# CONVOLUTION

Convolution may be the single most important arithmetic operation in electrical engineering because any linear, time-invariant system generates an output signal by convolving the input with the impulse response of the system. Because of its significance, convolution is now a well-understood operation and is covered in any textbook containing the terms *signals* or *systems* in the title.

This article is intended to review some of the most important aspects of convolution. The fundamental relationship between linear, time-invariant systems alluded to in the first paragraph is reexamined and important properties of convolution, including several important transform properties, are presented and discussed.

Then this article discusses computational aspects. Even though the name *convolution* may be a slight misnomer (it appears to intimidate students because of its similarity to the word *convoluted*), it is a fact that continuous-time convolution often cannot be carried out in closed form. This article discusses in some detail procedures for approximating continuous-time convolution through discrete-time convolution.

Continuing with computational considerations, the article addresses the problem of computationally efficient, fast algorithms for convolution. This has been an active area of research until fairly recently, and the article provides insight into the principal approaches for devising fast algorithms.

The article concludes by examining several areas in which convolution or related operations play a prominent role, including error-correcting coding and statistical correlation. Finally, the article provides a brief introduction to the idea of abstract signal spaces.

## NOTATION AND BASIC DEFINITIONS

Convolution is an algebraic operation that requires two input signals and produces a third signal as the result. Convolution is defined for signals from both the continuous-time and the discrete-time domain. Continuous-time signals are simply functions of a free parameter $t$ that takes on a continuum of values. We will denote continuous-time signals by a lowercase letter and indicate the continuous-time parameter in parentheses [e.g., $x(t)$]. Similarly, discrete-time signals are functions of a free parameter $n$ that takes integer values only. We denote discrete-time signals by a lowercase letter followed by the discrete-time parameter enclosed in square brackets (e.g., $x[n]$). We will treat continuous-time and discrete-time convolution in parallel and repeatedly explore connections between the two.

### Continuous Time

For continuous-time signals the convolution of two signals $x(t)$ and $y(t)$ is denoted as $z(t) = x(t) * y(t)$ and defined as

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

where we assume the integral exists for all values of $t$.

To alleviate common confusion about this definition, several observations can be made. First, the result $z(t)$ is a function of $t$ and, thus, a continuous-time signal. Furthermore, the variable $\tau$ is simply an integration variable and, therefore, does not appear in the result. Most important, convolution requires integration of the product of two signals; one of these, $y(t-\tau)$, is time reversed with respect to the integration variable $\tau$ and its location depends on the variable $t$. We illustrate these considerations by means of an example.

Let the signals to be convolved be given by

$$x(t) = \exp\left(-\frac{t}{2}\right)u(t) \tag{2}$$

$$y(t) = \begin{cases} \frac{t}{5} & \text{for } 0 \le t \le 5, \\ 0 & \text{else} \end{cases} \tag{3}$$

where $u(t)$ denotes the unit-step function [i.e., $u(t) = 1$ if $t \ge 0$ and $u(t) = 0$ otherwise]. The signals $x(t)$ and $y(t)$ are shown in Fig. 1.

The definition of Eq. (1) prescribes that we must integrate over the product of $x(\tau)$ and $y(t-\tau)$. Figure 2 shows these signals for three different values of $t$ in the left-hand column. Considering these graphs from top to bottom, we see that $y(t-\tau)$ slides from left to right with increasing $t$. Furthermore, the orientation of $y(t-\tau)$ is flipped relative to the orientation of the signal $y(t)$ in Fig. 1. The signal $x(\tau)$ is repeated for reference.

The right-hand column in Fig. 2 shows the product of the two signals in the respective left-column plot. The result of the convolution is the integral of the product (i.e., the area indicated in the plots in the right column). Note that the area depends on the value of $t$ and, hence, the result of the convolution operation is a function of $t$.

Once the principles of convolution are understood, it is fairly easy to evaluate Eq. (1) analytically for this example.

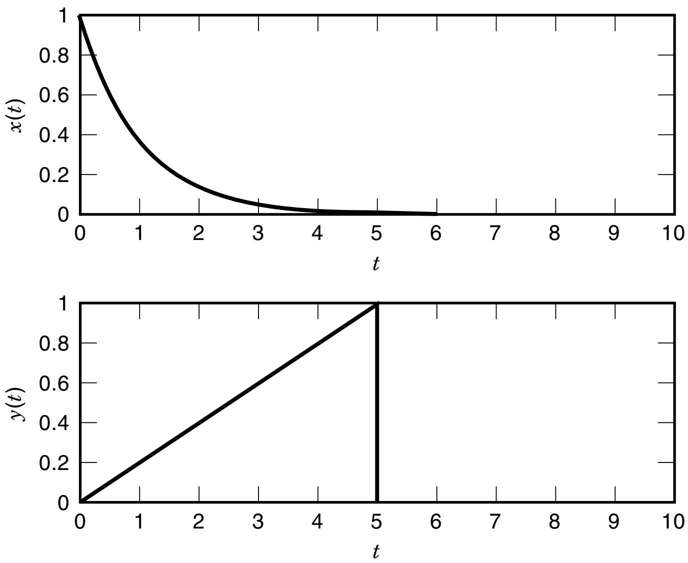


**Figure 1.** The signals $x(t)$ (top) and $y(t)$ (bottom) used to illustrate convolution.

First, note that $y(t - \tau)$ extends from $t - 5$ to $t$ (i.e., it is zero outside this range). Hence, we should consider three different cases as follows.

1. $t < 0$: In this case, the product of $x(\tau)$ and $y(t - \tau)$ is equal to zero and, thus, the result $z(t)$ equals zero for $t \leq 0$. This case is illustrated in the top row of Fig. 2.
2. $0 \leq t < 5$: Here, the nonzero part of $y(t - \tau)$ overlaps partially with the nonzero part of $x(\tau)$. Specifically, the product of $x(\tau)$ and $y(t - \tau)$ is nonzero for $\tau$ between zero and $t$. This is illustrated in the middle row in Fig. 2. Hence, we can write

$$\begin{aligned} z(t) &= \int_{-\infty}^{\infty} x(\tau) y(t-\tau)\, d\tau \\ &= \int_{0}^{t} \exp\left(-\frac{\tau}{2}\right) \frac{t-\tau}{5}\, d\tau \end{aligned} \tag{4}$$

This integral is easily evaluated by parts and yields

$$z(t) = \frac{2}{5}t - \frac{4}{5}\left(1 - \exp\left(-\frac{t}{2}\right)\right) \tag{5}$$

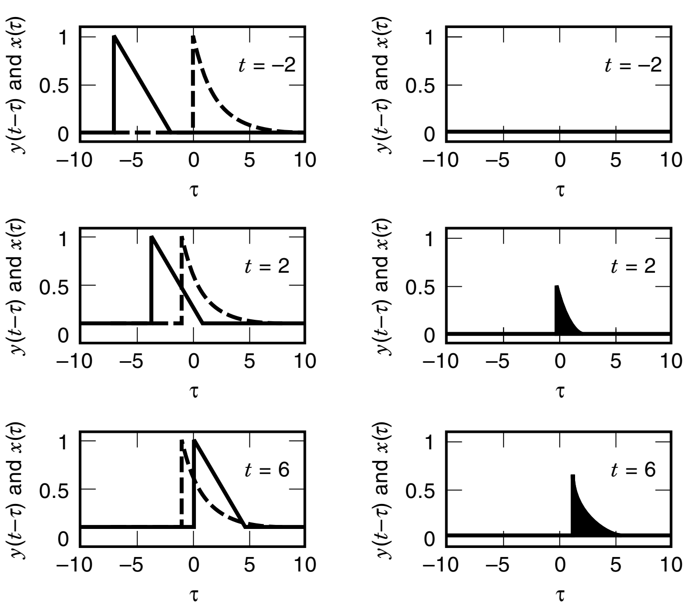


**Figure 2.** Illustration of convolution operation. The left-hand column shows $x(\tau)$ and $y(t - \tau)$ for three different values of $t$. The right-hand column indicates the intergral over the product of the two signals in the respective left-hand plots.

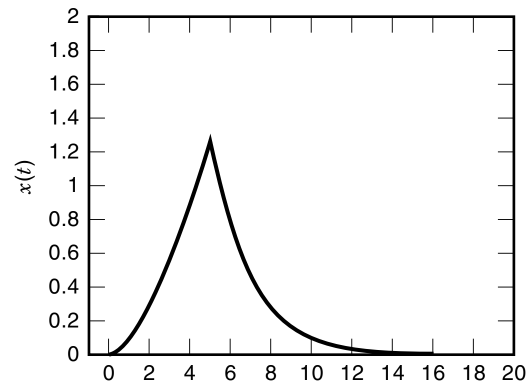


**Figure 3.** The result $z(t)$ of the convolution. Note that $z(t)$ retains features of both signals. For $t$ between zero and 5, $z(t)$ resembles the ramp signal $y(t)$. After $t = 5$, $z(t)$ is an exponentially decaying signal like $x(t)$.

3. $t \geq 5$: in this case, the nonzero part of $y(t - \tau)$ overlaps completely with the nonzero part of $x(\tau)$. Hence, the product of $x(\tau)$ and $y(t - \tau)$ is nonzero for $\tau$ between $t - 5$ and $t$. The last row in Fig. 2 provides an example for this case. To determine $z(t)$, we can write

$$\begin{aligned} z(t) &= \int_{-\infty}^{\infty} x(\tau) y(t-\tau)\, d\tau \\ &= \int_{t-5}^{t} \exp\left(-\frac{\tau}{2}\right) \frac{t-\tau}{5}\, d\tau \end{aligned} \tag{6}$$

Thus, the only difference to the previous case is the lower limit of integration. Again, the integral is easily evaluated and yields

$$z(t) = \frac{6}{5}\exp\left(-\frac{t-5}{2}\right) + \frac{4}{5}\exp\left(-\frac{t}{2}\right) \tag{7}$$

The resulting signal $z(t)$ is plotted in Fig. 3.

## Discrete Time

For discrete-time signals $x[n]$ and $y[n]$, convolution is denoted by $z[n] = x[n] * y[n]$ and defined as

$$z[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot y[n-k] \tag{8}$$

Notice the similarity between the definitions of Eqs. (1) and (8).

| | $n$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| | $x[n]$ | 2 | 4 | 6 | 4 | 2 | | | | |
| | $y[n]$ | 1 | −3 | 3 | −1 | | | | | |
| $k = 0$: | $x[0] \cdot y[n-0]$ | 2 | −6 | 6 | −2 | | | | | |
| $k = 1$: | $x[1] \cdot y[n-1]$ | | 4 | −12 | 12 | −4 | | | | |
| $k = 2$: | $x[2] \cdot y[n-2]$ | | | 6 | −18 | 18 | −6 | | | |
| $k = 3$: | $x[3] \cdot y[n-3]$ | | | | 2 | −6 | 6 | −2 | | |
| $k = 4$: | $x[4] \cdot y[n-4]$ | | | | | 4 | −12 | 12 | −4 | |
| | $z[n]$ | 2 | −2 | 0 | −6 | 12 | −12 | 10 | −4 | |

**Figure 4.** Convolution of finite length sequences.

A simple algorithm can be used to carry out the computations prescribed by Eq. (8) for finite length signals. Notice that $z[n]$ is computed by summing terms of the form $x[k] \cdot y[n-k]$. We can take advantage of this observation by organizing data in a tableau, as illustrated in Fig. 4. The example in Fig. 4 shows the convolution of $x[n] = \{2, 4, 6, 4, 2\}$ with $y[k] = \{1, -3, 3, -1\}$. We begin by writing out the signal $x[n]$ and $y[n]$. Then we use a process similar to "long multiplication" to form the output by summing shifted rows. The $k$th shifted row is produced by multiplying the $y[n]$ row by $x[k]$ and shifting the result $k$ positions to the right. The final answer is obtained by summing down the columns. It is easily seen from this procedure that the length of the resulting sequence $z[n]$ must be one less than the sum of the lengths of the inputs $x[n]$ and $y[n]$.

## LINEAR, TIME-INVARIANT SYSTEMS

The most frequent use of convolution arises in connection with the large and important class of linear, time-invariant systems. We will see that for any linear, time-invariant system the output signal is related to the input signal through a convolution operation. For simplicity, we will focus on discrete-time systems in this section and comment on the continuous-time case toward the end.

### Systems

To facilitate our discussion, let us briefly clarify what is meant by the term *system,* and more specifically *discrete-time system.* As indicated by the block diagram in Fig. 5, a discrete-time system accepts a discrete-time signal $x[n]$ as its input. This input is transformed by the system into the discrete-time output signal $y[n]$. We use the notation

$$x[n] \longmapsto y[n] \tag{9}$$

to symbolize the operation of the system. Linear, time-invariant systems form a subset of all systems. Before proceeding to demonstrate the main point of this section, we pause briefly to define the concepts of linearity and time invariance.

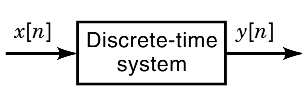


**Figure 5.** Discrete-time system.

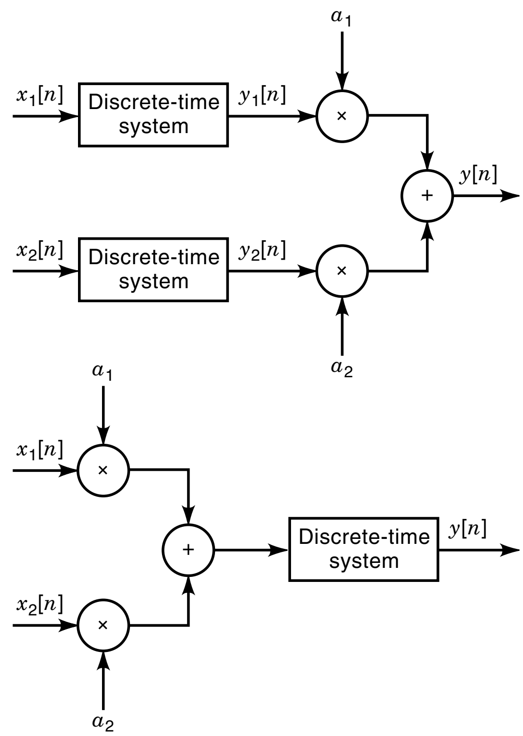


**Figure 6.** Linearity. For the discrete-time system to be linear, the outputs $y[n]$ of the two blocks must be equal to every choice of constants $a_1$ and $a_2$ and for all input signals $x_1[n]$ and $x_1[n]$.

**Linearity.** Linear system are characterized by the so-called principle of superposition. This principle says that if the input to the system is the sum of two scaled signals, then we can find the output by first computing the outputs due to each of the sequences and then add the two scaled outputs. More formally, linearity is defined as follows. Let $y_1[n]$ and $y_2[n]$ be the outputs of the system due to arbitrary inputs $x_1[n]$ and $x_2[n]$, respectively. Then the system is linear if, for arbitrary constants $a_1$ and $a_2$, the output of the system due to input $a_1x_1[n] + a_2x_2[n]$ equals $a_1y_1[n] + a_2y_2[n]$.

This property is illustrated by the block diagrams in Fig. 6. The figure also indicates that linearity implies that the addition and scaling of signals may be interchanged with the operation of the system.

**Time Invariance.** A system is time invariant if a delay of the input signal results in an equally delayed output signal. More specifically, let $y[n]$ be the output when $x[n]$ is the input. If the input is delayed by $n_0$ samples and becomes $x[n - n_0]$, then the resulting output must be $y[n - n_0]$ for the system to be time invariant.

Figure 7 illustrates the concept of time invariance. The diagram implies that for time-invariant systems the delay and the operation of the system can be interchanged.

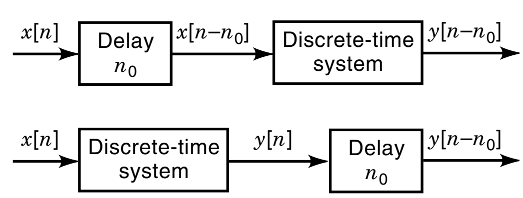


**Figure 7.** Time invariance. The outputs $y[n - n_0]$ must be equal for all delays $n_0$ for the system to be time invariant.

**Impulse Response.** The output of a system in response to an impulse input is called the impulse response. Mathematically, impulses are described by delta functions, and for discrete-time signals the delta function is defined as

$$\delta[n] = \begin{cases} 1 & \text{for } n = 0 \\ 0 & \text{for } n \neq 0 \end{cases} \tag{10}$$

It is customary to denote the impulse response as $h[n]$. Hence, we may write

$$\delta[n] \longmapsto h[n] \tag{11}$$

We have now accumulated enough definitions to proceed and demonstrate that there exists an intimate link between convolution and the operation of linear, time-invariant systems.

### Convolution and Linear, Time-Invariant Systems

We will show that the output $y[n]$ of any linear, time-invariant system in response to an input $x[n]$ is given by the convolution of $x[n]$ and the impulse response $h[n]$. This is an amazing result, as it implies that a linear, time-invariant system is completely described by its impulse response $h[n]$. Furthermore, even though linear, time-invariant systems form a very large and rich class of systems with numerous applications wherever signals must be processed, the only operation performed by these systems is convolution.

To begin, recall that the output of a system in response to the input $\delta[n]$ is the impulse response $h[n]$. For time-invariant systems, the response to a delayed impulse $\delta[n-k]$ must be a correspondingly delayed impulse response $h[n-k]$. Furthermore, the relationship $\delta[n-k] \mapsto h[n-k]$ must hold for any (integer) value $k$ if the system is time invariant.

Additionally, if the system is linear, we may scale the input by an arbitrary constant and effect only an equal scale on the output signal. In particular, the following relationships are all true for linear and time-invariant systems:

$$\begin{gathered} \vdots \\ x[-1]\delta[n+1] \longmapsto x[-1]h[n+1] \\ x[0]\delta[n] \longmapsto x[0]h[n] \\ x[1]\delta[n-1] \longmapsto x[1]h[n-1] \\ \vdots \\ x[k]\delta[n-k] \longmapsto x[k]h[n-k] \\ \vdots \end{gathered} \tag{12}$$

Here $x[n]$ is an arbitrary signal.

Finally, because of linearity, we may sum up all the signals on the right-hand side and be assured that this sum is the output for an input signal that is equal to the sum of the signals on the left-hand side. This means that

$$\sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \longmapsto \sum_{k=-\infty}^{\infty} x[k]h[n-k] = x[n] * h[n] \tag{13}$$

Thus, the output signal is equal to the convolution of $x[n]$ and $h[n]$.

The left-hand side requires a little more thought. For a given $k$, $x[k]\ \delta[n-k]$ is a signal with a single nonzero sample at $n = k$. Hence, the sum of all such signals is itself a signal and the samples are equal to $x[n]$. Thus, we conclude that

$$x[n] = x[n] * \delta[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \tag{14}$$

We will revisit this fact later in this article.

The preceding discussion can be summarized by the relationship

$$x[n] \longmapsto x[n] * h[n] \tag{15}$$

In words, the output of a linear, time-invariant system with impulse response $h[n]$ and input $x[n]$ is given by $x[n] * h[n]$. Recall that we have only invoked linearity and time invariance to derive this relationship. Hence, this fundamental result is true for any linear, time-invariant system.

### Continuous-Time Systems

The entire preceding discussion is valid for continuous-time systems, too. In particular, every linear, time-invariant system is completely characterized by its impulse response $h(t)$, and the output of the system in response to an input $x(t)$ is given by $y(t) = x(t) * h(t)$. A proof of this relationship is a little more cumbersome than in the discrete time case, mainly because the continuous-time impulse $\delta(t)$ is more cumbersome to manipulate than its discrete-time counterpart. We will discuss $\delta(t)$ later.

## FUNDAMENTAL PROPERTIES

The convolution operation possesses several useful properties. In many cases these properties can be exploited to simplify the manipulation of expressions involving convolution. We will rely on many of the properties presented here in the subsequent exposition.

### Symmetry

The order in which convolution is performed does not affect the final result [i.e., $x(t) * y(t)$ equals $y(t) * x(t)$]. This fact is easily shown by substituting $\sigma$ for $t - \tau$ in Eq. (1). Then we obtain

$$z(t) = \int_{-\infty}^{\infty} x(t-\sigma)y(\sigma)\,d\sigma \tag{16}$$

which obviously equals $y(t) * x(t)$. The corresponding relationship for discrete-time signals can be established in the same manner.

### Convolving with Delta Functions

The delta function is of fundamental importance in the analysis of signals and systems. The continuous-time delta function is defined implicitly through the relationship

$$\int_{-\infty}^{\infty} x(t)\delta(t-T)\,dt = x(T) \tag{17}$$

where the $x(t)$ is an arbitrary signal that is continuous at $t = T$. From this definition it follows immediately that

$$x(t) * \delta(t - t_0) = \int_{-\infty}^{\infty} x(\tau)\delta(t - t_0 - \tau)\, d\tau = x(t - t_0) \quad (18)$$

Hence, convolving a signal with a time-delayed delta function is equivalent to delaying the signal. The induced delay of the signal is equal to the delay $t_0$ of the delta function.

Analogous to the continuous-time case, when an arbitrary signal $x[n]$ is convolved with a delayed delta function $\delta[n - n_0]$, the result is a delayed signal $x[n - n_0]$. We have already seen this fact in Eq. (14) for the case $n_0 = 0$.

### Convolving with the Unit-Step Function

An ideal integrator computes the "running" integral over an input signal $x(t)$. That is, the output $y(t)$ of the ideal integrator is given by

$$y(t) = \int_{-\infty}^{t} x(\tau)\, d\tau \quad (19)$$

With the unit-step function $u(t)$, we may rewrite this equality as

$$y(t) = x(t) * u(t) = \int_{-\infty}^{\infty} x(\tau)u(t - \tau)\, d\tau \quad (20)$$

The equality between the two expressions follows from the fact that $u(t - \tau)$ equals one for $\tau$ between $-\infty$ and $t$ and $u(t - \tau)$ is zero for $\tau > t$.

The corresponding relationship for discrete-time signals is

$$y[n] = \sum_{k=-\infty}^{n} x[k] = \sum_{k=-\infty}^{\infty} x[k] \cdot u[n - k] = x[n] * u[n] \quad (21)$$

where $u[n]$ is equal to one for $n \geq 0$ and zero otherwise.

### Transform Relationships

For both continuous- and discrete-time signals there exist transforms for computing the frequency domain description of signals. While these transforms may be of independent interest in the analysis of signals, they also exhibit a very important relationship to convolution.

**Laplace and Fourier Transform.** The Laplace transform of a signal $x(t)$ is denoted by $\mathscr{L}\{x(t)\}$ or $X(s)$ and is defined as

$$X(s) = \mathscr{L}\{x(t)\} = \int_{-\infty}^{\infty} x(t)e^{-st}\, dt \quad (22)$$

where $s$ is complex valued and can be written as $s = \sigma + j\omega$. We will assume throughout this section that signals are such that their region of convergence for the Laplace transform includes the imaginary axis (i.e., the preceding integral converges for $\Re\{s\} = \sigma = 0$). Hence, we obtain the Fourier transform, $\mathscr{F}\{x(t)\}$ or $X(f)$, of $x(t)$ by evaluating the Laplace transform for $s = j2\pi f$.

The Laplace transform can be interpreted as the complex-valued magnitude of the response by a linear, time-invariant system with impulse response $h(t)$ to an input $x(t) = \exp(st)$. Then the output is given by

$$\begin{aligned} y(t) &= \int_{-\infty}^{\infty} h(\tau)x(t - \tau)\, d\tau \\ &= \int_{-\infty}^{\infty} h(\tau)e^{s(t-\tau)}\, d\tau \\ &= e^{st}\int_{-\infty}^{\infty} h(\tau)e^{-s\tau}\, d\tau \\ &= e^{st}\, H(s) \end{aligned} \quad (23)$$

where $H(s)$ denotes the Laplace transform of $h(t)$. $H(s)$ is commonly called the transfer function of the system. Notice, in particular, that the output $y(t)$ is an exponential signal with the same exponent as the input; the only difference between input and output is the complex-valued multiplicative constant $H(s)$. This observation is often summarized by the statement that (complex) exponential signals are eigenfunctions of linear, time-invariant systems.

The Laplace transform of the convolution of signals $x(t)$ and $y(t)$ can be written as

$$\begin{aligned} \mathscr{L}\{x(t) * y(t)\} &= \mathscr{L}\left\{\int_{-\infty}^{\infty} x(\tau)y(t - \tau)\, d\tau\right\} \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(\tau)y(t - \tau)e^{-st}\, d\tau\, dt \end{aligned} \quad (24)$$

Substituting $\sigma = t - \tau$ and $d\sigma = d\tau$ yields

$$\begin{aligned} \mathscr{L}\{x(t) * y(t)\} &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(\tau)y(\sigma)e^{-s(\tau+\sigma)}\, d\tau\, d\sigma \\ &= \int_{-\infty}^{\infty} x(\tau)e^{-s\tau}\, d\tau \cdot \int_{-\infty}^{\infty} y(\sigma)e^{-s\sigma}\, d\sigma \\ &= X(s) \cdot Y(s) \end{aligned} \quad (25)$$

Hence, we have the very important relationship that the Laplace transform of the convolution of two signals, $x(t) * y(t)$, is the product of the respective Laplace transforms, $X(s) \cdot Y(s)$. Clearly, this property also holds for Fourier transforms. This property may be used to simplify the computation of the convolution of two signals. One would first compute the Laplace (or Fourier) transform of the signals to be convolved, then multiply the two transforms, and finally compute the inverse transform of the product to obtain the final result. This procedure is often simpler than direct evaluation of the convolution integral of Eq. (1) when the signals to be convolved have simple transforms (e.g., when the signals are exponentials, including complex exponentials and sinusoids).

Finally, let $x(t)$ be a periodic signal of period $T$. Then $x(t)$ can be represented by a Fourier series

$$x(t) = \sum_{k=-\infty}^{\infty} x_k \exp(j2\pi kt/T) \quad (26)$$

where the Fourier series coefficients $x_k$ are given by

$$x_k = \frac{1}{T}\int_{0}^{T} x(t)\exp(-j2\pi kt/T)\, dt \quad (27)$$

A periodic signal is said to have a discrete spectrum.

If $x(t)$ is convolved with an aperiodic signal $y(t)$, then it is easily shown that the signal $z(t) = x(t) * y(t)$ is periodic and has a Fourier series representation

$$z(t) = \sum_{k=-\infty}^{\infty} z_k \exp(j2\pi kt/T) \tag{28}$$

with Fourier series coefficients $z_k$ equal to the product $x_k \cdot Y(k/T)$, where $Y(f)$ is the Fourier transform of of $y(t)$.

When two periodic signals are convolved, the convolution integral generally does not converge unless the spectra of the two signals do not overlap, in which case the convolution equals zero.

**$z$-Transform and Discrete-Time Fourier Transform.** For discrete-time signals, the $z$-transform plays a role equivalent to the Laplace transform for continuous-time signals. The $z$-transform $\mathscr{Z}\{x[n]\}$ or $X(z)$ of a discrete-time signal $x[n]$ is defined as

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n} \tag{29}$$

The variable $z$ is complex valued, $z = A \cdot e^{j\omega}$. Analogous to our assumption for the Laplace transform, we will assume throughout that signals are such that their region of convergence includes the unit circle (i.e., the preceding sum converges for $|z| = A = 1$). Then the (discrete-time) Fourier transform $X(f)$ can be found by evaluating the $z$-transform for $z = \exp(j2\pi f)$. Notice that the discrete-time Fourier transform is periodic in $f$ (with period 1); the continuous-time Fourier transform, in contrast, is not periodic.

Additionally, just as complex exponential signals are eigenfunctions of continuous-time, linear, time-invariant systems, signals of the form $x[n] = z^n$ are eigenfunctions of discrete-time, linear, time-invariant systems. Hence, if $x[n] = z^n$ is the input, then $y[n] = H(z)z^n$ is the output from a linear, time-invariant system with impulse response $h[n]$ and corresponding $z$-transform $H(z)$.

The $z$-transform of the convolution of sequences $x[n]$ and $y[n]$ is given by

$$\begin{aligned}\mathscr{Z}\{x[n] * y[n]\} &= \mathscr{Z}\left\{\sum_{k=-\infty}^{\infty} x[k] * y[n-k]\right\} \\ &= \sum_{n=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} x[k]y[n-k]z^{-n}\end{aligned} \tag{30}$$

By substituting $l = n - k$ and thus $n = l + k$, we obtain

$$\begin{aligned}\mathscr{Z}\{x[n] * y[n]\} &= \sum_{l=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} x[k]y[l]z^{-l-k} \\ &= \sum_{k=-\infty}^{\infty} x[k]z^{-k} \cdot \sum_{l=-\infty}^{\infty} y[k]z^{-l} \\ &= X(z) \cdot Y(z)\end{aligned} \tag{31}$$

Therefore, the $z$-transform of the convolution of two signals $x[n]$ and $y[n]$ equals the product of the $z$-transforms $X(z)$ and $Y(z)$ of the signals. Again, the same property also holds for Fourier transforms.

The discrete-time equivalent of the Fourier series is the discrete Fourier transform (DFT). Like the Fourier series, the DFT provides a signal representation using discrete, harmonically related frequencies. Both the Fourier series and the DFT representations result in periodic time functions or signals. For a discrete-time signal of length (or period) $N$ samples, the coefficients of the DFT are given by

$$X_k = \sum_{n=0}^{N-1} x[n]\exp(-j2\pi kn/N) \tag{32}$$

The signal $x[n]$ can be represented as

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X_k \exp(j2\pi kn/N) \tag{33}$$

When a periodic, discrete-time signal $x[n]$ with period $N$ and with DFT coefficients $X_k$ is convolved with a nonperiodic signal $y[n]$, the result is a periodic signal $z[n]$ of period $N$. Furthermore, the DFT coefficients of the result $z[n]$ are given by $X_k \cdot Y(k/N)$, where $Y(f)$ is the Fourier transform of $y[n]$.

**Circular Convolution.** An interesting problem arises when we ask ourselves which signal $z[n]$ has DFT coefficients $Z_k = X_k \cdot Y_k$, for $k = 0, 1, \ldots, N - 1$. First, because all three signals have DFTs of length $N$, they are implicitly assumed to be periodic with period $N$. Further, $z[n]$ can be written as

$$z[n] = \frac{1}{N}\sum_{k=0}^{N-1} Z_k \exp(j2\pi kn/N) = \frac{1}{N}\sum_{k=0}^{N-1} X_k \cdot Y_k \exp(j2\pi kn/N) \tag{34}$$

We can replace $X_k$ using the definition for the DFT and obtain

$$z[n] = \frac{1}{N}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} x[l]\exp(-j2\pi kl/N) \cdot Y_k \exp(j2\pi kn/N) \tag{35}$$

Reversing the order of summation, $z[n]$ can be expressed as

$$z[n] = \sum_{l=0}^{N-1} x[l]\frac{1}{N}\sum_{k=0}^{N-1} Y_k \exp(j2\pi k(n-l)/N) \tag{36}$$

The second summation is easily recognized to be equal to $y[\langle n - l\rangle]$, where $\langle n - l\rangle$ denotes the residue of $n - l$ modulo $N$ (i.e., the remainder of $n - l$ after division by $N$). The modulus of $n - l$ arises because of the periodicity of the complex exponential, specifically because $\exp(j2\pi k(n - l)/N)$ and $\exp(j2\pi k(\langle n - l\rangle)/N)$ are equal. Hence, $z[n]$ can be written as

$$z[n] = \sum_{k=0}^{N-1} x[k] \cdot y[\langle n-k\rangle] = x[n] \circledast y[n] \tag{37}$$

This operation is similar to convolution as defined in Eq. (8) and referred to as circular convolution. The subtle, yet important, difference from regular, or linear, convolution is the occurrence of the modulus in the index of the signal $y[n]$. An immediate consequence of this difference is the fact that the circular convolution of two length $N$ signals is itself of length $N$. The linear convolution of two signals of length $N$, however, yields a signal of length $2N - 1$.

Incidentally, a similar relationship exists in continuous time. Let $x(t)$ and $y(t)$ be periodic signals with period $T$ and Fourier series coefficients $X_k$ and $Y_k$, respectively. Then the signal

$$z(t) = \frac{1}{T}\int_0^T x(\tau)y(\langle t-\tau\rangle)\,d\tau \tag{38}$$

is periodic with period $T$ and has Fourier series coefficients $Z_k = X_k \cdot Y_k$. This property can be demonstrated in a manner analogous to that used for discrete-time signals.

We will investigate the relationship between linear and circular convolution later. We will demonstrate that circular convolution plays a crucial in the design of computationally efficient convolution algorithms.

## NUMERICAL CONVOLUTION

The continuous-time convolution integral is often not computable in closed form. Hence, numerical evaluation of the continuous-time convolution integral is of significant interest. When we are exploring means to compute the integral in Eq. (1) numerically, we will discover that the discrete-time convolution of sampled signals plays a key role. Furthermore, by employing ideal sampling arguments, we develop an understanding for the accuracy of numerical approximations to the convolution integral.

### Riemann Approximation

Let us begin by considering a straightforward approximation to continuous-time convolution based on the Riemann approximation to the integral. First, we approximate $z(t)$ by a stairstep function such that

$$z(t) \approx z(nT) \quad \text{for } nT \le \tau < (n+1)T \tag{39}$$

where $T$ is a positive constant. Consequently, the convolution integral needs to be evaluated only at discrete times $t = nT$, and for these times we have

$$z(nT) = \int_{-\infty}^{\infty} x(\tau)y(nT-\tau)\,d\tau \tag{40}$$

Next, we use the Riemann approximation to an integral as follows. The range of integration is broken up into adjacent, non-overlapping intervals of width $T$. On each interval, we approximate $x(\tau)$ and $y(nT - \tau)$ by

$$\begin{aligned} x(\tau) &\approx x(kT) \text{ for } kT \le \tau < (k+1)T \\ y(nT-\tau) &\approx y((n-k)T) \text{ for } iT \le \tau < (k+1)T \end{aligned} \tag{41}$$

If $T$ is sufficiently small, this approximation will be very accurate. In the limit as $T$ approaches zero, the exact solution $z(nT)$ is obtained. We will discuss the choice of $T$ in more detail later.

The Riemann approximation to the convolution integral is

$$z(nT) \approx \sum_{k=-\infty}^{\infty} \int_{kT}^{(k+1)T} x(kT)y((n-k)T)\,d\tau \tag{42}$$

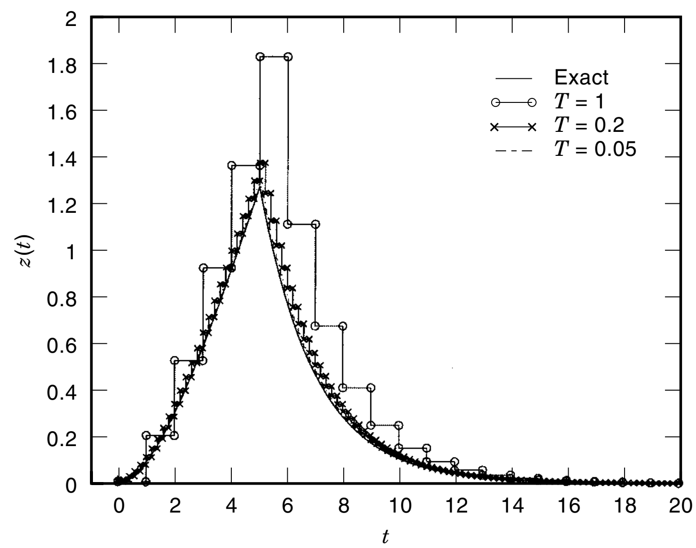


**Figure 8.** Numerical convolution.

Reversing the order of integration and summation and evaluating the (trivial) integral, we obtain

$$z(nT) \approx T \cdot \sum_{k=-\infty}^{\infty} x(kT)y((n-k)T) \tag{43}$$

Hence, apart from the constant $T$, this approximation is equal to the discrete-time convolution of sampled signals $x(t)$ and $y(t)$.

To illustrate, let us consider the two signals from the example given in the first section of this article. Figure 8 shows the exact result of the convolution together with approximations obtained by using $T = 1$, $T = 0.2$, and $T = 0.05$. Clearly, the accuracy of the approximation improves significantly with decreasing $T$. For $T = 0.05$, there is virtually no difference between the exact and the numerical solution.

How to select $T$ remains an open question. Our intuition tells us that $T$ must be small relative to the rate of change of the signals to be convolved. Then the error induced by approximating $x(\tau)$ and $y(nT - \tau)$ by the value of a nearby sample will be small. These notions can be made more precise by considering a system with ideal samplers.

### Numerical Convolution via Ideal Sampling

Consider the system in Fig. 9. The signals $x(t)$ and $y(t)$ are sampled before they are convolved. We will see that the result of this convolution depends directly on the discrete-time convolution of the samples $x(nT)$ and $y(nT)$. Finally, the signal $z_p(t)$ is filtered to yield the signal $\hat{z}(t)$. The objective of this analysis is to derive conditions on the sampling rate $T$ and the filter $h(t)$ such that $\hat{z}(t)$ is equal to $z(t)$.

**A System with Ideal Samplers.** The input signals $x(t)$ and $x(t)$ are first sampled using ideal samplers. Thus, the signals $x_p(t)$ and $y_p(t)$ are given by

$$\begin{aligned} x_p(t) &= \sum_{n=-\infty}^{\infty} x(nT)\delta(t-nT) \text{ and} \\ y_p(t) &= \sum_{n=-\infty}^{\infty} y(nT)\delta(t-nT) \end{aligned} \tag{44}$$

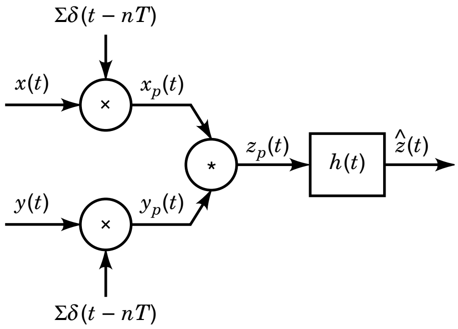


**Figure 9.** Convolution of ideally sampled signals. The input signals $x(t)$ and $y(t)$ are first sampled at rate $1/T$ and then convolved. The result $z_p(t)$ is then filtered [i.e., convolved with $h(t)$] to produce the approximation $\hat{z}(t)$ to $z(t)$.

Then $x_p(t)$ and $y_p(t)$ are convolved to produce the signal $z_p(t)$, which can be expressed as

$$\begin{aligned} z_p(t) = x_p(t) * y_p(t) &= \int_{-\infty}^{\infty} x_p(\tau) y_p(t-\tau)\, d\tau \\ &= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(nT)\delta(\tau - nT) \sum_{k=-\infty}^{\infty} y(kT)\delta(t-\tau-kT)\, d\tau \\ &= \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x(nT)y(kT) \int_{-\infty}^{\infty} \delta(\tau - nT)\delta(t-\tau-kT)\, d\tau \end{aligned} \tag{45}$$

Based on our considerations regarding the delta function, we recognize that the integral in the last equation is given by

$$\int_{-\infty}^{\infty} \delta(\tau - nT)\delta(t-\tau-kT)\, d\tau = \delta(t-(n+k)T) \tag{46}$$

Substituting this result back into our expression for $z_p(t)$, we obtain

$$z_p(t) = \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x(nT)y(kT)\delta(t-(n+k)T) \tag{47}$$

When we further substitute $l = n + k$, $z_p(t)$ becomes

$$z_p(t) = \sum_{l=-\infty}^{\infty} \left( \sum_{k=-\infty}^{\infty} x((k-l)T)y(kT) \right) \delta(t - lT) \tag{48}$$

The term in parentheses is simply the discrete-time convolution of the samples $x(nT)$ and $y(nT)$. Hence, $z_p(t)$ is equal to

$$z_p(t) = \sum_{n=-\infty}^{\infty} (x(nT) * y(nT))\delta(t-nT) \tag{49}$$

In other words, $z_p(t)$ is itself an ideally sampled signal with samples given by $x(nT) * y(nT)$. It is important to realize, however, that in general the discrete time signal $x(nT) * y(nT)$ is not equal to the signal $z(nT)$ obtained by sampling $z(t) = x(t) * y(t)$ unless the sampling period $T$ is chosen properly.

**Selection of Sampling Rate *T*.** To understand the impact of $T$, it is useful to consider the frequency domain representation of our signals. It is well known that the Fourier transform of an ideally sampled signal is obtained by periodic repetition and scaling of the Fourier transform of the original, nonsampled signal. Specifically, the Fourier transforms $X_p(f)$ and $Y_p(f)$ of the signals $x_p(t)$ and $y_p(t)$ are given by

$$X_p(f) = \frac{1}{T} \sum_{m=-\infty}^{\infty} X\left(f - \frac{m}{T}\right) \tag{50}$$

$$Y_p(f) = \frac{1}{T} \sum_{m=-\infty}^{\infty} Y\left(f - \frac{m}{T}\right) \tag{51}$$

where $X(f)$ and $Y(f)$ denote the Fourier transforms of $x(t)$ and $y(t)$, respectively. Since the Fourier transform of the convolution of $x_p(t)$ and $y_p(t)$ equals the product of $X_p(f)$ and $Y_p(f)$, it follows that the Fourier transform $Z_p(f)$ of $z_p(t)$ is

$$Z_p(f) = X_p(f) \cdot Y_p(f) = \frac{1}{T^2} \sum_{m=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} X\left(f - \frac{m}{T}\right) Y\left(f - \frac{k}{T}\right) \tag{52}$$

Recall that our objective is to obtain $\hat{z}(t)$ approximately equal to $z(t) = x(t) * y(t)$. On the other hand, we know that the Fourier transform $Z(f)$ of $z(t)$ equals $X(f) \cdot Y(f)$, and hence, we must seek to have $\hat{Z}(f)$ approximately equal to $X(f) \cdot Y(f)$. The simplest way to achieve this objective is to choose $T$ small enough that $X(f - m/T)Y(f - k/T) = 0$ whenever $m \neq k$. In other words, $T$ must be small enough that each replica $X(f - m/T)$ overlaps with exactly one replica $Y(f - k/T)$. Under this condition, the expression for $Z_p(f)$ simplifies to

$$Z_p(f) = \frac{1}{T^2} \sum_{m=-\infty}^{\infty} X\left(f - \frac{m}{T}\right) Y\left(f - \frac{m}{T}\right) \tag{53}$$

Notice that this is the Fourier transform of an ideally sampled signal with original spectrum $(1/T)\, X(f)Y(f)$. Expressed in the time domain, if $T$ is chosen to meet the preceding condition, $z_p(t)$ is the ideally sampled version of $z(t) = x(t) * y(t)$.

To summarize these observations, when $T$ is sufficiently small that $X(f - m/T)Y(f - k/T) = 0$ for all $m \neq k$, then

$$\begin{aligned} z_p(t) &= \sum_{n=-\infty}^{\infty} z(nT)\delta(t-nT) \\ &= \sum_{n=-\infty}^{\infty} (x(t) * y(t))|_{t=nT}\delta(t-nT) \\ &= \sum_{n=-\infty}^{\infty} (x(nT) * y(nT))\delta(t-nT) \end{aligned} \tag{54}$$

The convolution on the second line is in continuous time, while the one on the last line is in discrete time.

Most important, we may conclude that if $T$ is chosen properly then the samples $z(nT)$ of $z(t) = x(t) * y(t)$ are equal to $x(nT) * y(nT)$ [i.e., the discrete-time convolution of samples $x(nT)$ and $y(nT)$]. In other words, the order of convolution and sampling may be interchanged provided that the sampling period is sufficiently small.

How do we select the sampling period $T$ to be sufficiently small? Assume that both $x(t)$ and $y(t)$ are ideally band limited to $f_x$ and $f_y$, respectively. Then $X(f) = 0$ for $|f| > f_x$ and $Y(f) = 0$ for $|f| > f_y$. The first replica of $Y(f)$ [i.e., $Y(f - 1/T)$] extends from $1/T - f_y$ to $1/T + f_y$. For this replica not to overlap with the zeroth replica of $X(f)$ [i.e., $X(f)$ itself], $T$ must be such that

$$\frac{1}{T} - f_y > f_x \tag{55}$$

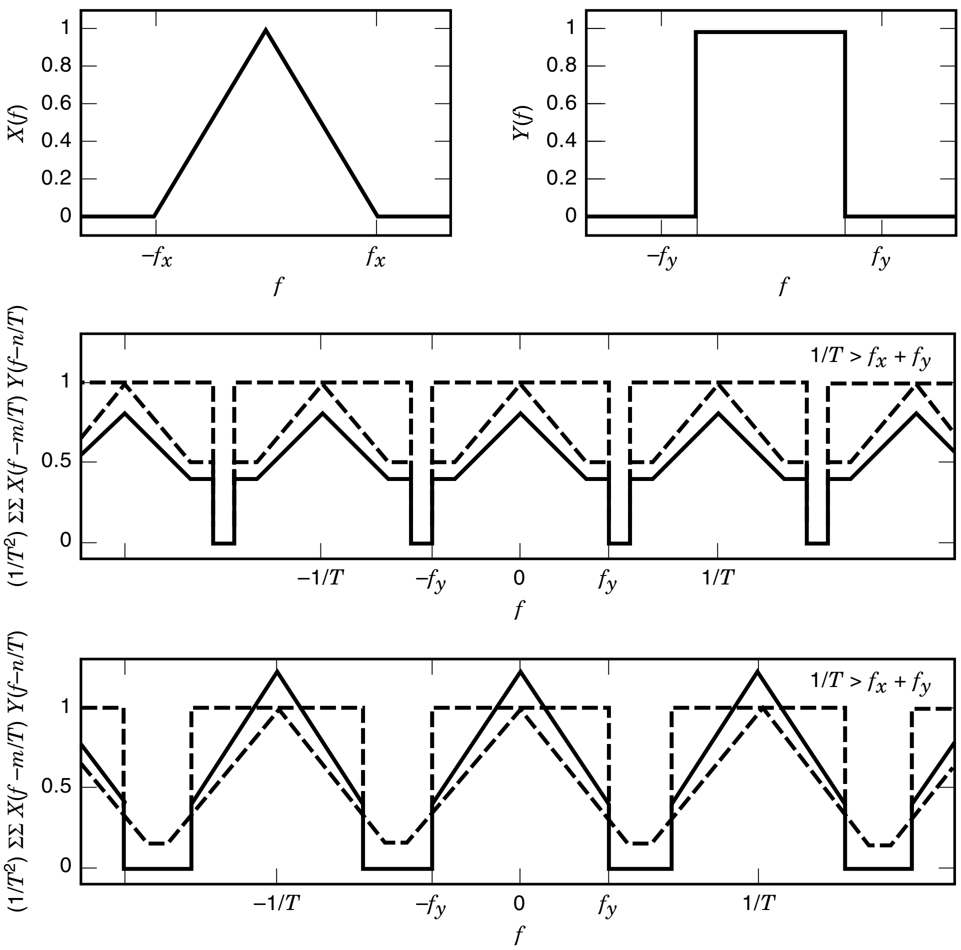


**Figure 10.** The influence of the sampling rate $T$ on numerical convolution. The spectra of the signals $x(t)$ and $y(t)$ to be convolved are shown on the top row. The spectra in the second and third rows are the result of first sampling and then convolving $x(t)$ and $y(t)$. On the second row, the sampling rate is insufficient and the resulting spectrum is not equal to the spectrum that results from ideally sampling $z(t) = x(t) * Y(t)$. On the bottom row, the sampling rate is sufficient. This is evident because the product of $X(f)$ and $Y(f)$ is visible between $-f_y$ and $f_y$.

Equivalently, $T$ must satisfy

$$T < \frac{1}{f_x + f_y} \tag{56}$$

Figure 10 illustrates these considerations. The first row shows the spectra $X(f)$ and $Y(f)$ of two strictly band-limited signals. The second and third rows contain plots that show the spectra resulting from first sampling signals $x(t)$ and $y(t)$ and then convolving the two sampled signals. An expression for the resulting expression is provided by Eq. (53). Both spectra are periodic with period $1/T$ and are thus spectra of ideally sampled signals.

However, the spectrum shown on the second row results from a violation of the condition of Eq. (56) on the sampling rate, while for the bottom plot this condition holds. Notice in particular that the segment between $-f_y$ and $f_y$ in the bottom plot is exactly equal to $X(f) \cdot Y(f)$, except for a scale factor. No such segment exists in the middle plot. Hence, the spectrum shown in the bottom plot corresponds to an ideally sampled signal with samples $z(nT)$; the middle plot does not.

Finally, even for the bottom plot, the sampling rate violates the Nyquist criterion ($T < 1/2f_x$) for the signal $x(t)$. This is evident, for example, in the region between $f_y$ and $1/T - f_y$, where aliasing is clearly evident.

**Interpolation.** We have demonstrated that the sampling rate $T$ should be selected such that $1/T$ exceeds the sum of the bandwidths of the signals to be convolved. Let us turn our attention to the choice of the filter labeled $h(t)$ in Fig. 9. The principal function of this filter is to interpolate between the sample values. It produces the signal $\hat{z}(t)$ by convolving $z_p(t)$ and $h(t)$. Since $\delta(t - lT) * h(t)$ equals $h(t - lT)$, we have immediately

$$\hat{z}(t) = \sum_{n=-\infty}^{\infty} (x(nT) * y(nT))h(t - nT) \tag{57}$$

In particular, for the choice

$$h(t) = \begin{cases} T & 0 \le t < T \\ 0 & \text{else} \end{cases} \tag{58}$$

the same stair-step approximation as in the previous section is obtained.

The function of the interpolation filter is easily expressed in the frequency domain. Because of the transform properties discussed previously, the Fourier transform $\hat{Z}(f)$ of $\hat{z}(t)$ is given by

$$\hat{Z}(f) = Z_p(f)H(f) \tag{59}$$

Assuming that our condition on the sapling rate is met, $\hat{Z}(f)$ equals

$$\hat{Z}(f) = \frac{1}{T^2} H(f) \sum_{m=-\infty}^{\infty} X\left(f - \frac{m}{T}\right) Y\left(f - \frac{m}{T}\right) \tag{60}$$

This equation demonstrates that for $\hat{Z}(f)$ to be similar to $Z(f)$, the interpolation filter must reject all replicas $X(f - m/T)Y(f - m/T)$ for $m \neq 0$.

Furthermore, it should introduce the appropriate gain and no distortion in the passband such that $(1/T^2)\ H(f)X(f)Y(f)$ equals $X(f)Y(f)$. Thus, the ideal choice for $H(f)$ is an ideal lowpass filter. However, the ideal lowpass filter has an infinite impulse response and is therefore not practical.

Frequently used interpolation filters in practice include the simple "hold filter" with $h(t)$, given in Eq. (58), or a linear interpolator, which can be realized by using a filter with impulse response

$$h(t) = \begin{cases} T \cdot (1-t) & 0 \le t < T \\ T \cdot (1+t) & -T \le t \le 0 \\ 0 & \text{else} \end{cases} \tag{61}$$

In particular, when $T$ is much smaller than specified by the preceding condition, these simpler interpolators provide excellent results.

Our discussion of numerical convolution can be summarized as follows: Continuous-time convolution can be approximated with arbitrary accuracy through discrete-time convolution of sampled versions of the signals to be convolved as long as the sampling rate is sufficiently large. Specifically, the sampling period $T$ must be chosen to exceed the sum of the bandwidths of the signals to be convolved. We have shown that under this condition, the discrete-time convolution produces a sequence of samples that is equal to samples of the original continuous-time convolution. Intermediate values may be produced via a suitable interpolation filter.

These considerations emphasize the practical importance of computationally efficient algorithms for discrete-time convolution. In the next section, we discuss convolution algorithms that rely heavily on ideas discussed in the context of transforms.

## FAST ALGORITHMS FOR CONVOLUTION

Filtering signals with linear, time-invariant systems is probably the most common form of signal processing. Hence, there is enormous interest in algorithms for computationally efficient (discrete-time) convolution. We will see that such algorithms take advantage of the transform relationships that were discussed previously. In particular, the development of fast algorithms for computing the discrete Fourier transform in the late 1960s has been seminal for the field of digital signal processing.

### Linear Convolution via Circular Convolution

The operation of linear, time-invariant filters is characterized by linear convolution. However, computationally attractive transform relationships exist for circular convolution. Previously, we showed that the DFT of two circularly convolved signals equals the product of the signals' DFTs. Furthermore, fast algorithms exist to compute the DFT of a signal. These algorithms are commonly called fast Fourier transforms (FFT).

Thus, the principal idea for a fast circular convolution algorithm is to compute the DFT of the signals to be convolved, to multiply the two DFTs, and finally to compute the inverse DFT of this product. All three DFTs can be computed efficiently using a suitable FFT algorithm.

We seek to take advantage of this approach for linear convolution. Toward this objective, let us take a closer look at the differences and similarities between linear and circular convolution.

**Convolution via Matrix Multiplication.** Both linear and circular convolution can be accomplished via matrix multiplication. This fact is of independent interest in many signal processing applications but will be used here to highlight the relationship between linear and circular convolution.

To fix ideas, consider the convolution of signals $x[n]$ and $y[n]$. Assume for the moment that both of these signals are of length $N$. The result of the linear convolution of $x[n]$ and $y[n]$ will be denoted $z_l[n]$ and the result of the circular convolution will be denoted $z_c[n]$. Recall that the length of $z_l[n]$ is $2N - 1$, while the length of $z_c[n]$ is $N$.

Both convolution operations can be expressed as the multiplication of a suitably chosen matrix and vector. Linear convolution can be written as

$$z_l[n] = x[n] * y[n] = X_l \cdot \boldsymbol{y} \tag{62}$$

where $X_l$ is a $(2N - 1) \times N$ matrix and $\boldsymbol{y}$ is the length $N$ vector with elements $y[n]$. The matrix $X_l$ is constructed with columns equal to shifted and zero-padded replicas of $x[n]$, specifically

$$X_l = \begin{pmatrix} x[0] & 0 & 0 & \dots & 0 \\ x[1] & x[0] & 0 & \dots & 0 \\ x[2] & x[1] & x[0] & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & x[N-1] & x[N-2] \\ 0 & \dots & 0 & 0 & x[N-1] \end{pmatrix} \tag{63}$$

The equivalence between convolution and multiplication of $X_l$ and $\boldsymbol{y}$ is easily verified. When the length of $y[n]$ is equal to $L$, $X_l$ is an $(N + L - 1) \times L$ matrix constructed as previously and $\boldsymbol{y}$ is a length $L$ vector.

Circular convolution can be written as

$$z_c[n] = x[n] * y[n] = X_c \cdot \boldsymbol{y} \tag{64}$$

where $X_c$ is a $N \times N$ matrix and $\boldsymbol{y}$ is as before. In contrast to $X_l$, the construction of $X_c$ does not involve zero padding. Instead, columns (and rows) are constructed from circular shifts of $x[n]$, specifically

$$X_c = \begin{pmatrix} x[0] & x[N-1] & x[N-2] & \dots & x[1] \\ x[1] & x[0] & x[N-1] & \dots & x[2] \\ x[2] & x[1] & x[0] & \dots & x[3] \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x[N-1] & \dots & x[2] & x[1] & x[0] \end{pmatrix} \tag{65}$$

This form of matrix is called a circulant matrix, a special form of Toeplitz matrix (1).

Comparison of the two matrices shows that we can transform linear convolution into an equivalent circular convolution. For that purpose, we must first pad both $x[n]$ and $y[n]$ with zeros to make them length $2N - 1$. We will refer to the zero-padded signals as $x_p[n]$ and $y_p[n]$, respectively. The product of the $(2N-1) \times (2N-1)$ circulant matrix $X_p$ generated from $x_p[n]$ and the vector $\boldsymbol{y}_p$ with elements $y_p[n]$ is equivalent to linear convolution of $x[n]$ and $y[n]$. In other words, $X_p \cdot y_p = X_e \cdot y_e$. This is evident because the first $N$ columns of this circulant matrix are equal to $X_l$ and the remaining $N - 1$ columns are multiplied by the appended zeros in $\boldsymbol{y}_p$.

Hence, linear convolution is equivalent to circular convolution of zero-padded sequences if the length of both padded sequences is equal to the length of the result of the linear convolution, in our case $2N - 1$. Actually, it may be advantageous to append even more zeros to $x[n]$ and $y[n]$ to yield sequence lengths for which particularly good FFT algorithms exist. In this case, excess zeros can simply be removed from the result.

We can summarize our observation as

$$z[n] = x[n] * y[n] = x_p[n] \circledast y_p[n] \tag{66}$$

If the length of $y[n]$ is equal to $L \neq N$, then $x_p[n]$ and $y_p[n]$ must be padded to length $N + L - 1$ (or greater) for this equality to hold.

### Fast Convolution via the FFT

We are now in position to exploit the fact that fast algorithms for computing the DFT of a discrete-time signal exist. Again, we focus on the case of two signals of equal length $N$. The fundamental idea of fast convolution algorithms is to zero pad the signals to be convolved to at least length $2N - 1$. Then the DFTs of the padded sequences are computed and multiplied. Finally, the inverse DFT of the product is computed and excess zeros are removed if necessary.

How does the computational efficiency of this algorithm compare to direct evaluation of the convolution sum? Let us consider both alternatives by first looking at the corresponding MATLAB code implementations.

The direct evaluation of the convolution sum can be programmed as

```
for n=1:2*N−1
  for m=max(1,n+1−N) :min(N,n)
    z(n) = z(n) + x(m)*y(n+1−m);
  end
end
```

A little thought reveals that the innermost statement is reached $N^2$ times and, hence, the direct computation of the convolution sum requires $N^2$ additions and multiplications.

A simple, FFT-based algorithm is given by

```
LEFT = 2^ceil(log2(2*N−1)); % choose FFT length
  as power of 2

xp = zeros(1,LFFT); % zero-padding
yp = zeros(1,LFFT);
xp(1:N) = x; % set first N samples to signal
yp(1:N) = y;

Xp = fft(xp); % forward FFTs
Yp = fft(yp);
Zp = Xp.*Yp; % multiplication of DFTs
zp = ifft(Zp); % inverse FFT

z = zp(1:2*N−1); % trim excess zeros
```

In the preceding program, the length $L_{\text{FFT}}$ of the FFT is chosen to be a power of 2 such that an efficient radix-2 (or split-radix) FFT algorithm may be employed. Then the number of additions and multiplications for each FFT is approximately proportional to $L_{\text{FFT}} \log_2 L_{\text{FFT}}$ (2). Hence, the entire algorithm requires approximately $3cL_{\text{FFT}} \log_2 L_{\text{FFT}} + L_{\text{FFT}}$ computations, where $c$ is a constant that depends on the specific FFT algorithm used.

To illustrate these ideas, we have conducted a simple numerical experiment using MATLAB. We generated signals varying in length between 10 and 10,000 and convolved these signals using three different algorithms: direct convolution, convolution via FFTs of length equal to the smallest power of 2 greater than $2N - 1$, and convolution via FFTs of length $2N - 1$. For each algorithm, the number of floating point operations, both additions and multiplications, was counted using the MATLAB built-in command `flops`.

The results of this experiment are shown in Fig. 11. The figure shows that the direct convolution requires nearly exactly $8N^2$ operations. For short sequences, $N \lesssim 50$, this algorithm is the most efficient. However, for longer sequences the algorithm using FFTs of length $2^m$ ($m$ integer) performs better. Furthermore, the advantage of the FFT based algorithm increases with the length of the sequence and reaches 2 orders of magnitude as $N = 10{,}000$. Notice that we must take advantage of the existence of a fast algorithm to realize a computational advantage through the use of transform-based convolution. If we rely on FFTs of length $2N - 1$, there are generally no highly efficient algorithms available and the

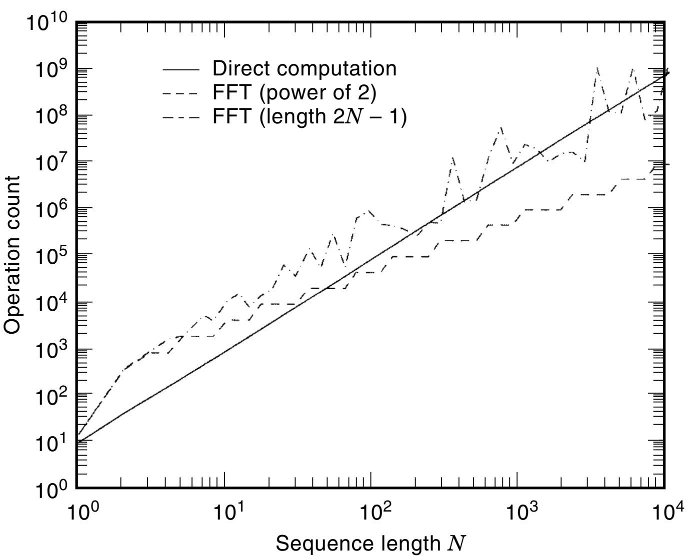


**Figure 11.** Computational complexity of convolution. The plot shows the number of floating point operations, additions, and multiplications, for three different convolutions algorithms as reported by the MATLAB command `flops`. All sequences are complex valued. The direct computation of the convolution sum requires nearly exactly $8N^2$ operations. For values of $N \gtrsim 50$ the number of operations for the (radix 2) FFT based algorithm is lower than for direct convolution. The third graph shows the number of operations if the length of the FFTs is not selected to be a power of 2. In this case, the transform-based algorithm is not efficient. The large variation in the operation count is related to MATLAB's FFT routine, which selects different FFT algorithms depending on the sequence length.

computational burden is often increased over direct computation of the convolution.

### Sequences of Different Length

It is often necessary to convolve sequences of very different length. The impulse response of a filter $h[n]$ is typically of length 50 to 100, but the input signal $x[n]$ may consist of thousands of samples. In this case, it is possible to segment the input data into shorter blocks, perform convolution on these blocks, and combine the intermediate results. In light of our preceding discussion, the block length $B$ of the input segments should be selected such that $B + L - 1$, where $L$ is the filter length, provides the opportunity to employ a good FFT algorithm. For example, $B + L - 1$ can be selected to equal a power of 2.

To illustrate the reassembly of the intermediate results, let us consider the convolution of a length 3 filter $h[n]$ with a length 6 input sequence. We use two segments of block length three, such that intermediate results are of length $3 + 3 - 1 = 5$. These must be combined to yield the final result $y[n]$ of length $6 + 3 - 1 = 8$. For our illustration, we use the matrix formulation of linear convolution as in Eq. (63):

$$\begin{pmatrix} y[0] \\ y[1] \\ y[2] \\ y[3] \\ y[4] \\ y[5] \\ y[6] \\ y[7] \end{pmatrix} = \left(\begin{array}{ccc|ccc} h[0] & 0 & 0 & & & \\ h[1] & h[0] & 0 & & & \\ h[2] & h[1] & h[0] & & & \\ 0 & h[2] & h[1] & h[0] & 0 & 0 \\ 0 & 0 & h[2] & h[1] & h[0] & 0 \\ & & & h[2] & h[1] & h[0] \\ & & & 0 & h[2] & h[1] \\ & & & 0 & 0 & h[2] \end{array}\right) \cdot \begin{pmatrix} x[0] \\ x[1] \\ x[2] \\ \hline x[3] \\ x[4] \\ x[5] \end{pmatrix} \tag{67}$$

In this example, two intermediate sequences are obtained by convolving $h[n]$ with the top and bottom half of $x[n]$, respectively. To assemble the final result requires that the two intermediate sequences are added such that they overlap by $L - 1 = 2$ samples. This method of convolving different length sequences is appropriately called overlap-add convolution.

More details on computational aspects of convolution are provided in the book by Burrus and Parks (3) or the recent book by Ersoy (4). An in-depth analysis and discussion of the state-of-the art in fast algorithms for DFT and convolution is presented in the tutorial article by Sorensen and Burrus (2).

## APPLICATIONS AND EXTENSIONS

We conclude this article by looking at several applications beyond filtering and signal processing in which convolution arises naturally. Additionally, we provide pointers to several interesting extensions of the material presented herein.

### Polynomial Multiplication

When two polynomials are multiplied, the result is another polynomial and the coefficients of the resulting polynomial are obtained from the original coefficients through convolution.

Let $p(x)$ and $q(x)$ be two polynomials with coefficients $p_i$ and $q_i$, respectively. When these polynomials are multiplied, we obtain the polynomial $r(x)$ as

$$\begin{aligned} r(x) &= p(x) \cdot q(x) \\ &= \sum_{k=0}^{N_p} p_k x^k \cdot \sum_{l=0}^{N_q} q_l x^l \\ &= \sum_{k=0}^{N_p} \sum_{l=0}^{N_q} p_k q_l x^{k+l} \end{aligned} \tag{68}$$

Substituting $k + l = n$, the last expression can be simplified to

$$r(x) = \sum_{n=0}^{N_p+N_q} \left( \sum_{k=0}^{\min(n,N_p)} p_k q_{n-k} \right) x^n \tag{69}$$

The term in parentheses is the convolution of the sequences of coefficients of $p(x)$ and $q(x)$.

Hence, the resulting polynomial is of degree equal to the sum of the original polynomials, and its coefficients are obtained by convolving the original coefficient sequences.

### Applications in Error-Correcting Coding

In all our discussions to this point, arithmetic operations were assumed to be based on real number arithmetic. Error-correcting coding relies on convolution with arithmetic over finite fields (e.g., binary arithmetic). Error-correcting coding is a field that has attracted considerable research efforts over the last 50 years, and we have to limit ourselves to simple examples here. Good introductions to the field and considerably more depth can be found in the classic book by Lin (5) or the more recent book by Wicker (6).

**Cyclic Codes.** Cyclic codes constitute an important class of practical error control codes and include such well-known representatives as Golay, BCH, and Reed-Solomon codes. An important reason for the continuing practical relevance of these codes is the fact that encoders and decoders can be implemented with simple, high-speed shift-register circuits. This is of great importance in high-speed communications applications.

Perhaps surprisingly, cyclic codes are based on ideas and concepts from mathematical algebra. The key to the structure of cyclic codes lies in the association of a code polynomial $c(x)$ with every code word $\boldsymbol{c} = (c_0, c_1, \ldots, c_{n-1})$. Skipping many important details, we only mention that code words (i.e., information sequences with error-correction capabilities) are obtained from unprotected message words $\boldsymbol{m} = (m_0, m_1, \ldots, m_{k-1})$ through polynomial multiplication.

Specifically, a polynomial $m(x) = m_0 + m_1x + m_2x^2 + \cdots + m_{k-1}x^{k-1}$ is constructed and then multiplied by a suitably chosen generator polynomial $g(x)$. The selection of $g(x)$ is crucial and discussed in detail in Chapter 5 of Ref. 6 or Chapter 4 of Ref. 5. Then every code polynomial can be expressed as $c(x) = g(x) \cdot m(x)$. Since $c(x)$ is obtained by polynomial multiplication, the coefficients of $c(x)$ and, hence, the elements of the code word $\boldsymbol{c}$ are obtained by convolution of the coefficients of $g(x)$ and $m(x)$. However, all arithmetic operations are defined over a finite algebraic field. For example, when binary arithmetic is used, the underlying field is referred to as the Galois field of size 2 and modulo 2 arithmetic is used.

To fix ideas, let us consider a well-known (7, 4) cyclic code with generator polynomial $g(x) = 1 + x + x^3$. To encode the message block $\boldsymbol{m} = (1110)$, we construct first the polynomial $m(x) = 1 + x + x^2$. Then the code polynomial $c(x)$ is computed as

$$\begin{aligned} c(x) = g(x) \cdot m(x) &= (1 + x + x^3) \cdot (1 + x + x^2) \\ &= 1 + (1+1)x + (1+1)x^2 \\ &\quad + (1+1)x^3 + x^4 + x^5 \\ &= 1 + x^4 + x^5 \end{aligned} \tag{70}$$

The last two lines are equal because in modulo 2 arithmetic $1 + 1 = 0$. The resulting code word is $\boldsymbol{c} = (1000110)$.

To verify that a code word has been transmitted without error, a decoder checks if the polynomial associated with a received word $\boldsymbol{r}$ is a valid code polynomial by verifying that it is divisible by $g(x)$.

While the operation of the encoder and decoder may appear awkward at first sight, they are implementable with very simple, high-speed digital hardware. Both encoder and decoder hardware can be implemented as feedback shift register circuits.

**Convolutional Coding.** As its name suggests, a convolutional encoder employs convolution to insert error-correction information into a sequence of information symbols. As in cyclic coding, all arithmetic operations are carried out over finite fields. While linear block codes, including cyclic codes, operate on message sequences of fixed length, convolutional codes can be used to encode message sequences that are not necessarily bounded in length.

Specifically, a convolutional encoder can be built around $K$ shift registers with $m_k$ memory elements, $k = 1, 2, \ldots, K$, into which the message sequence is fed. Let the message sequence be given by

$$\boldsymbol{x} = (x_0^1, x_0^2, \ldots, x_0^K, x_1^1, x_1^2, \ldots, x_1^K, \ldots, x_n^k, \ldots) \tag{71}$$

Then in symbol interval $n$ the information symbol $x_n^k$ is fed into the $k$th shift register. Also, at the beginning of symbol interval $n$, the $k$th shift register contains the information symbols $(x_{n-1}^k, x_{n-2}^k, \ldots, x_{n-m_k}^k)$. A rate $K/L$ convolutional encoder generates $L$ output sequences $\boldsymbol{y}^l$ by convolving (over a finite field) the information subsequences $\boldsymbol{x}^k = (x_0^k, x_1^k, \ldots, x_n^k, \ldots)$ with $KL$ generator sequences $\boldsymbol{g}^{k,l}$ of length $m_k + 1$. Hence, the $n$th symbol in the $l$th output sequence is given by

$$y_n^l = \sum_{k=1}^{K} \left( \sum_{m=0}^{m_k} x_{n-m}^k \cdot g_m^{k,l} \right) \tag{72}$$

where all arithmetic operations are performed over a finite field (e.g., in modulo 2 arithmetic).

To fix ideas, let us consider a rate $\frac{1}{2}$ convolutional code with generator sequences $\boldsymbol{g}^{1,1} = (1011)$ and $\boldsymbol{g}^{1,2} = (1101)$. All arithmetic operations are modulo 2. Then the input sequence $\boldsymbol{x}$ yields output sequences $\boldsymbol{y}^1$ and $\boldsymbol{y}^2$ with elements

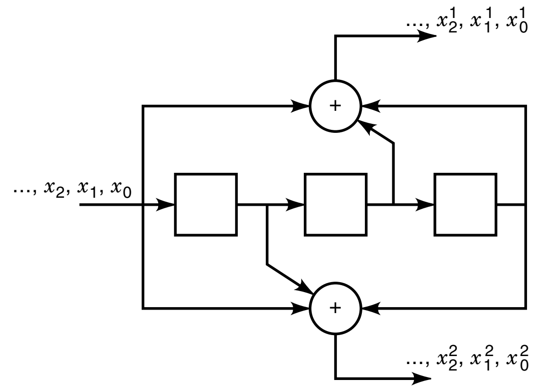


**Figure 12.** A rate-$\frac{1}{2}$ convolutional encoder.

$$y_n^1 = \sum_{m=0}^{m_1} x_{n-m} g_m^{1,1} = x_n + x_{n-1} + x_{n-3} \pmod 2 \tag{73}$$

and

$$y_n^2 = \sum_{m=0}^{m_2} x_{n-m} g_m^{1,2} = x_n + x_{n-2} + x_{n-3} \pmod 2 \tag{74}$$

The operation of this convolutional encoder can be summarized by the block diagram in Fig. 12.

Clearly, we have only scratched the surface on the topic of error-correcting codes. The inclined reader is referred to the Refs. 5 and 6 for further details.

## Convolution in Statistics

While filtering of random signals and the associated convolution operation are important operations in statistical signal processing processing, the convolution operation appears in other problems of statistics as well. Background information on the concepts discussed in this section are contained in Ref. 7.

**Sum of Independent Random Variables.** The probability density function of the sum of two independent random variables is related to the density function of the original random variables through convolution. To begin, let $X$ and $Y$ denote two independent, continuous random variables with probability density functions $f_X(x)$ and $f_Y(y)$. When we form a third random variable $Z$ as the sum of $X$ and $Y$, we can determine the probability density function of $Z$ as follows.

The distribution function $F_Z(z)$ of $Z$ is defined as

$$F_Z(z) = \Pr(Z \le z) = \Pr(X + Y \le z) \tag{75}$$

The last term can be expressed via the joint density function $f_{XY}(x, y)$ of $X$ and $Y$ as

$$F_Z(z) = \Pr(X + Y \le z) = \iint_{x+y \le z} f_{XY}(x, y)\, dx\, dy \tag{76}$$

For an independent random variable, $f_{XY}(x, y) = f_X(x)f_Y(y)$. Furthermore, the range of integration can be rewritten such that we obtain

$$F_Z(z) = \int_{-\infty}^{\infty} f_X(x) \int_{-\infty}^{z-x} f_Y(y)\, dy\, dx \tag{77}$$

Since we assumed continuous random variables, the density function $f_Z(z)$ is obtained by differentiation of $F_Z(z)$:

$$f_Z(z) = \frac{dF_Z(z)}{dz} = \int_{-\infty}^{\infty} f_X(x) f_Y(z-x)\, dx = f_X(z) * f_Y(z) \tag{78}$$

Hence, the sum of two independent random variables yields a density function that equals the convolution of the original densities. An analogous result can be derived for discrete random variables.

Finally, a transform relationship, very similar to those presented earlier, exists that captures the preceding result. For a random variable with density function $f(x)$, the moment generating function $M(j\nu)$ is defined as

$$M(j\nu) = \int_{\infty}^{\infty} f(x) e^{j\nu x}\, dx \tag{79}$$

Hence, the moment-generating function is essentially equal (except for the sign of the exponent) to the Fourier transform of the density function $f(x)$. If we denote the characteristic functions of our independent random variables $X$ and $Y$ by $M_X(j\nu)$ and $M_Y(j\nu)$, the characteristic function of $Z = X + Y$ is given by

$$M_Z(j\nu) = M_X(j\nu) M_Y(j\nu) \tag{80}$$

**Correlation.** The empirical autocorrelation of certain random processes is computed through an operation virtually identical and closely related to convolution. To be specific, let $X_t$ denote a real-valued, wide-sense stationary random process such that its autocorrelation function $R_X(\tau)$ is given by

$$R_X(\tau) = \mathbf{E}[X_t \cdot X_{t+\tau}] \tag{81}$$

where $\mathbf{E}[\,\cdot\,]$ denotes statistical expectation.

In practice, one is often faced with the problem of estimating the autocorrelation function from a single realization $x(t)$ of $X_t$ (observed for $0 \le t \le T$). If the random process is ergodic, we may estimate the statistical average in Eq. (81) via the time average

$$\hat{R}_X(\tau) = \frac{1}{T - |\tau|} \int_0^{T-|\tau|} x(t) x(t + |\tau|)\, dt \tag{82}$$

for $|\tau| < T$ In this expression, we have taken advantage of the symmetry property $R_X(\tau) = R_X(-\tau) = R_X(|\tau|)$.

It is easily shown that $\hat{R}_X(\tau)$ is an unbiased estimate of $R_X(\tau)$. However, the variance of $\hat{R}_X(\tau)$ becomes infinite as $\tau$ approaches $T$. To alleviate this problem, a weighting function, or window, may be used to ensure that the variance remains finite as $\tau$ approaches $T$. Further details can be found in Ref. 7, Chapter 13.

Notice that Eq. (82) bears a striking resemblance to the convolution integral of Eq. (1). In fact, it is easily verified that we can express $\hat{R}_X(\tau)$ as

$$\hat{R}_X(\tau) = w(\tau) \cdot x(\tau) * x(-\tau) \tag{83}$$

where $w(\tau) = (T - |\tau|)^{-1}$.

Thus, the empirical autocorrelation function is essentially equal to the convolution of a signal with a time-reversed version of itself. Consequently, our discussion on properties, numerical evaluation, and efficient computation of the convolution integral applies equally to the empirical autocorrelation function.

Analogous arguments can be made for discrete-time random processes, or random sequences, and for the cross-correlation function of two random processes or sequences.

## Signal Spaces

Modern signal processing and control theory rely extensively on the concept of linear spaces from the mathematical field of functional analysis. The results presented here are compiled mainly from Refs. 8–10. Also, we restrict ourselves to scalar signals; most of the referenced literature treats the more general case of vector signals.

Though most of our discussion is aimed at more abstract spaces of functions (signals), it may be useful for the reader to consider the space $\mathbb{R}^N$ of length $N$ real-valued vectors throughout our exposition for illustrative purposes.

**Linear Spaces, Norms, and Inner Products.** A linear space consists of a set $\mathscr{S}$ (of signals, functions, or vectors), a scalar field $\mathscr{F}$, usually the real or complex numbers, and rules that addition of elements of $\mathscr{S}$ as well as scalar multiplication of elements of $\mathscr{F}$ and $\mathscr{S}$ obey. More specifically, both the addition $x + y$ of two elements of the space $\mathscr{S}$ and the multiplication $\alpha x$, $\alpha \in \mathscr{F}$, satisfy the usual laws of commutativity and associativity. Also, inverse and neutral elements exist for addition over $\mathscr{S}$, and a neutral element for scalar multiplication is contained in $\mathscr{F}$. These properties of a linear space lend a well-behaved algebraic structure to $\mathscr{S}$.

By means of a norm on the elements of $\mathscr{S}$, we can provide a topological structure for our space $\mathscr{S}$. A linear space with a norm is called a normed, linear space. A norm is simply a real-valued functions defined for all elements of $S$. The norm of an element $x \in \mathscr{S}$ is denoted as $\|x\|$ and must satisfy the following conditions:

1. $\|x\| \ge 0$
2. $\|x + y\| \le \|x\| + \|y\|$ (triangle inequality)
3. $\|\alpha x\| = \alpha\|x\|$, $\alpha \in \mathscr{F}$
4. $\|x\| = 0$ if and only if $x = 0$

From an engineering perspective, the most important norm is defined for the space of finite-energy signals. For reasons that will become apparent shortly, this space is conventionally denoted $\mathscr{L}_2$. The norm of a signal $x(t)$ in $\mathscr{L}_2$ is defined by

$$\|x(t)\|^2 = \int_{-\infty}^{\infty} |x(t)|^2\, dt \tag{84}$$

Hence, $\|x(t)\|^2$ equals the energy of signal $x(t)$. It is easily verified that this norm meets all four of the preceding requirements. In general, norms are useful primarily for quantifying the difference between two elements $x$ and $y$ of a space $S$ through $\|x - y\|$.

Even more topological structure is induced if a space $\mathscr{S}$ also possesses an inner product. Fundamentally, an inner product introduces important geometrical concepts such as orthogonality. We may even say that an inner product space is more or less the generalization of Euclidean geometry to infinite dimensions. This leads directly to useful geometrical interpretations of many problems in signal processing and control.

The inner product is a function that associates with each pair $x$ and $y$ of elements of $\mathscr{S}$ a scalar. We denote the inner product of $x$ and $y$ as $(x, y)$. An inner product must satisfy the following properties:

1. $(x + y, z) = (x, z) + (y, z)$ (additivity)
2. $(\alpha x, y) = \alpha(x, y)$ (homogeneity, $\alpha \in \mathscr{F}$)
3. $(x, y) = (y, x)^*$ (symmetry, $*$ denotes the complex conjugate)
4. $(x, x) > 0$, unless $x = 0$

Two observations are useful. First, any inner product space is also normed, linear space because $\|x\|^2 = (x, x)$ satisfies all requirements for a norm. However, there are many norms that cannot be expressed through an inner product. Hence, inner product spaces form a subset of normed, linear spaces. Second, the following inequality is often extremely useful, particularly in optimization:

$$|(x, y)| \le \|x\| \|t\| \tag{85}$$

Furthermore, equality holds if and only if $x$ and $y$ are collinear. This result is known as the Schwarz inequality.

For the space $\mathscr{L}^2$ introduced previously, the inner product is given by

$$(x(t), y(t)) = \int_{-\infty}^{\infty} x^*(t) y(t)\, dt \tag{86}$$

The final concept we introduce is completeness. Completeness becomes important when we consider infinite sequences $x_n$ of elements of our abstract space $\mathscr{S}$. Specifically, a normed, linear space is complete when the limit of all sequences $x_n$ in $\mathscr{S}$ is itself an element of $\mathscr{S}$. Hence, in a complete space we may consider limits without fear that the result maybe outside of the space $\mathscr{S}$. Complete spaces play a crucial role, prompting the following terminology. A complete normed, linear space is called a Banach space and a complete inner product space is called a Hilbert space.

For example, the space $\mathbb{R}^N$ with the inner product

$$(x, y) = \sum_{k=1}^{N} x_k y_k \tag{87}$$

is a Hilbert space. The space $\mathbb{R}^N$ with norm

$$\|x\| = \sum_{k=1}^{N} |x_k| \tag{88}$$

is a Banach space.

**Lebesgue and Hardy Spaces.** The following Banach and Hilbert spaces are of frequent practical interest. These spaces consist of signals with certain properties and are distinct through their norm (or inner product).

***The Lebesgue Space $\mathscr{L}_2$.*** As indicated previously, the space of all finite energy signals is denoted $\mathscr{L}_2$. Formally, we may say

$$\mathscr{L}_2 = \{f : \|f\|_2 < \infty\} \tag{89}$$

in which

$$\|f\|_2 = \left( \int_{-\infty}^{\infty} |f(t)|^2\, dt \right)^{\frac{1}{2}} \tag{90}$$

The space $\mathscr{L}_2$ is a Hilbert space with inner product defined by Eq. (86). Two signals are said to be orthogonal if $(x(t), y(t)) = 0$. This provides a natural extension of orthogonality in $\mathbb{R}^N$.

Alternatively, we can consider the norm and inner product of the Fourier transforms of signals. Hence, the inner product of Fourier transforms $X(f)$ and $Y(f)$ is defined completely analogously to the time domain counterpart of Eq. (86) as

$$(X(f), Y(f)) = \int_{-\infty}^{\infty} X^*(f) Y(f)\, df \tag{91}$$

Using the definition of the Fourier transform in Eq. (22), we can rewrite the last expression as

$$\begin{aligned}(X(f), Y(f)) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x * (t) e^{j2\pi ft}\, dt \int_{-\infty}^{\infty} y(u) e^{j2\pi fu} du\, df \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x(t) y(u) \left( \int_{-\infty}^{\infty} e^{-j2\pi f(u-t)}\, df \right) dt\, du\end{aligned} \tag{92}$$

The expression in parentheses equals $\delta(u - t)$, and hence the entire expression simplifies to

$$\begin{aligned}(X(f), Y(f)) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x(t) y(u) \delta(u - t)\, dt\, du \\ &= \int_{-\infty}^{\infty} x(t) y(t)\, dt \\ &= (x(t), y(t))\end{aligned} \tag{93}$$

This result is known as Parseval's theorem. It establishes that there exists a so-called isomorphism between the time-domain and frequency-domain versions of $\mathscr{L}_2$.

***The Hardy Space $\mathscr{H}_2$.*** Hardy spaces in general contain only signals whose Laplace transforms $X(s)$ are analytic in the right half-plane $\Re(s) > 0$ [i.e., if $X(s)$ is rational it does not have poles in the right half plane]. The Hardy space $\mathscr{H}_2$ is formally defined as the set of signals with Laplace transforms $X(s)$ such that $X(s)$ is analytic for $\Re(s) > 0$ and the following norm is finite:

$$\|X(s)\|_2 = \left(\sup_{\alpha>0} \int_{-\infty}^{\infty} X^*(\alpha + j2\pi f) X(\alpha + j2\pi f)\, df\right)^{\frac{1}{2}} < \infty \tag{94}$$

It can be shown that the norm always assumes its supremum as $\alpha$ approaches zero. If we define $X_b(f) = \lim_{\alpha\downarrow 0} X(\alpha + j2\pi f)$, we may replace this norm by the simpler $\mathscr{L}_2$ norm

$$\|X(s)\|_2 = \left(\int_{-\infty}^{\infty} X_b^*(f) X_b(f)\, df\right)^{\frac{1}{2}} \tag{95}$$

Thus, we may regard $\mathscr{H}_2$ as a proper subspace of $\mathscr{L}_2$. Furthermore, by the Paley–Wiener criterion we can conclude that $\mathscr{H}_2$ is isomorphic to the subspace of $\mathscr{L}_2$ that contains only right-sided signals [i.e., signals such that $x(t) = 0$ for $t < 0$]. $\mathscr{H}_2$ is a Hilbert space.

***The Hardy Space $\mathscr{H}_\infty$.*** The Banach space $\mathscr{H}_\infty$ is the space of all signals whose Laplace transform is not only analytic in the right half plane but bounded. The norm for $\mathscr{H}_\infty$ is given by

$$\|X(s)\|_\infty = \sup_f |X_b(f)| \tag{96}$$

where, as before, $X_b(f) = \lim_{\alpha\downarrow 0} X(\alpha + j2\pi f)$.

As we will see shortly, the space $\mathscr{H}_\infty$ plays a crucial role in robust control theory.

**Examples.** To illustrate the usefulness of the concepts introduced previously, we consider two representative examples from the areas of signal processing and control.

***Optimum, Binary Detection.*** In a simple binary communication system, one of two equally likely signals $s_0(t)$ or $s_1(t)$ is transmitted to convey one bit of information. We assume that each signal is of finite duration $T$ and during transmission the signal is corrupted by white Gaussian noise with autocorrelation function $N_0/2\ \delta(\tau)$.

A crucial aspect in the receiver is the design of a linear filter that maximizes the ability to distinguish which of the two possible signals was transmitted. If we denote the impulse response of the filter by $h(T - t)$ and sample the output of the filter at time $t = T$, then a random variable $R$ with conditional Gaussian distribution is obtained.

Specifically, if $s_0(t)$ was transmitted, then the mean $\mu_0$ and variance $\sigma_0^2$ of $R$ are given by

$$\begin{aligned} \mu_0 &= s_0(t) * h(T-t)|_{t=T} = \int_0^T s_0(t)h(t)\,dt = (s_0(t), h(t)) \\ \sigma_0^2 &= \frac{N_0}{2}\int_0^T |h(t)|^2\,dt = \frac{N_0}{2}\|h(t)\|^2 \end{aligned} \tag{97}$$

Similarly, if $s_1(t)$ was transmitted, we obtain

$$\begin{aligned} \mu_1 &= s_1(t) * h(T-t)|_{t=T} = \int_0^T s_1(t)h(t)\,dt = (s_1(t), h(t)) \\ \sigma_1^2 &= \frac{N_0}{2}\int_0^T |h(t)|^2\,dt = \frac{N_0}{2}\|h(t)\|^2 \end{aligned} \tag{98}$$

It is straightforward to demonstrate that for equally likely signals, the probability of error is given by

$$P_e = Q\left(\frac{(s_0(t) - s_1(t), h(t))}{\sqrt{2N_0}\cdot\|h(t)\|}\right) \tag{99}$$

where

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}}\,dy \tag{100}$$

Thus, to minimize the probability of error of the receiver, we must choose $h(t)$ to maximize the ratio

$$\frac{(s_0(t) - s_1(t), h(t))}{\sqrt{2N_0}\|h(t)\|} \tag{101}$$

However, because of the Schwarz inequality, we know immediately that we must select $h(t) = s_0(t) - s_1(t)$ and the resulting probability of error equals

$$P^e = Q\left(\frac{\|s_0(t) - s_1(t)\|}{\sqrt{2N_0}}\right) \tag{102}$$

The filter with impulse response $h(T - t)$ is known as the matched filter for the signals set $s_0(t)$ and $s_1(t)$.

***Robust Control.*** A fundamental problem in robust control is to ensure that a system is designed such that its output in response to any finite energy signal has itself finite energy. This is crucial in systems with noise or other disturbances that can only be bounded in energy.

Let $x(t)$ denote the input to a system with impulse response $g(t)$ and let $y(t)$ be the resulting output. Then we would like to ensure that the ratio of output energy to input energy remains finite for all possible inputs. In terms of the norms defined previously, we can formulate this problem by defining the gain $G$ as

$$\begin{aligned} G &= \sup_{u(t)\neq 0} \frac{\|g(t) * u(t)\|_2}{\|u(t)\|_2} \\ &= \sup_{U(f)\neq 0} \frac{\|G(f)U(f)\|_2}{\|U(f)\|_2} \end{aligned} \tag{103}$$

Here we have employed the transform property of convolution. We can invoke the Schwarz inequality again and further simplify the last expression to obtain

$$G = \sup_f |G(f)| = \|G(s)\|_\infty \tag{104}$$

Thus, the $\mathscr{H}_\infty$ norm measures the maximum possible increase in signal energy for all possible finite energy inputs. Because of the preceding considerations, we say that the $\mathscr{H}_\infty$ norm is induced by the cal $H_2$ norm on the input and output signals.

## SUMMARY

In this article, we have examined in some detail the convolution operation. We have seen that convolution is an operation that is fundamental for all linear, time-invariant systems. After examining important properties of convolution, including several transform properties, we turned our attention to the numerical evaluation of the continuous-time convolution integral. Then we discussed possible approaches for computationally efficient convolution algorithms, emphasizing algorithms based on the fast Fourier transform.

We concluded by examining several applications in which convolution or related operations arise, including error correcting coding and correlation. Finally, we gave a brief introduction to the concept of abstract signal spaces.

## BIBLIOGRAPHY

1. R. Gray, *Toeplitz and circulant matices: A review* [Online], Technical report, Stanford University, 1971 (revised 1977, 1993, 1997). (Available www:http://www-isl.stanford.edu/~gray/toeplitz.pdf)
2. H. V. Sorensen and C. S. Burrus, Fast dft and convolution algorithms, in S. K. Mitra and J. F. Kaiser (eds.), *Handbook for Digital Signal Processing,* New York: Wiley, 1993, pp. 491–610.
3. C. S. Burrus and T. W. Parks, *DFT / FFT and Convolution Algorithms—Theory and Implementation,* New York: Wiley, 1985.
4. O. Ersoy, *Fourier-Related Transforms, Fast Algorithms and Applications,* Upper Saddle River, NJ: Prentice Hall, 1997.
5. S. Lin, *An Introduction to Error-Correcting Codes,* Englewood Cliffs, NJ: Prentice-Hall, 1970.
6. S. B. Wicker, *Error Control System for Digital Communiations and Storage,* Englewood Cliffs, NJ: Prentice-Hall, 1995.
7. A. Papoulis, *Probability, Random Variables, and Stochastic Processes,* 3rd ed., New York: McGraw-Hill, 1991.
8. J. C. Doyle, B. A. Francis, and A. R. Tannenbaum, *Feedback Control Theory,* New York: Macmillan, 1992.
9. D. G. Luenberger, *Optimization by Vector Space Methods,* New York: Wiley, 1969.
10. A. W. Naylor and G. R. Sell, *Linear Operator Theory in Engineering and Science,* vol. 40 of *Applied Mathematical Sciences,* New York: Springer-Verlag, 1982.

BERND-PETER PARIS
George Mason University