On a Construction of a Hierarchy of Best Linear Spline Approximations Using a Finite Element Approach

David F. WILEY [†], Martin BERTRAM [*,†], and Bernd HAMANN [†]

Abstract

We present a method for the hierarchical approximation of functions in one, two, or three variables based on the finite element method (Ritz approximation). Starting with a set of data sites with associated function, we first determine a smooth (scattered-data) interpolant. Next, we construct an initial triangulation by triangulating the region bounded by the minimal subset of data sites defining the convex hull of all sites. We insert only original data sites, thus reducing storage requirements. For each triangulation we solve a minimization problem: computing the best linear spline approximation of the interpolant of all data, based on a functional involving function values and first derivatives. The error of a best linear spline approximation is computed in a Sobolev-like norm, leading to element-specific error values. We use these interval-/triangle-/tetrahedron-specific values to identify the element to subdivide next. The subdivision of an element with largest error value requires the re-computation of all spline coefficients due to the global nature of the problem. We improve efficiency by (i) subdividing multiple elements simultaneously and (ii) by using a sparse-matrix representation and system solver.

Key words: Approximation; Finite element method; Grid generation; Multiresolution

[†] *Center for Image Processing and Integrated Computing (CIPIC), Department of Computer Science, University of California, Davis, CA 95616-8562, U.S.A.*; e-mail: {wiley, bertram, hamann}@cs.ucdavis.edu

[*] *University of Kaiserslautern, Fachbereich Informatik, P.O. Box 3049, D-67653 Kaiserslautern, Germany.*

method; Optimization; Ritz approximation; Scattered data; Spline; Triangulation; Unstructured grid; Visualization.

---

1. Introduction

Different methods are known and used for the hierarchical representation of very large data sets. Few methods are based on a well developed mathematical theory. In the context of visualizing very large data sets, it is imperative to develop hierarchical data representations that allow us to visualize and analyze data at various levels of detail. General and efficient methodologies are needed to support the generation of hierarchical data representations and their applicability for the visualization process.

This paper deals with the construction of hierarchies of triangulations and best linear spline approximations of functions. The main idea underlying the construction is the method of *finite elements*, see [Zienkiewicz & Taylor '00]. We minimize certain energy functionals by solving a finite-element problem for triangulated domains. We construct an approximation hierarchy by repeatedly subdividing triangulations and computing a best linear spline approximation for each one of them. Our initial approximation is based on a subset of the given data sites defining the convex hull of all sites. We then identify regions with relatively large error values and refine a triangulation by inserting more original data sites into the vertex set.

We compute the coefficients for each approximation in a best-approximation sense. Whenever we subdivide an interval (triangle, tetrahedron) one must compute new linear spline coefficients for all vertices in a new triangulation due to the global nature of the

minimization approach. It is possible to perform the necessary matrix inversions rather efficiently by using a sparse-matrix representation and system solver. We can further improve the efficiency of the finite element approach by inserting multiple original data sites simultaneously into an intermediate triangulation.

The generation of a hierarchy of data approximations is a pre-processing step for subsequent data visualization and analysis. The goal is to be able to utilize the data format describing a data hierarchy directly for the visualization process. Visualization technology and systems are capable of rendering triangulation-based data rather efficiently. Considering the generation of a data-approximation hierarchy, speed is not the primary concern; it is far more important to construct a hierarchy that can be stored compactly and supports interactive data visualization and analysis. It is important that hierarchical data approximations can handle arbitrarily complex regions and that error estimates are known for all approximation levels. The main criteria that have influenced our algorithm design are: simplicity, generality (in a multi-variate sense), efficiency, and compactness (in the sense of storage). Computing a hierarchy of best linear spline approximations, based on different triangulations of only original data sites, seems to be a good approach in this context.

Remark 1.1. We point out that, in many practical applications, the function to be approximated is only known at a finite number of random locations. If this is the case, a fitting step must take place. A localized version of *Hardy's multiquadric* method, for example, or a surface reconstruction technique similar to the one described in [Floater & Reimers '01] can be used to yield a smooth function interpolating the given discrete data.

We interpret this interpolant as the function for which an approximation hierarchy must be constructed. For a survey on scattered-data approximation and interpolation techniques, including a description of Hardy's multiquadric method, see [Franke '82] and [Nielson '93].

Over the past decade, various approaches supporting hierarchical data representations have been developed—and there is an increasing need for such representations. Triangulation-based schemes are a good choice for the representation of data defined for complicated regions in space. Triangulations offer a large degree of flexibility for discretizing space, and this is the main reason why we are using them.

Recent work on several related strategies exists. Wavelet techniques that produce multiresolution meshes discussed in [Bonneau et al. '96], [Gross et al. '95], [Staadt et al. '97], and [Nielson et al. '97a] are all limited to uniform grids while the latter is restricted to spherical grids. Staadt's method has the advantage of supporting both lossy and lossless compression or progressive and selective mesh refinement. Bonneau's method permits the user to choose between smooth and sharp approximation of discontinuities. This works well for entirely smooth or discontinuous data, but does not work well for data containing both types of regions.

[Cignoni et al. '94, '97], [Dyn et al. '90], [Gieng et al. '97, '98], [Hamann '94], [Hoppe '96, '97], and [Trotts et al. '98, '99] use simplification techniques, such as decimation and thinning, to reduce the input mesh while respecting data values. These methods produce multiresolution output that can often be progressively displayed. These methods are much more applicable than most wavelet methods since arbitrary input meshes can be converted to meet the requirements of the method.

Refinement techniques such as the ones described in [Grosso et al. '97], [Hamann et al. '99], and [Rippa '92], describe data-dependent multiresolution methods that are quite similar to the method we present here. These methods begin with a compact initial triangulation of the convex hull of the data to be approximated and then iteratively refine this mesh. These methods are generally applied to arbitrary meshes by converting any given mesh to the specific mesh type required.

Much work was done on data-dependent methods. For example, [Agarwal & Desikan '97] describe a method to approximate terrain data with a piecewise linear function producing an ϵ-approximation (an approximation within some tolerance ϵ) in $O(n^{2+a})$ time where n is the number of knots and a is an arbitrarily small positive number. [Hamann & Chen '94] and [Nadler '86] similarly produce piecewise linear functions to approximate bivariate data.

[Floater & Iske '96a, '96b] describe multiresolution methods that are not data dependent. These methods perform an iterative thinning step using radial basis functions on scattered points while maintaining a Delaunay triangulation of the points throughout the process. They propose several knot removal criteria that allow the user to balance speed and approximation quality. [Dyn et al. '00] describes a data-dependent thinning method similar to Floater & Iske's methods. Their results support the hypothesis that data-dependent techniques concentrate simplices around high-gradient areas producing higher-quality approximations.

Additional work in the area of data-dependent triangulations, multiresolution, and mesh decimation can be found in [Eck et al. '95] and [Xia & Varshney '96].

We present a data-dependent refinement method that captures the spirit of the previous work referenced above. Our method can be applied to arbitrary meshes and it can be applied to multidimensional domains. An important contribution of our work is use of first-derivative input to guide the approximation process. We use this information for error calculation and to select simplices for refinement as well as the production of the piecewise linear spline coefficients. The use of this information improves the resulting approximation by focusing the refinement on high-gradient areas and by correcting over- and under-shoots in the regions right next to discontinuities.

The approach we present in this paper applies to univariate, bivariate, and trivariate data. The main principles of our approach become evident from an in-depth discussion of the univariate case (Section 2). By providing an in-depth discussion of the univariate case, the required generalizations to the bivariate and trivariate cases (Sections 3 and 4) are more easily understood.

Eventually, we use the resulting hierarchical data representations for visualization. We apply standard visualization techniques (slicing) to our data hierarchies, see Section 5. Visualization methods are described in great detail in [Hagen et al. '93], [Kaufman '91], [Nielson et al. '97b], [Nielson & Shriver '90], and [Rosenblum et al. '94].

2. The univariate case

We consider first-derivative information in our method as well as functional data to potentially better represent data in high-gradient areas. Before describing our approximation technique, we review some required definitions and terminology. Our approximation approach requires concepts from linear algebra and approximation theory that we discuss

briefly. We define the inner product $\langle f, g \rangle$ of two functions $f(x)$ and $g(x)$, defined over the interval $[a, b]$, as

$$\langle f, g \rangle \;=\; \int_a^b w_0 f(x) g(x) + w_1 f'(x) g'(x)\; dx. \tag{1}$$

The "weights" w_0 and w_1 are user-specified, non-negative, and sum to one. This inner product defines a *Sobolev*-like L_2 norm—that we denote by $||\ ||_{\mathrm{Sob}}$—for a function $f(x)$:

$$||f||_{\mathrm{Sob}} \;=\; \langle f, f \rangle^{1/2} \;=\; \left(\int_a^b w_0 \big(f(x)\big)^2 + w_1 \big(f'(x)\big)^2\; dx \right)^{1/2}. \tag{2}$$

This norm allows us to measure the quality of an approximation $f(x)$ to a given function $F(x)$, by considering the norm of $D(x) = F(x) - f(x)$:

$$||D||_{\mathrm{Sob}} \;=\; \langle D, D \rangle^{1/2}. \tag{3}$$

We use this measure to compute interval-specific error values for the individual intervals $[x_i, x_{i+1}]$ with lengths $\triangle_i = x_{i+1} - x_i$. For a subdivision step, we identify an interval with maximal error and split it into two intervals. This error measure is appropriate because it permits the addition of first-derivative information. This is important since scientific data is often characterized by high gradients. By controlling first-derivative influence with user-specified weights, it is possible to force mesh refinement to be done in regions characterized by high gradients.

This is an outline of the overall algorithm we use to compute multiple best linear spline approximations in the univariate case:

I. Input. The input is a sequence of knots, $x_0 < \cdots < x_N$, with associated function values F_i, defining linear spline segments $s_i(x)$ defined as

$$s_i(x) \;=\; F_i \;\frac{x_{i+1}-x}{\triangle_i} \;+\; F_{i+1}\;\frac{x-x_i}{\triangle_i}, \qquad i=0,\cdots,N-1. \tag{4}$$

We can construct a continuous interpolant $F(x)$ from the given data by using, for example, the piecewise linear spline implied by the given knot set and function values.

II. First approximation. Compute a best linear spline approximation based on only the two end knots and compute the error, based on equation (3). Should the error value of this approximation be too large (larger than a user-specified tolerance) continue with step III, otherwise terminate.

III. Subdivision/knot selection. Identify an interval with maximal error and an original data site that is closest to the midpoint of the chosen interval. Insert this knot into the set of selected knots. (Alternatively, apply this strategy to the k intervals with largest error values and subdivide them simultaneously.)

IV. Next approximation. Compute a new best linear spline approximation based on the new knot set and order the spline segments according to the new interval-specific error values. Terminate the subdivision process when the maximal interval-specific error value is smaller than some user-specified tolerance, otherwise repeat steps III and IV.

V. Output. The output consists of subsets of the original knot set, defining the various levels of a hierarchy of best linear spline approximations; we store each best linear spline approximation as a set of spline coefficients.

We now discuss these steps in detail. The functional that we minimize in the univariate case is given by the expression

$$E \;=\; \int_a^b w_0\big(F(x)-f(x)\big)^2 + w_1\big(F'(x)-f'(x)\big)^2\; dx, \tag{5}$$

where F is the given function to be approximated on the interval $[a,b]$ by the function f. We assume that the "weights" w_i are non-negative and sum to one. Thus, equation (5) is equivalent to

$$\begin{aligned} E \;=\;& \int_a^b w_0\big(f(x)\big)^2 + w_1\big(f'(x)\big)^2 \;-\; 2\big(w_0F(x)f(x) + w_1F'(x)f'(x)\big)\; dx \\ &+ \int_a^b w_0\big(F(x)\big)^2 + w_1\big(F'(x)\big)^2\; dx. \end{aligned} \tag{6}$$

The minimization problem reduces to minimizing

$$\begin{aligned} E \;=\;& \int_a^b \frac{1}{2}\begin{pmatrix} f(x) & f'(x)\end{pmatrix}\begin{pmatrix} 2w_0 & 0 \\ 0 & 2w_1\end{pmatrix}\begin{pmatrix} f(x) \\ f'(x)\end{pmatrix} - \begin{pmatrix} f(x) & f'(x)\end{pmatrix}\begin{pmatrix} 2w_0F(x) \\ 2w_1F'(x)\end{pmatrix}\; dx \\ =\;& \int_a^b \frac{1}{2}\;\mathbf{f}^TQ\mathbf{f} - \mathbf{f}^T\mathbf{l}\; dx. \end{aligned} \tag{7}$$

(The superscript "T" denotes the transpose operator.) We solve the minimization problem by defining the approximation f to be the linear combination

$$f(x) \;=\; c_0f_0(x) + c_1f_1(x) + \cdots + c_nf_n(x) \;=\; \sum_{i=0}^{n} c_i\; f_i(x), \tag{8}$$

based on an associated knot sequence $x_0 < x_1 < \cdots < x_n$ ($x_0 = a$, $x_n = x_N = b$, $n \le N$). The basis function f_i is the hat function defined over the interval $[x_{i-1}, x_{i+1}]$ (outside of

which it is zero) satisfying the property $f_i(x_j) = \delta_{i,j}$ (*Kronecker delta*), see Fig. 1.

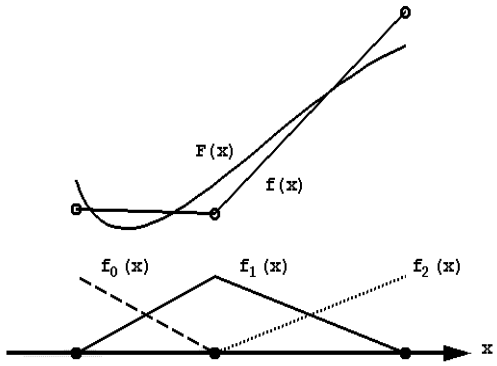

Fig. 1. Basis functions $f_i(x)$, function $F(x)$, and approximation $f(x)$.

Substituting equation (8) into equation (7) yields the matrix expression

$$\begin{aligned} E \;=\; & \frac{1}{2}\,(\,c_0 \quad c_1 \quad \dots \quad c_n\,) \begin{pmatrix} \int q(f_0,f_0) & \dots & \int q(f_0,f_n) \\ \vdots & & \vdots \\ \int q(f_n,f_0) & \dots & \int q(f_n,f_n) \end{pmatrix} \begin{pmatrix} c_0 \\ \vdots \\ c_n \end{pmatrix} \\ & - \quad (\,c_0 \quad c_1 \quad \dots \quad c_n\,) \begin{pmatrix} \int l(f_0) \\ \vdots \\ \int l(f_n) \end{pmatrix} \\ = \; & \frac{1}{2}\,\mathbf{c}^T A\mathbf{c} - \mathbf{c}^T\mathbf{l}, \end{aligned} \tag{9}$$

where $q(f_i, f_j)$ is quadratic and $l(f_i)$ linear in f_i and f_j and their derivatives. This *energy* term E is minimal for the set of coefficients c_i resulting from the *Ritz equations*, i.e., the linear system of equations

$$A\mathbf{c} \;=\; \mathbf{l}, \tag{10}$$

see [Boehm & Prautzsch '93]. The elements $a_{i,j}$ of the symmetric, positive definite matrix A are given by

$$a_{i,j} \;=\; w_0 \int_a^b f_i(x) f_j(x)\; dx \;+\; w_1 \int_a^b f_i{}'(x) f_j{}'(x)\; dx, \qquad i,j = 0,\cdots,n, \tag{11}$$

and the elements l_i of the column vector $\mathbf{l}$ are given by

$$l_i \;=\; w_0 \int_a^b F(x) f_i(x)\; dx \;+\; w_1 \int_a^b F'(x) f_i{}'(x)\; dx, \qquad i = 0, \cdots, n. \tag{12}$$

The integral expressions required to compute the elements of the matrix A are provided in the Appendix, Section A.1. The matrix A is a tridiagonal matrix, given by a weighted sum of two tridiagonal matrices, A_0 and A_1:

$$A \;=\; \frac{1}{6}\,\big(w_0\; A_0 + w_1\; A_1\big). \tag{13}$$

The matrices A_0 and A_1 are given by

$$A_0 \;=\; \begin{pmatrix} 2\triangle_0 & \triangle_0 & & & & \\ \triangle_0 & 2\big(\triangle_0+\triangle_1\big) & \triangle_1 & & & \\ \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & \\ & & \triangle_{n-2} & 2\big(\triangle_{n-2}+\triangle_{n-1}\big) & \triangle_{n-1} \\ & & & \triangle_{n-1} & 2\triangle_{n-1} \end{pmatrix} \qquad \text{and} \tag{14a}$$

$$A_1 \;=\; 6 \begin{pmatrix} \frac{1}{\triangle_0} & \frac{-1}{\triangle_0} & & & & \\ \frac{-1}{\triangle_0} & \frac{\triangle_0+\triangle_1}{\triangle_0\triangle_1} & \frac{-1}{\triangle_1} & & & \\ \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & \\ & & \frac{-1}{\triangle_{n-2}} & \frac{\triangle_{n-2}+\triangle_{n-1}}{\triangle_{n-2}\triangle_{n-1}} & \frac{-1}{\triangle_{n-1}} \\ & & & \frac{-1}{\triangle_{n-1}} & \frac{1}{\triangle_{n-1}} \end{pmatrix}. \tag{14b}$$

The right-hand side of equation (10) is given by

$$\begin{aligned} \mathbf{l} \;&=\; w_0\; \mathbf{l}_0 + w_1\; \mathbf{l}_1 \\ &=\; w_0 \begin{pmatrix} \int_{x_0}^{x_1} F(x) f_0(x)\; dx \\ \int_{x_0}^{x_2} F(x) f_1(x)\; dx \\ \vdots \\ \int_{x_{n-2}}^{x_n} F(x) f_{n-1}(x)\; dx \\ \int_{x_{n-1}}^{x_n} F(x) f_n(x)\; dx \end{pmatrix} \;+\; w_1 \begin{pmatrix} \int_{x_0}^{x_1} F'(x) f_0{}'(x)\; dx \\ \int_{x_0}^{x_2} F'(x) f_1{}'(x)\; dx \\ \vdots \\ \int_{x_{n-2}}^{x_n} F'(x) f_{n-1}{}'(x)\; dx \\ \int_{x_{n-1}}^{x_n} F'(x) f_n{}'(x)\; dx \end{pmatrix}. \end{aligned} \tag{15}$$

Remark 2.1. The function $F(x)$ is a smooth interpolant constructed from the given set of knots and function values. We use numerical integration, *Romberg integration*, to closely approximate the integral values in equation (15), see [Boehm & Prautzsch '93] and [Hamann et al. '99].

3. The bivariate case

In the bivariate case, we have to approximate a given function $F(x, y)$ by a piecewise linear spline function defined for a specific set of vertices (we use the terms *vertex* and *knot* interchangeably since in our case vertices always correspond directly to knots and vice versa) $\mathbf{v}_i = (x_i, y_i)^T$ with an associated triangulation $\mathcal{T}$[1]. The basis function $f_i = f_i(x, y)$ associated with vertex $\mathbf{v}_i$ is the linear spline function whose function value is one at $\mathbf{v}_i$ and zero at all other vertices, i.e., the basis function f_i that varies linearly between zero and one over all triangles defining $\mathbf{v}_i$'s platelet and is zero outside the platelet, see Fig. 2.

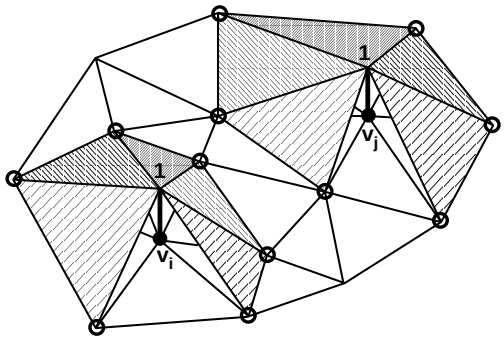

Fig. 2. Platelets of $\mathbf{v}_i$ and $\mathbf{v}_j$ and associated basis functions.

[1] We use the following notation: $\mathcal{T}$ denotes the set of all triangles (tetrahedra) in a triangulation; $\mathcal{T}_i$ denotes the set of triangles (tetrahedra) sharing the vertex $\mathbf{v}_i$ as a common vertex (= *platelet* of $\mathbf{v}_i$); and $\mathcal{T}_{i,j}$ denotes the set of triangles (tetrahedra) belonging to the platelets of both $\mathbf{v}_i$ and $\mathbf{v}_j$. (There must be an edge $\overline{\mathbf{v}_i\mathbf{v}_j}$ for the set $\mathcal{T}_{i,j}$ not to be empty.)

This is an outline of the algorithm we use to compute multiple best linear spline approximations in the bivariate case:

I. Input. The input is a set of vertices, $(x_i, y_i)^T$, $i = 0, \cdots, N$, with associated function values F_i, defining linear spline segments $s_j(x, y)$ over each triangle defined as

$$s_j(x, y) \;=\; \sum_{k=1}^{3} F_{j,k}\; f_{j,k}(x, y), \qquad j = 0, \cdots, N_{\mathcal{T}}, \tag{16}$$

where $N_{\mathcal{T}}+1$ is the number of triangles in the triangulation and $f_{j,k}(x, y)$ is the hat function associated with the k^{th} vertex of the j^{th} triangle. (This notation does not mean that we consider discontinuous spline functions. The function value/linear spline coefficient associated with a vertex in the triangulation is the same for all the triangles sharing this vertex.) We construct the piecewise linear interpolant $F(x, y)$ defined by the set of vertices, associated function values, and underlying triangulation.

II. First approximation. Compute the best linear spline approximation for a triangulation of the minimal set of vertices defining the convex hull of the set of all original data sites. Compute the best linear spline for this triangulation and compute error values that are triangle-specific. Should the error value of this approximation be too large (larger than a user-specified tolerance) continue with step III, otherwise terminate.

III. Subdivision. Inside a triangle T with maximal error, identify the original data site that is closest to the midpoint of the longest edge of T. (Arbitrarily choose an edge when multiple edges have the same maximal edge length.) Bisect the

triangles that share the chosen longest edge at the selected knot. This is illustrated in Fig. 3.

IV. Next approximation. Compute a new best linear spline approximation based on the new set of vertices and new triangulation and order the triangular spline segments according to the new triangle-specific error values. Terminate when the maximal triangle-specific error value is smaller than some user-specified tolerance, otherwise repeat steps III and IV. (If the tolerance is zero, the linear spline constructed in the last step will, in general, be based on the set of all original data sites—but may not be triangulated in the same way.)

V. Output. The output consists of subsets of the original data sites, defining the various levels of a hierarchy of best linear spline approximations. For each best linear spline approximation we store a different set of spline coefficients. Furthermore, we need to store the triangulations defining each level.

Regarding step III, one could alternatively identify the k triangles with largest error values and subdivide them simultaneously.

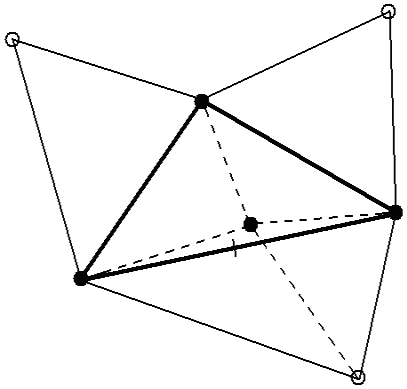

Fig. 3. Triangle bisection, using as split vertex an original data site closest to midpoint of longest edge (broken lines indicating new edges).

The functional that we minimize in the bivariate case is

$$E = \int_{\mathcal{T}} w_{0,0}\big(F(x,y) - f(x,y)\big)^2 + w_{1,0}\big(F_x(x,y) - f_x(x,y)\big)^2 + w_{0,1}\big(F_y(x,y) - f_y(x,y)\big)^2 \, dx \, dy, \tag{17}$$

where F_x denotes the partial derivative of F with respect to x, f_x denotes the partial derivative of f with respect to x, etc. The given function F must be approximated over the region whose boundary is the original vertex set's convex hull. The "weights" $w_{i,j}$ are non-negative and sum to one. The minimization problem reduces to minimizing

$$E = \int_{\mathcal{T}} \frac{1}{2} \begin{pmatrix} f(x,y) & f_x(x,y) & f_y(x,y) \end{pmatrix} \begin{pmatrix} 2w_{0,0} & 0 & 0 \\ 0 & 2w_{1,0} & 0 \\ 0 & 0 & 2w_{0,1} \end{pmatrix} \begin{pmatrix} f(x,y) \\ f_x(x,y) \\ f_y(x,y) \end{pmatrix} - \begin{pmatrix} f(x,y) & f_x(x,y) & f_y(x,y) \end{pmatrix} \begin{pmatrix} 2w_{0,0}F(x,y) \\ 2w_{1,0}F_x(x,y) \\ 2w_{0,1}F_y(x,y) \end{pmatrix} dx \, dy. \tag{18}$$

We define an approximation f to be the linear combination

$$f(x,y) = c_0 f_0(x,y) + c_1 f_1(x,y) + \cdots + c_n f_n(x,y) = \sum_{i=0}^{n} c_i \, f_i(x,y), \tag{19}$$

based on selected vertices $\mathbf{v}_i$ and an associated triangulation. Substituting equation (19) into equation (18) yields, formally, the same equation one obtains for the univariate case. In the bivariate case, the elements $a_{i,j}$ of the symmetric, positive definite matrix A—see equation (10)—are given by

$$\begin{aligned} a_{i,j} \;=\; & w_{0,0} \int_{\mathcal{T}} f_i(x,y) f_j(x,y)\; dx\; dy \;+\; w_{1,0} \int_{\mathcal{T}} f_{i\,x}(x,y) f_{j\,x}(x,y)\; dx\; dy \\ & +\; w_{0,1} \int_{\mathcal{T}} f_{i\,y}(x,y) f_{j\,y}(x,y)\; dx\; dy, \qquad i,j = 0,\cdots,n, \end{aligned} \tag{20}$$

and the elements l_i of the column vector $\mathbf{l}$—see equation (10)— are given by

$$\begin{aligned} l_i \;=\; & w_{0,0} \int_{\mathcal{T}} F(x,y) f_i(x,y)\; dx\; dy \;+\; w_{1,0} \int_{\mathcal{T}} F_x(x,y) f_{i\,x}(x,y)\; dx\; dy \\ & +\; w_{0,1} \int_{\mathcal{T}} F_y(x,y) f_{i\,y}(x,y)\; dx\; dy, \qquad i = 0,\cdots,n. \end{aligned} \tag{21}$$

The integral expressions required to compute the elements of the matrix A in the bivariate case are provided in the Appendix, Section A.2.

Remark 3.1. The function $F(x,y)$ is a smooth (scattered-data) interpolant constructed from the given set of vertices and function values. Again, we use Romberg integration (for triangulated domains) to approximate the integrals of the column vector $\mathbf{l}$.

4. The trivariate case

The trivariate case is a rather straightforward generalization of the bivariate case. We only describe the main differences between the bivariate and the trivariate case. This is an outline of our algorithm in the trivariate case:

I. Input. The input is a set of vertices, $(x_i, y_i, z_i)^T$, $i = 0, \cdots, N$, with associated function values F_i, defining linear spline segments $s_j(x, y, z)$ over each tetrahedron defined as

$$s_j(x, y, z) \;=\; \sum_{k=1}^{4} F_{j,k}\; f_{j,k}(x, y, z), \qquad j = 0, \cdots, N_{\mathcal{T}}, \tag{22}$$

where $N_{\mathcal{T}} + 1$ is the number of tetrahedra in the triangulation and $f_{j,k}(x, y, z)$ the hat function associated with the k^{th} vertex of the j^{th} tetrahedron. (This notation does not mean that we consider discontinuous spline functions. The function value/linear spline coefficient associated with a vertex in the triangulation is the same for all the tetrahedra sharing this vertex.) We first construct the piecewise linear interpolant $F(x, y, z)$ defined by the set of vertices, associated function values, and underlying triangulation.

II. First approximation. Compute the best linear spline approximation for a triangulation of the minimal set of vertices whose convex hull is the same as the convex hull of the set of all original data sites. Compute the best linear spline for this triangulation and compute error values that are tetrahedron-specific. Should the maximum of the tetrahedron-specific error values, all computed according to the generalization of equation (3), be too large (larger than a user-specified tolerance) continue with step III, otherwise terminate.

III. Subdivision. Inside a tetrahedron T with maximal error, identify the original data site that is closest to the midpoint of the longest edge of T. (Arbitrarily choose an edge when multiple edges have the same maximal edge length.) Bisect the tetrahedra that share the chosen longest edge. This is illustrated in Fig. 4.

IV. Next approximation. Compute a new best linear spline approximation based on the new set of vertices and new triangulation and order the tetrahedral spline segments according to the new tetrahedron-specific error values. Terminate when the maximal tetrahedron-specific error value is smaller than some user-specified tolerance, otherwise repeat steps III and IV. (If the tolerance is zero, the linear spline constructed in the last step will, in general, be based on the set of all original data sites—but may not be triangulated in the same way.)

V. Output. The output consists of subsets of the original data sites, defining the various levels of a hierarchy of best linear spline approximations. For each best linear spline approximation we store a different set of spline coefficients. Furthermore, we need to store the triangulations defining each level.

Alternatively, one could, in step III, identify the k tetrahedra with largest error values and subdivide them simultaneously.

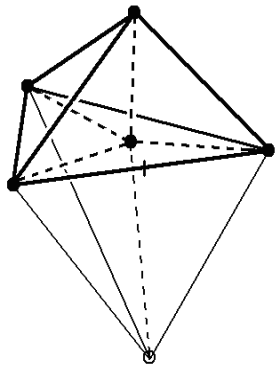

Fig. 4. Bisection of two tetrahedra, using as split vertex an original data site closest to midpoint of longest edge (broken lines indicating new edges).

The functional that we minimize in the trivariate case is given by

$$E \;=\; \int\limits_{\mathcal{T}} \sum_{\substack{i,j,k\geq 0 \\ i+j+k\leq 1}} \left(\frac{\partial^{i+j+k}}{\partial x^i \partial y^j \partial z^k} F(x,y,z) - \frac{\partial^{i+j+k}}{\partial x^i \partial y^j \partial z^k} f(x,y,z) \right)^2 dx\; dy\; dz. \qquad (23)$$

The given function F must be approximated over the region whose boundary is the original vertex set's convex hull. The "weights" $w_{i,j,k}$ are non-negative and sum to one. The minimization problem reduces to minimizing

$$\begin{aligned} E \;=\; \int\limits_{\mathcal{T}} \frac{1}{2} \big(f(x,y,z) \quad f_x(x,y,z) \quad f_y(x,y,z) \quad f_z(x,y,z) \big) \\ \begin{pmatrix} 2w_{0,0,0} & 0 & 0 & 0 \\ 0 & 2w_{1,0,0} & & 0 \\ 0 & 0 & 2w_{0,1,0} & 0 \\ 0 & 0 & 0 & 2w_{0,0,1} \quad 0 \end{pmatrix} \begin{pmatrix} f(x,y,z) \\ f_x(x,y,z) \\ f_y(x,y,z) \\ f_z(x,y,z) \end{pmatrix} \\ - \big(f(x,y,z) \quad f_x(x,y,z) \quad f_y(x,y,z) \quad f_z(x,y,z) \big) \\ \begin{pmatrix} 2w_{0,0,0}F(x,y,z) \\ 2w_{1,0,0}F_x(x,y,z) \\ 2w_{0,1,0}F_y(x,y,z) \\ 2w_{0,0,1}F_z(x,y,z) \end{pmatrix} dx\; dy\; dz. \end{aligned} \qquad (24)$$

We define an approximation f to be the linear combination

$$f(x,y,z) \;=\; \sum_{i=0}^{n} \; c_i \; f_i(x,y,z), \tag{25}$$

based on selected vertices $\mathbf{v}_i$ and an associated triangulation. Substituting equation (25) into equation (24) yields, formally, the same equation one obtains for the univariate and bivariate cases. In the trivariate case, the elements $a_{i,j}$ of the matrix A—see equation (10)—are given by

$$\begin{aligned} a_{i,j} \;=\;& w_{0,0,0} \int_{\mathcal{T}} f_i(x,y,z) f_j(x,y,z)\; dx\; dy\; dz \\ &+\; w_{1,0,0} \int_{\mathcal{T}} f_{i_x}(x,y,z) f_{j_x}(x,y,z)\; dx\; dy\; dz \\ &+\; w_{0,1,0} \int_{\mathcal{T}} f_{i_y}(x,y,z) f_{j_y}(x,y,z)\; dx\; dy\; dz \\ &+\; w_{0,0,1} \int_{\mathcal{T}} f_{i_z}(x,y,z) f_{j_z}(x,y,z)\; dx\; dy\; dz, \qquad i,j = 0,\cdots,n, \end{aligned} \tag{26}$$

and the elements l_i of the column vector $\mathbf{l}$—see equation (10)— are given by

$$\begin{aligned} l_i \;=\;& w_{0,0,0} \int_{\mathcal{T}} F(x,y,z) f_i(x,y,z)\; dx\; dy\; dz \\ &+\; w_{1,0,0} \int_{\mathcal{T}} F_x(x,y,z) f_{i_x}(x,y,z)\; dx\; dy\; dz \\ &+\; w_{0,1,0} \int_{\mathcal{T}} F_y(x,y,z) f_{i_y}(x,y,z)\; dx\; dy\; dz \\ &+\; w_{0,0,1} \int_{\mathcal{T}} F_z(x,y,z) f_{i_z}(x,y,z)\; dx\; dy\; dz, \qquad i = 0,\cdots,n. \end{aligned} \tag{27}$$

The integral expressions required to compute the elements of the matrix A in the trivariate case are provided in the Appendix, Section A.3.

Remark 4.1. We use Romberg integration (for tetrahedral domains) to compute the integrals of the column vector $\mathbf{l}$, see [Hamann et al. '99].

5. Examples

In this section, we present results for univariate, bivariate, and trivariate functions. We choose $[0,1]$, $[0,1] \times [0,1]$, and $[0,1] \times [0,1] \times [0,1]$ as domains for the univariate, bivariate, and trivariate test functions, respectively. We evaluate the test functions on a uniform, rectilinear grid, with resolution 100 in each dimension, defining the discrete input to our approximation method. We use the following initial triangulations that we refine iteratively by point insertion: (i) In the univariate case, we start with the interval $[0,1]$; (ii) in the bivariate case, we define the initial triangulation by splitting the unit square into the two triangles obtained by connecting $(0,0)^T$ and $(1,1)^T$; and (iii) in the trivariate case, we split the unit cube into the five tetrahedra, as shown in Fig. 5.

Tables 1 and 2 list, for various bivariate and trivariate test functions, the error values[2], the time it took with our particular implementation to generate the listed hierarchy levels, and the number of knots (in square brackets) for each level.

The univariate example was generated by sampling at uniformly spaced knot locations and then approximating the implied linear spline. The number of sample locations is equal to the number of knots used to generate the approximations. Concerning bivariate and

[2] The element-specific error values are based on the norm $||\ ||_{\mathrm{Sob}}$—as defined by equations (1)–(3)—and its generalizations to the bivariate and trivariate cases. The computing times listed were obtained on a Pentium III, 500MHz graphics workstation with 512MB of memory.

trivariate examples, our construction follows Sections 3 and 4.

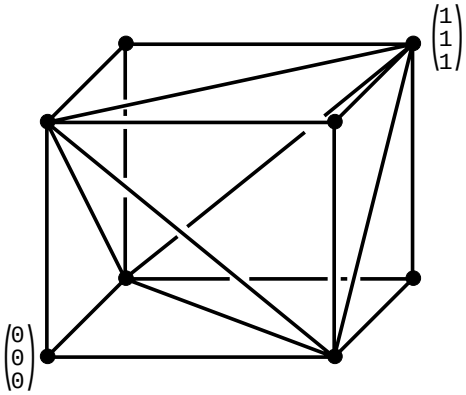

Fig. 5. Initial triangulation in trivariate case: unit cube split into five tetrahedra.

Fig. 6. Original test images: 2D (bivariate) checker board (151x151 pixels) and UC Davis logo (121x121 pixels).

Table 1. Computing times, errors, and numbers of knots (in square brackets) for bivariate functions. Error values are based on function value and first derivative.[3]

$10x\left(x-\frac{1}{4}\right)\left(x-\frac{3}{4}\right)y^2$ [$\approx$ 131s for 50 levels]	$\frac{1}{2}\sin\left(4\pi x^2\right)\cos\left(2\pi y^2\right)$ [$\approx$ 211s for 50 levels]	2D checker board [$\approx$ 188s for 50 levels]	UC Davis logo [$\approx$ 722s for 50 levels]
• Case 1: $w_{0,0}=1$, $w_{1,0}=0$, $w_{0,1}=0$			
0.38203[30]	187.681[30]	22.6008[261]	602.010[274]
0.15217[64]	078.699[60]	14.0151[517]	416.203[551]
0.09498[135]	010.299[126]	07.3432[1250]	248.964[1125]
0.08111[255]	003.762[238]	05.7065[2556]	138.907[2769]
0.07320[494]	001.999[546]	05.5198[5251]	109.316[5721]
0.06857[1165]	001.658[1291]	–	–
0.06647[2351]	001.547[2612]	–	–
0.06492[4004]	001.492[5327]	–	–
• Case 2: $w_{0,0}=\frac{3}{4}$, $w_{1,0}=\frac{1}{8}$, $w_{0,1}=\frac{1}{8}$			
0.46595[30]	148.612[29]	19.6899[309]	569.125[274]
0.28869[64]	053.518[63]	15.0050[517]	465.922[463]
0.25598[135]	010.835[134]	10.8139[1249]	310.202[1127]
0.24329[255]	007.461[254]	09.2595[2557]	243.782[2317]
0.23671[494]	006.189[491]	09.1743[5255]	208.509[5738]
0.22981[1160]	005.634[1372]	–	–
0.22863[2339]	005.531[2767]	–	–
0.22780[3982]	005.502[4714]	–	–
• Case 3: $w_{0,0}=\frac{1}{2}$, $w_{1,0}=\frac{1}{4}$, $w_{0,1}=\frac{1}{4}$			
0.62466[29]	97.524[29]	20.2099[290]	538.967[274]
0.45324[62]	42.130[63]	16.3124[579]	471.593[463]
0.41496[128]	13.253[133]	13.7895[1402]	373.310[1131]
0.39901[239]	10.954[252]	12.5213[2874]	327.849[2327]
0.39481[462]	10.045[486]	12.5026[5875]	301.407[5763]
0.39007[1085]	09.577[1356]	–	–
0.38989[2184]	09.497[2733]	–	–
0.38983[3717]	09.491[4656]	–	–

[3] We define the error as $E=\int_a^b w_{0,0}\big(F(x,y)-f(x,y)\big)^2+w_{1,0}\big(F_x(x,y)-f_x(x,y)\big)^2+w_{0,1}\big(F_y(x,y)-f_y(x,y)\big)^2\,dxdy$, where $F(x,y)$ is the function to be approximated and $f(x,y)$ is the approximant. A dash symbol indicates that a final error condition is satisfied.

Table 2. Computing times, errors, and numbers of knots (in square brackets) for trivariate functions. Error values are based on function values and first derivative.[4]

$10x\big(x-\frac{1}{4}\big)\big(x-\frac{3}{4}\big)$ $y^2\sqrt{z}$ [≈ 20m for 25 levels]	$\sin\big(10\pi x^2\big)\cos\big(5\pi y^2\big)$ $\sin^2\big(2\pi z\big)$ [≈ 15m for 23 levels]	3D checker board [≈ 13m for 25 levels]	Flame [≈ 2.2h for 27 levels]
• Case 1: $w_{0,0,0}=1$, $w_{1,0,0}=0$, $w_{0,1,0}=0$, $w_{0,0,1}=0$			
216.16[56]	21199.0[60]	14139.7[53]	5370.9[68]
027.70[224]	17846.9[294]	07940.8[236]	2728.2[271]
013.84[599]	14643.1[622]	05145.3[700]	1829.5[883]
010.19[1773]	10303.6[1369]	02823.7[2284]	1275.4[3068]
–	–	–	0941.6[10198]
• Case 2: $w_{0,0,0}=\frac{3}{4}$, $w_{1,0,0}=\frac{1}{12}$, $w_{0,1,0}=\frac{1}{12}$, $w_{0,0,1}=\frac{1}{12}$			
153.98[56]	15884.8[59]	10015.5[53]	3556.0[72]
024.71[235]	14296.9[281]	06309.0[232]	2401.9[206]
012.98[635]	11543.2[584]	04057.1[681]	1666.6[667]
010.31[1892]	07957.6[1288]	02833.1[1495]	1119.9[3523]
–	–	–	0945.8[11293]
• Case 3: $w_{0,0,0}=\frac{1}{2}$, $w_{1,0,0}=\frac{1}{6}$, $w_{0,1,0}=\frac{1}{6}$, $w_{0,0,1}=\frac{1}{6}$			
117.45[55]	10677.1[59]	7833.5[48]	2718.6[72]
020.66[230]	09933.3[281]	4765.3[203]	1876.3[207]
012.98[610]	07780.0[582]	2714.5[876]	1433.7[678]
011.41[1249]	–	1694.4[2916]	1090.5[3605]

[4] We define the error as $E=\int_a^b w_{0,0,0}\big(F(x,y,z)-f(x,y,z)\big)^2+w_{1,0,0}\big(F_x(x,y,z)-f_x(x,y,z)\big)^2+w_{0,1,0}\big(F_y(x,y,z)-f_y(x,y,z)\big)^2+w_{0,0,1}\big(F_z(x,y,z)-f_z(x,y,z)\big)^2\,dxdydz$, where $F(x,y,z)$ is the function to be approximated and $f(x,y,z)$ is the approximant.

Figs. 7–13 show approximations for analytically defined univariate, bivariate, and trivariate functions; digital photographs; and sampled scalar fields over trivariate domains (flame). Concerning the generation of the triangulation levels in the bivariate and trivariate cases, we have inserted multiple knots simultaneously by identifying the approximately 10% elements with largest error values and subdividing them in one step.

Figs. 7, 8, and 11 demonstrate how varying weights affect the quality of the approximation for the univariate, bivariate, and trivariate case, respectively. In each of these figures, it is possible to see the over- and under-shoots in areas where the first-derivative weights are not sufficiently large enough to reduce them. Figs. 7 and 8 show "peaks" and "valleys" where the over- and under-shoots occur (near the high-gradient areas). Light and dark regions near the high-gradient areas in Fig. 11 show the over- and under-shoots. We have rendered the trivariate checker board function by "cutting" through the underlying tetrahedral mesh.

Fig. 9 shows how varying weights affect the quality of the approximation for Franke's function, commonly used for testing scattered-data approximation schemes. This function is defined as

$$\begin{aligned} f(x,y) \;=\; & 0.75e^{-0.25(9x-2)^2 \,-\, 0.25(9y-2)^2} \;+\; 0.75e^{-(9x-2)^2/49 \,-\, (9y-2)^2/10} \\ & +\; 0.5e^{-0.25(9x-7)^2 \,-\, 0.25(9y-3)^2} \;-\; 0.2e^{-(9x-4)^2 \,-\, (9y-7)^2}. \end{aligned} \tag{28}$$

In this example, a Delaunay triangulation was constructed for the function from 2000 random sample points. The triangulation (i.e., the linear spline) was digitized to a 1000x1000 uniform grid before being processed. The upper-left image shows a rendering of the original data. The pair of perpendicular line segments in the approximations denote the origin.

Computation times ranged from four to 14 seconds, for 28 levels.

Fig. 12 demonstrates how varying weights affect the quality of the approximation for the flame data set ($40 \times 40 \times 20 = 32000$ normalized function values on a rectilinear grid). Element-specific error values computed with respect to the original discrete data set. Computation times ranged from 39 minutes to 51 minutes, for 27 levels. An image of a cutting plane through the original data set is centered above the group of four varying weight images in the figure.

Fig. 13 demonstrates how varying weights affect the quality of approximations of a Golden Gate Bridge photograph ($889 \times 557 = 495173$ RGB pixels distributed uniformly in x- and y-direction). Each color component (red, green, and blue) was treated independently from the other two components. Error values were determined by averaging the three color component specific error values, resulting in a single error value. Computation times ranged from 27 minutes to 42 minutes, for 55 levels.

Remark 5.1. We note that the described algorithm is computationally quite expensive, both with respect to time and space requirements. This is primarily due to the fact that one has to perform numerical integration and solve sparse linear equation systems to generate the approximation hierarchy. We have implemented our algorithm on a Pentium III, 500MHz graphics workstation with 512MB of memory. The constructions of the hierarchies for the most complicated trivariate examples, which consist of several thousand knots at the highest levels of resolution, require several minutes of computation. We believe that this is acceptable: interactive visualization of very large data sets is only possible if a pre-computed hierarchy is available at the time of rendering and data exploration. Fig. 14

shows running times for one bivariate and one trivariate function.

Examining the error and knot relationship shown in Tables 1 and 2 reveals various characteristics of the method. Initially, as expected, there is a large approximation error when using a relatively small number of knots (i.e., less than 1% of the number of original sites). This large error is quickly reduced as the number of knots approaches 1% of the number of original sites. Tables 1, 2, and the run time data shown in Fig. 14 lead to the following observation: when the number of knots is less than 1% of the original sites, the amount of computation time needed to reduce the error from E to $E/2$ is roughly twice the amount of time that was necessary to reduce the initial mesh error E_0 to E. However, as the number of knots increases beyond 1%, the error reduction rate decreases significantly. Considering the bivariate function shown in Fig. 14, we observed the following behavior: when the number of knots being used is greater than 1%, the amount of time necessary to reduce an error E to $0.94E$ is roughly three times the amount of time that was necessary to reduce the initial mesh error E_0 to E. Considering the trivariate function shown in Fig. 14, we observed the following behavior: when the number of knots being used is greater than 1%, the amount of time necessary to reduce an error E to $0.75E$ is roughly three times the amount of time that was necessary to reduce the initial mesh error E_0 to E.

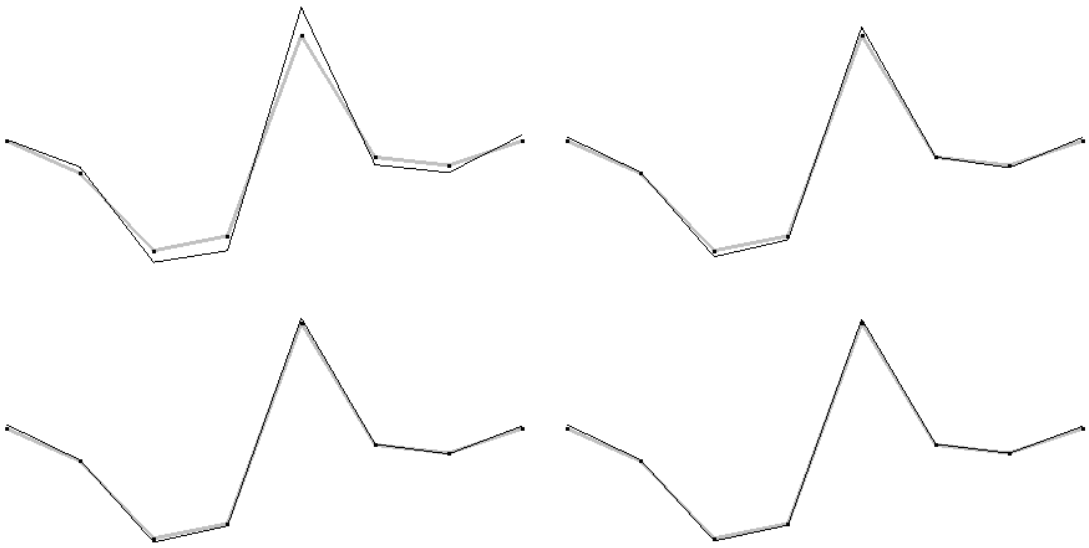

Fig. 7. Function $\sin(4\pi x^2)$ using eight uniformly spaced knots with varying weights (w_0, w_1): (1.0, 0.0), (0.99, 0.01), (0.98, 0.02), and (0.97, 0.03) — from upper-left to lower-right corner. The lighter polygon is the original function; the darker polygon is the approximation.

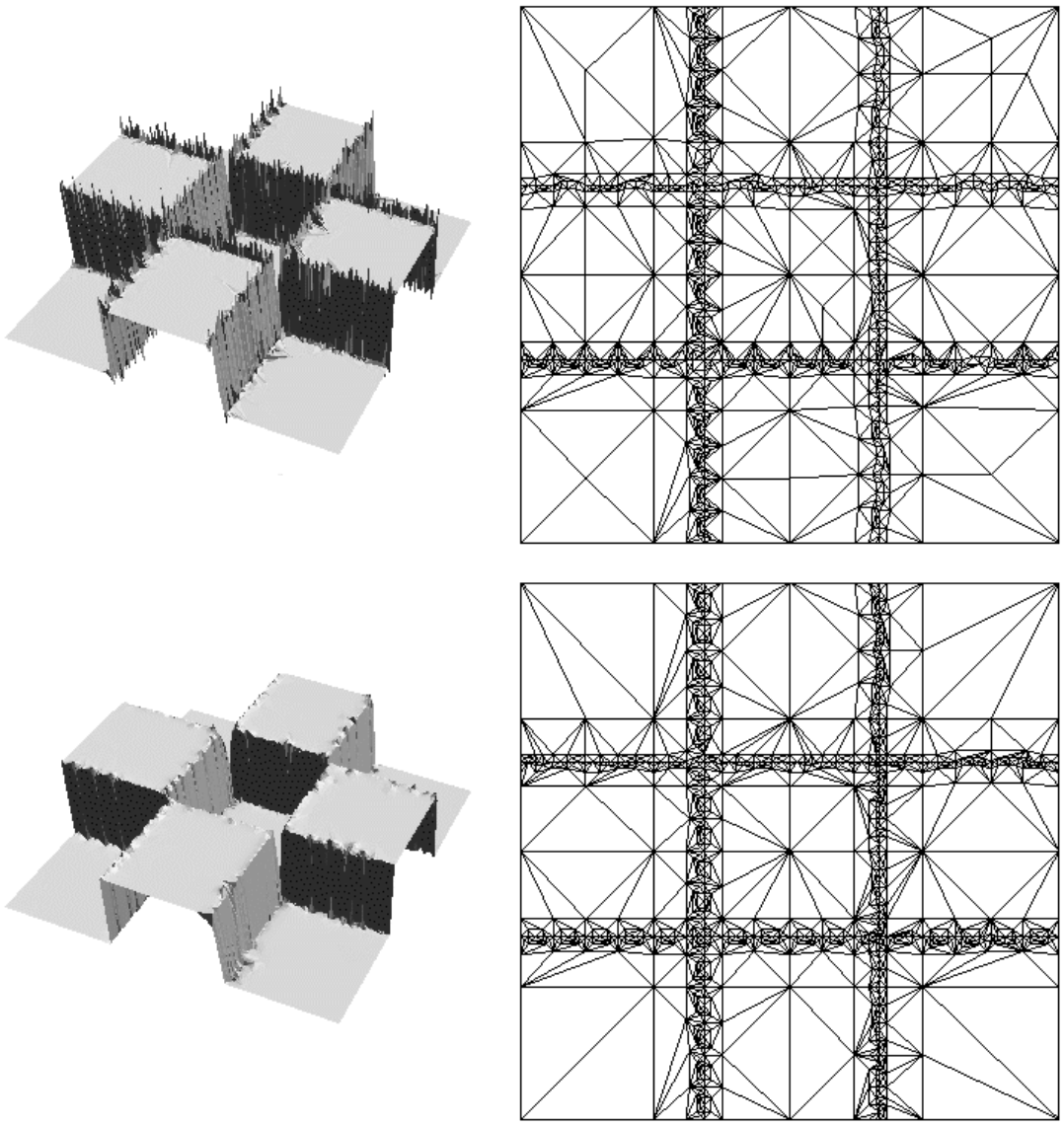

Fig. 8. 2D (bivariate) checker board function using between 5000 and 6000 knots with varying weights ($w_{0,0}$, $w_{1,0}$, $w_{0,1}$): (1, 0, 0) and $\left(\frac{1}{4}, \frac{3}{8}, \frac{3}{8}\right)$ — from top to bottom.

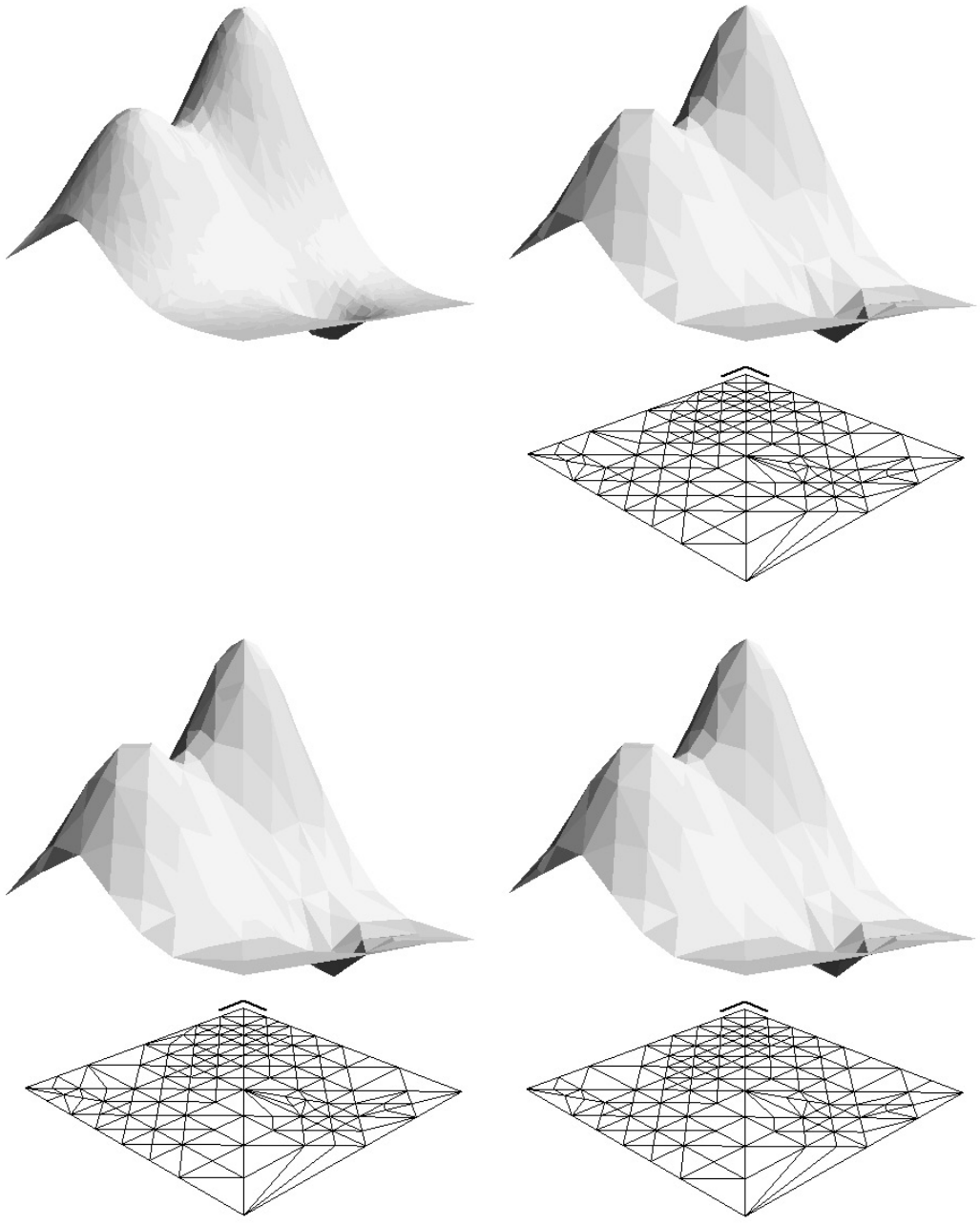

Fig. 9. Franke's function using varying weights ($w_{0,0}$, $w_{1,0}$, $w_{0,1}$) for three approximations: (1, 0, 0), $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right)$, and $\left(\frac{1}{4}, \frac{3}{8}, \frac{3}{8}\right)$ — from upper-right to lower-right corner; number of vertices: 132, 132, and 129; errors: 13.68, 13.43, and 13.50. The upper-left image shows the original data (2000 vertices).

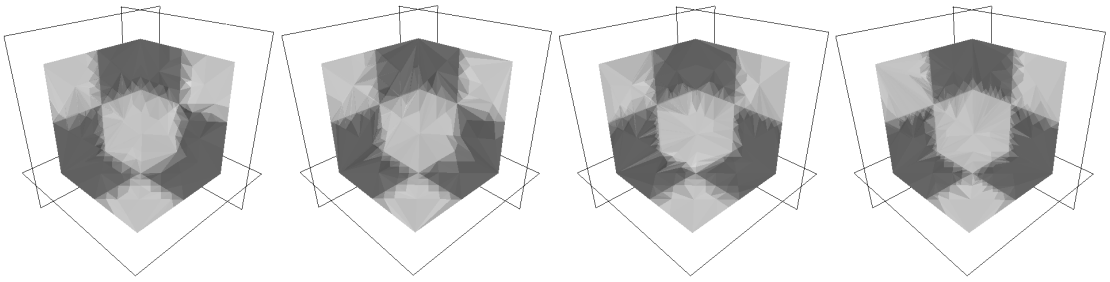

Fig. 10. Three mutually perpendicular cutting planes through 3D (trivariate) checker board function using varying weights ($w_{0,0,0}$, $w_{1,0,0}$, $w_{0,1,0}$, $w_{0,0,1}$): (1, 0, 0, 0), $\left(\frac{3}{4}, \frac{1}{12}, \frac{1}{12}, \frac{1}{12}\right)$, $\left(\frac{1}{2}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right)$, and $\left(\frac{1}{4}, \frac{3}{12}, \frac{3}{12}, \frac{3}{12}\right)$ — from left to right.

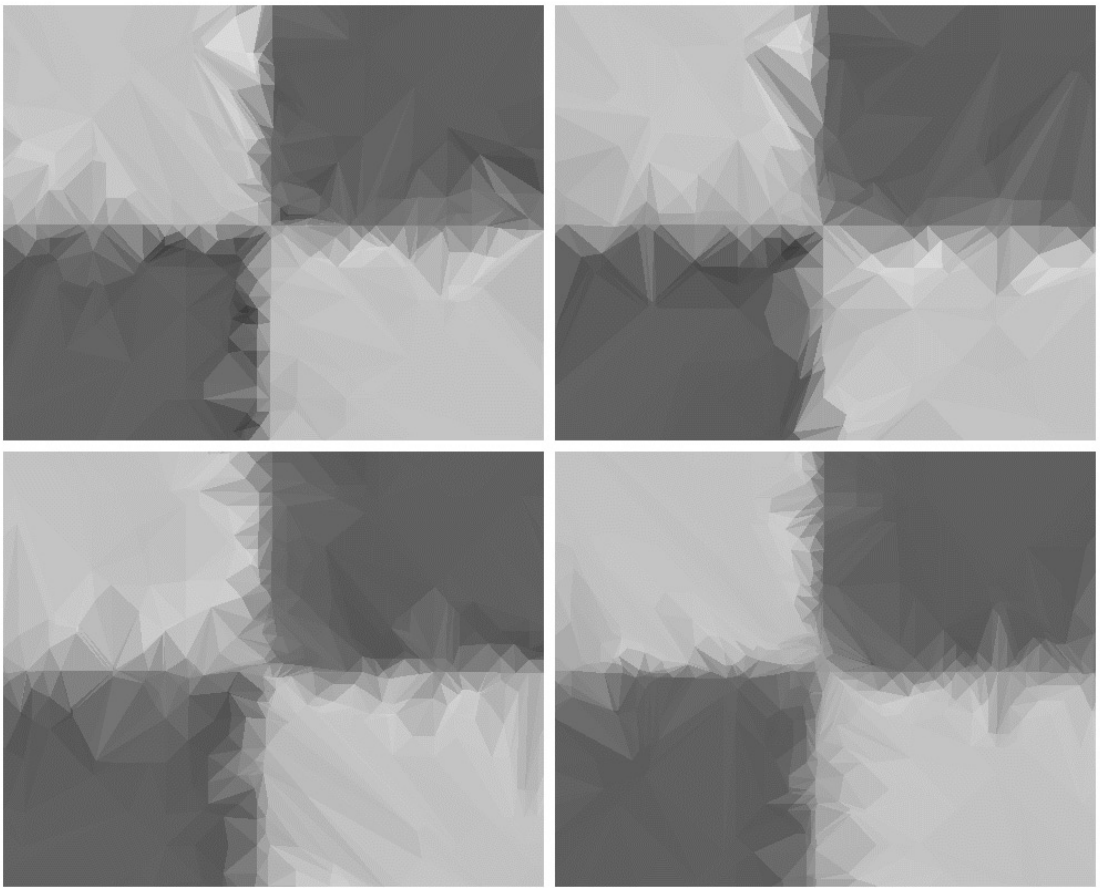

Fig. 11. Magnifications of four approximations for function shown in Fig. 10.

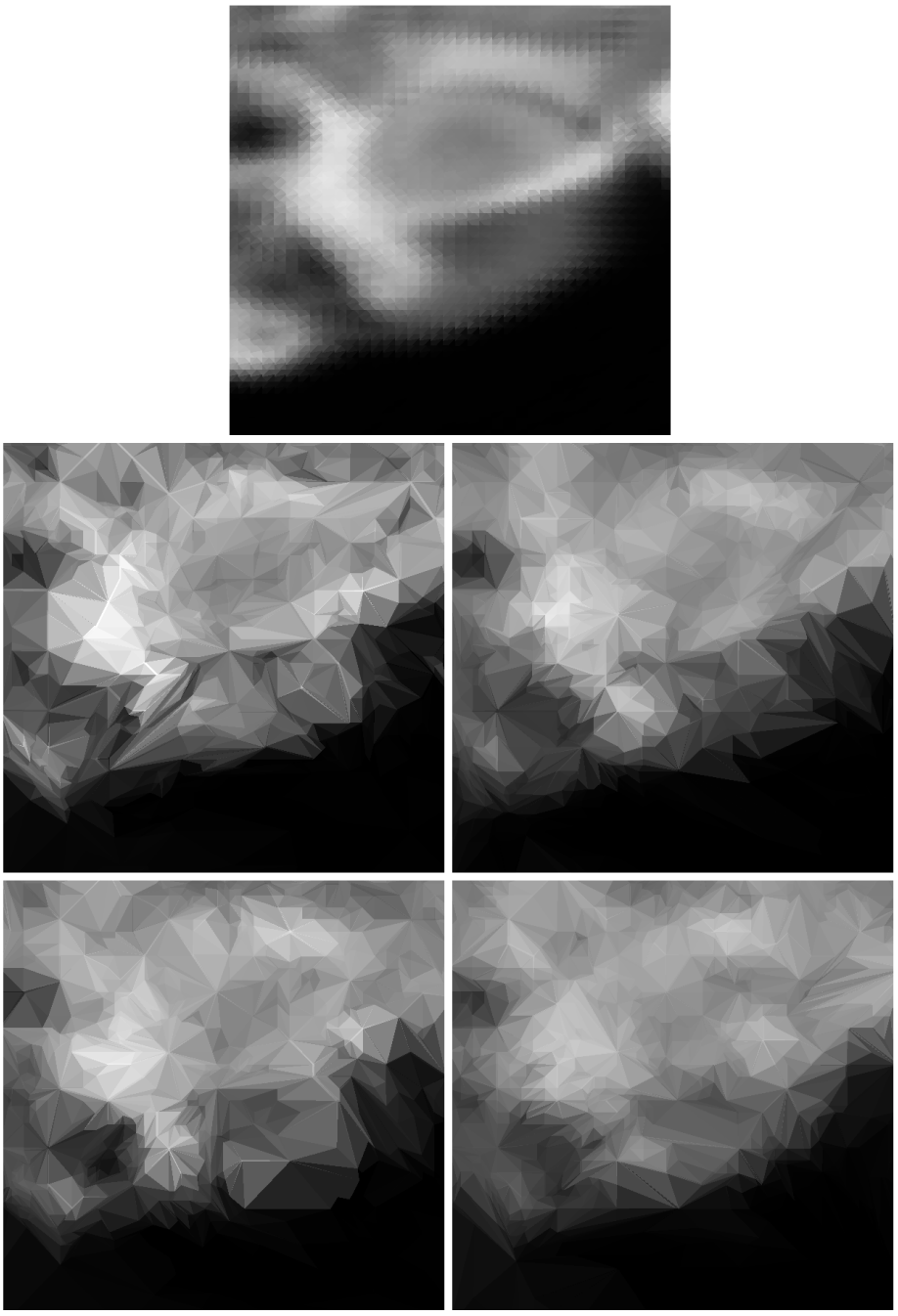

Fig. 12. One cutting plane through flame data set using varying weights ($w_{0,0,0}$, $w_{1,0,0}$, $w_{0,1,0}$, $w_{0,0,1}$) for four approximations: (1, 0, 0, 0), $\left(\frac{3}{4}, \frac{1}{12}, \frac{1}{12}, \frac{1}{12}\right)$, $\left(\frac{1}{2}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right)$, and $\left(\frac{1}{4}, \frac{3}{12}, \frac{3}{12}, \frac{3}{12}\right)$ — from upper-left to lower-right corner; number of vertices: 3127, 2859, 3432, and 3529; errors: 217.48, 233.77, 215.51, and 198.02. The top image shows the original data (32000 vertices).

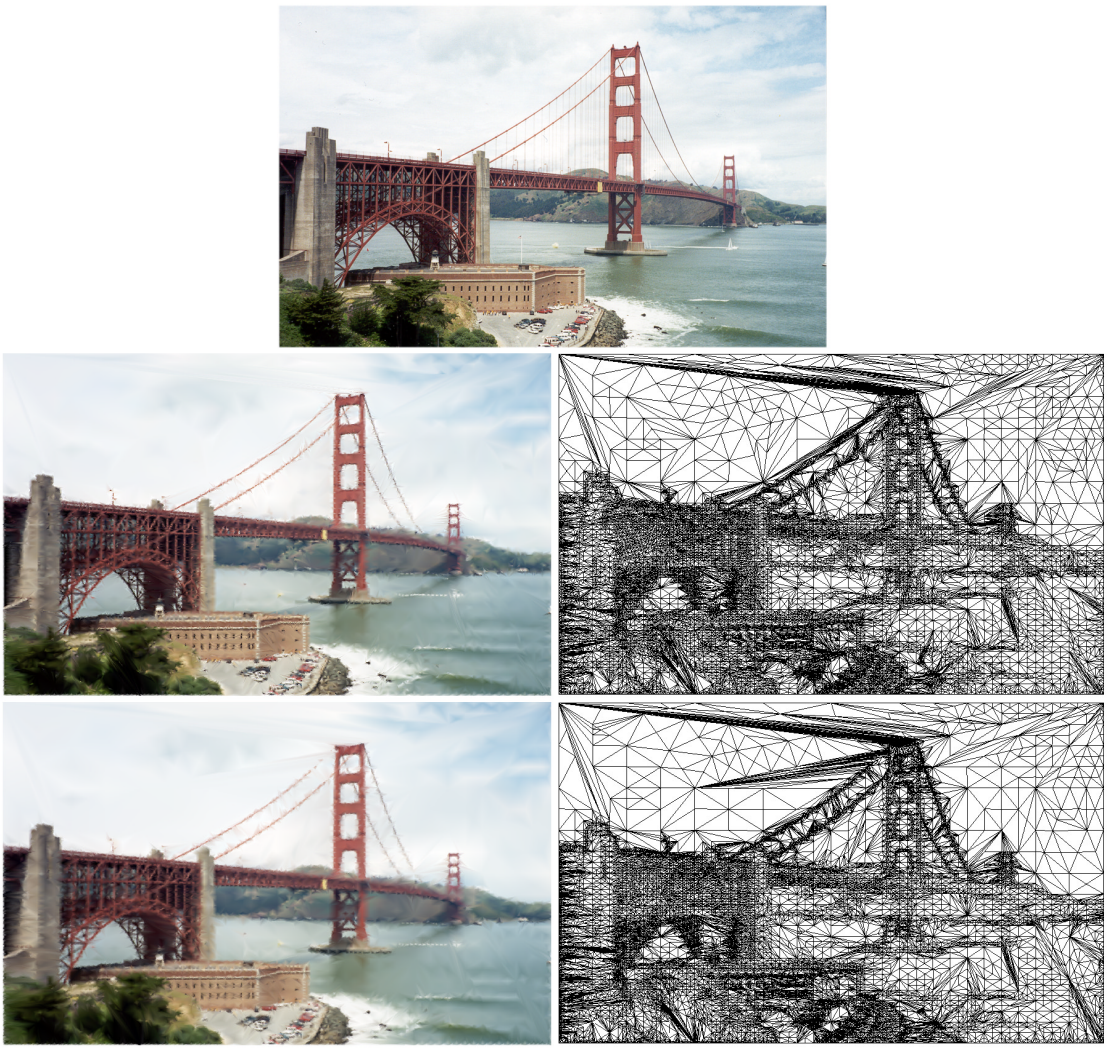

Fig. 13. Golden Gate Bridge approximations using varying weights $(w_{0,0},\, w_{1,0},\, w_{0,1})$: $(1,\, 0,\, 0)$ and $\left(\frac{1}{4},\, \frac{3}{8},\, \frac{3}{8}\right)$ — from top to bottom; number of vertices: 14776 and 14548; errors: 1274.38 and 1081.66. The top image shows the original data (495173 vertices).

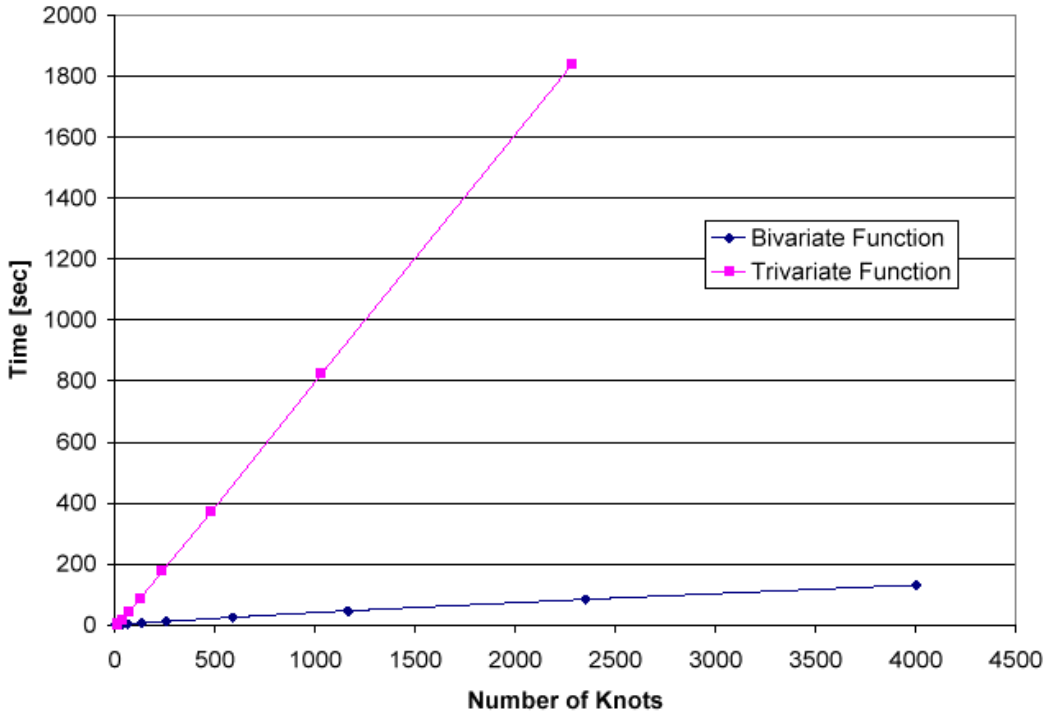

Fig. 14. Running times for bivariate function $10x\left(x-\frac{1}{4}\right)\left(x-\frac{3}{4}\right)y^2$ and trivariate function $10x\left(x-\frac{1}{4}\right)\left(x-\frac{3}{4}\right)y^2\sqrt{z}$ using weights $(w_{0,0},\, w_{1,0},\, w_{0,1}) = (1,\, 0,\, 0)$ and $(w_{0,0,0},\, w_{1,0,0},\, w_{0,1,0},\, w_{0,0,1}) = (1,\, 0,\, 0,\, 0)$, respectively. (These running times are representative for all functions tested. Times were obtained on a Pentium III, 500MHz graphics workstation with 512MB of memory.)

6. Conclusions and future work

We have presented a method for the generation of multiple best linear spline approximations based on a triangulation-based finite element approach. The approach is simple, and the bisection paradigm applies to simplicial decompositions of space in arbitrary dimensions. One should view the construction of a hierarchy of approximations as a pre-processing step for data visualization and analysis. From this point of view, the construction does not necessarily have to be an extremely efficient operation; an algorithm computing a data hierarchy serves its primary purpose as long as the resulting data hierarchy is compact, requires a simple data format, and can be used by standard visualization technology. We have presented a method that satisfies these conditions.

Regarding Fig. 9, while each of the three approximations accurately approximates Franke's function with relatively few knots there is little difference when using varying non-zero weights for first-derivative information in the approximation. This is because the function is considered "smooth" by our method since there are no discontinuities. When approximating highly discontinuous data, our method captures discontinuities better when considering first-derivative information compared to not using the information. The result is a concentration of simplices near discontinuities and correction of over- and under-shoots. For infinitely differentiable functions, such as Franke's function, and other "smooth" data our method performs well, though, first-derivative information only slightly improves approximation quality.

Considering the extremely large data set sizes produced by state-of-the-art numerical simulations and imaging devices, we plan to improve the efficiency of our approach and

generalize the approach to allow more general mesh refinement, including the consideration of knots that are not part of a original discrete data set. We have not yet studied the theoretical aspects concerning the approximation properties of our scheme, e.g., the rate of decay of the global error measure for certain classes of functions. This interesting and challenging work remains to be done.

It is possible to reduce storage requirements by storing the insertion order of sites rather than an explicit representation. In this case, the overall site storage requirements for a hierarchy are equal to those of the highest level of resolution. Connectivity information can be stored as well to eliminate the need for re-calculating the connectivity during site insertion.

It is possible to store the coefficients of each level, resulting in a storage that is nearly twice the size of the highest level of resolution. An alternative is to calculate differences from one level to another for the coefficients. These differences would require less precision to store and could be quantized to significantly reduce storage.

In general, it is not possible to store an entire hierarchy in main computer memory, and it is therefore necessary to store the hierarchy in a file format that supports easy and efficient access. We believe that an optimal format to store a data hierarchy depends on the particular visualization and analysis algorithms that are utilized. An optimal file format is application-dependent, and one should design it on a case-to-case basis.

Much of the work referenced in the Introduction considers a local minimization problem. Better approximations are achieved when using a global minimization approach, as used in our method. The global nature of our method seemingly increases the amount

of computation required; but, it is possible to treat the generated linear systems highly efficiently. The addition of first-derivative information, a more general concept for data-dependent triangulation and approximation, improves approximation quality in a more adaptive fashion. The user can vary the weighting of first-derivative information for the problem at hand to achieve better results.

The most time-sensitive portion of the method is integrating over the simplices. Integration occurs in both the computation of the right-hand-side of the linear system and in the error computation. Each of these steps accounts for nearly 50% of the total running time. Since integration over the simplices can be performed independently, these two steps can easily be parallelized. When solving the linear system, the second most time critical step, one can take advantage of sparse-system solvers. During each iteration, our system only changes "locally," which allows us to use so-called *updating schemes*. [Duff et al. '86] describe a partitioning scheme for system updating where new matrix entries are appended to the previous matrices. A topic of future research might be to extend such an updating scheme to our setting.

Comparison to existing models should include close inspection to the handling of both smooth and discontinuous regions. Our method handles both types of regions (independently and concurrently). Because of the first-derivative contribution our method handles discontinuities well. It might be possible to extend and improve other types of approximation methods by incorporating derivative information.

The selection of "good" weights for the functionals to be minimized depends generally on the function to be approximated. It is a topic of future research to determine whether

or not it is possible (and how) to automatically determine the "optimal" weights for a particular input function.

7. Acknowledgements

This work was supported by the National Science Foundation under contracts ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, and through the National Partnership for Advanced Computational Infrastructure (NPACI); the Office of Naval Research under contract N00014-97-1-0222; the Army Research Office under contract ARO 36598-MA-RIP; the NASA Ames Research Center through an NRA award under contract NAG2-1216; the Lawrence Livermore National Laboratory under ASCI ASAP Level-2 Memorandum Agreement B347878 and under Memorandum Agreement B503159; the Lawrence Berkeley National Laboratory; the Los Alamos National Laboratory; and the North Atlantic Treaty Organization (NATO) under contract CRG.971628. We also acknowledge the support of ALSTOM Schilling Robotics and SGI and thank the members of the Visualization and Graphics Research Group at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis.

APPENDIX

A.1. Formulae for the univariate case

The integral values required to compute the matrix elements $a_{i,j}$ in the univariate case are

$$\int_{x_0}^{x_1} \big(f_0(x)\big)^2 \, dx \;=\; \frac{1}{3}\,\triangle_0, \tag{29a}$$

$$\int_{x_{n-1}}^{x_n} \big(f_n(x)\big)^2 \, dx \;=\; \frac{1}{3}\,\triangle_{n-1}, \tag{29b}$$

$$\int_{x_{i-1}}^{x_{i+1}} \big(f_i(x)\big)^2 \, dx \;=\; \frac{1}{3}\,\big(\triangle_{i-1}+\triangle_i\big), \qquad i=1,\cdots,n-1, \qquad \text{and} \tag{29c}$$

$$\int_{x_i}^{x_{i+1}} f_i(x) f_{i+1}(x) \, dx \;=\; \frac{1}{6}\,\triangle_i, \qquad i=0,\cdots,n-1. \tag{29d}$$

The terms involving the first derivatives are

$$\int_{x_0}^{x_1} \big(f_0{}'(x)\big)^2 \, dx \;=\; \frac{1}{\triangle_0}, \tag{30a}$$

$$\int_{x_{n-1}}^{x_n} \big(f_n{}'(x)\big)^2 \, dx \;=\; \frac{1}{\triangle_{n-1}}, \tag{30b}$$

$$\int_{x_{i-1}}^{x_{i+1}} \big(f_i{}'(x)\big)^2 \, dx \;=\; \frac{1}{\triangle_{i-1}}+\frac{1}{\triangle_i}, \qquad i=1,\cdots,n-1, \qquad \text{and} \tag{30c}$$

$$\int_{x_i}^{x_{i+1}} f_i{}'(x) f_{i+1}{}'(x) \, dx \;=\; -\frac{1}{\triangle_i}, \qquad i=0,\cdots,n-1. \tag{30d}$$

A.2. Formulae for the bivariate case

The following integral expressions are required to compute the individual matrix elements $a_{i,j}$ in the bivariate case:

$$\int_{\mathcal{T}_i} \big(f_i(x,y)\big)^2 \, dx\, dy \;=\; \frac{1}{12}\,\sum_{j=0}^{n_i} |J_j|, \tag{31}$$

where n_i+1 is the number of platelet triangles in the triangle set $\mathcal{T}_i$ (= platelet) associated with vertex $\mathbf{v}_i$, and J_j is the Jacobian associated with the j^{th} platelet triangle. The Jacobian J of a triangle with vertices $(x_0,y_0)^T$, $(x_1,y_1)^T$ and $(x_2,y_2)^T$—which should be ordered counterclockwise to ensure consistency—is given by

$$J \;=\; \det\begin{pmatrix} x_1-x_0 & x_2-x_0 \\ y_1-y_0 & y_2-y_0 \end{pmatrix}. \tag{32}$$

(The platelet of triangles associated with vertex $\mathbf{v}_i$ is the set of triangles $\mathcal{T}_i = \{T_j \mid j = 0, \cdots, n_i\}$, and T_j denotes an individual triangle.) Two basis functions $f_i(x, y)$ and $f_j(x, y)$ whose associated vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ are connected by an edge in the triangulation imply the non-zero integral value

$$\int\limits_{\mathcal{T}_{i,j}} f_i(x,y) f_j(x,y)\ dx\ dy \ = \ \frac{1}{24} \sum_{k=0}^{n_{i,j}} |J_k|, \tag{33}$$

where the set $\mathcal{T}_{i,j} = \{T_k \mid k = 0, \cdots, n_{i,j}\}$ is the set of triangles belonging to both $\mathbf{v}_i$'s and $\mathbf{v}_j$'s platelets. (There can be at most two triangles belonging to both $\mathbf{v}_i$'s and $\mathbf{v}_j$'s platelets in the bivariate case.)

The linear polynomial interpolating the values one, zero, and zero at the vertices $(x_0, y_0)^T$, $(x_1, y_1)^T$, and $(x_2, y_2)^T$, respectively, has the partial derivatives

$$f_x(x,y) \ = \ -\frac{1}{J} \ \det \begin{pmatrix} 1 & y_1 \\ 1 & y_2 \end{pmatrix} \ = \ \frac{y_1 - y_2}{J} \quad \text{and} \tag{34a}$$

$$f_y(x,y) \ = \ -\frac{1}{J} \ \det \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \end{pmatrix} \ = \ \frac{x_2 - x_1}{J}. \tag{34b}$$

Integrals involving these partial derivatives are:

$$\int\limits_{\mathcal{T}_i} \big(f_{i\,x}(x,y)\big)^2\ dx\ dy \ = \ \frac{1}{2} \sum_{j=0}^{n_i} \frac{1}{|J_j|} \det{}^2 \begin{pmatrix} 1 & y_{j,1} \\ 1 & y_{j,2} \end{pmatrix} \quad \text{and} \tag{35a}$$

$$\int\limits_{\mathcal{T}_i} \big(f_{i\,y}(x,y)\big)^2\ dx\ dy \ = \ \frac{1}{2} \sum_{j=0}^{n_i} \frac{1}{|J_j|} \det{}^2 \begin{pmatrix} x_{j,1} & 1 \\ x_{j,2} & 1 \end{pmatrix}, \tag{35b}$$

where $(x_{j,0}, y_{j,0})^T$, $(x_{j,1}, y_{j,1})^T$, and $(x_{j,2}, y_{j,2})^T$ are the (preferably) counterclockwise-ordered vertices of the $n_i + 1$ platelet triangles associated with $\mathbf{v}_i$, see Fig. 15.

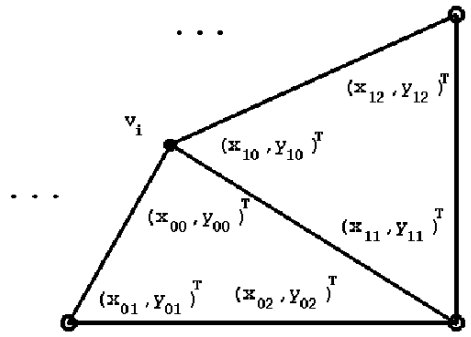

Fig. 15. Indexing scheme for platelet vertices in bivariate case (neighboring triangles oriented counterclockwise).

Other required values are

$$\int_{\mathcal{T}_{i,j}} f_{ix}(x,y) f_{jx}(x,y)\, dx\, dy \;=\; \frac{1}{2} \sum_{k=0}^{n_{i,j}} \frac{1}{|J_k|} \det \begin{pmatrix} 1 & y_{k,1} \\ 1 & y_{k,2} \end{pmatrix} \det \begin{pmatrix} 1 & y_{k,0} \\ 1 & y_{k,1} \end{pmatrix} \text{and} \quad (36a)$$

$$\int_{\mathcal{T}_{i,j}} f_{iy}(x,y) f_{jy}(x,y)\, dx\, dy \;=\; \frac{1}{2} \sum_{k=0}^{n_{i,j}} \frac{1}{|J_k|} \det \begin{pmatrix} x_{k,1} & 1 \\ x_{k,2} & 1 \end{pmatrix} \det \begin{pmatrix} x_{k,0} & 1 \\ x_{k,1} & 1 \end{pmatrix}, \quad (36b)$$

where $(x_{k,0}, y_{k,0})^T$, $(x_{k,1}, y_{k,1})^T$, and $(x_{k,2}, y_{k,2})^T$ are the vertices of a triangle being a common triangle of $\mathbf{v}_i$'s and $\mathbf{v}_j$'s platelets. The number of common triangles, which is at most two, is given by $n_{i,j} + 1$. Fig. 16 illustrates our indexing scheme.

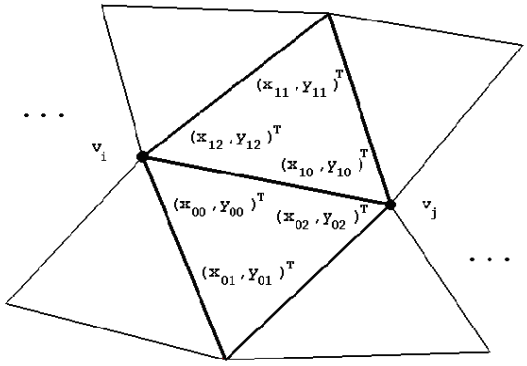

Fig. 16. Indexing scheme for intersecting platelets (bivariate case).

A.3. Formulae for the trivariate case

The following integral expressions are required to compute the individual matrix elements $a_{i,j}$ in the trivariate case:

$$\int\limits_{\mathcal{T}_i} \big(f_i(x,y,z)\big)^2 \; dx \; dy \; dz \;=\; \frac{1}{60} \sum_{j=0}^{n_i} |J_j|, \tag{37}$$

where n_i+1 is the number of platelet tetrahedra in the tetrahedra set $\mathcal{T}_i$ (= platelet) associated with vertex $\mathbf{v}_i$, and J_j is the Jacobian associated with the j^{th} platelet tetrahedron. The Jacobian J of a tetrahedron with vertices $(x_0,y_0,z_0)^T$, $(x_1,y_1,z_1)^T$, $(x_2,y_2,z_2)^T$, and $(x_3,y_3,z_3)^T$ is given by

$$J \;=\; \det \begin{pmatrix} x_1-x_0 & x_2-x_0 & x_3-x_0 \\ y_1-y_0 & y_2-y_0 & y_3-y_0 \\ z_1-z_0 & z_2-z_0 & z_3-z_0 \end{pmatrix}. \tag{38}$$

(The platelet of tetrahedra associated with vertex $\mathbf{v}_i$ is the set of tetrahedra $\mathcal{T}_i = \{T_j \mid j = 0,\cdots,n_i\}$, and T_j denotes an individual tetrahedron.) Two basis functions $f_i(x,y,z)$ and $f_j(x,y,z)$ whose associated vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ are connected by an edge in the triangulation imply the non-zero integral value

$$\int\limits_{\mathcal{T}_{i,j}} f_i(x,y,z) f_j(x,y,z) \; dx \; dy \; dz \;=\; \frac{1}{120} \sum_{k=0}^{n_{i,j}} |J_k|, \tag{39}$$

where the set $\mathcal{T}_{i,j} = \{T_k \mid k = 0,\cdots,n_{i,j}\}$ is the set of tetrahedra belonging to both $\mathbf{v}_i$'s and $\mathbf{v}_j$'s platelets.

The linear polynomial interpolating the values one, zero, zero, and zero at the vertices $(x_0,y_0,z_0)^T$, $(x_1,y_1,z_1)^T$, $(x_2,y_2,z_2)^T$, and $(x_3,y_3,z_3)^T$, respectively, has the partial derivatives

$$f_x(x,y,z) \;=\; -\frac{1}{J} \det \begin{pmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{pmatrix}, \tag{40a}$$

$$f_y(x,y,z) \;=\; -\frac{1}{J}\;\det\begin{pmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{pmatrix}, \qquad \text{and} \tag{40b}$$

$$f_z(x,y,z) \;=\; -\frac{1}{J}\;\det\begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{pmatrix}. \tag{40c}$$

Integrals involving these partial derivatives are

$$\int\limits_{\mathcal{T}_i} \big(f_{i\,x}(x,y,z)\big)^2\; dx\; dy\; dz \;=\; \frac{1}{6}\;\sum_{j=0}^{n_i}\;\frac{1}{|J_j|}\;\det^2\begin{pmatrix} 1 & y_{j,1} & z_{j,1} \\ 1 & y_{j,2} & z_{j,2} \\ 1 & y_{j,3} & z_{j,3} \end{pmatrix}, \tag{41a}$$

$$\int\limits_{\mathcal{T}_i} \big(f_{i\,y}(x,y,z)\big)^2\; dx\; dy\; dz \;=\; \frac{1}{6}\;\sum_{j=0}^{n_i}\;\frac{1}{|J_j|}\;\det^2\begin{pmatrix} x_{j,1} & 1 & z_{j,1} \\ x_{j,2} & 1 & z_{j,2} \\ x_{j,3} & 1 & z_{j,3} \end{pmatrix}, \qquad \text{and} \tag{41b}$$

$$\int\limits_{\mathcal{T}_i} \big(f_{i\,z}(x,y,z)\big)^2\; dx\; dy\; dz \;=\; \frac{1}{6}\;\sum_{j=0}^{n_i}\;\frac{1}{|J_j|}\;\det^2\begin{pmatrix} x_{j,1} & y_{j,1} & 1 \\ x_{j,2} & y_{j,2} & 1 \\ x_{j,3} & y_{j,3} & 1 \end{pmatrix}, \tag{41c}$$

where the points $(x_{j,1}, y_{j,1}, z_{j,1})^T$, $(x_{j,2}, y_{j,2}, z_{j,2})^T$, and $(x_{j,3}, y_{j,3}, z_{j,3})^T$ denote the boundary vertices of the faces of the platelet tetrahedra associated with vertex $(x_i, y_i, z_i)^T$, see Fig. 17.

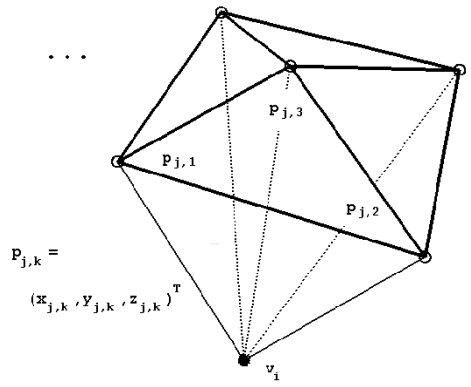

Fig. 17. Indexing scheme for platelet vertices (trivariate case).

Other required equations are

$$\int_{\mathcal{T}_{i,j}} f_{ix}(x,y,z) f_{jx}(x,y,z)\, dx\, dy\, dz$$

$$= \frac{1}{6} \sum_{k=0}^{n_{i,j}} \frac{1}{|J_k|} \det \begin{pmatrix} 1 & y_{k,1} & z_{k,1} \\ 1 & y_{k,2} & z_{k,2} \\ 1 & y_{k,3} & z_{k,3} \end{pmatrix} \det \begin{pmatrix} 1 & y_{k,0} & z_{k,0} \\ 1 & y_{k,3} & z_{k,3} \\ 1 & y_{k,2} & z_{k,2} \end{pmatrix}, \qquad (42a)$$

$$\int_{\mathcal{T}_{i,j}} f_{iy}(x,y,z) f_{jy}(x,y,z)\, dx\, dy\, dz$$

$$= \frac{1}{6} \sum_{k=0}^{n_{i,j}} \frac{1}{|J_k|} \det \begin{pmatrix} x_{k,1} & 1 & z_{k,1} \\ x_{k,2} & 1 & z_{k,2} \\ x_{k,3} & 1 & z_{k,3} \end{pmatrix} \det \begin{pmatrix} x_{k,0} & 1 & z_{k,0} \\ x_{k,3} & 1 & z_{k,3} \\ x_{k,2} & 1 & z_{k,2} \end{pmatrix}, \quad \text{and} \quad (42b)$$

$$\int_{\mathcal{T}_{i,j}} f_{iz}(x,y,z) f_{jz}(x,y,z)\, dx\, dy\, dz$$

$$= \frac{1}{6} \sum_{k=0}^{n_{i,j}} \frac{1}{|J_k|} \det \begin{pmatrix} x_{k,1} & y_{k,1} & 1 \\ x_{k,2} & y_{k,2} & 1 \\ x_{k,3} & y_{k,3} & 1 \end{pmatrix} \det \begin{pmatrix} x_{k,0} & y_{k,0} & 1 \\ x_{k,3} & y_{k,3} & 1 \\ x_{k,2} & y_{k,2} & 1 \end{pmatrix}, \qquad (42c)$$

where $(x_{k,0}, y_{k,0}, z_{k,0})^T$, $(x_{k,1}, y_{k,1}, z_{k,1})^T$, $(x_{k,2}, y_{k,2}, z_{k,2})^T$, and $(x_{k,3}, y_{k,3}, z_{k,3})^T$ are the vertices of a tetrahedron being a common tetrahedron of $\mathbf{v}_i$'s and $\mathbf{v}_j$'s platelets. The number of common tetrahedra is $n_{i,j} + 1$. Fig. 18 illustrates our indexing scheme.

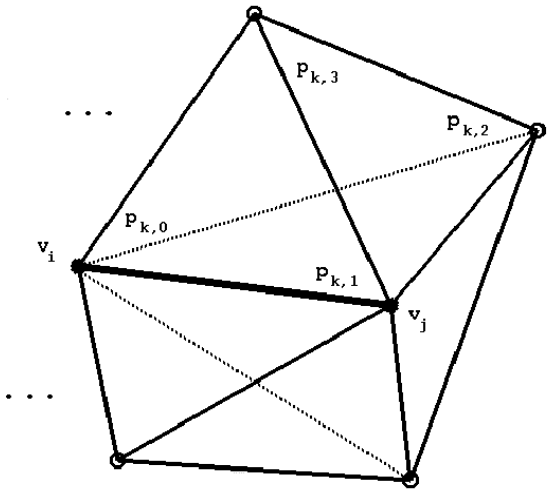

Fig. 18. Intersecting platelets (trivariate case).

References

Agarwal, P. K. and Desikan, P. K. (1997), An efficient algorithm for terrain simplification, in: *Proceedings of the 8th ACM/SIGACT-SIAM Symposium on Discrete Algorithms (SODA '97)*, Association for Computing Machinery, New York, NY, pp. 139–147.

Boehm, W. and Prautzsch, H. (1993), *Numerical Methods*, A K Peters, Ltd., Wellesley, MA.

Bonneau, G. P., Hahmann, S. and Nielson, G. M. (1996), BLAC-wavelets: A multiresolution analysis with non-nested spaces, in: Yagel, R. and Nielson, G. M., eds., *Visualization '96*, IEEE Computer Society Press, Los Alamitos, CA, pp. 43–48.

Cignoni, P., De Floriani, L., Montani, C., Puppo, E. and Scopigno, R. (1994), Multiresolution modeling and visualization of volume data based on simplicial complexes, in: Kaufman, A. E. and Krüger, W., eds., *1994 Symposium on Volume Visualization*, IEEE Computer Society Press, Los Alamitos, CA, pp. 19–26.

Cignoni, P., Montani, C., Puppo, E., and Scopigno, R. (1997), Multiresolution representation and visualization of volume data, IEEE Transactions of Visualization and Computer Graphics 3(4), pp. 352–369.

Duff, I. S., Erisman, A. M., and Reid, J. K. (1986), *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, England, pp. 239–251.

Dyn, N., Floater, M. S., and Iske, A. (2000), Adaptive thinning for bivariate scattered data, Technische Universität München, Fakultät für Mathematik, München, Germany, Report TUM M0006, 2000.

Dyn, N., Levin, D., and Rippa, S. (1990), Algorithms for the construction of data dependent triangulations, in: Mason, J. C. and Cox, M. G., eds., *Algorithms for Approximation II*, Chapman and Hall, New York, NY, pp. 185–192.

Eck, M., DeRose, A. D., Duchamp, T., Hoppe, H., Lounsbery, M. and Stuetzle, W. (1995), Multiresolution analysis of arbitrary meshes, in: Cook, R., ed., *Proceedings of SIGGRAPH 1995*, ACM Press, New York, NY, pp. 173–182.

Floater, M. S. and Iske, A. (1996a), Multistep scattered data interpolation using compactly supported radial basis functions, *Journal of Computational and Applied Mathematics*, 73(5), pp. 65–78.

Floater, M. S. and Iske, A. (1996b), Thinning and approximation of large sets of scattered data, in: F. Fontanella, K. Jetter and P. -J. Laurent, eds., *Advanced Topics in Multivariate Approximation*, World Scientific, Singapore, pp. 87–96.

Floater, M. S. and Reimers, M. (2001), Meshless parameterization and surface reconstruction, Computer Aided Geometric Design 18, pp. 77–92.

Franke, R. (1982), Scattered data interpolation: Tests of some methods, Math. Comp. 38, pp. 181–200.

Gieng, T. S., Hamann, B., Joy, K. I., Schussman, G. L. and Trotts, I. J. (1997), Smooth hierarchical surface triangulations, in: Yagel, R. and Hagen, H., eds., *Visualization '97*, IEEE Computer Society Press, Los Alamitos, CA, pp. 379–386.

Gieng, T. S., Hamann, B., Joy, K. I., Schussman, G. L. and Trotts, I. J. (1998), Constructing hierarchies for triangle meshes, IEEE Transactions on Visualization and Computer Graphics 4(2), pp. 145–161.

Gross, M. H., Gatti, R. and Staadt, O. (1995), Fast multiresolution surface meshing, in: Nielson, G. M. and Silver, D. eds., *Visualization '95*, IEEE Computer Society Press, Los Alamitos, CA, pp. 135–142.

Grosso, R., Lürig, C. and Ertl, T. (1997), The multilevel finite element method for adaptive mesh optimization and visualization of volume data, in: Yagel, R. and Hagen, H., eds., *Visualization '97*, IEEE Computer Society Press, Los Alamitos, CA, pp. 387–394.

Hagen, H., Müller, H. and Nielson, G. M., eds. (1993), *Focus on Scientific Visualization*, Springer-Verlag, New York, NY.

Hamann, B. (1994), A data reduction scheme for triangulated surfaces, Computer Aided Geometric Design 11(2), pp. 197–214.

Hamann, B. and Chen, J. L. (1994), Data point selection for piecewise linear curve approximation, Computer Aided Geometric Design 11(3), pp. 289–301.

Hamann, B. and Jordan, B. W. (1998), Triangulations from repeated bisection, in: Dæhlen, M., Lyche, T. and Schumaker, L. L., eds., *Mathematical Methods for Curves and Surfaces II*, Vanderbilt University Press, Nashville, TN, pp. 229–236.

Hamann, B., Jordan, B. W. and Wiley, D. F. (1999), On a construction of a hierarchy of best linear spline approximations using repeated bisection, IEEE Transactions on Visualization and Computer Graphics 5(1), pp. 30–46, and 5(2), p. 190 (errata).

Hoppe, H. (1996), Progressive meshes, in: Rushmeier, H., ed., *Proceedings of SIGGRAPH 1996*, ACM Press, New York, NY, pp. 99–108.

Hoppe, H. (1997), View-dependent refinement of progressive meshes, in: Whitted, T., ed., *Proceedings of SIGGRAPH 1997*, ACM Press, New York, NY, pp. 189–198.

Kaufman, A. E., ed. (1991), *Volume Visualization*, IEEE Computer Society Press, Los Alamitos, CA.

Nadler, E. (1986), Piecewise linear best L_2 approximation on triangulations, in: Ward, J. D., ed., *Approximation Theory V*, Academic Press, Inc., San Diego, CA, pp. 499–502.

Nielson, G. M. (1993), Scattered data modeling, IEEE Computer Graphics and Applications 13(1), pp. 60–70.

Nielson, G. M., Jung, I.-H. and Sung, J. (1997a), Haar wavelets over triangular domains with applications to multiresolution models for flow over a sphere, in: Yagel, R. and Hagen, H., eds., *Visualization '97*, IEEE Computer Society Press, Los Alamitos, CA, pp. 143–149.

Nielson, G. M., Müller, H. and Hagen, H., eds. (1997b), *Scientific Visualization: Overviews, Methodologies, and Techniques*, IEEE Computer Society Press, Los Alamitos, CA.

Nielson, G. M. and Shriver B. D., eds. (1990), *Visualization in Scientific Computing*, IEEE Computer Society Press, Los Alamitos, CA.

Rippa, S. (1992), Long and thin triangles can be good for linear interpolation, SIAM J. Numer. Anal. 29(1), pp. 257–270.

Rosenblum, L. J, Earnshaw, R. A., Encarnação, J. L., Hagen, H., Kaufman, A. E., Klimenko, S., Nielson, G. M., Post, F. and Thalmann, D., eds. (1994), *Scientific Visualization—Advances and Challenges*, IEEE Computer Society Press, Los Alamitos, CA.

Staadt, O. G., Gross, M. H. and Weber, R. (1997), Multiresolution compression and reconstruction, in: Yagel, R. and Hagen, H., eds., *Visualization '97*, IEEE Computer Society Press, Los Alamitos, CA, pp. 337–346.

Trotts, I. J., Hamann, B. and Joy, K. I. (1999), Simplification of tetrahedral meshes with error bounds, IEEE Transactions on Visualization and Computer Graphics 5(3), pp. 224–237.

Trotts, I. J., Hamann, B., Joy, K. I. and Wiley, D. F. (1998), Simplification of tetrahedral meshes, in: Ebert, D. S., Hagen, H. and Rushmeier, H. E., eds., *Visualization '98*, IEEE Computer Society Press, Los Alamitos, California, pp. 287–295.

Xia, J. C. and Varshney, A. (1996), Dynamic view-dependent simplification for polygonal meshes, in: Yagel, R. and Nielson, G. M., eds., *Visualization '96*, IEEE Computer Society Press, Los Alamitos, CA, pp. 327–334.

Zienkiewicz, O. C. and Taylor, R. L. (2000), *The Finite-Element Method*, vols. 1–3, fifth edition, Butterworth-Heinemann, Oxford, United Kingdom.