[**Paper Appeared in IEEE TVCG 5(1), 1999, pp. 30–46.**]

On a Construction of a Hierarchy of Best Linear Spline Approximations Using Repeated Bisection

Bernd HAMANN [†,+,*], Benjamin W. JORDAN [‡] and David F. WILEY [†]

Abstract

We present a method for the construction of hierarchies of single-valued functions in one, two, and three variables. The input to our method is a coarse decomposition of the compact domain of a function in the form of an interval (univariate case), triangles (bivariate case), or tetrahedra (trivariate case). We compute best linear spline approximations, understood in an integral least squares sense, for functions defined over such triangulations and refine triangulations using repeated bisection. This requires the identification of the interval (triangle, tetrahedron) with largest error and splitting it into two intervals (triangles, tetrahedra). Each bisection step requires the re-computation of all spline coefficients due to the global nature of the best approximation problem. Nevertheless, this can be done efficiently by bisecting multiple intervals (triangles, tetrahedra) in one step and by reducing the bandwidths of the matrices resulting from the normal equations.

Key words: Approximation; Bisection; Best approximation; Grid generation; Hierarchical representation; Linear spline; Multiresolution method; Scattered data; Spline; Triangulation; Unstructured grid; Visualization.

1. Introduction

Different methods are known and used for the hierarchical representation of very large data sets. Unfortunately, only a small number of these methods are based on a well developed mathematical theory. In the context of visualizing very large data sets in two and three dimensions, it is imperative to develop hierarchical data representations that allow us to visualize and analyze physical phenomena at various levels of detail. General, robust, and efficient methodologies are needed to support the generation of hierarchical

[†] *Department of Computer Science, University of California, Davis, CA 95616-8562, USA*

[+] Co-Director of the Center for Image Processing and Integrated Computing (CIPIC)

[*] **Corresponding author**; e-mail: `hamann@cs.ucdavis.edu`

[‡] *Pixar Feature Division, Pixar Animation Studios, 1001 West Cutting Boulevard, Richmond, CA 94804, USA*

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 of a data hierarchy is repeated bisection of intervals (triangles, tetrahedra). The coefficients associated with each vertex in a triangulation are computed in a best approximation sense. (We use the term *triangulation* to describe the general situation.) Whenever an interval (a triangle/tetrahedron) is bisected, due to a large local error, one needs to compute new linear spline coefficients for all vertices in the new, refined triangulation. It is possible to efficiently perform the necessary matrix inversions due to the fact that the matrices are generally very sparse which allows us to significantly reduce their *bandwidths*.

Over the past decade, the visualization community has developed various, often highly specialized, hierarchical data representations given the increasing need for such representations. Triangulations are a good choice when complex data domains must be handled. This is the reason why we are using triangulations for hierarchical data representation. Recent work related to our approach and dealing with the approximation of large data sets is discussed in [Agarwal & Desikan '97], [Bonneau *et al.* '96], [Cignoni *et al.* '94], [Dyn *et al.* '90], [Eck *et al.* '95], [Gieng *et al.* '97, '98], [Gross *et al.* '95], [Grosso *et al.* '97], [Hamann '94], [Hamann & Chen '94], [Hoppe '96], [Nielson *et al.* '97a], [Trotts *et al.* '98], and [Xia & Varshney '96]. These papers discuss multi-level approximations of functions depending on one, two, or three variables. We do not discuss the particular aspects when concerned with the representation and manipulation of complex, general meshes/triangulations.

We describe an approach that applies to univariate, bivariate, and trivariate data. The main principles of our approach become evident from the discussion of the univariate case. By providing an in-depth discussion of the univariate case first, the necessary generalizations for the bivariate and trivariate cases are understood more easily. We have developed our method having the requirements for bivariate and trivariate visualization in mind. Hierarchies of triangular and tetrahedral meshes provide the necessary flexibility to discretize complicated domains, and visualization systems can handle such data representations efficiently. Visualization techniques that are relevant in this context include contouring, slicing, and volume visualization (ray casting) methods. The books [Hagen *et al.* '93], [Kaufman '91], [Nielson *et al.* '97b], [Nielson & Shriver '90], and [Rosenblum *et al.* '94] provide an excellent overview of current visualization technology.

2. Overview of the technique

The generation of a hierarchical data representation should be viewed as a pre-processing step of data visualization and analysis. It should be possible to directly utilize the data format describing a data hierarchy in the visualization process. Visualization technology and systems are capable of rendering triangulation-based data. Regarding the generation of the hierarchy, speed is not the primary concern, as long as it is possible to construct the hierarchy in reasonable time. It is important that the hierarchical representation can describe arbitrarily complex domains and that error estimates can be computed for all levels of the hierarchy. The main criteria that have influenced the design of our hierarchical approximation technique are:

- **Simplicity.** The construction of a hierarchy of triangulations should be simple. The number of special cases to be considered should be small; a single refinement step should cause minimal topological change; and the computations involved in the generation of best linear spline approximations should be simple.

- **Generality.** The approach should be applicable to functions of any number of variables; it should be possible to extend the approach to multi-valued functions; and the approach should handle arbitrary, complicated regions.

- **Efficiency.** The computation of the hierarchy of triangulations should be efficient.

- **Applicability.** The hierarchical data representation should consider the needs of existing visualization technology. One should be able to directly apply known visualization methods to the hierarchical data format.

A hierarchy of best linear spline approximations, based on domain triangulations of varying resolution, seems to be a good choice.

The individual steps of our algorithm needed to compute such a hierarchical representation for univariate, bivariate, and trivariate functions are:

i. Initial approximation. Define an initial, coarse triangulation of the function's domain and, for all vertices, compute the coefficients defining the best linear spline approximation.

ii. Error estimation. Analyze the error of this approximation by computing appropriate global and local error estimates.

iii. Refinement. Identify the interval (triangle, tetrahedron) with largest local error estimate and bisect it (bisect its longest edge into two segments in the bivariate and trivariate cases—thereby effectively bisecting all triangles/tetrahedra sharing this edge). Bisection is illustrated in Fig. 1 for the bivariate and trivariate cases.

iv. Computation of best approximation. Based on the updated vertex set and updated triangulation, compute a new best linear spline approximation.

v. Iteration. Iterate steps **ii**, **iii**, and **iv** until a certain approximation error condition is satisfied.

We discuss these steps in detail in the following sections.

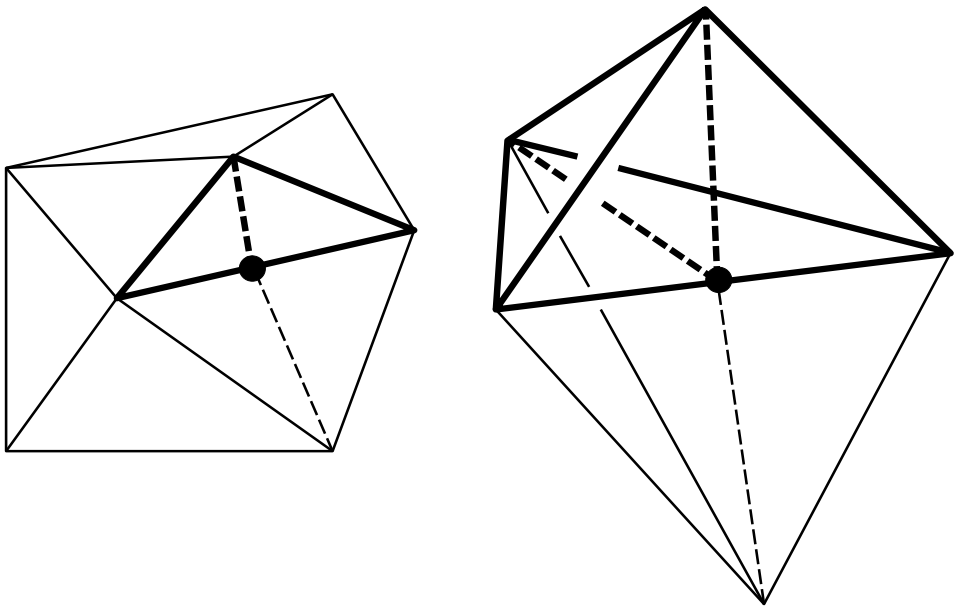

Fig. 1. Bisecting triangle/tetrahedron (boldface lines indicating original triangle/tetrahedron selected for bisection).

Remark 2.1. We point out that, in many practical applications, the function to be approximated is only known at a finite number of random locations (scattered data). If this is the case, we first compute an interpolant to the scattered data. We use a localized version of *Hardy's multiquadric* method [1], yielding a function interpolating the known function values at the random locations. We interpret this scattered data interpolant as the function for which an approximation hierarchy is to be constructed. For a survey on scattered data approximation and interpolation techniques, including a description of Hardy's multiquadric method, see [Franke '82]. In fact, we assume that the function to be approximated is the result of a previously performed (scattered) data approximation or interpolation step. Thus, the number of original (scattered) data should impact the degree of "complexity" of the function.

3. Best approximation and the univariate case

Our approach requires a few notions from linear algebra and approximation theory which we discuss briefly in the following. We use the standard scalar product $\langle f, g \rangle$ for two functions $f(x)$ and $g(x)$, defined over the interval $[a, b]$,

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

[1] There are several advantages to using Hardy's multiquadric approximation method, the main ones being smoothness of the approximation and ease of computation.

and the standard L_2 norm to measure a function $f(x)$,

$$||f|| \;=\; \langle f, f\rangle^{1/2} \;=\; \left(\int_a^b \big(f(x)\big)^2 \, dx \right)^{1/2} . \tag{2}$$

Remark 3.1. As an alternative, one might consider an *interval-weighted* scalar product for two functions f and g by dividing the right-hand side of (1) by $(b-a)$. This might potentially eliminate the influence of interval length when computing interval-specific error estimates. It is our experience that performing this "normalization" does not lead to a hierarchical representation whose visual appearance is significantly different (keeping the number of knots similar in both representations). Therefore, we do not perform "normalization" to obtain better computational efficiency.

It is well known from approximation theory, see *e.g.*, [Davis '75], that the best approximation [2] of a function F, when approximating it by a linear combination $\sum_{i=0}^{n-1} c_i f_i$ of independent functions f_0, ..., f_{n-1}, is defined by the normal equations

$$\begin{pmatrix} \langle f_0, f_0\rangle & \cdots & \langle f_{n-1}, f_0\rangle \\ \vdots & & \vdots \\ \langle f_0, f_{n-1}\rangle & \cdots & \langle f_{n-1}, f_{n-1}\rangle \end{pmatrix} \begin{pmatrix} c_0 \\ \vdots \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} \langle F, f_0\rangle \\ \vdots \\ \langle F, f_{n-1}\rangle \end{pmatrix} . \tag{3}$$

We also write this linear system as $M^{[n-1]}\mathbf{c}^{[n-1]} = \mathbf{F}^{[n-1]}$. This system is easily solved when dealing with a set of mutually orthogonal and normalized basis functions, *i.e.*, $\langle f_i, f_j\rangle = \delta_{i,j}$ (Kronecker delta). In this case, only the diagonal elements of $M^{[n-1]}$ are non-zero, and the coefficients c_i are given by $c_i = \langle F, f_i\rangle$. For an arbitrary set of basis functions f_i one has to investigate means for the efficient solution of the linear system or use a basis leading to sparse matrices $M^{[n-1]}$.

We are concerned with the computation of a hierarchy of best approximations of a given function F. More specifically, we are interested in a construction of a hierarchy of best linear spline approximations of F by increasing the number of basis functions—or, in other words, the number of *knots*. We increase the number of knots one-by-one until a best approximation is obtained whose associated error is smaller than some threshold. From the set of all best linear spline approximations to F we eventually select a subset consisting of those splines that we can associate with a particular *level* of an *approximation hierarchy*.

In general, the computation of a best linear spline approximation of a function F is an optimization problem, which, when also allowing variable knot locations, is quite involved, see [Cantoni '71], [Stone '61], and [Tomek '74]. Considering variable knot locations is a subject for future investigations. Our current approach does not permit variable knot locations, it is solely based on repeated bisection, *i.e.*, intervals (triangles and tetrahedra in the bivariate and trivariate cases). This approach is computationally much less expensive than the general, "knot selection," problem. Our bisection approach essentially requires two steps:

[2] We understand *best approximation* in an integral least squares sense.

- **Error computation.** One computes error estimates for each interval (triangle, tetrahedron) in a particular best linear spline approximation and determines the one with largest error.
- **Bisection.** The midpoint (of the longest edge) of the interval (triangle, tetrahedron) with largest error is inserted as an additional vertex. (All triangles/tetrahedra sharing this edge are bisected as well.) A new best linear spline approximation is computed for the new vertex set.

Initially, we approximate a univariate function F by a single linear spline segment by computing $M^{[1]}\mathbf{c}^{[1]} = \mathbf{F}^{[1]}$. We assume, unless specified otherwise, that F is defined over the interval $[0,1]$ and that the basis functions f_i are hat functions, linear spline basis functions with the property $f_i(x_j) = \delta_{i,j}$, see Fig. 2. The resulting initial error is $E^{[1]} = ||F - (c_0 f_0 + c_1 f_1)||$. If this value is larger than a specified threshold, we refine the approximation by inserting an additional knot at $x = \frac{1}{2}$. (The insertion of this knot changes the knot sequence, and therefore the hat basis functions change.) We compute a new best approximation by solving $M^{[2]}\mathbf{c}^{[2]} = \mathbf{F}^{[2]}$ for the new set of basis functions and obtain the new error value $E^{[2]} = \left|\left|F - \sum_{i=0}^{2} c_i f_i\right|\right|$. Should this error still be too large, we need a criterion that allows us to decide which segment to bisect. Since our method is based on repeated bisection of individual segments, we consider segment-specific, local errors associated with the intervals $\left[0, \frac{1}{2}\right]$ and $\left[\frac{1}{2}, 1\right]$.

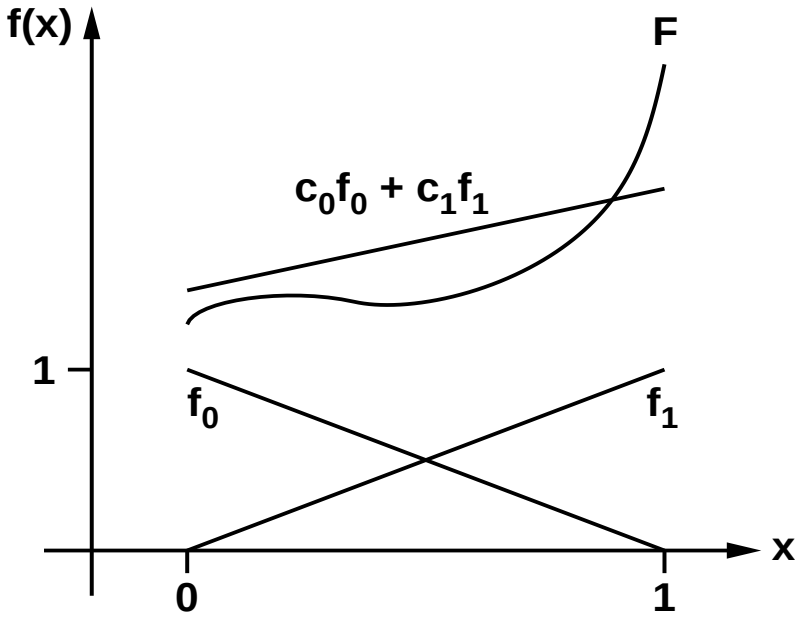

Fig. 2. Basis functions f_i, function F, and F's best approximation.

At this point, we introduce notions that allow us to describe the refinement process for a general intermediate knot sequence. We assume that the current approximation is

based on the knot sequence $0 = x_0 < x_1 < x_2 < ... < x_{k-2} < x_{k-1} = 1$ and that the coefficient vector associated with the hat basis functions f_i is $(c_0, ..., c_{k-1})^T$. We define the *global error* as

$$E^{[k-1]} \;=\; \left\| \; F - \sum_{i=0}^{k-1} \; c_i f_i \; \right\|. \tag{4}$$

In order to determine the interval $[x_i, x_{i+1}]$ to be subdivided next, we define segment-specific, *local errors* as

$$e_i^{[k-1]} \;=\; \left(\int_{x_i}^{x_{i+1}} \big(F - (c_i f_i + c_{i+1} f_{i+1})\big)^2 \; dx \right)^{1/2}, \qquad i = 0, ..., k-2. \tag{5}$$

We compute the local error values for each segment and bisect the segment with largest local error value. If there are multiple segments with the same maximal error value, we randomly pick one of them to be bisected. (It is of course possible to bisect all segments with the same maximal error in one step, thus leading to a unique answer.) One could generalize this approach by selecting more than just one segment for refinement, *e.g.*, one could simultaneously bisect the m segments with the m largest local error values. Such an approach seems appropriate when dealing with extremely large data sets leading to extremely complicated functions F to be approximated.

When using hat functions as basis functions, the only non-zero elements of $M^{[k-1]}$, for a particular row i, are the elements $\langle f_{i-1}, f_i \rangle$, $\langle f_i, f_i \rangle$, and $\langle f_{i+1}, f_i \rangle$. These scalar products are given by

$$\begin{aligned} \langle f_{i-1}, f_i \rangle \;&=\; \frac{1}{6} \; \triangle_{i-1}, \\ \langle f_i, f_i \rangle \;&=\; \frac{1}{3} \; \big(\triangle_{i-1} + \triangle_i\big), \qquad \text{and} \\ \langle f_{i+1}, f_i \rangle \;&=\; \frac{1}{6} \; \triangle_i, \end{aligned} \tag{6}$$

where $\triangle_j = x_{j+1} - x_j$. Thus, $M^{[k-1]}$ is the tridiagonal matrix

$$M^{[k-1]} \;=\; \frac{1}{6} \begin{pmatrix} 2\triangle_0 & \triangle_0 & & & & \\ \triangle_0 & 2(\triangle_0 + \triangle_1) & \triangle_1 & & & \\ & \triangle_1 & 2(\triangle_1 + \triangle_2) & \triangle_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \triangle_{k-2} & 2\triangle_{k-2} \end{pmatrix}. \tag{7}$$

This allows us to compute each best linear spline approximation in linear time. It is necessary, due to the global nature of the problem, to re-compute all components of the coefficient vectors $(c_0, c_1, c_2, ...)^T$ whenever one inserts a knot. An "insert-knots-one-by-one" approach is thus a fairly inefficient approach when several thousand knots are required to obtain a "good" approximation.

The scalar products $\langle F, f_i \rangle$ and the error values $E^{[k-1]}$ and $e_i^{[k-1]}$ are computed by numerical integration. We use *Romberg integration*, see [Boehm & Prautzsch '93] and [Hämmerlin & Hoffmann '91]. We review Romberg integration and describe our specific numerical integration schemes for the bivariate and trivariate cases in the Appendix.

Remark 3.2. We plan to devise a more efficient method that inserts multiple knots simultaneously. Such a method should relate the locations and the number of new knots to be inserted to the "local complexity" of F, the local error behavior of F.

When does one associate a certain best linear spline approximation with a particular level? Given an error tolerance tuple $\mathbf{E} = (\epsilon_0, ..., \epsilon_{m-1})$, $\epsilon_j > \epsilon_{j+1}$, we associate a best approximation with approximation level j when its associated global error lies in the interval $(\epsilon_{j+1}, \epsilon_j]$. A best linear spline approximation associated with level j is called the *representative of level* j if its number of knots (vertices) is minimal among all level-j approximations [3]. Often, one is interested only in a best approximation whose associated global error is below a certain threshold. This is the reason why we use the concept of the representative approximation—the approximation with minimal number of knots satisfying a level-specific upper threshold—and store only the representatives. It would also be impractical, if not impossible for large data sets, to store all the linear splines resulting from all individual bisection steps [4]. (The univariate case is discussed in depth in [Hamann & Jordan '98].)

4. The bivariate case

We proceed similarly in the bivariate case. Given an initial best linear spline approximation based on a small number of triangles, possibly just one, we compute the global approximation error of the associated best linear spline approximation and, should this error be too large, insert the midpoint of the longest edge of the triangle with largest local error as a new knot. In general, the principle of edge bisection leads to the split of one or two triangles—depending on whether the bisected edge is shared by a second triangle or not.

A bisection step leads to a new triangulation for which we compute a new best linear spline approximation. Bisection terminates when a certain global error condition is met. If the triangle with largest local error has more than one edge with maximal length, we choose to bisect, randomly, one of the edges with maximal length. (Alternatively, one could choose to split multiple edges simultaneously to guarantee uniqueness.) If multiple

[3] In practice, we refine an intermediate linear spline until we obtain a spline whose associated global error is smaller than or equal to ϵ_j. In order to construct a level-j linear spline with a minimal number of knots, obtained by repeated bisection, one would have to solve a multidimensional optimization problem for the determination of a sequence consisting of a minimal number of bisection steps. We do not determine this optimal sequence. Instead, we perform only one sequence of bisection steps and store, for each level j, the first generated linear spline whose global error is smaller than or equal to ϵ_j.

[4] It is possible that our construction does not produce a linear spline whose global error lies in the interval $(\epsilon_{j+1}, \epsilon_j]$. This is due to the fact that a single bisection step might reduce the global error drastically—but this has no severe consequences.

triangles share the maximal local error value, we choose, randomly, one of these triangles to be bisected. (Again, one could choose to bisect larger triangles first—or vice versa—or choose to simultaneously split all triangles sharing the maximal local error value.) Bisecting the m triangles with the m largest local errors is more appropriate when dealing with very large data sets.

We describe the construction of a hierarchy of best linear (also called triangular) spline approximations of functions F(x,y) defined over arbitrary triangulations. The union of the triangles in the initial triangulation defines the region in the plane over which we apply our refinement scheme. It is assumed that the boundary of the domain of interest is approximated well enough by the coarsest, initial triangulation. When dealing with scattered data we suggest to consider a triangulation of the data points defining the convex hull, the "natural boundary," as the initial triangulation. We introduce some necessary notions next.

Denoting the vertex (knot) set of an intermediate approximation by $\{\mathbf{v}_i = (x_i, y_i)^T | i = 0, ..., k-1\}$ and the coefficient vector associated with the hat basis functions by f_i is $(c_0, ..., c_{k-1})^T$, the global error is

$$E^{[k-1]} \;=\; \left|\left|\; F - \sum_{i=0}^{k-1} c_i f_i \;\right|\right|. \tag{8}$$

The triangle-specific, local errors for each triangle T_j are given by

$$e_j^{[k-1]} \;=\; \left(\int_{T_j} \big(F - \text{spline\_over\_}T_j\big)^2 \; dxdy \right)^{1/2}, \qquad j = 0, ..., n_T - 1, \tag{9}$$

where n_T is the number of triangles.

Assuming that there are m vertices in the initial, coarsest triangulation, we compute the first best linear spline approximation by solving $M^{[m-1]}\mathbf{c}^{[m-1]} = \mathbf{F}^{[m-1]}$. (Each vertex has an associated hat function and thus an associated spline coefficient.) The initial error is $E^{[m-1]} = \left|\left|F - \sum_{i=0}^{m-1} c_i f_i\right|\right|$. If this value is larger than a prescribed tolerance, we insert an additional knot at the midpoint of the longest edge of the triangle with largest local error—thereby altering, locally, the set of hat basis functions. We compute a new best approximation by solving $M^{[m]}\mathbf{c}^{[m]} = \mathbf{F}^{[m]}$ and stop when a best approximation is obtained that satisfies a global error condition. Due to the fact that our basis functions have compact support, the insertion of a new knot implies the computation of only a few new scalar products. Therefore, we store the matrices describing the normal equations of a previous level for efficiency.

In the bivariate setting one must determine the local errors by considering the difference between F and an intermediate best linear approximation over each triangle. Fig. 3 shows a function F, the hat functions, and F's best approximation (single-triangle case).

Remark 4.1. *Data-dependent triangulation* schemes use the approximation error as primary criterion to decide how to triangulate a function's domain, see, *e.g.*, [Dyn *et al.* '90]. Instead of always bisecting a triangle's longest edge one could choose to bisect a triangle in

such a way that the new global approximation error is minimized. Thus, in the bivariate case, one would have to consider three possible bisections.

Remark 4.2. Our algorithm does not include solving the following problem: *Given a particular error bound, how can one construct an approximation with a minimum number of nodes that satisfies the error bound?* This is an interesting and challenging problem, and one solution approach is described in [Agarwal & Desikan '97].

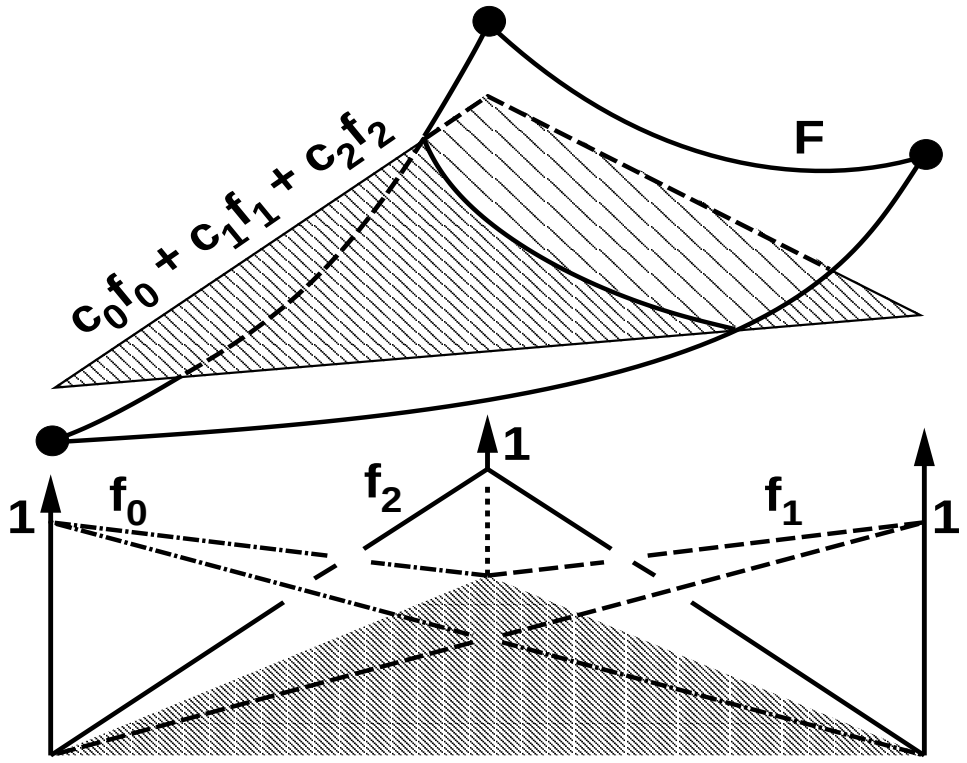

Fig. 3. Basis functions f_i, function F, and F's best approximation.

The solution of the normal equations, considering k knots, requires the inversion of a matrix $M^{[k-1]}$. The number of non-zero entries per column in $M^{[k-1]}$ is implied by the valences of the vertices in the triangular mesh. A vertex $\mathbf{v}_i$ having valence v_i causes $v_1 + 1$ non-zero entries in column i. In principle, the valences are unlimited, and one can obtain large numbers of non-zero elements in certain columns of $M^{[k-1]}$. Thus, the computation of the spline coefficients in the bivariate case can, in principle, be much more expensive than in the univariate case.

Remark 4.3. Efficiency is a concern when performing a bisection step for a triangulation consisting of a large number of vertices. Each bisection step requires the re-computation of the coefficients for all vertices, involving the (efficient) inversion of a banded matrix. The matrix $M^{[k-1]}$ is sparse, *i.e.*, relatively few matrix entries are different from zero. Unfortunately, the non-zero elements are scattered throughout the matrix and their locations depend on the indexing of the vertices in the triangulation. Finite element problems result in the same problem, inverting sparse matrices efficiently, see *e.g.*, [Zienkiewicz '77]. Al-

gorithms have been developed for the "compression" of sparse matrices by reducing their bandwidths. These algorithms determine a (nearly) optimal indexing scheme for the vertices such that the resulting matrices have (nearly) minimal bandwidths. The results of such algorithms are matrices where the non-zero elements are closely "packed" in a small number of bands close to the main matrix diagonal, see [Cuthill & McKee '69], [Gibbs *et al.* '76], and [Rosen '68]. We use the graph-based Cuthill-McKee algorithm to generate vertex indices leading to small-bandwidth matrices.

Remark 4.4. A lower bound for the bandwidth that one can achieve when applying any type of bandwidth-reducing strategy is defined by the vertex with maximal valence. In principle, arbitrarily high vertex valences can result from repeated bisection. One might therefore consider imposing threshold on maximal vertex valence.

Remark 4.5. Repeated bisection of longest triangle edges seems to lead to acceptable triangulations in the sense that the geometrical properties of the resulting triangles do not rapidly deteriorate to long, skinny triangles. This is an important observation considering that most visualization techniques are numerically very sensitive to triangle shape. In principle, arbitrarily skinny triangles can result from repeated bisection. Therefore, one should incorporate a constraint into the algorithm that prohibits the generation of triangles whose geometrical qualities are below a certain threshold. In our current implementation we do not impose any constraints limiting the shape of triangles. Our approach seems to favor the generation of equilateral triangles over skinny triangles overall, and it would be an interesting topic for future research why this is the case.

The bivariate case requires integration over triangles. For this purpose we utilize the *change-of-variables theorem* which allows us to effectively integrate functions over a triangle with vertices $\mathbf{v}_0 = (x_0, y_0)^T$, $\mathbf{v}_1 = (x_1, y_1)^T$, and $\mathbf{v}_2 = (x_2, y_2)^T$.

Change-of-variables theorem. Let R and R^* be regions in the plane and let $M : R^* \to R$ be a C^1-continuous, one-to-one mapping such that $M(R^*) = R$. Then, for any bivariate integrable function f, the equation

$$\int_R f(x,y)\; dxdy \;=\; \int_{R^*} f\big(x(u,v), y(u,v)\big)\; J\; dudv \tag{10}$$

holds, where J is the Jacobian of M,

$$J \;=\; \det \begin{pmatrix} \frac{\partial}{\partial u}x(u,v) & \frac{\partial}{\partial v}x(u,v) \\ \frac{\partial}{\partial u}y(u,v) & \frac{\partial}{\partial v}y(u,v) \end{pmatrix}. \tag{11}$$

Thus, we can effectively compute integrals of functions defined over triangles. We only need to consider the linear transformation

$$\begin{pmatrix} x(u,v) \\ y(u,v) \end{pmatrix} \;=\; \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}. \tag{12}$$

This transformation maps the *standard triangle* T^* with vertices $\mathbf{u}_0 = (0,0)^T$, $\mathbf{u}_1 = (1,0)^T$, and $\mathbf{u}_2 = (0,1)^T$ in the uv-plane to the arbitrary triangle T with vertices $\mathbf{v}_0 = (x_0, y_0)^T$,

$\mathbf{v}_1 = (x_1, y_1)^T$, and $\mathbf{v}_2 = (x_2, y_2)^T$ in the xy-plane. (Both triangles must be oriented counterclockwise.) For this linear mapping the change-of-variables theorem yields

$$\int_T f(x,y)\; dxdy \;=\; J \int_{v=0}^{1} \int_{u=0}^{1-v} f\big(x(u,v), y(u,v)\big)\; dudv, \tag{13}$$

and the Jacobian is given by

$$J \;=\; \det \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix}. \tag{14}$$

The only scalar products of basis function pairs one must consider in the bivariate case are $\langle N_0, N_0 \rangle$ and $\langle N_0, N_1 \rangle$, where $N_i(u_j, v_j) = \delta_{i,j}$ is a linear spline basis function defined over the standard triangle. (A scalar product $\langle f_i, f_j \rangle$ is different from zero only if the platelets of the vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ have a non-zero area intersection.) The values of these two scalar products are

$$\begin{aligned} \langle N_0, N_0 \rangle \;&=\; \int_{v=0}^{1} \int_{u=0}^{1-v} (1-u-v)^2\; dudv = \;\frac{1}{12} \qquad \text{and} \\ \langle N_0, N_1 \rangle \;&=\; \int_{v=0}^{1} \int_{u=0}^{1-v} (1-u-v)\; u\; dudv = \;\frac{1}{24}. \end{aligned} \tag{15}$$

A hat function f_i associated with vertex $\mathbf{v}_i$ is the linear spline basis function whose function value at $\mathbf{v}_i$ is one and whose function value at all other vertices is zero; the function f_i varies linearly between zero and one over all triangles defining $\mathbf{v}_i$'s *platelet*, see Fig. 4. (The platelet of vertex $\mathbf{v}_i$ is the set of all triangles sharing $\mathbf{v}_i$ as a common vertex.)

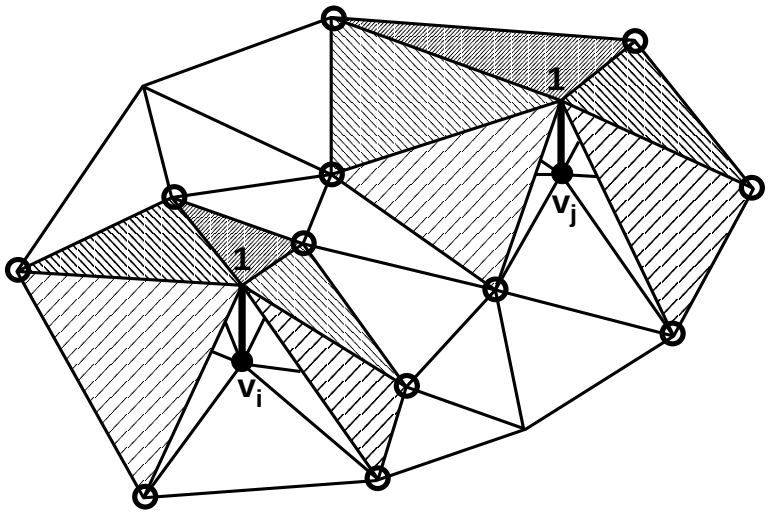

Fig. 4. Platelets of $\mathbf{v}_i$ and $\mathbf{v}_j$ and associated basis functions.

Thus, the scalar product $\langle f_i, f_i \rangle$ is given by

$$\langle f_i, f_i \rangle \;=\; \sum_{j=0}^{n_i-1} \int\limits_{T_j} f_i \; f_i \; dxdy \;=\; \frac{1}{12} \sum_{j=0}^{n_i-1} J_j, \tag{16}$$

where n_i is the number of platelet triangles associated with vertex $\mathbf{v}_i$, and J_j is the Jacobian associated with the j-th platelet triangle. (The platelet of triangles associated with vertex $\mathbf{v}_i$ is assumed to be the set of triangles $\mathcal{T}_i = \{T_j\}_{j=0}^{n_i-1}$.) The scalar product $\langle f_j, f_k \rangle$ of two basis functions whose associated vertices $\mathbf{v}_j$ and $\mathbf{v}_k$ are connected by an edge is given by

$$\langle f_j, f_k \rangle \;=\; \sum_{i=0}^{n_{j,k}-1} \int\limits_{T_i} f_j \; f_k \; dxdy \;=\; \frac{1}{24} \sum_{i=0}^{n_{j,k}-1} J_i. \tag{17}$$

(The set $\mathcal{T}_{j,k} = \{T_i\}_{i=0}^{n_{j,k}-1}$ is the set of triangles belonging to both $\mathbf{v}_j$'s and $\mathbf{v}_k$'s platelets.) All triangles must be oriented counterclockwise. We transform each triangle in $\mathbf{v}_j$'s platelet to the standard triangle such that $\mathbf{v}_j$ is mapped to $\mathbf{u}_0$ and transform each triangle in $\mathbf{v}_k$'s platelet to the standard triangle such that $\mathbf{v}_k$ is mapped to $\mathbf{u}_1$. This is the reason why one only needs to consider the scalar product $\langle N_0, N_1 \rangle$.

5. The trivariate case

The trivariate case is a rather straightforward generalization of the bivariate case. In the context of scientific visualization, the trivariate case is probably the most important one. The generalization to the trivariate case is simply done by replacing the notion of triangle by the notion of tetrahedron. The basic idea of applying repeated bisection remains the

same. We bisect tetrahedra according to their local errors and compute best linear spline approximation using hat basis functions defined over tetrahedral platelets, see Fig. 5. The computation of scalar products and errors requires integration over tetrahedral domains, see Appendix.

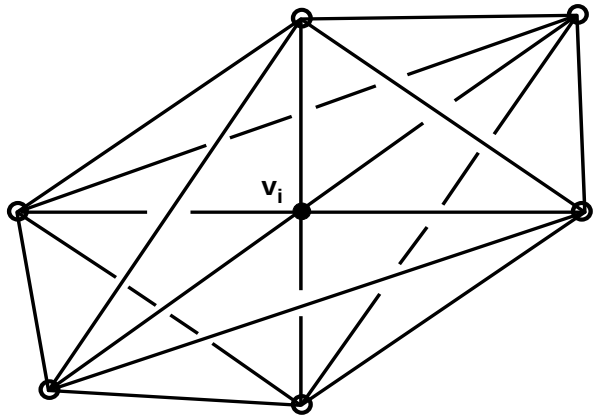

Fig. 5. Tetrahedral platelet of $\mathbf{v}_i$ consisting of eight tetrahedra.

We only discuss the implications of the change-of-variable theorem for tetrahedral meshes, which is important in the context of computing the required scalar products of functions defined over tetrahedral domains. We consider the transformation mapping the *standard tetrahedron* with vertices $\mathbf{u}_0 = (0,0,0)^T$, $\mathbf{u}_1 = (1,0,0)^T$, $\mathbf{u}_2 = (0,1,0)^T$, and $\mathbf{u}_3 = (0,0,1)^T$ in uvw-space to the tetrahedron with vertices $\mathbf{v}_0 = (x_0, y_0, z_0)^T$, $\mathbf{v}_1 = (x_1, y_1, z_1)^T$, $\mathbf{v}_2 = (x_2, y_2, z_2)^T$, and $\mathbf{v}_3 = (x_3, y_3, z_3)^T$ in xyz-space. The resulting linear transformation is thus given by

$$\begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 & x_3 - x_0 \\ y_1 - y_0 & y_2 - y_0 & y_3 - y_0 \\ z_1 - z_0 & z_2 - z_0 & z_3 - z_0 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}. \tag{18}$$

In this case, the change-of-variables theorem implies that

$$\int_T f(x,y,z)\; dxdydz \;=\; J \int_{w=0}^{1} \int_{v=0}^{1-w} \int_{u=0}^{1-v-w} f\big(x(u,v,w), y(u,v,w), z(u,v,w)\big)\; dudvdw, \tag{19}$$

where the Jacobian is given by

$$J \;=\; \det \begin{pmatrix} x_1 - x_0 & x_2 - x_0 & x_3 - x_0 \\ y_1 - y_0 & y_2 - y_0 & y_3 - y_0 \\ z_1 - z_0 & z_2 - z_0 & z_3 - z_0 \end{pmatrix}. \tag{20}$$

Using the same argument as in the bivariate case, the only scalar products one needs are $\langle N_0, N_0 \rangle$ and $\langle N_0, N_1 \rangle$, where $N_i(u_j, v_j) = \delta_{i,j}$ is a linear spline basis function over the

standard tetrahedron. The values of these two scalar products are

$$\langle N_0, N_0 \rangle = \int_{w=0}^{1} \int_{v=0}^{1-w} \int_{u=0}^{1-v-w} (1-u-v-w)^2 \; dudvdw = \frac{1}{60} \quad \text{and}$$

$$\langle N_0, N_1 \rangle = \int_{w=0}^{1} \int_{v=0}^{1-w} \int_{u=0}^{1-v-w} (1-u-v-w) \; u \; dudvdw = \frac{1}{120}. \tag{21}$$

Remark 5.1. The $d+1$ vertices of the *standard simplex* in d-dimensional space are $(0,0,0,...,0)^T$, $(1,0,0,...,0)^T$, $(0,1,0,...,0)^T$, $(0,0,1,...,0)^T$, ..., and $(0,0,0,...,1)^T$, where each point has d coordinates. One proves by induction that

$$\langle N_0, N_0 \rangle = \frac{2}{(d+2)!} \quad \text{and}$$

$$\langle N_0, N_1 \rangle = \frac{1}{(d+2)!}. \tag{22}$$

This is of interest in the multivariate case.

6. Examples

We have tested our approach for univariate, bivariate, and trivariate test functions. In the univariate case, we start with a single interval, the interval $[0,1]$. In the bivariate case, we have used the unit square $[0,1] \times [0,1]$ and defined the initial triangulation by splitting the square into the two triangles obtained by connecting $(0,0)^T$ and $(1,1)^T$. The initial triangulation in the trivariate case is the unit cube $[0,1] \times [0,1] \times [0,1]$, split into five tetrahedra, see Fig. 6.

Table 1 lists, for bivariate and trivariate test functions, global error estimates[5] and number of knots (in square brackets).

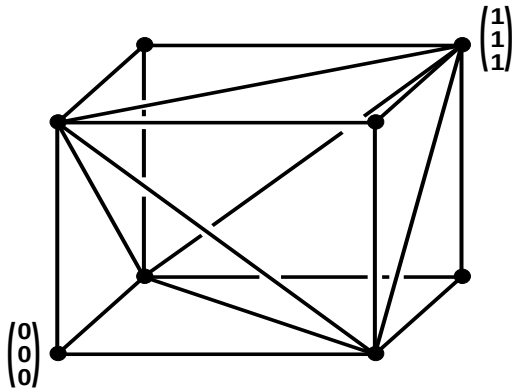

Fig. 6. Initial triangulation in trivariate case: unit cube split into five tetrahedra.

[5] The global error estimates are based on equation (8).

Table 1. Global errors and numbers of knots for approximations of test functions.

x^2+y^2	$10x\left(x-\frac{1}{4}\right)\left(x-\frac{3}{4}\right)y^2$	$x^{10}+y^{10}$	$\frac{1}{2}\sin\left(4\pi x^2\right)\cos\left(2\pi y^2\right)$
0.09428[4]	0.17892[4]	0.19691[4]	0.24133[4]
0.04501[6]	0.07340[8]	0.09896[12]	0.12099[34]
0.02209[10]	0.04321[9]	0.05048[23]	0.05869[75]
0.01109[16]	0.02336[16]	0.02395[37]	0.03304[109]
0.00571[27]	0.01009[38]	0.01117[71]	0.01566[207]
0.00267[52]	0.00574[58]	0.00599[107]	0.00696[391]
0.00136[96]	0.00267[129]	0.00312[217]	0.00370[759]
0.00067[180]	0.00126[237]	0.00156[334]	0.00183[1415]
0.00034[355]	0.00066[485]	0.00079[756]	–
–	0.00033[871]	–	–

$x^2+y^2+z^2$	$10x\left(x-\frac{1}{4}\right)\left(x-\frac{3}{4}\right)y^2\sqrt{z}$	$x^{10}+y^{10}+z^{10}$	$\sin\left(10\pi x^2\right)\cos\left(5\pi y^2\right)\sin\left(2\pi z\right)$
0.0993[8]	0.1374[8]	0.2399[8]	0.2959[374]
0.0508[15]	0.0694[20]	0.1172[31]	0.1924[1855]
0.0252[43]	0.0346[40]	0.0605[126]	0.0961[7048]
0.0169[121]	0.0177[118]	0.0329[257]	–
–	0.0080[358]	0.0161[832	–
–	0.0041[1053]	–	–

$0.4\left(x^2-y^2-z^2\right)$	$e^{-\frac{1}{2}\left(x^2+y^2+z^2\right)}$	Skull	Flame
0.0508[8]	0.0300[8]	0.0919[990]	0.2370[8]
0.0259[18]	0.0164[15]	0.0665[3121]	0.1213[68]
0.0127[38]	0.0100[93]	0.0399[9776]	0.0643[959]
0.0065[107]	–	–	0.0335[5145]
0.0039[518]	–	–	–

Figs. 7–16 show approximations for univariate, bivariate, and trivariate functions (and real-world trivariate data) [6]. Concerning the generation of the triangulation levels in the bivariate and trivariate cases, we have inserted multiple knots in a single refinement step by identifying the approximately 10% triangles/tetrahedra with largest local errors and bisecting them simultaneously, *i.e.*, inserting multiple knots in one step.

In each figure, the image in the upper-left corner is a high-resolution rendering of the original function to be approximated. Univariate functions (Figs. 7 and 8) are rendered as graphs $\big(x, f(x)\big)^T$. (The squares in Figs. 7 and 8 indicate the region $[0,1] \times [-1,1]$.) Bivariate functions (Figs. 9 and 10) are rendered as graphs $\big(x, y, f(x,y)\big)^T$. They are flat-shaded. The underlying triangulations are shown in the xy-plane. (The cubes in Figs. 9–16 indicate the region $[0,1] \times [0,1] \times [0,1]$. The pair of perpendicular line segments shown in some of the figures indicates the origin.) The global error estimate sequences in the figure captions of Figs. 9–14 are based on the equations (2), (4), and (8). (The global error definition for the trivariate case is a straightforward generalization of equation (8).) The numbers in square brackets are the numbers of knots of the specific approximations.

Figs. 11–14 show trivariate examples. We have rendered the trivariate functions by sets of planar slices "cutting" through the underlying tetrahedral meshes. The slices are shaded according to function values. Fig. 13 shows approximations of a computerized axial tomography (CAT) data set of a human skull ($64 \times 64 \times 68 = 278,528$ normalized data on a rectilinear grid), and Fig. 14 shows approximations of a flame data set ($130 \times 80 \times 20 = 208,000$ normalized data on a rectilinear grid). Errors are computed with respect to the original normalized, discrete data sets. Figs. 15 and 16 are renderings of the same data sets used for Figs. 13 and 14. We have used a ray casting algorithm for tetrahedral meshes to generate Figs. 13 and 14.

Remark 6.1. We note that the described hierarchy-generating 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, matrix bandwidth optimization, and solve linear equation systems to generate an approximation hierarchy. We have implemented our algorithm on an SGI Indigo^2 graphics workstation, and the construction of the hierarchies for the most complicated trivariate examples, which consist of several thousand knots at the highest levels of resolution, requires several minutes of computation. We believe that this is acceptable: Interactive visualization of very large data sets is only possible if one has pre-computed a hierarchy prior to the rendering and data analysis steps.

[6] In the case of the real-world trivariate data sets—the skull and the flame data sets—we use a piecewise trilinear interpolant to "evaluate" these scalar fields. Such an analytical definition is needed to compute the required integral expressions involved in the computation of inner products and errors.

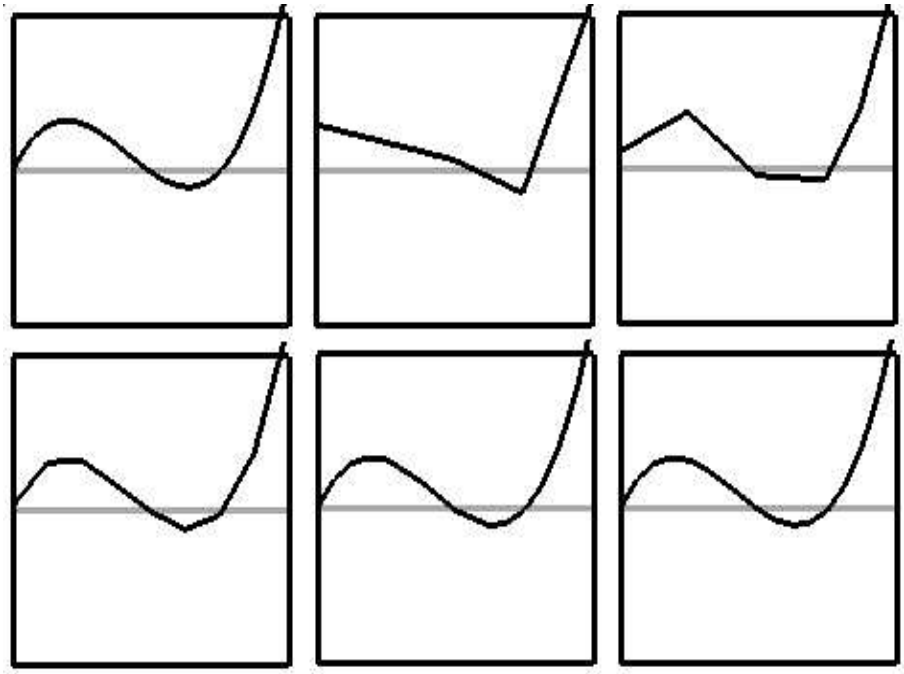

Fig. 7. Five approximations of $F(x) = 10x\big(x - \frac{1}{2}\big)\big(x - \frac{3}{4}\big)$
(no. of knots in approximations: 4, 6, 8, 14, 19).

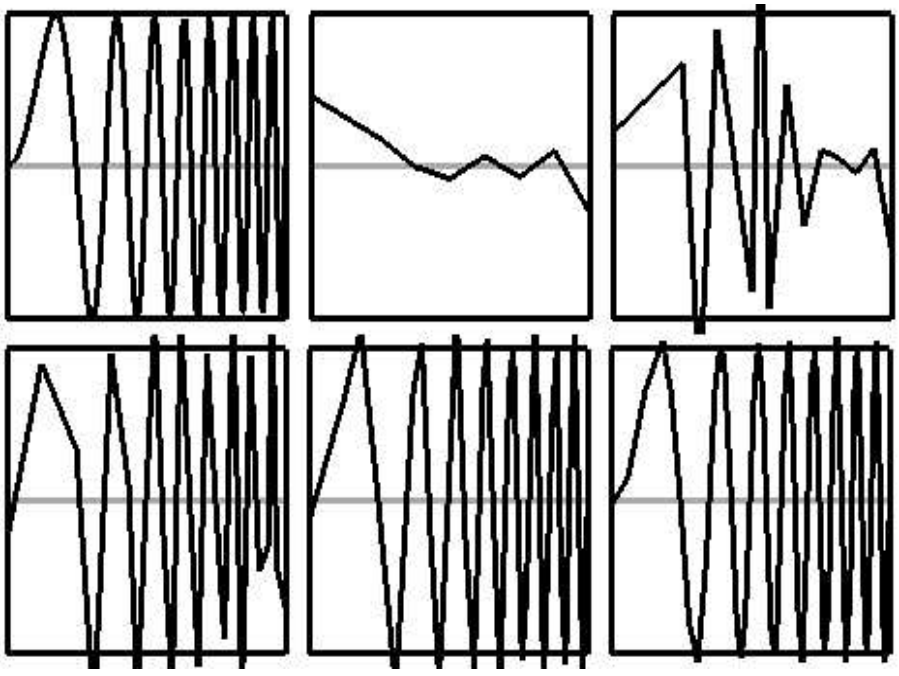

Fig. 8. Five approximations of $F(x) = \sin\big(16\pi x^2\big)$
(no. of knots in approximations: 8, 14, 24, 33, 52).

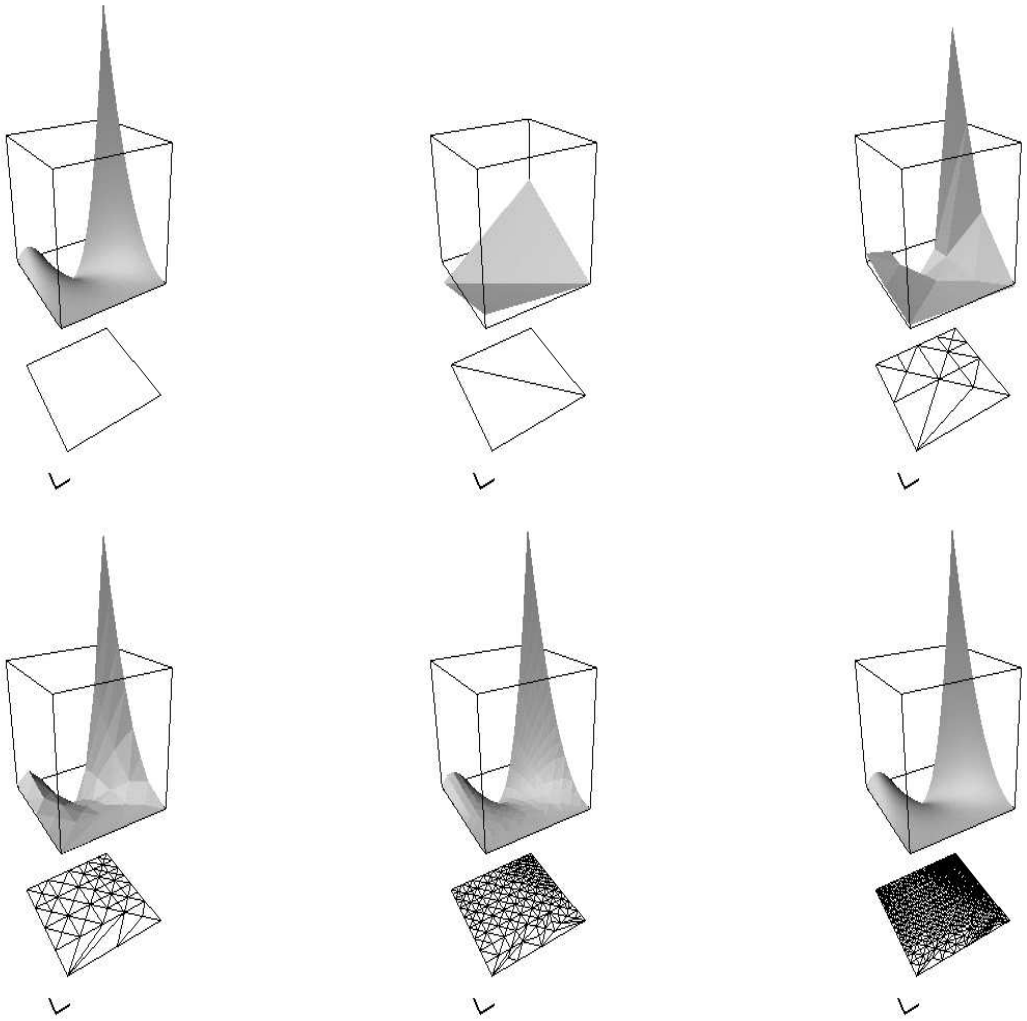

Fig. 9. Five approximations of $F(x,y) = 10x\big(x-\frac{1}{4}\big)\big(x-\frac{3}{4}\big)y^2$
(errors: 0.1789[4], 0.0234[16], 0.0057[58], 0.0011[267], 0.0002[1241]).

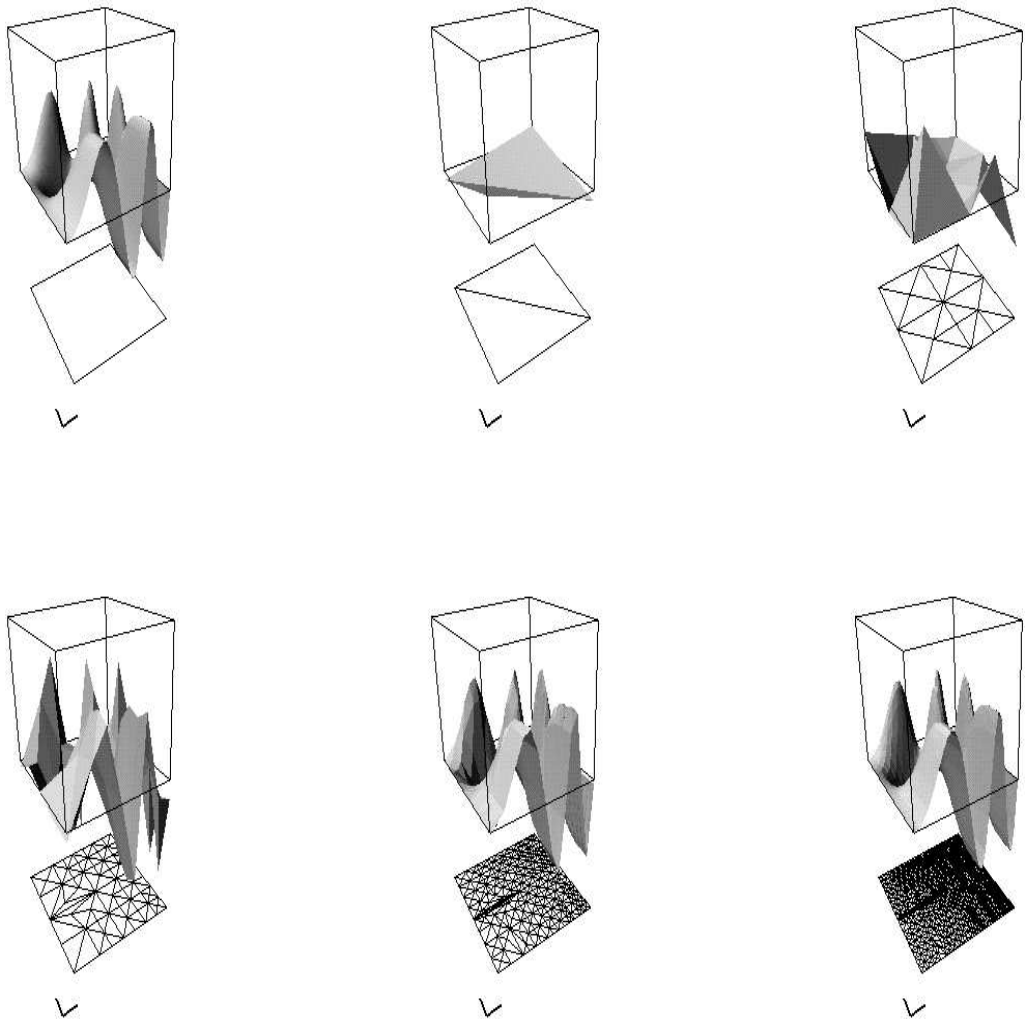

Fig. 10. Five approximations of $F(x,y) = 0.5 \sin\left(4\pi x^2\right) \cos\left(2\pi y^2\right)$ (errors: 0.2413[4], 0.1977[16], 0.0772[61], 0.0116[271], 0.0018[1415]).

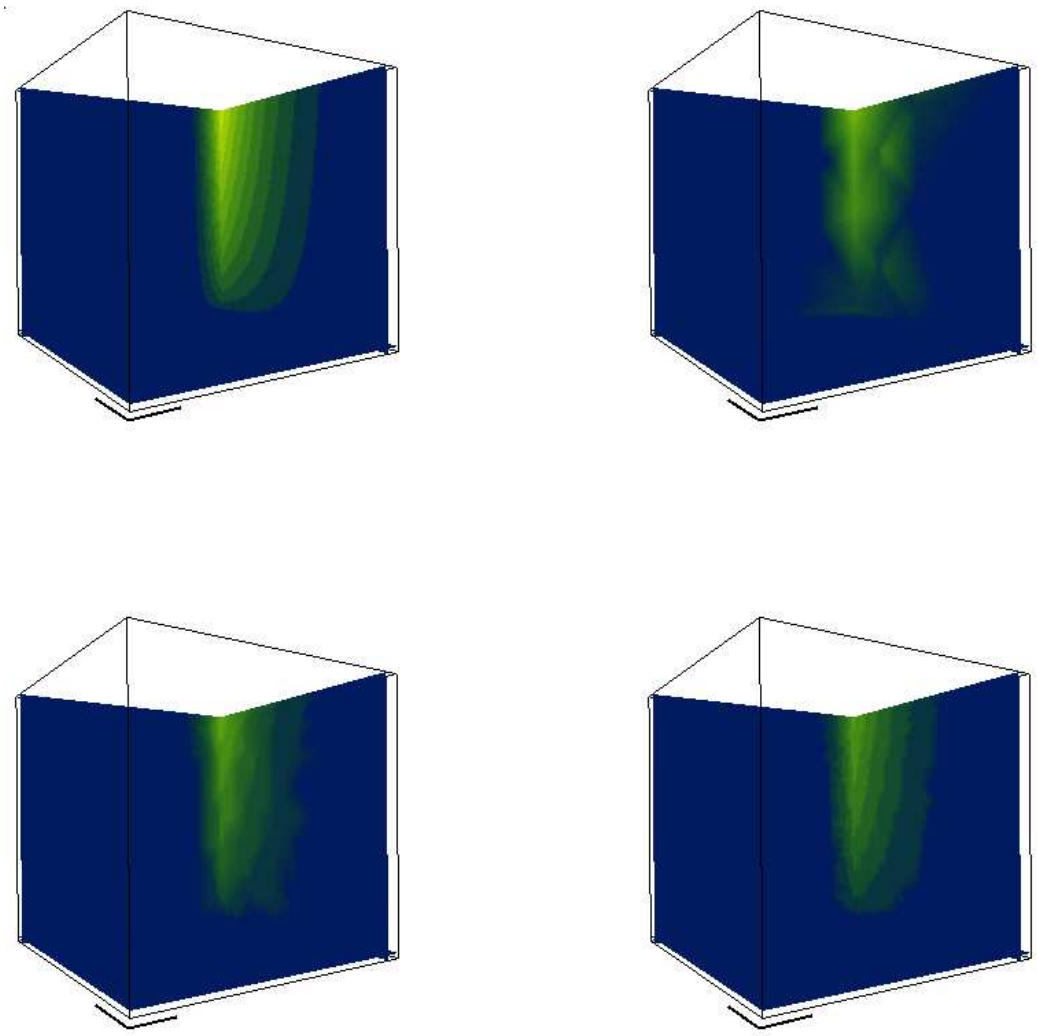

Fig. 11. Three approximations of $F(x,y,z) = 10x\big(x-\frac{1}{4}\big)\big(x-\frac{3}{4}\big)\ y^2\sqrt{z}$ (errors: 0.0346[40], 0.0093[289], 0.0029[2617]).

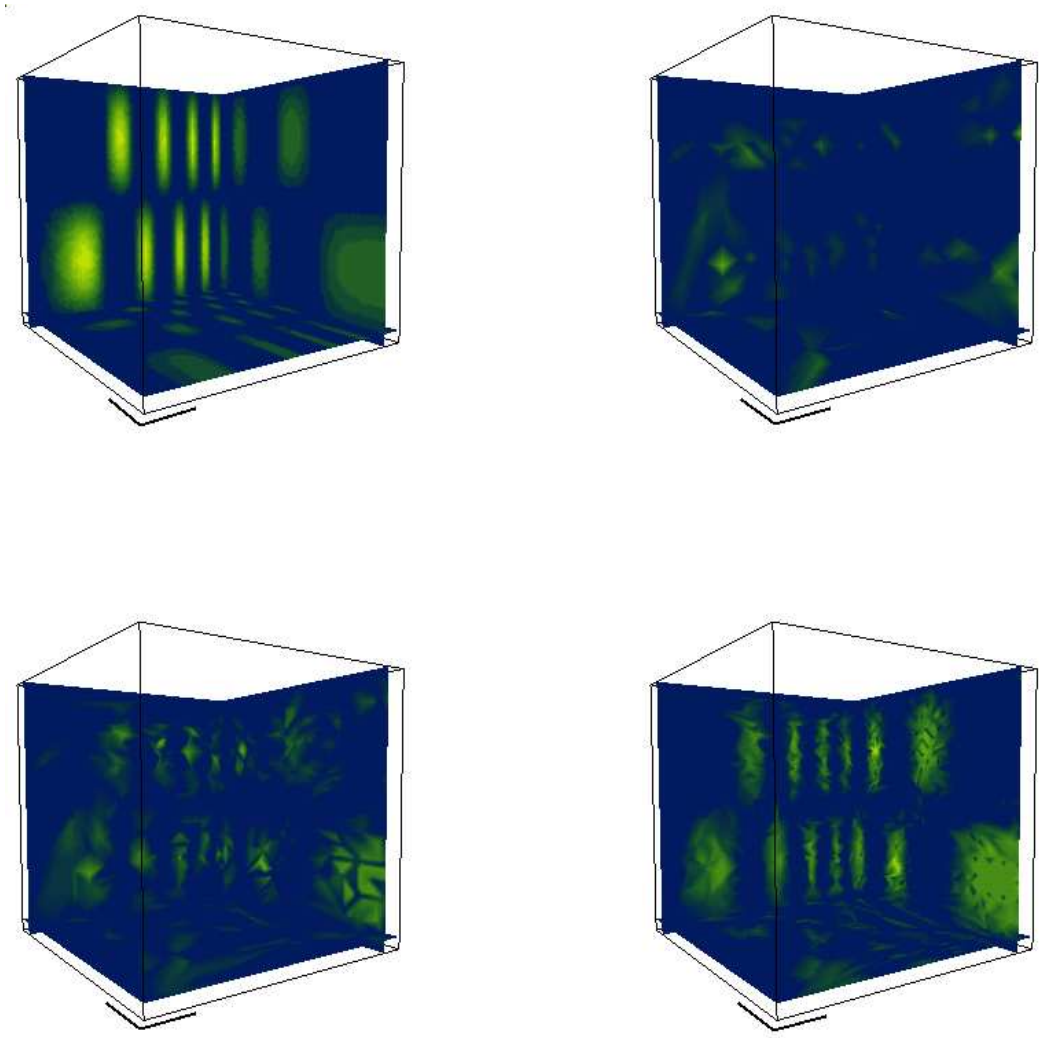

Fig. 12. Three approximations of $F(x,y,z) = \sin\big(10\pi x^2\big)\cos\big(5\pi y^2\big)\sin\big(2\pi z\big)$ (errors: 0.2824[482], 0.1924[1855], 0.0961[7048]).

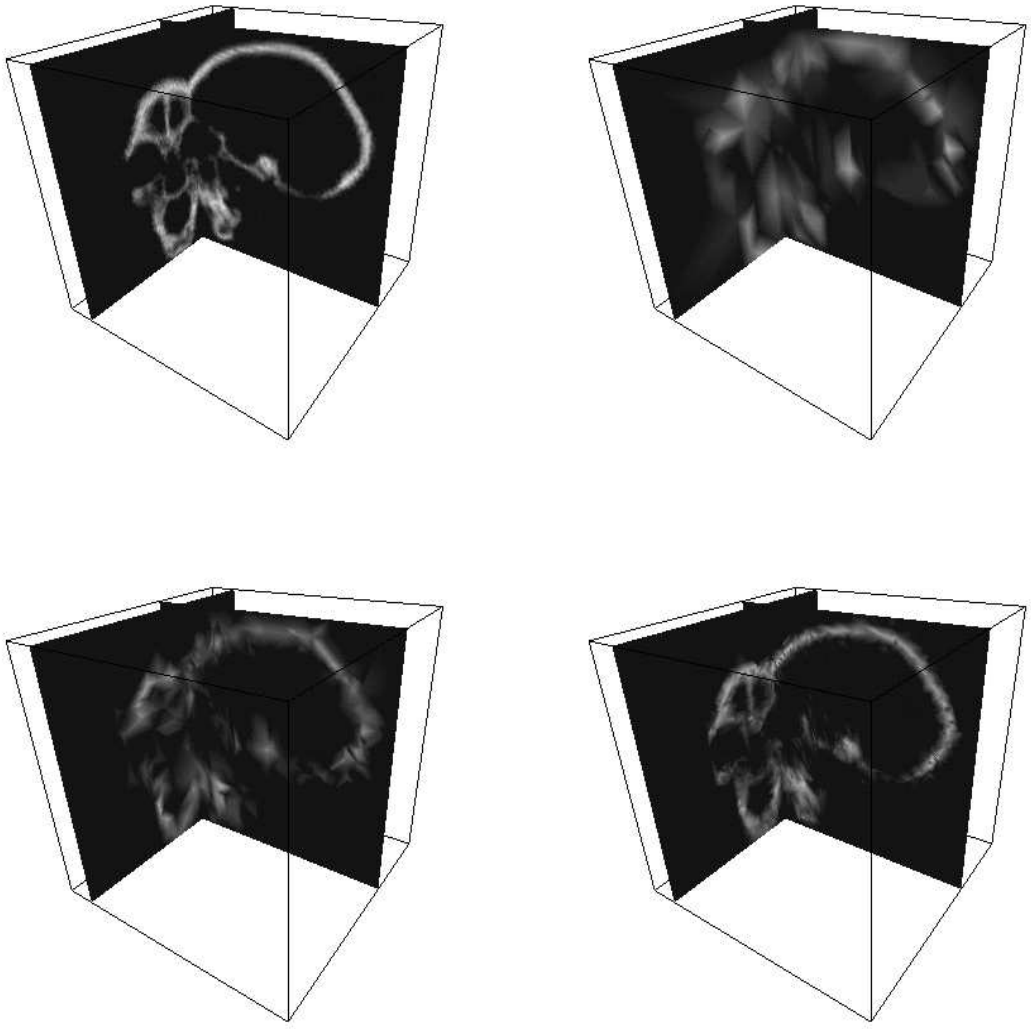

Fig. 13. Three approximations of skull data (orig. no. of vertices: 278,528) (errors: 0.1087[326], 0.0794[1775], 0.0399[9776]).

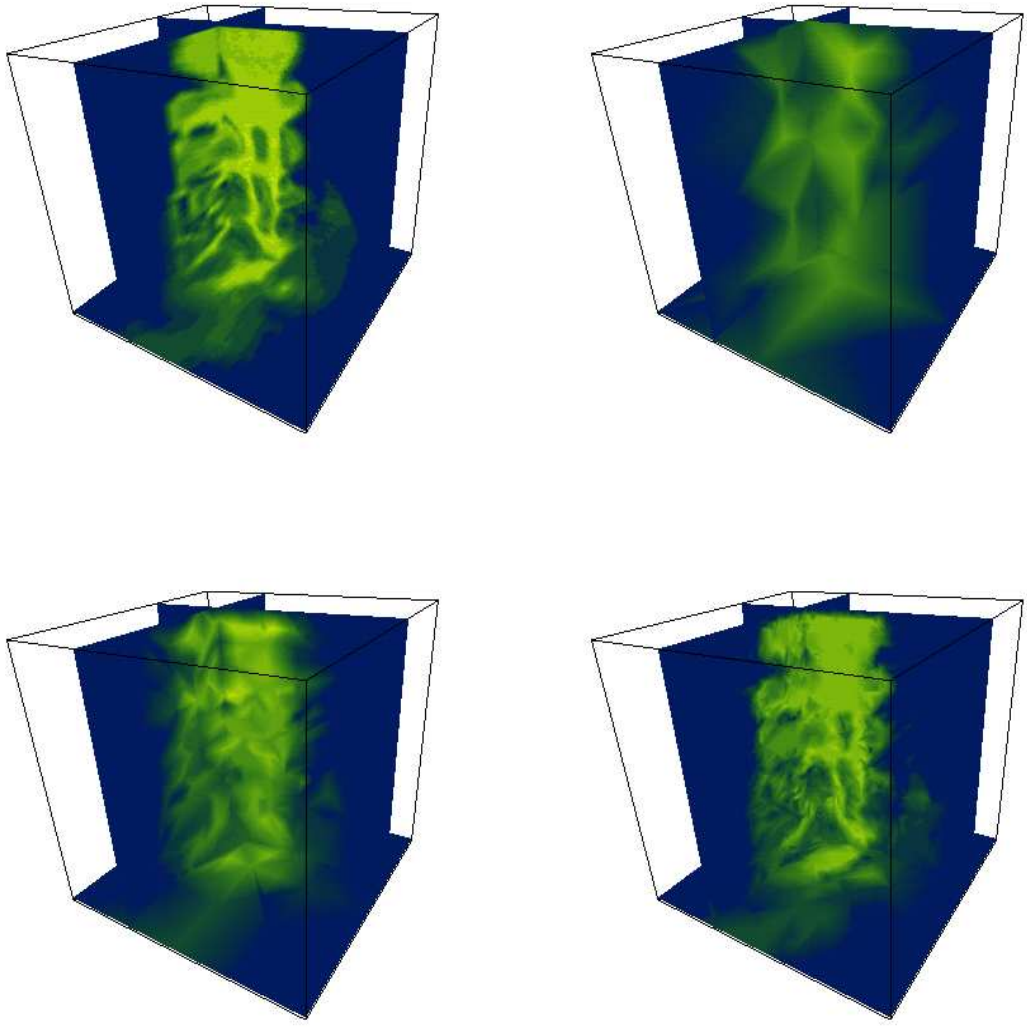

Fig. 14. Three approximations of flame data (orig. no. of vertices: 208,000) (errors: 0.1213[68], 0.0756[514], 0.0335[5145]).

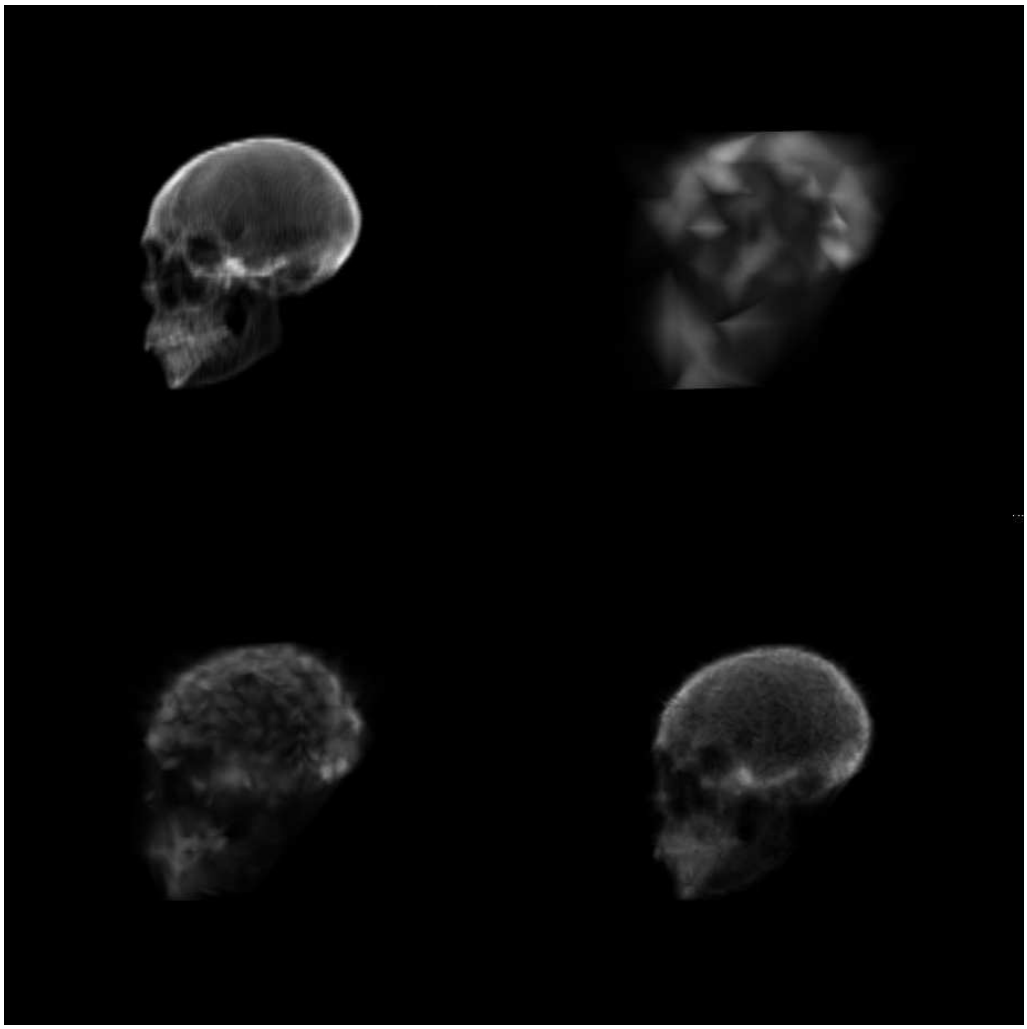

Fig. 15. Ray casting applied to skull data set
(same approximations as used for Fig. 13).

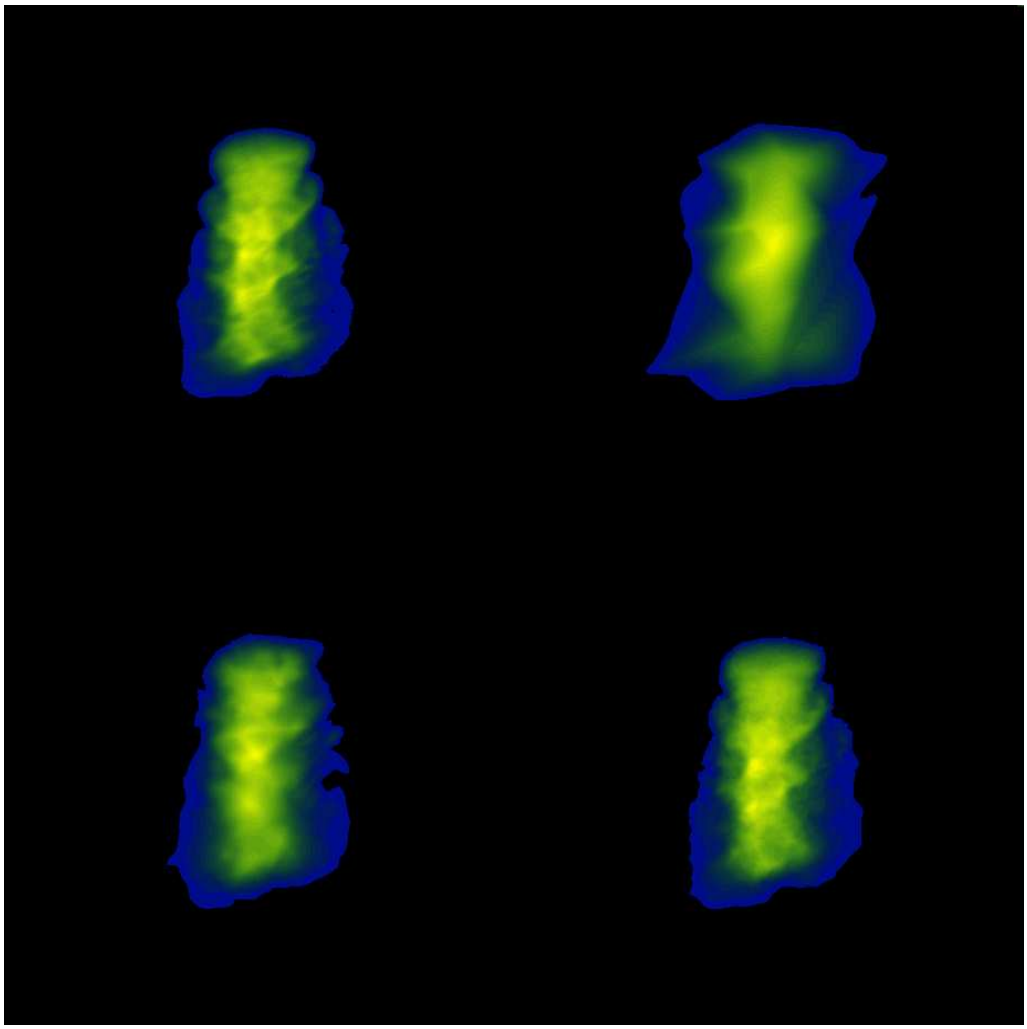

Fig. 16. Ray casting applied to flame data set
(same approximations as used for Fig. 14).

7. Conclusions and future work

We have presented a method for the generation of hierarchies of best linear spline approximations for univariate, bivariate, and trivariate functions using hat functions as spline basis functions in all dimensions. The approach is sound, general, robust, and fairly efficient. One should view the process of constructing a data hierarchy as a pre-processing step for subsequent data visualization. From this point of view the construction of the hierarchy does not necessarily have to be an extremely fast operation. A hierarchy-generating algorithm serves its purpose as long as the resulting data hierarchy provides a compact and simple data format at increasing levels of detail that is usable by visualization systems. This is the case for our method.

We plan to improve the efficiency of our approach and generalize the approach to allow more general subdivision schemes and variable knot placement. Such a generalization would most likely produce very good approximations requiring fewer knots. Furthermore, we plan to extend our approach to multi-valued functions, in particular bivariate and trivariate vector fields. 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. Several numerical methods exist for efficiently solving the normal equations. Such methods, including the conjugate gradient method, take advantage of the band structure of the system of normal equations and its good condition number. We recommend to use such methods when utilizing our scheme for applications of considerable size.

We have only discussed how to generate unstructured mesh hierarchies but not how to store them appropriately in the context of subsequent data visualization and analysis. 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 used. An optimal file format is application-dependent, and one should design it on a case-to-case basis.

8. Acknowledgements

This work was supported by the National Science Foundation under contract ACI 9624034 (CAREER Award), 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 through an ASCI ASAP Level-2 under contract W-7405-ENG-48 (and B335358, B347878), and the North Atlantic Treaty Organization (NATO) under contract CRG.971628 awarded to the University of California, Davis. We also acknowledge the support of Silicon Graphics, Inc., and thank the members of the Visualization Thrust at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis.

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.

Barnhill, R. E. and Little, F. F. (1984), Adaptive triangular cubatures, The Rocky Mountain Journal of Mathematics 14(1), pp. 53–75.

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.

Cantoni, A. (1971), Optimal curve fitting with piecewise linear functions, IEEE Transactions on Computers C-20(1), pp. 59–67.

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.

Cuthill, E. and McKee, J. (1969), Reducing the bandwidth of sparse symmetric matrices, in: *Proceedings of the ACM National Conference*, Association for Computing Machinery, New York, NY, pp. 157–172.

Davis, P. J. (1975), *Interpolation and Approximation*, Dover Publications, Inc., New York, NY.

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.

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

Gibbs, N. E., Poole, W. G. and Stockmeyer P. K. (1976), An algorithm for reducing the bandwidth and profile of a sparse matrix, SIAM J. Numer. Anal. 13(2), pp. 236–250.

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.

Hämmerlin, G. and Hoffmann, K.-H. (1991), *Numerical Mathematics*, Springer-Verlag, New York, NY.

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), Elsevier, 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.

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

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

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.

Rosen, R. (1968), Matrix bandwidth minimization, in: *Proceedings of the ACM National Conference*, ACM publication no. P-68, Brandon Systems Press, Princeton, NJ, pp. 585–595.

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.

Stone, H. (1961), Approximation of curves by line segments, Mathematics of Computation 15, pp. 40–47.

Tomek, I. (1974), Two algorithms for piecewise-linear continuous approximation of functions of one variable, IEEE Transactions on Computers C-23, pp. 445–448.

Trotts, I. J., Hamann, B., Joy, K. I. and Wiley, D. F. (1998), Efficient and robust simplification of tetrahedral meshes, in: Ebert, D. S., Hagen, H. and Rushmeier, H. E., eds., *Visualization '98*, IEEE Computer Society Press, Los Alamitos, CA, 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. (1977), *The Finite-Element Method in Engineering Science*, McGraw-Hill, London, United Kingdom.

APPENDIX

A.1. Romberg integration in the univariate case

For the integration of a function f over the unit interval $[0,1]$ one computes a sequence of trapezoidal sums,

$$A_p \;=\; \frac{1}{2^p}\;\frac{1}{2}\left(f_0 + f_{2^p} + 2\sum_{i=1}^{2^p-1} f_i\right), \qquad p = 0, ..., n, \tag{23}$$

where $f_i = f\left(\frac{i}{2^p}\right)$, and uses this sequence to compute extrapolated, usually better, integral approximations. The sequence of A_p values converges linearly to the exact value of the integral of f defined over the unit interval. (One can expand the sum A_p in terms of the step size $h_p = \frac{1}{2^p}$, and the resulting expansion is of order two.) Having computed the values $A_{0,0} := A_0$, $A_{1,0} := A_1$, $A_{2,0} := A_2$, ..., and $A_{n,0} := A_n$, we compute the extrapolated approximations

$$A_{i,j} \;=\; \frac{A_{i+1,j-1} - 2^{-2j}A_{i,j-1}}{1 - 2^{-2j}}, \qquad j = 1, ..., n, \qquad i = 0, ..., n-j. \tag{24}$$

Initially, we compute the triangular Romberg scheme for $n = 1$, and we increase n one-by-one until the condition $|A_{0,n} - A_{1,n-1}| < \epsilon$ is satisfied. Fig. A.1 illustrates the triangular Romberg scheme resulting from equation (24).

$$\begin{array}{ccccccccccc}
A_{0,0} & \rightarrow & A_{0,1} & \rightarrow & A_{0,2} & \rightarrow & \ldots & A_{0,n-2} & \rightarrow & A_{0,n-1} & \rightarrow & A_{0,n} \\
A_{1,0} & \nearrow & A_{1,1} & \nearrow & \vdots & & & A_{1,n-2} & \nearrow & A_{1,n-1} & \nearrow & \\
A_{2,0} & \nearrow & \vdots & & & & & A_{2,n-2} & \nearrow & & & \\
\vdots & & & & & & & & & & & \\
A_{n,0} & \nearrow & & & & & & & & & &
\end{array}$$

Fig. A.1. Triangular Romberg scheme.

A.2. Romberg integration in the bivariate case

Regarding the computation of the scalar products $\langle F, f_i \rangle$ and error estimates in the bivariate case, one can choose from a large pool of numerical integration schemes. A rather simple yet robust and efficient adaptive triangular cubature scheme is described in [Barnhill & Little '84], and one could use it as an alternative to the bivariate Romberg scheme that we describe in the following. We choose to use Romberg integration, since it generalizes nicely to arbitrary dimensions. Regardless of the chosen numerical method, all cubature methods assume that one knows an analytical definition of the function to be integrated—this can be a C^{-1}-, C^0-, C^1-, ..., or C^∞-continuous definition—and that one can effectively evaluate the function. We briefly describe our Romberg scheme and the construction of a sequence of integral estimates based on linear, triangular elements. We limit our description to the standard triangle.

An initial estimate of the value of the integral $\int f(x,y)\,dxdy$ of some function f defined over the standard triangle is the value

$$A_0 \;=\; \frac{1}{2}\,\frac{1}{3}\,\big(f(0,0)+f(1,0)+f(0,1)\big), \tag{25}$$

which is the area of the standard triangle—having vertices $\mathbf{v}^0_{0,0}=(0,0)^T$, $\mathbf{v}^0_{1,0}=(1,0)^T$, and $\mathbf{v}^0_{0,1}=(0,1)^T$—multiplied by the average of the three function values at the three vertices. One obtains a better estimate by splitting the standard triangle uniformly into four sub-triangles, estimating integral values for the individual sub-triangles, and adding the resulting values. The six vertices defining the vertices of the four sub-triangles are $\mathbf{v}^1_{0,0}=(0,0)^T$, $\mathbf{v}^1_{1,0}=\big(\frac{1}{2},0\big)^T$, $\mathbf{v}^1_{2,0}=(1,0)^T$, $\mathbf{v}^1_{0,1}=\big(0,\frac{1}{2}\big)^T$, $\mathbf{v}^1_{1,1}=\big(\frac{1}{2},\frac{1}{2}\big)^T$, and $\mathbf{v}^1_{0,2}=(0,1)^T$, and the four vertex triples defining the sub-triangles are $(\mathbf{v}^1_{0,0},\mathbf{v}^1_{1,0},\mathbf{v}^1_{0,1})$, $(\mathbf{v}^1_{1,0},\mathbf{v}^1_{2,0},\mathbf{v}^1_{1,1})$, $(\mathbf{v}^1_{0,1},\mathbf{v}^1_{1,1},\mathbf{v}^1_{0,2})$, and $(\mathbf{v}^1_{1,0},\mathbf{v}^1_{1,1},\mathbf{v}^1_{0,1})$. Multiplying the areas of the sub-triangles with the average of the three function values at their respective vertices and adding the individual results yields the approximation

$$A_1 \;=\; \frac{1}{2^2}\,\frac{1}{2}\,\frac{1}{3}\,\big(f_{0,0}+f_{2,0}+f_{0,2}+3f_{1,0}+3f_{0,1}+3f_{1,1}\big), \tag{26}$$

where $f_{i,j}$ is the function value at $\mathbf{v}^1_{i,j}$. The general *level-p approximation* is given by

$$\begin{aligned} A_p \;=\; \frac{1}{2^{2p}}\,\frac{1}{2}\,\frac{1}{3}\,\Bigg(& f_{0,0}+f_{2^p,0}+f_{0,2^p}+3\sum_{i=1}^{2^p-1} f_{i,0} \\ & +3\sum_{j=1}^{2^p-1} f_{0,j}+3\sum_{\substack{i\neq 2^p\,\wedge\, j\neq 2^p \\ \wedge\, i+j=2^p}} f_{i,j}+6\sum_{j=1}^{2^p-2}\;\sum_{i=1}^{2^p-1-j} f_{i,j}\Bigg), \end{aligned} \tag{27}$$

where $f_{i,j}=f\big(\frac{i}{2^p},\frac{j}{2^p}\big)$.

Fig. A.2 shows the vertices and their respective weights needed for the computation of A_p. Formula (27) is a generalization of the trapezoidal sums used in the univariate case. The A_p values are then used to compute integral approximations $A_{i,j}$ using the triangular Romberg scheme.

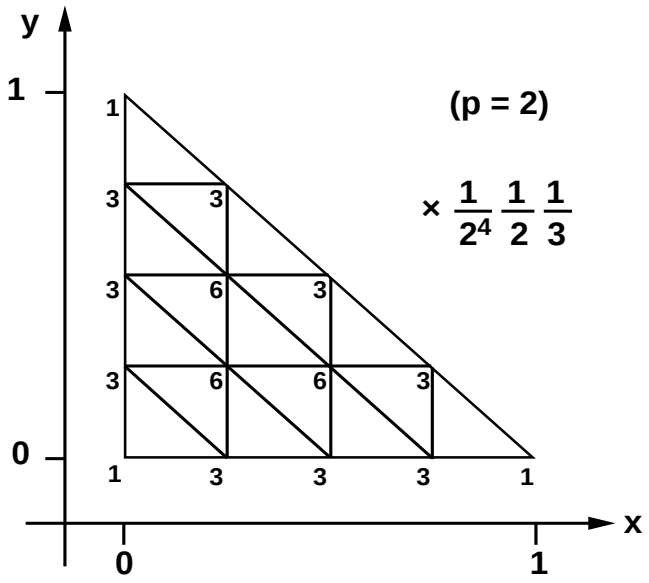

Fig. A.2. Vertices and weights used for approximation of $\int_{y=0}^{1} \int_{x=0}^{1-y} f(x,y)\ dxdy$.

A.3. Romberg integration in the trivariate case

We use Romberg integration for the computation of scalar products $\langle F, f_i \rangle$ and error values. We describe a Romberg integration scheme for the standard tetrahedron. Subdivision of the standard tetrahedron into sub-polyhedra is more complicated than subdividing the standard triangle. An initial estimate of the integral $\int f(x,y,z)\ dxdydz$ of some function f defined over the standard tetrahedron is

$$A_0 \;=\; \frac{1}{6}\,\frac{1}{4}\,\big(f(0,0,0) + f(1,0,0) + f(0,1,0) + f(0,0,1)\big), \tag{28}$$

which is the volume of the standard tetrahedron—having vertices $\mathbf{v}^0_{0,0,0} = (0,0,0)^T$, $\mathbf{v}^0_{1,0,0} = (1,0,0)^T$, $\mathbf{v}^0_{0,1,0} = (0,1,0)^T$, and $\mathbf{v}^0_{0,0,1} = (0,0,1)^T$ —multiplied by the average of the four function values at the four vertices. One obtains a better estimate of the integral by decomposing the standard tetrahedron into four tetrahedra and one octahedron (which is split into four sub-tetrahedra), estimating integral values for the four tetrahedra and the octahedron, and adding the individual results. Our level-1 decomposition of the standard tetrahedron is shown in Fig. A.3.

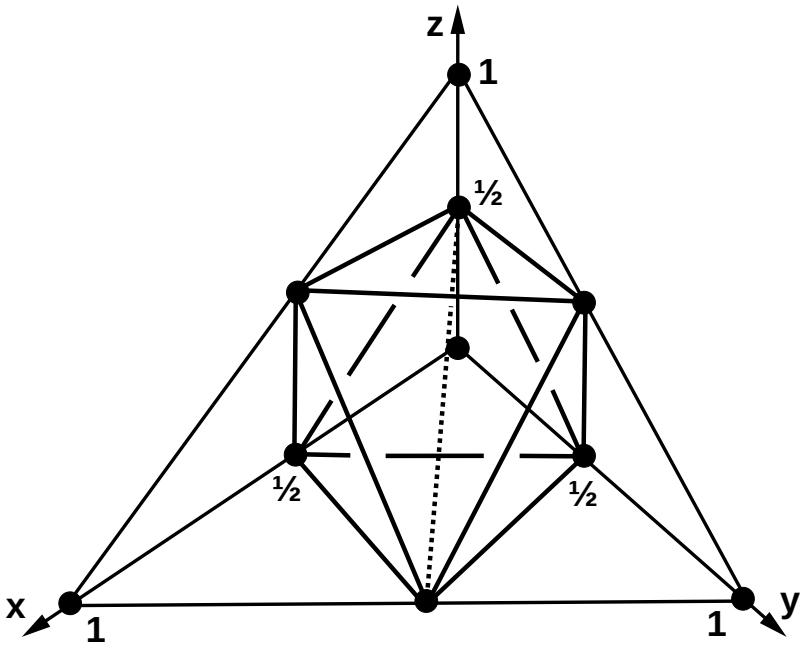

Fig. A.3. Splitting standard tetrahedron into four tetrahedra and one octahedron.

The ten vertices defining the level-1 decomposition are $\mathbf{v}^1_{i,j,k} = \left(\frac{i}{2}, \frac{j}{2}, \frac{k}{2}\right)^T$, $k = 0, ..., 2$, $j = 0, ..., 2-k$, $i = 0, ..., 2-k-j$. The vertex quadruples defining the four tetrahedra are $(\mathbf{v}^1_{0,0,0}, \mathbf{v}^1_{1,0,0}, \mathbf{v}^1_{0,1,0}, \mathbf{v}^1_{0,0,1})$, $(\mathbf{v}^1_{1,0,0}, \mathbf{v}^1_{2,0,0}, \mathbf{v}^1_{1,1,0}, \mathbf{v}^1_{1,0,1})$, $(\mathbf{v}^1_{0,1,0}, \mathbf{v}^1_{1,1,0}, \mathbf{v}^1_{0,2,0}, \mathbf{v}^1_{0,1,1})$, and $(\mathbf{v}^1_{0,0,1}, \mathbf{v}^1_{1,0,1}, \mathbf{v}^1_{0,1,1}, \mathbf{v}^1_{0,0,2})$. The vertex tuple $(\mathbf{v}^1_{1,0,0}, \mathbf{v}^1_{1,1,0}, \mathbf{v}^1_{0,1,0}, \mathbf{v}^1_{0,0,1}, \mathbf{v}^1_{1,0,1}, \mathbf{v}^1_{0,1,1})$ defines the "inner" octahedron. We split the "inner" octahedron into four sub-tetrahedra of equal volume by adding an edge connecting $\mathbf{v}^1_{0,0,1}$ and $\mathbf{v}^1_{1,1,0}$. Multiplying the volumes of the tetrahedra and sub-tetrahedra with the average of the four function values at their respective vertices and adding the individual results yields the approximation

$$A_1 = \frac{1}{2^3}\,\frac{1}{6}\,\frac{1}{4}\left(\sum_{k=0}^{1}\sum_{j=0}^{1-k}\sum_{i=0}^{1-k-j} f_{i,j,k} + f_{i+1,j,k} + f_{i,j+1,k} + f_{i,j,k+1} + 4f_{0,0,1} + 4f_{1,1,0} + 2f_{1,0,0} + 2f_{0,1,0} + 2f_{1,0,1} + 2f_{0,1,1} \right), \quad (29)$$

where $f_{i,j,k}$ is the function value at $\mathbf{v}^1_{i,j,k}$. The general level-p approximation is given by

$$
\begin{aligned}
A_p \;=\; \frac{1}{2^{3p}}\,\frac{1}{6}\,\frac{1}{4}\Bigg(& \sum_{k=0}^{2^p-1}\sum_{j=0}^{2^p-1-k}\sum_{i=0}^{2^p-1-k-j} f_{i,j,k} + f_{i+1,j,k} + f_{i,j+1,k} + f_{i,j,k+1} \\
& + \sum_{k=0}^{2^p-2}\sum_{j=0}^{2^p-2-k}\sum_{i=0}^{2^p-2-k-j} 4f_{i,j,k+1} + 4f_{i+1,j+1,k} + 2f_{i+1,j,k} \\
& \qquad + 2f_{i,j+1,k} + 2f_{i+1,j,k+1} + 2f_{i,j+1,k+1} \\
& + \sum_{k=0}^{2^p-3}\sum_{j=0}^{2^p-3-k}\sum_{i=0}^{2^p-3-k-j} f_{i+1,j+1,k} + f_{i+1,j,k+1} + f_{i,j+1,k+1} + f_{i+1,j+1,k+1}\Bigg),
\end{aligned}
\tag{30}
$$

where $f_{i,j,k} = f\big(\frac{i}{2^p}, \frac{j}{2^p}, \frac{k}{2^p}\big)$.

Fig. A.4 shows a part of our level-p decomposition of the standard tetrahedron into sub-polyhedra—which is just one of many possible decompositions. There are three types of sub-polyhedra: a scaled version of the standard tetrahedron, an octahedron, and a "hole-filling" tetrahedron. The "hole-filling" tetrahedron implies the third term of the sum appearing in equation (30). Once again, these initial integral estimates are extrapolated using the triangular Romberg scheme.

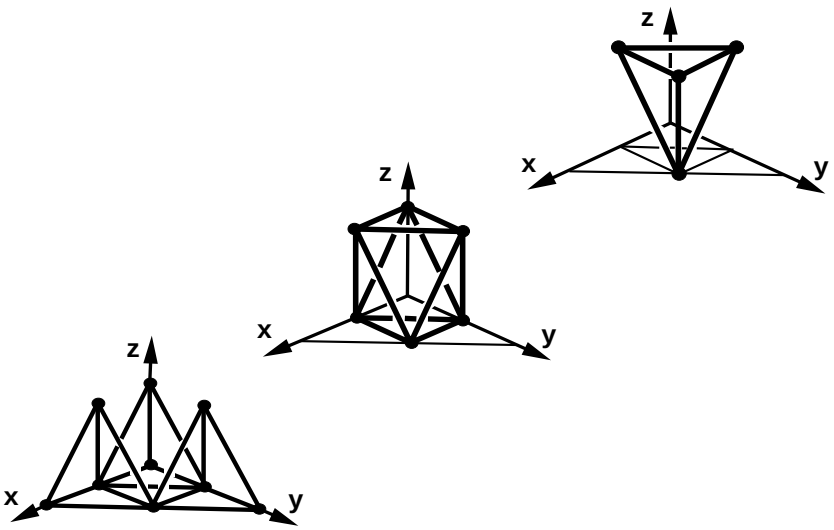

Fig. A.4. Types of polyhedra used for decomposition of standard tetrahedron.

Remark A.1. One could write equation (30) in the form of equation (27) used in the bivariate case. This would require counting the number of tetrahedra and octahedra sharing a particular vertex $\mathbf{v}^p_{i,j,k}$ in the level-p decomposition of the standard tetrahedron.

Bernd Hamann is an Associate Professor of Computer Science and Co-Director of the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis, and an Adjunct Professor of Computer Science at Mississippi State University. He is a Participating Guest researcher at the Lawrence Livermore National Laboratory. From 1991 to 1995 Hamann was a faculty member in the Department of Computer Science at Mississippi State University and a research faculty member at the NSF Engineering Research Center for Computational Field Simulation. His primary interests are visualization, computer-aided geometric design (CAGD), and computer graphics with a focus on multiresolution methods. He is the author or co-author of over 50 reviewed publications and has presented his research at leading conferences in the U.S. and in Europe. Hamann received a B.S. in computer science, a B.S. in mathematics, and an M.S. in computer science from the Technical University of Braunschweig, Germany. He received his Ph.D. in computer science from Arizona State University in 1991. Hamann was awarded a 1992 Research Initiation Award by Mississippi State University, a 1992 Research Initiation Award by the National Science Foundation, and a 1996 CAREER Award by the National Science Foundation. In 1995, he received a Hearin-Hess Distinguished Professorship in Engineering by the College of Engineering, Mississippi State University. Hamann is a member of the Association for Computing Machinery (ACM), the Institute of Electrical and Electronics Engineers (IEEE), and the Society for Industrial and Applied Mathematics (SIAM).

Benjamin W. Jordan is a technical director at Pixar Animation Studios in Richmond, California. Prior to joining Pixar in 1997, he worked as a computer graphics researcher at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis. His primary interests are technical direction for computer generated images in film, computer-aided geometric design (CAGD), rendering, and computer graphics. Ben's list of credits includes: Pixar's Academy Award-winning short film "Geri's Game," Disney's Animal Kingdom attraction "It's Tough to be a Bug," and the feature films "A Bug's Life" and "Toy Story 2." Ben received a B.S. in computer science and engineering from the University of California, Davis, in 1997. Ben is a member of the Association for Computing Machinery (ACM) and the ACM Special Interest Group on Computer Graphics (SIGGRAPH).

David F. Wiley is a Ph.D. student in Computer Science at the University of California, Davis, where he has participated for the past two years in research projects performed at the Center for Image Processing and Integrated Computing (CIPIC). David's primary interests are scientific visualization and computer graphics. David has been working as a program developer for PC software at WinWay Corporation, Sacramento, California, for the past two years. He received a B.S. in computer science from the University of California, Davis, in 1998.