Journal of Computational Mathematics
Vol.29, No.4, 2011, 396–414.

http://www.global-sci.org/jcm
doi:10.4208/jcm.1101-m3203

FITTING C^1 SURFACES TO SCATTERED DATA WITH $S_2^1(\Delta_{m,n}^{(2)})^*$

Kai Qu
School of Mathematical Sciences, Dalian University of Technology, Dalian 116024, China
Department of Mathematics, Dalian Maritime University, Dalian 116026, China
Email: qukai8@dlmu.edu.cn
Renhong Wang and Chungang Zhu
School of Mathematical Sciences, Dalian University of Technology, Dalian 116024, China
Email: renhong@dlut.edu.cn, cgzhu@dlut.edu.cn

Abstract

This paper presents a fast algorithm (BS2 Algorithm) for fitting C^1 surfaces to scattered data points. By using energy minimization, the bivariate spline space $S_2^1(\Delta_{m,n}^{(2)})$ is introduced to construct a C^1-continuous piecewise quadratic surface through a set of irregularly $3D$ points. Moreover, a multilevel method is also presented. Some experimental results show that the accuracy is satisfactory. Furthermore, the BS2 Algorithm is more suitable for fitting surfaces if the given data points have some measurement errors.

Mathematics subject classification: 41A15, 65D07, 65D10.
Key words: Bivariate spline, Scattered data, Surface fitting, Energy minimization, Type-2 triangulation, C^1-continuous.

1. Introduction

Fitting surface to scattered data is a fast growing research area. It deals with the problem of reconstructing an unknown function from given scattered data. The main aim of this paper is to solve the following problem:

> (Θ): Let D be a domain in the (x, y)-plane, and suppose F is a real-valued function defined on D. Suppose we are given the values $z_j = F(x_j, y_j)$ of F at some set of points (x_j, y_j) located in $D, j = 1, 2, \cdots, N$. Find a function f defined on D which reasonably approximates F.

This problem is, of course, precisely the problem of fitting a surface to given data. Naturally, it has many applications, such as terrain modeling, surface reconstruction, fluid-structure interaction, numerical solutions of partial differential equations, kernel learning, and parameter estimation, to name a few. Moreover, these applications come from such different fields as applied mathematics, computer science, geology, biology, engineering, and even business studies([28]).

C^1 surface is the simplest and most practical surfaces in scientific computation and engineering, and it has very important applications. There are lots of efficient methods for fitting surfaces such as Shepard's method, tensor product splines, multiquadratics(MQ), and finite element methods. Shepard defined a C^0-continuous interpolation function as the weighted average

* Received September 7, 2009 / Revised version received November 1, 2010 / Accepted December 22, 2010 / Published online June 27, 2011 /

of the data([19]). Using tensor product B-splines, Lee *et al.* constructed a C^2-continuous interpolation function ([15]). Hardy's multiquadratics are among the most successful and applied methods, it constructs a C^∞-continuous interpolation function ([10]). However, the solvability of multiquadratics interpolation depends on the selection of parameters. Franke and Nielson introduced the modified quadratic Shepard's method to produce C^1-continuous interpolation ([7]), but it is sensitive to triangulation and data distribution, just like finite element methods ([1]). Compactly supported functions were presented in [28, 29], which could produce an interpolation function with arbitrary smoothness. However, it needs to solve some equations. In this paper, using energy minimization, we proposed a very fast algorithm (BS2 Algorithm) for reconstructing a C^1-continuous interpolation function from arbitrary scattered data.

Comparing with all of methods mentioned above, the BS2 Algorithm presented in this paper has some advantages, such as:

- the BS2 Algorithm doesn't need to solve any equations.
- the BS2 Algorithm produces a C^1-continuous function which degree is just 2.
- according to the different requirements, the BS2 Algorithm could construct a function f which interpolates the data exactly, or approximately fits the given data.
- the BS2 Algorithm is very simply because the basis which we used here is centrosymmetric (see Fig. 3.2).
- the surfaces produced by the BS2 Algorithm have minimum energy since the minimum energy constraint is used.

This paper is organized as follows. Section 2 reviews previous work. In Section 3, the spline space $S_2^1(\Delta_{m,n}^{(2)})$ is introduced. Surfaces' energy minimization is introduced in Section 4. The basic idea and the algorithm are given in Section 5. That serves to motivate the discussion of multilevel approximation, given in Section 6. In Section 7, the numerical results and comparison with some algorithms are illustrated. Conclusions and future works are presented in Section 8.

2. Previous Work

There are basically two approaches to handling (Θ). First, we may try to construct a function f which interpolates the data exactly, i.e., such that

$$f(x_j, y_j) = z_j, \qquad j = 1, 2, \cdots, N.$$

This approach may be desirable when the function values at the data points are known to high precision and where it is highly desirable that these values be preserved by the approximating function. The problem of interpolation of scattered data has been addressed by numerous authors ([1, 2, 8, 11, 24]). One of the earliest algorithm in this field was based on inverse distance weighting of data, namely Shepard's method ([7, 19]). Another popular approach to scattered interpolation is to define the interpolation function as a linear combination of radially symmetric basis functions (RBF). Popular choices for the basis functions include Gaussian, multiquadratics (MQ) ([10]), compactly supported functions(CSF) ([27, 29]). Another class of solutions to scattered data interpolation is due to finite element methods ([14]). Lee proposed an algorithm (BA Algorithm) for scattered data interpolation with B-splines ([15]). A recent view of methods for scattered data interpolation is given by [28].

The second approach to handling (Θ) is to construct f which only approximately fits the data. This may be regraded as data fitting (data smooth) and will be desirable when (as is often the case) the data are subject to inaccurate measurement or even errors. The approach of data fitting significantly differs from that of interpolation: While interpolation methods approximate the underlying function by finding a surface that passes through the known data points, data fitting methods find an approximating surface that best fits the known data points according to some specified criteria. This means that the approximating function will pass close to the known values, but not necessarily exactly though them. Data fitting methods have been largely discussed in literatures ([3, 16]).

Timoshenko pointed that if a system is a position of stable equilibrium, its total energy is a minimum ([21]). For the geometric shape design industry, a better understanding of the performance of the strain energy approximation method is important since some of the design processes use the energy as a means to optimize the shape of a curve or surface ([9, 12, 17, 18, 20]). One example is the process of constructing a smooth surface to interpolate a network of curves ([13]), where an energy function is minimized to find the optimal twist vectors for the interpolating surface. Another example is the curve interproximation process ([6]), where a curve with the smoothest shape is sought to interpolate given data. An interproximation scheme for B-spline surface is presented in [26], it shows that the energy form has much bigger impact on the generated curve than the parametrization technique.

3. Spline Space $S_2^1(\Delta_{m,n}^{(2)})$

Splines are piecewise polynomials with certain smoothness. The first author of this paper established the basic theory on multivariate splines over arbitrary partition, and presented the so-called conformality method of smoothing cofactor (the CSC method) which is suitable for studying the multivariate spline over arbitrary partition ([22, 24, 25]).

Let Ω be a domain in R^2, P_k the collection of all these bivariate polynomials with real coefficients and total degree no more than k, i.e.,

$$P_k := \left\{ p = \sum_{i=0}^{k}\sum_{j=0}^{k-i} c_{ij}x^i y^j \middle| c_{ij} \in R \right\}.$$

Using a finite number of irreducible algebraic curves to carry out the partition Δ of the domain Ω, then the domain Ω is divided into M sub-domains $\delta_1, \cdots, \delta_M$, each of such sub-domains is called a cell of Δ. These line segments that form the boundary of each cell are called the edges, intersection points of the edges are called the vertices. If two vertices are two end points of a single edge, then these two vertices are called the adjacent vertices. The vertices which are not lying on the boundary of domain Ω are called interior vertices, and the others are called boundary vertices. The space of bivariate splines with degree k and smoothness μ over Δ is defined by

$$S_k^\mu(\Delta) := \left\{ s \in C^\mu(\Omega) \;\middle|\; s\big|_{\delta_i} \in P_k, i = 1, \cdots, M \right\}.$$

As well known, many regions including the so-called L-form regions and their combinations, can be translated to many rectangular regions. Type-2 triangulations are yielded by connecting two diagonals at each small rectangular cell which are based on rectangular regions. Clearly, if the original rectangular partition is uniform, then the induced type-2 triangulations are called uniform type-2 triangulations. All of type-2 triangulations mentioned in our paper are uniform, see Fig. 3.1.

Without loss the generality, let Ω be a unit square region as follows:

$$\Omega = [0,1] \otimes [0,1].$$

The type-2 triangulation $\Delta_{m,n}^{(2)}$ is yielded by the following partition lines:

$$\begin{aligned} mx - i &= 0, & ny - i &= 0, \\ ny - mx - i &= 0, & ny + mx - i &= 0, \end{aligned}$$

where $i = \cdots, -1, 0, 1, \cdots$. The dimension of $S_k^\mu(\Delta_{m,n}^{(2)})$ is presented as follows.

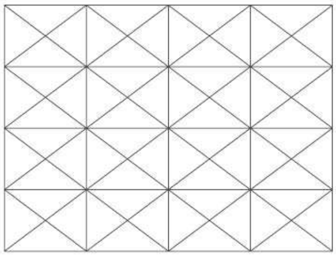


Fig. 3.1. Uniform type-2 triangulation, $m = 4, n = 4$.

Theorem 3.1. ([23, 24])

$$\begin{aligned} \dim S_k^\mu(\Delta_{m,n}^{(2)}) = \binom{k+2}{2} + (3m+3n-4)\binom{k-\mu+1}{2} \\ + mn\binom{ck-2\mu}{2} + (m-1)(n-1)\cdot d_k^\mu(4), \end{aligned} \tag{3.1a}$$

where

$$d_k^\mu(4) = \frac{1}{2}\left(k - \mu - \left[\frac{\mu+1}{3}\right]\right)_+ \cdot \left(3k - 5\mu + 3\left[\frac{\mu+1}{3}\right] + 1\right). \tag{3.1b}$$

Lemma 3.1. ([23, 24]) *For a given partition Δ, let $B(x,y) \in S_k^\mu(\Delta)(0 \le \mu \le k-1)$ be a B-spline that its support is a convex polygon G. If A_i is a given vertex of G, and the number of edges inside of G (including the boundary of G) that passing through A_i is N_i, then*

$$N_i > (k+1)/(k-\mu). \tag{3.2}$$

Lemma 3.1 is a fundmental result. It points out that in order to obtain a B-spline with local support, the lower bound of the number of edges at each vertex on its support should be $(k+1)/(k-\mu)$.

According to Eq. (3.2), in order to construct the locally supported splines over partition $\Delta_{m,n}^{(2)}$ ($N_i = 4$), the degree k of their piecewise polynomials and the smoothness μ must satisfy the following inequality: $k > (4\mu+1)/3$. When μ is given, we always expect the smallest k. The C^1 splines are considered in our paper, so that the most interesting spline space is $S_2^1(\Delta_{m,n}^{(2)})$.

From Eq. (3.1a), we can get

$$\dim\, S_2^1(\Delta_{m,n}^{(2)}) = (m+2)(n+2) - 1. \tag{3.3}$$

We first introduce a locally supported spline in $S_2^1(\Delta_{m,n}^{(2)})$ with its support octagon Q centered at (0, 0) as shown in Fig. 3.2. It is known that a bivariate polynomial of degree 2 on a triangle can be uniquely determined by the values of three vertices and three midpoints of the edges. In Fig. 3.2, the values are given on some triangles, and other values are obtained by the symmetry of lines $x = 0, y = 0, x + y = 0, x - y = 0$.

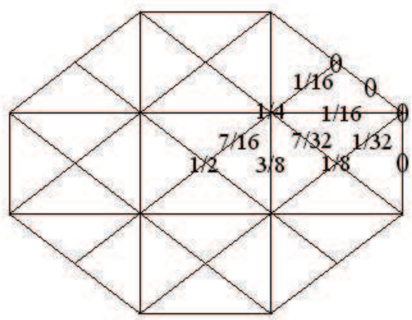


Fig. 3.2. A locally supported spline.

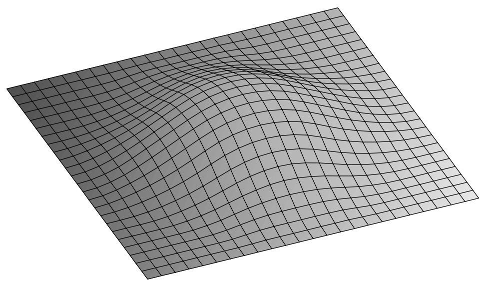


Fig. 3.3. Image of the locally supported spline.

Let $B(x, y)$ be a piecewise polynomial with degree 2 defined in R^2, that is, zero outside of Q, and let its representation in every triangle of Q be determined by the values. Clearly, $B(x, y) \in C^1(R^2)$, and $B(x, y) > 0$ inside of Q. Hence, $B(x, y)$ is a bivariate B-spline over the partition as shown in Fig. 3.3. Using the conformality conditions at vertices ([24]), the $B(x, y)$ is uniquely determined by the symmetry of lines $x = 0, y = 0, x + y = 0, x - y = 0$, and normalized condition $B(0, 0) = 1/2$. By the CSC method, we can point out that the support of $B(x, y)$ is the smallest one ([23]).

Denote

$$B_{ij}(x, y) := B(mx - i + 1/2,\ ny - j + 1/2), \tag{3.4}$$

then collection

$$A = \Big\{ B_{ij}(x, y) :\ i = 0, \cdots, m + 1,\ j = 0, \cdots, n + 1 \Big\}$$

is a subspace of $S_2^1(\Delta_{m,n}^{(2)})$. Note that each element of A is a nontrivial element of $S_2^1(\Delta_{m,n}^{(2)})$, and the number of elements in A is $(m+2)(n+2)$. From Eq. (3.3), A must be a linearly dependent set. Wang ([24]) gave the following results.

Theorem 3.2. ([24]) *The bivariate B-splines of A defined by Eq.* (3.4) *satisfy*

$$\sum_{i=0}^{m+1}\sum_{j=0}^{n+1}(-1)^{i+j}B_{ij} = 0.$$

For any $i_0, j_0, 0 \le i_0 \le m+1, 0 \le j_0 \le n+1$, the collection

$$A_{i_0 j_0} = \Big\{B_{ij}(x,y) \in A: \ (i,j) \neq (i_0, j_0)\Big\}$$

is a basis of $S_2^1(\Delta_{m,n}^{(2)})$.

Since the collection A of bivariate B-splines yield the entire space $S_2^1(\Delta_{m,n}^{(2)})$, therefore, for each bivariate spline $s(x,y) \in S_2^1(\Delta_{m,n}^{(2)})$, there must exist $a_{ij} \in R, i = 0, \cdots, m+1, j = 0, \cdots, n+1$, such that

$$s(x,y) = \sum_{i=0}^{m+1}\sum_{j=0}^{n+1} a_{ij}B_{ij}.$$

4. Surfaces' Energy

The bending properties of a plate depend greatly on its thickness as compared with its other dimensions. In the following, we shall study one kind of plates: thin plates with small deflections ([21]).

Definition 4.1. *If deflections ω of a plate are small in comparison with its thickness, we call it a thin plate with small deflections.*

A very satisfactory approximate theory of bending of the plate by lateral loads can be developed by making the following assumptions.

- There is no deformation in the middle plane of the plate. This plane remains neutral during bending.
- Points of the plate lying initially on a normal-to-the -middle plane of the plate remain on the normal-to-the-middle surface of the plate after bending.
- The normal stresses in the direction transverse to the plate can be disregarded.

Using these assumption, all stress components can be expressed by deflection ω of the plate, which is a function of the two coordinates in the plane of the plate, say $\omega = \omega(x,y)$.

If a plate is bent by uniformly distributed bending moments M_x and M_y (see Fig. 4.1) so that the xz and yz planes are the principal planes of the deflection surface of the plate, the strain energy stored in an element, such as shown in Fig. 4.2, is obtained by calculating the work done by the moments M_xdy and M_ydx on the element during bending of the plate ([21]). Since the sides of the element remain plane, the work done by the moments M_xdy is obtained by taking half the product of the moment and the angle between the corresponding sides of the element after bending. Since $-\partial^2\omega/\partial x^2$ represents the curvature of the plate in the xz plane, the angle corresponding to the moments $M_x dy$ is $-(\partial^2\omega/\partial x^2)dx$, and the work done by these moments is

$$-\frac{1}{2}M_x\frac{\partial^2\omega}{\partial x^2}dxdy.$$

An analogous expression is also obtained for the work produced by the moments $M_y dx$. Then the total work, equal to the strain energy of the element, is

$$-\frac{1}{2}\left(M_x\frac{\partial^2\omega}{\partial x^2}+M_y\frac{\partial^2\omega}{\partial y^2}\right)dxdy. \tag{4.1}$$

If the directions x and y do not coincide with the principal planes of curvature, there will act on the sides of the element (see Fig. 4.2) not only the bending moments $M_x dy$ and $M_y dx$ but also the twisting moments $M_{xy}dy$ and $M_{yx}dx$. In deriving the expression for the strain energy due to twisting moments $M_{xy}dy$ we observe that the corresponding angle of twist is equal to the rate of change of the slope $\partial\omega/\partial y$, as x varies, multiplied with dx; hence the strain energy due to $M_{xy}dy$ is

$$\frac{1}{2}M_{xy}\frac{\partial^2\omega}{\partial x\partial y}dxdy.$$

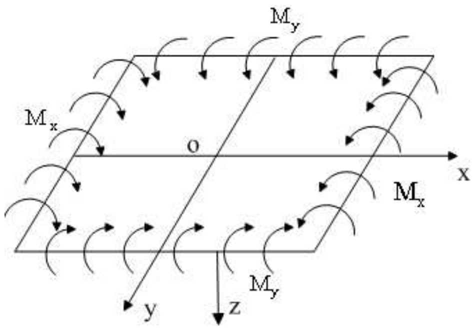


Fig. 4.1. Pure bending of thin plate.

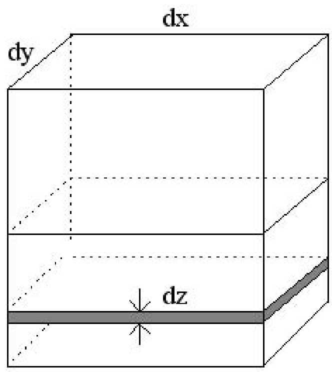


Fig. 4.2. The strain energy stored in an element.

The same amount of energy will also be produced by the couples $M_{yx}dx$, so that the strain energy due to both twisting couple is

$$M_{xy}\frac{\partial^2\omega}{\partial x\partial y}dxdy. \tag{4.2}$$

Since the twist does not affect the work produced by the bending moments, the total strain energy of an element of the plate is obtained by adding together the energy of bending Eq.(4.1) and the energy of twist Eq. (4.2). Thus we obtain

$$dV = -\frac{1}{2}\left(M_x\frac{\partial^2\omega}{\partial x^2} + M_y\frac{\partial^2\omega}{\partial y^2}\right)dxdy + M_{xy}\frac{\partial^2\omega}{\partial x\partial y}dxdy. \tag{4.3}$$

From [21], we have

$$M_x = -D_0\left(\frac{\partial^2\omega}{\partial x^2} + \nu\frac{\partial^2\omega}{\partial y^2}\right), \qquad M_y = -D_0\left(\frac{\partial^2\omega}{\partial y^2} + \nu\frac{\partial^2\omega}{\partial x^2}\right), \tag{4.4}$$

$$M_{xy} = D_0(1-\nu)\frac{\partial^2\omega}{\partial x\partial y}, \tag{4.5}$$

where D_0 is the flexural rigidity of the plate and ν is Poisson's ratio of the material, they are constants. $\omega = \omega(x,y)$ denotes small deflections of the plate.

Substituting Eqs. (4.4) and (4.5) into Eq. (4.3), the total strain energy of an element of the plate is represented in the following form:

$$dV = \frac{1}{2}D_0\left[\left(\frac{\partial^2\omega}{\partial x^2}\right)^2 + \left(\frac{\partial^2\omega}{\partial y^2}\right)^2 + 2\nu\frac{\partial^2\omega}{\partial x^2}\frac{\partial^2\omega}{\partial y^2}\right]dxdy + D_0\,(1-\nu)\left(\frac{\partial^2\omega}{\partial x\partial y}\right)^2 dxdy.$$

The strain energy of the entire plate is now obtained

$$V = \frac{1}{2}D_0\iint\left[\left(\frac{\partial^2\omega}{\partial x^2}\right)^2 + \left(\frac{\partial^2\omega}{\partial y^2}\right)^2 + 2\nu\frac{\partial^2\omega}{\partial x^2}\frac{\partial^2\omega}{\partial y^2} + 2\,(1-\nu)\left(\frac{\partial^2\omega}{\partial x\partial y}\right)^2\right]dxdy. \tag{4.6}$$

Let $\nu = 0$, Eq.(4.6) can be simplified as

$$V = \frac{1}{2}D_0\iint\left[\left(\frac{\partial^2\omega}{\partial x^2}\right)^2 + \left(\frac{\partial^2\omega}{\partial y^2}\right)^2 + 2\left(\frac{\partial^2\omega}{\partial x\partial y}\right)^2\right]dxdy.$$

In this paper, a fitting surface $f(x,y)$ to scattered data could be considered a thin plate with small deflections, so that it's total energy is

$$V = \frac{1}{2}D_0\iint\left[\left(\frac{\partial^2 f}{\partial x^2}\right)^2 + \left(\frac{\partial^2 f}{\partial y^2}\right)^2 + 2\left(\frac{\partial^2 f}{\partial x\partial y}\right)^2\right]dxdy. \tag{4.7}$$

5. Fitting Surface to Scattered Data

Recently, a B-spline approximation technique has been proposed for scattered data interpolation ([15]). In this section, we elaborate on scattered data interpolation with $S_2^1(\Delta_{m,n}^{(2)})$ and present the details of the algorithm.

Timoshenko pointed that if a system is a position of stable equilibrium, its total energy is a minimum ([21]). For given scattered data points, we want to find a new method to get a fitting surface $f(x,y)$, so that Eq. (4.7) is minimum. Fortunately, such method is proposed in this section.

5.1. The energy of the locally supported splines in $S_2^1(\Delta_{m,n}^{(2)})$

First of all, we talk about the energy of a locally supported spline in $S_2^1(\Delta_{m,n}^{(2)})$ mentioned in Section 3. Without loss of the generality, Fig. 3.2 is a locally supported spline in $S_2^1(\Delta_{m,n}^{(2)})$ with its support octagon Q centered at $(0,0)$. It is known that a bivariate polynomial of degree 2 on a triangle can be uniquely determined by the values of three vertices and three midpoints of the edges. In Fig. 3.2, the values are given on some triangles, and other values are obtained by the symmetry of lines $x=0, y=0, x+y=0, x-y=0$. So that the expression of this locally supported spline is given (see Fig. 5.1).

$$\begin{aligned}
q_1(x,y) &= -\frac{1}{2}x^2 - \frac{1}{2}y^2 + \frac{1}{2},\\
q_2(x,y) &= -\frac{1}{2}y^2 - \frac{1}{2}x + \frac{5}{8},\\
q_6(x,y) &= \frac{1}{4}x^2 - \frac{1}{2}xy - \frac{1}{4}y^2 - x + \frac{1}{2}y + \frac{7}{8},\\
q_7(x,y) &= \frac{1}{2}x^2 - \frac{3}{2}x + \frac{9}{8},\\
q_9(x,y) &= \frac{1}{4}x^2 + \frac{1}{2}xy + \frac{1}{4}y^2 - x - y + 1,
\end{aligned}$$

By the centrosymmetric of this locally supported spline, we obtain the expressions of $q_i(x,y), i = 3,4,5,8,10,11,\cdots,28$. Substituting these expressions into Eq. (4.7), we obtain the distributing of energy of a locally supported spline (see Fig. 5.2).

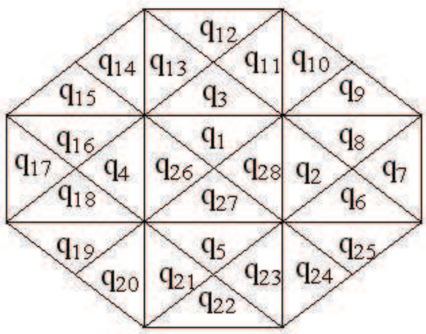


Fig. 5.1. The expression of a locally supported spline.

5.2. Basic idea

Without loss of generality, let $\Omega = [0,1]\otimes[0,1]$ be a square domain in the xy-plane. Consider a set of scattered data points $P = \{p_j = (x_j,\ y_j,\ z_j)\}_{j=1,\cdots,\ N}$ in $3D$ space, where $\tilde{p}_j := (x_j,\ y_j) \in \Omega$.

To approximate scattered data set P, a suitable partition $\Delta_{m,n}^{(2)}$ of domain Ω should be given first. Denote $B_{(x_0,y_0)}(x,y)$ a locally supported spline centered at (x_0,y_0). By moving $B_{(-1/2m,-1/2n)}(x,y)$ and using the following formulate

$$B_{((2i-1)/2m,(2j-1)/2n)}(x,y) = B_{(-1/2m,-1/2n)}(x-i/m, y-j/n),$$

the collection

$$\hat{A} = \left\{B_{((2i-1)/2m,(2j-1)/2n)}(x,y), i=0,1,\cdots,m+1,\ j=0,1,\cdots,n+1\right\}$$

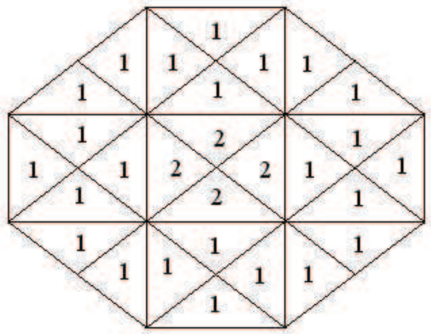


Fig. 5.2. The distributing of energy of a locally supported spline.

is a subspace of $S_2^1(\Delta_{m,n}^{(2)})$. Denote

$$\Phi = \Big\{B_{((2i-1)/2m,(2j-1)/2n)}(x,y) \in \hat{A}, i = 0, 1, \cdots, m+1, j = 0, 1 \cdots, n+1, (i,j) \neq (0,0)\Big\},$$

and reorder the elements in Φ. It follows from Theorem 3.2 that $\Phi = \{B_i(x,y),\ i = 1, 2, \cdots, d\}$ is a basis of $S_2^1(\Delta_{m,n}^{(2)})$, where $d = (m+2)(n+2) - 1$.

The approximation function $f(x,y)$ is defined in terms of Φ by

$$f(x,y) = \sum_{i=1}^{d} a_i B_i(x,y), \quad B_i(x,y) \in \Phi,\ i = 1, \cdots, d. \tag{5.1}$$

Denote $\Gamma = \{\triangle_i, i = 1, \cdots, 4mn\}$ the set of all the cells of partition $\Delta_{m,n}^{(2)}$, clearly, each element in Γ is a triangle. We say that $p \in \triangle^o$ if $\tilde{p}$ lies in $\triangle$. Denote $P_i = \{p \in P \mid p \in \triangle_i^o, \triangle_i \in \Gamma\}$, and $N_i = |P_i|$ the number of elements in P_i, that is, there are N_i scattered data in $\triangle_i$, $i = 1, \cdots, 4mn$. To ensure $\bigcup_{i=1}^{4mn} P_i = P$, $\Sigma_{i=1}^{4mn} N_i = N$, we use the following notions.

Given $p \in P$, ε is an arbitrary small positive number. Let $\tau = (c_1, c_2)$ be an orient vector, where $c_1 > 0,\ c_2 > 0$, and $mc_1 < nc_2$.

- If $\tilde{p}$ is an interior vertex, or a boundary vertex lying on $y = 0$, or a boundary vertex lying on $x = 0$, we say that $\tilde{p}$ and $\tilde{p} + \varepsilon\tau$ lie in the same cell $\triangle \in \Gamma$;
- If $\tilde{p}$ is a boundary vertex lying on $y = 1$, or a boundary vertex lying on $x = 1$, we say that $\tilde{p}$ and $\tilde{p} - \varepsilon\tau$ lie in the same cell;
- If $\tilde{p}$ lies on $x = i/m, i = 0, 1, \cdots, m-1$ (not vertex), $\tilde{p}$ and $\tilde{p} + \varepsilon e_1$ lie in the same cell, where $e_1 = (1,0)$ is the unit vector;
- If $\tilde{p}$ lies on $x = 1$ (not vertex), $\tilde{p}$ and $\tilde{p} - \varepsilon e_1$ lie in the same cell;
- If $\tilde{p}$ lies on $y = j/n, j = 0, 1, \cdots, n-1$ (not vertex), $\tilde{p}$ and $\tilde{p} + \varepsilon e_2$ lie in the same cell, where $e_2 = (0,1)$ is the unit vector;
- If $\tilde{p}$ lies on $y = 1$ (not vertex), $\tilde{p}$ and $\tilde{p} - \varepsilon e_2$ lie in the same cell.

Definition 5.1. *Suppose $\tilde{B} \in \Phi$, we call it an associated locally supported splines respect to the data point p_j, if it influences the value of f at $\tilde{p}_j := (x_j, y_j)$.*

Denote ϕ_j the set of associated locally supported splines respect to p_j. We discuss ϕ_j as follows.

Let $\triangle_1$ be the cell enclosed by the edges: $y = 0, mx + ny - 1 = 0, mx - ny = 0$, and let $\triangle_2$ be the cell enclosed by the edges: $x = 0, mx + ny - 1 = 0, mx - ny = 0$.

(i). If $p_j \in \triangle_1^o$,

$$\phi_j = \Big\{ B_{(1/2m,\ -1/2n)},\ B_{(3/2m,\ -1/2n)},\ B_{(-1/2m,\ 1/2n)},\ B_{(1/2m,\ 1/2n)},\ B_{(3/2m,\ 1/2n)},\ B_{(1/2m,\ 3/2n)} \Big\}.$$

(ii). If $p_j \in \triangle_2^o$,

$$\phi_j = \Big\{ B_{(1/2m,\ -1/2n)},\ B_{(-1/2m,\ 3/2n)},\ B_{(-1/2m,\ 1/2n)},\ B_{(1/2m,\ 1/2n)},\ B_{(3/2m,\ 1/2n)},\ B_{(1/2m,\ 3/2n)} \Big\}.$$

(iii). If p_j lies in the other cells, there are 7 locally supported splines which influence the value of $f(x_j, y_j)$, see Fig. 5.3.

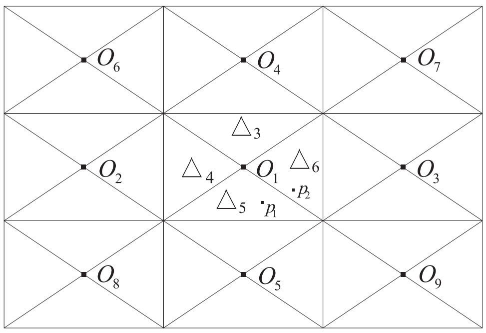


Fig. 5.3. Associated supported splines.

- If $p_j \in \triangle_3^o$, $\phi_j = \{B_{O_t},\ t = 1, \cdots, 7\}$, where B_{O_t} denote the locally supported spline centered at O_t;
- If $p_j \in \triangle_4^o$, $\phi_j = \{B_{O_8}, B_{O_t},\ t = 1, \cdots, 6\}$;
- If $p_j \in \triangle_5^o$, $\phi_j = \{B_{O_8}, B_{O_9}, B_{O_t},\ t = 1, \cdots, 5\}$;
- If $p_j \in \triangle_6^o$, $\phi_j = \{B_{O_7}, B_{O_9}, B_{O_t},\ t = 1, \cdots, 5\}$.

Obviously, from (i),(ii), (iii), we have

Theorem 5.1. *For each data point $p_j \in P, j = 1, \cdots, N$, there are $\mid T_j \mid$ locally supported splines influencing the value of the approximation function f at the data point $\tilde{p}_j := (x_j, y_j)$. It is said that*

$$f(x_j, y_j) = \sum_{i \in T_j} b_{ji} \tilde{B}_{ji}, \qquad j = 1, \cdots, N,$$

where $\tilde{B}_{ji} \in \phi_j$, $\quad T_j = \{t : B_{O_t} \in \phi_j\}$, *and* $\mid T_j \mid = \begin{cases} 6, & \textit{if } \ p_j \in \triangle_1^o, \\ 6, & \textit{if } \ p_j \in \triangle_2^o, \\ 7, & \textit{others.} \end{cases}$

We call $b_{ji}, i \in T_j$ the coefficients of associated supported splines respect to p_j. For approximation function f to take on the value z_j at (x_j, y_j), the locally supported splines $\tilde{B}_{ji}$ must satisfy

$$z_j = f(x_j, y_j) = \sum_{i \in T_j} b_{ji}\tilde{B}_{ji}, \quad \tilde{B}_{ji} \in \phi_j. \tag{5.2}$$

There are many values for the b_{ji}'s that satisfy Eq. (5.2). We choose one in the minimum energy sense that minimizes V_j, which denotes the energy of the surface obtained by the data point $p_j := (x_j, y_j, z_j)$. With the data point $p_j \in \triangle_5^o$, for example, from Theorem 5.1, we have

$$z_j = f(x_j, y_j) = \sum_{i=1}^{5} b_{ji}B_{O_i} + b_{j8}B_{O_8} + b_{j9}B_{O_9}. \tag{5.3}$$

Substituting Eq. (5.3) into Eq. (4.7), we have

$$V_j = \frac{10D_0}{mn}\mathbf{b_j}\mathbf{J}\mathbf{b_j}^\top,$$

where $\mathbf{b_j} = \begin{pmatrix} b_{j4} & b_{j2} & b_{j1} & b_{j3} & b_{j8} & b_{j5} & b_{j9}\end{pmatrix}$, and

$$\mathbf{J} = \begin{pmatrix} 32 & -8 & -4 & -8 & 2 & 0 & 2 \\ -8 & 32 & -4 & 0 & -4 & -8 & 2 \\ -4 & -4 & 32 & -4 & -8 & -4 & -8 \\ -8 & 0 & -4 & 32 & 2 & -8 & -4 \\ 2 & -4 & -8 & 2 & 32 & -4 & 0 \\ 0 & -8 & -4 & -8 & -4 & 32 & -4 \\ 2 & 2 & -8 & -4 & 0 & -4 & 32 \end{pmatrix}.$$

By solving the following programming:
(Ξ_j):

$$\begin{aligned} \min \quad & V_j = \frac{10D_0}{mn}\mathbf{b_j}\mathbf{J}\mathbf{b_j}^\top, \\ \text{s.t.} \quad & z_j = f(x_j, y_j) = \sum_{i=1}^{5} b_{ji}B_{O_i} + b_{j8}B_{O_8} + b_{j9}B_{O_9}, \end{aligned}$$

we obtain an unique group of solution for the b_{ji}'s that satisfy Eq. (5.3), notice that the surface

$$f(x, y) = \sum_{i=1}^{5} b_{ji}B_{O_i} + b_{j8}B_{O_8} + b_{j9}B_{O_9}$$

has minimum energy.

Now we consider all the data points in P. For each data point p_j, solving the programming (Ξ_j) can be used to determine $| T_j |$ coefficients of associated supported splines respect to it. We notice that, given different data points in P, their sets of associated locally supported splines may share elements in common probably. See Fig. 5.3, it is clear that, $\phi_1 \bigcap \phi_2 = \{B_{O_i},\ i = 1, \cdots, 5\}$, in another word, $B_{O_i},\ i = 1, \cdots, 5$ influence both p_1 and p_2, where $p_1 \in \triangle_5^o,\ p_2 \in \triangle_6^o$. This brings us some problems that we could not determine the coefficients of $B_{O_i},\ i = 1, \cdots, 5$. In order to solve this problem, we are able to consider a given locally supported spline and some data points lying in it.

If $\tilde{p}$ lies in a given locally supported spline $B_i(x,y)$, we say that, $p \in B_i^o$. Denote $\varsigma_i := \{p \in P \mid p \in B_i^o\}$ the set of data points associated with $B_i(x,y)$. Suppose that $\mid \varsigma_i \mid = r \neq 0$, for each element in ς_i, it's coefficient b_{ki}, which is respect to $B_i(x,y)$, can be derived by solving the programming (Ξ_k). To compromise among the coefficients, a_i is chosen to minimize error

$$e_i = \sum_{k=1}^{r} (b_{ki}B_i(p_k) - a_iB_i(p_k))^2.$$

The term $(b_{ki}B_i(p_k) - a_iB_i(p_k))$ is the difference between the real and expected contributions of a_i to function f at the data point $p_k \in \varsigma_i$. By differentiating the error e_i with respect to a_i, we get

$$a_i = \frac{\sum_{k=1}^{r} B_i^2(p_k)b_{ki}}{\sum_{k=1}^{r} B_i^2(p_k)}. \tag{5.4}$$

If $\mid q_i \mid = 0$, it is said that there is no data point associated with $B_i(x,y)$, a_i equals zero. Considering all of $B_i(x,y) \in \Phi, i = 1, \cdots, d$, we can get a sequence of suitable coefficients $a_i, i = 1, \cdots, d$. A desired approximation function f can be derived by Eq. (5.4).

5.3. Algorithm

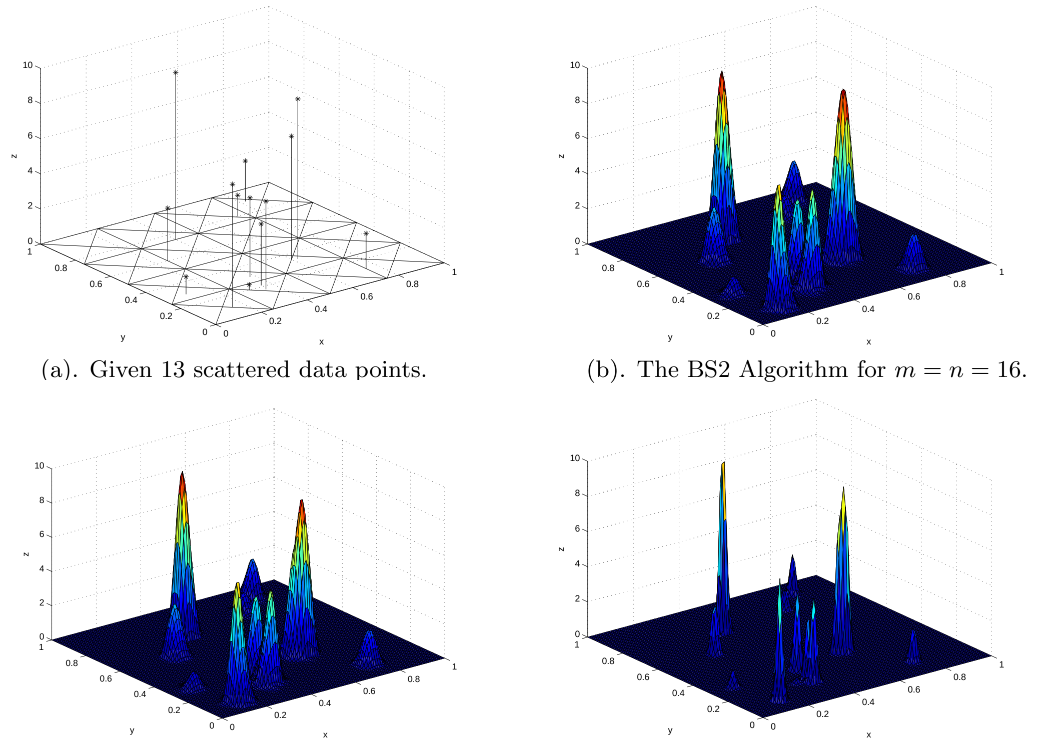
Since each data point in P is associated with $\mid T_j \mid$ locally supported splines in Φ, it belongs only to the set of associated locally supported splines respect to it. Hence, we can efficiently solve the programming (Ξ_k) for each locally supported spline by considering each data point in turn. The values of a group of the coefficients are then obtained. The following pseudocode outlines this approximation method, which we denote as the BS2 Algorithm.

BS2 Algorithm
Input: m, n, and $P = \{(x_j,\ y_j,\ z_j)\}_{j=1,\cdots,N}$.
Output: the coefficients of the locally supported splines, $a_i,\ i = 1, \cdots, d$.
compute $B_0,\ B_i,\ i = 1, \cdots, d$;
for all j, **do**
compute b_{ji} by solving the programming (Ξ_j);
for all i, **do**
compute a_i with Eq.(5.4).
end.

The time and space complexity of the BS2 algorithm is $\mathcal{O}(N+mn)$, where N is the number of data points, and $S_2^1(\Delta_{m,n}^{(2)})$ is the interpolation space. Although the coefficients are determined locally, we minimize the approximation error so that the resulting function properly reflects the scattered data. Fig. 5.4 shows several examples. The error here is 2-norm-error of 13 isolated points in Fig. 5.4(a). Fig. 5.4(b) shows the approximation function f derived by the BS2 algorithm for $m = n = 16$, the error is 1.729831e-003. Notice that f nicely approximates P at densely distributed data points and interpolates P at the isolated points.

The density of partition $\Delta_{m,n}^{(2)}$ overlaid on domain Ω directly affects the shape to approximation function f. As $\Delta_{m,n}^{(2)}$ becomes coarser, each of locally supported spline covers a larger number of data points in P. This causes many data points to be blended together to yield a smoother shape for f at the expense of approximation accuracy. However, if we select partition $\Delta_{m',n'}^{(2)}$, instead of $\Delta_{m,n}^{(2)}$, where $m' > m,\ n' > n$, the influence of a locally supported spline is limited to fewer data points in P. This enables P to be more closely approximated, although f will tend to contain local peaks near the data points. These characteristics are evident in Figs. 5.4(b) and 5.4(c), where $m = n = 16$ and $m = n = 32$, respectively. The error in Fig. 5.4(c) is 4.966154e-004.

The BS2 algorithm runs very fast even when the number of data points is large. Furthermore, since locally supported splines have local support, only a small neighborhood on domain Ω needs to be updated when a data point is added or deleted.



(a). Given 13 scattered data points.

(b). The BS2 Algorithm for $m = n = 16$.

(c). The BS2 Algorithm for $m = n = 32$.

(d). The MBS Algorithm.

Fig. 5.4. The BS2 Algorithm and the MBS Algorithm.

6. Multilevel Approximation

A tradeoff exists between the shape smoothness and accuracy of the approximation function generated by the BS2 algorithm. In this section, we present a multilevel approximation algorithm to circumvent this tradeoff. The resulting function simultaneously achieves a smooth shape while closely approximating the given data P. The algorithm makes use of a hierarchy of partitions to generate a sequence of functions f_k whose sum approaches the desired approximation function. In the sequence, a function from a coarse partition provides a rough approximation, which is further refined in accuracy by functions derived from finer partition.

Consider a hierarchy of partitions $\Delta^{(2)}_{m^{(k)},\ n^{(k)}}$, where $k = 0, \cdots h$, $m^{(k)} = 2m^{(k-1)}$, $n^{(k)} = 2n^{(k-1)}$, the multilevel approximation begins by applying the BS2 algorithm to P with the coarsest partition $\Delta^{(2)}_{m^{(0)},\ n^{(0)}}$. The resulting function f_0 serves as a smooth initial approximation that possibly leaves large discrepancies at the data points in P. In particular, f_0 leaves a deviation $\delta_1 z_j = z_j - f_0(x_j,\ y_j)$ for each point $p_j \in P$. The next finer partition $\Delta^{(2)}_{m^{(1)},\ n^{(1)}}$ is then used to obtain function f_1 that approximates the difference $P_1 = \{(x_j,\ y_j,\ \delta_1 z_j)\}$. Then, the sum $f_0 + f_1$ yields a smaller deviation $\delta_2 z_j = z_j - f_0(x_j,\ y_j) - f_1(x_j,\ y_j)$ for p_j.

In general, for a level k in the hierarchy, we derive function f_k by using partition $\Delta^{(2)}_{m^{(k)},\ n^{(k)}}$ to approximate data $P_k = \{(x_j,\ y_j,\ \delta_k z_j)\}$, where

$$\delta_k z_j = z_j - \sum_{i=0}^{k-1} f_i(x_j,\ y_j) = \delta_{k-1} z_j - f_{k-1}(x_j,\ y_j),$$

and $\delta_0 z_j = z_j$. This process starts from the coarsest partition $\Delta^{(2)}_{m^{(0)},\ n^{(0)}}$ and continues incrementally to the finest partition $\Delta^{(2)}_{m^{(h)},\ n^{(h)}}$. The final approximation function f is defined as the sum of functions $\{f_k\}$, i. e., $f = \sum_{k=0}^{h} f_k$. Note that only the coarsest partition $\Delta^{(2)}_{m^{(0)},\ n^{(0)}}$ is applied to the original data P to derive the global shape of function f. All successively finer partitions serve to approximate and remove the residual error. In this manner, we have an incremental solution for function f that yields a smooth and close approximation to P. The following pseudocode outlines the BS2 algorithm for multilevel spline approximation, which we denote as the MBS algorithm. Note that a hierarchy of partitions is sufficient to represent function f, because each of f_k can be represented by $S^1_2(\Delta^{(2)}_{m^{(k)},\ n^{(k)}})$, and f is the sum of the f_k's.

MBS Algorithm
Input: $m, n, m^{(0)}, n^{(0)}, e$, and $P = \{(x_j,\ y_j,\ z_j)\}_{j=1,\cdots,N}$.
Output: an approximation function f.
let $k = 0$;
while $E > e$, **do**
$k = k + 1$
let $P_k = \{(x_j,\ y_j,\ \delta_k z_j)\}$, $m^{(k)} = 2m^{(k-1)},\ n^{(k)} = 2n^{(k-1)}$;
compute f_k from P_k by the BS2 Algorithm;
compute $\delta_{k+1} z_j = \delta_k z_j - f_k(x_j,\ y_j)$ for each data point;
$E = \sum_{j=1}^{N} (\delta_{k+1} z_j)^2$.
end

Let N be the number of data points, and $\Delta^{(2)}_{m,n}$ be the finest partition. The time complexity of the MBS algorithm is

$$\mathcal{O}(N + mn) + \mathcal{O}(N + mn/4) + \cdots + \mathcal{O}(N + mn/4^k) = \mathcal{O}(N + 4mn/3)$$

and the space complexity is $\mathcal{O}(N + 4mn/3)$ because we have to store all the coefficients in the hierarchy. A function obtained from the MBS algorithm is C^1-continuous because it is the sum of C^1-continuous splines. Fig. 5.4(d) shows an example, the given data is the same as that in Fig. 5.4(a), $m^{(0)} = n^{(0)} = 4$ and $m^{(h)} = n^{(h)} = 64$, the error is 4.577080e-005.

Notice that multilevel approximation generates a much smoother and more accurate function than spline approximation given in the last section. Recall that function f_k, for level $k > 0$ in the hierarchy, is derived to approximate and remove the residual error $\delta_k z_j$ at each data point. The final function f is made to interpolate data P once this error goes to zero at some level k.

7. Numerical Results

Fig. 5.4 illuminate clearly that the BS2 Algorithm and the MBS Algorithm could generate interpolation functions from some given isolated points. In this section, we made two tests. Example 1 is surface reconstructing with some data points lying on the known surface; Example 7.2 is surface fitting with some data points which have measurement errors, namely, given data points $P = \{p_j = (x_j,\ y_j,\ z_j)\}_{j=1,\cdots,\ N}$, there exits a relationship $z_j = f(x_j,\ y_j) + \xi_j$, where ξ_j represents the unknown error, we want to get f or a function approximating it.

Both of examples were implemented in MATLAB. MQ and CSF we used here are

$$R_{mq} = \sqrt{c^2 + r^2},$$

and

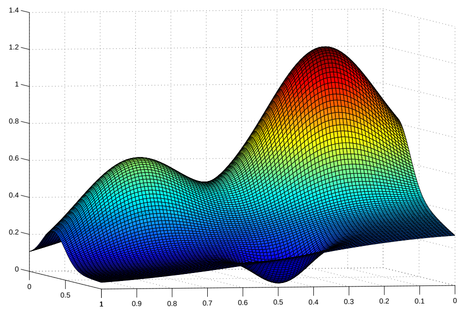
$$R_{csf} = max(0,\ (1-r)^4) \cdot (1 + 4r),$$

respectively. The errors mentioned here are ∞-norm-error.

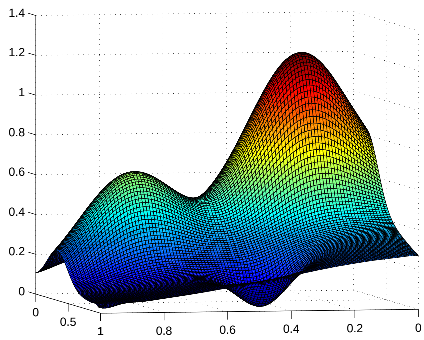
Example 7.1. The test surface is (see Fig. 7.1(a))

$$f(x,y) = \frac{3}{4}e^{-\frac{(9x-2)^2+(9y-2)^2}{4}} + \frac{3}{4}e^{-\frac{(9x+1)^2}{49} - \frac{9y+1}{10}} - \frac{1}{5}e^{-(9x-4)^2-(9y-7)^2} + \frac{1}{2}e^{-\frac{(9x-7)^2+(9y-3)^2}{4}}.$$

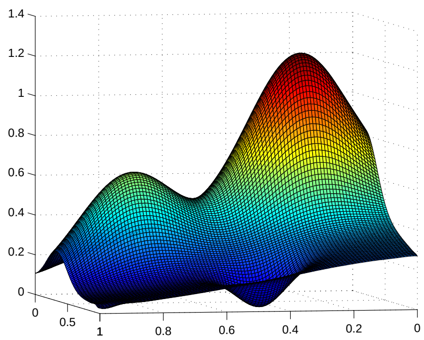
We select 289 uniformity data points on the test surface. Fig. 7.1(b) shows the interpolation function derived by the BS2 Algorithm for $m = n = 32$. Fig. 7.1(c) was obtained by R_{mq}, where



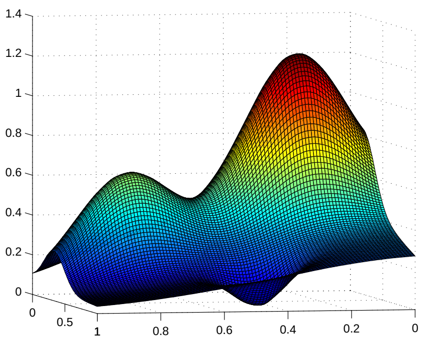
(a). Test function $f(x,y)$.



(b). The BS2 Algorithm, error is $5.767472e-002$.



(c). MQ, error is 5.393265e-002.



(d). CSF, error is 1.855920e-002

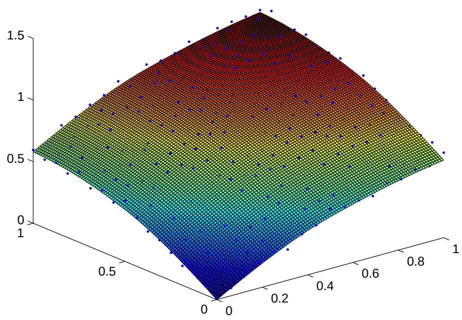
Fig. 7.1. Fitting f(x,y) with the BS2 Algorithm, MQ, and CSF.

$c = 0.1$. Fig. 7.1(d) was obtained by R_{csf}. Obviously, in Fig. 7.1, three methods generate three interpolation surfaces, which have very little difference from the vision. Actually, the errors also suggest that the BS2 Algorithm is suitable for fitting surfaces, and its accuracy is fine. It should be pointed out that, the choice of the parameter c is important and difficult when we use MQ to reconstruct surfaces. Moreover, MQ and CSF need to solve some equations which coefficients matrices will be ill.

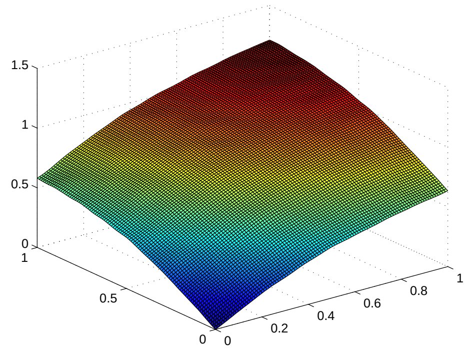
Example 7.2. Given a surface $g(x, y)$ and some data points $(x_j,\ y_j,\ z_j)$, where $z_j = g(x_j,\ y_j)+\xi_j$, ξ_j represents the unknown error (see Fig. 7.2(a)).

$$g(x,y) = \begin{cases} -x^2 - y^2 + 2.1, & x \le 0.5,\ y \le 0.5, \\ -x^2 - 3y^2 + 2y + 1.6, & x \le 0.5,\ y \ge 0.5, \\ -4x^2 - 3y^2 + 3x + 2y + 0.85, & x \ge 0.5,\ y \ge 0.5, \\ -4x^2 - y^2 + 3x + 1.35, & x \ge 0.5,\ y \le 0.5. \end{cases}$$

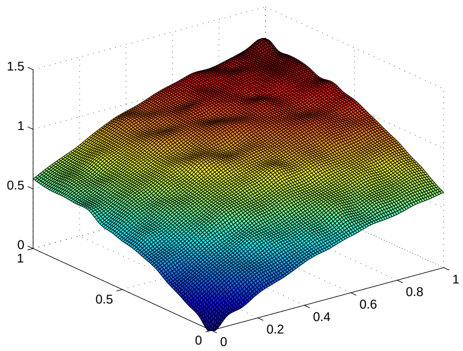
Obviously, $g(x, y) \in S_2^1(\Delta)$. Fig. 7.2(b) shows the interpolation function derived by the BS2 Algorithm for $m = n = 48$. Fig. 7.2(c) was obtained by R_{mq}, where $c = 0.01$. Fig. 7.2(d) was obtained by R_{csf}. We see that, Fig. 7.2(c) and Fig. 7.2(d) are accidented, since MQ and CSF



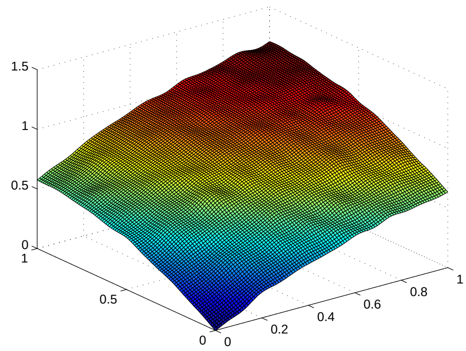
(a). $g(x, y)$, and some given data points.



(b). the BS2 Algorithm, error is 3.630491e-003.



(c). MQ, error is 6.090860e-002.



(d). CSF, error is 4.855920e-002.

Fig. 7.2. Fitting g(x,y) with the BS2 Algorithm, MQ, and CSF.

represent their abilities of interpolation. They generate surfaces that passes though the given data points which have measurement errors, obviously, both of MQ and CSF are not suitable for fitting surfaces if the given data points have some measurement errors. The BS2 Algorithm is more suitable for solving this problem because it's ability of approximation is fine, Fig. 7.2(b) indicates this point.

From Examples 7.1 and 7.2, we can see that, the BS2 Algorithm not only have the ability of interpolation, but also approximation. This advantage could be used to fit surface more convenient. Moreover, this method is easy to realize.

8. Conclusions and Future Works

This paper presented a fast approximation and interpolation algorithm for scattered data points. The algorithm is based on spline approximation with $S_2^1(\Delta_{m,n}^{(2)})$. A sequence of coefficients is efficiently determined by minimizing a local approximation error for each data point. The resulting C^1-continuous function passes near densely distributed data points and interpolates isolated points. However, a tradeoff exists between the shape smoothness and approximation accuracy of the function, depending on the partition density.

Multilevel spline approximation was introduced to circumvent this tradeoff. The algorithm makes use of a hierarchy of partitions to generate a sequence of functions whose sum approaches the desired approximation function. In the sequence, a function from the coarsest partition provides an initial estimate, which is further refined in accuracy by functions derived at finer levels. Interpolation is achieved when the finest partition becomes sufficiently small relative to the data distribution.

In this paper, in order to get the coefficients of associated supported splines respect to p_j, we use minimum energy and the programming (Ξ_j). Actually, minimal area or curvature should be considered in some cases. Moreover, quasi-interpolation will be considered in future works.

Acknowledgments. This work was supported by the National Natural Science Foundation of China (Nos. U0935004,11071031,11071037,10801024), and the Fundamental Funds for the Central Universities. should be changed to Acknowledgments. This work is partly supported by the National Natural Science Foundation of China (Nos. U0935004,11071031,10801024), the Fundamental Funds for the Central Universities (DUT10ZD112, DUT11LK34), and National Engineering Research Center of Digital Life, Guangzhou 510006, China.

References

[1] R.E. Barnhill, Representation and Approximation of Surfaces, in: Mathematical Software III (J. R. Rice, Ed.), Academic Press, Newyork, 1977, 69-120.

[2] C. de Boor and J.R. Rice, Least Squares Cubic Splines Approximation. I: Fixed Knots, CSD TR 20, Purdue Univ., Lafayette, 1968.

[3] C. de Boor and G.J. Fix, Spline approximation by quasi-interpolants, *J. Approx. Theory*, **8** (1973), 19-45.

[4] M.D. Buhmann, Radial functions on compact support, *Proc. Edinb. Math. Soc.*, **41** (1998), 33-46.

[5] M.D. Buhmann, A new class of radial basis functions with compact support, *Math. Comput.*, **70** (2001), 307-318.

[6] F. Cheng and B.A. Barsky, Interproximation: interpolation and approximation using cubic spline curves, *Comput. Aided Design*, **10** (1991), 700-706.

[7] R. Franke and G.M. Nielson, Smooth interpolation of large sets of scattered data, *Int. J. Numerical Methods in Eng.*, **15** (1980), 1691-1704.

[8] R. Franke and G.M. Nielson, Scattered Data Interpolation and Applications: A Tutorial and Survey, in: H. Hagen and D. Roller (Eds.), Geometric Modelling: Methods and Their Application, Springer-Verlag, Berlin, 1991, 131-160.

[9] H. Hagen and G. Schulze, Automatic smoothing with geometric surface patches, *Computer Aided Geometric Design*, **4** (1987), 231-236.

[10] R. Hardy, Multiquadratic equations of topography and other irregular surfaces, *J. Geophys. Res.*, **76**:8 (1971), 1905-1915.

[11] J. Hoschek and D. Lasser, Computer Aided Geometric Design, A. K. Peters, 1993.

[12] M. Kallay, A method to approximate the space curve of minimal energy and prescribed length, *Comput. Aided Design*, **19**:2 (1987), 73-76.

[13] M. Kallay and B. Rarani, Optimal twist vectors as a tool for interpolating a network of curves with a minimum energy surface, *Comput. Aided Geom. D.*, **7** (1990), 465-473.

[14] C. Lawson, Software for C^1 surface interpolation. in: Mathematical Software III (J. R. Rice, Ed.), Academic Press, Newyork, 1977, 161-194.

[15] S. Lee, G. Wolberg, and S.Y. Shin, Scattered data interpolation with multilevel B-splines, *IEEE T. Vis. Comput. Gr.*, **3**:3 (1997), 228-244.

[16] T. Lyche and L. Schumaker, Local spline approximation methods, *J. Approx. Theory*, **15** (1975), 294-325.

[17] E. Quak and L.L. Schumaker, Calculation of the energy of a piecewise polynomial surface, in: M. G. Cox and J. C. Mason (Eds.), Algorithms for Approximation II, Clarendon Press, Oxford, 1989, 134-143.

[18] E. Quak and L.L. Schumaker, Cubic spline fitting using data dependent triangulations, *Comput Aided Geom. D.*, **7** (1990), 293-301.

[19] D. Shepard, A two dimentional interpolation function for irregularly spaced data. in: Proc. ACM 23rd Nat'l Conf., 1968, 517-524.

[20] R. Szeliski, Fast surface interpolation using hierarchial basis functions, *IEEE Trans. Pattern Anal. Machine Intell.*, **12** (1990), 513-528.

[21] S. Timoshenko and S. Woinowsky-Krieger, *Theory of Plates and Shells*, Mcgraw-Hill International Book Company, New York, 1959.

[22] R.H. Wang, The structural characterization and interpolation for multivariate splines, *Acta Math Sin.*, **18**:2 (1975), 91-106. (English transl., ibid. 18 (1975), 10-39.

[23] R.H. Wang, The dimension and basis of spaces of multivariate splines, *J. Comput. Appl. Math.*, **12-13** (1985), 163-177.

[24] R.H. Wang, Multivariate Spline Function and Their Applications, Science Press/Kluwer Acad. Pub., Beijing/New York, 2001.

[25] R.H. Wang, C.J. Li and C.G. Zhu, A Course for Computational Geometry, Science Press, Beijing, 2008.

[26] X.F. Wang, F. Cheng and B.A. Barsky, Energy and B-spline interproximation, *Computer-Aided Design*, **29** (1997), 485-496.

[27] H. Wendland, Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv. Comput. Math.*, **4** (1995), 389-396.

[28] H. Wendland, Scattered Data Approximation, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, 2005.

[29] Z.M. Wu, Compactly supported positive definite radial functions, *Adv. Comput. Math.*, **4** (1995), 283-292.