On Curved Simplicial Elements and Best Quadratic Spline Approximation for Hierarchical Data Representation

Bernd Hamann[1]

1 Institute for Data Analysis and Visualization (IDAV)
   Department of Computer Science
   University of California, Davis
   Davis, CA 95616
   hamann@cs.ucdavis.edu

—— **Abstract** ——

We present a method for hierarchical data approximation using curved quadratic simplicial elements for domain decomposition. Scientific data defined over two- or three-dimensional domains typically contain boundaries and discontinuities that are to be preserved and approximated well for data analysis and visualization. Curved simplicial elements make possible a better representation of curved geometry, domain boundaries, and discontinuities than simplicial elements with non-curved edges and faces. We use quadratic basis functions and compute best quadratic simplicial spline approximations that are C^0-continuous everywhere except where field discontinuities occur whose locations we assume to be given. We adaptively refine a simplicial approximation by identifying and bisecting simplicial elements with largest errors. It is possible to store multiple approximation levels of increasing quality. Our method can be used for hierarchical data processing and visualization.

1 Introduction

Scalar and vector field data often contain discontinuities that should be preserved for data approximation and analysis purposes. It is important to represent domain boundaries—including geometry such as a car body, an aircraft, or a ship hull—and the locations of field discontinuities, represented by curves and surfaces. To better approximate these curves and surfaces we investigate the use of curved quadratic simplicial elements. We do not address the problem of extracting discontinuities from a given scalar or vector field data set; we assume that this information is known. We consider data defined over two-dimensional (2D) and three-dimensional (3D) domains.

We utilize only curved simplicial elements that are quadratic. In the 2D case, we use curved triangles whose edges may be straight line segments or parabolae; in the 3D case, we use curved tetrahedral elements whose edges/faces may be straight line segments/planar triangles or curved. Generally, we refer to both non-curved and curved simplicial elements as just simplicial elements. We use a quadratic polynomial transformation to map the so-called *standard simplex* to the corresponding simplicial region in 2D/3D space. Furthermore, we use a quadratic polynomial defined over each simplicial element to locally approximate the dependent

variable(s). We use curved elements with curved edges/faces to better approximate domain boundaries and discontinuities. All simplicial elements that do not "touch" geometry, domain boundaries, or discontinuities are non-deformed elements. Nevertheless, the polynomials we use over all simplicial elements are all quadratic.

Our overall goal is the construction of a hierarchical data over 2D or 3D domains using a best approximation approach based on curved quadratic finite elements and quadratic polynomials defined over these elements. We start with a coarse decomposition of the domain, using a relatively small number of simplicial elements and placing curved simplices in areas where boundaries and discontinuities occur. We then compute a (globally) best least squares approximation, a quadratic spline approximation for the dependent variable(s) that is C^0-continuous. (Due to the C^0 continuity requirement we can place simplices with curved edges/faces only along boundaries and where discontinuities occur, i.e., in areas where the curved edges/faces are not shared by other elements. The physical locations of discontinuities play the same roles as domain boundaries: Two simplicial elements may share the—geometrically—same edge/face defining the locus of a discontinuity, but the field function defined over the two elements is discontinuous along/on the shared edge/face.) Based on local errors that we compute for each simplicial element, we bisect a certain percentage of the elements with largest errors, update the simplicial domain decomposition accordingly, and compute a new best quadratic spline approximation. We iterate this process until a specified error condition is met or the number of simplicial elements exceeds some threshold.

Our approach belongs to the class of *refinement* methods. These methods are based on the principle of refining intermediate data approximations by inserting additional points or elements until a certain termination criterion is satisfied. We have developed our method with a focus on the needs of massive scientific data analysis and visualization, see [22, 39, 45]. To enable interactive frame rates for massive data visualization, for example, it is possible to use low-resolution best approximations everywhere or adaptively insert high-resolution approximations locally into an otherwise relatively coarse approximation. The overall approximation algorithm is based on these steps:

- **Initial simplicial domain decomposition.** Assuming that either a polygonal/polyhedral or an analytical definition is known for all boundaries and discontinuities in the 2D/3D domain of interest, we construct a coarse simplicial decomposition of this domain. We use curved edges/faces only in areas where they are needed to better approximate curved boundaries and/or discontinuities. (The quadratic transformations, mapping the standard simplex defined in so-called *parameter space* to deformed simplices in so-called *physical space*, are defined by specifying corresponding point pairs in the two spaces such that one obtains a one-to-one, bijective mapping.) Figure 1 shows a possible initial simplicial decomposition, including curved elements, of space around a wing cross section.
- **Best approximation.** In the 2D case, each simplicial element has six associated *knots*, one knot per corner and one knot per edge. Six knots in parameter space are associated with six points in physical space, and this defines the needed quadratic mapping for a simplex. (Accordingly, the number of knots is ten in the 3D case.) For simplicity, we consider only knots that are uniformly distributed along the edges of the standard simplex, see Appendix A. We associate a quadratic polynomial with each simplicial element, which approximates the dependent variable(s) over the corresponding region in space. We represent each quadratic basis polynomial in so-called *Bernstein-Bézier form*, see [12, 41]. Assuming that the function to be approximated, a scalar- or vector-valued function, is known in analytical form, it is possible to compute the unique best quadratic spline approximation defined as a linear combination of the set of quadratic basis functions.
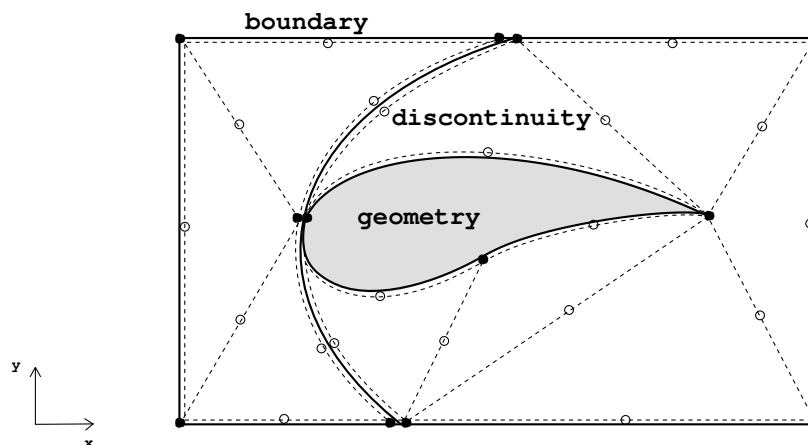
Figure 1 Decomposition of space around wing using curved 2D simplices (geometry, domain boundary, and discontinuity shown in bold).

The best approximation, understood in a least squares sense, is the result of solving the *normal equations*, see [9].

- **Adaptive bisection.** We compute a local error value for each simplicial element once a best approximation is known. We use the L_2 norm to compute simplex-specific error values. The set of simplices is ordered according to the simplex-specific, local error values. To compute a next-level best quadratic approximation we determine a certain percentage of simplices with largest error values and bisect them by splitting them at the mid point of their longest edge. If a simplex' longest edge is not unique, we choose the edge to be split randomly. In the case of curved edges we use arc length to determine the longest edge to be bisected. Splitting a specific simplex into two simplices induces additional splits for all those simplices that share the split edge. We update a simplicial domain decomposition by considering all edge bisections and compute a new best approximation. We repeat the process of identifying simplices with largest errors, bisecting these simplices, and computing a new best approximation until we obtain an approximation for which the maximal simplex-specific error is below a certain error threshold or until a maximal number of simplices is reached.

- **Hierarchical data representation.** To support hierarchical data processing and visualization, for example, we can store multiple best approximations of different simplicial resolutions. For each best approximation, we need to store the polynomial coefficients of each simplicial element—for its shape and the polynomial defined over it. Considering a non-curved simplicial element, we only need to store its three (four) corner points and the coefficients of the quadratic polynomial defined over the element. Considering a curved element, we need to store all polynomial coefficients defining the shape of the element in addition to the coefficients of the quadratic polynomial defined over the element. We store a fixed number of best approximations such that either the number of simplices increases in a specified fashion or the maximal simplex-specific error decreases in a certain way from one resolution to the next.

We discuss these steps in more detail in the sections to follow.

Related work in the areas of hierarchical data representation and approximation is discussed in [1, 4, 5, 6, 7, 11, 16, 17, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 33, 38, 48,
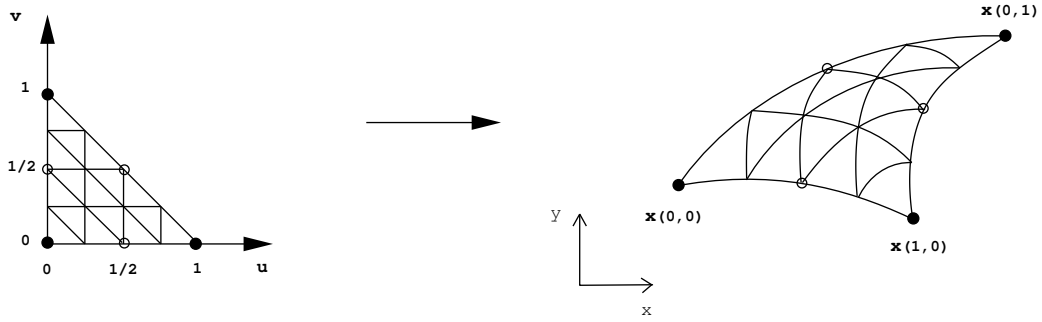
Figure 2 Mapping standard triangle to arbitrary curved triangle (iso-parametric lines shown in parameter and physical space).

51, 52, 59]. Best-approximation methods are described in [53, 55], and effective processing and visualization approaches for data approximated by higher-order elements are covered in [19, 55, 56, 57]. So-called *data-dependent triangulation* schemes, i.e., schemes concerned with the construction of piecewise linear approximations using near-optimal simplicial elements, are described in [10, 36, 43]. In [46, 47] various data structures are covered in-depth that can be used for efficient storage of hierarchical data approximations. From a broader perspective, our work is related to *grid generation*, and references for this area are [14, 32, 49, 50]. Finite element methods, which also are closely related to our work, are discussed in detail in [60].

2   Mapping the Standard Simplex

In the 2D case, the standard simplex in parameter space is the triangle with corners $(0,0)$, $(1,0)$, and $(0,1)$. The triangle with these three corners is mapped to a curved triangular region in physical space by mapping the six knots $\mathbf{u_i} = (u_{i,j}, v_{i,j}) = \left(\frac{i}{2}, \frac{j}{2}\right)$, $i, j \geq 0$, $i + j \leq 2$ (abbreviated in multi-index notation as $|\mathbf{i}| = 2$), in parameter space to six corresponding points $\mathbf{x_i} = (x_{i,j}, y_{i,j})$ in physical space, using a quadratic mapping. The quadratic mapping in the 2D case, using Bernstein-Bézier polynomials $B_{\mathbf{i}}^2(\mathbf{u})$ as basis functions, see [12, 41] and Appendix A, is given by

$$\mathbf{x}(\mathbf{u}) = \begin{pmatrix} x(u,v) \\ y(u,v) \end{pmatrix} = \sum_{|\mathbf{i}|=2} \mathbf{b_i} \, B_{\mathbf{i}}^2(\mathbf{u}) = \begin{pmatrix} \sum_{|\mathbf{i}|=2} c_{i,j} \, B_{i,j}^2(u,v) \\ \sum_{|\mathbf{i}|=2} d_{i,j} \, B_{i,j}^2(u,v) \end{pmatrix}. \tag{1}$$

The mapping between parameter and physical space must be one-to-one. Figure 2 depicts the general mapping of the standard triangle in parameter space to a curved triangle in physical space.

In the same way, we define the mapping of the standard tetrahedron with corners $(0,0,0)$, $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$ to a curved tetrahedron in physical space, mapping the ten knots $\mathbf{u_i} = (u_{i,j,k}, v_{i,j,k}, w_{i,j,k}) = \left(\frac{i}{2}, \frac{j}{2}, \frac{k}{2}\right)$, $|\mathbf{i}| = 2$, to ten corresponding points $\mathbf{x_i} = (x_{i,j,k}, y_{i,j,k}, z_{i,j,k})$ in physical space. Thus, the quadratic mapping in the 3D case is
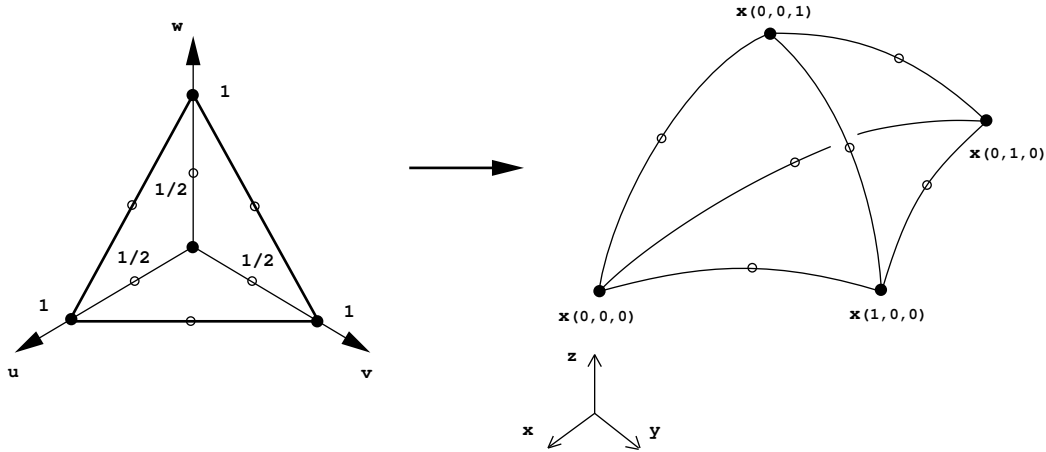
Figure 3 Mapping standard tetrahedron to arbitrary curved tetrahedron.

given by

$$
\mathbf{x}(\mathbf{u}) \;=\; \left(\begin{array}{c} x(u,v,w) \\ y(u,v,w) \\ z(u,v,w) \end{array}\right) \;=\; \sum_{|\mathbf{i}|=2} \mathbf{b_i}\, B_{\mathbf{i}}^2(\mathbf{u}) \;=\; \left(\begin{array}{c} \sum\limits_{|\mathbf{i}|=2} c_{i,j,k}\, B_{i,j,k}^2(u,v,w) \\ \sum\limits_{|\mathbf{i}|=2} d_{i,j,k}\, B_{i,j,k}^2(u,v,w) \\ \sum\limits_{|\mathbf{i}|=2} e_{i,j,k}\, B_{i,j,k}^2(u,v,w) \end{array}\right). \tag{2}
$$

Figure 3 depicts the general mapping of the standard tetrahedron to a curved tetrahedron.

We use quadratic Bernstein-Bézier polynomials as basis functions for the approximation of a field function defined over non-curved simplicial elements as well. We denote these quadratic basis functions as $B_{\mathbf{i}}^2(\mathbf{x}) = B_{i,j}^2(x,y)$ ($= B_{i,j,k}^2(x,y,z)$ in the 3D case). A generalization of the standard Bernstein-Bézier polynomials is necessary for curved simplicial elements. We define the needed generalized quadratic basis polynomials for curved elements in Appendix A. Figure 4 shows the graph of a quadratic polynomial defined over its associated curved triangular domain.

3 Initial Simplicial Domain Decomposition

The main objective driving the development of our method is the hierarchical representation of very large scientific data sets enabling real-time and adaptive data processing and visualization. Data sets resulting from computational simulations are typically defined on a grid, and the dependent variables are associated with either the vertices, also called *nodes* in the finite element literature, or the elements defining the grid. We assume that a data set is provided on a high-resolution grid. The original grid, its boundaries, and possibly known locations of field discontinuities (in the dependent variables) influence how we define an initial simplicial decomposition of the relevant 2D/3D domain.

The objective is to initially represent the 2D/3D domain with a relatively small number of curved simplicial elements, using curved elements only where they help to better approximate domain boundaries and known field discontinuities. In the 2D case, the grid points discretizing the domain boundary represent curves, while they represent surfaces in the 3D setting. For practical purposes we proceed as follows: First, we compute the bounding box of the original set of grid points and decompose this box into two (five) non-curved triangles (tetrahedra).
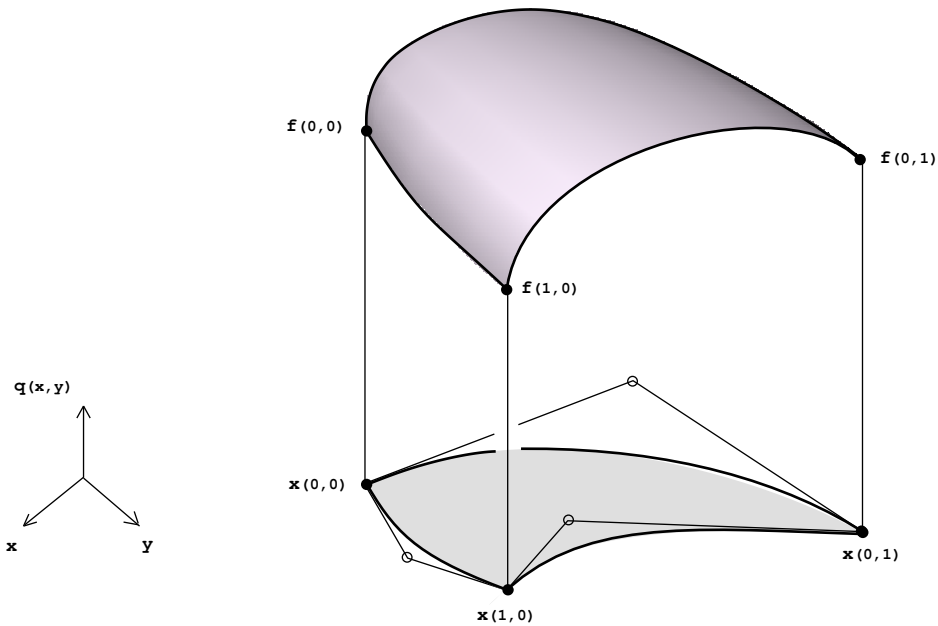
■ **Figure 4** Graph of quadratic polynomial over its curved simplicial domain (Bernstein-Bézier control net shown for curved domain simplex).

Second, we clip these non-curved simplices against the curves (surfaces) defining the domain boundaries. Third, we identify the portion of the initial two (five) simplices that lies inside the domain over which the dependent variable(s) must be approximated; we represent this portion by using initially non-curved simplices only.

We consider perpendicular distance values to determine the quality of a simplicial domain decomposition. We compute the distances of the original grid boundary points from the boundary edges (faces) of the initially non-curved (boundary) simplices. If these distance values are larger than a certain threshold, we must solve a local optimization problem, i.e., we deform an edge/face of a non-curved simplex in a quadratic fashion such that the original grid points in the affected areas are (nearly) optimally approximated by quadratic curves (surfaces). We can solve this problem locally as a univariate (bivariate) approximation problem by considering the distances of original grid points in normal direction of the associated edges/faces of the simplicial boundary elements. We note that the construction of a globally optimal boundary curve/surface approximation is a subject in its own right, but it is not the focus of this paper. We continue our discussion by assuming that boundary approximation schemes suitable for incorporation into our overall scheme are available.

Geometry and domain boundaries can be identified easily from an originally supplied grid. One simply needs to identify elements with edges/faces that are not shared by other elements. It is much harder to identify the locations in 2D/3D space where field discontinuities, i.e., discontinuities of the functions describing the dependent variables, occur. Such discontinuities should be preserved in a simplicial approximation as much as possible. Such discontinuities, once there locations are known, can be treated by curved simplicial elements just like domain boundaries.
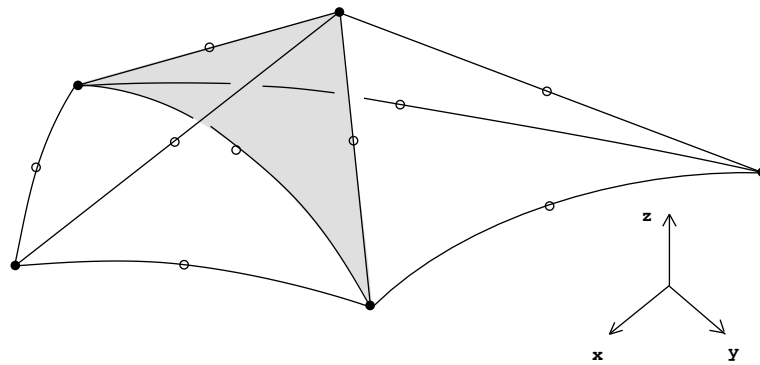
■ **Figure 5** Shared face of two simplicial elements in 3D space is planar but may have curved edges.

The detection of discontinuities of scalar-valued functions over 2D/3D domains has been an active research area in several disciplines, including digital image analysis, pattern recognition/feature extraction from satellite imagery, and scientific data visualization. We refer to the methods described in [2, 40] for more detail. For our purposes, we assume that discontinuities can effectively be extracted from a given data set and that curves/surfaces are used to represent them in the 2D/3D domain. Topologically, we treat these curves (surfaces) in the same way as we treat boundary curves (surfaces) by using simplices with curved edges/faces where they touch these discontinuities. Thus, every discontinuity is approached from two sides, and the simplicial elements touching a discontinuity do not share vertices. (The geometrical information of vertices of these elements *is* shared by vertices along/on field discontinuities, but the coefficients used for field function approximation are different.) Once an initial decomposition of the domain is constructed, we compute the implied best quadratic spline approximation, which we describe in the next section.

In the 3D case, we ensure that each face that is shared by two simplices is planar. Nevertheless, certain edges of a shared planar face may be curved whenever these edges belong to a simplex face that approximates the domain boundary geometry (surface) or a discontinuity in the 3D domain. This situation is illustrated in Figure 5.

4 Best Approximation

We assume that the field/function to be approximated over a 2D/3D domain is known analytically. Should this not be the case, e.g., in the case of *scattered data* (randomly distributed points with associated function values but without connectivity information), it is always possible to construct an analytical representation by performing a prior data interpolation or approximation step, see [13, 37]. In the case that a data set is defined on a grid, the required analytical definition is given by a piecewise linear function for a simplicial (triangular, tetrahedral) grid and a piecewise bilinear/trilinear function in the case of quadrilateral/hexahedral grid cells. (We assume that function values are associated with grid vertices.) We denote the analytical function to be approximated over the domain by $F(\mathbf{x})$. Based on an initial simplicial domain decomposition, we compute the corresponding best piecewise quadratic approximation of $F(\mathbf{x})$ by solving the normal equations, see [9]. The normal equations determine the set of coefficients for the desired quadratic spline representation, a best approximation in the least squares sense.

Corner vertices of simplicial elements may be shared by any number of simplices, and we
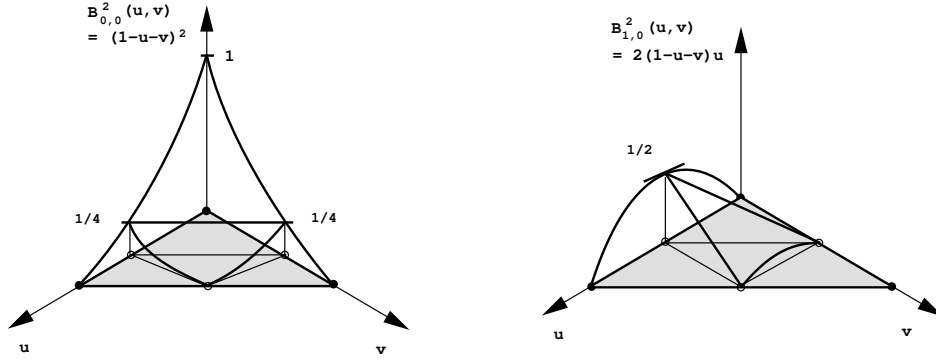
■ **Figure 6** Types of basis functions. Basis function associated with shared corner (left) and shared edge (right).

denote the basis function that we associate with a corner vertex $\mathbf{v}_i$ by $f_i(\mathbf{x})$. An edge of a simplicial element may be shared by no more than two simplices in the 2D case and by an arbitrary number of simplices in the 3D case. We denote a basis function that we associate with a simplex edge e_j by $g_j(\mathbf{x})$. We refer to the set of simplices sharing a common corner vertex as the *platelet* of this corner, and we call the set of simplices sharing a common edge *edge neighbors*. Thus, a set of platelet simplices defines the region in space over which a basis function associated with the corresponding corner vertex is non-zero. Edge neighbors, associated with a particular edge, define the region in space over which a basis function associated with this edge is non-zero. Instead of providing a formal definition for these two types of basis functions, we refer to Figure 6 that depicts the types for the bivariate case.

We denote a best approximation as $a(\mathbf{x})$, and we write it as a linear combination of the basis functions associated with all distinct simplex corners and simplex edges. Assuming that there are m distinct corners and n distinct edges, we can write a best approximation as

$$a(\mathbf{x}) \;=\; \sum_{i=1}^{m} c_i\; f_i(\mathbf{x}) + \sum_{j=1}^{n} d_j\; g_j(\mathbf{x}). \tag{3}$$

We must solve the normal equations to obtain the unknown coefficients c_i and d_j. In matrix form, the normal equations are

$$\begin{pmatrix} \langle f_1, f_1 \rangle & \cdots & \langle f_1, f_m \rangle & \langle f_1, g_1 \rangle & \cdots & \langle f_1, g_n \rangle \\ \vdots & & & & & \vdots \\ \langle f_m, f_1 \rangle & \cdots & \langle f_m, f_m \rangle & \langle f_m, g_1 \rangle & \cdots & \langle f_m, g_n \rangle \\ \langle g_1, f_1 \rangle & \cdots & \langle g_1, f_m \rangle & \langle g_1, g_1 \rangle & \cdots & \langle g_1, g_n \rangle \\ \vdots & & & & & \vdots \\ \langle g_n, f_1 \rangle & \cdots & \langle g_n, f_m \rangle & \langle g_n, g_1 \rangle & \cdots & \langle g_n, g_n \rangle \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_m \\ d_1 \\ \vdots \\ d_n \end{pmatrix} = \begin{pmatrix} \langle F, f_1 \rangle \\ \vdots \\ \langle F, f_m \rangle \\ \langle F, g_1 \rangle \\ \vdots \\ \langle F, g_n \rangle \end{pmatrix}, \tag{4}$$

where $\langle G, H \rangle$ denotes the inner product of the two functions G and H, i.e.,

$$\langle G, H \rangle \;=\; \int\limits_{\text{common domain of } G \text{ and } H} G(\mathbf{x})\; H(\mathbf{x})\; d\mathbf{x}. \tag{5}$$

We must compute inner products involving curved and non-curved simplices. Since all simplicial elements in physical space are defined as quadratic (or linear) mappings of the

standard simplex, we can simplify integration by making use of the *change-of-variables theorem*, see [34], which relates integration in physical space to integration in parameter space for parametrically defined regions. In the 2D case, integrals are computed according to the formula

$$\int\limits_{\text{curved simplex}} G(x,y) \ dx \ dy \ = \ \int\limits_{\text{standard simplex}} G\Big(x(u,v), y(u,v)\Big) \ J(u,v) \ du \ dv, \qquad (6)$$

where $J(u,v)$ denotes the *Jacobian* associated with the mapping of the standard simplex to the corresponding simplex in physical space. The Jacobian is the determinant

$$J(u,v) \ = \ |\mathbf{x_u}| \ = \ \begin{vmatrix} x_u & x_v \\ y_u & y_v \end{vmatrix} \ = \ \begin{vmatrix} \frac{\partial}{\partial u} x(u,v) & \frac{\partial}{\partial v} x(u,v) \\ \frac{\partial}{\partial u} y(u,v) & \frac{\partial}{\partial v} y(u,v) \end{vmatrix}. \qquad (7)$$

(The 3D case is a straightforward extension.) When using a simple linear transformation to map the parameter space knots $\mathbf{u_i}$ to associated physical space points $\mathbf{x_i}$, the Jacobian has a constant value C. This constant value is given by the determinant

$$C \ = \ \Big| \ \mathbf{d}_{2,0} \quad \mathbf{d}_{0,2} \ \Big| \qquad (8)$$

in the 2D case and

$$C \ = \ \Big| \ \mathbf{d}_{2,0,0} \quad \mathbf{d}_{0,2,0} \quad \mathbf{d}_{0,0,2} \ \Big| \qquad (9)$$

in the 3D case, where the column vectors $\mathbf{d_i}$ are given by $\mathbf{d_i} = \mathbf{x_i} - \mathbf{x_0}$.

The matrices resulting for the best approximation problems for different levels of simplicial resolution are sparse, and several methods exist for bandwidth reduction, efficient factorization, and inversion of such sparse matrices, see [8, 15, 18, 42, 44]. Matrix bandwidth is related to the indexing scheme used for the set of basis functions, i.e., the indexing used for simplex corners and simplex edges. We apply a bandwidth reduction step prior to matrix factorization/inversion.

The computation of the inner products appearing in the normal equations requires multi-dimensional integration over non-curved and curved simplicial elements. While the change-of-variables theorem reduces this integration to integration over the standard simplex, we still need to perform numerical integration for the calculation of the inner products appearing on the right-hand side of the normal equations, i.e., integrals of the types $\langle F, f_i \rangle$ and $\langle F, g_j \rangle$, since $F(\mathbf{x})$ can be any integrable function. We use *Romberg integration* for the computation of these right-hand-side inner products, see [3, 27]. Appendix B lists some of the needed inner product values for quadratic Bernstein-Bézier polynomials.

Once we have computed a best approximation for a particular simplicial domain decomposition, we analyze the local approximation quality to identify those simplices that should be refined (bisected) to further improve approximation quality. In the following section, we discuss the general principles used for adaptive bisection.

5   Adaptive Bisection

For each simplicial element in a particular domain decomposition, we compute a local approximation error. We define this error as

$$E(S_i) \ = \ \int\limits_{\text{curved simplex } S_i} \Big(F(\mathbf{x}) - a(\mathbf{x})\Big)^2 \ d\mathbf{x}. \qquad (10)$$
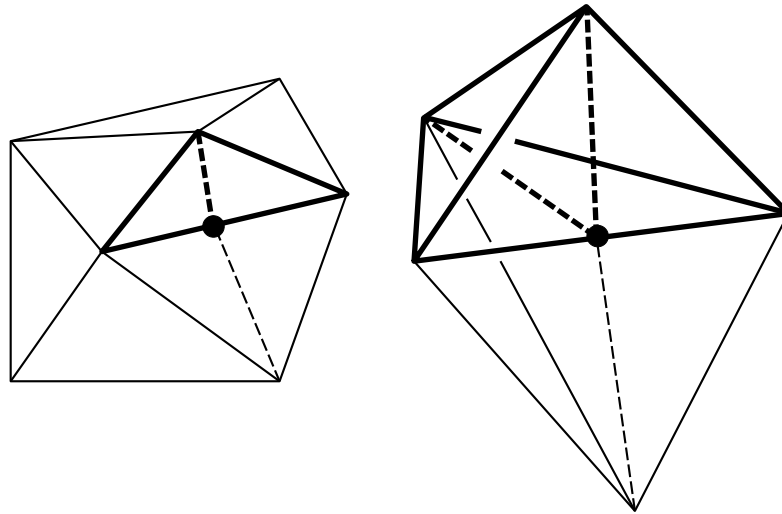
Figure 7 Bisection of simplices in bivariate and trivariate cases.

We order the set $\{S_i\}$ of simplicial elements in decreasing order of their associated error values $E(S_i)$. To construct a new, refined best approximation we specify a percentage of simplices to be bisected and choose the simplices with largest approximation errors.

We bisect a simplicial element marked for refinement by identifying an edge of maximal length, using arc length in the case of curved edges, and split this element by using the midpoint of the split edge as a new simplicial corner vertex. The bisection step is shown in Figure 7. All simplices sharing the split edge are bisected as well to avoid so-called hanging nodes and thus preserve a *conforming mesh*. The bisection steps lead to a new simplicial domain decomposition, and we must compute a new best quadratic spline approximation.

We continue to bisect a certain percentage of simplices in the resulting intermediate simplicial domain decompositions until either the number of simplices in a decomposition exceeds some threshold or an approximation is obtained whose maximal simplex-specific error value is smaller than some tolerance. In principle, it is possible to store all intermediate best quadratic spline approximations in addition to the originally supplied data, possibly including a grid. For practical purposes, this might not always be possible due to storage limitations. Therefore, the number of different best approximations that one stores usually depends on the original resolution of a given data set and its underlying grid, the "complexity" of a given analytical field function, the amount of storage available, and the criterion used to terminate adaptive bisection. The final result of our method is a set of independent best quadratic spline approximations to be used for the purposes of real-time, adaptive, or hierarchical analysis and visualization.

6   Data Visualization Issues

Our data approximation method based on curved simplicial elements must also be considered in the context of visualization techniques applied to data sets defined over 2D and 3D domains. In the case of scalar-valued data sets, the particularly relevant visualization approaches to be considered are (i) extraction and visualization of isocurves/isosurfaces or *contours*; (ii)

slicing the data domain with lines/planes (*slicing lines/planes*); and (iii) *ray casting*, see [19, 55, 56]. Applying these types of visualization techniques to curved simplicial elements over which the dependent variable varies in a quadratic fashion requires us to generalize standard visualization methods that often can deal only with elements with planar faces and linearly or trilinearly varying dependent scalar value.

It is reasonable to view an approximation consisting of curved quadratic simplicial elements to be competitive with a representation consisting of only non-curved, linear simplicial elements when the higher-degree polynomial representation can be rendered nearly as efficiently as the linear one. To study the competitiveness of the piecewise quadratic approximation scheme one must compare rendering efficiency and simplicity for two types of approximation: a piecewise quadratic approximation based on a combination of non-curved and curved simplices and a piecewise linear approximation based on only non-curved simplices. In order to compare two approximation schemes properly, one must require that their respective overall approximation errors are nearly the same.

The application of slicing methods, contouring techniques, and ray casting to non-curved quadratic elements is done routinely. As discussed in [58], for example, the intersection of a ray with an isosurface inside a non-curved 3D simplicial quadratic element, for example, reduces to solving a univariate quadratic equation. The *volume rendering integral* along a ray segment, see [35], is generally too complex to be integrated in closed form, and it is therefore computed numerically. A cut plane intersects a non-curved simplicial element in a polygon on which the scalar field is a quadratic function. Quadratic texture coordinates can be computed in software, or in hardware by taking advantage of texture look-up tables.

Visualization of curved simplicial elements is much more difficult. A quadratic mapping from parameter to physical space must be inverted prior to evaluating a scalar field function at a point in physical space. In the case of curved tetrahedral elements, this requires one to solve three quadratic equations in three variables simultaneously, which can be done with numerical techniques. One could require that a tetrahedral face shared by two tetrahedra is planar, and thus it would be possible to define the field function directly in terms of physical space. The construction of the necessary basis functions for this case is described in Appendix A. Similar problems arise when intersecting a ray with an isosurface or a curved simplex face. We intend to investigate in the future how to render curved simplices directly by solving the involved algebraic equations most elegantly and most efficiently.

A simple solution is to subdivide a curved simplex adaptively, depending on a view, and approximate a curved simplex by non-curved simplices resulting from a properly chosen subdivision scheme. One can replace edge endpoints with the respective edge midpoints. A reasonable criterion to use when deciding when to terminate the subdivision process could be based on the image-space projected maximal deviation of the (union of) the non-curved simplices from their curved "parent" simplex, to allude to just one possibility. Since we represent curved simplicial elements in Bernstein-Bézier form, one could also apply subdivision techniques used in computer-aided geometric design, see [12]. An algorithm like the one described in in [58] could then be applied to the set of non-curved simplices, having quadratic variation only in scalar value.

7 Conclusions

We have described a method for the construction of hierarchical approximations of functions over 2D and 3D domains. The method uses curved simplicial elements to represent the finite domain of a function to be approximated and constructs a best piecewise quadratic

approximation in the least squares sense. Curved simplicial elements are promising in the context of approximating complicated 2D or 3D domains and the dependent functions defined over these domains. Such higher-order elements allow one to construct approximations with relatively smaller error when compared to lower-order and non-curved elements.

8   Acknowledgments

References

1 Bajaj, C. L, Pascucci, V. and Zhuang, G. (1999), Progressive compression and transmission of arbitrary triangular meshes, in: Gross, M., Ebert, D. S. and Hamann, B., eds., *Visualization '99*, IEEE Computer Society Press, Los Alamitos, California, pp. 307–316.

2 Ballard, D. H. and Brown, C. M. (1982), *Computer Vision*, Prentice-Hall, Inc., Englewood Cliffs, NJ.

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

4 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.

5 Bonneau, G. P. (1999), Optimal triangular Haar bases for spherical data, in: Gross, M., Ebert, D. S. and Hamann, B., eds., *Visualization '99*, IEEE Computer Society Press, Los Alamitos, California, pp. 279–284.

6 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.

7 Cohen-Or, D., Levin, D. and Remez, O. (1999), Progressive compression of arbitrary triangular meshes, in: Gross, M., Ebert, D. S. and Hamann, B., eds., *Visualization '99*, IEEE Computer Society Press, Los Alamitos, California, pp. 67–72.

8 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.

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

10 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.

11 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.

12 Farin, G. (2001), *Curves and Surfaces for CAGD: A Practical Guide*, fifth edition, Morgan Kaufmann Publishers, Inc., San Francisco, CA.

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

14 George, P. L. (1991), *Automatic Mesh Generation*, Wiley & Sons, New York, NY.

15 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.

16 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.

17 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.

18 Golub, G. H. and Van Loan, C. F. (1989), *Matrix Computations*, second edition, Johns Hopkins University Press, Baltimore, MD.

19 Gregorski, B. F., Wiley, D. F, Childs, H. R., Hamann, B. and Joy, K. I. (2006), Adaptive contouring with quadratic tetrahedra, in: Bonneau, G.-P., Ertl, T. and Nielson, G. M., eds., *Scientific Visualization: The Visual Extraction of Knowledge from Data*, Springer-Verlag, Heidelberg, Germany, pp. 3–15.

20 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.

21 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.

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

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

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

25 Hamann, B. and Chen, J. L. (1994b), Data point selection for piecewise trilinear approximation, Computer Aided Geometric Design 11(5), pp. 477–489.

26 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.

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

28 Heckel, B., Weber, G. H., Hamann, B. and Joy, K. I. (1999), Construction of vector field hierarchies, in: Gross, M., Ebert, D. S. and Hamann, B., eds., *Visualization '99*, IEEE Computer Society Press, Los Alamitos, California, pp. 19–25.

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

30 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.

31 Hoppe, H. (1999), New quadric metric for simplifying meshes with appearance attributes, in: Gross, M., Ebert, D. S. and Hamann, B., eds., *Visualization '99*, IEEE Computer Society Press, Los Alamitos, California, pp. 59–66.

32 Knupp, P. M. and Steinberg, S. (1993), *Fundamentals of Grid Generation*, CRC Press, Boca Raton, FL.

33 Kreylos, O. and Hamann, B. (1999), On simulated annealing and the construction of linear spline approximations for scattered data, in: Gröller, E., Löffelmann, H. and Ribarsky, W., eds., *Data Visualization '99*, Springer-Verlag, Vienna, Austria, pp. 189–198.

34   Marsden, J. E. and Tromba, A. J. (1988), *Vector Calculus*, third edition, W. H. Freeman and Company, New York, NY.

35   Max, N. L. (1995), Optical models for direct volume rendering, IEEE Transactions on Visualization and Computer Graphics 1(2), pp. 99–108.

36   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.

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

38   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.

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

40   Pavlidis, T. (1980), *Structural Pattern Recognition*, second printing, Springer-Verlag, New York, NY.

41   Piegl, L. A. and Tiller, W. (1996), *The NURBS Book*, second edition, Springer-Verlag, New York, NY.

42   Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (1992), *Numerical Recipes in C*, second edition, Cambridge University Press, New York, NY.

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

44   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.

45   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.

46   Samet, H. (1990), *The Design and Analysis of Spatial Data Structures*, Addison Wesley, New York, NY.

47   Samet, H. (2006), *Foundations of Multidimensional and Metric Data Structures*, Elsevier B. V., Amsterdam, The Netherlands.

48   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.

49   Thompson, J. F., Warsi, Z. U. A. and Mastin, C. W. (1985), *Numerical Grid Generation*, North-Holland, New York, NY.

50   Thompson, J. F., Soni, B. K. and Weatherill, N. P., eds. (1999), *Handbook of Grid Generation*, CRC Press, Boca Raton, FL.

51   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.

52   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.

53   Wiley, D. F., Bertram, M. and Hamann, B. (2004), On a construction of a hierarchy of best linear spline approximations using a finite element approach, IEEE Transactions on Visualization and Computer Graphics 10(5), pp. 548–563.

54   Wiley, D. F., Bertram, M., Jordan, B. W., Hamann, B., Joy, K. I., Max, N. L. and Scheuermann, G. (2003), Hierarchical spline approximation, in: Farin, G., Hamann, B. and

Hagen, H., eds., *Hierarchical and Geometrical Methods in Scientific Visualization*, Springer-Verlag, Heidelberg, Germany, pp. 63–88.

55   Wiley, D. F., Childs, H. R., Gregorski, B. F., Hamann, B. and Joy, K. I. (2003), Contouring curved quadratic elements, in: Bonneau, G.-P., Hahmann, S. and Hansen, C. D., eds., *Data Visualisation 2003*, Eurographics Association, Aire-la-Ville, Switzerland, pp. 167–176.

56   Wiley, D. F., Childs, H. R., Hamann, B. and Joy, K. I. (2004), Ray casting curved-quadratic elements, in: Deussen, O., Hansen, C. D., Keim, D. A. and Saupe, D., eds., *Data Visualization 2004*, Eurographics Association, Aire-la-Ville, Switzerland, pp. 201–209.

57   Wiley, D. F., Childs, H. R., Hamann, B., Joy, K. I. and Max, N. L. (2002), Best quadratic spline approximation for hierarchical visualization, in: Ebert, D. S., Brunet, P. and Navazo, I., eds., *Data Visualisation 2002*, Eurographics Association, Aire-la-Ville, Switzerland, pp. 133–140.

58   Williams, P. L., Max, N. L. and Stein, C. M. (1998), A high accuracy volume renderer for unstructured data, IEEE Transactions on Visualization and Computer Graphics 4(1), pp. 37–54.

59   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.

60   Zienkiewicz, O. C. and Taylor, R. L. (2006), *The Finite Element Method*, sixth edition, Elsevier B. V., Amsterdam, The Netherlands.

A   Quadratic Basis Polynomials

Our method requires quadratic basis polynomials for non-curved and curved simplicial elements. We review the definition of the standard Bernstein-Bézier polynomials used for non-curved elements before generalizing these polynomials for curved elements.

The quadratic Bernstein-Bézier polynomial basis functions, defined for the standard simplex in parameter space, are

$$B_{\mathbf{i}}^2(\mathbf{u}) \ = \ \frac{2!}{(2-i-j)! \, i! \, j!} \, (1-u-v)^{2-i-j} \, u^i \, v^j \tag{11}$$

in the bivariate case and

$$B_{\mathbf{i}}^2(\mathbf{u}) \ = \ \frac{2!}{(2-i-j-k)! \, i! \, j! \, k!} \, (1-u-v-w)^{2-i-j-k} \, u^i \, v^j \, w^k \tag{12}$$

in the trivariate case. Through a linear parameter transformation we can evaluate these quadratic polynomials over all non-curved simplices in physical space. Figure 8 illustrates the graphs of two quadratic Bernstein-Bézier basis functions in the 2D case over the standard triangle.

We must define quadratic basis polynomials for field function approximation over curved simplices in a different way. These more general polynomials are not the result of applying a simple linear parameter transformation. We define quadratic basis polynomials for curved simplices in a way such that they are a generalization of the standard Bernstein-Bézier polynomials for non-curved simplices and guarantee continuity in function value along/on the shared edges/faces of all simplices. We define these generalized quadratic basis polynomials in physical space: In the 2D case, it is possible to think of a set of six simplex-specific quadratic basis polynomials, denoted as $\{Q_{i,j}(x,y)\}$, as a set of six quadratic polynomials satisfying certain interpolation conditions. We specify interpolation conditions at points $(x_{k,l}, y_{k,l})$ that are distributed uniformly with respect to arc length along the edges of a curved simplex.
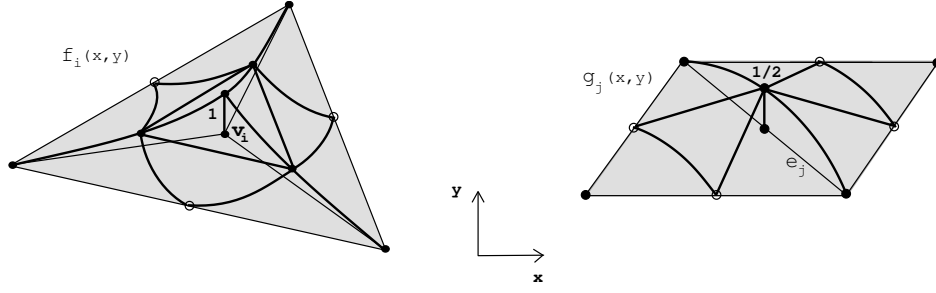
Figure 8 Graphs of quadratic Bernstein-Bézier basis polynomials.

Using the short-hand notation $Q_{i,j}^{k,l}$ for $Q_{i,j}(x_{k,l}, y_{k,l})$, the interpolation conditions, when written in matrix form, are given by

$$
\begin{pmatrix}
Q_{0,0}^{0,0} & Q_{0,0}^{1,0} & Q_{0,0}^{2,0} & Q_{0,0}^{0,1} & Q_{0,0}^{1,1} & Q_{0,0}^{0,2} \\
Q_{1,0}^{0,0} & Q_{1,0}^{1,0} & Q_{1,0}^{2,0} & Q_{1,0}^{0,1} & Q_{1,0}^{1,1} & Q_{1,0}^{0,2} \\
Q_{2,0}^{0,0} & Q_{2,0}^{1,0} & Q_{2,0}^{2,0} & Q_{2,0}^{0,1} & Q_{2,0}^{1,1} & Q_{2,0}^{0,2} \\
Q_{0,1}^{0,0} & Q_{0,1}^{1,0} & Q_{0,1}^{2,0} & Q_{0,1}^{0,1} & Q_{0,1}^{1,1} & Q_{0,1}^{0,2} \\
Q_{1,1}^{0,0} & Q_{1,1}^{1,0} & Q_{1,1}^{2,0} & Q_{1,1}^{0,1} & Q_{1,1}^{1,1} & Q_{1,1}^{0,2} \\
Q_{0,2}^{0,0} & Q_{0,2}^{1,0} & Q_{0,2}^{2,0} & Q_{0,2}^{0,1} & Q_{0,2}^{1,1} & Q_{0,2}^{0,2}
\end{pmatrix}
=
\frac{1}{4}
\begin{pmatrix}
4 & 1 & 0 & 1 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 & 0 \\
0 & 1 & 4 & 0 & 1 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 1 & 1 & 4
\end{pmatrix}. \tag{13}
$$

These interpolation conditions lead to the standard Bernstein-Bézier polynomials when applied to a non-curved simplex. The construction of the basis polynomials in the 3D case is based on the same principle.

Each generalized quadratic basis polynomial is zero outside the particular simplex for which it is defined. The quadratic basis polynomials associated with curved simplices are not as easily constructed and evaluated as those associated with non-curved simplices. Nevertheless, once the generalized quadratic basis polynomials are determined for all curved simplices, we can still compute inner products involving them by applying the change-of-variables theorem.

B   Inner Products of Basis Polynomials

In the following, we define some of the required values of inner products of quadratic polynomials. We only consider the case of these polynomials being defined over the standard simplex in parameter space. For the quadratic Bernstein-Bézier basis polynomials $B_{\mathbf{i}}^2(\mathbf{u})$ defined for knots spaced uniformly along the edges of the standard simplex one obtains these values for inner products $I_{i,j}^{k,l} = \langle B_{i,j}, B_{k,l} \rangle$ in the 2D case:

$$
\begin{pmatrix}
I_{0,0}^{0,0} & I_{0,0}^{1,0} & I_{0,0}^{2,0} & I_{0,0}^{0,1} & I_{0,0}^{1,1} & I_{0,0}^{0,2} \\
I_{1,0}^{0,0} & I_{1,0}^{1,0} & I_{1,0}^{2,0} & I_{1,0}^{0,1} & I_{1,0}^{1,1} & I_{1,0}^{0,2} \\
I_{2,0}^{0,0} & I_{2,0}^{1,0} & I_{2,0}^{2,0} & I_{2,0}^{0,1} & I_{2,0}^{1,1} & I_{2,0}^{0,2} \\
I_{0,1}^{0,0} & I_{0,1}^{1,0} & I_{0,1}^{2,0} & I_{0,1}^{0,1} & I_{0,1}^{1,1} & I_{0,1}^{0,2} \\
I_{1,1}^{0,0} & I_{1,1}^{1,0} & I_{1,1}^{2,0} & I_{1,1}^{0,1} & I_{1,1}^{1,1} & I_{1,1}^{0,2} \\
I_{0,2}^{0,0} & I_{0,2}^{1,0} & I_{0,2}^{2,0} & I_{0,2}^{0,1} & I_{0,2}^{1,1} & I_{0,2}^{0,2}
\end{pmatrix}
=
\frac{1}{180}
\begin{pmatrix}
6 & 3 & 1 & 3 & 1 & 1 \\
3 & 4 & 3 & 2 & 2 & 1 \\
1 & 3 & 6 & 1 & 3 & 1 \\
3 & 2 & 1 & 4 & 2 & 3 \\
1 & 2 & 3 & 2 & 4 & 3 \\
1 & 1 & 1 & 3 & 3 & 6
\end{pmatrix}. \tag{14}
$$

In the 3D case, the needed values of inner products $I_{i,j,k}^{l,m,n} = \langle B_{i,j,k}, B_{l,m,n} \rangle$ are

$$
\begin{pmatrix}
I_{0,0,0}^{0,0,0} & I_{0,0,0}^{1,0,0} & I_{0,0,0}^{2,0,0} & I_{0,0,0}^{0,1,0} & I_{0,0,0}^{1,1,0} & I_{0,0,0}^{0,2,0} & I_{0,0,0}^{0,0,1} & I_{0,0,0}^{1,0,1} & I_{0,0,0}^{0,1,1} & I_{0,0,0}^{0,0,2} \\
I_{1,0,0}^{0,0,0} & I_{1,0,0}^{1,0,0} & I_{1,0,0}^{2,0,0} & I_{1,0,0}^{0,1,0} & I_{1,0,0}^{1,1,0} & I_{1,0,0}^{0,2,0} & I_{1,0,0}^{0,0,1} & I_{1,0,0}^{1,0,1} & I_{1,0,0}^{0,1,1} & I_{1,0,0}^{0,0,2} \\
I_{2,0,0}^{0,0,0} & I_{2,0,0}^{1,0,0} & I_{2,0,0}^{2,0,0} & I_{2,0,0}^{0,1,0} & I_{2,0,0}^{1,1,0} & I_{2,0,0}^{0,2,0} & I_{2,0,0}^{0,0,1} & I_{2,0,0}^{1,0,1} & I_{2,0,0}^{0,1,1} & I_{2,0,0}^{0,0,2} \\
I_{0,1,0}^{0,0,0} & I_{0,1,0}^{1,0,0} & I_{0,1,0}^{2,0,0} & I_{0,1,0}^{0,1,0} & I_{0,1,0}^{1,1,0} & I_{0,1,0}^{0,2,0} & I_{0,1,0}^{0,0,1} & I_{0,1,0}^{1,0,1} & I_{0,1,0}^{0,1,1} & I_{0,1,0}^{0,0,2} \\
I_{1,1,0}^{0,0,0} & I_{1,1,0}^{1,0,0} & I_{1,1,0}^{2,0,0} & I_{1,1,0}^{0,1,0} & I_{1,1,0}^{1,1,0} & I_{1,1,0}^{0,2,0} & I_{1,1,0}^{0,0,1} & I_{1,1,0}^{1,0,1} & I_{1,1,0}^{0,1,1} & I_{1,1,0}^{0,0,2} \\
I_{0,2,0}^{0,0,0} & I_{0,2,0}^{1,0,0} & I_{0,2,0}^{2,0,0} & I_{0,2,0}^{0,1,0} & I_{0,2,0}^{1,1,0} & I_{0,2,0}^{0,2,0} & I_{0,2,0}^{0,0,1} & I_{0,2,0}^{1,0,1} & I_{0,2,0}^{0,1,1} & I_{0,2,0}^{0,0,2} \\
I_{0,0,1}^{0,0,0} & I_{0,0,1}^{1,0,0} & I_{0,0,1}^{2,0,0} & I_{0,0,1}^{0,1,0} & I_{0,0,1}^{1,1,0} & I_{0,0,1}^{0,2,0} & I_{0,0,1}^{0,0,1} & I_{0,0,1}^{1,0,1} & I_{0,0,1}^{0,1,1} & I_{0,0,1}^{0,0,2} \\
I_{1,0,1}^{0,0,0} & I_{1,0,1}^{1,0,0} & I_{1,0,1}^{2,0,0} & I_{1,0,1}^{0,1,0} & I_{1,0,1}^{1,1,0} & I_{1,0,1}^{0,2,0} & I_{1,0,1}^{0,0,1} & I_{1,0,1}^{1,0,1} & I_{1,0,1}^{0,1,1} & I_{1,0,1}^{0,0,2} \\
I_{0,1,1}^{0,0,0} & I_{0,1,1}^{1,0,0} & I_{0,1,1}^{2,0,0} & I_{0,1,1}^{0,1,0} & I_{0,1,1}^{1,1,0} & I_{0,1,1}^{0,2,0} & I_{0,1,1}^{0,0,1} & I_{0,1,1}^{1,0,1} & I_{0,1,1}^{0,1,1} & I_{0,1,1}^{0,0,2} \\
I_{0,0,2}^{0,0,0} & I_{0,0,2}^{1,0,0} & I_{0,0,2}^{2,0,0} & I_{0,0,2}^{0,1,0} & I_{0,0,2}^{1,1,0} & I_{0,0,2}^{0,2,0} & I_{0,0,2}^{0,0,1} & I_{0,0,2}^{1,0,1} & I_{0,0,2}^{0,1,1} & I_{0,0,2}^{0,0,2}
\end{pmatrix}
$$

$$
= \frac{1}{1260}
\begin{pmatrix}
6 & 3 & 1 & 3 & 1 & 1 & 3 & 1 & 1 & 1 \\
3 & 4 & 3 & 2 & 2 & 1 & 2 & 2 & 1 & 1 \\
1 & 3 & 6 & 1 & 3 & 1 & 1 & 3 & 1 & 1 \\
3 & 2 & 1 & 4 & 2 & 3 & 2 & 1 & 2 & 1 \\
1 & 2 & 3 & 2 & 4 & 3 & 1 & 2 & 2 & 1 \\
1 & 1 & 1 & 3 & 3 & 6 & 1 & 1 & 3 & 1 \\
3 & 2 & 1 & 2 & 1 & 1 & 4 & 2 & 2 & 3 \\
1 & 2 & 3 & 1 & 2 & 1 & 2 & 4 & 2 & 3 \\
1 & 1 & 1 & 2 & 2 & 3 & 2 & 2 & 4 & 3 \\
1 & 1 & 1 & 1 & 1 & 1 & 3 & 3 & 3 & 6
\end{pmatrix}.
\tag{15}
$$

We use Romberg integration to compute inner products of quadratic basis polynomials $f_i(\mathbf{x})$ and $g_j(\mathbf{x})$ and the function $F(\mathbf{x})$ to be approximated.