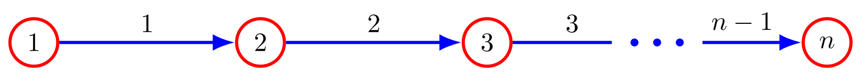


Figure 7.4 Chain graph.

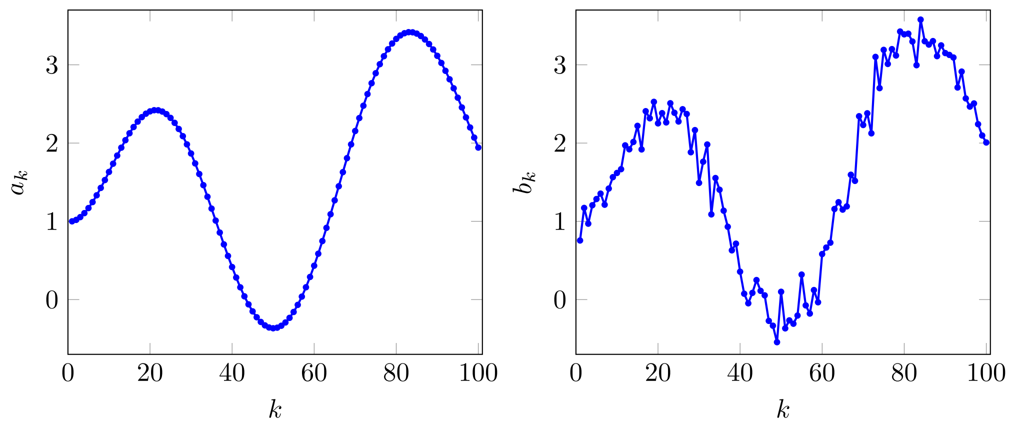


Figure 7.5 Two vectors of length 100, with Dirichlet energy $\mathcal{D}(a) = 1.14$ and $\mathcal{D}(b) = 8.99$.

Chain graph. The incidence matrix and the Dirichlet energy function have a particularly simple form for the *chain graph* shown in figure 7.4, with n vertices and $n-1$ edges. The $n \times (n-1)$ incidence matrix is the transpose of the difference matrix D described on page 119, in (6.5). The Dirichlet energy is then

$$\mathcal{D}(v) = \|Dv\|^2 = (v_2 - v_1)^2 + \cdots + (v_n - v_{n-1})^2,$$

the sum of squares of the differences between consecutive entries of the n-vector v. This is used as a measure of the non-smoothness of the vector v, considered as a time series. Figure 7.5 shows an example.

7.4 Convolution

The *convolution* of an n-vector a and an m-vector b is the $(n+m-1)$-vector denoted $c = a * b$, with entries

$$c_k = \sum_{i+j=k+1} a_i b_j, \quad k = 1, \ldots, n+m-1, \tag{7.2}$$

where the subscript in the sum means that we should sum over all values of i and j in their index ranges $1, \ldots, n$ and $1, \ldots, m$, for which the sum $i+j$ is $k+1$. For example, with $n = 4$, $m = 3$, we have

$$\begin{aligned}
c_1 &= a_1 b_1 \\
c_2 &= a_1 b_2 + a_2 b_1 \\
c_3 &= a_1 b_3 + a_2 b_2 + a_3 b_1 \\
c_4 &= a_2 b_3 + a_3 b_2 + a_4 b_1 \\
c_5 &= a_3 b_3 + a_4 b_2 \\
c_6 &= a_4 b_3.
\end{aligned}$$

Convolution reduces to ordinary multiplication of numbers when $n = m = 1$, and to scalar-vector multiplication when either $n = 1$ or $m = 1$. Convolution arises in many applications and contexts.

As a specific numerical example, we have $(1, 0, -1) * (2, 1, -1) = (2, 1, -3, -1, 1)$, where the entries of the convolution result are found from

$$\begin{aligned}
2 &= (1)(2) \\
1 &= (1)(1) + (0)(2) \\
-3 &= (1)(-1) + (0)(1) + (-1)(2) \\
-1 &= (0)(-1) + (-1)(1) \\
1 &= (-1)(-1).
\end{aligned}$$

Polynomial multiplication. If a and b represent the coefficients of two polynomials

$$p(x) = a_1 + a_2 x + \cdots + a_n x^{n-1}, \qquad q(x) = b_1 + b_2 x + \cdots + b_m x^{m-1},$$

then the coefficients of the product polynomial $p(x)q(x)$ are represented by $c = a * b$:

$$p(x)q(x) = c_1 + c_2 x + \cdots + c_{n+m-1} x^{n+m-2}.$$

To see this we will show that c_k is the coefficient of x^{k-1} in $p(x)q(x)$. We expand the product polynomial into mn terms, and collect those terms associated with x^{k-1}. These terms have the form $a_i b_j x^{i+j-2}$, for i and j that satisfy $i + j - 2 = k - 1$, *i.e.*, $i + j = k - 1$. It follows that $c_k = \sum_{i+j=k+1} a_i b_j$, which agrees with the convolution formula (7.2).

Properties of convolution. Convolution is symmetric: We have $a * b = b * a$. It is also associative: We have $(a * b) * c = a * (b * c)$, so we can write both as $a * b * c$. Another property is that $a * b = 0$ implies that either $a = 0$ or $b = 0$. These properties follow from the polynomial coefficient property above, and can also be directly shown. As an example, let us show that $a * b = b * a$. Suppose p is the polynomial with coefficients a, and q is the polynomial with coefficients b. The two polynomials $p(x)q(x)$ and $q(x)p(x)$ are the same (since multiplication of numbers is commutative), so they have the same coefficients. The coefficients of $p(x)q(x)$ are $a * b$ and the coefficients of $q(x)p(x)$ are $b * a$. These must be the same.

A basic property is that for fixed a, the convolution $a * b$ is a linear function of b; and for fixed b, it is a linear function of a. This means we can express $a * b$ as a matrix-vector product:

$$a * b = T(b)a = T(a)b,$$

where $T(b)$ is the $(n+m-1)\times n$ matrix with entries

$$T(b)_{ij} = \begin{cases} b_{i-j+1} & 1 \le i-j+1 \le m \\ 0 & \text{otherwise} \end{cases} \tag{7.3}$$

and similarly for $T(a)$. For example, with $n=4$ and $m=3$, we have

$$T(b) = \begin{bmatrix} b_1 & 0 & 0 & 0 \\ b_2 & b_1 & 0 & 0 \\ b_3 & b_2 & b_1 & 0 \\ 0 & b_3 & b_2 & b_1 \\ 0 & 0 & b_3 & b_2 \\ 0 & 0 & 0 & b_3 \end{bmatrix}, \qquad T(a) = \begin{bmatrix} a_1 & 0 & 0 \\ a_2 & a_1 & 0 \\ a_3 & a_2 & a_1 \\ a_4 & a_3 & a_2 \\ 0 & a_4 & a_3 \\ 0 & 0 & a_4 \end{bmatrix}.$$

The matrices $T(b)$ and $T(a)$ are called *Toeplitz* matrices (named after the mathematician Otto Toeplitz), which means the entries on any diagonal (*i.e.*, indices with $i-j$ constant) are the same. The columns of the Toeplitz matrix $T(a)$ are simply shifted versions of the vector a, padded with zero entries.

Variations. Several slightly different definitions of convolution are used in different applications. In one variation, a and b are infinite two-sided sequences (and not vectors) with indices ranging from $-\infty$ to ∞. In another variation, the rows of $T(a)$ at the top and bottom that do not contain all the coefficients of a are dropped. (In this version, the rows of $T(a)$ are shifted versions of the vector a, reversed.) For consistency, we will use the one definition (7.2).

Examples.

- *Time series smoothing.* Suppose the n-vector x is a time series, and $a = (1/3, 1/3, 1/3)$. Then the $(n+2)$-vector $y = a * x$ can be interpreted as a *smoothed* version of the original time series: for $i = 3, \ldots, n$, y_i is the average of x_i, x_{i-1}, x_{i-2}. The time series y is called the (3-period) *moving average* of the time series x. Figure 7.6 shows an example.

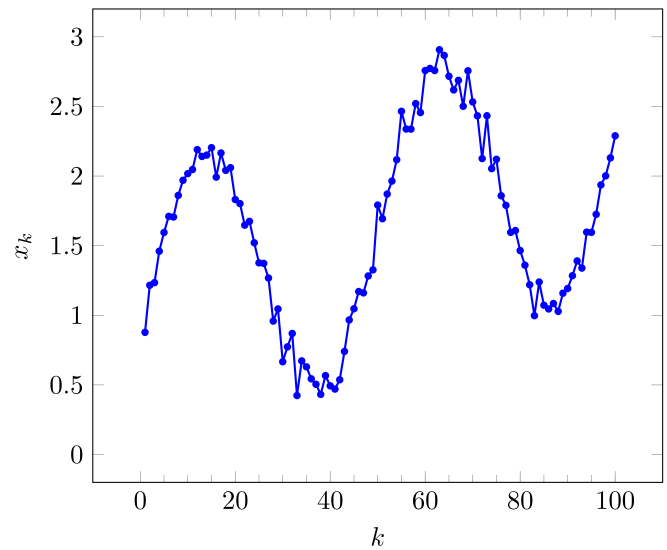
- *First order differences.* If the n-vector x is a time series and $a = (1, -1)$, the time series $y = a * x$ gives the first order differences in the series x:

  $$y = (x_1,\ x_2 - x_1,\ x_3 - x_2,\ \ldots,\ x_n - x_{n-1},\ -x_n).$$

  (The first and last entries here would be the first order difference if we take $x_0 = x_{n+1} = 0$.)

- *Audio filtering.* If the n-vector x is an audio signal, and a is a vector (typically with length less than around 0.1 second of real time) the vector $y = a * x$ is called the *filtered* audio signal, with *filter coefficients* a. Depending on the coefficients a, y will be perceived as enhancing or suppressing different frequencies, like the familiar audio tone controls.

- *Communication channel.* In a modern data communication system, a time series u is transmitted or sent over some channel (*e.g.*, electrical, optical, or radio) to a receiver, which receives the time series y. A very common model is that y and u are related via convolution: $y = c * u$, where the vector c is the *channel impulse response*.



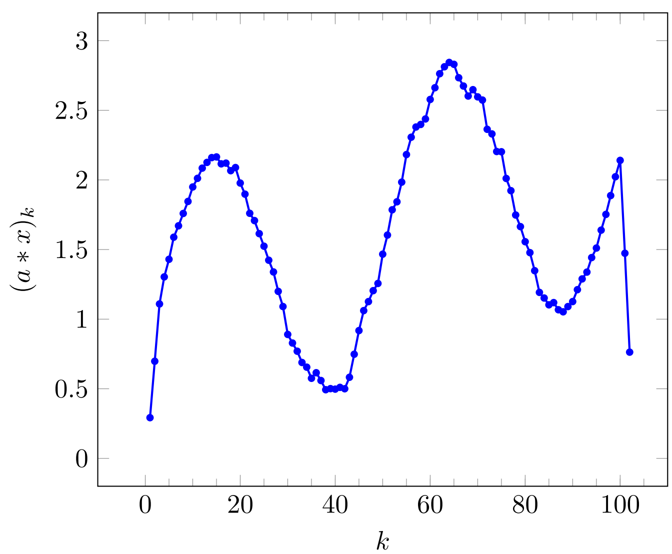


Figure 7.6 *Top.* A time series represented by a vector x of length 100. *Bottom.* The 3-period moving average of the time series as a vector of length 102. This vector is the convolution of x with $a = (1/3, 1/3, 1/3)$.

Input-output convolution system. Many physical systems with an *input* (time series) m-vector u and *output* (time series) y are well modeled as $y = h*u$, where the n-vector h is called the *system impulse response*. For example, u_t might represent the power level of a heater at time period t, and y_t might represent the resulting temperature rise (above the surrounding temperature). The lengths of u and y, n and $m+n-1$, are typically large, and not relevant in these applications. We can express the ith entry of the output y as

$$y_i = \sum_{j=1}^{n} u_{i-j+1} h_j,$$

where we interpret u_k as zero for $k < 0$ or $k > n$. This formula states that y at time i is a linear combination of $u_i, u_{i-1}, \ldots, u_{i-n+1}$, *i.e.*, a linear combination of the current input value u_i, and the past $n-1$ input values $u_{i-1}, \ldots, u_{i-n+1}$. The coefficients are precisely $h_1, \ldots, h_n$. Thus, h_3 can be interpreted as the factor by which the current output depends on what the input was, 2 time steps before. Alternatively, we can say that h_3 is the factor by which the input at any time will affect the output 2 steps in the future.

Complexity of convolution. The naïve method to compute the convolution $c = a * b$ of an n-vector a and an m-vector b, using the basic formula (7.2) to calculate each c_k, requires around $2mn$ flops. The same number of flops is required to compute the matrix-vector products $T(a)b$ or $T(b)a$, taking into account the zeros at the top right and bottom left in the Toeplitz matrices $T(b)$ and $T(a)$. Forming these matrices requires us to store mn numbers, even though the original data contains only $m+n$ numbers.

It turns out that the convolution of two vectors can be computed far faster, using a so-called *fast convolution algorithm*. By exploiting the special structure of the convolution equations, this algorithm can compute the convolution of an n-vector and an m-vector in around $5(m+n)\log_2(m+n)$ flops, and with no additional memory requirement beyond the original $m+n$ numbers. The fast convolution algorithm is based on the *fast Fourier transform* (FFT), which is beyond the scope of this book. (The Fourier transform is named for the mathematician Jean-Baptiste Fourier.)

7.4.1 2-D convolution

Convolution has a natural extension to multiple dimensions. Suppose that A is an $m\times n$ matrix and B is a $p\times q$ matrix. Their convolution is the $(m+p-1)\times(n+q-1)$ matrix

$$C_{rs} = \sum_{i+k=r+1,\ j+l=s+1} A_{ij} B_{kl}, \quad r = 1, \ldots, m+p-1, \quad s = 1, \ldots, n+q-1,$$

where the indices are restricted to their ranges (or alternatively, we assume that A_{ij} and B_{kl} are zero, when the indices are out of range). This is *not* denoted

$C = A * B$, however, in standard mathematical notation. So we will use the notation $C = A \star B$.

The same properties that we observed for 1-D convolution hold for 2-D convolution: We have $A \star B = B \star A$, $(A \star B) \star C = A \star (B \star C)$, and for fixed B, $A \star B$ is a linear function of A.

Image blurring. If the $m \times n$ matrix X represents an image, $Y = X \star B$ represents the effect of *blurring* the image by the *point spread function* (PSF) given by the entries of the matrix B. If we represent X and Y as vectors, we have $y = T(B)x$, for some $(m + p - 1)(n + q - 1) \times mn$-matrix $T(B)$.

As an example, with

$$B = \begin{bmatrix} 1/4 & 1/4 \\ 1/4 & 1/4 \end{bmatrix}, \tag{7.4}$$

$Y = X \star B$ is an image where each pixel value is the average of a 2×2 block of 4 adjacent pixels in X. The image Y would be perceived as the image X, with some blurring of the fine details. This is illustrated in figure 7.7 for the 8×9 matrix

$$X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{7.5}$$

and its convolution with B,

$$X \star B = \begin{bmatrix} 1/4 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/4 \\ 1/2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1/2 \\ 1/2 & 1 & 3/4 & 1/2 & 1/2 & 1/2 & 1/2 & 3/4 & 1 & 1/2 \\ 1/2 & 1 & 3/4 & 1/4 & 1/4 & 1/2 & 1/4 & 1/2 & 1 & 1/2 \\ 1/2 & 1 & 1 & 1/2 & 1/2 & 1 & 1/2 & 1/2 & 1 & 1/2 \\ 1/2 & 1 & 1 & 1/2 & 1/2 & 1 & 1/2 & 1/2 & 1 & 1/2 \\ 1/2 & 1 & 1 & 3/4 & 3/4 & 1 & 3/4 & 3/4 & 1 & 1/2 \\ 1/2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1/2 \\ 1/4 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/4 \end{bmatrix}.$$

With the point spread function

$$D^{\text{hor}} = \begin{bmatrix} 1 & -1 \end{bmatrix},$$

the pixel values in the image $Y = X \star D^{\text{hor}}$ are the horizontal first order differences of those in X:

$$Y_{ij} = X_{ij} - X_{i,j-1}, \quad i = 1, \ldots, m, \quad j = 2, \ldots, n$$

(and $Y_{i1} = X_{i1}$, $X_{i,n+1} = -X_{in}$ for $i = 1, \ldots, m$). With the point spread function

$$D^{\text{ver}} = \begin{bmatrix} 1 \\ -1 \end{bmatrix},$$

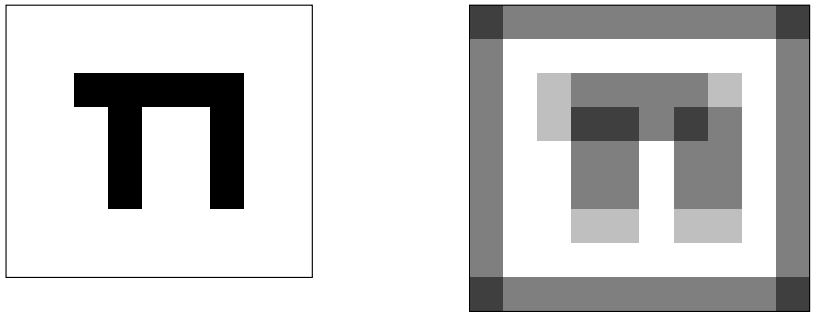


Figure 7.7 An 8×9 image and its convolution with the point spread function (7.4).

the pixel values in the image $Y = X \star D^{\text{ver}}$ are the vertical first order differences of those in X:

$$Y_{ij} = X_{ij} - X_{i-1,j}, \quad i = 2, \ldots, m, \quad j = 1, \ldots, n$$

(and $Y_{1j} = X_{1j}$, $X_{m+1,j} = -X_{mj}$ for $j = 1, \ldots, n$). As an example, the convolutions of the matrix (7.5) with D^{hor} and D^{ver} are

$$X \star D^{\text{hor}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 1 & 0 & 0 & -1 & 1 & 0 & -1 & 1 & 0 & -1 \\ 1 & 0 & 0 & -1 & 1 & 0 & -1 & 1 & 0 & -1 \\ 1 & 0 & 0 & -1 & 1 & 0 & -1 & 1 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

and

$$X \star D^{\text{ver}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}.$$

Figure 7.8 shows the effect of convolution on a larger image. The figure shows an image of size 512×512 and its convolution with the 8×8 matrix B with constant entries $B_{ij} = 1/64$.

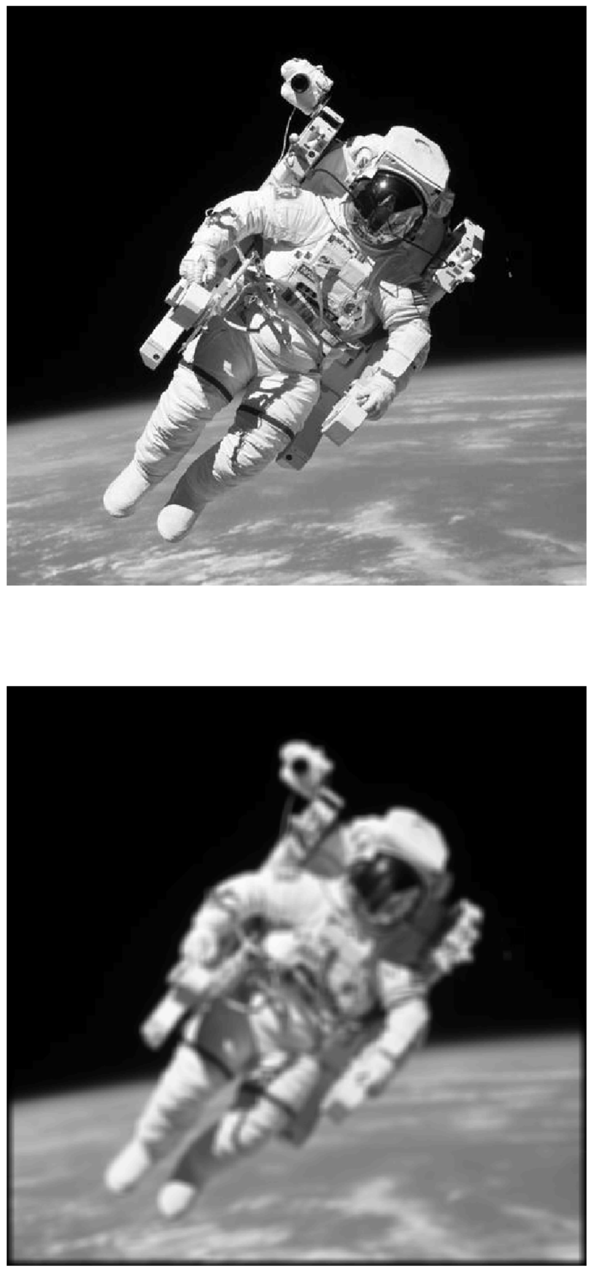


Figure 7.8 512×512 image and the 519×519 image that results from the convolution of the first image with an 8×8 matrix with constant entries $1/64$. Image credit: NASA.