Computing Bivariate Splines in Scattered Data Fitting and the Finite-element Method

Larry L. Schumaker [1)]

Abstract. A number of useful bivariate spline methods are global in nature, i.e., all of the coefficients of an approximating spline must be computed at one time. Typically this involves solving a system of linear equations. Examples include several well-known methods for fitting scattered data, such as the minimal energy, least-squares, and penalized least-squares methods. Finite-element methods for solving boundary-value problems are also of this type. It is shown here that these types of globally-defined splines can be efficiently computed, provided we work with spline spaces with stable local minimal determining sets.

§1. Introduction

Bivariate splines defined over triangulations are important tools in several application areas including scattered data fitting and the numerical solution of boundary-value problems by the finite element method. Methods for computing spline approximations fall into two classes:

1) *Local methods*, where the coefficients of the spline are computed one at a time or in small groups,
2) *Global methods*, where all of the coefficients of the spline have to be computed simultaneously, usually as the solution of a single linear system of equations.

In this paper we focus on global methods, and in particular those that arise from minimizing a quadratic form, possibly with some constraints. The purpose of this paper is to show how such minimization problems can be efficiently solved for spline spaces that possess stable local minimal determining sets (see Sect. 2). In particular, we show how our approach applies to three commmonly used scattered data fitting methods: the minimal energy method, the discrete least-squares method, and the penalized least-squares method. In addition, we discuss how it works for solving boundary-value problems involving partial differential equations.

The standard approach to solving global variational problems involving piecewise polynomials on triangulations is to use Lagrange multipliers to enforce interpolation and smoothness conditions, see Remark 1. This results in a linear system

[1)] Department of Mathematics, Vanderbilt University, Nashville, TN 37240, larry.schumaker@vanderbilt.edu.

of equations for the coefficients of the spline. This approach can be used even when the dimension of the approximating spline space is not known. In this case the linear system may be singular, and we can only find an approximate solution. This leads to spline fits which only satisfy the desired smoothness conditions approximately. In addition, the Lagrange multiplier approach generally leads to very large systems of equations, which is its main disadvantage. In comparison, the method proposed here

- involves much smaller systems of equations,
- is generally much faster,
- produces spline fits lying in the prescribed spline spaces.

Our method can be used with spline spaces of any degree and smoothness, as long as they have stable local minimal determining sets. There are many examples of such spaces in [13]. We discuss the method for bivariate splines, but the approach applies to spherical splines and trivariate splines as well, see Remarks 2 and 3.

The paper is organized as follows. In Sect. 2 we recall some of the basic theory of bivariate splines, including the concepts of minimal determining sets and stable local bases. We also recall how to compute with bivariate splines using the Bernstein–Bézier representation. In Sect. 3 we discuss the computation of splines solving general quadratic minimization problems. Three explicit examples of such problems are described in Sects. 4–6, namely, minimal energy, discrete least-squares, and penalized least-squares spline fitting. The Galerkin method for solving elliptic boundary-value problems is treated in Sect. 7. Sect. 8 is devoted to numerical experiments to illustrate the performance of our method. In Sect. 9 we describe a useful algorithm for computing all of the Bernstein basis polynomials (or their directional derivatives) at a given point in an efficient manner. Finally, we collect several remarks in Sect. 10.

§2. Preliminaries

In working with bivariate splines, we follow the notation used in the book [13]. For convenience, we review some key concepts here. Given $d > r \geq 0$, and a triangulation $\triangle$ of a domain $\Omega \subset \mathbb{R}^2$, the associated space of bivariate splines of smoothness r and degree d is defined to be

$$\mathcal{S}_d^r(\triangle) := \{s \in C^r(\Omega) : s|_T \in \mathcal{P}_d,\ \text{all } T \in \triangle\}.$$

Here $\mathcal{P}_d$ is the $\binom{d+2}{2}$-dimensional space of bivariate polynomials of degree at most d. Given $r \leq \rho \leq d$, we also work with the superspline space

$$\mathcal{S}_d^{r,\rho}(\triangle) := \{s \in \mathcal{S}_d^r(\triangle) : s \in C^\rho(v),\ \text{all } v \in \mathcal{V}\},$$

where $\mathcal{V}$ is the set of vertices of $\triangle$, and where $s \in C^\rho(v)$ means that all polynomial pieces of s on triangles sharing the vertex v have common derivatives up to order ρ at v.

2.1. Bernstein–Bézier methods

We make use of the Bernstein–Bézier representation of splines. Given d and $\triangle$, let $\mathcal{D}_{d,\triangle} := \cup_{T\in\triangle}\mathcal{D}_{d,T}$ be the corresponding set of domain points, where for each $T := \langle v_1, v_2, v_3\rangle$,

$$\mathcal{D}_{d,T} := \{\xi^T_{ijk} := \frac{iv_1 + jv_2 + kv_3}{d}\}_{i+j+k=d}.$$

Then every spline $s \in \mathcal{S}^0_d(\triangle)$ is uniquely determined by its set of coefficients $\{c_\xi\}_{\xi\in\mathcal{D}_{d,\triangle}}$, and

$$s|_T := \sum_{\xi\in\mathcal{D}_{d,T}} c_\xi B^T_\xi,$$

where $\{B^T_\xi\}$ are the Bernstein basis polynomials associated with the triangle T.

Suppose now that $\mathcal{S}(\triangle)$ is a subspace of $\mathcal{S}^0_d(\triangle)$. Then a set $\mathcal{M} \subseteq \mathcal{D}_{d,\triangle}$ of domain points is called a minimal determining set (MDS) for $\mathcal{S}(\triangle)$ provided it is the smallest set of such points such that the corresponding coefficients $\{c_\xi\}_{\xi\in\mathcal{M}}$ can be set independently, and all other coefficients of s can be consistently determined from smoothness conditions, i.e., in such a way that all smoothness conditions are satisfied, see p. 136 of [13]. The dimension of $\mathcal{S}(\triangle)$ is then equal to the cardinality of $\mathcal{M}$. Clearly, $\mathcal{M} = \mathcal{D}_{d,\triangle}$ is a minimal determining set for $\mathcal{S}^0_d(\triangle)$, and thus the dimension of $\mathcal{S}^0_d(\triangle)$ is $n_V + (d-1)n_E + \binom{d-1}{2}n_T$, where n_V, n_E, n_T are the number of vertices, edges, and triangles of $\triangle$.

For each $\eta \in \mathcal{D}_{d,\triangle} \setminus \mathcal{M}$, let Γ_η be the smallest subset of $\mathcal{M}$ such that c_η can be computed from the coefficients $\{c_\xi\}_{\xi\in\Gamma_\eta}$ by smoothness conditions. Then $\mathcal{M}$ is called local provided there exists an integer ℓ not depending on $\triangle$ such that

$$\Gamma_\eta \subseteq \mathrm{star}^\ell(T_\eta), \qquad \text{all } \eta \in \mathcal{D}_{d,\triangle} \setminus \mathcal{M}, \tag{2.1}$$

where T_η is a triangle containing η. Recall that given a set $U \subset \Omega$, then $\mathrm{star}(U)$ is the set of triangles in $\triangle$ intersecting U, while $\mathrm{star}^\ell(U) := \mathrm{star}(\mathrm{star}^{\ell-1}(U))$. $\mathcal{M}$ is said to be stable provided there exists a constant K depending only on ℓ and the smallest angle in the triangulation $\triangle$ such that

$$|c_\eta| \le K \max_{\xi\in\Gamma_\eta} |c_\xi|, \qquad \text{all } \eta \in \mathcal{D}_{d,\triangle} \setminus \mathcal{M}. \tag{2.2}$$

2.2. Stable local bases

Suppose $\mathcal{M}$ is a stable local MDS for $\mathcal{S}(\triangle)$. For each $\xi \in \mathcal{M}$, let ψ_ξ be the spline in $\mathcal{S}(\triangle)$ such that $c_\xi = 1$ while $c_\eta = 0$ for all other $\eta \in \mathcal{M}$. Then the splines $\{\psi_\xi\}_{\xi\in\mathcal{M}}$ are clearly linearly independent and form a basis for $\mathcal{S}(\triangle)$. This basis is called the $\mathcal{M}$-basis for $\mathcal{S}(\triangle)$, see Sect. 5.8 of [13]. It is stable and local in the sense that for all $\xi \in \mathcal{M}$,

1) $\|\psi_\xi\|_\Omega \le K$,
2) $\operatorname{supp}\psi_\xi \subseteq \operatorname{star}^\ell(T_\xi)$, where T_ξ is a triangle containing ξ,

where ℓ is the integer constant in (2.1), and the constant K depends only on ℓ and the smallest angle in $\triangle$. Here and in the sequel, for any set $U \subset \mathbb{R}^2$, $\|\cdot\|_U$ denotes the ∞-norm over points in U. There are many spaces with stable local bases. For example, the spaces $\mathcal{S}_d^0(\triangle)$ have stable local bases with $\ell = 1$. The same is true for the superspline spaces $\mathcal{S}_{4r+1}^{r,2r}(\triangle)$ for all $r \ge 1$. There are also several families of macro-element spaces defined for all $r \ge 1$ with the same property, see [13].

2.3. Computational methods

We now list several useful techniques for working with splines numerically.

1) Using the well-known smoothness conditions for two polynomial patches to join together smoothly across an edge of a triangulation (see Theorem 2.28 of [13]), for each $\eta \in \mathcal{D}_{d,\triangle} \setminus \mathcal{M}$, we can find real numbers $\{a_\xi^\eta\}_{\xi\in\Gamma_\eta}$ such that for every spline $s \in \mathcal{S}(\triangle)$ with coefficients $\{c_\beta\}_{\beta\in\mathcal{M}}$,
$$c_\eta = \sum_{\xi\in\Gamma_\eta} a_\xi^\eta c_\xi. \tag{2.3}$$
Note that for spaces with locally supported bases, the number of nonzero a_ξ^η in (2.3) will generally be quite small. If we intend to use the spline space $\mathcal{S}(\triangle)$ for several fitting problems, these weights can be precomputed and stored.
2) To store a particular spline s in the space $\mathcal{S}(\triangle)$, we simply store its coefficient vector $c := (c_\xi)_{\xi\in\mathcal{M}}$. Recall that the length of this vector is just the dimension of $\mathcal{S}(\triangle)$.
3) To evaluate s at a point (x, y) in a triangle T, we use the formulae (2.3) to compute the B-coefficients of s corresponding to all domain points in T. Then we can use the well-known de Casteljau algorithm to find $s(x, y)$, see p. 26 of [13].
4) If we need to evaluate s at many points, for example to display the surface corresponding to s, it may be most efficient to compute and store all of the B-coefficients $\{c_\xi\}_{\xi\in\mathcal{D}_{d,\triangle}}$ of s using (2.3). The total number of B-coefficients is equal to the dimension of $\mathcal{S}_d^0(\triangle)$, which is $nco := n_V + (d-1)n_E + \binom{d-1}{2} n_T$, where n_V, n_E, n_T are the numbers of vertices, edges, and triangles of $\triangle$.

§3. Computing Splines Solving Quadratic Minimization Problems

Suppose $\mathcal{S}(\triangle) \subseteq \mathcal{S}_d^0(\triangle)$ is a space of splines with a stable local minimal determining set $\mathcal{M}$. Let $\{\psi_\xi\}_{\xi\in\mathcal{M}}$ be the associated $\mathcal{M}$-basis. In this paper we focus on variational spline problems where the coefficients of the desired spline are the solution of a system of equations of the form
$$\sum_{\xi\in\mathcal{M}} \langle\psi_\xi, \psi_\eta\rangle c_\xi = r_\eta, \qquad \eta \in \mathcal{M}, \tag{3.1}$$

where $\langle \cdot, \cdot \rangle$ is some appropriate inner-product. We suppose the inner-product is such that

$$\langle \psi, \phi \rangle = \sum_{T \in \triangle} \langle \psi|_T, \phi|_T \rangle, \qquad \text{all } \psi, \phi \in \mathcal{S}(\triangle). \tag{3.2}$$

Note that if $\mathcal{S}(\triangle)$ has a local basis, then only a small number of terms in (3.1) will be nonzero.

To find a coefficient vector $c := \{c_\xi\}_{\xi \in \mathcal{M}}$ satisfying (3.1), we have to compute the matrix

$$M := [\langle \psi_\xi, \psi_\eta \rangle]_{\xi, \eta \in \mathcal{M}}$$

and the vector $r = (r_\eta)_{\eta \in \mathcal{M}}$. This matrix will be sparse whenever $\mathcal{S}(\triangle)$ has a local basis. To describe an efficient algorithm for finding M, we make the following observation.

Lemma 3.1. *For every $\xi \in \mathcal{M}$ and every triangle $T \in \triangle$,*

$$\psi_\xi|_T = \sum_{\eta \in \mathcal{D}_{d,T}} a_\xi^\eta B_\eta^T, \tag{3.3}$$

where a_ξ^η are the weights in formula (2.3).

Proof: The spline ψ_ξ is the spline with $c_\xi = 1$, and $c_\beta = 0$ for all other $\beta \in \mathcal{M}$. Fix a domain point η in T. Then by (2.3), the coefficient c_η of ψ_ξ is given by $c_\eta = a_\xi^\eta$. $\square$

Algorithm 3.2.

```
For all $T \in \triangle$,
  • Compute the matrix $[\langle B_\alpha^T, B_\beta^T \rangle]_{\alpha,\beta \in \mathcal{D}_{d,T}}$.
  • For all $\xi, \eta \in \mathcal{M}$,
        $\langle \psi_\xi, \psi_\eta \rangle \leftarrow \langle \psi_\xi, \psi_\eta \rangle + \sum_{\alpha,\beta \in \mathcal{D}_{d,T}} a_\xi^\alpha a_\eta^\beta \langle B_\alpha^T, B_\beta^T \rangle$.
```

This algorithm works by looping through the triangles of $\triangle$. The entries in M are obtained by an accumulation process. If the value r_η on the right-hand side of (3.1) involves an inner-product of a given function with ψ_η, it can be computed by adding an additional accumulation step to the algorithm. We discuss this algorithm in more detail for the particular variational spline methods treated below.

§4. Minimal Energy Interpolating Splines

In this section we discuss the minimal energy method for computing an interpolating spline. Suppose we are given values $\{f_i\}_{i=1}^{n_d}$ associated with a set of $n_d \geq 3$ abscissae $\mathcal{A} := \{(x_i, y_i)\}_{i=1}^{n_d}$ in the plane. The problem is to construct a smooth function s that interpolates this data in the sense that

$$s(x_i, y_i) = f_i, \quad i = 1, \ldots, n_d. \tag{4.1}$$

To solve this problem, suppose $\triangle$ is a triangulation with vertices at the points of $\mathcal{A}$, and suppose $\mathcal{S}(\triangle)$ is a spline space defined over $\triangle$ with dimension $n \geq n_d$. Then the set of all splines in $\mathcal{S}(\triangle)$ that interpolate the data is given by

$$\Lambda(f) = \{s \in \mathcal{S}(\triangle) : s(x_i, y_i) = f_i,\ i = 1, \cdots, n_d\}.$$

Given a spline $s \in \Lambda(f)$, we measure its energy using the well-known thin-plate energy functional

$$E(s) = \int_\Omega [(D_{xx}s)^2 + 2(D_{xy}s)^2 + (D_{yy}s)^2]\,dx\,dy. \tag{4.2}$$

Definition 4.1. *The* minimal energy (ME) interpolating spline *is the spline s_e in Λ such that $E(s_e) = \min\{E(s),\ s \in \Lambda(f)\}$.*

It follows from standard Hilbert space approximation results that if $\Lambda(f)$ is not empty, then there exists a unique ME-spline which is characterized by the property

$$\langle s_e, \psi\rangle_E = 0, \qquad \text{all } \psi \in \Lambda(0), \tag{4.3}$$

where

$$\langle \phi, \psi\rangle_E = \int_\Omega [D_{xx}\phi\, D_{xx}\psi + 2D_{xy}\phi\, D_{xy}\psi + D_{yy}\phi\, D_{yy}\psi]\,dx\,dy.$$

To compute the minimal energy spline, we now suppose that there exists a minimal determining set $\mathcal{M}$ for $\mathcal{S}(\triangle)$ such that the corresponding $\mathcal{M}$-basis $\{\psi_i\}_{i=1}^n$ is stable and local. For all standard spline spaces, we may choose $\mathcal{M}$ to include the set $\mathcal{A}$, and in fact, can assume that the first n_d points in $\mathcal{M}$ are (x_i, y_i), $i = 1, \ldots, n_d$. Now suppose s is written in the form

$$s = \sum_{i=1}^n c_i \psi_i. \tag{4.4}$$

Let $\underline{c} := (c_1, \ldots, c_{n_d})^T$ and $\overline{c} := (c_{n_d+1}, \ldots, c_n)^T$.

Theorem 4.2. *The spline s is the ME interpolating spline if and only if $\underline{c} = (f_1, \ldots, f_{n_d})^T$, and*

$$M\overline{c} = r, \tag{4.5}$$

where

$$M_{ij} := \langle \psi_{n_d+i}, \psi_{n_d+j}\rangle_E, \qquad i, j = 1, \ldots, n - n_d,$$

$$r_j := \sum_{i=1}^{n_d} f_i \langle \psi_i, \psi_{n_d+j}\rangle_E, \qquad j = 1, \ldots, n - n_d.$$

Proof: By our assumptions on the $\mathcal{M}$-basis, s belongs to $\Lambda(f)$ if and only if $\underline{c} = (f_1, \ldots, f_{n_d})^T$. Since $\psi_{n+1}, \ldots, \psi_N$ span $\Lambda(0)$, it follows that (4.3) is equivalent to

$$\langle \sum_{i=1}^{n_d} c_i \psi_i, \psi_j\rangle_E = 0, \quad j = n_d + 1, \ldots, n, \tag{4.6}$$

which can immediately be rewritten as (4.5). □

The uniqueness of the ME spline insures that the matrix M in (4.5) is nonsingular. Thus, to compute the coefficient vector of the ME spline, we simply need to compute M and r, and then solve (4.5) for $\bar{c}$. We now discuss the assembly of M and r, which we base on the algorithm of Sect. 3. First, we observe that in this case the inner-product is an integral, and thus clearly satisfies (3.2). To carry out Algorithm 3.2, all we need to do is compute the inner-products $\langle B^T_\alpha, B^T_\beta \rangle_E$. Derivatives of the B^T_α can be expressed in terms of Bernstein basis polynomials of lower degree, see Lemma 2.11 of [13]. But inner-products of Bernstein basis polynomials of any degree can be computed explicitly, see Theorem 2.34 in the book [13]. For tables of the inner-products for degrees $d = 1, 2, 3$, see pages 46–47 of the book.

§5. Discrete Least Squares Splines

In this section we discuss a well-known method for fitting bivariate scattered data in the case when the number n_d of data locations is very large. We assume that the data are the measurements $\{f_i := f(x_i, y_i)\}_{i=1}^{n_d}$ of an unknown function f defined on Ω. To create a spline fit, we work with a spline space $\mathcal{S}(\triangle)$ defined on a triangulation $\triangle$ with a set $\mathcal{V}$ of vertices which are not necessarily at the points of $\mathcal{A}$. Typically we choose the number of vertices to be (much) smaller than n_d.

Definition 5.1. *The* discrete least-squares (DLSQ) spline fit of f *is the spline* $s_l \in \mathcal{S}(\triangle)$ *that minimizes*

$$\|s - f\|^2_{\mathcal{A}} := \sum_{j=1}^{n_d} [s(x_j, y_j) - f_j]^2.$$

It is well known that if $\mathcal{S}(\triangle)$ satisfies the property

$$s(x_i, y_i) = 0, \quad i = 1, \ldots, n_d, \qquad \text{implies } s \equiv 0, \tag{5.1}$$

then there is a unique discrete least-squares spline s_l fitting the data. This requires that the number n_d of data points be at least equal to the dimension n of the spline space. In practice n_d will typically be much larger than n. However, in general this is not sufficient – we need the data points to be reasonably well distributed among the triangles of $\triangle$ in order to make (5.1) hold.

It follows from standard Hilbert space approximation results that there exists a unique DLSQ-spline s_l which is characterized by the property

$$\langle s_l - f, \psi \rangle_{\mathcal{A}} = 0, \qquad \text{all } \psi \in \mathcal{S}(\triangle), \tag{5.2}$$

where

$$\langle \phi, \psi \rangle_{\mathcal{A}} := \sum_{i=1}^{n_d} \phi(x_i, y_i)\psi(x_i, y_i).$$

To compute the least-squares spline s_l, we now write it in the form (4.4), where $\{\psi_i\}_{i=1}^n$ is a stable local $\mathcal{M}$-basis for $\mathcal{S}(\triangle)$. Let $c := (c_1, \ldots, c_n)$ be the vector of coefficients.

Theorem 5.2. *The spline s_l is the DLSQ spline fit of f if and only if*

$$Mc = r, \tag{5.3}$$

where

$$M_{ij} := \langle \psi_i, \psi_j \rangle_{\mathcal{A}}, \qquad i, j = 1, \ldots, n,$$

$$r_j := \langle f, \psi_j \rangle_{\mathcal{A}}, \qquad j = 1, \ldots, n.$$

Proof: Since $\psi_1, \ldots, \psi_n$ span $\mathcal{S}(\triangle)$, it follows that (5.2) is equivalent to

$$\langle \sum_{i=1}^{n} c_i \psi_i - f, \psi_j \rangle_{\mathcal{A}} = 0, \quad j = 1, \ldots, n, \tag{5.4}$$

which can immediately be rewritten as (5.3). $\square$

The assumption (5.1) insures that the matrix M in (5.3) is nonsingular. Thus, to compute the coefficient vector of the least-squares spline, we simply need to compute M and r, and then solve (5.3) for c. Note that here the matrix M is of size $n \times n$ in contrast to the minimal-energy case where it is of size $(n - n_d) \times (n - n_d)$. For spline spaces $\mathcal{S}(\triangle)$ with stable local minimal determining sets $\mathcal{M}$, the computation of M can be done efficiently using Algorithm 3.2. To apply it, we need to compute the inner-products $\langle B_\alpha^T, B_\beta^T \rangle_{\mathcal{A}}$. For this, we simply need the values of the Bernstein basis polynomials at all of the points of $\mathcal{A}$ that lie in T. These values can also be used to compute the vector r in (5.3). We give an efficient algorithm in Sect. 9 for computing the values of all Bernstein-Basis polynomials at a fixed point.

§6. Penalized Least-Squares Splines

In this section we discuss a method for fitting bivariate scattered data which is preferable to least-squares when the data $\{f_i\}_{i=1}^{n_d}$ are noisy. It is a generalization of the least-squares fitting method in Sect. 5. Suppose $\mathcal{A} := \{x_i, y_i\}_{i=1}^{n_d}$, and $\mathcal{S}(\triangle)$ are as in the previous section. For each $s \in \mathcal{S}(\triangle)$, let $E(s)$ be its associated thin-plate energy defined in (4.2).

Definition 6.1. *Given $\lambda \geq 0$, the* penalized least-squares (PLSQ) spline fit of *f is the spline s_λ in $\mathcal{S}(\triangle)$ that minimizes*

$$E_\lambda(s) := \|s - f\|_{\mathcal{A}} + \lambda E(s).$$

It is well known that if the spline space $\mathcal{S}(\triangle)$ satisifies (5.1), then there exists a unique PLSQ-spline s_λ. Moreover, s_λ is characterized by

$$\langle s_\lambda - f, s \rangle_{\mathcal{A}} + \lambda \langle s_\lambda, s \rangle_E = 0, \qquad \text{all } s \in \mathcal{S}(\triangle).$$

To compute the PLSQ spline s_λ, we write it in the form (4.4), where $\{\psi_i\}_{i=1}^n$ is a stable local $\mathcal{M}$-basis for $\mathcal{S}(\triangle)$. Let $c := (c_1, \ldots, c_n)$ be the vector of coefficients. We immediately have the following result.

Theorem 6.2. *The spline s_λ is the PLSQ fit of f if and only if*

$$Mc = r, \tag{6.1}$$

where

$$M_{ij} := \langle \psi_i, \psi_j \rangle_{\mathcal{A}} + \lambda (\psi_i, \psi_j)_E, \qquad i, j = 1, \ldots, n,$$

and r is the vector in (5.3).

The assumption (5.1) insures that the matrix M in (6.1) is nonsingular. We can efficiently compute both M and r by the methods of the previous two sections. Note that when $\lambda = 0$, we get the least-squares spline fit.

§7. Solution of Boundary-Value Problems

In this section we describe how Algorithm 3.2 can be used to compute Galerkin approximations to solutions of elliptic boundary-value problems.

7.1. Second order problems with homogeneous boundary conditions

We first describe the method for the simple model problem

$$\begin{aligned} -\nabla \cdot (\kappa \nabla u) &= f, \qquad \text{on } \Omega, \\ u &= 0, \qquad \text{on } \partial\Omega, \end{aligned} \tag{7.1}$$

where f and κ are given functions on Ω. Suppose that $\triangle$ is a triangulation of Ω, and let $\mathcal{S}(\triangle)$ be a bivariate spline space defined on $\triangle$ with a stable MDS $\mathcal{M}$. Let

$$U_0 := \{ s \in \mathcal{S}(\triangle) : s(x,y) = 0, \text{ all } (x,y) \in \partial\Omega \}.$$

We look for an approximation s_g of u in U_0. Suppose $\psi_1, \ldots, \psi_n$ is a stable local basis for U_0, and suppose s_g is written in the form (4.4). Then by the Galerkin method, see [8], the coefficients of s_g should be chosen to be the solution of the system of equations

$$Mc = r, \tag{7.2}$$

where

$$M_{ij} = \langle \psi_i, \psi_j \rangle_G := \int_\Omega \kappa(x,y) \nabla \psi_i(x,y) \cdot \nabla \psi_j(x,y) \, dx dy,$$

for $i, j = 1, \ldots, n$, and

$$r_i = \langle f, \psi_i \rangle_2 := \int_\Omega f(x,y) \psi_i(x,y) \, dx dy, \qquad i = 1, \ldots, n.$$

Since $\nabla \psi_i \cdot \nabla \psi_j = D_x \psi_i \, D_x \psi_j + D_y \psi_i \, D_y \psi_j$, the computation of M involves computing inner-products of first derivatives of the Bernstein basis polynomials. If $\kappa \equiv 1$, these inner-products can be computed explicitly. For general κ and for the computation of the vector r, we will need to use a quadrature rule. We use Gaussian quadrature, see Remark 7.

We emphasize that for this problem, to use Algorithm 3.2 we need a stable local minimal determing set for the subspace of splines $U(0)$. In particular, all spline coefficients associated with domain points on the boundary of Ω must be set to 0. Thus, for example, if the approximating spline space is $\mathcal{S}_5^{1,2}(\triangle)$, then for each boundary vertex v whose exterior angle is less than π, only one of the domain points in the disk $D_2(v)$ of radius 2 around v can be included in $\mathcal{M}$. Recall, that for the full space, we chose six points in each $D_2(v)$, see Fig. 1. When the exterior angle at v is equal to π, we must choose three points in $D_2(v)$ to include in $\mathcal{M}$.

7.2. Second order problems with inhomogeneous boundary conditions

Only minor modifications are required to solve the inhomogeneous boundary-value problem

$$\begin{aligned} -\nabla \cdot (\kappa \nabla u) &= f, \qquad \text{on } \Omega, \\ u &= g, \qquad \text{on } \partial\Omega, \end{aligned} \tag{7.3}$$

where f and κ are given function on Ω, and g is a given function on $\partial\Omega$. As before, we choose a spline space $\mathcal{S}(\triangle)$ defined on a triangulation $\triangle$ of Ω with a stable local $\mathcal{M}$-basis $\{\psi_i\}_{i=1}^n$. As is well known, see e.g. [8], in this case we look for an approximation to u of the form $s = s_b + s_h$, where s_b is a spline such that $s_b \approx g$ on $\partial\Omega$, and s_h is the Galerkin approximation of the solution of the boundary-value problem with homogeneous boundary conditions and right-hand side $f - s_b$. Writing s_h in the form (4.4), we can compute its coefficients from the equations

$$Mc = \tilde{r},$$

where M is as in (7.2) and

$$\tilde{r}_j = \langle f, \psi_j \rangle_2 - \langle s_b, \psi_j \rangle_G, \qquad j = 1, \ldots, n.$$

It is straightforward to construct s_b. We first choose its Bernstein–Bézier coefficients associated with domain points on the boundary edges of $\triangle$ so that s_b does a good job of approximating g on $\partial\Omega$. For example, we may interpolate g at an appropriate number of points on each edge of $\triangle$. Then we set as many remaining coefficients to zero as possible while observing all smoothness conditions.

7.3. Fourth order problems

As an example of a fourth order problem, consider the biharmonic equation

$$\triangle^2 u = f, \qquad \text{on } \Omega, \tag{7.4}$$

subject to the boundary conditions

$$\begin{aligned} u &= g, \qquad \text{on } \partial\Omega, \\ \frac{\partial u}{\partial n} &= h, \qquad \text{on } \partial\Omega, \end{aligned} \tag{7.5}$$

where ∂n stands for the normal derivative to the boundary. As before, we construct an approximation $s_g := s_b + s_h$ to u by first constructing a spline s_b that approximately satisfies the boundary conditions. We then compute s_h as the Galerkin spline for the corresponding problem with homogeneous boundary conditions. Starting with a spline space $\mathcal{S}(\triangle)$, we define the subspace

$$U_0 := \{s \in \mathcal{S}(\triangle) : s \text{ satisfies the boundary conditions (7.5) with } g \equiv h \equiv 0.\}$$

Assuming $\psi_1, \ldots, \psi_n$ is a stable local basis for U_0, and writing s_h in the form (4.4), the Galerkin method provides the following linear system of equations for the coefficients of s_h:

$$Mc = r,$$

where

$$M_{ij} := \int_\Omega \triangle\psi_i \, \triangle\psi_j \, dx\,dy, \qquad i,j = 1, \ldots, n,$$

and

$$r_j = \langle f, \psi_j \rangle_2 - \int_\Omega \triangle s_g \triangle\psi_j \, dx\,dy, \qquad j = 1, \ldots, n.$$

Note that although we are allowed to use C^0 splines in the Galerkin method for solving second order boundary-value problems, for fourth order problems the well-known conformality conditions require that we use a space of splines that is C^1 at least.

§8. Examples

In this section we give several numerical examples. Given a triangulation $\triangle$, we write $\mathcal{V}$ and $\mathcal{E}$ for the sets of vertices and edges of $\triangle$. We also write n_V, n_E, and n_T for the numbers of vertices, edges, and triangles of $\triangle$. We work with the following spline spaces:

1) $\mathcal{S}^0_d(\triangle)$. The dimension of this space is $n_V + (d-1)n_E + \binom{d-1}{2} n_T$. A stable local MDS is given by the full set $\mathcal{D}_{d,\triangle}$ of domain points. This space approximates smooth functions up to order $\mathcal{O}(|\triangle|^{d+1})$, see Remark 5.

2) $\mathcal{S}^{1,2}_5(\triangle)$. This space has dimension $6n_V + n_E$, and as shown in Theorem 6.1 of [13], a stable local minimal determining set $\mathcal{M}$ is given by the set of domain points

$$\mathcal{M} := \bigcup_{v \in \mathcal{V}} \mathcal{M}_v \cup \bigcup_{e \in \mathcal{E}} \mathcal{M}_e,$$

where for each vertex v of $\triangle$, $\mathcal{M}_v$ is the set of six domain points in $D_2(v) \cup T_v$ for some triangle attached to v. For each edge e of $\triangle$, $\mathcal{M}_e$ consists of the domain point $\xi^{T_e}_{122}$ for some triangle T_e containing the edge e and with first vertex opposite e. Fig. 1 (left) shows the points in $\mathcal{M}$ for a typical triangulation, where points in the sets $\mathcal{M}_v$ are marked with black dots, while those in the

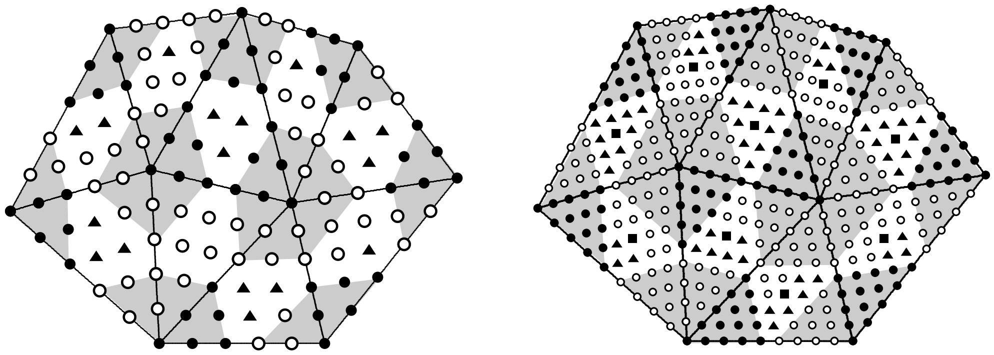


Fig. 1. Minimal determining sets for $\mathcal{S}_5^{1,2}(\triangle)$ and $\mathcal{S}_9^{2,4}(\triangle)$.

sets $\mathcal{M}_e$ are marked with triangles. Since $\mathcal{S}_5^{1,2}(\triangle)$ has a stable local basis, it approximates smooth functions up to order $\mathcal{O}(|\triangle|^6)$, see Remark 5.

3) $\mathcal{S}_9^{2,4}(\triangle)$. This space has dimension $15n_V + 3n_E + n_T$, and as shown in Theorem 7.1 of [13], a stable local minimal determining set $\mathcal{M}$ is given by the set of domain points

$$\mathcal{M} := \bigcup_{v\in\mathcal{V}} \mathcal{M}_v \cup \bigcup_{e\in\mathcal{E}} \mathcal{M}_e \cup \bigcup_{T\in\triangle} \mathcal{M}_T,$$

where for each vertex v, $\mathcal{M}_v$ is the set of fifteen domain points in $D_4(v) \cup T_v$ for some triangle attached to v. For each edge e of $\triangle$, $\mathcal{M}_e$ consists of the three domain points $\xi_{144}^{T_e}, \xi_{243}^{T_e}, \xi_{234}^{T_e}$ for some triangle T_e containing the edge e and with first vertex opposite e. For each triangle T, $\mathcal{M}_T$ consists of the domain point ξ_{333}^T. Fig. 1 (right) shows $\mathcal{M}$ for a typical triangulation. Here squares are used to mark points in the set $\mathcal{M}_T$. Since $\mathcal{S}_9^{2,4}(\triangle)$ has a stable local basis, it approximates smooth functions up to order $\mathcal{O}(|\triangle|^{10})$, see Remark 5.

8.1. Minimal Energy Fit

Example 8.1. *We construct a C^1 surface based on measurements of the height of a nose cone at 803 points in the domain shown in Fig. 2 (left). The figure shows the Delaunay triangulation $\triangle_{nose}$ associated with the data points.*

Discussion: For this application we choose the minimal energy interpolating spline in the space $\mathcal{S}_5^{1,2}(\triangle_{nose})$. This triangulation has $n_E = 2357$ edges and $n_T = 1555$ triangles. Thus, the dimension of the space is 7175, and finding the minimal energy fit requires solving a system of 6372 equations. The associated matrix is sparse with only 243,402 of the 40,602,384 entries being nonzero. The computation took 34 seconds on a desktop, but see Remark 8. The resulting spline fit is shown in Fig. 2 (right). $\square$

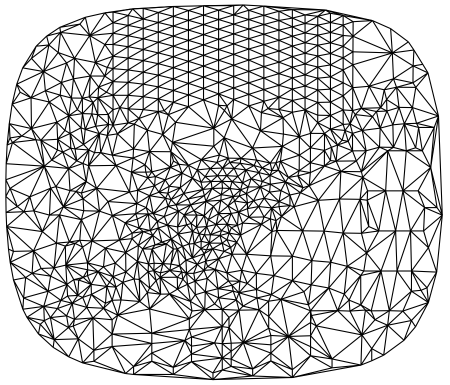
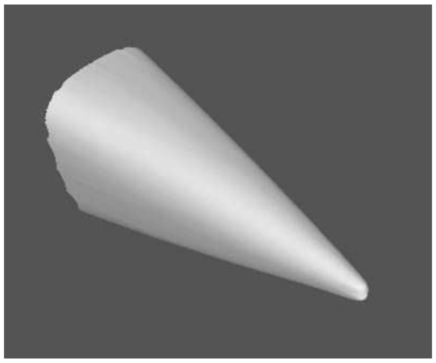



Fig. 2. The minimal energy fit of Example 8.1.

To further illustrate the behavior of minimal energy spline fits, we give a second example involving a known function where we can compute errors. We use the well-known Franke function

$$
\begin{aligned}
F(x,y) = {} & 0.75\exp(-0.25(9x-2)^2 - 0.25(9y-2)^2) \\
& + 0.75\exp(-(9x+1)^2/49 - (9y+1)/10) \\
& + 0.5\exp(-0.25(9x-7)^2 - 0.25(9y-3)^2) \\
& - 0.2\exp(-(9x-4)^2 - (9y-7)^2)
\end{aligned}
\tag{8.1}
$$

defined on the unit square. The surface corresponding to F is shown in Fig. 3.

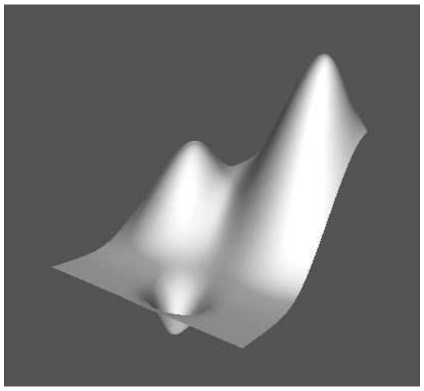
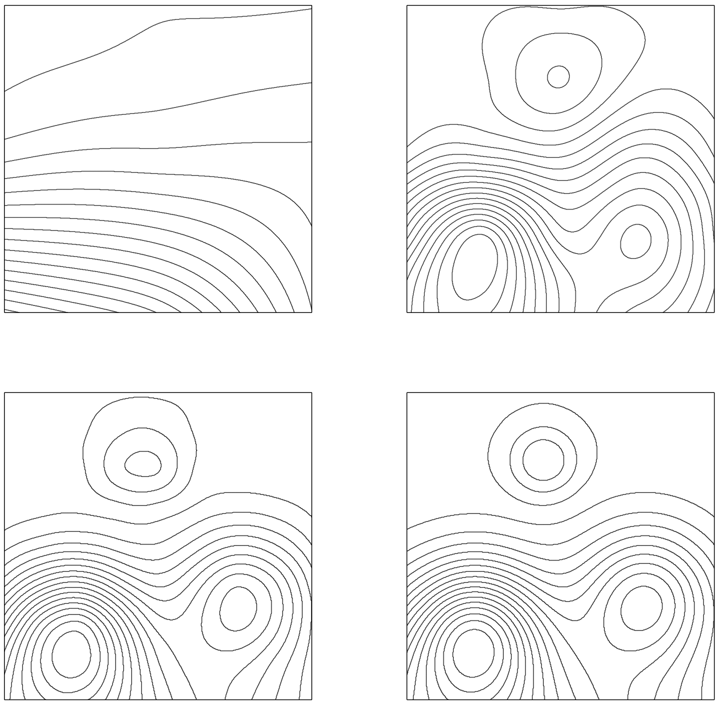


Fig. 3. The Franke function.

Example 8.2. *We construct minimal energy spline fits of F from the spaces $\mathcal{S}_5^{1,2}(\triangle_k)$ based on type-I triangulations $\triangle_k$ with $k = 9$, 25, 81, and 289 vertices.*

Discussion: Type-I triangulations are obtained by first forming a rectangular grid, and then drawing in the northeast diagonals. We show the results of our computations in the table in Fig. 4, where the columns labelled *nt* and *nsys* give the number



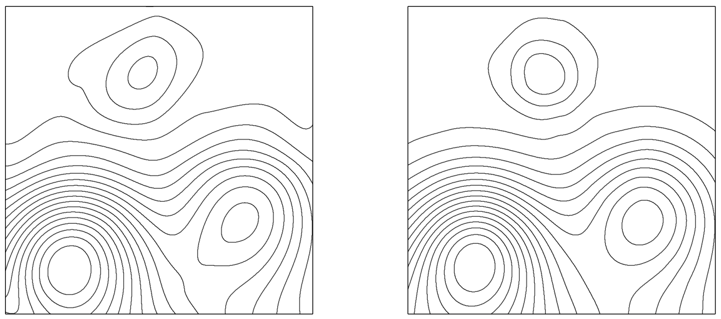
	k	nt	$nsys$	e_∞	e_2	$time$
$\mathcal{S}_5^{1,2}(\triangle_k)$	9	8	61	1.5(−1)	1.9(−1)	.02
	25	32	181	8.8(−2)	3.0(−2)	.09
	81	128	613	5.0(−2)	6.2(−3)	.5
	289	512	2245	3.9(−3)	4.0(−4)	3.7

Fig. 4. Minimal energy fits of F from $\mathcal{S}_5^{1,2}(\triangle_k)$, see Example 8.2.

of triangles in $\triangle_k$ and the size of the linear system being solved, respectively. The columns labelled e_∞ and e_2 give the maximum error and the RMS errors measured over a grid of 640,000 points. The last column contains the computational times in seconds. To illustrate the quality of the fits, we give contour plots for each of the four cases. Clearly, the fit on $\triangle_9$ is not very good, and completely misses the peaks and valley in the function. On the other hand, the contour plot for $\triangle_{289}$ is almost identical to the contour plot of f itself. In comparing Figs. 3 and 4, note that the 3D view of F in Fig. 3 is from behind the surface so that the peaks do not obscure the valley. The results here are very comparable to those obtained for the space of C^2 quintic splines on Powell-Sabin splits, see Table 3 of [10], which were computed by the Lagrange multiplier method of [4]. $\square$

8.2. Least-squares Fitting

In this section we give examples of least-squares fitting with splines. Let F be the



$\mathcal{S}_5^{1,2}(\triangle_k)$

k	nd	$nsys$	e_∞	e_2	$time$
9	289	70	4.5(−2)	9.5(−3)	.1
25	289	206	1.6(−2)	1.9(−3)	.12
25	1089	206	1.1(−2)	1.6(−3)	.43
81	1089	694	5.3(−4)	5.1(−5)	.44
81	4225	694	5.0(−4)	4.8(−5)	1.8

$\mathcal{S}_9^{2,4}(\triangle_k)$

k	nd	$nsys$	e_∞	e_2	$time$
9	289	191	1.8(−2)	1.9(−3)	2.5
9	1089	191	1.1(−2)	1.4(−3)	11
25	1089	575	5.2(−4)	4.4(−5)	12
25	4225	575	3.5(−4)	4.1(−5)	51
81	4225	1967	1.3(−6)	1.1(−7)	57

Fig. 5. Least-squares fits of F from $\mathcal{S}_5^{1,2}(\triangle_k)$ and $\mathcal{S}_9^{2,4}(\triangle_k)$, see Example 8.3.

Franke function (8.1) defined on the unit square.

Example 8.3. *We construct least-squares fits of F using the spaces $\mathcal{S}_5^{1,2}(\triangle_k)$ and $\mathcal{S}_9^{2,4}(\triangle_k)$ defined on type-I triangulations based on measurements on rectangular grids of $nd := 289, 1089$ and 4225 points.*

Discussion: The table in Fig. 5 shows the results of our computations, where the columns labelled e_∞ and e_2 give the maximum error and RMS errors on a grid of 640,000 points. The column labelled $nsys$ gives the size of the linear system being solved. The last column contains the computational times in seconds. We show contour plots of the fits based on 289 data points, where the plot on the left is the fit from $\mathcal{S}_5^{1,2}(\triangle_9)$ while the one on the right is the fit from $\mathcal{S}_9^{2,4}(\triangle_9)$. $\square$

8.3. Penalized least-squares

Let F be the Franke function (8.1) defined on the unit square. We approximate F based on measurements on a grid of 1089 points. However, here we add random

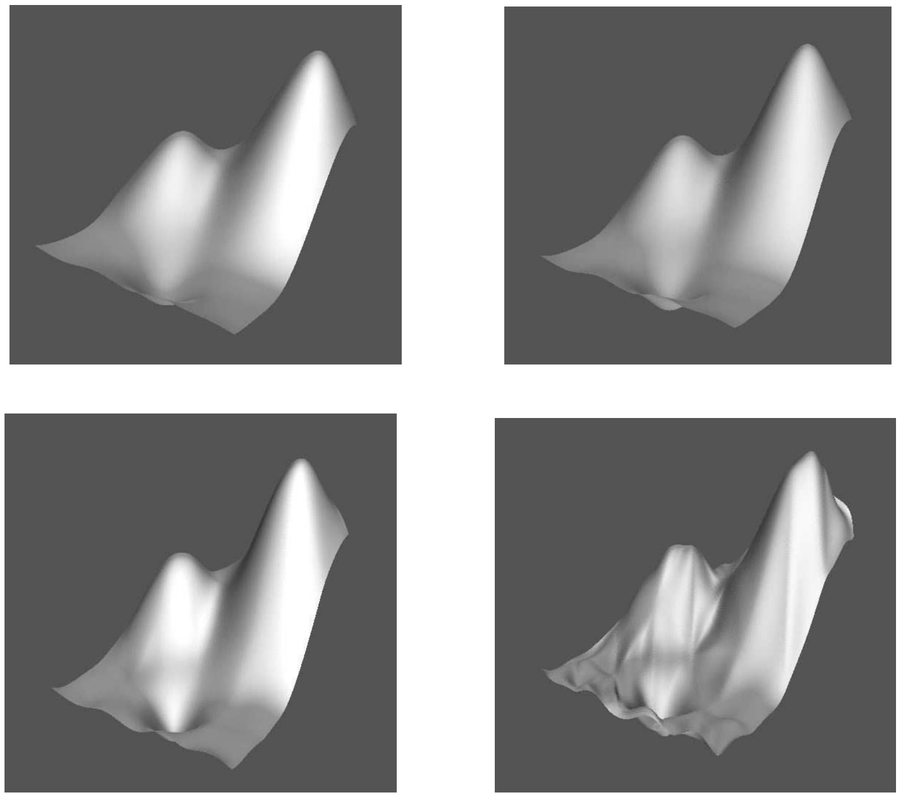


Fig. 6. Penalized least-squares fits of F with $\lambda = .01, .005, .001, 0$.

errors ε_i to the measurements f_i, where the ε_i are uniformly distributed in $[-.1, .1]$. This is a rather significant amount of noise since the values of F lie in the interval $[-.2, 1.1]$.

Example 8.4. *We compute the penalized least-squares spline fits* $s_\lambda \in \mathcal{S}_5^{1,2}(\triangle_{25})$ *for* $\lambda = .01, .005, .001$ *and* 0.

Discussion: The maximum errors for these choices of λ were .127, .111, .08, and .10. The corresponding surfaces are shown in Fig. 6. Clearly, the value of λ makes a big difference. If it is too small, the fit follows the noise, and is not very smooth. If it is too large, some of the shape is lost. $\square$

8.4. Solution of boundary-value problems

Example 8.5. *Compute a solution to the boundary-value problem (7.3) with*

$$f(x,y) := 4\cos(x^2+y^2) - 4(x^2+y^2)\sin(x^2+y^2) \\ + 10\cos(25(x^2+y^2)) - 250\sin(25(x^2+y^2)),$$

$$g(x,y) := \sin(x^2+y^2) + .1\sin(25(x^2+y^2)),$$

and $\kappa \equiv 1$ on the unit square Ω.

Discussion: In this case, the solution of the boundary-value problem (7.3) is

$$u(x,y) := \sin(x^2+y^2) + .1\sin(25(x^2+y^2)).$$

We computed spline approximations of u by the Galerkin method using spline spaces defined on type-I triangulations $\triangle_k$ with $k = 81, 289, 1089$ and 4225 vertices. The results are shown in the tables in Fig. 7, where k and nt are the number of vertices and triangles, and dim is the dimension of the associated space. We give results not only for $\mathcal{S}_5^{1,2}(\triangle_k)$, but also for $\mathcal{S}_1^0(\triangle_k)$ for comparison purposes. As above, e_∞ and e_2 are the maximum and RMS errors on a grid of 25,600 points, and the last column shows the computational time in seconds. The figure on the left shows the traditional C^0 linear fit with $k = 1089$, while the one on the right shows the fit with $\mathcal{S}_5^{1,2}(\triangle_k)$ with $k = 289$. In addition to having much smaller errors, it is much smoother. The results here can be compared with those in Table 13 of [4] which were computed with C^1 quintic splines, but with f and g multiplied by 10. $\square$

Finally, we give an example involving a fourth-order differential equation.

Example 8.6. *Compute a solution to the fourth-order boundary-value problem (7.4)–(7.5) with*

$$f(x,y) := 4\exp(x+y), \qquad g(x,y) = h(x,y) := \exp(x+y),$$

on the unit square Ω.

Discussion: The solution of this boundary-value problem is $u(x,y) := \exp(x+y)$. We computed spline approximations of u by the Galerkin method using the space $\mathcal{S}_5^{1,2}(\triangle_k)$ defined on type-I triangulations with $n = 9$, 25, and 81 vertices. The errors are shown in Tab. 1 along with the computational times. For a comparison with an approximation computed using $\mathcal{S}_5^1(\triangle_{25})$ using the Lagrange multiplier method, see Example 26 in [4].

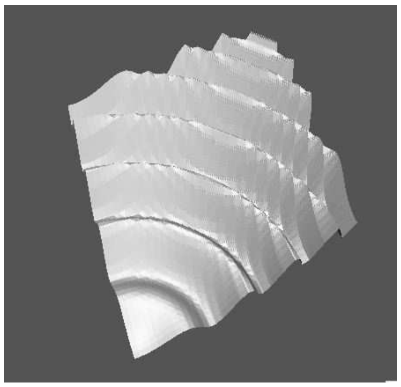
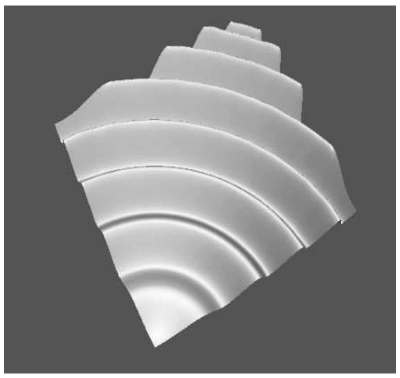
§9. Computing Bernstein Basis Polynomials and Their Derivatives

In this section we describe algorithms for efficiently evaluating Bernstein basis polynomials of degree d and their derivatives. Fix a triangle $T := \langle v_1, v_2, v_3\rangle$ with vertices $v_i = (x_i, y_i)$ for $i = 1, 2, 3$. The associated Bernstein basis polynomials are defined by

$$B^d_{ijk}(x,y) := \frac{d!}{i!\,j!\,k!} b_1^i b_2^j b_3^k, \qquad i+j+k = d, \tag{9.1}$$

where b_1, b_2, b_3 are the barycentric coordinates of (x,y) relative to the triangle T. Specifically,

$$b_1(x,y) := \frac{1}{det}\begin{vmatrix} 1 & 1 & 1 \\ x & x_2 & x_3 \\ y & y_2 & y_3 \end{vmatrix}, \qquad det := \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix}, \tag{9.2}$$

$\mathcal{S}_5^{1,2}(\triangle_k)$

k	nt	dim	e_∞	e_2	time
81	128	590	2.1(−2)	2.2(−3)	0.2
289	512	2334	1.4(−3)	8.1(−5)	.9
1089	2048	9278	2.5(−5)	1.0(−6)	4.3
4225	8192	36990	3.8(−7)	1.5(−8)	24

$\mathcal{S}_1^{0}(\triangle_k)$

k	nt	dim	e_∞	e_2	time
81	128	169	1.8(−1)	6.5(−2)	.006
289	512	225	9.6(−2)	2.6(−3)	.02
1089	2048	961	9.6(−2)	2.6(−3)	.13
4225	8192	3969	7.0(−3)	1.8(−3)	.90
16641	32768	16129	1.9(−3)	3.6(−4)	7.2

Fig. 7. Spline approximations of the BVP of Example 8.5.

$\mathcal{S}_5^{1,2}(\triangle_k)$

k	nt	dim	e_∞	e_2	time
9	8	18	3.4(−5)	1.1(−5)	.01
25	32	106	3.7(−8)	8.9(−9)	.07
81	128	498	6.1(−10)	1.3(−10)	.46

Tab. 1. Spline approximations of the BVP of Example 8.6.

with similar definitions for b_2 and b_3. This shows that b_1, b_2, b_3 are linear functions of x and y.

The following algorithm simultaneously computes the values of all of the Bernstein basis polynomials $\{B^d_{ijk}\}_{i+j+k=d}$ at a fixed point (x, y). Suppose the barycentric coordinates of (x, y) relative to T are b_1, b_2, b_3.

Algorithm 9.1. *Computation of all Bernstein basis polynomials at a point* (x, y).

- *Set* $B(0, 0) = 1$

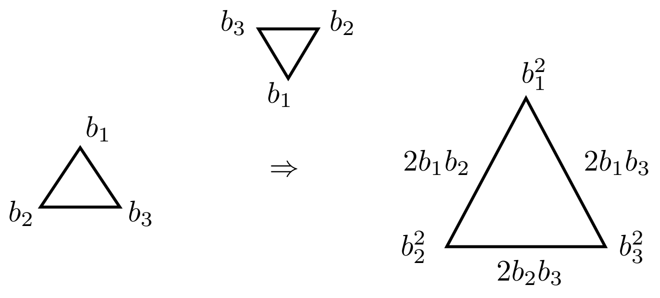


Fig. 8. One step of Algorithm 9.1.

```
• For $k = 1$ to $d$
  For $i = k$ to $0$
  For $j = i$ to $0$
      $B(i,j) = b_1 B(i,j) + b_2 B(i-1,j) + b_3 B(i-1,j-1)$
```

A simple inductive proof shows that the following lemma holds.

Lemma 9.2. *Algorithm 9.1 produces the values*

$$\begin{pmatrix} B^d_{d,0,0}(x,y) & & & \\ B^d_{d-1,1,0}(x,y) & B^d_{d-1,0,1}(x,y) & & \\ \cdots & \cdots & & \\ B^d_{0,d,0}(x,y) & B^d_{0,1,d-1}(x,y) & \cdots & B^d_{0,0,d}(x,y) \end{pmatrix}.$$

We illustrate one step of this algorithm ($k = 2$) in Fig. 8. It shows how the six Bernstein basis polynomials of degree 2 are computed from the three of degree 1. The idea is to use the inverted triangle with b_1, b_2, b_3 at its vertices as a mask which is applied at six different positions to the triangular array of 1st degree Bernstein basis polynomials on the left to get the triangular array of Bernstein basis polynomials of degree 2 on the right. Applying the mask again would lead to an array containing the 10 values of the cubic Bernstein basis polynomials, etc.

For the applications in Sect. 7, we also need an algorithm for evaluating derivatives of the Bernstein basis polynomials at a given point (x,y). We now show that Algorithm 9.1 can be easily adapted to produce the values of arbitrary directional derivatives of the Bernstein basis polynomials. Recall (see e.g. [13]) that if $u = u_1 - u_0$ is a vector in $\mathbb{R}^2$, then its directional coordinates (a_1, a_2, a_3) relative to T are just the differences of the barycentric coordinates of u_1 and u_0 relative to T. This implies $a_1 + a_2 + a_3 = 0$. The following lemma shows how Algorithm 9.1 can be used to compute the values of the set of functions $\{D^m_u B^d_{ijk}(x,y)\}_{i+j+k=d}$ at a fixed point (x,y) whose barycentric coordinates relative to T are b_1, b_2, b_3.

Lemma 9.3. *Fix $0 \le m \le d$. Suppose that in carrying out Algorithm 9.1, in the first m times through the k-loop we use a_1, a_2, a_3 in place of b_1, b_2, b_3. Then the* *algorithm produces* $\frac{(d-\nu)\,!}{d\,!}$ *times the matrix*

$$\begin{pmatrix} D_u^m B_{d,0,0}^d(x,y) & & & \\ D_u^m B_{d-1,1,0}^d(x,y) & D_u^m B_{d-1,0,1}^d(x,y) & & \\ \cdots & \cdots & & \\ D_u^m B_{0,d,0}^d(x,y) & D_u^m B_{0,1,d-1}^d(x,y) & \cdots & D_u^m B_{0,0,d}^d(x,y) \end{pmatrix}.$$

Proof: It is well known that

$$D_u B_{ijk}^d = d\big[a_1 B_{i-1,j,k}^{d-1} + a_2 B_{i,j-1,k}^{d-1} + a_3 B_{i,j,k-1}^{d-1}\big],$$

see e.g. Lemma 2.11 of [13], The result then follows from a minor variant of the inductive proof of Lemma 9.2. $\square$

This result can be generalized to compute arbitrary mixed directional derivatives. In particular, suppose we are given m directional vectors $u_1, \ldots, u_m$ whose direction coordinates relative to T are $a^{(\nu)} := (a_1^{(\nu)}, a_2^{(\nu)}, a_3^{(\nu)})$, for $\nu = 1, \ldots, m$. Then we can compute $(d-m)\,!\,/d\,!$ times the matrix of values $[D_{u_1} \ldots D_{u_m} B_{ijk}^d(x,y)]$ by running Algorithm 9.1, but using $a^{(\nu)}$ in place of (b_1, b_2, b_3) in the ν-th pass through the k-loop for $\nu = 1, \ldots, m$.

Since we are interested in computing the derivatives in the directions of the Cartesian axes, we now give formulae for the direction coordinates of the special directional derivatives D_x and D_y.

Lemma 9.4. *Suppose* $T := \langle v_1, v_2, v_3 \rangle$ *where* $v_i := (x_i, y_i)$ *for* $i = 1, 2, 3$. *Then the direction coordinates of* D_x *are*

$$a_1^x = (y_2 - y_3)/det, \quad a_2^x = (y_3 - y_1)/det, \quad a_3^x = (y_1 - y_2)/det, \tag{9.3}$$

while those of D_y *are*

$$a_1^y = (x_3 - x_2)/det, \quad a_2^y = (x_1 - x_3)/det, \quad a_3^y = (x_2 - x_1)/det, \tag{9.4}$$

where det is the determinant in (9.2).

These quantities are just the derivatives of the barycentric coordinate functions b_1, b_2, b_3 with respect to x and y.

§10. Remarks

Remark 1. My first work on programs to compute minimal energy splines was done with Ewald Quak in the mid 1980's during his stay at Texas A&M. Our programs used C^1 cubic and quartic splines, although even to this day the dimension of $\mathcal{S}_3^1(\triangle)$ has not been determined for arbitrary triangulations $\triangle$. The Bernstein–Bézier representation was used, and smoothness conditions were simply incorporated with Lagrange multipliers. The codes were not made public, but the details of how to compute energies were published in [15]. This approach was later used in [7] in connection with the construction of minimal energy surfaces with C^1 cubic parametric splines, where in some cases certain C^2 smoothness conditions were also incorporated via Lagrange multipliers. The Lagrange multiplier approach was further developed in [4], where the Bernstein–Bézier representation is based on the space of piecewise polynomials rather than on $\mathcal{S}_d^0(\triangle)$ as in our earlier papers. A useful iterative method for solving the resulting systems of equations was also developed in [4], see also the survey [11].

Remark 2. There is an extensive theory of splines defined on spherical triangulations which is remarkably similar to the theory of bivariate splines, see [1–3] and Chapters 13–14 of [13]. Such splines are piecewise spherical harmonics. The techniques described here also work for solving variational problems involving spherical spline spaces with stable local minimal determining sets. A number of such spaces are described in [13]. For some computational experiments with minimal energy spherical splines based on Lagrange multiplier methods, see [2] and [5].

Remark 3. There has been considerable recent work on trivariate splines defined on tetrahedral partitions, see Chapters 15–18 of [13]. The methods described here can also be carried over to solve variational problems associated with trivariate splines with stable local minimal determining sets. A number of such spaces are described in [13].

Remark 4. The global methods described here are not the only way to fit scattered data using splines. There are a host of local methods that work with various macro-element spaces. For numerical experiments with some of these methods in the spherical case, see [2].

Remark 5. It is well known, see Chap. 10 of [13], that if a spline space $\mathcal{S}(\triangle)$ has a stable local basis, then it approximates sufficiently smooth functions to order $\mathcal{O}(|\triangle|^{d+1})$, where $|\triangle|$ is the mesh size of $\triangle$, i.e., the diameter of the largest triangle in $\triangle$. However, the minimal energy interpolation method does not attain this optimal approximation order for $d \geq 2$ since it only reproduces linear functions. Indeed, the results of [10] show that the order of approximation using minimal energy splines is only $\mathcal{O}(|\triangle|^2)$. This can be seen in the table in Fig. 4, which is based on a nested sequence of type-I triangulations whose mesh sizes decrease by a factor of .5 at each step.

Remark 6. Our method can also be applied to compute minimal energy fits using the spaces $\mathcal{S}_d^0(\triangle)$. However, it doesn't really make sense to minimize energy for splines in C^0 spline spaces, since minimizing energy leads to splines which are as close to piecewise linear as possible. Even though $\mathcal{S}_d^0(\triangle)$ is a much larger space than $\mathcal{S}_5^{1,2}(\triangle)$, the minimal energy fit from $\mathcal{S}_d^0(\triangle)$ is much worse that the one from $\mathcal{S}_5^{1,2}(\triangle)$.

Remark 7. A Gaussian quadrature rule for approximating integrals of functions over a triangle T is defined by a set of positive numbers $\{w_k, r_k, s_k, t_k\}_{k=1}^m$ where $r_k + s_k + t_k = 1$ for all k. Then for any function g defined on T, its integral is approximated by

$$\int_T g(x,y)\,dxdy \approx \sum_{k=1}^m w_k g(x_k^T, y_k^T), \tag{10.1}$$

where

$$x_k^T = r_k x_1 + s_k x_2 + t_k x_3, \quad y_k^T = r_k y_1 + s_k y_2 + t_k y_3.$$

For each m, there is a d_m such that the corresponding Gaussian quadrature rule integrates all polynomials up to degree d_m exactly. Formulae for various values of m can be found in the literature. In our experiments we use rules from [6] with $m = 25$ and $m = 79$ which integrate bivariate polynomials up to degree 10 and 20 exactly.

Remark 8. The numerical experiments presented here were performed on a Macintosh G5 computer using Fortran. The programs were not optimized for performance, and the times reported are only meant to give a general impression of the speed of computation and to provide a basis for comparing different methods with various parameters.

Remark 9. The programs used for the experiments presented here work with spline spaces up to smoothness C^2. However, it is straightforward to write similar programs for spline spaces with higher smoothness. For example, we could work with any of the macro-element spaces discussed in [13] which are defined for arbitrary smoothness r.

Remark 10. While it may appear that the practical use of global spline fitting methods is limited by the need to solve linear systems of equations which can become very large, in [14] we recently described a method for decomposing large variational spline problems into smaller more managable pieces.

References

1. Alfeld, P., M. Neamtu, and L. L. Schumaker, Bernstein–Bézier polynomials on spheres and sphere–like surfaces, Comput. Aided Geom. Design **13** (1996), 333–349.

2. Alfeld, P., M. Neamtu, and L. L. Schumaker, Fitting scattered data on sphere-like surfaces using spherical splines, J. Comput. Appl. Math. **73** (1996), 5–43.
3. Alfeld, P., M. Neamtu, and L. L. Schumaker, Dimension and local bases of homogeneous spline spaces, SIAM J. Math. Anal. **27** (1996), 1482–1501.
4. Awanou, G., M. J. Lai, and P. Wenston, The multivariate spline method for scattered data fitting and numerical solution of partial differential equations, in *Wavelets and Splines: Athens 2005,* G. Chen and M.-J. Lai (eds), Nashboro Press, Brentwood, 2006, 24–74.
5. Baramidze, V., M. J. Lai, and C. K. Shum, Spherical splines for data interpolation and fitting, SIAM J. Scient. Computing **28** (2006), 241–259.
6. Dunavant, D., High degree efficient symmetrical gaussian quadrature rules for the triangle, International Journal for Numerical Methods in Engineering **21** (1985), 1129-1148.
7. Fasshauer, G. and L. L. Schumaker, Minimal energy surfaces using parametric splines, Comput. Aided Geom. Design **13** (1996), 45–79.
8. Gockenbach, M. S., *Understanding and Implementing the Finite Element Method,* SIAM, Philadelphia, 2006.
9. Golitschek, M. and L. L. Schumaker, Bounds on projections onto bivariate polynomial spline spaces with stable bases, Constr. Approx. **18** (2002), 241–254.
10. Golitschek, M. von, M.-J. Lai, and L. L. Schumaker, Error bounds for minimal energy bivariate polynomial splines, Numer. Math. **93** (2002), 315–331.
11. Lai, M.-J., Multivariate splines for data fitting and approximation, in *Approximation Theory XII: San Antonio 2007*, M. Neamtu and L. L. Schumaker (eds.), Nashboro Press, Brentwood, 2008, to appear.
12. Lai, M. J. and L. L. Schumaker, On the approximation power of bivariate splines, Advances in Comp. Math. **9** (1998), 251–279.
13. Lai, M. J. and L. L. Schumaker, *Spline Functions on Triangulations,* Cambridge University Press, Cambridge, 2007.
14. Lai, M. J. and L. L. Schumaker, A domain decomposition method for computing bivariate spline fits, submitted, 2007.
15. Quak, E. and L. L. Schumaker, Calculation of the energy of a piecewise polynomial surface, in *Algorithms for Approximation II,* J. C. Mason and M. G. Cox (eds), Chapman & Hall, London, 1990, 134–143.