known that appropriate averaging requires the weighting of each data point g_n with the inverse of the variance $w_n = 1/\sigma_n^2$. Then, an estimate of the mean value is given by

$$\langle g \rangle = \sum_{n=1}^{N} g_n / \sigma_n^2 \Bigg/ \sum_{n=1}^{N} 1/\sigma_n^2 \tag{9.106}$$

while the standard deviation of the mean is

$$\sigma_{\langle g \rangle}^2 = 1 \Bigg/ \sum_{n=1}^{N} 1/\sigma_n^2 \tag{9.107}$$

The application of *weighted averaging* to image processing is known as *normalized convolution* [9]. The averaging is now extended to a local neighborhood. Each pixel enters the convolution sum with a weighting factor associated with it. Thus, normalized convolution requires two signals. One is the image $\boldsymbol{G}$ to be processed, the other an image $\boldsymbol{W}$ with the weighting factors.

By analogy to Eqs. (9.106) and (9.107), normalized convolution with the mask $\boldsymbol{H}$ is defined as

$$\boldsymbol{G}' = \frac{\boldsymbol{H} * (\boldsymbol{W} \cdot \boldsymbol{G})}{\boldsymbol{H} * \boldsymbol{W}} \tag{9.108}$$

A normalized convolution with the mask $\boldsymbol{H}$ essentially transforms the image $\boldsymbol{G}$ and the weighting image $\boldsymbol{W}$ into a new image $\boldsymbol{G}'$ and a new weighting image $\boldsymbol{W}' = \boldsymbol{H} * \boldsymbol{W}$, which can undergo further processing.

Normalized convolution is just adequate consideration of pixels with spatially variable statistical errors. "Standard" convolution can be regarded as a special case of normalized convolution. Then all pixels are assigned the same weighting factor and it is not required to use a weighting image, because the factor remains a constant.

The flexibility of normalized convolution is given by the choice of the weighting image. The weighting image is not necessarily associated with an error. It can be used to select and/or amplify pixels with certain features. In this way, normalized convolution becomes a versatile nonlinear operator. The application of normalized convolution is discussed in a number of contributions in the application gallery: Sections A18, A20, A16, and A23.

9.6 Interpolation

Interpolation of digital signals is required for a wide range of signal-processing tasks whenever any operation shifts the digital points of the output signal so that they no longer coincide with the grid points of the input signal. This occurs, among others, with the following operations:

Geometric operations. For many applications, the geometrical distortions introduced by optical systems [CVA1, Chapter 4]) are not acceptable and must be corrected. For satellite images, it is often required to recompute the image data to a different projective mapping.

Signal registration. If data are taken with different sensors, these sensors will almost never be in perfect spatial alignment. Thus it is required to map them onto common spatial coordinates for further joint processing.

Multiresolution signal processing. For multigrid data structures, such as pyramids (Section 8.10), signals are represented at different resolution levels. On such data structures it is necessary to interpolate missing points from coarser levels to be able to process them at a finer level.

Coarse-to-fine strategies. Coarse-to-fine strategies are an often used concept on multigrid data structures if the processing involves images that are shifted to each other either because of a different sensor (image registration), a different perspective (stereo images) or motion of objects (Chapter 10). In all these cases it is required to warp the images with the determined displacement vector field before processing at the next finer resolution [CVA2, Chapter 14].

Test image generation. In order to evaluate algorithms, it is important to apply them to known signals. For image-sequence processing, for example, it is useful to simulate displacement vector fields by warping images correspondingly.

For a long time there was little effort put into interpolation algorithms for computer vision. Thus most of the available procedures have been invented for computer graphics in the framework of *photorealistic rendering*. An excellent survey in this respect is provided by Wolberg [11]. Only with increasing demand for subpixel-accurate computer vision algorithms have the researchers become aware of the importance of accurate interpolation algorithms. The demands are quite high. As a rule of thumb, interpolation should neither change the amplitude of a signal by more than 1 % nor shift any signal by more than 0.01.

The analysis of the structure in small neighborhoods is a key element in higher-dimensional signal processing. Changes in the gray values reveal either the edge of an object or the type of texture.

9.6.1 Interpolation as convolution

The basis of interpolation is the sampling theorem (Section 8.4.2). This theorem states that the digital signal *completely* represents the continuous signal provided the sampling conditions are met. This basic fact suggests the following general framework for interpolation:

Reconstruction of continuous signal. From the sampled signal a continuous or a higher-resolution representation is reconstructed.

Filtering. Before a resampling can be performed, it is necessary to check whether a prefiltering of the data is required. Whenever the data are to be resampled with a coarser resolution, aliasing could occur because the sampling condition is no longer met (Section 8.4.3).

Resampling. This step finally forms the new digital signal.

Of course, a certain procedure for interpolation can perform two or even all of these steps in a single operation. However, it is still helpful for a better understanding of the procedure to separate it into these steps.

Although these procedures sound simple and straightforward, they are not. The problem is related to the fact that the reconstruction of the continuous signal from the sampled signal in practice is quite involved and can be performed only approximately. Thus, we need to balance the computational effort with the residual error for a given interpolation task.

Generally, a continuous multidimensional signal is interpolated from values at all points of a lattice by (Section 8.4.4)

$$g_r(\boldsymbol{x}) = \sum_{p=1}^{P} \sum_{\boldsymbol{n}} g_s(\boldsymbol{r_n} + \boldsymbol{s}_p) h(\boldsymbol{x} - (\boldsymbol{r_n} + \boldsymbol{s}_p)) \tag{9.109}$$

In this equation $\boldsymbol{r_n}$ are the translation vectors of the lattice and $\boldsymbol{s}_p$ the offsets of the P points in the *primitive cell* of the lattice. If a continuous signal is required but only the value at a shifted point $\boldsymbol{p}$ (Eq. (9.109)) reduces to

$$g_r(\boldsymbol{p}) = \sum_{p=1}^{P} \sum_{\boldsymbol{n}} g_s(\boldsymbol{r_n} + \boldsymbol{s}_p) h(\boldsymbol{p} - (\boldsymbol{r_n} + \boldsymbol{s}_p)) \tag{9.110}$$

This equation reveals that interpolation is nothing else but a generalized convolution operation of the points on a discrete lattice with sampled values from the interpolation kernel. The only difference is that the result of the operation is not written back to the same lattice but to a shifted lattice. Thus an interpolation operation can be described by a transfer function. According to the discussion of the *sampling theorem* in Sections 8.4.2 and 8.4.4, the ideal interpolation function has a transfer function that is constantly one within the first Brillouin zone and zero outside.

For the rest of this section, we will restrict all considerations to orthogonal lattices because interpolation of multidimensional signals is much easier to handle on these grids. On an orthogonal lattice with only one point per primitive cell ($P = 1$), the interpolation in Eq. (9.109)

reduces to

$$g_r(\tilde{\boldsymbol{x}}) = \sum_{\boldsymbol{n}} g_s(\boldsymbol{n}) h(\tilde{\boldsymbol{x}} - \boldsymbol{n}) \tag{9.111}$$

In this equation all vectors in the spatial domain are expressed in units of the lattice constants: $\tilde{x}_d = x_d/\Delta x_d$. Thus, the components of the translation vector $\boldsymbol{r}$ are integers and are replaced by $\boldsymbol{n} = [n_1, \ldots, n_D]^T$, the vectorial index that counts the translations vectors on a D-dimensional lattice.

The ideal transfer function for interpolation of a D-dimensional signal is then a D-dimensional box function

$$\hat{g}_r(\tilde{\boldsymbol{k}}) = \hat{g}(\tilde{\boldsymbol{k}}) \prod_{d=1}^{D} \Pi(2\tilde{\boldsymbol{k}}) \tag{9.112}$$

where $\tilde{\boldsymbol{k}}$ is the wave number normalized to the Nyquist limit according to Eq. (8.34). It follows that the ideal interpolation function h is the Fourier transform of the box function, the sinc function

$$h(\tilde{\boldsymbol{x}}) = \prod_{d=1}^{D} \frac{\sin(\pi \tilde{x}_d)}{\pi \tilde{x}_d} = \prod_{d=1}^{D} \operatorname{sinc}(\tilde{x}_d) \tag{9.113}$$

This ideal interpolation mask cannot be used in practice as it is infinite. Thus an optimal approximation must be found that minimizes the deviations from the ideal transfer function.

9.6.2 General properties of interpolation kernels

In this section some general properties of interpolation are summarized that are useful for the design of optimal interpolation masks.

Symmetries. An interpolation mask can have an even or odd number of coefficients. Because of symmetry reasons, the interpolation interval of these two types of interpolation masks is different. For a mask with an even number of coefficients (Fig. 9.14a), the symmetry center is between the two central points of the interpolation mask. Because any interpolation mask can have an interpolation interval of one distance between two points of the mask, the interpolation interval is limited to the interval between the two central points of the mask. For points outside of this range, the mask is shifted a corresponding number of points on the lattice, so that the point to be interpolated again is within this central interval.

For a mask with an odd number of coefficients (Fig. 9.14b), the symmetry center coincides with the central point. Thus the interpolation interval is now half the distance between points on the lattice on both

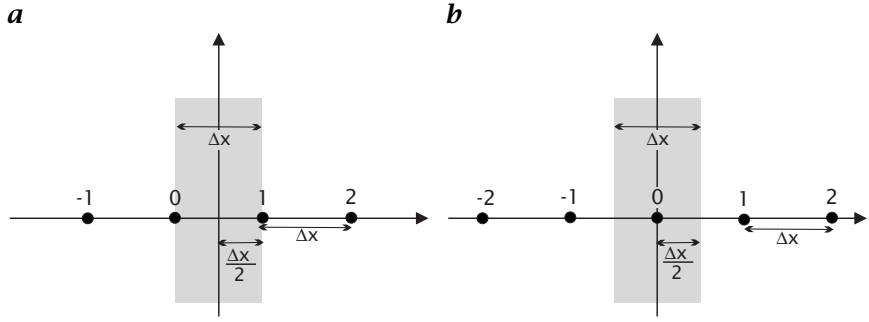

Figure 9.14: *Interpolation interval for interpolation masks with* ***a*** *an even and* ***b*** *an odd number of coefficients .*

sides of the central point. The symmetry conditions for these two types of interpolation filters are analogous to type I and type II averaging filters discussed in Sections 9.2.5 and 9.5.3.

Interpolation condition. There are some general constraints that must be met by any interpolation filter. They result from the simple fact that the interpolated values in Eq. (9.111) at the lattice points $\boldsymbol{n}$ should reproduce the lattice points and not depend on any other lattice points. From this condition, we can infer the *interpolation condition*:

$$h(\boldsymbol{n}) = \begin{cases} 1 & \boldsymbol{n} = \boldsymbol{0} \\ 0 & \text{otherwise} \end{cases} \tag{9.114}$$

Therefore any interpolation mask must have zero crossings at all grid points except the zero point where it is one. The ideal interpolation mask in Eq. (9.113) meets this interpolation condition.

More generally, we can state that any discrete interpolation mask sampled from the continuous interpolation kernel should meet the following condition:

$$^{\tilde{\boldsymbol{x}}}H_{\boldsymbol{n}} = \sum_{\boldsymbol{n}} h(\boldsymbol{n} + \tilde{\boldsymbol{x}}) = 1 \quad \Longleftrightarrow \quad ^{\tilde{\boldsymbol{x}}}\hat{H}_{\boldsymbol{0}} = 1 \tag{9.115}$$

This generalized condition indicates only that a constant signal ($\tilde{\boldsymbol{k}} = \boldsymbol{0}$) is not changed by an interpolation operation.

Separability. The ideal interpolation function in Eq. (9.113) is separable. Therefore, interpolation can as easily be formulated for higher-dimensional images. We can expect that all solutions to the interpolation problem will also be separable. Consequently, we need only dis-

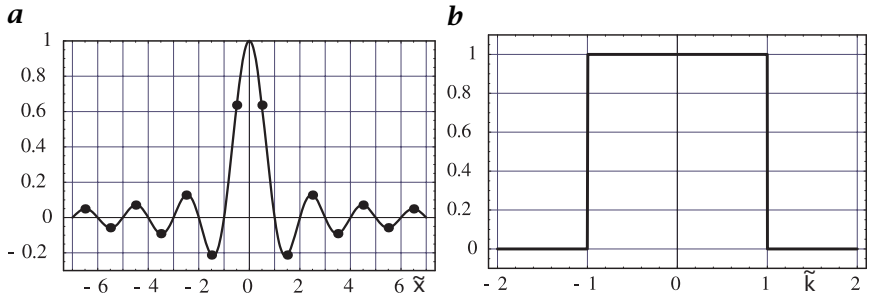

Figure 9.15: ***a*** *Ideal 1-D interpolation mask and* ***b*** *its transfer function. The values for the coefficients of the discrete mask to interpolate intermediate lattice points ($\tilde{x} = 1/2$) are marked by dots.*

cuss the 1-D interpolation problem

$$g_r(\tilde{x}) = \sum_{n=-R}^{R} g_n h(\tilde{x} - n) \tag{9.116}$$

where n and R take half-integer values for interpolation masks with an even number of coefficients and integer values for interpolation masks with an odd number of coefficients; x is given here in units of the lattice constant $\tilde{x} = x/\Delta x$. The 1-D ideal interpolation mask $\mathrm{sinc}(\tilde{x})$ and its transfer function $\Pi(2\tilde{k})$ are illustrated in Fig. 9.15.

Once a good interpolation mask is found for 1-D interpolation, we also have a solution for the D-dimensional interpolation problem.

An important special case is the interpolation to intermediate lattice points half-way between the existing lattice points. This scheme doubles the resolution and image size in all directions in which it is applied. The coefficients of the corresponding interpolation mask are the values of the $\mathrm{sinc}(\tilde{x})$ function sampled at all half-integer values:

$$h = \left[\frac{(-1)^{r-1}\,2}{(2r-1)\pi} \;\cdots\; -\frac{2}{3\pi} \;\; \frac{2}{\pi} \;\; \frac{2}{\pi} \;\; -\frac{2}{3\pi} \;\cdots\; \frac{(-1)^{r-1}\,2}{(2r-1)\pi} \right] \tag{9.117}$$

The coefficients are of alternating sign.

Interpolation error analysis. The fact that interpolation is a convolution operation and thus can be described by a transfer function in Fourier space Equation (9.113) gives us a tool to rate the errors associated with an interpolation technique. The box-type transfer function for the ideal interpolation function simply means that all wave numbers within the range of possible wave numbers $|k_d| \leq \Delta x_d/\pi$ experience neither a phase shift nor amplitude damping. Also, no wave number beyond the allowed interval is present in the interpolated signal, because the transfer function is zero there.

9.6.3 Interpolation in Fourier space

Interpolation reduces to a simple operation in the Fourier domain. The transfer function of an ideal interpolation kernel is a box function that is zero outside the wave numbers that can be represented (see Eq. (9.113)). This basic fact suggests the following interpolation procedure in Fourier space:

1. *Enlargement of the Fourier transform of the signal.* If the discrete Fourier transform of an M^D multidimensional signal is increased to an M'^D array, the array in the spatial domain is also increased to the same size. Because of the reciprocity of the Fourier transform, the image *size* remains unchanged. Only the spacing between pixels in the spatial domain is decreased, resulting in a higher spatial resolution:

$$M\Delta k_d \rightarrow M'\Delta k_d \quad \Longleftrightarrow \quad \Delta x = \frac{2\pi}{M\Delta k} \rightarrow \Delta x' = \frac{2\pi}{M'\Delta k} \tag{9.118}$$

   The padded area in the Fourier space is filled with zeroes.
2. *Inverse Fourier transform.* All that needs to be done is the computation of an inverse Fourier transform to obtain a higher resolution signal.

The Fourier transform can also be used to shift a signal by any distance without changing the signal resolution. Then the following three-step procedure must be applied.

1. *Forward Fourier transform.*
2. *Multiplication with a phase factor.* According to the *shift theorem* (Table 8.6), a shift in the spatial domain by a distance x_s corresponds to the multiplication of the Fourier transform by the following phase factor:

$$g(\boldsymbol{x}) \rightarrow g(\boldsymbol{x}-\boldsymbol{s}) \quad \Longleftrightarrow \quad \hat{G}_{\boldsymbol{u}} \rightarrow \exp(-2\pi \mathrm{i}\boldsymbol{u}\boldsymbol{s})\hat{G}_{\boldsymbol{u}} \tag{9.119}$$

   where the vectorial shift $\boldsymbol{s}$ is given in units of the lattice constants Δx_d.
3. *Inverse Fourier transform.*

Theoretically, these simple procedures result in perfectly interpolated signals. A closer look, however, reveals that these techniques have some serious drawbacks.

First, the Fourier transform of a finite image implies a cyclic repetition of the image both in the spatial and Fourier domain. Thus, the convolution performed by the Fourier transform is also cyclic. This means that at the right or left edge of the image, convolution continues with the image at the opposite side. Because the real world is not

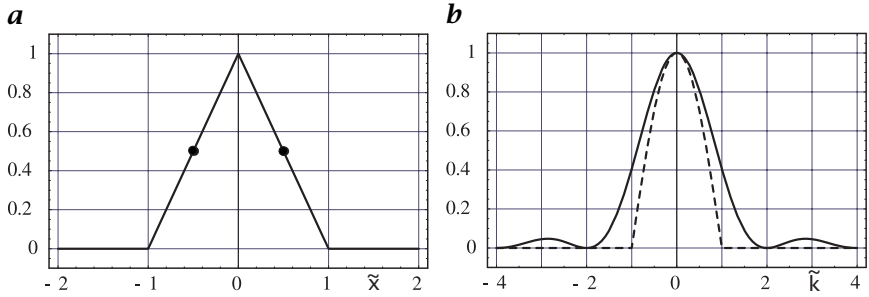

Figure 9.16: ***a*** *One-dimensional linear interpolation:* ***a*** *continuous interpolation mask and* ***b*** *its transfer function. The values for the coefficients of the discrete mask to interpolate intermediate lattice points ($\tilde{x} = 1/2$) are marked by dots.*

periodic and interpolation masks are large, this may lead to significant distortions of the interpolation even at quite large distances from the edges of the image.

Second, the Fourier transform can be computed efficiently only for a specified number of values for M' [CVA2, Section 3.4]. Therefore, the Fourier-transform based interpolation is limited to scaling factors of powers of two.

Third, the Fourier transform is a global transform. Thus it can be applied only to a global scaling of the signal by an integer factor.

9.6.4 Polynomial interpolation

Linear interpolation. *Linear interpolation* is the classic approach to interpolation. The interpolated points are on pieces of straight lines connecting neighboring grid points. In order to simplify the expressions in the following, we use normalized spatial coordinates $\tilde{x} = x/\Delta x$. We locate the two grid points at $-1/2$ and $1/2$. This yields the interpolation equation

$$g(\tilde{x}) = \frac{g_{1/2} + g_{-1/2}}{2} + (g_{1/2} - g_{-1/2})\,\tilde{x} \quad \text{for} \quad |\tilde{x}| \leq 1/2 \tag{9.120}$$

By comparison of Eq. (9.120) with Eq. (9.116), we can conclude that the continuous interpolation mask for linear interpolation is the triangle function

$$h_1(\tilde{x}) = \Lambda(\tilde{x}) = \begin{cases} 1 - |\tilde{x}| & |\tilde{x}| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{9.121}$$

The transfer function of the interpolation mask for linear interpolation, the triangle function $h_1(x)$ Eq. (9.121), is the squared sinc function

$$\hat{h}_1(\tilde{k}) = \frac{\sin^2(\pi\tilde{k}/2)}{(\pi\tilde{k}/2)^2} = \operatorname{sinc}^2(\tilde{k}/2) \tag{9.122}$$

A comparison with the ideal transfer function for interpolation Equation (9.112) (see also Fig. 9.15b and Fig. 9.16b), shows that two distortions are introduced by linear interpolation:

1. While low wave numbers (and especially the mean value $\tilde{k} = 0$) are interpolated correctly, high wave numbers are reduced in amplitude, resulting in some degree of smoothing. At $\tilde{k} = 1$, the transfer function is reduced to about 40%: $\hat{h}_1(1) = (2/\pi)^2 \approx 0.4$.
2. As $\hat{h}_1(\tilde{k})$ is not zero at wave numbers $\tilde{k} > 1$, some spurious high wave numbers are introduced. If the continuously interpolated image is resampled, this yields moderate aliasing. The first sidelobe has an amplitude of $(2/3\pi)^2 \approx 0.045$.

If we interpolate only the intermediate grid points at $\tilde{x} = 0$, the continuous interpolation function Eq. (9.121) reduces to a discrete convolution mask with values at $\tilde{x} = [\ldots\ -3/2\ \ -1/2\ 1/2\ 3/2\ \ldots]$. As Eq. (9.121) is zero for $|\tilde{x}| \geq 1$, we obtain the simple interpolation mask $H = 1/2[1\ 1]$ with the transfer function

$$\hat{H}_1(\tilde{k}) = \cos \pi \tilde{k}/2 \tag{9.123}$$

The transfer function is real, so no phase shifts occur. The significant amplitude damping at higher wave numbers, however, shows that structures with high wave numbers are not correctly interpolated.

Higher-order polynomial interpolation. Given the significant limitations of linear interpolation, we ask whether higher-order interpolation schemes perform better. The basic principle of linear interpolation was that a straight line was drawn to pass through two neighboring points. In the same way, we can use a polynomial of degree P with $P + 1$ unknown coefficients a_p to pass through $P + 1$ points:

$$g_r(\tilde{x}) = \sum_{p=0}^{P} a_p \tilde{x}^p \tag{9.124}$$

For symmetry reasons, the lattice points are placed at the positions

$$\tilde{k}_p = \frac{2p - P}{2} \tag{9.125}$$

For an even number of points (P is odd), the lattice points are located at half-integer values.

From the interpolation condition at the grid points $g_r(\tilde{k}_p) = g_p$, we obtain a linear equation system with $P+1$ equations and $P+1$ unknowns

a_P of the following form when P is odd:

$$\begin{bmatrix} g_0 \\ \vdots \\ g_{(P-1)/2} \\ g_{(P+1)/2} \\ \vdots \\ g_P \end{bmatrix} = \begin{bmatrix} 1 & -P/2 & P^2/4 & -P^3/8 & \cdots \\ \vdots & & & & \\ 1 & -1/2 & 1/4 & -1/8 & \cdots \\ 1 & 1/2 & 1/4 & 1/8 & \cdots \\ \vdots & & & & \\ 1 & P/2 & P^2/4 & P^3/8 & \cdots \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_{(P-1)/2} \\ a_{(P+1)/2} \\ \vdots \\ a_P \end{bmatrix} \tag{9.126}$$

or written as a matrix equation:

$$\boldsymbol{g} = \boldsymbol{M}\boldsymbol{a} \quad \text{with} \quad M_{pq} = \left(\frac{2q-P}{2}\right)^p, \; p,q \in [0,P] \tag{9.127}$$

For a cubic polynomial ($P = 3$), the solution of the equations system is

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \frac{1}{48} \begin{bmatrix} -3 & 27 & 27 & -3 \\ 2 & -54 & 54 & -2 \\ 12 & -12 & -12 & 12 \\ -8 & 24 & -24 & 8 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} \tag{9.128}$$

Using Eqs. (9.124) and (9.128) we can express the interpolated values for the position ϵ in the interval $[-1/2, 1/2]$ as

$$\begin{aligned} g(\epsilon) &= \frac{9-4\epsilon^2}{16}(g_1+g_2) - \frac{1-4\epsilon^2}{16}(g_0+g_3) \\ &+ \frac{\epsilon(9-4\epsilon^2)}{8}(g_2-g_1) - \frac{\epsilon(1-4\epsilon^2)}{24}(g_3-g_0) \end{aligned} \tag{9.129}$$

Thus the interpolation mask is

$$\left[\frac{-\alpha}{16} + \frac{\epsilon\alpha}{24}, \; \frac{8+\alpha}{16} + \frac{\epsilon(8+\alpha)}{8}, \; \frac{8+\alpha}{16} - \frac{\epsilon(8+\alpha)}{8}, \; \frac{-\alpha}{16} - \frac{\epsilon\alpha}{24}\right] \tag{9.130}$$

with $\alpha = 1 - 4\epsilon^2$. For $\epsilon = 0$ ($\alpha = 1$), the mask reduces to

$$\frac{1}{16}[-1 \; 9 \; 9 \; -1] \tag{9.131}$$

It is not very helpful to go to higher-order polynomial interpolation. With increasing degree P of the interpolating polynomial, the transfer function approaches the ideal transfer function better but convergence is too slow (Fig. 9.17). Less than 1% amplitude error is given only for a polynomial of degree 7 for $\tilde{k} < 0.45$. Thus the extra effort of higher-order polynomial interpolation does not pay off.

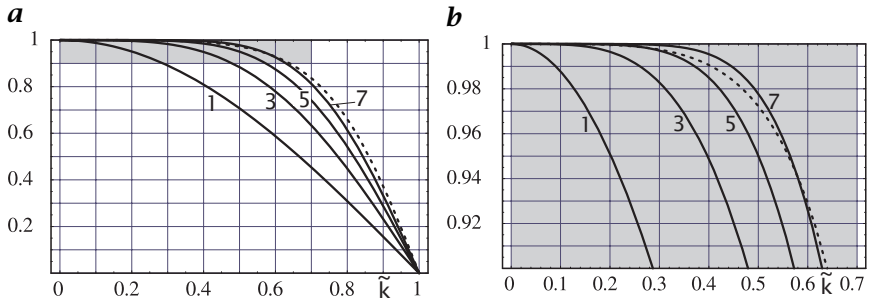

Figure 9.17: *Transfer function of polynomial interpolation filters to interpolate the value between two grid points ($\epsilon = 0$). The degree of the polynomial (1 = linear, 3 = cubic, etc.) is marked in the graph. The dashed line marks the transfer function for cubic B-spline interpolation (Section 9.6.5):* ***a*** *Full range;* ***b*** *sector as marked in* ***a****.*

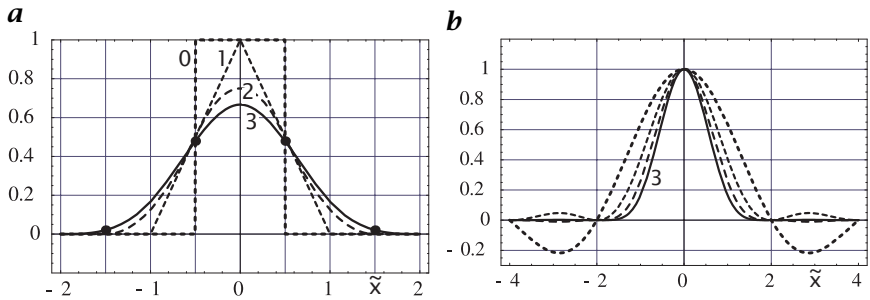

Figure 9.18: ***a*** *B-spline interpolation kernels of order 0 (nearest neighbor), 1 (linear interpolation), 2 (quadratic B-spline), and 3 (cubic B-spline);* ***b*** *corresponding transfer functions.*

9.6.5 Spline-based interpolation

Besides the still limited accuracy, polynomial interpolation has another significant disadvantage. The interpolated curve is not continuous at the grid points already in its first derivative. This is due to the fact that for each interval between grid points another polynomial is taken. Thus, only the interpolated function is continuous at the grid points but not the derivatives.

Splines avoid this disadvantage by additional constraints for the continuity of derivatives at the grid points. From the many classes of splines, we will here discuss only one class, *B-splines*, and introduce *cubic B-spline interpolation*. From the background of signal processing, the easiest access to B-splines is their convolution property. The kernel of a P-order B-spline curve is generated by convolving the box function

P times with itself (Fig. 9.18a):

$$\beta_P(\tilde{x}) = \underbrace{\Pi(\tilde{x}) * \ldots * \Pi(\tilde{x})}_{(P+1)\text{ times}} \tag{9.132}$$

The transfer function of the box function is the sinc function (see Fig. 9.15). Therefore, the transfer function of the P-order B-spline is

$$\hat{\beta}_P(\hat{k}) = \left(\frac{\sin \pi\tilde{k}/2}{(\pi\tilde{k}/2)}\right)^{P+1} \tag{9.133}$$

Figure 9.18b shows that the B-spline function does not make a suitable interpolation function. The transfer function decreases too early, indicating that B-spline interpolation performs too much averaging. Moreover, the B-spline kernel does not meet the interpolation condition Eq. (9.114) for $P > 1$. Thus, B-splines can be used only for interpolation if the discrete grid points are first transformed in such a way that a following convolution with the B-spline kernel restores the original values at the grid points.

This transformation, known as the B-spline transformation, is constructed from the following condition:

$$g_p(x) = \sum_n c_n \beta_P(x - x_n) \quad \text{with} \quad g_p(x_n) = g(x_n) \tag{9.134}$$

If centered around a grid point, the cubic B-spline interpolation kernel is unequal to zero for only three grid points. The coefficients $\beta_3(-1) = \beta_{-1}, \beta_3(0) = \beta_0$, and $\beta_3(1) = \beta_1$ are 1/6, 2/3, and 1/6. The convolution of this kernel with the unknown B-spline transform values c_n should result in the original values g_n at the grid points. Therefore,

$$\boldsymbol{g} = \boldsymbol{c} * \boldsymbol{\beta_3} \quad \text{or} \quad g_n = \sum_{n'=-1}^{1} c_{n+n'} \beta_{n'} \tag{9.135}$$

Equation (9.134) constitutes the sparse linear equation system

$$\begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{N-1} \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 4 & 1 & 0 & \ddots & 0 & 1 \\ 1 & 4 & 1 & 0 & \ddots & 0 \\ 0 & 1 & 4 & 1 & 0 & \ddots \\ \ddots & & & & \ddots & \ddots \\ \ddots & \ddots & 1 & 4 & 1 & 0 \\ 0 & \ddots & 0 & 1 & 4 & 1 \\ 1 & 0 & \ddots & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{bmatrix} \tag{9.136}$$

using cyclic boundary conditions. The determination of the B-spline transformation thus requires the solution of a linear equation system with N unknowns. The special form of the equation system as a convolution operation, however, allows for a more efficient solution. In Fourier space, Eq. (9.135) reduces to

$$\hat{\boldsymbol{g}} = \hat{\boldsymbol{\beta}}_3 \hat{\boldsymbol{c}} \tag{9.137}$$

The transfer function of $\boldsymbol{\beta}_3$ is $\hat{\beta}_3(\tilde{k}) = 2/3 + 1/3\cos(\pi\tilde{k})$. As this function has no zeroes, we can compute $\boldsymbol{c}$ by inverse filtering, that is, convoluting $\boldsymbol{g}$ with a mask that has the transfer function

$$\hat{\beta}_3^{-1}(\tilde{k}) = \hat{\beta}_T(\tilde{k}) = \frac{1}{2/3 + 1/3\cos(\pi\tilde{k})} \tag{9.138}$$

This is the transfer function of a recursive relaxation filter (Section 9.3.6) that is applied first in the forward and then in the backward direction with the following recursion [12]:

$$\begin{aligned} g'_n &= g_n - (2-\sqrt{3})(g'_{n-1} - g_n) \\ c'_n &= g'_n - (2-\sqrt{3})(c_{n+1} - g'_n) \end{aligned} \tag{9.139}$$

The entire operation takes only two multiplications and four additions.

The B-spline interpolation is applied after the B-spline transformation. In the continuous cubic case this yields the effective transfer function using Eqs. (9.133) and (9.138),

$$\hat{\beta}_I(\tilde{k}) = \frac{\sin^4(\pi\tilde{k}/2)/(\pi\tilde{k}/2)^4}{(2/3 + 1/3\cos(\pi\tilde{k}))} \tag{9.140}$$

Essentially, the B-spline transformation performs an amplification of high wave numbers (at $\tilde{k} = 1$ by a factor 3), which compensates the smoothing of the B-spline interpolation to a large extent.

We investigate this compensation at both the grid points and the intermediate points. From the equation of the cubic B-spline interpolating kernel (Eq. (9.132); see also Fig. 9.18a) the interpolation coefficients for the grid points and intermediate grid points are

$$1/6\,[1\ 4\ 1] \quad \text{and} \quad 1/48\,[1\ 23\ 23\ 1] \tag{9.141}$$

respectively. Therefore, the transfer functions are

$$2/3 + 1/3\cos(\pi\tilde{k}) \quad \text{and} \quad 23/24\cos(\pi\tilde{k}/2) + 1/24\cos(3\pi\tilde{k}/2) \tag{9.142}$$

respectively. At the grid points, the transfer functions compensate exactly—as expected—the application of the B-spline transformation