# CHAPTER 7

# Tensor Product Spline Surfaces

Earlier we introduced parametric spline curves by simply using vectors of spline functions, defined over a common knot vector. In this chapter we introduce spline surfaces, but again the construction of *tensor product surfaces* is deeply dependent on spline functions. We first construct spline functions of two variables of the form $z = f(x, y)$, so called *explicit* spline surfaces, whose graph can be visualized as a surface in three dimensional space. We then pass to *parametric* surfaces in the same way that we passed from spline functions to spline curves.

The advantage of introducing tensor product surfaces is that all the approximation methods that we introduced in Chapter 5 generalize very easily as we shall see below. The methods also generalize nicely to parametric tensor product surfaces, but here we get the added complication of determining a suitable parametrization in the case where we are only given discrete data.

## 7.1 Explicit tensor product spline surfaces

The reader is undoubtedly familiar with polynomial surfaces of degree one and two. A linear surface

$$z = ax + by + c$$

represents a plane in 3-space. An example of a quadratic surface is the circular paraboloid

$$z = x^2 + y^2$$

shown in Figure 7.1 (a). The spline surfaces we will consider are made by gluing together polynomial "patches" like these.

### 7.1.1 Definition of the tensor product spline

For $x \in [0, 1]$ the line segment

$$b_0(1 - x) + b_1 x$$

connects the two values $b_0$ and $b_1$. Suppose $b_0(y)$ and $b_1(y)$ are two functions defined for $y$ in some interval $[c, d]$. Then for each $y \in [c, d]$ the function $b_0(y)(1 - x) + b_1(y)x$ is a line segment connecting $b_0(y)$ and $b_1(y)$. When $y$ varies we get a family of straight lines representing a surface

$$z = b_0(y)(1 - x) + b_1(y)x.$$

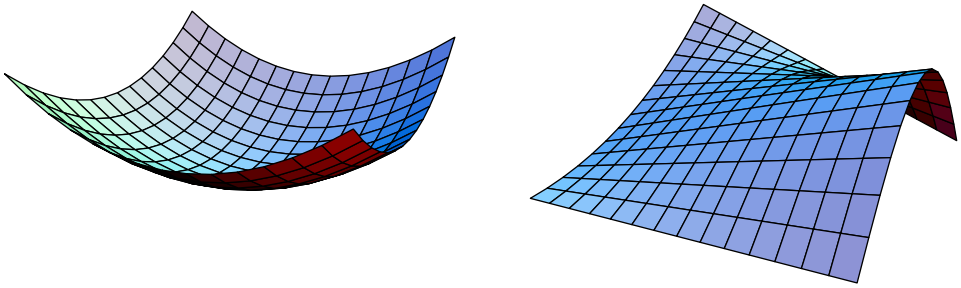
Such a "ruled" surface is shown in Figure 7.1 (b). Here we have chosen $b_0(y) = y^2$ and $b_1(y) = \sin(\pi y)$ for $y \in [0, 1]$.

An interesting case is obtained if we take $b_0$ and $b_1$ to be linear polynomials. Specifically, if

$$b_0(y) = c_{0,0}(1 - y) + c_{0,1}y, \quad \text{and} \quad b_1(y) = c_{1,0}(1 - y) + c_{1,1}y,$$

we obtain

$$f(x, y) = c_{0,0}(1 - x)(1 - y) + c_{0,1}(1 - x)y + c_{1,0}x(1 - y) + c_{1,1}xy,$$



(a) (b)

**Figure 7.1**. A piece of the circular paraboloid $z = x^2 + y^2$ is shown in (a), while the surface $(1-x)y^2 + x\sin(\pi y)$ is shown in (b).

for suitable coefficients $c_{i,j}$. In fact these coefficients are the values of $f$ at the corners of the unit square. This surface is ruled in both directions. For each fixed value of one variable we have a linear function in the other variable. We call $f$ a *bilinear polynomial*. Note that $f$ reduces to a quadratic polynomial along the diagonal line $x = y$.

We can use similar ideas to construct spline surfaces from families of spline functions. Suppose that for some integer $d$ and knot vector $\boldsymbol{\sigma}$ we have the spline space

$$\mathbb{S}_1 = \mathbb{S}_{d,\boldsymbol{\sigma}} = \text{span}\{\phi_1, \ldots, \phi_{n_1}\}.$$

To simplify the notation we have denoted the B-splines by $\{\phi_i\}_{i=1}^{n_1}$. Consider a spline in $\mathbb{S}_1$ with coefficients that are functions of $y$,

$$f(x,y) = \sum_{i=1}^{n_1} c_i(y)\phi_i(x). \tag{7.1}$$

For each value of $y$ we now have a spline in $\mathbb{S}_1$, and when $y$ varies we get a family of spline functions that each depends on $x$. Any choice of functions $c_i$ results in a surface, but a particularly useful construction is obtained if we choose the $c_i$ to be splines as well. Suppose we have another spline space of degree $\ell$ and with knots $\boldsymbol{\tau}$,

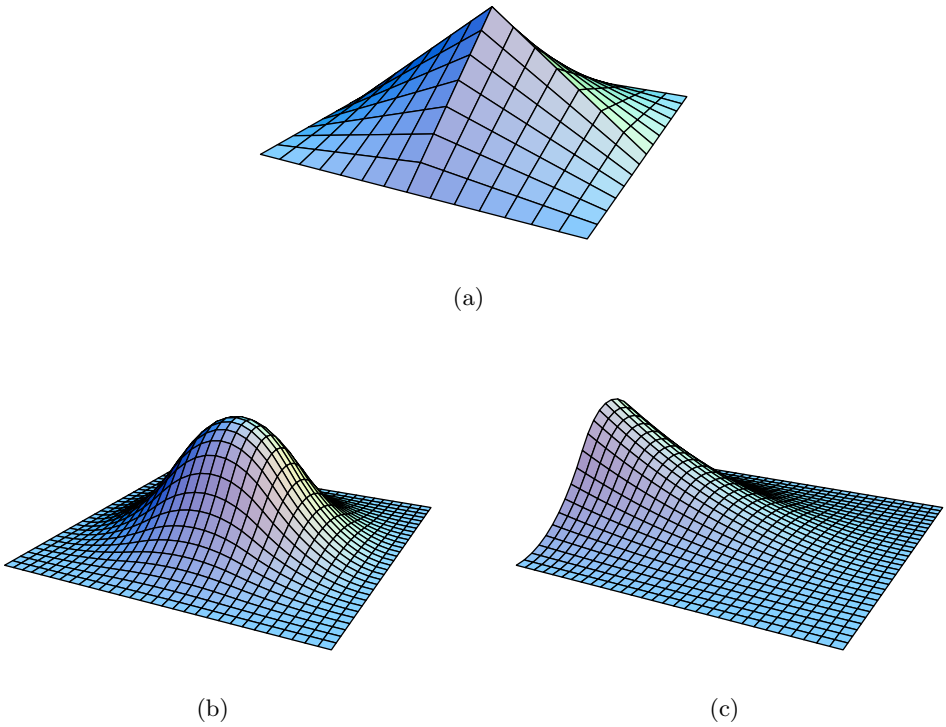
$$\mathbb{S}_2 = \mathbb{S}_{d_2,\boldsymbol{\tau}_2} = \text{span}\{\psi_1, \ldots, \psi_{n_2}\}$$

where $\{\psi_j\}_{j=1}^{n_2}$ denotes the B-spline basis in $\mathbb{S}_2$. If each coefficient function $c_i(y)$ is a spline in $\mathbb{S}_2$, then

$$c_i(y) = \sum_{j=1}^{n_2} c_{i,j}\psi_j(y) \tag{7.2}$$

for suitable numbers $(c_{i,j})_{i,j=1}^{n_1,n_2}$. Combining (7.1) and (7.2) we obtain

$$f(x,y) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} c_{i,j}\phi_i(x)\psi_j(y). \tag{7.3}$$



(a)

(b) (c)

**Figure 7.2**. A bilinear B-spline (a), a biquadratic B-spline (b) and biquadratic B-spline with a triple knot in one direction (c).

**Definition 7.1.** *The tensor product of the two spaces* $\mathbb{S}_1$ *and* $\mathbb{S}_2$ *is defined to be the family of all functions of the form*

$$f(x,y) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} c_{i,j} \phi_i(x) \psi_j(y),$$

*where the coefficients* $(c_{i,j})_{i,j=1}^{n_1,n_2}$ *can be any real numbers. This linear space of functions is denoted* $\mathbb{S}_1 \otimes \mathbb{S}_2$.

The space $\mathbb{S}_1 \otimes \mathbb{S}_2$ is spanned by the functions $\{\phi_i(x)\psi_j(y)\}_{i,j=1}^{n_1,n_2}$ and therefore has dimension $n_1 n_2$. Some examples of these basis functions are shown in Figure 7.2. In Figure 7.2 (a) we have $\phi = \psi = B(\cdot \,|\, 0, 1, 2)$. The resulting function is a bilinear polynomial in each of the four squares $[i, i+1) \times [j, j+1)$ for $i, j = 0, 1$. It has the shape of a curved pyramid with value one at the top. In Figure 7.2 (b) we show the result of taking $\phi = \psi = B(\cdot \,|\, 0, 1, 2, 3)$. This function is a biquadratic polynomial in each of the 9 squares $[i, i+1) \times [j, j+1)$ for $i, j = 0, 1, 2$. In Figure 7.2 (c) we have changed $\phi$ to $B(\cdot \,|\, 0, 0, 0, 1)$.

Tensor product surfaces are piecewise polynomials on rectangular domains. A typical example is shown in Figure 7.3. Each vertical line corresponds to a knot for the $\mathbb{S}_1$ space, and similarly, each horizontal line stems from a knot in the $\mathbb{S}_2$ space. The surface will usually have a discontinuity across the knot lines, and the magnitude of the discontinuity is inherited directly from the univariate spline spaces. For example, across a vertical knot

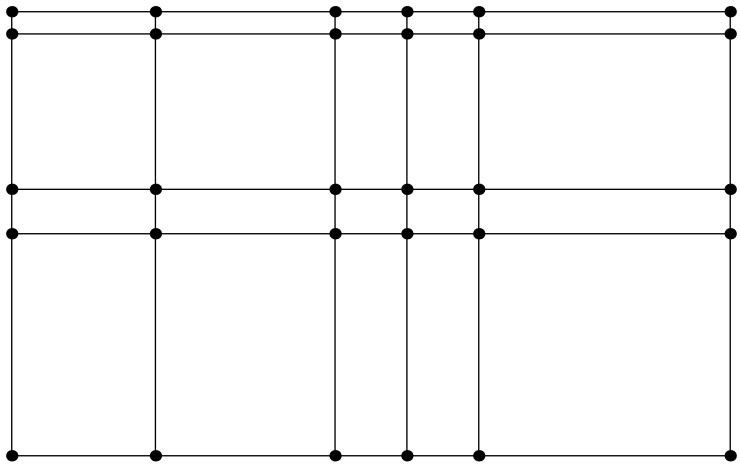


**Figure 7.3**. The knot lines for a tensor product spline surface.

line, partial derivatives with respect to $x$ have the continuity properties of the univariate spline functions in $\mathbb{S}_1$. This follows since the derivatives, say the first derivative, will involve sums of terms of the form

$$\frac{\partial}{\partial x}\left(c_{i,j}\phi_i(x)\psi_j(y)\right) = c_{i,j}\phi_i'(x)\psi_j(y).$$

A tensor product surface can be written conveniently in matrix-vector form. If $f(x,y)$ is given by (7.3) then

$$f(x,y) = \boldsymbol{\phi}(x)^T\boldsymbol{C}\boldsymbol{\psi}(y), \tag{7.4}$$

where

$$\boldsymbol{\phi} = (\phi_1,\ldots,\phi_{n_1})^T, \qquad \boldsymbol{\psi} = (\psi_1,\ldots,\psi_{n_2})^T,$$

and $\boldsymbol{C} = (c_{i,j})$ is the matrix of coefficients. This can be verified quite easily by expanding the multiplications.

### 7.1.2 Evaluation of tensor product spline surfaces

There are many ways to construct surfaces from two spaces of univariate functions, but the tensor product has one important advantage: many standard operations that we wish to perform with the surfaces are very simple generalisations of corresponding univariate operations. We will see several examples of this, but start by showing how to compute a point on a tensor product spline surface.

To compute a point on a tensor product spline surface, we can make use of the algorithms we have for computing points on spline functions. Suppose we want to compute $f(x,y) = \boldsymbol{\phi}(x)^T\boldsymbol{C}\boldsymbol{\psi}(y)^T$, and suppose for simplicity that the polynomial degree in the two directions are equal, so that $d = \ell$. If the integers $\mu$ and $\nu$ are such that $\sigma_\nu \le x < \sigma_{\nu+1}$ and $\tau_\mu \le y < \tau_{\mu+1}$, then we know that only $(\phi_i(x))_{i=\nu-d}^{\nu}$ and $(\psi_j(y))_{j=\mu-\ell}^{\mu}$ can be nonzero at $(x,y)$. To compute

$$f(x,y) = \boldsymbol{\phi}(x)^T\boldsymbol{C}\boldsymbol{\psi}(y) \tag{7.5}$$

we therefore first make use of Algorithm 2.21 to compute the $d+1$ nonzero B-splines at $x$ and the $\ell+1$ nonzero B-splines at $y$ with the triangular down algorithm. We can then

pick out that part of the coefficient matrix $\boldsymbol{C}$ which corresponds to these B-splines and multiply together the right-hand side of (7.5).

A pleasant feature of this algorithm is that its operation count is of the same order of magnitude as evaluation of univariate spline functions. If we assume, for simplicity, that $\ell = d$, we know that roughly $3(d+1)^2/2$ multiplications are required to compute the nonzero B-splines at $x$, and the same number of multiplications to compute the nonzero B-splines at $y$. To finish the computation of $f(x,y)$, we have to evaluate a product like that in (7.5), with $\boldsymbol{C}$ a $(d+1)\times(d+1)$-matrix and the two vectors of dimension $d+1$. This requires roughly $(d+1)^2$ multiplications, giving a total of $4(d+1)^2$ multiplications. The number of multiplications required to compute a point on a spline surface is therefore of the same order as the number of multiplications required to compute a point on a univariate spline function. The reason we can compute a point on a surface this quickly is the rather special structure of tensor products.

## 7.2 Approximation methods for tensor product splines

One of the main advantages of the tensor product definition of surfaces is that the approximation methods that we developed for functions and curves can be utilized directly for approximation of surfaces. In this section we consider some of the approximation methods in Chapter 5 and show how they can be generalized to surfaces.

### 7.2.1 The variation diminishing spline approximation

Consider first the variation diminishing approximation. Suppose $f$ is a function defined on a rectangle

$$\Omega = \Big\{(x,y) \mid a_1 \le x \le b_1 \ \&\ c \le y \le b_2\Big\} = [a_1,b_1]\times[a_2,b_2].$$

Let $\boldsymbol{\sigma} = (\sigma_i)_{i=1}^{n_1+d+1}$ be a $d+1$-regular knot vector with boundary knots $\sigma_d = a_1$ and $\sigma_{n_1} = b_1$, and let $\boldsymbol{\tau} = (\tau_j)_{j=1}^{n_2+\ell+1}$ be an $\ell+1$-regular knot vector with boundary knots $\tau_\ell = a_2$ and $\tau_{n_2} = b_2$. As above we let $\phi_i = B_{i,d,\boldsymbol{\sigma}}$ and $\psi_j = B_{j,\ell,\boldsymbol{\tau}}$ be the B-splines on $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ respectively. The spline

$$Vf(x,y) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} f(\sigma_i^*,\tau_j^*)\phi_i(x)\psi_j(y) \tag{7.6}$$

where

$$\begin{aligned}\sigma_i^* &= \sigma_{i,d}^* = (\sigma_{i+1}+\ldots+\sigma_{i+d})/d\\ \tau_j^* &= \tau_{j,\ell}^* = (\tau_{j+1}+\ldots+\tau_{j+\ell})/\ell,\end{aligned} \tag{7.7}$$

is called the *variation diminishing spline approximation on* $(\boldsymbol{\sigma},\boldsymbol{\tau})$ *of degree* $(d,\ell)$. If no interior knots in $\boldsymbol{\sigma}$ has multiplicity $d+1$ then

$$a_1 = \sigma_1^* < \sigma_2^* < \ldots < \sigma_{n_1}^* = b_1,$$

and similarly, if no interior knots in $\boldsymbol{\tau}$ has multiplicity $\ell+1$ then

$$a_2 = \tau_1^* < \tau_2^* < \ldots < \tau_{n_2}^* = b_2.$$

This means that the nodes $(\sigma_i^*,\tau_j^*)_{i,j=1}^{n_1,n_2}$ divides the domain $\Omega$ into a rectangular grid.

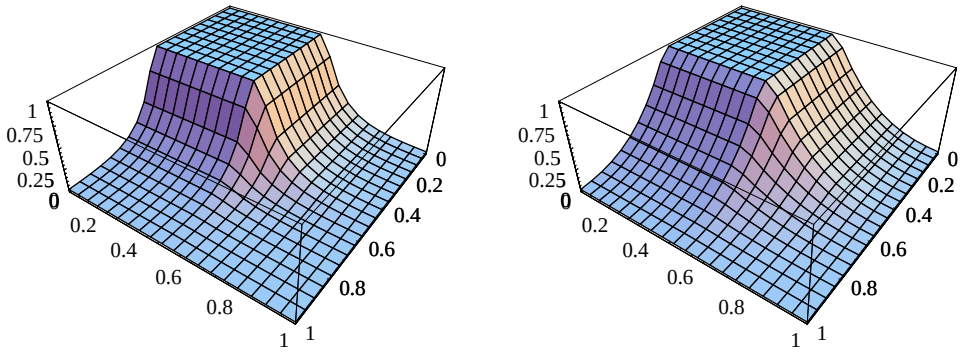



**Figure 7.4**. The function $f(x, y)$ given in Example 7.2 is shown in (a) and its variation diminishing spline approximation is shown in (b).

**Example 7.2.** Suppose we want to approximate the function

$$f(x, y) = g(x)g(y), \tag{7.8}$$

where

$$g(x) = \begin{cases} 1, & 0 \le x \le 1/2, \\ e^{-10(x-1/2)}, & 1/2 < x \le 1, \end{cases}$$

on the unit square

$$\Omega = \Big\{(x, y) \;\mid\; 0 \le x \le 1 \;\&\; 0 \le y \le 1\Big\} = [0, 1]^2.$$

A graph of this function is shown in Figure 7.4 (a), and we observe that $f$ has a flat spot on the square $[0, 1/2]^2$ and falls off exponentially on all sides. In order to approximate this function by a bicubic variation diminishing spline we observe that the surface is continuous, but that it has discontinuities partial derivatives across the lines $x = 1/2$ and $y = 1/2$. We obtain a tensor product spline space with similar continuity properties across these lines by making the value $1/2$ a knot of multiplicity 3 in $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$. For an integer $q$ with $q \ge 2$ we define the knot vectors by

$$\begin{aligned} \boldsymbol{\sigma} = \boldsymbol{\tau} = (0, 0, 0, 0, 1/(2q), \ldots, 1/2 - 1/(2q), 1/2, 1/2, 1/2, \\ 1/2 + 1/(2q), \ldots 1 - 1/(2q), 1, 1, 1, 1). \end{aligned}$$

The corresponding variation diminishing spline approximation is shown in Figure 7.4 (b) for $q = 2$.

The tensor product variation diminishing approximation $Vf$ has shape preserving properties analogous to those discussed in Section 5.4 for curves. In Figures 7.4 and **??** we observe that the constant part of $f$ in the region $[0, 1/2] \times [0, 1/2]$ is reproduced by $Vf$, and $Vf$ appears to have the same shape as $f$. These and similar properties can be verified formally, just like for functions.

### 7.2.2 Tensor Product Spline Interpolation

We consider interpolation at a set of gridded data

$$(x_i, y_j, f_{i,j})_{i=1,j=1}^{m_1,m_2}, \tag{7.9}$$

where

$$a_1 = x_1 < x_2 < \cdots < x_{m_1} = b_1, \qquad a_2 = y_1 < y_2 < \cdots < y_{m_2} = b_2.$$

For each $i, j$ we can think of $f_{i,j}$ as the value of an unknown function $f = f(x, y)$ at the point $(x_i, y_j)$. Note that these data are given on a grid of the same type as that of the knot lines in Figure 7.3.

We will describe a method to find a function $g = g(x, y)$ in a tensor product space $\mathbb{S}_1 \otimes \mathbb{S}_2$ such that

$$g(x_i, y_j) = f_{i,j}, \quad i = 1, \ldots, m_1, \quad j = 1, \ldots, m_2. \tag{7.10}$$

We think of $\mathbb{S}_1$ and $\mathbb{S}_2$ as two univariate spline spaces

$$\mathbb{S}_1 = \text{span}\{\phi_1, \ldots, \phi_{m_1}\}, \qquad \mathbb{S}_2 = \text{span}\{\psi_1, \ldots, \psi_{m_2}\}, \tag{7.11}$$

where the $\phi$'s and $\psi$'s are bases of B-splines for the two spaces. Here we have assumed that the dimension of $\mathbb{S}_1 \otimes \mathbb{S}_2$ agrees with the number of given data points since we want to approximate using interpolation. With $g$ in the form

$$g(x, y) = \sum_{p=1}^{m_1} \sum_{q=1}^{m_2} c_{p,q} \psi_q(y) \phi_p(x) \tag{7.12}$$

the interpolation conditions (7.10) lead to a set of equations

$$\sum_{p=1}^{m_1} \sum_{q=1}^{m_2} c_{p,q} \psi_q(y_j) \phi_p(x_i) = f_{i,j}, \quad \text{for all } i \text{ and } j.$$

This double sum can be split into two sets of simple sums

$$\sum_{p=1}^{m_1} d_{p,j} \phi_p(x_i) = f_{i,j}, \tag{7.13}$$

$$\sum_{q=1}^{m_2} c_{p,q} \psi_q(y_j) = d_{p,j}. \tag{7.14}$$

In order to study existence and uniqueness of solutions, it is convenient to have a matrix formulation of the equations for the $c_{p,q}$. We define the matrices

$$\begin{aligned} &\boldsymbol{\Phi} = (\phi_{i,p}) \in \mathbb{R}^{m_1, m_1}, \quad \phi_{i,p} = \phi_p(x_i), \\ &\boldsymbol{\Psi} = (\psi_{j,q}) \in \mathbb{R}^{m_2, m_2}, \quad \psi_{j,q} = \psi_q(y_j), \\ &\boldsymbol{D} = (d_{p,j}) \in \mathbb{R}^{m_1, m_2}, \ \boldsymbol{F} = (f_{i,j}) \in \mathbb{R}^{m_1, m_2}, \ \boldsymbol{C} = (c_{p,q}) \in \mathbb{R}^{m_1, m_2}. \end{aligned} \tag{7.15}$$

We then see that in (7.13) and (7.14)

$$\begin{aligned} \sum_{p=1}^{m_1} d_{p,j} \phi_p(x_i) &= \sum_{p=1}^{m_1} \phi_{i,p} d_{p,j} = (\boldsymbol{\Phi D})_{i,j} = (\boldsymbol{F})_{i,j}, \\ \sum_{q=1}^{m_2} c_{p,q} \psi_q(y_j) &= \sum_{q=1}^{m_2} \psi_{j,q} c_{p,q} = (\boldsymbol{\Psi C}^T)_{j,p} = (\boldsymbol{D}^T)_{j,p}. \end{aligned}$$

It follows that (7.13) and (7.14) can be written in the following matrix form

$$\boldsymbol{\Phi D} = \boldsymbol{F} \quad \text{and} \quad \boldsymbol{C \Psi}^T = \boldsymbol{D}. \tag{7.16}$$

From these equations we obtain the following proposition.

**Proposition 7.3.** *Suppose the matrices $\mathbf{\Phi}$ and $\mathbf{\Psi}$ are nonsingular. Then there is a unique $g \in S_1 \otimes S_2$ such that (7.10) holds. This $g$ is given by (7.12) where the coefficient matrix $\boldsymbol{C} = (c_{p,q})$ satisfies the matrix equation*

$$\mathbf{\Phi} \boldsymbol{C} \mathbf{\Psi}^T = \boldsymbol{F}.$$

**Proof.** The above derivation shows that there is a unique $g \in \mathbb{S}_1 \otimes \mathbb{S}_2$ such that (7.10) holds if and only if the matrix equations in (7.16) have unique solutions $\boldsymbol{D}$ and $\boldsymbol{C}$. But this is the case if and only if the matrices $\mathbf{\Phi}$ and $\mathbf{\Psi}$ are nonsingular. The final matrix equation is just the two equations in (7.16) combined. ∎

There is a geometric interpretation of the interpolation process. Let us define a family of $x$-curves by

$$X_j(x) = \sum_{p=1}^{m_1} d_{p,j} \phi_p(x), \quad j = 1, 2, \ldots, m_2.$$

Here the $d_{p,j}$ are taken from (7.13). Then for each $j$ we have

$$X_j(x_i) = f_{i,j}, \quad i = 1, 2, \ldots, m_1.$$

We see that $X_j$ is a curve which interpolates the data $\boldsymbol{f}_j = (f_{1,j}, \ldots, f_{m_1,j})$ at the $y$-level $y_j$. Moreover, by using (7.10) we see that for all $x$

$$X_j(x) = g(x, y_j), \quad j = 1, 2, \ldots, m_2.$$

This means that we can interpret (7.13) and (7.14) as follows:

(i) Interpolate in the $x$-direction by determining the curves $X_j$ interpolating the data $\boldsymbol{f}_j$.

(ii) Make a surface by filling in the space between these curves.

This process is obviously symmetric in $x$ and $y$. Instead of (7.13) and (7.14) we can use the systems

$$\sum_{q=1}^{m_2} e_{i,q} \psi_q(y_j) = f_{i,j}, \tag{7.17}$$

$$\sum_{p=1}^{m_1} c_{p,q} \phi_p(x_i) = e_{i,q}. \tag{7.18}$$

In other words we first make a family of $y$-curves $Y_i(y) = \sum_{q=1}^{m_2} e_{i,q} \psi_q(y)$ interpolating the row data vectors $F_i = (f_{i,1}, \ldots, f_{i,m_2})$. We then blend these curves to obtain the same surface $g(x, y)$.

The process we have just described is a special instance of a more general process which we is called *lofting*. By lofting we mean any process to construct a surface from a family of parallel curves. The word lofting originated in ship design. To draw a ship hull, the designer would first make parallel cross-sections of the hull. These curves were drawn in full size using mechanical splines. Then the cross-sections were combined into a surface

by using longitudinal curves. Convenient space for this activity was available at the loft of the shipyard.

We have seen that tensor product interpolation is a combination of univariate interpolation processes. We want to take a second look at this scheme. The underlying univariate interpolation process can be considered as a map converting the data $\boldsymbol{x}, \boldsymbol{f}$ into a spline interpolating this data. We can write such a map as

$$g = I[\boldsymbol{x}, \boldsymbol{f}] = \sum_{p=1}^{m_1} c_p \phi_p.$$

The coefficients $\boldsymbol{c} = (c_p)$ are determined from the interpolation requirements $g(x_i) = f_i$ for $i = 1, 2, \ldots, m_1$. We also have a related map $\tilde{I}$ which maps the data into the coefficients

$$\boldsymbol{c} = \tilde{I}[\boldsymbol{x}, \boldsymbol{f}].$$

Given $m_2$ data sets $(x_i, f_{i,j})_{i=1}^{m_1}$ for $j = 1, 2, \ldots, m_2$, we combine the function values into a matrix

$$\boldsymbol{F} = (\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n) = (f_{i,j}) \in \mathbb{R}^{m_1, m_2}$$

and define

$$\boldsymbol{C} = \tilde{I}[\boldsymbol{x}, \boldsymbol{F}] = (\tilde{I}[\boldsymbol{x}, \boldsymbol{f}_1], \ldots, \tilde{I}[\boldsymbol{x}, \boldsymbol{f}_n]). \tag{7.19}$$

With this notation the equations in (7.16) correspond to

$$\boldsymbol{D} = \tilde{I}_1[\boldsymbol{x}, \boldsymbol{F}], \quad \boldsymbol{C}^T = \tilde{I}_2[\boldsymbol{y}, \boldsymbol{D}^T],$$

where $\tilde{I}_1$ and $\tilde{I}_2$ are the univariate interpolation operators in the $x$ and $y$ directions, respectively. Combining these two equations we have

$$\boldsymbol{C} = (\tilde{I}_1 \otimes \tilde{I}_2)[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}] = \tilde{I}_2[\boldsymbol{y}, \tilde{I}_1[\boldsymbol{x}, \boldsymbol{F}]^T]^T. \tag{7.20}$$

We call $\tilde{I}_1 \otimes \tilde{I}_2$, defined in this way, for the tensor product of $\tilde{I}_1$ and $\tilde{I}_2$. We also define $(I_1 \otimes I_2)[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}]$ as the spline in $\mathbb{S}_1 \otimes \mathbb{S}_2$ with coefficients $(\tilde{I}_1 \otimes \tilde{I}_2)[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}]$.

These operators can be applied in any order. We can apply $I_1$ on each of the data vectors $\boldsymbol{f}_j$ to create the $X_j$ curves, and then use $I_2$ for the lofting. Or we could start by using $I_2$ to create $y$-curves $Y_i(y)$ and then loft in the $x$-direction using $I_1$. From this it is clear that

$$(\tilde{I}_1 \otimes \tilde{I}_2)[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}] = (\tilde{I}_2 \otimes \tilde{I}_1)[\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{F}^T].$$

Tensor product interpolation is quite easy to program on a computer. In order to implement the $\tilde{I}[\boldsymbol{x}, \boldsymbol{F}]$ operation we need to solve linear systems of the form given in (7.16). These systems have one coefficient matrix, but several right hand sides.

Two univariate programs can be combined easily and efficiently as in (7.20) provided we have a linear equation solver that can handle several right-hand sides simultaneously. Corresponding to the operator $I[\boldsymbol{x}, \boldsymbol{f}]$ we would have a program

$$I^P[\boldsymbol{x}, \boldsymbol{f}, d, \boldsymbol{\tau}, \boldsymbol{c}],$$

which to given data $\boldsymbol{x}$ and $\boldsymbol{f}$ will return a spline space represented by the degree $d$ and the knot vector $\boldsymbol{\tau}$, and the coefficients $\boldsymbol{c}$ of an interpolating spline curve in the spline space. Suppose we have two such programs $I_1^P$ and $I_2^P$ corresponding to interpolation in spline spaces $\mathbb{S}_1 = \mathbb{S}_{q,\boldsymbol{\sigma}}$ and $\mathbb{S}_2 = \mathbb{S}_{\ell,\boldsymbol{\tau}}$. Assuming that these programs can handle data of the form $\boldsymbol{x}, \boldsymbol{F}$, a program to carry out the process in (7.20) would be

1. $I_1^P[\boldsymbol{x}, \boldsymbol{F}, d, \boldsymbol{\sigma}, \boldsymbol{D}]$;
2. $I_2^P[\boldsymbol{y}, \boldsymbol{D}^T, \ell, \boldsymbol{\tau}, \boldsymbol{G}]$;
3. $\boldsymbol{C} = \boldsymbol{G}^T$;

### 7.2.3 Least Squares for Gridded Data

The least squares technique is a useful and important technique for fitting of curves and surfaces to data. In principle, it can be used for approximation of functions of any number of variables. Computationally there are several problems however, the main one being that usually a large linear system has to be solved. The situation is better when the data is gridded, say of the form (7.9). We study this important special case in this section and consider the following problem:

**Problem 7.4.** *Given data*

$$(x_i, y_j, f_{i,j})_{i=1,j=1}^{m_1,m_2},$$

*positive weights* $(w_i)_{i=1}^{m_1}$ *and* $(v_j)_{j=1}^{m_2}$*, and univariate spline spaces* $\mathbb{S}_1$ *and* $\mathbb{S}_2$*, find a spline surface* $g$ *in* $\mathbb{S}_1 \otimes \mathbb{S}_2$ *which solves the minimization problem*

$$\min_{g \in \mathbb{S}_1 \otimes \mathbb{S}_2} \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} w_i v_j \left[ g(x_i, y_j) - f_{i,j} \right]^2 .$$

We assume that the vectors of data abscissas $\boldsymbol{x} = (x_i)_{i=1}^{m_1}$ and $\boldsymbol{y} = (y_j)_{j=1}^{m_2}$ have distinct components, but that they do not need to be ordered. Note that we only have $m_1 + m_2$ independent weights. Since we have $m_1 \times m_2$ data points it would have been more natural to have $m_1 \times m_2$ weights, one for each data point. The reason for associating weights with grid lines instead of points is computational. As we will see, this assures that the problem splits into a sequence of univariate problems.

We assume that the spline spaces $\mathbb{S}_1$ and $\mathbb{S}_2$ are given in terms of B-splines

$$\mathbb{S}_1 = \operatorname{span}\{\phi_1, \ldots, \phi_{n_1}\}, \quad \mathbb{S}_2 = \operatorname{span}\{\psi_1, \ldots, \psi_{n_2}\},$$

and seek the function $g$ in the form

$$g(x, y) = \sum_{p=1}^{n_1} \sum_{q=1}^{n_2} c_{p,q} \psi_q(y) \phi_p(x).$$

Our goal in this section is to show that Problem 7.4 is related to the univariate least squares problem just as the interpolation problem in the last section was related to univariate interpolation. We start by giving a matrix formulation analogous to Lemma 5.21 for the univariate case.

**Lemma 7.5.** *Problem 7.4 is equivalent to the following matrix problem*

$$\min_{\boldsymbol{C} \in \mathbb{R}^{n_1,n_2}} \|\boldsymbol{A}\boldsymbol{C}\boldsymbol{B}^T - \boldsymbol{G}\|^2, \tag{7.21}$$

*where*

$$\begin{aligned} &\boldsymbol{A} = (a_{i,p}) \in \mathbb{R}^{m_1,n_1}, \quad a_{i,p} = \sqrt{w_i}\phi_p(x_i), \\ &\boldsymbol{B} = (b_{j,q}) \in \mathbb{R}^{m_2,n_2}, \quad b_{j,q} = \sqrt{v_j}\psi_q(y_j), \\ &\boldsymbol{G} = (\sqrt{w_i}\sqrt{v_j} f_{i,j}) \in \mathbb{R}^{m_1,m_2}, \ \boldsymbol{C} = (c_{p,q}) \in \mathbb{R}^{n_1,n_2}. \end{aligned} \tag{7.22}$$

*Here, the norm $\|\cdot\|$ is the Frobenius norm,*

$$\|\boldsymbol{E}\| = \Big(\sum_{i=1}^{m}\sum_{j=1}^{n}|e_{i,j}|^2\Big)^{1/2} \tag{7.23}$$

*for any rectangular $m \times n$ matrix $\boldsymbol{E} = (e_{i,j})$.*

**Proof.** Suppose $\boldsymbol{C} = (c_{p,q})$ are the B-spline coefficients of some $g \in \mathbb{S}_1 \otimes \mathbb{S}_2$. Then

$$\begin{aligned}
\|\boldsymbol{A}\boldsymbol{C}\boldsymbol{B}^T - \boldsymbol{G}\|^2 &= \sum_{i=1}^{m_1}\sum_{j=1}^{m_2}\Big(\sum_{p=1}^{n_1}\sum_{q=1}^{n_2} a_{i,p}c_{p,q}b_{j,q} - g_{i,j}\Big)^2 \\
&= \sum_{i=1}^{m_1}\sum_{j=1}^{m_2}\Big(\sum_{p=1}^{n_1}\sum_{q=1}^{n_2} \sqrt{w_i}\phi_p(x_i)c_{p,q}\sqrt{v_j}\psi_q(y_j) - \sqrt{w_i}\sqrt{v_j}f_{i,j}\Big)^2 \\
&= \sum_{i=1}^{m_1}\sum_{j=1}^{m_2} w_i v_j \left[g(x_i,y_j) - f_{i,j}\right]^2.
\end{aligned}$$

This shows that the two minimization problems are equivalent. ∎

We next state some basic facts about the matrix problem (7.21).

**Proposition 7.6.** *The problem (7.21) always has a solution $\boldsymbol{C} = \boldsymbol{C}^*$, and the solution is unique if and only if both matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ have linearly independent columns. The solution $\boldsymbol{C}^*$ can be found by solving the matrix equation*

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{C}^*\boldsymbol{B}^T\boldsymbol{B} = \boldsymbol{A}^T\boldsymbol{G}\boldsymbol{B}. \tag{7.24}$$

**Proof.** By arranging the entries of $\boldsymbol{C}$ in a one dimensional vector it can be seen that the minimization problem (7.21) is a linear least squares problem. The existence of a solution then follows from Lemma 5.22. For the rest of the proof we introduce some additional notation. For matrices $\boldsymbol{H} = (h_{i,j})$ and $\boldsymbol{K} = (k_{i,j})$ in $\mathbb{R}^{m,n}$ we define the scalar product

$$(\boldsymbol{H},\boldsymbol{K}) = \sum_{i=1}^{m}\sum_{j=1}^{n} h_{i,j}q_{i,j}.$$

This is a scalar product of the matrices $\boldsymbol{H}$ and $\boldsymbol{K}$ regarded as vectors. We have $(\boldsymbol{H},\boldsymbol{H}) = \|\boldsymbol{H}\|^2$, the Frobenius norm of $\boldsymbol{H}$, squared. We also observe that for any $m \times n$ matrices $\boldsymbol{H}$ and $\boldsymbol{K}$, we have

$$\|\boldsymbol{H}+\boldsymbol{K}\|^2 = \|\boldsymbol{H}\|^2 + 2(\boldsymbol{H},\boldsymbol{K}) + \|\boldsymbol{K}\|^2.$$

Moreover,

$$(\boldsymbol{E},\boldsymbol{H}\boldsymbol{K}) = (\boldsymbol{H}^T\boldsymbol{E},\boldsymbol{K}) = (\boldsymbol{E}\boldsymbol{K}^T,\boldsymbol{H}), \tag{7.25}$$

for any matrices $\boldsymbol{E},\boldsymbol{H},\boldsymbol{K}$ such that the matrix operations make sense. For any $\boldsymbol{C} \in \mathbb{R}^{n_1,n_2}$ we let

$$q(\boldsymbol{C}) = \|\boldsymbol{A}\boldsymbol{C}\boldsymbol{B}^T - \boldsymbol{G}\|^2.$$

This is the function we want to minimize. Suppose $\boldsymbol{C}^*$ is the solution of (7.24). We want to show that $q(\boldsymbol{C}^* + \epsilon \boldsymbol{D}) \geq q(\boldsymbol{C}^*)$ for any real $\epsilon$ and any $\boldsymbol{D} \in \mathbb{R}^{n_1 \times n_2}$. This will follow from the relation

$$q(\boldsymbol{C}^* + \epsilon \boldsymbol{D}) = q(\boldsymbol{C}^*) + 2\epsilon(\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{C}^*\boldsymbol{B}^T\boldsymbol{B} - \boldsymbol{A}^T\boldsymbol{G}\boldsymbol{B}, \boldsymbol{D}) + \epsilon^2\|\boldsymbol{A}\boldsymbol{D}\boldsymbol{B}^T\|^2. \tag{7.26}$$

For if $\boldsymbol{C}^*$ satisfies (7.24) then the complicated middle term vanishes and

$$q(\boldsymbol{C}^* + \epsilon \boldsymbol{D}) = q(\boldsymbol{C}^*) + \epsilon^2\|\boldsymbol{A}\boldsymbol{D}\boldsymbol{B}^T\|^2 \geq q(\boldsymbol{C}^*).$$

To establish (7.26) we have to expand $q(\boldsymbol{C}^* + \epsilon \boldsymbol{D})$,

$$\begin{aligned} q(\boldsymbol{C}^* + \epsilon \boldsymbol{D}) &= \|(\boldsymbol{A}\boldsymbol{C}^*\boldsymbol{B}^T - \boldsymbol{G}) + \epsilon\boldsymbol{A}\boldsymbol{D}\boldsymbol{B}^T\|^2 \\ &= q(\boldsymbol{C}^*) + 2\epsilon(\boldsymbol{A}\boldsymbol{C}^*\boldsymbol{B}^T - \boldsymbol{G}, \boldsymbol{A}\boldsymbol{D}\boldsymbol{B}^T) + \epsilon^2\|\boldsymbol{A}\boldsymbol{D}\boldsymbol{B}^T\|^2. \end{aligned}$$

Using (7.25) on the middle term, we can move $\boldsymbol{A}$ and $\boldsymbol{B}^T$ to the left-hand side of the inner product form, and we obtain (7.26). The uniqueness is left as a problem.

Conversely, suppose that $\boldsymbol{C}$ does not satisfy (7.24). We need to show that $\boldsymbol{C}$ does not minimize $q$. Now, for at least one matrix component $i, j$ we have

$$z = (\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{B}^T\boldsymbol{B} - \boldsymbol{A}^T\boldsymbol{G}\boldsymbol{B})_{i,j} \neq 0.$$

We choose $\boldsymbol{D}$ as the matrix where the $i, j$ element is equal to 1 and all other entries are 0. Then (7.26) takes the form

$$q(\boldsymbol{C} + \epsilon \boldsymbol{D}) = q(\boldsymbol{C}) + 2\epsilon z + \epsilon^2\|\boldsymbol{A}\boldsymbol{D}\boldsymbol{B}^T\|^2,$$

and this implies that $q(\boldsymbol{C} + \epsilon \boldsymbol{D}) < q(\boldsymbol{C})$ for $\epsilon z < 0$ and $|\epsilon|$ sufficiently small. But then $\boldsymbol{C}$ cannot minimize $q$. ∎

In order to find the solution of Problem 7.4, we have to solve the matrix equation (7.24). We can do this in two steps:

1. Find $\boldsymbol{D}$ from the system $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{D} = \boldsymbol{A}^T\boldsymbol{G}$.
2. Find $\boldsymbol{C}$ from the system $\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{C}^T = \boldsymbol{B}^T\boldsymbol{D}^T$.

The matrix $\boldsymbol{C}$ is then the solution of (7.24). The first step is equivalent to

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{d}_j = \boldsymbol{A}^T\boldsymbol{g}_j, \quad j = 1, 2, \ldots, m_2,$$

where $\boldsymbol{D} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_{m_2})$ and $\boldsymbol{G} = (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{m_2})$. This means that we need to solve $m_2$ linear least squares problems

$$\min \|\boldsymbol{A}\boldsymbol{d}_j - \boldsymbol{g}_j\|_2^2, \quad j = 1, 2, \ldots, m_2.$$

We then obtain a family of $x$-curves

$$X_j(x) = \sum_{p=1}^{n_1} d_{p,j}\phi_p(x).$$

In the second step we solve $n_1$ linear least squares problems of the form

$$\min \|\boldsymbol{B}\boldsymbol{h}_i - \boldsymbol{e}_i\|_2^2, \quad i = 1, 2, \ldots, n_1,$$

where the $\boldsymbol{e}_i$ are the rows of $\boldsymbol{D}$, and the $\boldsymbol{h}_i$ are the rows of $\boldsymbol{C}$

$$\boldsymbol{D} = \begin{pmatrix} \boldsymbol{e}_1^T \\ \vdots \\ \boldsymbol{e}_{n_1}^T \end{pmatrix}, \quad \boldsymbol{C} = \begin{pmatrix} \boldsymbol{h}_1^T \\ \vdots \\ \boldsymbol{h}_{n_1}^T \end{pmatrix}.$$

Alternatively we can do the computation by first performing a least squares approximation in the $y$-direction by constructing a family of $y$-curves, and then use least squares in the $x$-direction for the lofting. The result will be the same as before. To minimize the number of arithmetic operations one should start with the direction corresponding to the largest of the integers $m_1$ and $m_2$.

Corresponding to Problem 7.4 we have the univariate least squares problem defined in Problem 5.20. Associated with this problem we have an operator $L[\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{f}]$ which to given univariate data $\boldsymbol{x} = (x_i)_{i=1}^m$ and $\boldsymbol{f} = (f_i)_{i=1}^m$, and positive weights $\boldsymbol{w} = (w_i)_{i=1}^m$, assigns a spline

$$g = L[\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{f}] = \sum_{p=1}^{n} c_p \phi_p,$$

in a spline space $\mathbb{S} = \text{span}\{\phi_1, \ldots, \phi_n\}$. We also have the operator $\tilde{L}[\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{f}]$ which maps the data into the B-spline coefficients and is defined analagously to (7.19). With $\tilde{L}_1$ and $\tilde{L}_2$ being least squares operators in the $x$ and $y$ direction, respectively, the B-spline coefficients of the solution of Problem 7.4 can now be written

$$\boldsymbol{C} = (\tilde{L}_1 \otimes \tilde{L}_2)[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}, \boldsymbol{w}, \boldsymbol{v}] = \tilde{L}_2[\boldsymbol{y}, \boldsymbol{v}, \tilde{L}_1[\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{F}]^T]^T, \tag{7.27}$$

in analogy with the interpolation process (7.20).

## 7.3 General tensor product methods

In the previous sections we saw how univariate approximation schemes could be combined into a surface scheme for gridded data. Examples of this process is given by (7.20) and (7.27). This technique can in principle be applied quite freely. We could for example combine least squares in the $x$ direction with cubic spline interpolation in the $y$ direction. If $\tilde{Q}_1[\boldsymbol{x}, \boldsymbol{f}]$ and $\tilde{Q}_2[\boldsymbol{y}, \boldsymbol{g}]$ define univariate approximation methods then we define their tensor product as

$$(\tilde{Q}_1 \otimes \tilde{Q}_2)[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}] = \tilde{Q}_2[\boldsymbol{y}, \tilde{Q}_1[\boldsymbol{x}, \boldsymbol{F}]^T]^T. \tag{7.28}$$

In this section we want to show that

$$(\tilde{Q}_1 \otimes \tilde{Q}_2)[x, y, \boldsymbol{F}] = (\tilde{Q}_2 \otimes \tilde{Q}_1)[y, x, \boldsymbol{F}^T]$$

for a large class of operators $Q_1$ and $Q_2$. Thus, for such operators we are free to use $Q_2$ in the $y$-direction first and then $Q_1$ in the $x$-direction, or vice-versa.

We need to specify more abstractly the class of approximation schemes we consider. Suppose $Q[\boldsymbol{x}, \boldsymbol{f}]$ is a univariate approximation operator mapping the data into a spline in a univariate spline space

$$\mathbb{S} = \operatorname{span}\{\phi_1, \ldots, \phi_n\}.$$

Thus

$$Q[\boldsymbol{x}, \boldsymbol{f}] = \sum_{p=1}^{n} a_p(\boldsymbol{f})\phi_p(x). \tag{7.29}$$

The coefficients $a_p(\boldsymbol{f})$ of the spline are functions of both $\boldsymbol{x}$ and $\boldsymbol{f}$, but here we are mostly interested in the dependence of $\boldsymbol{f}$. We also let $(a_p(\boldsymbol{f})) = \tilde{Q}[\boldsymbol{x}, \boldsymbol{f}]$ be the coefficients of $Q[\boldsymbol{x}, \boldsymbol{f}]$. We are interested in the following class of operators $Q$.

**Definition 7.7.** *The operator $Q : \mathbb{R}^m \to \mathbb{S}$ given by (7.29) is linear if*

$$a_p(\boldsymbol{f}) = \sum_{i=1}^{m} a_{p,i} f_i, \tag{7.30}$$

*for suitable numbers $a_{p,i}$ independent of $\boldsymbol{f}$.*

If $Q$ is linear then

$$Q[\boldsymbol{x}, \alpha\boldsymbol{g} + \beta\boldsymbol{h}] = \alpha Q[\boldsymbol{x}, \boldsymbol{g}] + \beta Q[\boldsymbol{x}, \boldsymbol{h}]$$

for all $\alpha, \beta \in \mathbb{R}$ and all $\boldsymbol{g}, \boldsymbol{h} \in \mathbb{R}^m$.

**Example 7.8.** All methods in Chapter 5 are linear approximation schemes.

1. For the Schoenberg Variation Diminishing Spline Approximation we have $\boldsymbol{f} = (f_1, \ldots, f_m) = (f(\tau_1^*), \ldots, f(\tau_m^*))$. Thus $Vf$ is of the form (7.29) with $a_p(\boldsymbol{f}) = f_p$, and $a_{p,i} = \delta_{p,i}$.

2. All the interpolation schemes in Chapter 5, like cubic Hermite, and cubic spline with various boundary conditions are linear. This follows since the coefficients $\boldsymbol{c} = (c_p)$ are found by solving a linear system $\boldsymbol{\Phi c} = \boldsymbol{f}$. Thus $\boldsymbol{c} = \boldsymbol{\Phi}^{-1}\boldsymbol{f}$, and $c_p$ is of the form (7.30) with $a_{p,i}$ being the $(p, i)$-element of $\boldsymbol{\Phi}^{-1}$. For cubic Hermite interpolation we also have the explicit formulas in Proposition 5.5.

3. The least squares approximation method is also a linear approximation scheme. Recall that $Q$ in this case is constructed from the solution of the minimization problem

$$\min_{\boldsymbol{c}} \sum_{i=1}^{m} w_i \left[ \sum_{p=1}^{n} c_p \phi_p(x_i) - f_i \right]^2.$$

The vector $\boldsymbol{c}$ is determined as the solution of a linear system

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{c} = \boldsymbol{A}^T\boldsymbol{f}.$$

Thus $a_{p,i}$ is the $(p, i)$-element of the matrix $(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$.

Consider now the surface situation. Suppose we are given a set of gridded data and two univariate approximation operators $Q_1$ and $Q_2$, and associated with these operators we have the coefficient operators $\tilde{Q}_1$ and $\tilde{Q}_2$ assigning the coefficient vectors to the data.

**Proposition 7.9.** *Suppose $Q_1$ and $Q_2$ are linear operators of the form given by (7.29). Then for all data*

$$(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}) = (x_i, y_j, f_{i,j})_{i=1,j=1}^{m_1,m_2}, \tag{7.31}$$

*we have*

$$(\tilde{Q}_1 \otimes \tilde{Q}_2)[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}] = (\tilde{Q}_2 \otimes \tilde{Q}_1)[\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{F}^T].$$

**Proof.** To see this we go through the constructions in detail. Suppose that

$$Q_1[\boldsymbol{x}, \boldsymbol{f}] = \sum_{p=1}^{n_1} a_p(\boldsymbol{f})\phi_p, \quad a_p(\boldsymbol{f}) = \sum_{i=1}^{m_1} a_{p,i} f_i,$$

$$Q_2[\boldsymbol{y}, \boldsymbol{g}] = \sum_{q=1}^{n_2} b_p(\boldsymbol{g})\psi_p, \quad b_q(\boldsymbol{g}) = \sum_{j=1}^{m_2} b_{q,j} g_j.$$

The matrix $\boldsymbol{F} = (f_{i,j})) \in \mathbb{R}^{m_1,m_2}$ can be partitioned either by rows or by columns.

$$\boldsymbol{F} = (\boldsymbol{f}_1, \ldots, \boldsymbol{f}_{m_2}) = \begin{pmatrix} \boldsymbol{g}_1 \\ \vdots \\ \boldsymbol{g}_{m_1} \end{pmatrix}.$$

If we use $Q_1$ first then we obtain a family of $x$-curves from the columns $\boldsymbol{f}_j$ of the data $\boldsymbol{F}$

$$Q_1[\boldsymbol{x}, \boldsymbol{f}_j] = \sum_{p=1}^{n_1} a_p(\boldsymbol{f}_j)\phi_p(x), \quad j = 1, 2, \ldots, m_2.$$

From these curves we get the final surface

$$g(x, y) = \sum_{p=1}^{n_1}\sum_{q=1}^{n_2} c_{p,q}\psi_q(y)\phi_p(x),$$

where

$$c_{p,q} = b_q\left(a_p(\boldsymbol{f}_1), \ldots, a_p(\boldsymbol{f}_{m_2})\right).$$

Using the linearity we obtain

$$c_{p,q} = \sum_{j=1}^{m_2} b_{q,j} a_p(\boldsymbol{f}_j) = \sum_{j=1}^{m_2}\sum_{i=1}^{m_1} b_{q,j} a_{p,i} f_{i,j}. \tag{7.32}$$

Suppose now we use $Q_2$ first and then $Q_1$. We then obtain a surface

$$h(x, y) = \sum_{q=1}^{n_2}\sum_{p=1}^{n_1} d_{p,q}\psi_q(y)\phi_p(x),$$

where

$$d_{p,q} = a_p\left(b_q(\boldsymbol{g}_1), \ldots, b_q(\boldsymbol{g}_{m_1})\right).$$

Thus,

$$d_{p,q} = \sum_{i=1}^{m_1} a_{p,i} b_q(\boldsymbol{g}_i) = \sum_{i=1}^{m_1}\sum_{j=1}^{m_2} a_{p,i} b_{q,j} f_{i,j}.$$

Comparing this with (7.32) we see that $d_{p,q} = c_{p,q}$ for all integers $p$ and $q$, and hence $g = h$. We conclude that we end up with the same surface in both cases. ∎

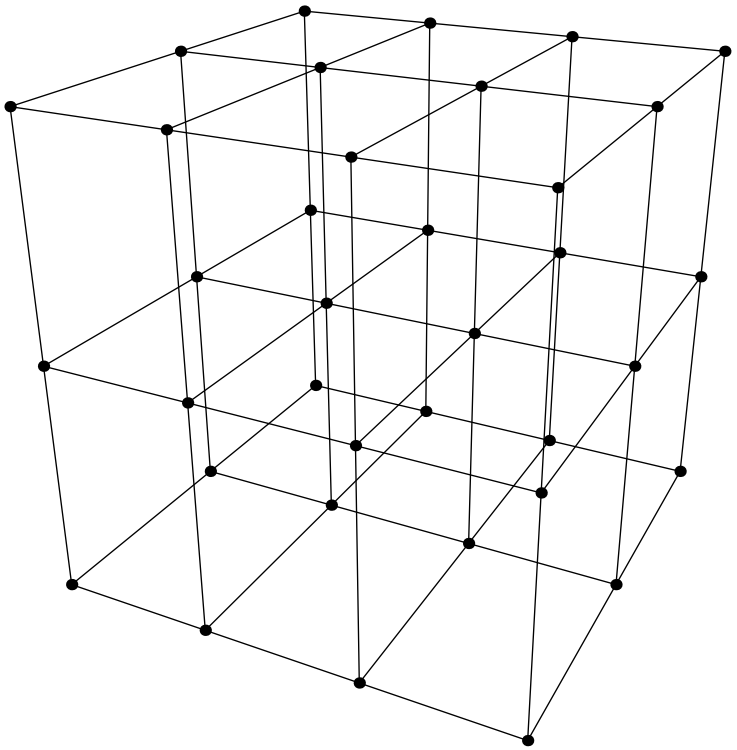


**Figure 7.5**. A cubical gridded region in space.

## 7.4 Trivariate Tensor Product Methods

The tensor product construction can be extended to higher dimensions. For trivariate approximation we can combine three univariate approximation schemes into a method to approximate trivariate data

$$(x_i, y_j, z_k, f_{i,j,k})_{i=1,j=1,k=1}^{m_1,\ m_2,\ m_3}. \tag{7.33}$$

Here the $f$'s are function values of an unknown trivariate function

$$f = f(x, y, z).$$

The data is given on a cubical region determined from the grid points $(x_i, y_j, z_k)$ in space. We write

$$\boldsymbol{F} = (f_{i,j,k}) \in \mathbb{R}^{m_1,m_2,m_3}$$

to indicate that the data can be thought of as sitting in a cube of dimensions $m_1, m_2, m_3$. Such a cubical grid is shown in Figure 7.5.

The approximation we seek have the form

$$g(x, y, z) = \sum_{p=1}^{n_1} \sum_{q=1}^{n_2} \sum_{r=1}^{n_3} c_{p,q,r} \omega_r(z) \psi_q(y) \phi_p(x). \tag{7.34}$$

Here

$$\mathbb{S}_1 = \text{span}\{\phi_1, \ldots, \phi_{n_1}\}, \quad \mathbb{S}_2 = \text{span}\{\psi_1, \ldots, \psi_{n_2}\}, \quad \mathbb{S}_3 = \text{span}\{\omega_1, \ldots, \omega_{n_3}\},$$

are three univariate spline spaces spanned by some B-splines. We can construct $g$ by forming a a sequence of simpler sums as follows

$$\begin{aligned} g(x,y,z) &= \sum_{p=1}^{n_1} d_p(y,z)\phi_p(x), \\ d_p(y,z) &= \sum_{q=1}^{n_2} e_{p,q}(z)\psi_q(y), \\ e_{p,q}(z) &= \sum_{r=1}^{n_3} c_{p,q,r}\omega_r(z). \end{aligned} \tag{7.35}$$

In order to interpolate the data given by (7.33) we obtain the following set of equations

$$\begin{aligned} \sum_{p=1}^{n_1} d_p(y_j,z_k)\phi_p(x_i) &= f_{i,j,k}, \quad i = 1,2,\ldots,m_1, \\ \sum_{q=1}^{n_2} e_{p,q}(z_k)\psi_q(y_j) &= d_p(y_j,z_k), \quad j = 1,2,\ldots,m_2, \\ \sum_{r=1}^{n_3} c_{p,q,r}\omega_r(z_k) &= e_{p,q}(z_k). \quad k = 1,2,\ldots,m_3, \end{aligned} \tag{7.36}$$

These are square systems if $n_i = m_i$, and have to be solved in the least squares sence if $m_i > n_i$ for one or more $i$.

Consider now writing these systems in matrix form. The equations involve arrays with 3 subscripts. For a positive integer $s$ we define a rank $s$ tensor to be a $s$-dimensional table of the form

$$\boldsymbol{A} = (a_{i_1,i_2,\ldots,i_s})_{i_1=1,i_2=1,\ldots,i_s=1}^{m_1,\ m_2,\ \ldots\ ,m_s}.$$

We write

$$\boldsymbol{A} \in \mathbb{R}^{m_1,m_2,\ldots,m_s} = \mathbb{R}^{\boldsymbol{m}},$$

for membership in the class of all rank $s$ tensors with real elements. These *tensors* are generalizations of ordinary vectors and matrices. A rank $s$ tensor can be arranged in a $s$-dimensional cuboidal array. This is the usual rectangular array for $s = 2$ and a rectangular parallelepiped for $s = 3$.

The operations of addition and scalar multiplication for vectors and matrices extend easily to tensors. The product of two tensors, say $\boldsymbol{A} \in \mathbb{R}^{m_1,m_2,\ldots,m_s}$ and $\boldsymbol{B} \in \mathbb{R}^{n_1,n_2,\ldots,n_e}$ can be defined if the last dimension of $\boldsymbol{A}$ equals the first dimension of $\boldsymbol{B}$. Indeed, with $m = m_s = n_1$, we define the product $\boldsymbol{AB}$ as the tensor

$$\boldsymbol{C} = \boldsymbol{AB} \in \mathbb{R}^{m_1,m_2,\ldots,m_{s-1},n_2,\ldots,n_s}$$

with elements

$$c_{i_1,\ldots,i_{s-1},j_2,\ldots,j_e} = \sum_{i=1}^{m} a_{i_1,\ldots,i_{s-1},i} b_{i,j_2,\ldots,j_e}.$$

For $s = e = 2$ this is the usual product of two matrices, while for $s = e = 1$ we have the inner product of vectors. In general this 'inner product' of tensors is a tensor of rank $s+e-2$. We just contract the last index of $\boldsymbol{A}$ and the first index of $\boldsymbol{B}$. Another productis known as the outer product.

Let us now write the equations in (7.36) in tensor form. The first equation can be written

$$\boldsymbol{\Phi D} = \boldsymbol{F}. \tag{7.37}$$

Here

$$\begin{aligned}
&\boldsymbol{\Phi} = (\phi_{i,p}) = (\phi_p(x_i)) \in \mathbb{R}^{m_1,n_1},\\
&\boldsymbol{D} = (d_{p,j,k}) = d_p(y_j, z_k) \in \mathbb{R}^{n_1,m_2,m_3}, \quad \boldsymbol{F} = (f_{i,j,k}) \in \mathbb{R}^{m_1,m_2,m_3}.
\end{aligned}$$

The system (7.37) is similar to the systems we had earlier for bivariate approximation. We have the same kind of coefficient matrix, but many more right-hand sides.

For the next equation in (7.36) we define

$$\begin{aligned}
&\boldsymbol{\Psi} = (\psi_{j,q}) = (\psi_q(y_j)) \in \mathbb{R}^{m_2,n_2},\\
&\boldsymbol{E} = (e_{q,k,p}) = (e_{p,q}(z_k)) \in \mathbb{R}^{n_2,m_3,n_1}, \quad \boldsymbol{D'} = (d_{j,k,p}) \in \mathbb{R}^{m_2,m_3,n_1}.
\end{aligned}$$

The next equation can then be written

$$\boldsymbol{\Psi E} = \boldsymbol{D'}. \tag{7.38}$$

The construction of $\boldsymbol{D'}$ from $\boldsymbol{D}$ involves a cyclic rotation of the dimensions from $(n_1, m_2, m_3)$ to $(m_2, m_3, n_1)$. The same operation is applied to $\boldsymbol{E}$ for the last equation in (7.36). We obtain

$$\boldsymbol{\Omega G} = \boldsymbol{E'}, \tag{7.39}$$

where

$$\begin{aligned}
&\boldsymbol{\Omega} = (\omega_{k,r}) = (\omega_r(z_k)) \in \mathbb{R}^{m_3,n_3},\\
&\boldsymbol{E'} = (e_{k,p,q}) = (e_{p,q}(z_k)) \in \mathbb{R}^{m_3,n_1,n_2}, \quad \boldsymbol{G} = (g_{r,p,q}) \in \mathbb{R}^{n_3,n_1,n_2}.
\end{aligned}$$

The coefficients $\boldsymbol{C'}$ are obtained by a final cyclic rotation of the dimensions

$$\boldsymbol{C} = \boldsymbol{G'}. \tag{7.40}$$

The systems (7.37), (7.38), and (7.39) corresponds to three univariate operators of the form $Q[\boldsymbol{x}, \boldsymbol{f}]$. We denote these $Q_1, Q_2$, and $Q_3$. We assume that $Q_i$ can be applied to a tensor. The tensor product of these three operators can now be defined as follows

$$(Q_1 \otimes Q_2 \otimes Q_3)[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{F}] = Q_3[\boldsymbol{z}, Q_2[\boldsymbol{y}, Q_1[\boldsymbol{x}, F]']']'. \tag{7.41}$$

The actual implementation of this scheme on a computer will depend on how arrays are sorted in the actual programming language used. Some languages arrange by columns, while others arrange by rows.

## 7.5 Parametric Surfaces

Parametric curves and explicit surfaces have a natural generalization to parametric surfaces. Let us consider the plane $P$ through three points in space which we call $\boldsymbol{p}_0$, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$. We define the function $\boldsymbol{f} : \mathbb{R}^2 \mapsto P$ by

$$\boldsymbol{f}(u, v) = \boldsymbol{p}_0 + (\boldsymbol{p}_1 - \boldsymbol{p}_0)u + (\boldsymbol{p}_2 - \boldsymbol{p}_0)v. \tag{7.42}$$

We see that $\boldsymbol{f}(0, 0) = \boldsymbol{p}_0$, while $\boldsymbol{f}(1, 0) = \boldsymbol{p}_1$ and $\boldsymbol{f}(0, 1) = \boldsymbol{p}_2$, so that $\boldsymbol{f}$ interpolates the three points. Since $\boldsymbol{f}$ is also a linear function, we conclude that it is indeed a representation for the plane $P$.

We start by generalizing and formalizing this.

**Definition 7.10.** *A parametric representation of class $C^m$ of a set $S \subseteq \mathbb{R}^3$ is a mapping $\boldsymbol{f}$ of an open set $\Omega \subseteq \mathbb{R}^2$ onto $S$ such that*

1. *$\boldsymbol{f}$ has continuous derivatives up to order $m$.*

*Suppose that $\boldsymbol{f}(u, v) = \big(f^1(u, v), f^2(u, v), f^3(u, v)\big)$ and let $D_1\boldsymbol{f}$ and $D_2\boldsymbol{f}$ denote differentiation with respect to the first and second variables of $\boldsymbol{f}$, respectively. The parametric representation $\boldsymbol{f}$ is said to be regular if in addition*

*(ii) the Jacobian matrix of $\boldsymbol{f}$ given by*

$$J(\boldsymbol{f}) = \begin{pmatrix} D_1 f^1(u, v) & D_2 f^1(u, v) \\ D_1 f^2(u, v) & D_2 f^2(u, v) \\ D_1 f^3(u, v) & D_2 f^3(u, v) \end{pmatrix}$$

*has full rank for all $(u, v)$ in $\Omega$.*

That $J(\boldsymbol{f})$ has full rank means that its two columns must be linearly independent for all $(u, v) \in \Omega$, or equivalently, that for all $(u, v)$ there must be at least one nonsingular $2 \times 2$ submatrix of $J(\boldsymbol{f})$.

A function of two variables $z = h(x, y)$ can always be considered as a parametric surface through the representation $\boldsymbol{f}(u, v) = \big(u, v, h(u, v)\big)$.

In the following we will always assume that $\boldsymbol{f}$ is sufficiently smooth for all operations on $\boldsymbol{f}$ to make sense.

It turns out that there are many surfaces that cannot be described as the image of a regular parametric representation. One example is a sphere. It can be shown that it is impossible to find one regular parametric representation that can cover the whole sphere. Instead one uses several parametric representations to cover different parts of the sphere and call the collection of such representations a parametric surface. For our purposes this is unnecessary, since we are only interested in analyzing a single parametric representation given as a spline. We will therefore often adopt the sloppy convention of referring to a parametric representation as a surface.

Let us check that the surface given by (7.42) is regular. The Jacobian matrix is easily computed as

$$J(\boldsymbol{f}) = \big(\boldsymbol{p}_1 - \boldsymbol{p}_0, \boldsymbol{p}_2 - \boldsymbol{p}_0\big)\,,$$

(the two vectors $\boldsymbol{p}_1 - \boldsymbol{p}_0$ and $\boldsymbol{p}_2 - \boldsymbol{p}_0$ give the columns of $J(\boldsymbol{f})$). We see that $J(\boldsymbol{f})$ has full rank unless $\boldsymbol{p}_1 - \boldsymbol{p}_0 = \lambda(\boldsymbol{p}_2 - \boldsymbol{p}_0)$ for some real numer $\lambda$, i.e., unless all three points lie on a straight line.

A curve on the surface $S$ of the form $\boldsymbol{f}(u, v_0)$ for fixed $v_0$ is called a $u$-curve, while a curve of the form $\boldsymbol{f}(u_0, v)$ is called a $v$-curve. A collective term for such curves is *iso-parametric curves*.

Iso-parametric curves are often useful for plotting. By drawing a set of $u$- and $v$-curves, one gets a simple but good impression of the surface.

The first derivatives $D_1\boldsymbol{f}(u, v)$ and $D_2\boldsymbol{f}(u, v)$ are derivatives of, and therefore tangent to, a $u$- and $v$-curve respectively. For a regular surface the two first derivatives are linearly independent and therefore the cross product $D_1\boldsymbol{f}(u, v) \times D_2\boldsymbol{f}(u, v)$ is nonzero and normal to the two tangent vectors.

**Definition 7.11.** *The unit normal of the regular parametric representation $\boldsymbol{f}$ is the vector*

$$\boldsymbol{N}(u, v) = \frac{D_1\boldsymbol{f}(u, v) \times D_2\boldsymbol{f}(u, v)}{\|D_1\boldsymbol{f}(u, v) \times D_2\boldsymbol{f}(u, v)\|}.$$

The normal vector will play an important role when we start analyzing the curvature of surfaces.

Let $\big(u(\sigma), v(\sigma)\big)$ be a regular curve in the domain $\Omega$ of a parametric representation $\boldsymbol{f}$. This curve is mapped to a curve $\boldsymbol{g}(\sigma)$ on the surface,

$$\boldsymbol{g}(\sigma) = \boldsymbol{f}\big(u(\sigma), v(\sigma)\big).$$

The tangent of $\boldsymbol{g}$ is given by

$$\boldsymbol{g}'(\sigma) = u'(\sigma)D_1\boldsymbol{f}\big(u(\sigma), v(\sigma)\big) + v'(\sigma)D_2\boldsymbol{f}\big(u(\sigma), v(\sigma)\big),$$

in other words, a linear combination of the two tangent vectors $D_1\boldsymbol{f}\big(u(\sigma), v(\sigma)\big)$ and $D_2\boldsymbol{f}\big(u(\sigma), v(\sigma)\big)$. Note that $\boldsymbol{g}$ is regular since $\boldsymbol{g}'(\sigma) = 0$ implies $u'(\sigma) = v'(\sigma) = 0$.

All regular curves on $S$ through the point $\boldsymbol{f}(u, v)$ has a tangent vector on the form $\delta_1 D_1\boldsymbol{f} + \delta_2 D_2\boldsymbol{f}$, where $\boldsymbol{\delta} = (\delta^1, \delta^2)$ is a vector in $\mathbb{R}^2$. The space of all such tangent vectors is the tangent plane of $S$ at $\boldsymbol{f}(u, v)$.

**Definition 7.12.** *Let $S$ be a surface with a regular parametric representation $\boldsymbol{f}$. The tangent space or tangent plane $T\boldsymbol{f}(u, v)$ of $S$ at $\boldsymbol{f}(u, v)$ is the plane in $\mathbb{R}^3$ spanned by the two vectors $D_1\boldsymbol{f}(u, v)$ and $D_2\boldsymbol{f}(u, v)$, i.e., all vectors on the form $\delta_1 D_1\boldsymbol{f}(u, v)+\delta_2 D_2\boldsymbol{f}(u, v)$.*

Note that the normal of the tangent plane $T\boldsymbol{f}(u, v)$ is the normal vector $\boldsymbol{N}(u, v)$.

### 7.5.1 Parametric Tensor Product Spline Surfaces

Given how we generalized from spline functions to parametric spline curves, the definition of parametric tensor product spline surfaces is the obvious generalization of tensor product spline functions.

**Definition 7.13.** *A parametric tensor product spline surface is given by a parametric representation on the form*

$$\boldsymbol{f}(u, v) = \sum_{i=1}^{m}\sum_{j=1}^{n} \boldsymbol{c}_{i,j} B_{i,d,\boldsymbol{\sigma}}(u) B_{j,\ell,\boldsymbol{\tau}}(v),$$

*where the coefficients $(\boldsymbol{c}_{i,j})_{i,j=1}^{m,n}$ are points in space,*

$$\boldsymbol{c}_{i,j} = (c^1_{i,j}, c^2_{i,j}, c^3_{i,j}),$$

*and $\boldsymbol{\sigma} = (\sigma_i)_{i=1}^{m+d+1}$ and $\boldsymbol{\tau} = (\tau_j)_{j=1}^{n+\ell+1}$ are knot vectors for splines of degrees $d$ and $\ell$.*

As for curves, algorithms for tensor product spline surfaces can easily be adapted to give methods for approximation with parametric spline surfaces. Again, as for curves, the only complication is the question of parametrization, but we will not consider this in more detail here.