# Least–squares fitting of algebraic spline curves via normal vector estimation

Bert Jüttler

Darmstadt University of Technology, Dept. of Mathematics,
Schlossgartenstr. 7, 64289 Darmstadt, Germany

**Summary.** We describe an algorithm for fitting implicitly defined algebraic spline curves to given planar data. By simultaneously approximating points and associated normal vectors, we obtain a method which is both computationally simple, as the result is obtained by solving a system of linear equations, and geometrically invariant. The initial result of the curve fitting procedure is improved by iteratively adjusting the associated field of normal vectors. It is planned to generalize the approach to algebraic spline surfaces.

## 1 Introduction

This paper is devoted to the reconstruction of planar curves from scattered data. The (possibly noisy) data are assumed to be generated be taking sample points of a certain planar object. We construct a planar curve that approximately matches the shape of the data. Our approach is based on implicitly defined algebraic spline curves. More precisely, the planar curve is obtained as the zero contour of a bivariate real spline function.

The present study is intended to serve as a prototype for planned research on methods for reconstructing algebraic spline surfaces from scattered data in 3–space. However, the reconstruction of planar shapes from measurement data is also an interesting subject in its own right (see [13] for references), with various applications, e.g. in medical imaging.

Compared to the parametric representations, such as NURB (**N**on–**U**niform **R**ational **B**-spline, see [7,9]) curves and surfaces, implicitly defined curves and surfaces offer several advantages:

- The curve and surface fitting procedures do not need the estimation of auxiliary parameter values which are associated with the given data. In the parametric case, by contrast, these parameters have a strong influence to the resulting shape, and it may be difficult to generate appropriate values, in particular for more complex shapes (see e.g. [9]).
- It is possible to bypass the initial polygonalization resp. triangulation step that is required in many parametric curve and surface fitting procedures.
- They can be used to define planar domains resp. solids; the point membership can easily be decided by evaluating the sign of the generating real function.

- Relatively simple algorithms are available for computing the intersection(s) with straight lines, as this reduces to a one–dimensional root–finding problem. This is particularly advantageous for the visualization with the help of ray–tracing methods.
- Additional shape constraints can easily be added to the curve resp. surface fitting procedure. For instance, the convexity of the resulting curve resp. surface can be guaranteed by the convexity of the underlying real function; see [11] for suitable linear convexity criteria. This leads to relatively simple constrained optimization problems, as the feasible domain is a convex set. Convexity criteria for truly parametric representations, by contrast, are far more complicated, cf. [12].

On the other hand, implicitly defined curves and surfaces cause some extraneous difficulties which need to be taken care of.

- The visualization and evaluation of the surface needs special contour–finding algorithms, such as 'marching cubes', see [9].
- In order to exchange data with commercial CAD systems, the implicitly defined curves and surfaces have to be converted into the industrial NURBS standard, cf. [3].
- Approximation or interpolation schemes may produce algebraic curves or surfaces which consist of several disconnected components. This needs special treatment in order to avoid the resulting problems.

Methods for curve and surface fitting with implicitly defined algebraic curves and surfaces have been discussed in an enormous number of publications, and it is virtually impossible to give a complete survey. We list only a few references which had a major influence to the present research.

Pratt [14], Taubin [15], and Bajaj et al. [2] describe methods for implicit curve and surface fitting. The methods are based on the algebraic distance, combined with suitable normalizations of the unknown coefficients. For instance, Pratt's 'simple fit' method [14] keeps the value of one of the coefficients, leading to a linear normalization constraint. The results, however, are not geometrically invariant; they depend on the choice of the coordinates. Taubin's method [15] constrains the sum of the squared gradients at the data sites. This leads to a geometrically invariant quadratic normalization.

The approximants are computed by solving constrained quadratic programming problems (minimization of a quadratic objective function subject to linear, resp. quadratic, constraints). In the case of quadratic constraints (which are required in order to get geometrically invariant results), the solutions are found by numerically solving generalized eigenvalue problems, i.e., by computing the eigenvector which is associated with the smallest eigenvalue of a certain matrix. The dimension of the matrix is equal to the number of unknown coefficients. Based on experimental evidence, Pratt's and Taubin's methods have been compared by Umasuthan and Wallace [16].

In a number of publications, Bajaj and various co–authors [5,1] have developed implicit algebraic surfaces into a powerful tool for reconstructing curves

and surfaces from measurement data ('reverse engineering'). Their approach focuses on the use of low–degree patches whose coefficients satisfy certain sign conditions, in order to guarantee the desired topology of the result.

Recently, Werghi et al. [17] have developed an incremental framework, incorporating geometric constraints (such as orthogonality), for fitting implicitly defined geometric primitives, such as planes and quadrics.

The present paper describes a novel approximation technique for implicitly defined algebraic spline curves. By simultaneously approximating points and associated normal vectors, we obtain a method which is both computationally simple (as the result is obtained by solving a system of linear equations) and geometrically invariant. This will help to overcome some of the limitations of the normalization–based algebraic curve fitting procedures.

In the remainder of this paper we describe an algorithm that fits an implicitly defined algebraic spline curve to given planar data. More precisely, consider a set of points

$$\boldsymbol{p}_i = (p_{i,1}\ p_{i,2}) \in \mathbb{R}^2; \quad i = 1, \ldots, N; \tag{1}$$

in the plane. The approximating curve is to be described as the zero contour of a bivariate real function $z = f(x_1, x_2)$. The function $f$ is chosen as a piecewise polynomial function, leading to a approximating algebraic spline curve.

The approximating curve is constructed in several steps, as follows. Firstly we estimate normal vectors $\boldsymbol{n}_i$ which are associated with the given data. In the second step we fit an algebraic spline curve, matching both the data and the associated normals. Using the additional normal vector information, we are able to avoid the various normalizations that are used for fitting algebraic curves and surfaces. Finally, in order to improve the results, one may update the associated normals and iterate the curve fitting procedure.

## 2 Estimating associated normal vectors

As the first step of the curve fitting procedure, we generate unit normal vectors

$$\boldsymbol{n}_i = (n_{i,1}\ n_{i,2}) \in \mathbb{S}^1; \quad i = 1, \ldots, N; \tag{2}$$

from the unit circle $\mathbb{S}^1 = \{\ \boldsymbol{z} \in \mathbb{R}^2 \mid \|\boldsymbol{z}\| = 1\ \}$ which are associated with the given data (1). The estimation of normal vectors from measurement data is a standard problem in scattered data approximation. In many applications, the normal vectors can be generated directly from additional information accompanying the data. For instance, if the points $\boldsymbol{p}_i$ are generated from a certain image, such as in Computer Tomography, then the normal vectors could be chosen as the (normalized) gradients of the color function. If no additional information is available, however, then the normal vectors have

to be estimated directly from the data. We summarize the basic idea of the method. More details can be found in [13], see also [8] for the 3D case.

In order to associate a unit normal vector $\boldsymbol{n}_i$ with one the points $\boldsymbol{p}_i$, we fit a simple curve to the neighborhood of that point, see Figure 1 for a schematic illustration.

Firstly we compute the associated local line of regression $L_i$. It is found by minimizing the weighted sum of squared distances of the points $(\boldsymbol{p}_j)_{j=1,\dots,N}$ from the line $L_i$. The weight function $w = w(r)$ is chosen such that influence of a point $\boldsymbol{p}_j$ to the line of regression $L_i$ decreases with its distance $r = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|$ from $\boldsymbol{p}_i$. Possible choices include the characteristic function $w(r) = \chi_{[0,h]}$ (i.e., taking only points $\boldsymbol{p}_j$ with maximum distance $h$ from $\boldsymbol{p}_i$ into account), or the exponential weight function $w(r) = \exp(-r^2/H^2)$, with certain suitable constants $h, H$. The resulting optimization problem has a quadratic objective function which minimized is subject to the quadratic equality constraint $\|\boldsymbol{n}^*\|^2 = 1$, where $\boldsymbol{n}^*$ is the unit normal vector of $L_i$. It can be solved by computing the eigenvectors of a certain $2 \times 2$ matrix.

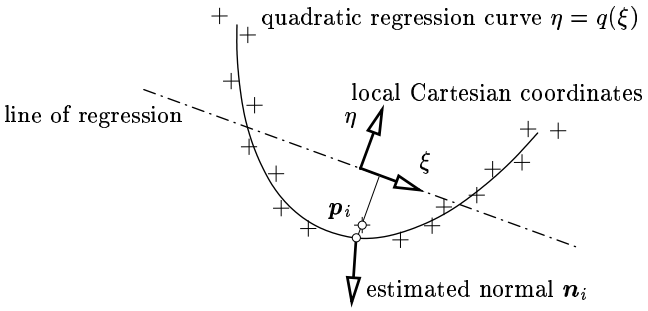



**Fig. 1.** Estimating the normal vector from the given data (shown as crosses).

In the second step we choose a new Cartesian coordinate system whose axis of abscissae is parallel to $L_i$, and fit a local quadratic regression curve to the data. This curve is the graph of a quadratic polynomial with respect to the new Cartesian system. Its coefficients are computed minimizing the weighted sum of squared residuals, leading to a $3 \times 3$ system of linear equations.

The direction $\pm\boldsymbol{n}_i$ of the unit normal vector which is associated with $\boldsymbol{p}_i$ is then chosen such that it is parallel to the normal of the quadratic polynomial, evaluated at the abscissa of $\boldsymbol{p}_i$. In order to get useful results, however, we have to guarantee that neighboring normal vectors have the same orientation. That is, if two points $\boldsymbol{p}_i$, $\boldsymbol{p}_j$ are relatively close together, then the inner product $\boldsymbol{n}_i \cdot \boldsymbol{n}_j$ of the associated normal vectors is expected to be positive. In order to choose the orientation of the normals, one could compute the minimum spanning tree of the data and use the resulting neighborhood information, see [13]. As a cheaper alternative, one may simply choose an appropriate rectangular grid, and select – for each of the resulting quadrangular

**Fig. 2.** Estimation of associated normal vectors from scattered data (examples with different levels of noise).

cells – a representative of the points which are contained in it (if such points exist). The orientation of the normals that are associated with the representatives is chosen according to the neighborhood structure of the grid. Then, the orientation of the remaining normals is chosen according to that of the representatives. Clearly, the success of this simple method depends on the appropriate specification of the grid size, requiring some user interaction.

Two examples are shown in Figure 2. We have sampled 100 points from a planar shape and added some noise to it, with two different levels. The plots show the data $(\boldsymbol{p}_i)_{i=1,\ldots,N}$, along with the estimated normal vectors $(\boldsymbol{n}_i)_{i=1,\ldots,N}$. The normal vectors have been estimated by considering – for each point – the 10 nearest neighbors, and fitting the line of regression and a quadratic regression curve to them.

## 3 Implicit algebraic tensor–product spline curves

The implicitly defined algebraic spline curve is described by a tensor–product spline function of (bi–) degree $d$ ($d \geq 2$). The segments of the resulting spline are algebraic curves of order $2d$. The use of tensor–product spline offers several advantages, including simple implementation, simple conditions for global smoothness, simple evaluation, sufficient flexibility and refinability (e.g. using hierarchical B-spline representations, see [4])

The approximating curve is described as the zero contour $f(x, y) = 0$ of the tensor–product spline function

$$f(x, y) = \sum_{(i,j)\in\mathcal{I}} M_i(x)\, N_j(y)\, c_{i,j} \tag{3}$$

with the real coefficients (control points) $c_{i,j}$, where $\mathcal{I}$ is a certain index set, see below. The basis functions $(M_i(x))_{i=1..m}$ and $(N_j(y)_{j=1..n})$ are B-splines of degree $d$ with respect to the knot sequences $\mathcal{X} = (\xi_i)_{i=1..m+1}$ and

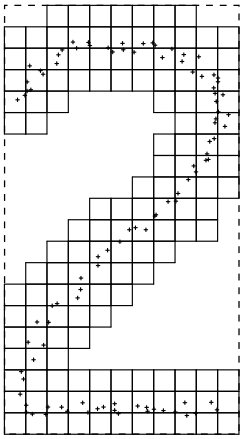


**Fig. 3.** The domain $\mathcal{D}$ of the tensor–product spline function (3) consists of all cells that contain data, and the neighbouring cells (within the enlarged bounding box of the data, shown as dashed lines).

$\mathcal{Y} = (\eta_j)_{j=1..n+1}$, see [6,9]. We choose the first and last $d+1$ knots of $\mathcal{X}$ resp. $\mathcal{Y}$ as the abscissas resp. ordinates of the slightly enlarged (by blowing it up) bounding box of the data $(\boldsymbol{p}_k)_{k=1,\ldots,N}$. The remaining inner knots are chosen equidistant.

The two knot sequences define a partition of the (slightly enlarged) bounding box $[\xi_1, \xi_{m+d+1}] \times [\eta_1, \eta_{n+d+1}]$ into rectangular cells. Clearly, not all of these cells necessarily contain data points. We choose the domain $\mathcal{D}$ of the tensor–product spline function (3) to be the union of all cells that contain data, and of the neighboring cells, cf. Figure 3. The index set $\mathcal{I}$ in (3) is chosen such that the summation includes all products $M_i(x)\ N_j(y)$ that do not vanish on $\mathcal{D}$,

$$\begin{aligned} \mathcal{I} = \{\ (i,j) \mid \exists\ k \in \{1,\ldots,N\}, r \in \{1,\ldots,m\}, s \in \{1,\ldots,n\} : \\ M_r(p_{k,1})\ N_s(p_{k,2}) \neq 0\ \wedge\ \max\{|i-r|, |j-s|\} \leq 1\ \} \end{aligned} \tag{4}$$

Clearly, the domain $\mathcal{D}$ of the tensor–product spline function (3) depends on the choice of the Cartesian coordinate system. An geometrically invariant choice could be obtained by fitting a line of regression (which serves as one of the coordinate axes) to all data.

In order to keep the algebraic order as low as possible (if this is desired), it would be more appropriate to chose a bivariate spline function of total degree $d$ (e.g., defined with respect to a union of triangles, such as Powell–Sabin spline or simplex spline, see [9]), rather than a tensor–product one. Within the tensor–product setting, a lower algebraic degree could be guaranteed by introducing additional side–conditions. For instance, a biquadratic tensor–product spline, along with the linear constraints $f_{xxy} = 0$ and $f_{xyy} = 0$, gives $C^1$ spline curves of algebraic order 2, i.e., $C^1$ conic splines. Adding these constraints, however, leads to a highly redundant representation.

The approximation method which is described in the next section can be applied to any implicit algebraic spline curve, not only to curves in tensor–

product representation. Using tensor–products we obtain a method which is computationally simple, but whose results depend on the choice of the system of coordinates (unless we choose the coordinates with the help of the line of regression, as outlined earlier). This dependency, however, is caused only by the choice of the space of functions, not by the approximation method. For instance, applying the approximation method to bivariate polynomials of total degree $d$ gives geometrically invariant results.

## 4 Fitting curves to points with associated normals

Let $\boldsymbol{c} = (c_{i,j})_{(i,j)\in\mathcal{I}}$ be the vector obtained by gathering all B-spline coefficients (control points) of the approximating algebraic spline curve, in a suitable ordering. Its components will be computed by minimizing a quadratic objective function $F = F(\boldsymbol{c})$, which is formed as a certain linear combination of four terms.

### 4.1 Approximating the data

The first two terms deal with the data $(\boldsymbol{p}_i)_{i=1,\ldots,N}$ and with the associated normal vectors $(\boldsymbol{n}_i)_{i=1,\ldots,N}$. The given data are approximated by minimizing the sum of the squared 'algebraic distances' (see [14,15]),

$$L(\boldsymbol{c}) = \sum_{i=1}^{N} [\, f(p_{i,1}, p_{i,2}) \,]^2. \tag{5}$$

The sum $L$ is a homogeneous quadratic form of the unknown control points $\boldsymbol{c}$. Hence, it is minimized by the null vector $c_{i,j} = 0$, leading to the tensor–product spline function $f(x,y) \equiv 0$. In order to get results which are more useful, one has to introduce a normalization. Various normalizations have been described in the literature [2,14,15], most of them based on a suitable norm in the coefficient space. Our approach is based on the simultaneous approximation of the data and the associated normal vectors, by minimizing – in addition to $L$ – the sum

$$\begin{aligned} M(\boldsymbol{c}) &= \sum_{i=1}^{N} \| \nabla f(p_{i,1}, p_{i,2}) - \boldsymbol{n}_i \|^2 \\ &= \sum_{i=1}^{N} (f_x(p_{i,1}, p_{i,2}) - n_{i,1})^2 + (f_y(p_{i,1}, p_{i,2}) - n_{i,2})^2. \end{aligned} \tag{6}$$

That is, the gradients $\nabla f = (f_x,\, f_y)$ of the tensor–product spline function $f(x,y)$ at the given data $\boldsymbol{p}_i$ are to match the estimated normal vectors $\boldsymbol{n}_i$. Consequently, as the given normals $\boldsymbol{n}_i$ are unit vectors, the algebraic distances in (5) can be expected to approximate the real distances.

### 4.2 Tension terms

Although the minimization of weighted linear combination of $L$ and $G$ leads to good results in many cases, it may produce approximating spline curves that split into several disconnected components. There are several possibilities to address this well–known problem of algebraic curve and surface fitting. For instance, based on the signs of the coefficients, one may derive criteria which guarantee the desired topology of the result, see [5].

As we wish to compute the solution by solving a system of linear equations, we use the simpler approach of adding suitable 'tension terms' that pull the approximating curve towards a simpler shape. If the the tension terms have a sufficiently strong influence (which is governed by certain weights), then the approximating curve has the desired topology.

A global tension term is given by the quadratic functional

$$G(\boldsymbol{c}) = \iint_{\mathcal{D}} f_{xx}^2 + 2\, f_{xy}^2 + f_{yy}^2 \; dx\; dy \tag{7}$$

which measures the deviation of $f(x,y)$ from a linear function. Hence, by increasing the influence of this tension term, the resulting spline curve gets closer to a straight line.

In addition to the global tension term we have tested a data–dependent tension term also. It is based on the following simple observation.

**Lemma 1.** *Consider a tensor–product polynomial $p(x,y)$. If the quadratic functional (with respect to the coefficients of the polynomial)*

$$Q = \iint_{[0,1]^2} \sum_{(i,j)\in\mathcal{L}} \left[ \left(\frac{\partial}{\partial x}\right)^i \left(\frac{\partial}{\partial y}\right)^j p(x,y) \right]^2 dx\; dy \tag{8}$$

*vanishes, with the index set*

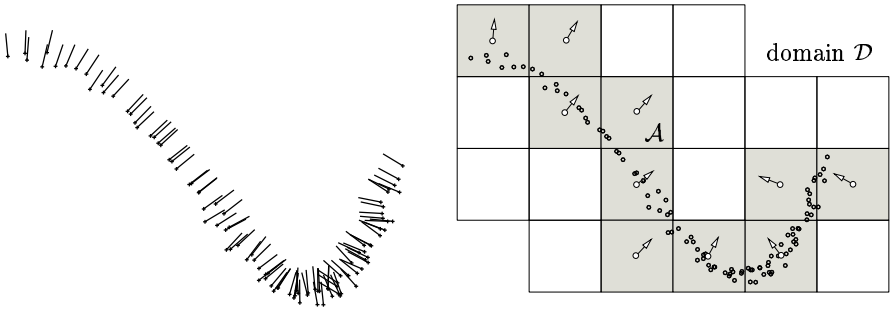
$$\mathcal{L} = \{(i,j) \in \mathbb{Z}_+^2 \mid j \geq 2 \;\vee\; (j = 1 \wedge i \geq 1)\}, \tag{9}$$

*and $p_y(x,y) \neq 0$ holds, then the level curves $p(x,y) = d$ are graphs of polynomials $y = q_d(x)$.*

*Proof.* If a tensor–product polynomial $p(x,y)$ satisfies

$$\left(\frac{\partial}{\partial x}\right)^i \left(\frac{\partial}{\partial y}\right)^j p(x,y) = 0 \quad \forall (i,j) \in \mathcal{L} \tag{10}$$

and $p_y(x,y) \neq 0$, then it has the form $p(x,y) = C\, y + q(x)$ with a real constant $C \neq 0$. Consequently, the level curves $p(x,y) = d$ are simply graphs of the polynomials $y = (d - q(x))/C$. The linear equations (10) are equivalent to the quadratic equation $Q = 0$. □




(a) (b)

**Fig. 4.** Generating the data–dependent tension term $D(\boldsymbol{c})$. The data with the associated normal vectors (a) and the set $\mathcal{A}$ of cells with the normals $\boldsymbol{n}_c$ (b).

By replacing the partial derivatives in (8) with *directional* derivatives with respect two perpendicular unit vectors, one obtains a condition for the level curves to be graphs of polynomials with respect to another system of coordinates. We obtain the data–dependent tension term $L$ by collecting the left–hand sides of (8), and using suitable directional derivatives, whose directions depend on the data.

More precisely, let $\mathcal{A}$ be the set of all cells (belonging to the grid defined by the knot sequences $\mathcal{X}$ and $\mathcal{Y}$ of the spline function) which contain at least one of the data $(\boldsymbol{p}_i)_{i=1,\ldots,N}$. The data–dependent tension term is given by

$$D(\boldsymbol{c}) = \sum_{C \in \mathcal{A}} \iint_C \sum_{(i,j) \in \mathcal{L}} \left[ \left( \frac{\partial}{\partial \boldsymbol{n}_c^{\perp}} \right)^i \left( \frac{\partial}{\partial \boldsymbol{n}_c} \right)^j f(x,y) \right]^2 dx\, dy = 0, \quad (11)$$

where $\boldsymbol{n}_c$ is the normalized average of the associated normal vectors of the points in $C$, and $\boldsymbol{n}_c^{\perp}$ obtained by a clockwise rotation of $\pi/2$.

An example is presented in Figure 4. Its left part (a) shows the data $(\boldsymbol{p}_i)_{i=1,\ldots,N}$ and the associated normals $(\boldsymbol{n}_i)_{i=1,\ldots,N}$. The right figure (b) visualizes the domain $\mathcal{D}$ of the tensor–product spline function, the cells $\mathcal{A}$ that contain data (marked in grey), and the associated normal vectors $\boldsymbol{n}_c$ (shown as arrows) which are used for generating $D$.

The tension term $D$ measures – for each cell $C$ – the deviation of the algebraic curves $f(x,y) = d$ from graphs of polynomials with respect to an adapted system of Cartesian coordinates. By minimizing it one may obtain the desired topology of the solution, without getting results which are 'too flat'. An example is given below.

### 4.3 Computing the solution

The approximating implicit spline curve is found by minimizing the weighted linear combination

$$F(\boldsymbol{c}) = L(\boldsymbol{c}) + w_1\, M(\boldsymbol{c}) + w_2\, G(\boldsymbol{c}) + w_3\, D(\boldsymbol{c}) \quad \rightarrow \quad \text{Min}, \tag{12}$$

see (5), (6), (7) and (11), with certain non–negative weights $w_1$, $w_2$, and $w_3$. As this leads to a quadratic objective function of the unknown control points $\boldsymbol{c} = (c_{i,j})_{(i,j)\in\mathcal{I}}$, the solution can be found by solving the sparse linear system of equations

$$\frac{\partial}{\partial c_{i,j}} F(\boldsymbol{c}) = 0, \quad (i,j) \in \mathcal{I}, \tag{13}$$

with the help of appropriate methods from numerical linear algebra. Alternatively, one may also compute the solution of (12) by generating an overconstrained system of linear equations, and computing a least–squares solution to it using QU factorization, see e.g. [6].

The weight $w_1$ controls the influence of the estimated normal vectors $\boldsymbol{n}_i$ to the resulting curve. The weights $w_2$ and $w_3$ act as tension parameters, see Section 4.5 for an example.

As the main benefit from the simultaneous approximation of points $\boldsymbol{p}_i$ and normal vectors $\boldsymbol{n}_i$, the approximating spline curve is found by solving a system of linear equations. The required computations have the complexity $\mathcal{O}(h^2)$ (without taking the sparsity into account), where $h = |\mathcal{I}|$ is the number of coefficients.

The traditional, normalization–based methods [14,15], by contrast, generate the solution of the curve fitting problem by solving a generalized eigenvalue problem, requiring iterative numerical procedures for computing the result. According to [15], the complexity is $\mathcal{O}(h^3)$.

### 4.4 Existence and uniqueness

Under certain mild assumptions, the problem (12) can easily be shown to be uniquely solvable.

**Proposition 1.** *If the weights $w_1$ and $w_2$ are positive, and $w_3 \neq 0$, then the quadratic optimization problem (12) has a unique solution. Consequently, the rank of the square coefficient matrix of the linear system (13) equals the number of coefficients $|\mathcal{I}|$.*

*Proof.* It is easy to see that the quadratic functionals (5), (6), (7) and (11) are convex. That is,

$$Q(\,\lambda\,\boldsymbol{c}^{(1)} + (1-\lambda)\,\boldsymbol{c}^{(2)}\,) \;\leq\; \lambda\, Q(\boldsymbol{c}^{(1)}) + (1-\lambda)\, Q(\boldsymbol{c}^{(2)}), \;\; 0 < \lambda < 1, \tag{14}$$

holds for all of them, $Q \in \{L, N, G, D\}$. However, they are not strictly convex, as both sides of (14) may be equal. This case is characterized by certain conditions to the difference vector of the coefficients, as follows.

Consider a tensor–product spline function $g(x, y)$ with the coefficients $\boldsymbol{c}^{(1)} - \boldsymbol{c}^{(2)} = (c_i^{(1)} - c_i^{(2)})_{i=1,\ldots,N}$, where both sides of (14) are assumed to be equal. If $Q = L$ then these coefficients represent a function $f(x, y)$ such that the associated zero contour interpolates all data, i.e.

$$g(p_{i,1}, p_{i,2}) = 0 \quad \text{holds for} \quad i = 1, \ldots, N. \tag{15}$$

If $Q = N$ then the difference coefficients represent a function $f(x, y)$ whose gradient vanishes at all data, i.e.

$$\nabla g(p_{i,1}, p_{i,2}) = 0 \quad \text{holds for} \quad i = 1, \ldots, N. \tag{16}$$

Finally, if $Q = G$ then they simply describe a linear function $g(x, y)$, i.e.,

$$g_{xx}(x, y) = g_{xy}(x, y) = g_{yy}(x, y) = 0 \quad \text{holds for} \quad (x, y) \in \mathcal{D}. \tag{17}$$

Thus, the linear combination (12) with positive coefficients $w_1$, $w_2$ (and non–negative $w_3$) is even strictly convex, as only the zero function ($\boldsymbol{c}^{(1)} - \boldsymbol{c}^{(2)} = \boldsymbol{0}$) simultaneously fulfills the three conditions (15), (16), (17). This proves the assertion. □

Note that the tension term $G$ is the only part of the objective function (12) that acts on all cells of the domain $\mathcal{D}$. The other parts act only on cells which contain data. Consequently, the choice $w_2 = 0$ will always produce a singular system (13), if at least one of the products $M_i(x)\, N_j(y)$ vanishes at all data $(x, y) = (p_{i,1}, p_{i,2})$. This may happen very easily, as the domain of the spline function $f(x, y)$ consists of all cells containing data, and the neighboring cells.

### 4.5 Examples

As a first example, we approximate the two sets of data shown in Figure 2 by implicit algebraic tensor–product spline curves of degree $d = 2$. We choose knot vectors which generate a grid of $3 \times 2$ cells, leading to $|\mathcal{I}| = 20$ unknown B-spline coefficients $c_{i,j}$. The weights of the objective function are chosen as $w_1 = 1$, $w_2 = w_3 = 0.001$. The resulting curves are shown in Figure 5.

In order to compare these results with the ones described in the next section, we provide the $\ell_1$ (total error), $\ell_2$ (least–squares), and $\ell_\infty$ (maximum error) norm of the residual errors

$$\boldsymbol{R} = (R_i)_{i=1,\ldots,N} \quad \text{with} \quad R_i = \inf\{\, \|\boldsymbol{p}_i - \boldsymbol{z}\| \;\mid f(z_1, z_2) = 0 \,\}, \tag{18}$$

measuring the orthogonal distances from the data $\boldsymbol{p}_i$ to the approximating curve. The orthogonal distances have been generated by computing the foot-points of the data with the help of Newton–Raphson iterations. The curves

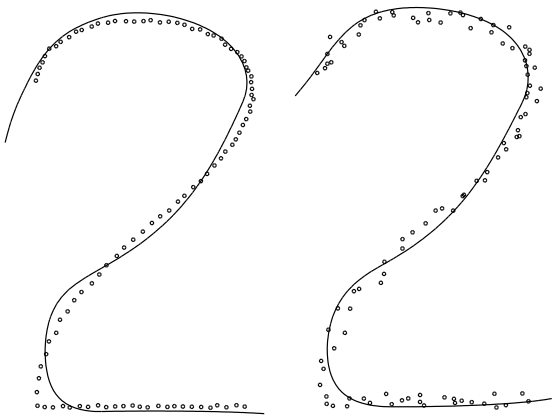


**Fig. 5.** Approximation of the data from Figure 2 by algebraic tensor–product spline curves of degree 2.

from Figure 5 give the following results, where the residuals $\boldsymbol{R}_l$ resp. $\boldsymbol{R}_r$ refer to the example on the left– resp. right–hand side:

$$\begin{aligned} &\|\boldsymbol{R}_l\|_1 = 5.286, \ \|\boldsymbol{R}_l\|_2 = 0.623, \ \|\boldsymbol{R}_l\|_\infty = 0.204, \\ &\|\boldsymbol{R}_r\|_1 = 5.959, \ \|\boldsymbol{R}_r\|_2 = 0.732, \ \|\boldsymbol{R}_r\|_\infty = 0.178. \end{aligned} \tag{19}$$

The next figure visualizes the influence of the two different tension terms. Here, the knot vectors generate a grid of $6 \times 4$ cells, see Figure 4 for the resulting domain $\mathcal{D}$. This leads to $|\mathcal{I}| = 45$ unknown B-spline coefficients $c_{i,j}$. The curves in Figures 6a resp. b have been generated by adding global resp. data–dependent tension. We chose the weights of the objective function such that the ratios of the approximation and tension parts

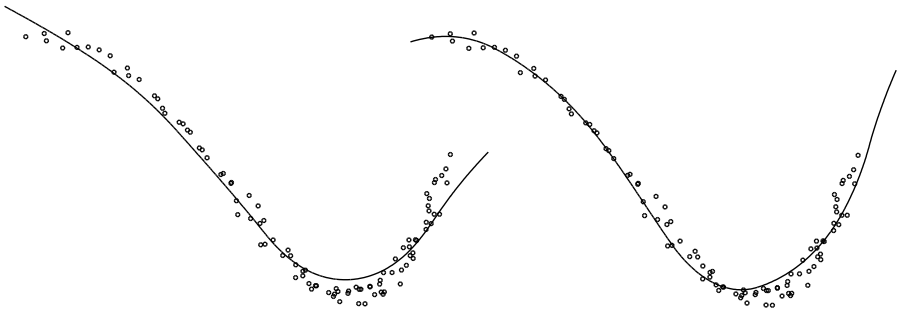
$$\frac{L(\boldsymbol{c}) + w_1 \, M(\boldsymbol{c})}{w_2 \, G(\boldsymbol{c}) + w_3 \, D(\boldsymbol{c})} \tag{20}$$

have the same values ($\approx 1.2$) for both curves. Thus, both tension terms have approximately the same influence to the result. The data–dependent tension term gives the better result, as the global tension term tends to straighten out the resulting curve.

Clearly, as a slight modification of the method, one could also apply tension locally, to specific parts of the curve only. This could be achieved simply by using weight functions $w_2 = w_2(x, y)$ and $w_3 = w_3(x, y)$ which depend on the coordinates $x$ and $y$.

## 5 Updating the objective function

By appropriately updating the objective function, the result of the curve fitting procedure can be improved. This leads to an iterative curve fitting procedure.



(a) global tension
($w_1 = 0.1,\ w_2 = 1.0,\ w_3 = 0.0$)

(b) data–dependent tension
($w_1 = 0.1,\ w_2 = 0.001,\ w_3 = 1.8$)

**Fig. 6.** Comparison of the tension terms. Approximating curve with global tension (a) and with data–dependent tension (b). Both tension terms have approximately the same influence.

### 5.1 Weighted least–squares

Clearly, the algebraic distances (5) do not represent the real distances between the data and the approximating spline curve. In order to obtain a better approximation, it is a standard approach in implicit curve and surface fitting to use – instead of (5) – the weighted least–squares sum

$$L^*(\boldsymbol{c}) = \sum_{i=1}^{N} [\ \omega_i\ f(p_{i,1}, p_{i,2})\ ]^2. \tag{21}$$

with certain positive weights $\omega_i$, leading to the modified optimization problem

$$F(\boldsymbol{c}) = L^*(\boldsymbol{c}) + w_1\ M(\boldsymbol{c}) + w_2\ G(\boldsymbol{c}) + w_3\ D(\boldsymbol{c}) \quad \rightarrow \quad \text{Min.} \tag{22}$$

Ideally, the weights would have the values

$$\omega_i = \frac{1}{\|\nabla f(p_{i,1}, p_{i,1})\|}, \tag{23}$$

where $f$ is the solution to (22). Then, the weighted least–squares sum (21) would be a good approximation of the squared Euclidean distances, see [15]. The following 'reweight procedure' is described in [15]; it is similar in spirit to various related methods of weighted least–squares.

(Iterative fitting procedure)

1. Compute an initial solution $f^{(1)}(x, y)$ by choosing the weights $\omega_i = 1$, $i = 1, \ldots, N$. Let $j = 1$.
2. Choose the weights $\omega_i = 1/\|\nabla f^{(j)}(p_{i,1}, p_{i,1})\|$ and compute the new solution $f^{(j+1)}$ of (22).

3. If a certain termination criterion is satisfied (e.g., the improvement of Euclidean distances and/or the change of weights stays below a certain limit) then stop, $f^{(j+1)}$ is the approximate solution. Otherwise increase $j$ by 1 and continue with Step 2.

In the original algorithm [15], the computation of the next solution in Step 1 resp. 2 requires the solution of a generalized eigenvalue problem. By simultaneously approximating points and associated normal vectors, we have to solve a system of linear equations instead.

### 5.2 Adjusting the normal vectors

A similar iterative procedure can be applied to the normal vectors $(\boldsymbol{n}_i)_{i=1,\dots,N}$ which are associated with the given data. It is analogous to the method of 'parameter correction' for parametric curve and surface fitting, see [9, Section 4.4.3]. There, the new parameters of a point are chosen according to the location of its nearest neighbor on the approximating curve resp. surface; they are then used for computing an improved solution.

Similarly, we may use the gradients of the first approximation at the data $\boldsymbol{p}_i$ in order to adjust the normal vectors $\boldsymbol{n}_i$. Clearly, the lengths of these vectors will be non–uniform in general. In order to avoid contraction to a sequence of null vectors, we scale them such that the sum of the squared lengths equals $\sqrt{N}$,

$$\sum_{i=0}^{N} \|\boldsymbol{n}_i\|^2 = N. \tag{24}$$

This leads to the following modified Step 2 of the iterative fitting procedure from the previous section:

2′. Choose the normal vectors according to

$$\boldsymbol{n}_i = \frac{\sqrt{N}}{\sqrt{\sum_{k=1}^{N} \|\nabla f^{(j)}(p_{k,1}, p_{k,2})\|^2}} \nabla f^{(j)}(p_{i,1}, p_{i,2}); \quad i = 1, \dots, N; \tag{25}$$

and continue with the original Step 2.

Clearly, also the termination criterion in Step 3 has to be modified, by taking the change of the associated normal vectors into account.

One may identify the normal vectors

$$\boldsymbol{n}^* = (\boldsymbol{n}_i)_{i=1,\dots,N} \tag{26}$$

with a single vector $\boldsymbol{n}^* \in \mathbb{R}^{2N}$. The vectors satisfying (24) form the sphere $\sqrt{N}\,\mathbb{S}^{2N-1}$ with radius $\sqrt{N}$ and center at the origin. The adjustment of the

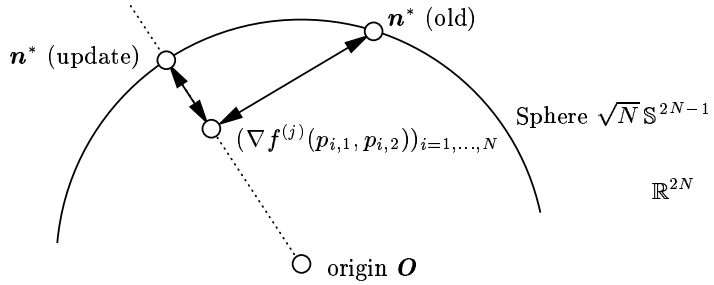



**Fig. 7.** Geometric interpretation of the normal vector adjustment. The arrows '$\leftrightarrow$' represent the value of the normal vector part $M(\boldsymbol{c})$ of the objective function.

normal vectors can be interpreted as follows. The new normals $\boldsymbol{n}^*$ are found by intersecting the ray

$$\lambda\,(\nabla f^{(j)}(p_{i,1}, p_{i,2}))_{i=1,\ldots,N}, \quad \lambda \in \mathbb{R}_+, \tag{27}$$

with the sphere, see Figure 7. The normal vector part $M(\boldsymbol{c})$ of the objective function measures the distance of the gradients (which are also considered as a point in $\mathbb{R}^{2N}$) and the (previous) normals $\boldsymbol{n}^*$. Consequently, the adjustment of the normal vectors (25) always leads to a smaller value of $M(\boldsymbol{c})$, as the new normals $\boldsymbol{n}^*$ have the minimum possible distance from the gradients.

### 5.3 Examples

The modified (by using Step 2′ instead of Step 2) iterative curve fitting procedure has been applied to the data of the previous examples, see Figures 2 and 5. This leads to the normal vectors $(\boldsymbol{n}_i)_{i=1,\ldots,N}$ and the approximating curves which are shown in Figures 8 and 9. These result have been obtained after 12 iterations of the fitting procedure. In order to facilitate the comparison, the initial curves have been drawn in grey, along with the result after 12 steps.

Again we we provide the $\ell_1$ (total error), $\ell_2$ (least–squares), and $\ell_\infty$ (maximum error) norm of the residual errors, compare with the initial values (19):

$$\begin{aligned} &\|\boldsymbol{R}_l\|_1 = 2.807, \;\; \|\boldsymbol{R}_l\|_2 = 0.381, \;\; \|\boldsymbol{R}_l\|_\infty = 0.155, \\ &\|\boldsymbol{R}_r\|_1 = 4.613, \;\; \|\boldsymbol{R}_r\|_2 = 0.555, \;\; \|\boldsymbol{R}_r\|_\infty = 0.133. \end{aligned} \tag{28}$$

According to our numerical experience, the iterative adjustment of the normal vectors leads only to relatively small changes in the shape of the curves. Consequently, the initial estimates $\boldsymbol{n}_i$ seem to be fairly good. Also, the biggest improvement of the approximation errors is often achieved in the few first steps of the iterative fitting procedure, and the effects of the remaining steps are rather small. Figure 10 shows the $\ell_2$ norms of the residual error, and the angle between new and old normal vector $\boldsymbol{n}^* \in \mathbb{R}^{2N}$, for each of the 12 iteration steps of the previous example.

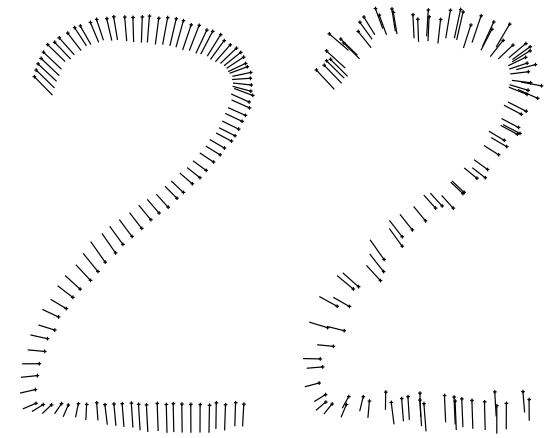


**Fig. 8.** Normal vectors obtained after 12 steps of the modified iterative curve fitting procedure, compare with the initial values in Figure 2.

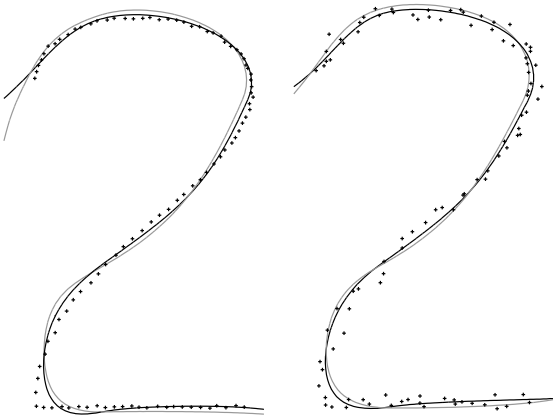


**Fig. 9.** Approximating curves obtained after 12 steps of the modified iterative curve fitting procedure. The initial curves (see also Figure 5) are shown in grey.

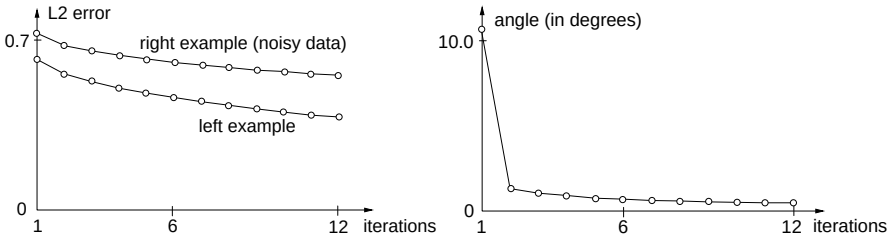



**Fig. 10.** The $\ell_2$ norm of the residual errors (left) and the angle (in degrees) between new and old normal vector(s) $\boldsymbol{n}^* \in \mathbb{R}^{2N}$ for the 12 iteration steps.

## Concluding remarks

We have described a novel technique for fitting implicitly defined algebraic spline curves to planar scattered data. By simultaneously approximating points and associated normal vectors, which are estimated from the data, one obtains a method which is both computationally simple, as the result is obtained by solving a system of linear equations, and geometrically invariant, as no auxiliary normalization of the spline coefficients is needed. Weighted least–squares and an iterative adjustment of the normal vectors have been used in order to improve the initial result.

As demonstrated in this paper, both the problems of implicit (algebraic) and parametric curve fitting can be dealt with by solving certain sequences of systems of linear equations. The fitting of parametric curves needs the estimation of auxiliary parameter values, which can then be iteratively improved via parameter correction. Analogously, curve fitting with implicit representations requires auxiliary normal vectors, which can then be adjusted, in order to obtain better results.

Also, the method presented in this paper can be seen as a contribution to methods for approximating dual data. That is, a curve resp. surface is to be generated by approximating tangents resp. tangent planes, rather than points, cf. [10]. Our curve fitting scheme deals with 'mixed' data, which are obtained by combining point and tangent information.

The success of the curve fitting scheme depends on the robustness of the method which is used for estimating the normal vectors from the data. The sensitivity of the result with respect to errors contained in the data should therefore be analyzed.

Future research will focus on implicitly defined algebraic spline surface. One the one hand, we plan to explore the use of low–degree spline surfaces, such as quadrics and cubicoids, for surface fitting applications. On the other hand, surfaces of higher order will be used in order for more complex objects, with potential applications in reverse engineering.

## References

1. Bajaj, C.L. (2000) Publications, available at `www.ticam.utexas.edu/CCV/papers/fhighlights.html`.
2. Bajaj, C.L.; Ihm, I.; Warren, J. (1993) Higher–Order Interpolation and Least–Squares approximation Using Implicit Algebraic Surfaces. ACM Transactions on Graphics **12**, 327–347.
3. Bajaj, C.L.; Xu, G. (1997) Spline approximations of real algebraic surfaces. J. Symb. Comput. **23**, 315-333
4. Forsey, D.; Bartels, R. (1995) Surface Fitting with Hierarchical Splines, ACM Transactions on Graphics **14**, 134–161.
5. Bernardini, F.; Bajaj, C.L.; Chen, J.; Schikore, D.R. (1999) Automatic Reconstruction of 3D CAD Models from Digital Scans. Int. J. Comp. Geom. Appl. **9**, 327–370.

6. Boehm, W.; Prautzsch, H (1993) Numerical methods. AK Peters, Wellesley (Mass.), and Vieweg, Braunschweig.
7. Farin, G. (1995) NURB curves and surfaces. AK Peters, Wellesley (Mass.).
8. Hoppe, H.; DeRose, T.; Duchamp, T.; McDonald, J.; Stuetzle, W. (1992) Surface reconstruction from unorganized points. Computer Graphics **26**, 71–78.
9. Hoschek, J.; Lasser, D. (1993) Fundamentals of Computer Aided Geometric Design. AK Peters, Wellesley (Mass.).
10. Hoschek, J.; Schwanecke, U. (1997) Interpolation and approximation with ruled surfaces. In: Cripps, R. (ed.) The Mathematics of Surfaces VIII, Information Geometers, Winchester, 213–232.
11. Jüttler, B. (1997) Surface fitting using convex tensor–product splines. J. Comp. Appl. Math. **84**, 23–44.
12. Koras, G.D.; Kaklis, P.D. (1999), Convexity conditions for parametric tensor–product B-spline surfaces. Adv. Comput. Math. **10**, 291–309.
13. Lee, I.-K. (2000), Curve reconstruction from unorganized points. Comput. Aided Geom. Des. **17**, 161–177.
14. Pratt, V. (1987) Direct Least–Squares Fitting of Algebraic Surfaces. ACM Computer Graphics **21** (Siggraph'87), 145–152.
15. Taubin, R. (1991) Estimation of Planar Curves, Surfaces, and Nonplanar Space Curves Defined by Implicit Equations with Applications to Edge and Range Image Segmentation. IEEE Trans. Pattern Analysis and Machine Intelligence **13**, 1115–1138.
16. Umasuthan, M.; Wallace, A.M. (1994) A comparative analysis of algorithms for fitting planar curves and surfaces defined by implicit polynomials. In: Fisher, R.B. (ed.) Design and applications of Curves and Surfaces (Mathematics of Surfaces V). Clarendon Press, Oxford, 495–514.
17. Werghi, N; Fisher, R.; Robertson, C.; Ashbrook, A. (1999) Object reconstruction by incorporating geometric constraints in reverse engineering. Comp. Aided Des. **31**, 363–399.