex sequence. In the theory of signal processing, we tend to like sequences that are in some way *summable*, since summability is associated with desirable mathematical properties. In particular, *absolute summability* (a strong condition) is associated with system stability, while *square summability* (a weaker condition) is associated to finite energy. In general, theoretical proofs will be easier when the signals involved are assumed to be absolutely summable[6] whereas extending the proofs to square summable sequences is often quite a bit of work.

Periodic Signals. An N-periodic sequence $\tilde{\mathbf{x}}$ is an infinite, two-sided sequence for which

$$\tilde{x}[n] = \tilde{x}[n + pN], \qquad p \in \mathbb{Z}. \tag{1.8}$$

[6] Mostly because of Fubini's theorem, thanks to which we can freely reorder summations and integrals.

The tilde notation will be used whenever we want to explicitly stress a periodic behavior. Clearly an N-periodic sequence is completely defined by its N values over a period; that is, a periodic sequence "carries no more information" than a finite-length signal of length N. In this sense, periodic signals represent the first (and, for reasons that will be clear later on, the most natural) bridge between finite-length signals and infinite sequences, since we can always convert a finite-length signal $\mathbf{x} \in \mathbb{C}^N$ into a sequence via the periodic extension $\tilde{\mathbf{x}}$:

$$\tilde{x}[n] = x[n \mod N], \qquad n \in \mathbb{Z}. \tag{1.9}$$

Finite-Support Signals. A discrete-time sequence $\bar{\mathbf{x}}$ is said to have *finite support* if its values are zero for all indexes outside of a given range, that is, if there exist two values $N \in \mathbb{N}^+$ and $M \in \mathbb{Z}$ such that

$$\bar{x}[n] = 0 \qquad \text{for } n < M \text{ or } n > M + N - 1.$$

Note that, although $\bar{\mathbf{x}}$ is an infinite sequence, knowledge of its N nonzero values (and of the start time M) completely determines the entire signal. This suggests another way to embed a finite-length signal into a sequence $\bar{\mathbf{x}}$, by simply extending the original data with zeros to the left and to the right:

$$\bar{x}[n] = \begin{cases} x[n] & \text{if } 0 \leq n < N - 1 \\ 0 & \text{otherwise} \end{cases} \qquad n \in \mathbb{Z}. \tag{1.10}$$

In general, we will use the bar notation $\bar{\mathbf{x}}$ for sequences defined as the finite support extension of a finite-length signal.

1.4 Elementary Discrete-Time Signals

The following discrete-time signals are basic building blocks that will appear repeatedly throughout the rest of the course. The signals are defined for all $n \in \mathbb{Z}$ and are therefore two-sided infinite sequences; the expressions can however be used to define finite-length signals as well, simply by restricting the range of the index to the interval $[0, N-1]$.

Impulse. The discrete-time impulse $\boldsymbol{\delta}$ (also known, from its symbol, as the discrete-time *delta*[7]) is the simplest discrete-time signal and it is defined as:

$$\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0; \end{cases} \tag{1.11}$$

the signal is shown in Figure 1.5. The delta signal has a single nonzero sample and is therefore the most "concentrated" signal in time; a shifted version of the delta, in which

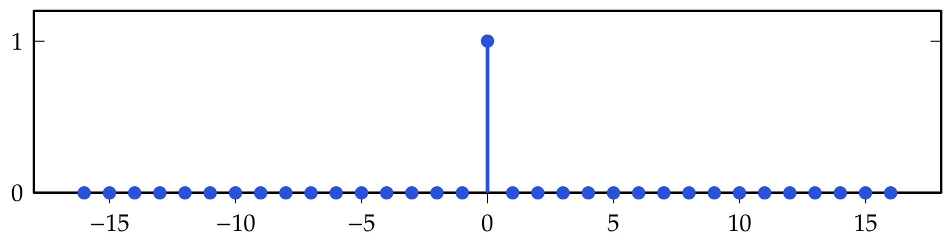

__Figure 1.5:__ The discrete-time delta.

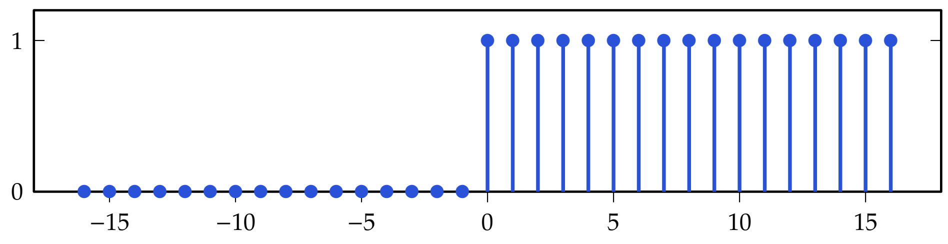

__Figure 1.6:__ The unit step signal.

the nonzero sample occurs for $n = k$, is indicated by $\boldsymbol{\delta}_k$. The discrete-time delta models a physical phenomenon with the shortest possible duration.

Unit Step. The discrete-time unit step $\mathbf{u}$ is shown in Figure 1.6 and is defined as:

$$u[n] = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0. \end{cases} \tag{1.12}$$

The unit step models a physical transition at time zero between an initial and a final state, much like the flipping of a switch.

Exponential Decay. A discrete-time exponential decay is defined by the formula:

$$x[n] = a^n u[n], \qquad a \in \mathbb{C},\ |a| < 1 \tag{1.13}$$

an example is shown in Figure 1.7. The exponential decay is an important signal since it models the response of a discrete-time first order recursive filter, as we will study in detail later on in the course. Exponential sequences are "well-behaved" only for values of a less than one in magnitude; sequences in which $|a| > 1$ grow unbounded and represent an unstable behavior.

Complex Exponential. A complex exponential signal is defined by the expression:

$$x[n] = e^{j(\omega n + \phi)} \tag{1.14}$$

[7]Not to be confused with the Dirac delta functional, that we will encounter later.

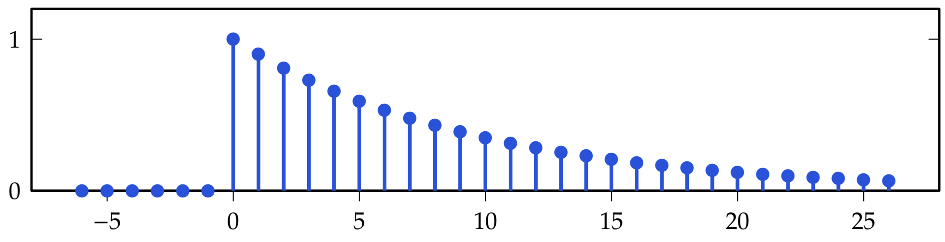

Figure 1.7: The exponential decay $\alpha^n u[n]$ with $\alpha = 0.9$.

where ω is the *frequency* and ϕ is the initial *phase*; both phase and frequency are dimensionless quantities expressed in radians. The complex exponential describes the prototypical oscillatory behavior in signal processing. The standard real-valued oscillations can always be obtained via Euler's formula, that is:

$$e^{j(\omega n+\phi)} = \cos(\omega_0 n + \phi) + j\sin(\omega_0 n + \phi) \tag{1.15}$$

as shown by the example in Figure 1.2.

1.5 Working with signals

1.5.1 Signal operators

From the algorithmic point of view, discrete-time signal processing is surprisingly straightforward, involving operators whose effect on sequences are completely intuitive.

Scaling. We can scale a signal $\mathbf{x}$ by a factor $\alpha \in \mathbb{C}$ simply by multiplying each sample by α:

$$(\alpha\mathbf{x})[n] = \alpha x[n] \tag{1.16}$$

If α is real, then the scaling represents a simple amplification (for $\alpha > 1$) or an attenuation (for $\alpha < 1$) of the signal. If α is complex, amplification and attenuation are compounded with a phase shift, whose meaning will be clearer in the next chapter.

Sum. The sum of two sequences is their term-by-term sum:

$$(\mathbf{x}+\mathbf{y})[n] = x[n] + y[n] \tag{1.17}$$

Sum and scaling are linear operators; informally, this means that they behave "intuitively":

$$\alpha(\mathbf{x}+\mathbf{y}) = \alpha\mathbf{x} + \alpha\mathbf{y}$$

Product. The product of two sequences is their term-by-term product

$$(\mathbf{x}\,\mathbf{y})\,[n] = x[n]\,y[n] \tag{1.18}$$

Time shift. The time shift operator moves a sequence forward by one sample:

$$(\mathcal{S}\,\mathbf{x})\,[n] = x[n+1]; \tag{1.19}$$

to advance a sequence by k we can apply the operator k times in a row and we will write

$$(\mathcal{S}\mathcal{S}\ldots\mathcal{S}\,\mathbf{x})\,[n] = \left(\mathcal{S}^k\,\mathbf{x}\right)[n] = x[n+k]. \tag{1.20}$$

The delay operator moves a sequence backwards; since this is the inverse of a forward shift, we will use the notation

$$\left(\mathcal{S}^{-1}\,\mathbf{x}\right)[n] = x[n-1] \tag{1.21}$$

for a one-step delay and

$$\left(\mathcal{S}^{-k}\,\mathbf{x}\right)[n] = x[n-k]$$

for arbitrary delay values. Note that this notation follows the standard exponentiation rules:

$$\mathcal{S}^k\,(\mathcal{S}^h\,\mathbf{x}) = \mathcal{S}^{k+h}\,\mathbf{x}$$

and, in particular,

$$\mathcal{S}^k\,(\mathcal{S}^{-k}\,\mathbf{x}) = \mathcal{S}^0\,\mathbf{x} = \mathbf{x}.$$

When dealing with finite-length signals, we need to adjust the definition of the time shift operator since the notation $x[n-k]$ is not defined for all values of n and k. There are two ways to do so, and they correspond to applying the shift operator to either a periodic extension or a finite-support extension of the original length-N signal $\mathbf{x}$:

- if we use a finite-support extension, the operator $\mathcal{S}^k$ performs a *logical shift* on the elements of the signal vector, that is, the elements are shifted k times to the left or to the right and zeros are inserted as a replacement, as illustrated here:

$$\begin{array}{rcl} \bar{\mathbf{x}} &=& [\ \ldots\ 0\ \ 0\ \ \overbrace{\underset{(n=0)}{x_0}\ \ x_1\ \ x_2\ \ldots\ x_{N-2}\ \ x_{N-1}}^{\mathbf{x}}\ \ 0\ \ 0\ \ldots\ ] \\ \mathcal{S}^{-1}\bar{\mathbf{x}} &=& [\ \ldots\ 0\ \ 0\ \ \underbrace{0\ \ x_0\ \ x_1\ \ldots\ x_{N-3}\ \ x_{N-2}}_{\mathcal{S}^{-1}\mathbf{x}}\ \ x_{N-1}\ \ 0\ \ldots\ ] \end{array}$$

- if we use a periodic extension, the operator $\mathcal{S}^k$ performs a *circular shift* on the elements of the signal vector, that is, the elements are shifted k times to the left or to the right and zeros are inserted as a replacement, as illustrated here:

$$\begin{array}{rcl}
\tilde{\mathbf{x}} & = & [\ \ldots\ x_{n-2}\ x_{N-1}\ \overbrace{\underset{(n=0)}{x_0}\ x_1\ x_2\ \ldots\ x_{N-2}\ x_{N-1}}^{\mathbf{x}}\ x_0\ x_1\ \ldots\ ] \\
\mathcal{S}^{-1}\tilde{\mathbf{x}} & = & [\ \ldots\ x_{N-3}\ x_{N-2}\ \underbrace{x_{N-1}\ x_0\ x_1\ \ldots\ x_{N-3}\ x_{N-2}}_{\mathcal{S}^{-1}\mathbf{x}}\ x_{N-1}\ x_0\ \ldots\ ]
\end{array}$$

As we said, the periodic extension is the natural extension of a finite-length signal so that, unless otherwise stated, in the rest of this course we will assume that all shifts applied to finite-length signals are circular shifts. Mathematically, we can formulate the circular shift via a modulo operation:

$$\mathbf{x} \in \mathbb{C}^N : \quad \left(\mathcal{S}^k\mathbf{x}\right)[n] = x[(n+k) \mod N]. \tag{1.22}$$

Note that, with signals represented as column vectors, time shift operator can be expressed as a matrix-vector multiplication; for example, for a signal of length four, a circular delay can be represented as:

$$\mathcal{S}^{-1}\mathbf{x} = \mathbf{D}\,\mathbf{x} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{x}. \tag{1.23}$$

In general, the $N \times N$ delay matrix $\mathbf{D}$ is a matrix with ones under the main diagonal, a one in the top right element and zeros everywhere else. Matrix notation can be extended to infinite-length signals as well by using infinite matrices, although we will not pursue this line of analysis further in this course.

Time reversal. The time reversal operator flips a sequence in time. For infinite-length signals we have:

$$(\mathcal{R}\mathbf{x})[n] = x[-n]. \tag{1.24}$$

If the length is finite, first notice that (1.24) can be rewritten as

$$(\mathcal{R}\mathbf{x})[n] = x[0-n] = (\mathcal{S}^{-n}\mathbf{x})[0];$$

with this, and using the definition in (1.20), the time reversal of a finite-length signal is

$$\begin{aligned}(\mathcal{R}\mathbf{x})[n] &= (\mathcal{S}^{-n}\mathbf{x})[0] \\ &= x[(0-n) \mod N] \\ &= x[(N-n) \mod N] \\ &= \begin{cases} x[0] & n = 0 \\ x[N-n] & 1 \le n < N \end{cases}\end{aligned}$$

The result can be easily understood if we flip around $n = 0$ the periodic extension of the original signal:

$$\begin{array}{rcl} \tilde{\mathbf{x}} &=& [\ \dots\ x_{n-2}\ \ x_{N-1}\ \ \overbrace{\underset{(n=0)}{x_0}\ \ x_1\ \ x_2\ \dots\ x_{N-2}\ \ x_{N-1}}^{\mathbf{x}}\ \ x_0\ \ x_1\ \dots\ ] \\ \mathcal{R}\tilde{\mathbf{x}} &=& [\ \dots\ x_2\ \ x_1\ \ \underbrace{x_0\ \ x_{N-1}\ \ x_{N-2}\ \dots\ x_2\ \ x_1}_{\mathcal{R}\mathbf{x}}\ \ x_0\ \ x_{N-1}\ \dots\ ] \end{array}$$

1.5.2 The Reproducing Formula

The reproducing formula is a simple application of the basic operations we just described; given a signal $\mathbf{x}$, and recalling that we denote a shifted delta as $\boldsymbol{\delta}_n = \mathcal{S}^{-n}\boldsymbol{\delta}$, we can always write

$$\mathbf{x} = \sum_n x[n]\,\boldsymbol{\delta}_n, \tag{1.25}$$

where the sum is taken over all valid values for n. The formula states that any signal can be expressed as a *linear combination of shifted impulses*. The weights are simply the signal values themselves and, while self-evident, this formula is important because it introduces the idea that a signal can be expressed as a linear combination of elementary building blocks. We will see in the next chapter that this is an instance of basis decomposition for a signal; while shifted impulses represent an intuitive canonical set of building blocks, by changing the basis set we can decompose a signal in many other ways, in order to highlight specific properties.

1.5.3 Energy and Power

We define the *energy* of a discrete-time signal as

$$E_x = \|\mathbf{x}\|_2^2 = \sum_n \left|x[n]\right|^2; \tag{1.26}$$

the squared-norm notation will be clearer after the next chapter but here it means that the sum is taken over all legal values for the index n, that is, over $\mathbb{Z}$ for infinite sequences and over the range $[0, N-1]$ for finite-length signals. This definition is consistent with the idea that, if the values of the sequence represent a time-varying voltage, the above sum would be proportional to the total energy (in joules) dissipated over a $1\,\Omega$ resistor. Obviously, the energy is finite only if the above sum converges; for finite-length signals this is always the case but for infinite sequences $\mathbf{x}$ must be *square-summable*. A signal with this property is sometimes referred to as a *finite- energy signal*. For a simple example of the converse, any periodic signal which is not identically zero is *not* square-summable.

We define the *power* of a infinite-length signal as the limit of the ratio of energy over time, taking the limit over the entire number of samples:

$$P_x = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{-N}^{N} \left|x[n]\right|^2 \tag{1.27}$$

Clearly, signals whose energy is finite have zero total power since their energy dilutes to zero over an infinite time duration. Conversely, unstable sequences such as diverging exponential sequences (i.e. e^{an} with $|a| > 1$) possess infinite power. Constant signals, on the other hand, whose energy is infinite, do have finite power; the same holds for periodic signals: although, formally, the limit in (1.27) is undefined for periodic sequences, we simply define their power to be their *average energy over a period*. Assuming that the period is N samples, we have

$$P_x = \frac{1}{N} \sum_{n=0}^{N-1} \left|x[n]\right|^2 \tag{1.28}$$