# Chapter 1
# Techniques for fair and shape–preserving surface fitting with tensor–product B-splines

J. Hoschek and B. Jüttler

**Abstract.** Tensor–product B-spline surfaces combine computational efficiency with geometric flexibility; this makes them a powerful tool for describing free–form surfaces in Computer–Aided Geometric Design. This chapter reports on methods for fair and shape–preserving surface fitting with tensor–product B-splines, both for functional and parametric surfaces. The techniques for fair surface design are based on several fairness measures, including isophotes, reflection lines, and various energy functionals. In addition, we describe shape–preserving techniques, where the desired sign of the Gaussian curvature can be chosen by the user.

## 1. Introduction

Tensor–product B-spline surface have become the de–facto standard for surface representation in Computer–Aided Design, as they are computationally efficient and geometrically flexible. In this chapter we give a survey on fair and shape–preserving techniques for surface construction, fitting, and modification with tensor–product B-splines. Many of these techniques are based on suitable linearizations of the non–linear fairness and shape criteria. This leads to computationally efficient algorithms for shape design. We discuss both parametric and functional surface representations.

This chapter is organized as follows. Section 2 introduces some notations and provides some background from spline theory. The next two sections are devoted to functional surfaces, whereas Sections 5 and 6 deal with truly parametric surface patches. Section 3 discusses fair functional surfaces. Here, we focus mainly on high–quality surfaces such as lenses, where fairness is judged by distribution and flow of isophotes and reflection lines. We describe methods for constructing and modifying fair functional surfaces, according to these criteria. Section 4 is devoted to shape–preserving surface fitting with tensor–product spline functions. We develop an approximation scheme where it is possible to specify the signs of the Gaussian curvature (i.e., convexity and concavity) for the individual spline segments. The scheme is based on a suitable linearization of the shape criteria that leads to arbitrarily weak linear convexity and concavity conditions. Section 5 reports on methods for designing fair parametric surfaces. We discuss several fairness functionals and describe related techniques for surface fairing and surface fitting. These methods can be used in reverse engineering for the reconstruction process of surfaces of physical objects into geometric surface models. By digitizing the objects (e.g., using laser–scanning devices) one obtains discrete data, i.e. (possibly triangulated) point sets. In order to get a representation that is suitable for CAD applications, these data have to be approximately converted into smooth surface models. Finally, Section 6 gives a survey on convexity criteria for parametric surfaces and related techniques for shape–preserving surface design.

## 2. Surface fitting with tensor–product B-Splines

This section recalls some fundamental facts and notions concerning tensor–product (TP) B-spline surfaces. Consider two monotonically increasing knot sequences, $U = (u_0, \ldots, u_m)$ and $V = (v_0, \ldots, v_n)$. Let $(M_i(u))_{i=0,\ldots,m^*}$ and $(N_j(v))_{j=0,\ldots,n^*}$ be the associated B–spline functions of degree $d$, where $m^* = m{-}d{-}1$, $n^* = n{-}d{-}1$. For the definitions of these functions, and for other related information, we refer to one of the various textbooks on this topic, e.g., [**21,34**]. We assume that both knot sequences have $(d+1)$–fold boundary knots $u_0 = \ldots = u_d$, $u_{m-d} = \ldots = u_m$ (and similarly for $V$), but single inner knots. Hence, the B–spline functions $M_i$, $N_j$ span the linear space of all spline functions of degree $d$ from $C^{d-1}[u_0, u_m]$ and $C^{d-1}[v_0, v_n]$ with the inner knots from $U$ and $V$.

The real–valued function function $x : [u_0, u_m] \times [v_0, v_n] \to \mathbf{R}$,

$$x(u,v) = \sum_{i=0}^{m^*} \sum_{j=0}^{n^*} M_i(u)\ N_j(v)\ d_{i,j}, \quad (u,v) \in [u_0, u_m] \times [v_0, v_n], \tag{2.1}$$

with the coefficients (or control points) $d_{i,j} \in \mathbf{R}$ is called a *TP B–spline surface patch* of degree $(d, d)$. More precisely, the surface is given by the *graph* $(\, u \quad v \quad x(u,v)\,)^\top$ of the above function.

By choosing vector–valued (rather than scalar) coefficients $\mathbf{d}_{i,j} \in \mathbf{R}^3$ we obtain the *parametric TP B-spline surface patch*

$$\mathbf{x}(u,v) = \sum_{i=0}^{m^*} \sum_{j=0}^{n^*} M_i(u)\ N_j(v)\ \mathbf{d}_{i,j}, \quad (u,v) \in [u_0, u_m] \times [v_0, v_n], \tag{2.2}$$

that describes the surface by its parametric representation, $\mathbf{x} : [u_0, u_m] \times [v_0, v_n] \to \mathbf{R}^3$.

Clearly, by using B-spline surfaces (2.1) we limit ourselves to surfaces which can be interpreted as the graph of a function. This limitation can be avoided with the help of the parametric surface representations (2.2). On the other hand, the construction of truly parametric surfaces raises more complicated mathematical problems, as we will see later. In many applications one will therefore prefer to use the simpler functional spline surfaces (2.1).

Of course, the parametric surfaces (2.2) can describe the graphs of spline functions (2.1) also. This is achieved by choosing the first two components of the control points equal to the *Greville abscissae* associated with the given knot sequences $U$, $V$, cf. [**34**]. Then, the first two components of the parametric surface are the parameter values, i.e.,

$$\mathbf{x}(u,v) = \begin{pmatrix} u \\ v \\ x(u,v) \end{pmatrix}, \quad (u,v) \in [u_0, u_m] \times [v_0, v_n]. \tag{2.3}$$

As a first example for the problems that will be discussed in this chapter, we briefly recall the problem of *surface fitting.* It is one of the typical problems that arise in many applications of TP B-spline surfaces, such as reverse engineering or data exchange. This problem can be stated as follows: given a "cloud" of $P$ data (e.g. measurement data) $\mathbf{p}_i = (p_{i,1}\ p_{i,2}\ p_{i,3}) \in \mathbf{R}^3$, $i = 0, \ldots, P$, from an initial surface ($P$ large), find a TP B-spline surface that approximately matches the data. If the surface is described as the graph of the spline function (2.1), then the TP B-spline surface is to satisfy

$$x(p_{i,1}, p_{i,2}) \approx p_{i,3}, \quad i = 0, \ldots, P. \tag{2.4}$$

Note that the functional approach can be used only if the initial surface can be interpreted as the graph of a function, and $(p_{i,1}, p_{i,2}) \in [u_0, u_m] \times [v_0, v_n]$. In many cases these assumptions can easily be satisfied by introducing suitable coordinates, e.g., by choosing the $xy$–plane as the best–fitting plane of the given data.

Otherwise, if one uses the parametric surface description (2.2), then one firstly has to associate suitable parameter values $(u_i, v_i)$ with the data (see Section 5 for more information). The parametric surface patch is to satisfy

$$\mathbf{x}(u_i, v_i) \approx \mathbf{p}_i, \quad i = 0, \ldots, P. \tag{2.5}$$

As the standard technique, the unknown coefficients $d_{i,j}$ resp. $\mathbf{d}_{i,j}$ of the surface are often computed by minimizing the sum of squared errors

$$\mathcal{L} = \sum_{i=0}^{P} (x(p_{i,1}, p_{i,2}) - p_{i,3})^2 \quad \text{resp.} \quad \mathcal{L}^* = \sum_{i=0}^{P} \|\mathbf{x}(u_i, v_i) - \mathbf{p}_i\|^2, \tag{2.6}$$

see e.g. [**21**]. These coefficients can then easily be found by solving a system of linear equations.

For certain data (e.g., so–called uncertain data which contain "outliers") it may be advantageous to consider other kinds of error functions, like the $\ell_1$ or $\ell_\infty$ norm of the errors. Many related references can be found in [**26**]. For these error functions the unknown coefficients $d_{i,j}$ resp. $\mathbf{d}_{i,j}$ can be computed via linear programming.

In the remaining sections of this chapter we will study problems that are related to surface fitting, but with additional shape and fairness constraints for the desired spline surfaces. We will not address the question of how to find suitable knot sequences $U$, $V$. An automatic adaptive algorithm has been developed by Dierckx [**7**]. In practice, however, one will often prefer to use a semi–automatic procedure with some user interaction, as the optimal knot placement depends heavily on the shape of the given data.

As an alternative to tensor–product splines (where additional degrees of freedom can only be introduced in strips parallel to either the $u$ or the $v$–direction, due to the global influence of the knots), the so–called hierarchical B–splines [**14,16**] provide the possibility of adding degrees of freedom that act only locally. The methods described below can readily be adapted to hierarchical B-spline representations.

## 3. Fair surfaces

Fairness functionals or visual properties play an important role for designing fair surfaces. Suitable fairness functionals include the thin plate energy (which can be expressed by Gaussian curvature $K$ and mean curvature $H$), linearized versions of it, or functionals involving higher derivatives. Visual properties can be the distribution and flow of reflection lines and/or isophotes. Clearly, all these criteria for fair surfaces are non–linear. In order to get algorithms which are suitable for real–time applications, approximations of these functionals or restrictions to B-spline (functional) surface patches are widely used in practice.

While the approximation of fairness functionals for parametric surfaces will be discussed in Section 4, we will now focus on fair B-spline surfaces $\mathbf{x}(u,v) = (\, u \;\; v \;\; x(u,v)\,)^\top$, that describe the graph of a function, see (2.1) and (2.3). We consider surfaces with a desired distribution of the mean curvature $H = \frac{1}{2}(\kappa_1 + \kappa_2)$ (with $\kappa_i$ as principal curvatures in the point $(u,v)$), and surfaces with a desired flow of isophotes or reflection lines. At first we discuss the problem of lens design for spectacles combining near– and far–vision areas, where high–quality surfaces are needed [**31,35**]. The second part of this section is devoted to the problem of smoothing a given surface by prescribing its isophotes or reflection lines. The potential applications of this part include the design of car body surfaces in automotive industry.

The quality of the front surface of a lens is measured by the following error functional

$$Q(x) = \int_\Omega \alpha(\kappa_1 - \kappa_2)^2 + \beta\left(\frac{\kappa_1+\kappa_2}{2} - H_{req}\right)^2 du\, dv\,, \tag{3.1}$$

cf. [**29,31**], with the principal curvatures $\kappa_1$, $\kappa_2$ of the surface, the design parameters $\alpha$, $\beta > 0$, and with the required mean curvature $H_{req}$ of the front surface. The back surface of the lens is assumed to be spherical. The user has to specify a certain mean curvature $H_l$ in the far–vision area. In optical terms, the mean curvature is called (surface) *power*. The required surface power $H_n$ in the near vision area is not specified directly. Instead, the *progression* $H_n - H_l > 0$ corresponding to the amount of missing accommodation of the eye will be chosen. Ideally, the front surface should be 'locally spherical', i.e. the so–called surface *astigmatism* $|\kappa_1 - \kappa_2|$ should vanish. Unfortunately, this requirement cannot be fulfilled exactly, as locally spherical surfaces would have constant mean curvature. In order to get an approximate solution one uses a surface with minimal astigmatism $|\kappa_1 - \kappa_2|$ instead. Usually, one tries to minimize the astigmatism in the far– and near–vision areas (the upper and the lower part of the lens) and within in a zone of progression connecting both regions.

The front surface of a lens is represented by a graph of a TP B-spline function $x = x(u,v)$, whose domain $\Omega \in \mathbf{R}^2$ is chosen as a disc. The optimal front surface is then determined as an approximate solution to a variational problem:

- The desired mean curvature distribution is specified by a function $H_{req} = H_{req}(u,v)$.
- The weight functions $\alpha = \alpha(u,v)$, $\beta = \beta(u,v)$ (design parameters) are chosen in a suitable way. They control the influence of the two parts of the error functional $Q(x)$ to the different regions of the lens surface.

- The quality of the surface is measured by the error functional $Q(x)$ in (3.1).
- The front surface of the lens is obtained as an (approximate) solution of $Q(x) \to \min$. The result depends on the design parameters $\alpha, \beta$ and on the mean curvature $H_{req}$.

The error functional (3.1) is highly non–linear. Loos et al. [**29,31**] have proposed to minimize it with the help of a Newton–Raphson–like iteration algorithm. We introduce in (3.1) the well–known formulas for Gaussian curvature $K$ and mean curvature $H$ for functional surfaces. Moreover, we approximate (3.1) by its Taylor expansion. Let $J = J(x_u, x_v, x_{uu}, x_{uv}, x_{uu})$ be the integrand in (3.1), and let $L$ be the operator $L(x) = (x_u, x_v, x_{uu}, x_{uv}, x_{vv})$. After some calculations we obtain from (3.1) the following second–order Taylor approximation at $x = x_0$,

$$Q_{x_0}(x) = Q(x_0) + \int_\Omega \operatorname{grad} J(L(x_0))^T \, L(h) \, du \, dv + \frac{1}{2} \int_\Omega L(h)^T \operatorname{Hess} J(L(x_0)) \, L(h) \, du \, dv \tag{3.2}$$

with $h = x - x_0$, and the gradient $\operatorname{grad} J$ and the Hessian $\operatorname{Hess} J$ of $J$. Then $Q_{x_0}(x) = Q_{x_0}(x_0 + h)$ is a quadratic functional in $h$ and can therefore be minimized efficiently. That is, the control points of the B-spline surface $x_0 + h$ minimizing (3.2) can be computed by solving a system of linear equations.

The Newton–Raphson–like iteration algorithm needs an initial solution to start with. It turns out that it would be sufficient to use a planar disc, $x(u, v) \equiv 0$. However, in order get faster convergence of the iterative algorithm, a part of a sphere or the surface representation of a given lens can be used. Loos et al. [**29,31**] propose to use bicubic tensor–product surfaces which are found by interpolating or approximating (least–squares fit) of given point data. The data are sampled from an existing lens using a mechanical device.

In order to guarantee existence and uniqueness of a numerical solution, certain boundary conditions have to be added. As observed by Loos et al. [**30,31**], it suffices to require $x(0,0) = x_u(0,0) = x_v(0,0) = 0$.

As an example, Figure 3.1 shows the improvement of a given progressive lens. The left figures show power and astigmatism of lenses available on the market today. The right figure visualizes the optimization results.

Now, we will determine fair surfaces by visual properties, such as the distribution and the flow of the isophotes and/or reflection lines. With the unit normal vector

$$\mathbf{n} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{|\mathbf{x}_u \times \mathbf{x}_v|} = \frac{(-x_u, -x_v, 1)}{\sqrt{1 + x_u^2 + x_v^2}}$$

of the required surface and $\mathbf{l} = (a, b, c)$ as the direction of the incident light, the light intensity $I^x$ is given by

$$I^x = \mathbf{n} \cdot \mathbf{l} = \frac{c - a x_u - b x_v}{\sqrt{x_u^2 + x_v^2}} = \text{const.}, \tag{3.3}$$

cf. [**1,21,31**] The implicit equation $I^x(u, v) = \text{const.}$ describes the field of *isophotes*.

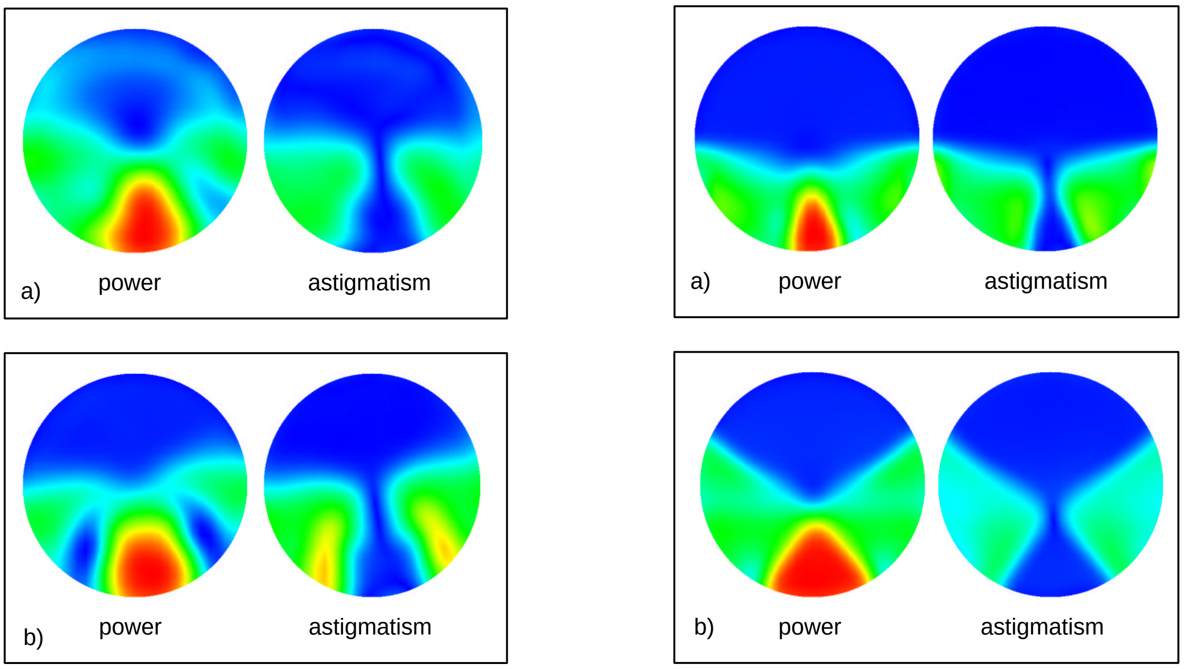

Figure 3.1. Refractive power and optical error (astigmatism) of a progressive lens without and with optimization. (Courtesy by G. Greiner and J. Loos, Erlangen, Germany)

Andersson [**1**] has considered (3.3) as a partial differential equation of first order and determined the solution with the method of characteristic curves. Loos et al. have developed another approach. The desired lightness function $I^*$ is given. A surface $x(u,v)$ is to be found such that the contour curves of $I^x$ and $I^*$ are as similar as possible. This leads to the fairness functional

$$J(x) = \int_\Omega (I^x - I^*)^2 du\, dv = \int_\Omega \left( \frac{c - ax_u - bx_v}{\sqrt{x_u^2 + x_v^2}} - I^* \right)^2 du\, dv\,. \tag{3.4}$$

Unfortunately, this objective function will generally not produce a satisfying surface shape, see [**29,31**]. Therefore, Loos proposes to use another functional which is based on the gradients of the lightness functions,

$$J(x) = \int_\Omega (\mathrm{grad} I^x - \mathrm{grad} I^*)^2\, du\, dv = \int_\Omega \left( (I_u^x - I_u^*)^2 + (I_v^x - I_v^*)^2 \right) du\, dv. \tag{3.5}$$

Again, this functional is minimized using a Newton–Raphson–like method, based on a Taylor expansion that is analogous to (3.2).

Similar to the isophotes, the *reflection lines* can be described by a coordinate function $C(u,v)$, see [**19,21,29,31**]. Then, the individual reflection lines of the surface $\mathbf{x}$ are implicitly given by $C(u,v) = \text{const}$. The coordinate function depends on the type of the light sources: axial or radial lines are used in practice.

In order to construct surfaces with a prescribed distribution of reflection lines, the following fairness functional has been introduced by Loos et al. [**31**],

$$J(x) = \int_\Omega (\mathrm{grad} C^x - \mathrm{grad} C^*)^2 \, du \, dv = \int_\Omega \Big( (C^x_u - C^*_u)^2 + (C^x_v - C^*_v)^2 \Big) \, du \, dv. \tag{3.6}$$

The function $C^*(u, v)$ describes the desired flow of the reflection lines.

The potential applications of this approach include surface fairing: if the isophotes or reflection lines have, in a region of interest of a given surface, an undesired flow, then the user can specify the desired isophotes or reflection lines either graphically or by choosing the coordinate functions $I^*$ or $C^*$. Then the given surface is faired by minimizing the fairness functionals (3.5) or (3.6). Figure 3.2 gives an example of such a fairing process.

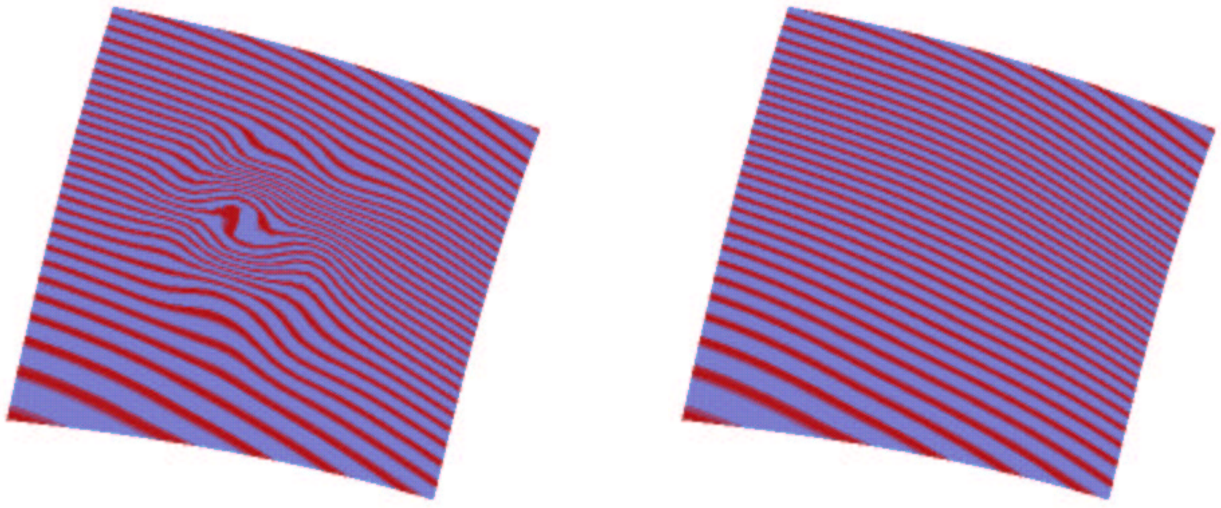
Figure 3.2. Flow of isophotes on a given surface (left) and after fairing using (3.6) (right). (Courtesy by G. Greiner and J. Loos, Erlangen, Germany)

Unfortunately, this approach cannot easily be extended to parametric surfaces, which are often used in applications. In order to overcome this difficulty, we will discuss in Section 5 the approximation of the normal field of the desired surface as a simulation of the use of isophotes or reflection lines.

## 4. Convex surfaces

In this section we present a method for surface fitting with TP splines subject to convexity and concavity conditions. Only very few methods for the construction of convex spline surfaces are available in the literature, and the existing ones deal exclusively with Powell–Sabin (PS) splines. These spline surfaces are described by the graphs of piecewise quadratic functions, defined over special triangulations in $\mathbf{R}^2$.

Carnicer and Dahmen [**4**] describe a method for convexity–preserving interpolation with PS splines. Unfortunately, as recently observed by Floater [**13**], in certain cases this method fails to produce convex interpolants to convex data.

An algorithm for least–squares surface fitting using convex PS splines has been developed by Willemans and Dierckx [**37**]. This algorithm is based on the convexity criterion by Chang and Feng [**6**], that leads to a set of linear and quadratic inequalities which are necessary and sufficient for convexity of PS splines. The coefficients of the PS spline surface are found by minimizing the sum of squared errors subject to the linear and quadratic inequality constraints. The resulting nonlinear optimization problem is then solved with the help of a suitable function from the NAG library.

The shortage of methods seems to be due to the lack of suitable convexity conditions, particularly for tensor–product spline surfaces. In the sequel we describe a method for convex surface fitting using TP splines. Based on a suitable linearization of the convexity conditions, we are able to formulate this task as a quadratic programing problem (minimization of a quadratic objective function subject to linear constraints). This is one of the standard problems in optimization, and there is a number of powerful solvers available.

The given knot sequences $U$, $V$ define a partition of the domain of the spline function $x(u,v)$ (see Eq.(2.1)) into cells

$$C_{i,j} = [u_i, u_{i+1}] \times [v_j, v_{j+1}], \quad i = d, \ldots, m^*;\ j = 0, \ldots, n^*. \tag{4.1}$$

The restriction $x(u,v)|_{C_{i,j}}$ of the spline function (2.1) to each cell is a bivariate TP polynomial of degree $(d,d)$. Firstly we have to decide for each cell whether we wish to fit a convex or concave spline segment. We list two possible criteria for this decision.

1. One may try to find a convex or concave triangulation for the data that belong to the cell, possibly also including the data from the neighbouring cells. The construction of such a triangulation has recently been discussed by Carnicer and Floater [**5**]. If a convex (resp. concave) triangulation exists, then the corresponding spline segment $x(u,v)|_{C_{i,j}}$ is restricted to be convex (resp. concave). As a disadvantage of this criterion, it will not detect data that are 'approximately' convex, within some tolerance.

2. We fit a quadratic polynomial to the data that are contained in the cell, and possibly also to the data from the neighbouring cells. The polynomial can easily be computed by solving a $6 \times 6$ system of linear equations. Its shape can then be used as the criterion for the shape of the spline segment $x(u,v)|_{C_{i,j}}$. If the Hessian matrix (which is constant for quadratic polynomials) is positive (resp. negative) definite, then the spline segment is chosen to be convex (resp. concave). No shape constraints are applied in the case of an indefinite Hessian.

As the result, we associate with each cell $C_{i,j}$ a value $\gamma_{i,j} \in \{-1, 0, +1\}$ in order to specify whether $x(u,v)|_{C_{i,j}}$ should be convex ($\gamma_{i,j} = 1$) or concave ($\gamma_{i,j} = -1$). No constraints are imposed if $\gamma_{i,j} = 0$. We assume, however, that no convex and concave spline segments sharing a boundary or a vertex exist,

$$(i,k) \neq (j,l) \quad \wedge \quad \gamma_{i,j} \cdot \gamma_{k,l} = -1 \quad \Rightarrow \quad \max(|i-k|, |j-l|) > 1. \tag{4.2}$$

Clearly, convexity (resp. concavity) of the spline surface implies that the second directional derivatives are nonnegative (nonpositive). Hence, if convex and concave spline segments would meet along a common boundary, then the second directional derivatives were to vanish there. Thus, the spline function had to be linear along the common edge. Similarly, if concave and convex cells of the spline surface would be to share a vertex, then the shape of the surface in the neighbourhood was very restricted.

The above two shape criteria do not take into account the additional condition (4.2). If they propose to use neighbouring spline segments with constrained, but different shapes, then one should simply choose the shape of these cells to be unconstrained ($\gamma_{i,j} = 0$) instead.

As an example we consider the surface from Figure 4.1. (left). We sampled randomly 200 data from the surface and perturbed the $z$–components by random numbers from the interval $[-0.15, 0.15]$. The data and the knot lines of the approximating spline surfaces ($4 \times 4$ bicubic segments) have been drawn on the right–hand side of the figure. We used the second criterion in order to decide the shape of the approximating spline surface; the grey cells correspond to concave regions of the surface, white cells are unconstrained. The second criterion were based on quadratic functions (paraboloids) obtained by least–squares approximation of the data from one cell and from all its neighbouring cells.

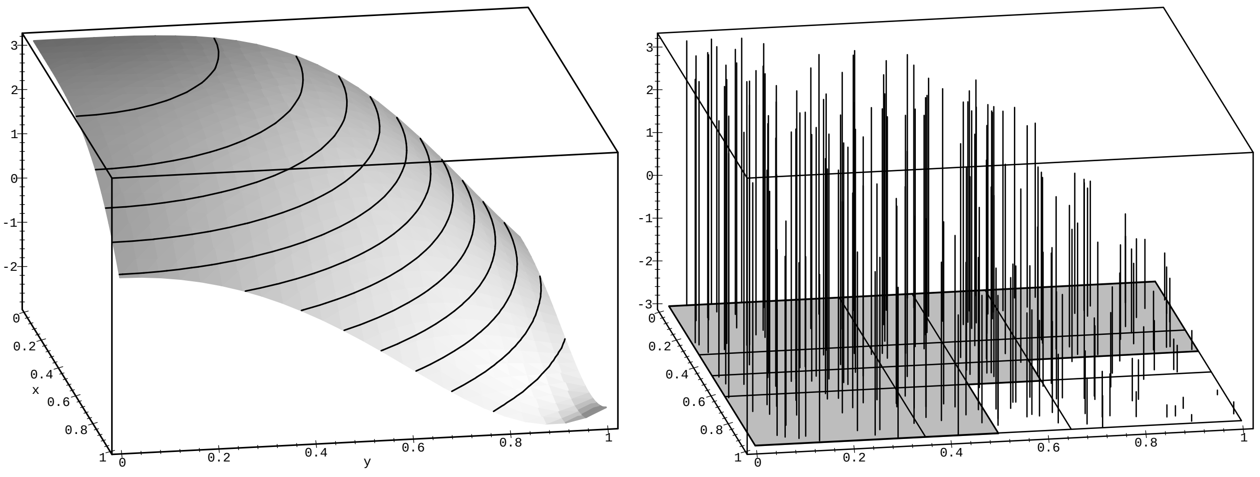

Figure 4.1. Original surface (left). Data and associated shape indicators $\gamma_{i,j}$ (right).

Recall that the surface $x(u, v)$ is convex (concave) in a cell (4.1) if and only if the second directional derivatives in all directions $\vec{\mathbf{r}} = (r_1\ r_2)^\top$,

$$\left(\frac{\partial}{\partial \vec{\mathbf{r}}}\right)^2 x(u,v) = \frac{d^2}{dt^2} x(u+tr_1, v+tr_2)\bigg|_{t=0} = (r_1\ r_2) \underbrace{\begin{pmatrix} x_{uu}(u,v) & x_{uv}(u,v) \\ x_{uv}(u,v) & x_{vv}(u,v) \end{pmatrix}}_{=H(u,v)} \begin{pmatrix} r_1 \\ r_2 \end{pmatrix}, \quad (4.3)$$

are nonnegative (resp. nonpositive) for all $(u, v) \in C_{i,j}$. That is, the Hessian matrix $H(u, v)$ of the spline function is *positive* (resp. negative) *semidefinite.* Using standard B-spline techniques (knot insertion) we get a Bézier representation of the Hessian,

$$H(u,v) = \sum_{r=0}^{d} \sum_{s=0}^{d} B_r^d(u, u_i, u_{i+1})\ B_s^d(v, v_j, v_{j+1})\ B_{r,s} \quad (4.4)$$

over the cell $C_{i,j}$. Here we use the Bernstein polynomials

$$B_k^d(t,a,b) = \binom{d}{k} \frac{(t-a)^k(b-t)^{d-k}}{(b-a)^d} \tag{4.5}$$

of degree $d$ with respect to the interval $[a,b]$. The second partial derivatives $x_{uu}$, $x_{uv}$, and $x_{vv}$ are bivariate TP polynomials of degree $(d{-}2,d)$, $(d{-}1,d{-}1)$, and $(d,d{-}2)$, respectively. Thus, the Hessian is a matrix–valued polynomial of degree $(d,d)$. The components of the $(d{+}1)^2$ symmetric coefficient matrices $B_{r,s} \in \mathbf{R}^{2\times 2}$ are certain constant linear combinations of the B-spline control points $d_{i,j}$, the coefficients of which depend on the given knot sequences. In the bicubic case, explicit formulae for the matrices are given in [**24**]. Clearly, the coefficient matrices $B_{r,s}$ depend on the cell $C_{i,j}$ also. In order to keep notations simple we do not refer to the indices $i,j$.

In order to guarantee the desired shape features of the spline surface (i.e., convexity and/or concavity of the segments) one has to find conditions which imply that the Hessian matrix is nonnegative (resp. nonpositive) definite. For $(u,v) \in C_{i,j}$, the Hessian matrix is a nonnegative linear combinations of the coefficient matrices $B_{r,s}$. Hence it is sufficient to find such conditions for the individual coefficient matrices. A symmetric $2\times 2$–matrix $B = B_{r,s} = (b_{k,l})_{k,l=1,2}$ is positive (resp. negative) semidefinite if and only if the inequalities

$$b_{1,1} \geq 0,\ b_{2,2} \geq 0\ \ (\text{resp. } b_{1,1} \leq 0,\ b_{2,2} \leq 0) \quad \text{and} \quad b_{1,1}b_{2,2} - b_{1,2}^2 \geq 0 \tag{4.6}$$

hold. These conditions, however, would produce quadratic inequalities for the control points $d_{i,j}$, as the components of the matrices $B_{r,s}$ are linear combinations of them. It is advantageous to use a stronger set of linear sufficient conditions instead, as the resulting optimization problem is easier to deal with. Such linear conditions can be generated with the help of the following simple observation.

**Lemma 4.1.** *Let $N \geq 2$ be an integer. Consider the $2N$ linear inequalities*

$$\begin{aligned} (1-\tau_{k-1})(1-\tau_k)\, b_{1,1} + (\tau_{k-1} + \tau_k - 2\tau_{k-1}\tau_k)\, b_{1,2} + \tau_{k-1}\tau_k\, b_{2,2} \geq 0 \\ (1-\tau_{k-1})(1-\tau_k)\, b_{1,1} - (\tau_{k-1} + \tau_k - 2\tau_{k-1}\tau_k)\, b_{1,2} + \tau_{k-1}\tau_k\, b_{2,2} \geq 0 \end{aligned} \tag{4.7}$$

*with $\tau_k = \frac{k}{N}$, $k = 1,\ldots,N$.*

(*i*) *If the components of the symmetric $2\times 2$–matrix $B$ satisfy the linear inequalities (4.7), then $B$ is positive semidefinite.*

(*ii*) *Consider finitely many positive definite symmetric $2\times 2$–matrices. If $N$ is big enough, then the inequalities (4.7) are satisfied for all matrices.*

**Proof.** The matrix B is nonnegative definite if and only of the quadratic polynomials

$$\begin{aligned} q_1(t) &= (1-t)^2\, b_{1,1} + 2t(1-t)\, b_{1,2} + t^2\, b_{2,2} \quad \text{and} \\ q_2(t) &= (1-t)^2\, b_{1,1} - 2t(1-t)\, b_{1,2} + t^2\, b_{2,2} \end{aligned} \tag{4.8}$$

have nonnegative values for $t \in [0,1]$, this is equivalent to (4.6). We consider the Bézier representation of these polynomials with respect to the $N$ segments $[\tau_{k-1}, \tau_k]$ (which form a partition of the unit interval) and obtain

$$q_l(t) = B_0^2(t, \tau_{k-1}, \tau_k)\, q_l(\tau_{k-1}) + B_1^2(\ldots)\, Q_{k,l} + B_2^2(\ldots)\, q_l(\tau_k), \quad l = 1, 2, \tag{4.9}$$

where the middle coefficients $Q_{k,l}$ are exactly the left–hand sides of the inequalities (4.7). For proving ($i$) we observe that the inequalities (4.7) guarantee that all Bézier coefficients of the segments (4.9) are nonnegative (the boundary coefficients can be shown to be nonnegative linear combinations of neighbouring coefficients $Q_{k,l}$). On the other hand, ($ii$) can be concluded from the well–known fact that the Bézier coefficients $Q_{k,l}$ converge uniformly to values of the polynomials $q_l(t)$ for decreasing interval length $\frac{1}{N} \to 0$. Hence, for positive definite matrices $B$ (where both polynomials are guaranteed to be positive on $[0,1]$, the inequalities (4.7) are satisfied for sufficiently large values of $N$. ∎

This lemma provides a construction which produces arbitrarily weak linear conditions for nonnegative definite matrices. By applying this construction to the $(d+1)^2$ coefficient matrices $B_{r,s}$ from Eq. (4.4) we obtain linearized convexity conditions for the spline segment $x(u,v)|_{C_{i,j}}$. Clearly, non–negative definiteness of the coefficient matrices is again only a sufficient condition for a non–negative definite Hessian matrix $H(u,v)$. However, it is again possible to weaken the resulting constraints, simply by splitting the individual cells $C_{i,j}$ into sub–cells, and considering the Bézier representation with respect to them. This is explained in somewhat more detail in [**24**]. Similar to Lemma 4.1 ($ii$) one may construct *arbitrarily weak linear convexity conditions;* they can be adapted to any finite set of strongly convex spline functions. In most applications, however, it is not necessary to do the splitting of the cells, as the convexity criteria are already relatively weak.

Now we are ready to formulate the shape–preserving surface fitting procedure which leads to an approximate solution of the constrained approximation problem.

1. For each individual cell, choose the desired shape $\gamma_{i,j} \in \{-1, 0, +1\}$ of the approximating spline function. See the remarks at the beginning of this section.

2. For all cells with $\gamma_{i,j} = 1$ (resp. $= -1$) we generate linear sufficient convexity (resp. concavity) conditions, by applying the construction of Lemma 4.1 (with appropriate refinement level, e.g., $N = 4$) to the $(d+1)^2$ coefficient matrices $B_{r,s}$ (resp. to $-B_{r,s}$) from Eq. (4.4). This leads to a system of linear inequalities for the unknown control points $d_{i,j}$.

3. Choose the quadratic objective function $\mathcal{F}$. For instance, a suitable choice is a weighted linear combination (with weight $w \in [0,1]$) of the squared errors sum (2.6) with any one of the 'energy' terms from (5.2),

$$\mathcal{F}((d_{i,j})_{i=0,\ldots,m^*, j=0,\ldots,n^*}) = (1-w)\, \mathcal{L}(\ldots) + w\, \mathcal{E}(\ldots). \tag{4.10}$$

   Depending on the number of the data and on its distribution in the domain $[u_0, u_m] \times [v_0, v_n]$, it may happen that the first part of the objective function is not a positive definite quadratic functional of the control points. Then, the solution of the problem

$\mathcal{F} \to$ Min would not be unique, as the resulting system of linear equations would not have full rank. By adding the weighted 'energy' term, however, a unique solution is guaranteed to exist.

4. The control points $d_{i,j}$ of the spline surface (2.1) are found by minimizing the above quadratic objective function subject to the linear inequality constraints obtained from Step 2. This is a so–called *quadratic programming problem*, see e.g. [**11**]. A number of powerful algorithms for quadratic programming are available. Our implementation relies on the LOQO package by Vanderbei [**36**].

As an example we consider again the data and the knots from Figure 4.1. Figure 4.2. shows two bicubic spline surfaces which approximate these data. The left surface has been obtained with the help of the above shape–preserving approximation procedure. The resulting quadratic programming problem (449 inequalities for 49 control points) has been solved using LOQO (1.2 seconds needed on a HP 715/64 workstation); the residual sum $\mathcal{L}$ (no energy term has been used) is equal to $\mathcal{L} = 1.40$. The second surface, by contrast, is the result of the unconstrained minimization of the squared errors sum $\mathcal{L}$, with the residual sum $\mathcal{L} = 1.18$. Clearly, the constrained approximation gives the desired shape, but the unconstrained one has a lot of undesired oscillations. The difference between both approximations becomes even more obvious by looking at the distribution of the determinant of the Hessian matrix, see Figure 4.3. Only regions with positive values of the determinant have been plotted; they correspond to surface regions that are either concave or convex.

Figure 4.4. may serve as a schematic illustration of the shape–preserving surface fitting procedure. They spline surfaces satisfying the shape constraints are represented by the dark grey region. They can be identified with a convex region (as convex linear combinations of convex spline functions are again convex) in a linear space whose dimension is equal to the number of control points $d_{i,j}$. By generating the linear sufficient convexity conditions (see step 2 of the above procedure) we inscribe a convex polyhedron. As observed earlier, the convexity conditions are arbitrarily weak, they can be adapted to any finite set of strongly convex (resp. concave) spline surfaces. That is, by increasing the number of inequalities the inscribed polyhedron converges to the full convex set of spline functions satisfying the shape constraints. The quadratic programming leads to a spline surface that approximately minimizes the quadratic objective function (visualized by the circles).

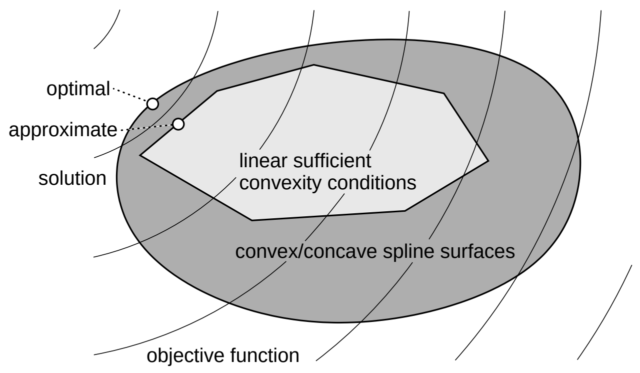

Figure 4.4. Schematic illustration of the shape–preserving surface fitting procedure.

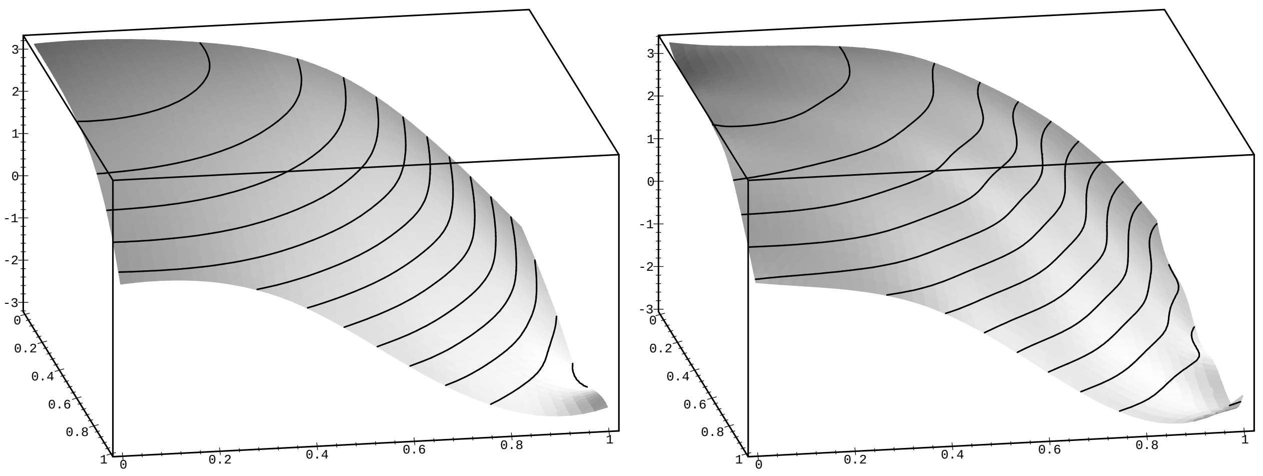

Figure 4.2. Approximation with convexity constraints (left) and unconstrained approximation (right).

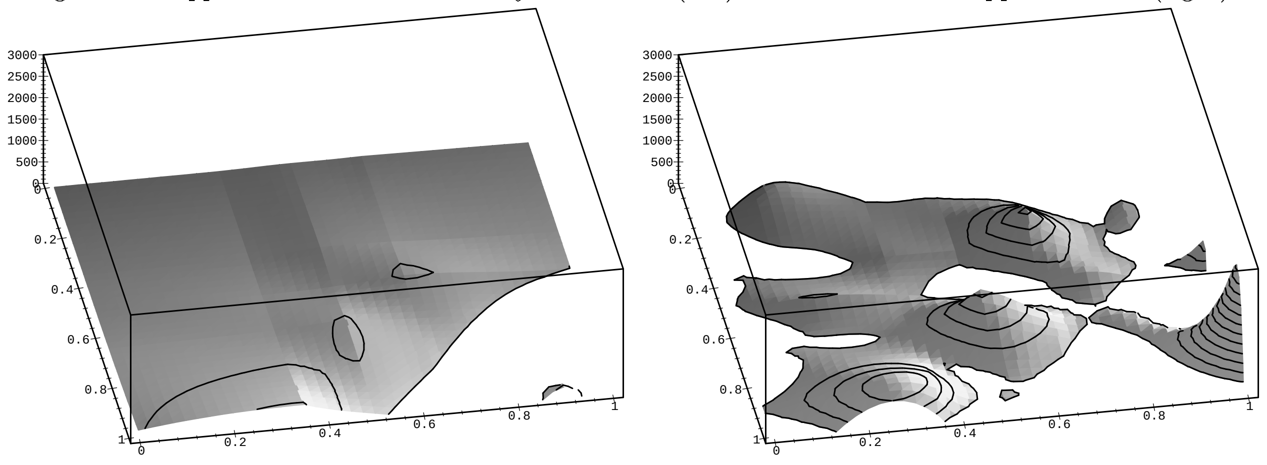

Figure 4.3. Determinant of the Hessian for the surfaces from Figure 4.2.

Clearly, the above shape–preserving surface fitting procedure can be improved by generating constraints that are adapted to the data: when a first solution to the quadratic programming has been found, we modify the linearized constraints by splitting the boundary faces of the inscribed polyhedron, cf. Figure 4.4. This is done by using non–uniform sequences $(\tau_k)_{k=0,\dots,N}$ in Lemma 4.1., and also by splitting the Bézier representation (4.4) of the Hessian into sub–cells. For any details the reader should consult [ **24** ]. As the result, the exact non–linear optimization problem (minimization of the objective function (4.10) subject to the non–linear shape constraints) is approximated by a sequence of quadratic programming problems. This leads to a *sequential quadratic programming* approach to shape–preserving surface fitting.

Note that the construction of arbitrarily weak linear convexity conditions is only possible as the spline functions satisfying the shape constraints form a convex set. That is, the approximating surface is found by minimizing a quadratic objective function on a convex domain. Clearly, this property is specific to the case of surfaces described by the graphs of *functions*. It is much more complicated to develop shape–preserving techniques for truly parametric surfaces, see Section 6.

## 5. Fair parametric surfaces

During the design process of a B-Spline surface, usually not all the available parameters (control points, parameterization) are uniquely determined by the user specifications and requirements. In many applications, the remaining degrees of freedom have been chosen by suitable heuristics. Recently, also some highly successful optimization techniques have been introduced [**2,15,16**]. Related to surface fitting and surface construction, the following two problems must be addressed.

- Fairing: given a not sufficiently smooth surface, find a fair surface.
- Surface fitting: given a set of points, find a fair surface approximating the data.

For both problems, suitable fairness criteria are needed. A reasonable measure of the surface fairness is the thin–plate–energy

$$Q = \int_\Omega (\kappa_1^2 + \kappa_2^2 + 2(1-b)\kappa_1\kappa_2)\, d\omega \tag{5.1}$$

with the principal curvatures $\kappa_i$ of the surface $\mathbf{x}(u,v)$. Unfortunately, this functional is highly non–linear. Consequently, it is not suitable for real–time applications.

Many methods described in the literature rely on the simplified versions of the thin–plate–energy and similar functionals, such as

$$\begin{aligned} Q_1 &= \int_\Omega (\mathbf{x}_{uu}^2 + 2\mathbf{x}_{uv}^2 + \mathbf{x}_{vv}^2)\, du\, dv\,, \\ Q_2 &= \int_\Omega ((\mathbf{x}_{uuu} + \mathbf{x}_{uvv})^2 + (\mathbf{x}_{uuv} + \mathbf{x}_{vvv})^2)\, du\, dv\,, \\ Q_3 &= \int_\Omega (\mathbf{x}_{uvv})^2 + (\mathbf{x}_{vuu})^2\, du\, dv, \end{aligned} \tag{5.2}$$

see [**21**] for more details and for related references. However, these 'energy' functionals are not invariant, as they depend on the parameterization of the surface and of the data. In order to overcome this dependency, Greiner [**15,16**] has introduced *data–dependent functionals:* After choosing an auxiliary parameterized surface $\mathbf{x}_0$ (called the reference surface) which specifies the shape of the desired surface $\mathbf{x}$, consider the functional which is based on the corresponding Beltrami operator

$$Q_4 = \int (\mathrm{grad}_{\mathbf{x}_0}\mathbf{x} \cdot \mathrm{grad}_{\mathbf{x}_0}\mathbf{x})_{\mathbf{x}_0}\, d\omega_{\mathbf{x}_0}. \tag{5.3}$$

That is, the metric ('inner geometry') of the reference surface is used in order to approximate the exact non–linear functionals by simpler ones. During the approximation process, the approximation surface of the step $k$ is used as a reference surface for the following step $k+1$. This procedure can be interpreted as a Newton–Raphson iteration, whereas in each step the nonlinear functional $\int_\Omega (H_\mathbf{x}^2)\, d\omega$ (with $H$ as mean curvature of $\mathbf{x}$) is locally approximated by the quadratic functional (5.3). Since this functional converges fast to the nonlinear one, the approximation surface minimizes the nonlinear functional. In each step of the iteration, the new surface is computed simply by solving a linear system.

For fairing a given surface $\mathbf{x}(u,v)$ an efficient, automatic, and fast iterative method has been developed by Hadenfeld [**17,18**]. The main idea is the following: in order to minimize the value of an energy constraint, one modifies only one suitably chosen control point $\tilde{\mathbf{d}}_{r,s}$ in each step and keeps the remaining ones. Iterating this fairing step gives the final result. In addition, in order to keep a prescribed continuity between the original and the faired surface, the control points at the boundaries of the surface may be kept.

Now we describe this method in somewhat more detail. Consider the situation after a certain number of iteration steps; the initial surface $\mathbf{x}(u,v)$ with the control points $\mathbf{d}_{i,j}$ has been modified to $\overline{\mathbf{x}}(u,v)$ with the control points $\overline{\mathbf{d}}_{i,j}$. After the next step we get a new surface $\tilde{\mathbf{x}}(u,v)$ with the representation

$$\tilde{\mathbf{x}}(u,v) = \sum_{\substack{i=0 \\ (i,j)\neq(r,s)}}^{m} \sum_{j=0}^{n} \overline{\mathbf{d}}_{i,j} M_i(u) N_j(v) + \tilde{\mathbf{d}}_{r,s} M_r(u) N_s(v), \tag{5.4}$$

where the indices $(r,s)$ specify the control point which is to be modified. The minimum of the energy functional $Q_p$, $p \in \{1,2,3\}$, with respect to $\tilde{\mathbf{d}}_{r,s}$ is determined by the normal equation

$$\frac{\partial Q_p}{\partial \tilde{\mathbf{d}}_{r,s}} = 0. \tag{5.5}$$

These equations are linear, as we consider only quadratic functionals, $p \in \{1,2,3\}$. Then, an explicit formula for the new position of the control point $\tilde{\mathbf{d}}_{r,s}$ can be derived,

$$\tilde{\mathbf{d}}_{r,s} = \sum_{\substack{i=i_0 \\ (i,j)\neq(r,s)}}^{i=i_1} \sum_{j=j_0}^{j=j_1} \overline{\mathbf{d}}_{i,j}\, \gamma_{i,j} \qquad \left(\sum \gamma_{i,j} = 1\right) \tag{5.6}$$

with certain summation limits $i_0, i_1, j_0, j_1$. The constant coefficients $\gamma_{i,j}$ can be computed by integrals over the B-Spline basis functions and their derivatives [**17**].

In most applications of surface fairing, a prescribed tolerance $\delta$ between the old and the smoothed surface has to be guaranteed. The distance of the control points is an upper bound of the deviation between the old surface $\mathbf{x}$ and the modified surface $\tilde{\mathbf{x}}$, cf. [**32**]. Hence, Hadenfeld [**17**] uses the error measure

$$|\mathbf{d}_{r,s} - \tilde{\mathbf{d}}_{r,s}| \leq \delta \tag{5.7}$$

instead of $|\mathbf{x}(u,v) - \tilde{\mathbf{x}}(u,v)| < \delta$. If the modified control point $\tilde{\mathbf{d}}_{r,s}$ fulfills (5.7), then it is used as the new control $\tilde{\mathbf{d}}^*_{r,s}$ of the faired surface. Otherwise we determine the new location $\tilde{\mathbf{d}}^*_{r,s}$ as follows. As shown in [**17**], the iso–surfaces of the energy functionals $Q_p$, $p \in \{1,2,3\}$, are concentric spheres with centered at $\tilde{\mathbf{d}}_{r,s}$. Hence, the point $\mathbf{d}^*_{r,s}$

$$\tilde{\mathbf{d}}^*_{r,s} = \mathbf{d}_{r,s} + \delta \frac{\tilde{\mathbf{d}}_{r,s} - \mathbf{d}_{r,s}}{|\tilde{\mathbf{d}}_{r,s} - \mathbf{d}_{r,s}|} \tag{5.8}$$

solves the constrained problem $Q_p(\mathbf{d}_{r,s}) \to \min$ subject to (5.7); it is chosen as the new control point.

In each step of the iterative algorithm, we modify the control point which leads to the biggest improvement of the fairing functional $Q_p$. Hadenfeld [**17**] calls the potential improvement of a control point,

$$z_{r,s} = Q_p(\overline{\mathbf{d}}_{r,s}) - Q_p(\tilde{\mathbf{d}}^*_{r,s}), \tag{5.9}$$

its 'ranking number'. These numbers are calculated for all control point which are free for modification, and the resulting list is sorted. Clearly, the whole ranking list has to be calculated only in the very first step of the iteration, otherwise only a local update is required. Modifying the control point $\overline{\mathbf{d}}_{r,s}$ influences all ranking numbers $z_{i,j}$ with indices $|i-r| \leq d$, $|j-s| \leq d$, where $d$ is the degree of the B–Spline surface.

As an example, Figure 5.1 shows a part of a motor hood before and after fairing. The fairness of both surfaces is visualized by its isophotes. Irregularities in the isophotes point to undesirable behaviour (left), whereas the smooth flow of the isophotes (right) indicates a fair surface.

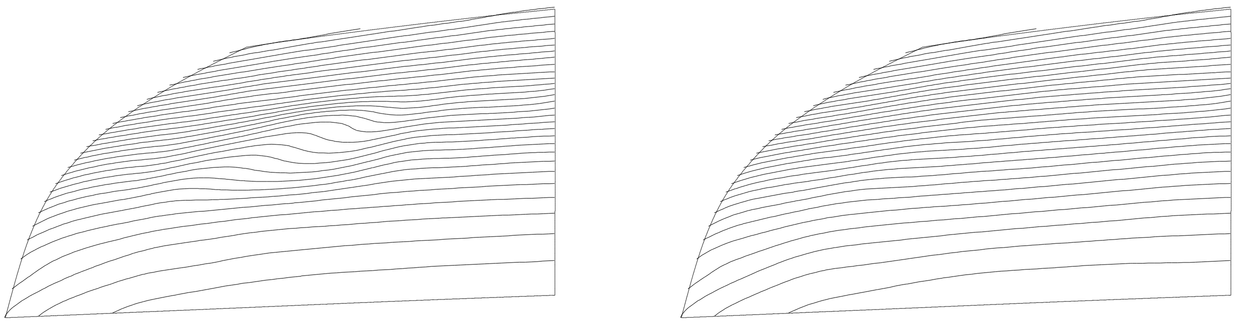
Figure 5.1: A part of a motor hood before (left) and after (right) fairing.

Now we consider the second fairing problem: approximation of a given cloud of unorganized points $\mathbf{p}_i$ $(i = 0, \dots, P)$, possibly with arbitrary boundaries and holes in the interior, by a fair parametric TP B-Spline surface. A straightforward way to solve this problem would be to use a least–squares algorithms by minimizing the squared error $\mathcal{L}^*$, cf. (2.6). In many cases, however, this method would fail as the holes in the data set may lead to 'gaps' in the coefficient matrix of the corresponding normal equations. Therefore, we use the modified error functional

$$\mathcal{L}^{**} = \mathcal{L}^* + \lambda \sum_k \alpha_k Q_k\,, \tag{5.10}$$

with the design parameters $\alpha_k \geq 0$ $(\sum_k \alpha_k = 1)$, certain energy functionals $Q_k$, and a positive weight $\lambda \in \mathbf{R}$. In order to get linear and therefore fast schemes, we assume that the energy functionals $Q_p$ are quadratic in the unknown control points, such as $Q_1, \dots, Q_3$ in (5.2). The energy functionals will 'close' the 'gaps' in the coefficient matrix of the minimization problem $\mathcal{L}^{**} \to \min$, cf. (5.10). Differentiating (5.10) leads to the necessary condition

$$\frac{\partial \mathcal{L}^{**}}{\partial \mathbf{d}_{j,k}} = \frac{\partial \mathcal{L}^{*}}{\partial \mathbf{d}_{j,k}} + \lambda \sum \alpha_p \frac{\partial Q_p}{\partial \mathbf{d}_{jk}} = 0\,, \tag{5.11}$$

which may be rewritten as a linear system $\tilde{A}\,\mathbf{d} = \mathbf{b}$, where the components of the unknown control points are collected in a vector $\mathbf{d}$.

The crucial step in parametric surface fitting is the assignment of suitable parameter values $(u_i, v_i)$ to the given points $\mathbf{p}_i$. The quality of the final approximation surface heavily depends on the choice of an appropriate parameterization. This parameterization also defines the topology of the data, corresponding to the location of the parameter values in the parametric domain of the surface, see [ **16** ]. In the literature, many proposals and heuristics can be found to compute appropriate initial parameterizations for the purpose of parametric surface fitting [ **12,16,21,22** ]. Most of these methods guarantee a valid parameterization, but not an optimal one. That is, the distance vectors between the data and the corresponding points on the surface are generally not perpendicular to the surface. In addition, most parameterization methods assume a special topological structure of the data, which cannot be expected in the general case.

In the sequel we describe an iterative method which adapts the parameterization to the data. Note that our method to provide a reasonable parameterization may fail if the geometry of the points is too complicated.

Dietz [ **9,10** ] has developed an iterative method which produces a fair approximation surface. His method can be seen as simulation of the deep–drawing process of metal sheets. He starts with a least–squares fitting plane. An initial parameterization can simply be obtained by orthogonal projection of the points onto this plane. In the subsequent iteration steps the plane is successively deformed until it reaches the final shape. In each iteration step, the new surface is found by solving the minimization problem $\mathcal{L}^{**} \to \min$, where the value of the weight $\lambda$ decreases in each step, thus leading to more flexibility of the surface. In addition, the parameterization of the data is modified after each step; it is adapted to the current surfaces with the help of the so–called parameter correction, see [ **21** ]. That is, an optimal parameterization (orthogonal distance vectors) is obtained by minimizing the total error sum with respect to the parameters $(u_i, v_i)$. The necessary conditions yield the 2-dimensional non–linear equations

$$\begin{aligned} (\mathbf{p}_i - \mathbf{x}(u_i, v_i)) \cdot \mathbf{x}_u(u_i, v_i) &= 0\,, \\ (\mathbf{p}_i - \mathbf{x}(u_i, v_i)) \cdot \mathbf{x}_v(u_i, v_i) &= 0\,, \end{aligned} \tag{5.12}$$

for each point $\mathbf{p}_i$. They can be solved efficiently with a damped Gauss–Newton method with correction terms

$$\begin{pmatrix} \Delta u \\ \Delta v \end{pmatrix} = \begin{pmatrix} \mathbf{x}^2 & \mathbf{x}_u \mathbf{x}_u \\ \mathbf{x}_u \mathbf{x}_v & \mathbf{x}_v^2 \end{pmatrix}^{-1} \begin{pmatrix} (\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{x}_u \\ (\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{x}_v \end{pmatrix}. \tag{5.13}$$

Clearly, parameter correction will generally converge to a local minimum. But since we adapt the parameterization after each small deformation of the surface, we get a more global character than for traditional parameter correction (as described in [ **21** ]). As an additional benefit of this process, it produces approximately isometric parameterizations. Unlike the majority of the available approximation methods, Dietz's method does not need a sophisticated initial parameterization to start with; suitable parameter values are automatically computed during the iterative algorithm. Figure 5.2 shows a sequence of surfaces minimizing the energy functional $Q_1$ while deforming a plane towards the points (left upper corner) which can be said to serve as a mould.

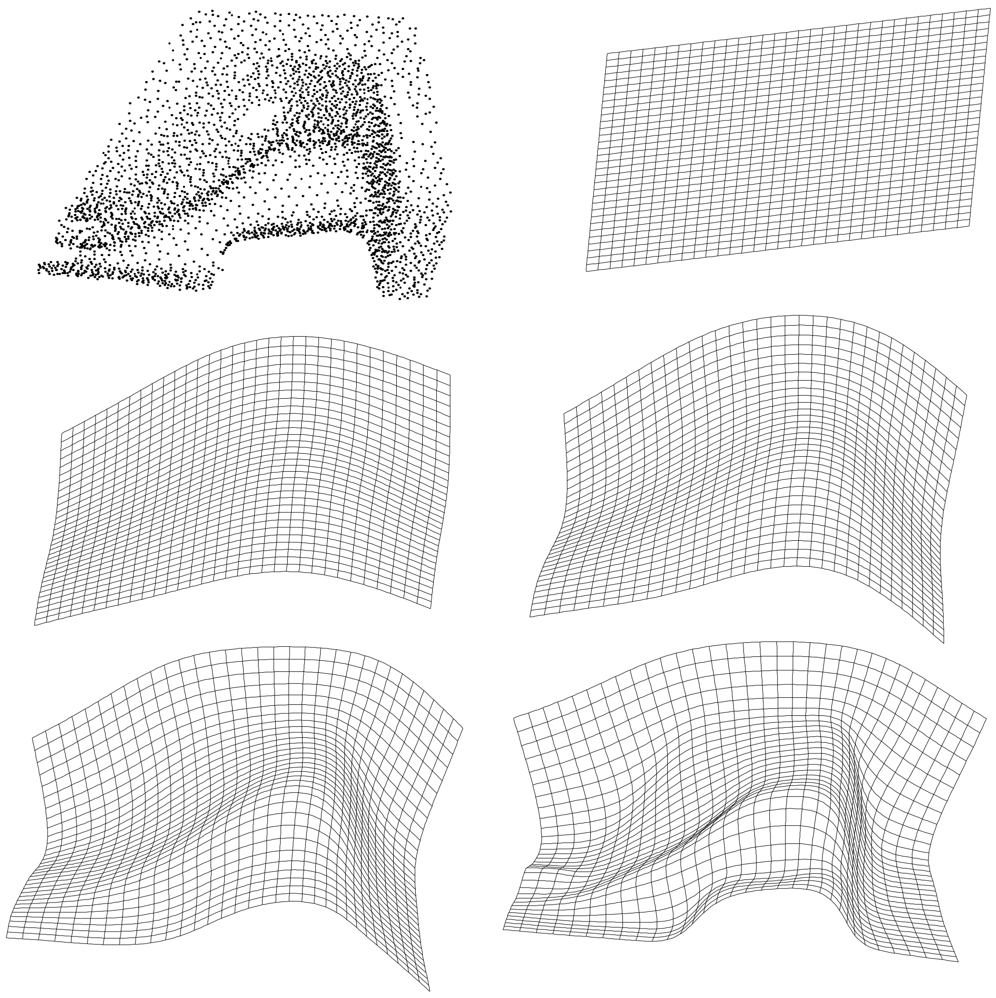
Figure 5.2. A sequence of surfaces which are generated by Dietz's method.

For the approximation of the set of points in Fig. 5.2, free boundaries of the surface patch have been used. In order to restrict the approximation surface to the domain which is given by a set of points, additional trimming curves describing the boundaries of the point sets are needed, cf. [ **8,9,10** ]. Fig. 5.3 shows the complete (trimmed) solution of the approximation surface of the set of points in Fig. 5.2.

Besides fairness, additional constraints on normals and derivatives may be used for geometric surface design. For instance, in the automotive industry, the quality of a surface is often judged by the distribution of reflection lines. These curves are determined by the

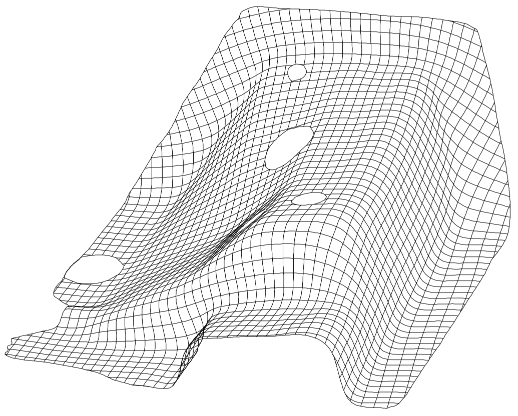

Figure 5.3. Trimmed approximation surfaces as approximation of the set of points in Figure 5.2.

normal vector field of the given surface. If a user wants to prescribe a desired flow of reflection lines, he can specify a suitable field of normal vectors $\mathbf{n}_i$ ($|\mathbf{n}_i| = 1$) in certain points $\mathbf{p}_i$ ($i = 0, \ldots, P$) of the desired surface. As another motivation for prescribing normal vectors, one may wish to guarantee approximate $G^1$-continuity at trimmed surface boundaries.

In order to avoid nonlinearity, Dietz [**9,10**] has introduced the quadratic functional

$$Q_5 = \sum_{i=1}^{M} (\mathbf{n}_i \cdot \mathbf{x}_{r_i})^2 + (\mathbf{n}_i \cdot \mathbf{x}_{s_i})^2 \tag{5.14}$$

with

$$\mathbf{x}_{r_i} \mid \mathbf{x}_{s_i} = \left( \frac{\mathbf{x}_{u_i}}{|\hat{\mathbf{x}}_{u_i}|} \pm \frac{\mathbf{x}_{v_i}}{|\hat{\mathbf{x}}_{v_i}|} \right) \Big/ \left| \frac{\hat{\mathbf{x}}_{u_i}}{|\hat{\mathbf{x}}_{u_i}|} \pm \frac{\hat{\mathbf{x}}_{v_i}}{|\hat{\mathbf{x}}_{v_i}|} \right| , \tag{5.15}$$

where $\hat{\mathbf{x}}_{u_i}$, $\hat{\mathbf{x}}_{v_i}$ denote the derivatives of the approximation surface from the previous iteration step. $Q_5$ can be interpreted as $\sin^2 \alpha_i$ where $\alpha_i$ is the angle between the given surface normal and the surface normal of the approximation surface in a point $\mathbf{p}_i$. The definition of the surface tangent vectors $\mathbf{x}_{r_i}$ and $\mathbf{x}_{s_i}$ leads to 'approximate invariance' with respect to the parameterization [**10**]. The functional $Q_5$ is quadratic in the control points, hence linear algorithms can be derived from (5.14). Using the angles $\alpha_i$ directly, by contrast, we would obtain non–linear procedures which are more difficult to deal with.

## 6. Convex parametric surfaces

We give a survey on available convexity conditions for parametric surfaces and discuss their applications to surface fitting and surface design.

Probably the very first convexity criterion for parametric TP Bézier surfaces is due to Schelske's Ph.D. thesis [**21,33**]. If all meshes of the control net are parallelograms and all control points belong to the boundary of the convex hull of the control net, then the surface patch is guaranteed to be convex. However, this condition is of little practical use at it is only satisfied by convex translational surface patches.

Zhou [ **38** ] discusses convexity conditions for parametric triangular Bézier surfaces. His conditions lead to a system of inequalities whose left–hand sides are polynomials of degree 6 in the components of the control points. In addition, a stronger sufficient conditions is presented which leads to polynomial inequalities of degree 3.

Cao and Hua [ **3** ] derive a sufficient convexity criterion for parametric triangular Bézier surface patches of degree 2. Their approach is based on the distribution of the Gaussian curvature of such a patch.

A recent manuscript by Koras and Kaklis derives similar convexity conditions for parametric tensor–product B–spline surfaces [ **28** ]. They derive a system of polynomial inequalities for the components of the control points that guarantees convexity. The left–hand sides of these inequalities are polynomials of degree 6 in the components of the control points.

An approximate method for removing convexity flaws from tensor–product B–spline surfaces has been developed by Kaklis and Koras [ **27** ]. The algorithm tries to modify a small number of control points of a given non–convex surface patch such that the surface becomes convex. This modification is based on the following nice observations.

The possible locations of a control point $\mathbf{d}_{i,j}$ for which the Gaussian (resp. mean) curvature at some fixed point $\mathbf{x}(u_0, v_0)$ vanishes, are called the parabolic (resp. minimal) loci of $\mathbf{d}_{i,j}$ with respect to $(u_0, v_0)$, where $(u_0, v_0)$ is assumed to be within the support of the corresponding TP B–spline basis function. (Minimal surfaces are characterized by vanishing mean curvature, whereas vanishing Gaussian curvature characterizes parabolic surface points.) The parabolic loci of a control point with respect to a surface point turn out to form a quadric, whereas the minimal loci form a cubic surface. Based on these results, Kaklis and Koras derive an optimization–based technique to remove shape flaws (non–convex regions) in the neighbourhood of one (or more) fixed point(s) $\mathbf{x}(u_0, v_0)$. As demonstrated by the examples, it seems to be possible to handle data from industrial applications with this approach.

A construction of linear sufficient convexity conditions for parametric Bézier surface patches has been developed in [ **23** ]. With the help of a so–called *reference surface* (which specifies the expected shape) it is possible to generate a system of linear inequalities for the control points that implies the desired shape properties. Unlike the case of bivariate functions (see Section 4), however, convex parametric surface do not form a convex set in some linear space. Thus, it is impossible to approximate the full set of convex patches by an inscribed convex polyhedron with any desired accuracy. Nevertheless it is still possible to circumscribe a convex polyhedron to any given interior point (which corresponds to the reference surface). The details of this construction are described in [ **23** ], where also an application to shape preserving surface modification (‘lifting’) is described. A shape–preserving surface fitting procedure for parametric surfaces has been presented in [ **25** ].

## REFERENCES

[ **1** ] Andersson, R. (1996), Surface design based on brightness intensity or isophotes – theory and practice. In: *Advanced Course on FAIRSHAPE* (J. Hoschek, P. Kaklis, eds.), Teubner, Stuttgart, 131–143.

[**2**] Brunnett, G.; Hagen, H.; Santarelli, P. (1993), Variational design of curves and surfaces. *Surveys on Mathematics for Industry* **3**, 1–27.

[**3**] Cao, Y.; Hua, X.-J. (1991), The convexity of quadratic parametric triangular Bernstein-Bézier surfaces, *Comput. Aided Geom. Design* **8**, 1–6.

[**4**] Carnicer, J. M.; Dahmen, W. (1992), Convexity preserving interpolation and Powell–Sabin elements, *Comput. Aided Geom. Design* **9**, 279–289.

[**5**] Carnicer, J.M.; Floater, M.S. (1996), Piecewise linear interpolants to Lagrange and Hermite convex scattered data, *Numer. Algorithms* **13**, 345-364.

[**6**] Chang, G.-Z.; Feng, Y.-Y. (1984), An improved condition for the convexity of Bernstein–Bézier surfaces over triangles, *Comput. Aided Geom. Design* **1**, 279–283.

[**7**] Dierckx, P. (1993), *Curve and Surface Fitting with Splines*, Oxford, Clarendon Press.

[**8**] Dietz, U. (1996), B–Spline Approximation with Energy Constraints. In: *Advanced Course on FAIRSHAPE* (J. Hoschek, P. Kaklis, eds.), Teubner, Stuttgart, 229–240.

[**9**] Dietz, U. (1998), Fair surface reconstruction from point clouds. In: *Mathematical Methods for Curves and Surfaces II* (M. Dæhlen, T. Lyche, L.L. Schumaker, eds.), Vanderbilt University Press, Nashville, 79–86.

[**10**] Dietz, U. (1998), Geometrie–Rekonstruktion aus Meßpunktwolken mit glatten B–Spline–Flächen. PhD thesis, TU Darmstadt; Shaker, Aachen.

[**11**] Fletcher, R. (1990), *Practical methods of optimization*, Chichester, Wiley-Interscience.

[**12**] Floater, M.S. (1997), Parametrization and smooth approximation of surface triangulations, *Comput. Aided Geom. Design* **14**, 231-250.

[**13**] Floater, M.S. (1997), A counterexample to a theorem on Powell–Sabin elements, *Comput. Aided Geom. Design* **14**, 383–385.

[**14**] Forsey, D.R.; Bartels, R.H. (1995), Surface fitting with hierarchical splines, *ACM Trans. on Graphics* **14**, 134–161.

[**15**] Greiner, G. (1994), Variational design and fairing of spline surfaces, *Computer Graphic Forum* **13**, 143–154.

[**16**] Greiner, G.; Hormann, K. (1997), Interpolating and approximating scattered 3D data with hierarchical tensor product B–splines. In: *Surface Fitting and Multiresolution Methods* (A. Le Méhauté, C. Rabut, L.L. Schumaker, eds.), Vanderbilt University Press, Nashville, 163–172.

[**17**] Hadenfeld, J. (1995), Local Energy Fairing of B-Spline Surfaces. In: *Mathematical Methods for Curves and Surfaces* (M. Dæhlen, T. Lyche, L.L. Schumaker, eds.), Vanderbilt University Press, Nashville, 213–212.

[**18**] Hadenfeld, J. (1998), Iteratives Glätten von B-Spline–Kurven und B-Spline–Flächen. PhD thesis, TU Darmstadt; Shaker, Aachen.

[**19**] Halstead, M.; Barsky, B.; Klein, S.; Mandell, R. (1996), Reconstructing curved surfaces from specular reflection patterns using spline surface fitting of normals. In: *SIGGRAPH'96 proceedings* (H. Rushmeier, ed.), Addison Wesley, 335–342.

[**20**] Hoschek, J.; Dietz, U. (1996), Smooth B–Spline Surface Approximation to Scattered Data. In: *Reverse Engineering* (J. Hoschek, W. Dankwort, eds.), Teubner, Stuttgart, 143–152.

[**21**] Hoschek, J.; Lasser, D. (1993), *Fundamentals of computer aided geometric design*, Wellesley MA, AK Peters.

[**22**] Hoschek, J.; Schneider, F.-J. (1992), Approximate spline conversion for integral and rational Bézier- and B–spline surfaces. In: *Geometry Processing for Design and Manufacturing* (R.E. Barnhill ed.), SIAM, 45–86.

[**23**] Jüttler, B. (1997), Linear convexity conditions for parametric tensor–product Bézier surface patches. In: *The Mathematics of Surfaces VII* (T.N.T. Goodman, R. Martin, eds.), Winchester, Information Geometers, 189–208.

[**24**] Jüttler, B. (1997), Surface fitting using convex tensor-product splines, *J. Comput. Appl. Math.* **84**, 23-44.

[**25**] Jüttler, B. (1998), Convex surface fitting with tensor–product Bézier surfaces. In: *Mathematical Methods for Curves and Surfaces II* (M. Dæhlen, T. Lyche, L.L. Schumaker, eds.), Nashville, Vanderbilt University Press, 263–270.

[**26**] Jüttler, B. (1998), Computational methods for parametric discrete $\ell_1$ and $\ell_\infty$ curve fitting, *Int. J. of Shape Modeling*, to appear.

[**27**] Kaklis, P.D.; Koras, G.D. (1997), A quadratic–programming method for removing shape–failures from tensor–product B–spline surfaces, NTU Athens, Dept. of Naval Architecture, Technical report TR–SDL–CAGD–1/97.

[**28**] Koras, G.D.; Kaklis, P.D. (1997), Convexity conditions for parametric tensor–product B–spline surfaces, NTU Athens, Dept. of Naval Architecture, manuscript.

[**29**] Loos, J. (1997), Konstruktion von Flächen mit vorgegebenen Krümmungseigenschaften und Anwendungen in der Augenoptik. PhD thesis, Universität Erlangen.

[**30**] Loos, J.; Greiner, G.; Seidel, H.-P. (1997), Computing Surface Geometry from Isophotes and Reflection Lines, Technical Report 4/97, Universität Erlangen.

[**31**] Loos, J.; Greiner, G.; Seidel, H.-P., (1998), A variational approach to progressive lens design. *Computer-aided design* **8**, 595–602.

[**32**] Schaback, R. (1993), Error estimates for approximations from control nets. *Comput. Aided Geom. Design* **10**, 57–66.

[**33**] Schelske, H.-J. (1984), *Lokale Glättung segmentierter Bézierkurven und Bézierflächen*, Dissertation, TH Darmstadt, 1984.

[**34**] Schumaker, L.L. (1981), *Spline Functions: Basic Theory*, New York, Wiley.

[**35**] Tazeroualti, M. (1993), Designing a progressive lens. In: *Curves and surfaces in geometric design* (P.J. Laurent, A. Le Mehaute, L.L. Schumaker, eds.), AK Peters, Wellesley MA, 467–474.

[**36**] Vanderbei, R. (1992), *LOQO Version 1.08,* available via anonymous ftp from `elib.zib-berlin.de` at `/pub/opt-net/software/loqo/1.08.`

[**37**] Willemans, K.; Dierckx P. (1994), Surface fitting using convex Powell–Sabin splines, *J. Comput. Appl. Math.* **56**, 263–282.

[**38**] Zhou, C.-Z. (1990), On the convexity of parametric Bézier triangular surfaces, *Comput. Aided Geom. Design* **7**, 459–463.