Interval Simplex Splines for Scientific Databases

by

Jingfang Zhou

M.S. Naval Architecture and Marine Engineering (May 1988)
B.S. Naval Architecture and Marine Engineering (July 1985)
Huazhong University of Science and Technology, China

Submitted to the Department of Ocean Engineering
in partial fulfillment of the requirements for the degree of

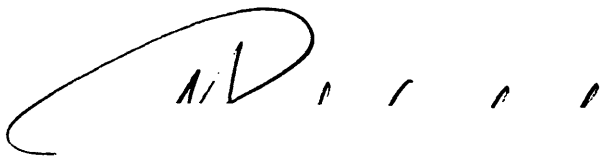
Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1995

Author .....
Department of Ocean Engineering
30 May 1995



Certified by .....
Nicholas M. Patrikalakis
Associate Professor of Ocean Engineering
Thesis Supervisor

Accepted by .....
A. Douglas Carmichael
Chairman. Departmental Committee on Graduate Students
Department of Ocean Engineering

Interval Simplex Splines for Scientific Databases

by

Jingfang Zhou

Submitted to the Department of Ocean Engineering
on 30 May 1995, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Modern technology has enabled the scientific community to collect vast amounts of physical data. These data are often sampled at unstructured spatial locations because of physical constraints, and measured with uncertainty because of measurement technique errors. However, most existing data representation and visualization methods apply only to gridded or regular data and ignore the uncertainty in the data. There currently exists no general and robust solution to the n-dimensional nonlinear continuous, uncertain data management problem supporting computer interrogation and visualization.

In this thesis, we investigate the application of multivariate simplex splines on scientific database systems to represent, manipulate and visualize efficiently huge collections of 2D and 3D data. Our methods are extensible to general n-dimensional data. The system we develop consists of preprocessing (domain establishment), construction of data representation functions, and visualization.

The domain of 2D or 3D physical data is created by a generalization of the convex hull of unstructured data, the α-shape, such that the intuitive shape of the domain of the data can be captured in a realistic manner. The α-shape technique is used to detect interactively the boundary of the domain of a finite set of points and thus to generate an appropriate domain for a set of points for data modeling and representation.

Multivariate simplex splines are introduced and applied in this work for non-linear physical data representation. Methods of efficient evaluation of bivariate and trivariate simplex splines are presented and implemented in the system. By using simplex spline functions in data modeling, optimal smoothness up to C^{k-1} for piecewise polynomials of degree k over irregular domains is achieved automatically. Least-squares approximation is adopted in 2D and 3D data modeling. Automatic knot generation and adaptive subdivision are developed in the data modeling procedure to enhance the modeling efficiency and accuracy.

Interval simplex splines (ISS) are introduced next and a method for creating bivariate and trivariate ISS functions from measured data based on linear optimization is developed to capture all possible behavior of physical data with uncertainties in irregular domains. In this way, the uncertainties in the data set are represented in a manner that provides for high accuracy and resolution with a guaranteed bounded approximation error.

Current algorithms in the marching cubes category for rendering implicit or isovalued surfaces lack the power to accurately render surfaces in irregular domains. Based on the cell-tuple topological modeling data structure and adaptive subdivision, marching simplices, a method of visualizing general scalar functions in 2D and 3D irregular domains, is developed and implemented. It has the following advantages: ability in modeling scalar functions in irregular domains; elimination of ambiguous cases; provision of automatic consistency between pieces of the function in adjacent simplices; and generation of isosurfaces ready for further interrogation since their topology is encoded.

Examples illustrate the computational properties of the above techniques.

Thesis Supervisor: Nicholas M. Patrikalakis
Title: Associate Professor of Ocean Engineering

To Xing ..
for making my life complete.

Acknowledgements

Personally, I would like to thank the following people: First, my advisor, Prof. N. M. Patrikalakis who supported and guided me through five years at MIT; the members of my committee, Prof. C. Chryssostomidis, Prof. D. C. Gossard, Prof. J. Peraire, Dr. S. T. Tuohy, and Dr. X. Ye, who provided very valuable information and helped me by reviewing and commenting on the thesis. I am grateful to Prof. H.-P. Seidel for sending me his papers on simplex splines and answering various related technical questions. I would like to thank Dr. E. Brisson for useful discussions on data structure issues in the early stages of this work. I am also grateful to Dr. S. C. Lee for valuable discussions on simplex spline topics. I thank Prof. F.-E. Wolter for providing me with good ideas in my early years at MIT. Finally, I also thank every teacher that I have had, especially Prof. D. Xia and Prof. J. Yu, for guidance and encouragement.

Additionally, I wish especially to thank the members and staff of the Design Laboratory, in the Department of Ocean Engineering: Mr. M. S. Drooker, who helped me whenever I needed assistance with the computers; Mr. S. L. Abrams, who was always ready to answer my questions on programming and helped me with visualization of my work; and Dr. T. Maekawa, Dr. E. C. Sherbrooke, Dr. C.-Y. Hu, and Mr. W. Cho for useful discussions on CAD and computational geometry. The time here was better because of all of you.

I am thankful to my English tutor and my friend Ms. Tara Zend for teaching and improving my English writing skills in my early years at MIT.

I want to thank all my family members, my mother and father, and my young brothers, for their love, patience, understanding and support throughout my life. Finally, I want to thank my husband for his support through all these years; this thesis could not be finished without his love and good caring, and particularly the stimulating discussions on computer programming skills.

The Charles river, Boston, MA, USA, bathymetric data used in this work was gathered by Ms. J. Brener and Ms. M. Frey during the 1993 MIT Sea Grant Summer Undergraduate Workshop in AUVs under the supervision of Dr. D. K. Atwood and Dr. S. T. Tuohy.

The volumetric data was obtained from Mr. E. Dever of the Woods Hole Oceanographic Institution (WHOI), Woods Hole, MA, USA and consists of CTD observations made off the northern coast of California under the Shelf Mixed Layer Experiment (SMILE).

Financial support for this work was provided by the MIT Sea Grant College Program under grant number NA90AA-D-SG-424 and by the National Science Foundation under grant number IRI-9224640. Related work on interval models was supported in part by the National Science Foundation and the Office of Naval Research under grant numbers DMI-9215411, and N00014-94-1-1014 and N00014-94-1-1001.

Contents

Abstract 2

List of Figures 8

List of Tables 11

1 Introduction 12
- 1.1 Motivations . . . 12
- 1.2 Overview of Previous Work . . . 13
  - 1.2.1 Data Representation . . . 13
  - 1.2.2 Data Representation with Uncertainty . . . 21
  - 1.2.3 Review of Visualization Methods . . . 22
- 1.3 Problem Statement . . . 25
- 1.4 Methodology of the Thesis . . . 26
- 1.5 Organization of the Thesis . . . 30

2 Preprocessing for Data Modeling 31
- 2.1 Introduction . . . 31
- 2.2 The Domain of Data . . . 31
  - 2.2.1 2D α-Shape . . . 32
  - 2.2.2 3D α-Shape . . . 35
- 2.3 Data Segmentation . . . 38
  - 2.3.1 2D Data Segmentation . . . 39
  - 2.3.2 3D Data Segmentation . . . 44
- 2.4 Topological Modeling Structure . . . 47

3 Simplex Spline Representation and Evaluation 50
- 3.1 Bivariate Simplex Splines . . . 50
  - 3.1.1 Mathematical Background . . . 50
  - 3.1.2 Evaluation of Bivariate Simplex Spline Functions . . . 54
  - 3.1.3 Differentiation of Bivariate Simplex Spline Functions . . . 59
  - 3.1.4 Integration of Bivariate Simplex Spline Functions . . . 60
- 3.2 Trivariate Simplex Splines . . . 60
  - 3.2.1 Mathematical Background . . . 60
  - 3.2.2 Evaluation of Trivariate Simplex Spline Functions . . . 63
  - 3.2.3 Differentiation of Trivariate Simplex Spline Functions . . . 67
  - 3.2.4 Integration of Trivariate Simplex Spline Functions . . . 68
- 3.3 Properties of Simplex Splines . . . 68

3.4 Discussion 69
3.5 Examples of Bivariate/Trivariate Simplex Spline Functions 71

4 Modeling Physical Data in Irregular Domains 75
4.1 Introduction 75
4.2 2D Data Modeling 76
4.2.1 Knot Generation 77
4.2.2 Least Squares Approximation 80
4.2.3 Error Estimation 81
4.2.4 Adaptive Domain Subdivision 81
4.3 3D Data Modeling 82
4.3.1 Knot Generation 84
4.3.2 Least Squares Approximation 85
4.3.3 Error Estimation 86
4.3.4 Adaptive Domain Subdivision 86
4.4 Examples of Data Modeling 87

5 Interval Simplex Spline Representations 93
5.1 Introduction 93
5.2 Interval Arithmetic 94
5.3 Interval Functions 95
5.4 Interval Simplex Spline Functions 96
5.5 Construction of Bivariate (Trivariate) ISS Functions 98
5.5.1 Preprocessing 98
5.5.2 Fitting 2D Data with ISS Bivariate Functions 99
5.5.3 Fitting 3D Data with ISS Trivariate Functions 99
5.6 Examples of Interval Simplex Spline Surfaces 100

6 Marching Simplices for Interrogation 102
6.1 Introduction 102
6.2 Problem Definition 103
6.3 The Relation between Domains and Surfaces 104
6.4 Approximation of Isovalued Functions 106
6.4.1 Contour Lines 106
6.4.2 Implicit or Isovalued Surfaces 106
6.5 Refinement of Isovalued Curves and Surfaces 109
6.6 Overview of Marching Simplices Algorithm 110
6.7 Examples 111
6.7.1 Bivariate Functions 112
6.7.2 Trivariate Functions 112

7 Conclusions and Recommendations 116
7.1 Contributions 116
7.2 Future Research Issues 117

A Delaunay Triangulation 118

A.1 General Delaunay Triangulation 118
A.2 Constrained Delaunay Triangulation 119
A.3 α-Shape and Delaunay Triangulation 120

B Determining the Simplex Enclosing a Point **121**
B.1 Introduction 121
B.2 Construction of Buckets 121
B.3 Putting Vertices into Buckets 122
B.4 Putting Edges into Buckets 123
B.5 Determining the Simplex Enclosing a Point 125

C Geometrical Interpretation of Simplex Splines **127**

Bibliography **130**

List of Figures

1-1 Discrete surface analysis. 15
1-2 The look-up-table for marching cubes algorithm. 23
1-3 2D example of ambiguities. 24
1-4 Flow chart of the methodology. 29

2-1 α-hull for a) $\alpha > 0$ and b) $\alpha < 0$. 33
2-2 α-shape is illustrated procedurally from Delaunay triangulation. 34
2-3 The Delaunay triangulation of a set of 303 planar points. 35
2-4 The α-shapes for a set of 303 points. 36
2-5 The constrained Delaunay triangulation of a set of planar points. 37
2-6 The α-shapes of a set of 3D data at different α values. 37
2-7 1D data segmentation. 40
2-8 The bathymetric data from an area of Charles river, Boston, MA, USA. . . 41
2-9 The filtered and unfiltered data of the Charles river. 42
2-10 The smoothing of the boundary of a domain. 43
2-11 The smoothing of the boundary of a domain. 43
2-12 The constrained Delaunay triangulation on the boundary nodes of the α shape. 44
2-13 The constrained Delaunay triangulation after one interior node is inserted. . 45
2-14 The constrained Delaunay triangulation after four interior nodes are inserted. 45
2-15 The triangulation ready for data modeling. 45
2-16 A triangle. 48
2-17 The triangle's incidence graph. 48

3-1 Two half-open 2-simplices. 51
3-2 The region Ω_2 formed by intersections. 53
3-3 Determining if a point "o" is in the half-open triangle $[\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)$. 57
3-4 Efficient evaluation of a quadratic bivariate simplex spline basis function. . 57
3-5 Efficiently evaluating a set of quadratic bivariate simplex spline basis functions. 58
3-6 The direction "out" for a tetrahedron 61
3-7 Project the 3-simplex onto y-z plane. 64
3-8 Efficiently evaluating a quadratic trivariate simplex spline basis function. . 66
3-9 The domain of a simplex spline surface. 72
3-10 A quadratic simplex spline surface. 72
3-11 The domain tetrahedron and the knots of a trivariate simplex spline. 73
3-12 Isosurface of a trivariate simplex spline function in a tetrahedron. 74

4-1 Flow chart for data modeling. 76
4-2 Knots associated with a vertex are in a circle centered at the vertex with radius $r \leq \frac{L_{min}}{2}$. 77
4-3 A new set of constraints is formed after introducing new knots in one domain triangle. 78
4-4 Boundary knot generation: knots associated with a boundary vertex should be in the shaded region. 79
4-5 Inner knot generation. 79
4-6 Subdivide every triangle into four. 82
4-7 Subdivide the triangle with E_{max} at its longest edge into two and the adjacent triangle into two. 82
4-8 Subdivide the triangles into two or four. 83
4-9 Subdivide the triangles into two or four depending on their shapes. 83
4-10 Subdivision of a tetrahedron into two on its longest edge. 86
4-11 Subdivision of a tetrahedron into eight sub-tetrahedra: 87
4-12 Concave domain with a hole (slightly shaded) and the knot configuration. . 87
4-13 Surface defined on a 2D domain. 89
4-14 Domain and knots generated for the initial triangulation for the bathymetric data on a portion of the Charles river. 89
4-15 Subdivision on the initial triangulation. 90
4-16 The wireframe of the topography surface of a portion of the Charles river. . 90
4-17 The topography surface of a portion of the Charles river. 91
4-18 3D domain and three isosurfaces of a trivariate simplex spline function. . . 91
4-19 An other view of the 3D domain and three isosurfaces. 92

5-1 Interval function representation. 96
5-2 Interval simplex splines. 97
5-3 The upper surface for a set of data taken from a portion of Charles river. . 100
5-4 The lower surface. 101
5-5 The ISS surface. 101

6-1 A domain in $\mathbf{R}^2$ represented by a triangulation with a polygonal hole. . . . 103
6-2 Two adjacent triangles in a domain. 104
6-3 An incidence graph for a generated contour. 105
6-4 Two adjacent tetrahedra in a domain. 105
6-5 The incidence graph for the generated isovalued surface. 105
6-6 The look-up-table for marching 2-simplices. 107
6-7 The look-up-table for marching 3-simplices. 108
6-8 Contour lines of a bivariate function over a square with a hole. 113
6-9 A saddle surface in a cube. 113
6-10 Surface of product of three cylinders in a cube. 114
6-11 The domain of the cylinders in a cube with a tetrahedral cavity. 114
6-12 Surface of product of three cylinders in a cube with a tetrahedral cavity in the domain. 115
6-13 Wireframe of the implicit surface. 115

A-1 The Voronoi diagram (solid) and the Delaunay diagram (dashed) (adapted from [52]). 118
A-2 Sample of triangulation and adjacency storage (adapted from [14]). 119

B-1 A rectangle which bounds the edges of a triangulation in the coordinate system. 122
B-2 Buckets containing the end points. 124
B-3 Intermediate buckets. 124
B-4 A 2D bucket associated with three transformed edges. 125

C-1 Geometrical interpretation for cases $s = 1$, $m = 2$ and $s = 1$, $m = 3$. 129
C-2 Bivariate quadratic simplex spline basis function over 5 knots. 129
C-3 Bivariate quadratic simplex spline basis function. 129

List of Tables

3.1 The number of items needed for evaluation of a bivariate simplex spline basis function. 54
3.2 The number of items needed for evaluation of a trivariate simplex spline basis function. 65
3.3 Comparison of simplex spline, B-spline, and composite Bézier triangles. . . 69
3.4 Complexity of evaluation of a quadratic simplex spline surface and a quintic Bézier triangular patche. 70

Chapter 1

Introduction

"The purpose of computing is insight, not numbers"
— Richard Hamming, 1962 [54].

"Scientific visualization is concerned with exploring data and information graphically – as a means of gaining understanding and insight into the data."
— Rae Earnshaw, 1992 [27].

1.1 Motivations

Computer databases have been used for business and administrative applications for more than twenty years. In the early days of computing machinery, the economic demands caused by administrative applications (e.g., census problem) and automated bookkeeping were the most important driving forces behind the development of computing machinery and its accompanying software. Therefore, this area of business-oriented database concepts has reached a level of practical maturity. The situation appears quite different in the area of scientific databases. Here exist huge collections of data, for example, 20 gigabytes containing geographical and geophysical data of the US, see [102]. However major problems remain when evaluating and interrogating these data entities to answer important questions using the information encapsulated in the data [58]. One of the reasons is that, while in administrative problems, software developed for storing, evaluating, comparing, and sorting discrete records carries very far, it is generally not sufficient to describe and evaluate complex physical phenomena [51]. Here the amount of single data items is simply much larger than what is encountered in the business world and the questions mostly posed by evaluations are different. Generally the scientist is not so much interested in single data but rather in evaluating continuous functions defined by using the data and in the knowledge of the physical behavior of the system. As far as scientific data have a physical meaning, they can be modeled by continuous functions because these functions (derived from a few basic physical principles) have been extremely successful in describing and predicting

physical phenomena. We must take this into consideration for our database because those *continuity properties* offer the possibility to *reduce the amount of data*[1] *enormously by using highly accurate approximations of the continuous function* [104]. The representation of those approximations may need many orders of magnitude less data than the representation of the direct data measurements. This data reduction problem is one of the fields where techniques from *spline approximation and representation* offer powerful tools ideal for representation of functions on computing machinery. Note that continuous data representation is not only necessitated by the request for data reduction but also by the fact that physical properties depending on (possibly higher order) derivatives (e.g., velocity, acceleration, curvature) can only be recognized properly through continuous representations of the data. (See also [104].)

Physical and engineering data usually have inaccuracies and uncertainties. Those *uncertainties* exist because of limitations in measurement precision, methods of measurement reporting or, often, because of difficulties in making observations and measurements in inaccessible or hostile regions. An important fact of scientific data is that they are often sampled in irregular regions, that is, a domain that cannot be accurately decomposed into square or cubical elements (quadtree, octree and voxel) and have non-uniform population distribution. However, most current data modeling and database systems do not offer robust solutions to capture and process the uncertainties in physical data and handle data defined in irregular domains. Thus, it is desirable to develop a database system to represent, interrogate and visualize scientific data with uncertainty in irregular domains.

Scientific databases are also driven by important applications as e.g., (1) measurements of temperature, salinity, and conductivity at various locations in 3D space in the ocean; (2) mineral concentrations known at various locations in 3D space; (3) precipitation measurements at various weather stations; (4) density measurements at various locations inside a human body; and (5) geographical information systems (GIS).

1.2 Overview of Previous Work

Three main components for establishing a scientific database system are:

1. preprocessing, which facilitates the data modeling process;
2. representation, which is the key in the construction of a scientific database system;
3. and interrogation/visualization, which helps to further analyze and understand the physical phenomena represented by the data.

1.2.1 Data Representation

The database research community is currently devoting considerable attention to scientific database issues [51]. The techniques developed for scientific data management have been

[1] While keeping all the relevant information they contain.

driven by applications and are often case-specific. There is currently no general and robust solution to the n-dimensional nonlinear continuous, and uncertain data management problem supporting computer interrogation and visualization.

Data representation is an important issue in many fields such as geology, medical imaging, shape design, CAD/CAM, and computer vision. The vast range of applications has stimulated diverse methods for this problem since the early sixties. Different methods, developed to fulfill different application requirements, can be categorized by several criteria. Data representation can be classified in three categories: *discrete*, *linear*, and *nonlinear*. Discrete representation includes quadtree representation techniques; linear representation uses faceted models for the data; and nonlinear representation, which is more related to this thesis, contains the class of algebraic, Coons, B-spline, and interval B-spline functions. In our work, we focus on nonlinear representations. Again, in the category of nonlinear representation, there are several classifications which will be addressed in the following subsections.

Discrete Representation

In the literature for discrete representation, many researchers use quadtree, octree, or voxel methods to represent data, especially in image processing. Samet [81] discussed quadtree and octree representations. In [81] the quadtree is a refinement of the array representation of an image that attempts to save storage by taking advantage of a regularity in the image by decomposing the array into homogeneous disjoint cubes centered at predetermined positions. A shortcoming of the quadtree representation is that it involves a considerable amount of overhead in computation.

Modeling is key to the development of scientific data visualization and interrogation. Many current volume rendering applications are based on very regular and dense 3D image data from a scanning instrument, as in magnetic resonance imaging. This type of data and its constituent "voxels" are the direct 3D analog of the 2D images and pixels associated with raster graphics. We often view this data as samples (over a regular Cartesian grid) of a scalar-valued trivariate function. Some authors have used "volume modeling" to refer to the process of identifying and synthesizing objects contained in this type of 3D data set, see [71] for details.

Linear Representation

A typical approach to surface reconstruction in this category is to use *surface characteristics* extracted directly from the discrete data set. Combining classical topological techniques with differential geometry, the evaluation of Gaussian curvature is used in [34, 33, 5]. Gaussian curvature can be extracted from a faceted model by use of the angle excess associated with the polygonal paths of the triangular subdivision. The Gaussian curvature K_v of a

faceted surface at a vertex v can be computed with the following relations.

$$\Delta_v = 2\pi - \sum_{i=0}^{\Omega} \theta_i \tag{1.1}$$

$$\theta_i = \cos^{-1}\left(\frac{a_i^2 + b_i^2 - c_i^2}{2a_i b_i}\right) \tag{1.2}$$

$$K_v = \frac{2\Delta_v}{\sum_{i=0}^{\Omega} A_i} \tag{1.3}$$

$$A_i = \sqrt{s_i(s_i - a_i)(s_i - b_i)(s_i - c_i)} \tag{1.4}$$

$$s_i = \frac{1}{2}(a_i + b_i + c_i) \tag{1.5}$$

where Δ_v is the angle excess measured in radians (as defined in Figure 1-1), Ω is the number of adjacent vertices, θ_i is the angle of the triangle formed by sides with lengths a_i and b_i (emanating from the vertex being interrogated), c_i is the side connecting the adjacent vertices, and A_i is the area of the triangle. The curvature information is mainly used to

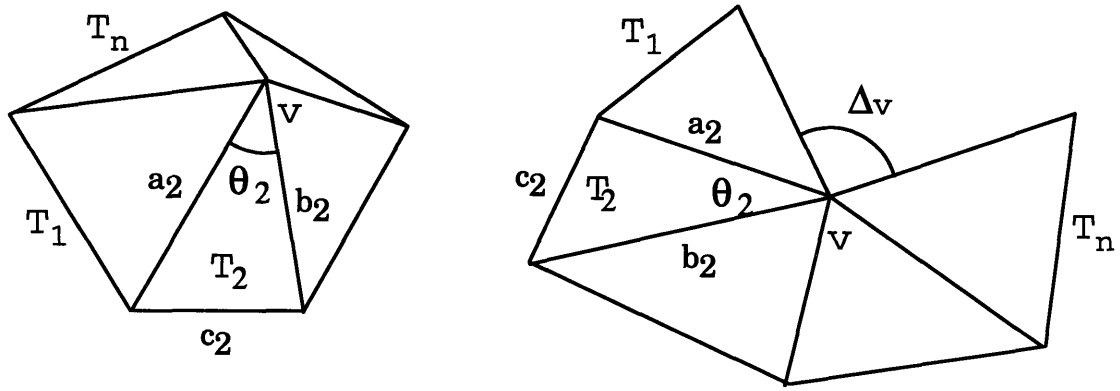


Figure 1-1: Discrete surface analysis (adapted from Tuohy [96]). T_i is the i^{th} triangle.

extract the convex, concave, planar and saddle regions of the surface for the surface shape description.

In [21, 32], an algorithm for constructing a *Delaunay triangulation* constrained [2] by extracted surface characteristics is used to construct a faceted model of the data. In the latter, surface refinement by incremental (based on deviations from the underlying data) and stochastic means (the addition of a random displacement in a tiling approach) is used to construct the surface representation.

[2] A *constrained Delaunay* triangulation is a process of surface refinement that allows the violation of the Delaunay criteria to enhance the accuracy of the representation. For further details on definition of Delaunay triangulation, see Appendix A.

Nonlinear Representation

In this category, for 2D data, many basic ideas and different approaches have been surveyed by Schumaker [83] and Barnhill [4]. Many developed methods have been tested and compared by Franke [44, 45]. For both 2D and 3D data, Foley *et al.* [40] outlined several methods that were recently developed for computer visualization and data interpolation. They include *multiquadric(MQ) interpolation, distance and nearest neighbor methods, triangulation-based methods.* In this section, we list the ones in [40] and several recently developed methods. All these methods are briefly described in the following subsections.

Multiquadric (MQ) Interpolation

The MQ method is an infinitely differentiable function of the form

$$F(\mathbf{p}) = f(\mathbf{p}) = \sum_{i=1}^{N} a_i B_i(\mathbf{p}) \tag{1.6}$$

where $\mathbf{p} = (x, y), B_i(\mathbf{p}) = \sqrt{d_i^2(\mathbf{p}) + R^2}$, $d_i^2(\mathbf{p}) = ||\mathbf{p} - \mathbf{p}_i||^2 = (x - x_i)^2 + (y - y_i)^2$ and R^2 is a positive constant. The coefficients a_i are computed by solving the $N \times N$ linear system of equations $F(\mathbf{p}_j) = f_j$ for $j = 1, ..., N$, where N is the number of the scattered data. In matrix form, the vector $a = (a_1, ..., a_N)^T$ is the solution of $Aa = f$, where $f = (f_1, ..., f_N)^T$ and the (i, j)-element of matrix A is $B_j(\mathbf{p}_i)$. See Hardy [55] and Franke [45] for a complete description.

The reciprocal multiquadric (RMQ) interpolant is similar to the MQ interpolant, except that the basis functions are $[B_i(\mathbf{p})]^{-1}$, $i = 0, ..., N$.

Multiquadric interpolation is one of the simplest methods to implement and provides one of the most effective solutions. It could be used for 3D data modeling easily. A negative aspect of the MQ method is that if N is large, the linear system of equations can be ill-conditioned and costly to solve. The accuracy of the MQ method depends heavily upon the value chosen for R^2. The optimal value of R^2 is problem-dependent, and most early formulas for R^2 involve the number of data points together with the size and shape of the domain containing the data. A critical unsolved problem involving the use of the MQ method is how to compute the optimum value (or even a consistently good value) for R^2.

Distance and Nearest Neighbor Methods

These methods are also called *Shepard's Methods* [86]. They were developed by meteorologists and geologists [17, 16], and are still undergoing development. Shepard defined his interpolating function $f(x, y)$ to be a weighted mean of the ordinates f_i:

$$f(x, y) = \sum_{i=1}^{N} w_i(x, y) f_i \tag{1.7}$$

with weight functions (basis functions)

$$w_i(x,y) = \frac{\sigma_i(x,y)}{\sum_{j=1}^{N} \sigma_j(x,y)} \tag{1.8}$$

where

$$\sigma_i(x,y) = \frac{1}{[(x-x_i)^2 + (y-y_i)^2]^{u_i/2}} \qquad u_i \in R^+ \tag{1.9}$$

is a power of the inverse Euclidean distance $d_i(x,y) = \sqrt{(x-x_i)^2 + (y-y_i)^2}$. This method is simple and could be used for 3D data modeling. But it may have cusps at data points or its partial derivatives may be ill-defined at these points. This type of method is global, that is, as soon as one point is changed or a data point is added or removed, the interpolating process needs to be repeated to permit interrogation/visualization.

Triangulation-based Methods

Generally in this category, there are sub-classifications, including *classical methods*, *minimum norm network methods* and *physically-based methods.*

1. *Classical methods*: This type of method involves triangulating the domain vertices and defining the fitting function $F(\mathbf{p})$ piecewise over each triangle. For interpolation of bivariate scattered data using a triangle-based method, the following steps are involved. The points $\mathbf{p}_i = (x_i, y_i)$ in the plane are first triangulated using local or global optimality criteria. The next step involves estimating partial derivatives of the underlying function at the vertices p_i based upon some or all of the data (x_i, y_i, f_i). Finally, the interpolating function is defined over each triangle using the position and derivative information at the vertices of the triangle.

   It has been shown in Farin [38] that it is not possible, in general, to solve the scattered data interpolation problem with cubic polynomials defined over arbitrary triangulated data with given tangents and guarantee C^1 continuity. The Clough-Tocher method [15] overcomes the limitation by first splitting each domain triangle into three triangles, and then a piecewise cubic interpolation is formed over the finer triangulation. Given a triangulation of the scattered data and first-order partial-derivative estimates at the vertices, a degree of freedom remains in the Clough-Toucher method for each edge of the original triangulation. A commonly used approach is to assume that the cross boundary derivatives vary linearly on the edges. A more useful approach for selecting the value of this degree of freedom is given in Farin [38], where the cross-boundary derivatives are selected to minimize a measure of the jump in the second derivative across the edge.

2. *Minimum norm network methods*:
   After constructing a triangulation of the data points in the plane, the minimum norm network (MNN) method in Nielson [70] computes first-order partial derivatives at

the vertices (x_i, y_i) that define cubic polynomials over each edge of the triangulation. The first-order partial derivatives are computed by solving a sparse linear system of equations so that the network of piecewise cubic polynomials minimizes the integral of the square of the second derivatives over all edges in the triangulation. Because of this minimization, the piecewise cubic network can be considered to be a bivariate analog of the univariate cubic spline. The MNN method is a C^1 interpolant defined on the convex hull of a set of data points, and it is perhaps the most-effective of the triangle-based methods tested in Franke [45].

Examples of cellular decomposition methods for volumetric data are the 3D version of the Minimum Norm Network (MNN) spline and the localized version of the volume spline. The MNN can be applied to very large data sets, but its implementation requires a tetrahedrization algorithm and a fairly complicated iterative method for solving a large, sparse equation system. There are general strategies for local methods but these approaches still need considerable work before they are truly viable.

3. *Physically-based methods*:
In related work, Celniker [9] and Celniker and Gossard [10] developed methods for *finite element based free-form shape design.* The finite element method is applied to generate shapes that minimize a surface energy functional subject to user-specified geometric constraints while responding to user-specified forces. Because of energy minimization, these surfaces seek globally fair shapes. Fair shapes with C^1 continuity are constructed. Fang [35] and Fang and Gossard [37, 36] presented physically-based methods for parametric curve and surface reconstruction in case the parametric values of interior points of these entities are not known. The basic ideas involve simulating the deformations of an elastic beam and plate under the application forces by minimizing energy functionals which are specifically designed to meet CAD requirements. Similar to work by Celniker, these functions have C^1 continuity and the method was developed for univariate and bivariate problems.

   Terzopoulos and Qin [93] developed a dynamic generalization of the nonuniform rational B-spline (NURBS) model, or D-NURBS, to represent free-form shapes as well as common analytic shapes. This model is a physics-based model that incorporates mass distributions, internal deformation energies, and other physical quantities into the popular NURBS geometric substrate. Using D-NURBS, a modeler can interactively sculpt curves and surfaces and design complex shapes to required specifications not only in the traditional indirect fashion, by adjusting control points and weights, but also through direct physical manipulation, by applying simulated forces and local and global shape constraints. Their dynamic behavior results from the numerical integration of a set of nonlinear differential equations that automatically evolve the control points and weights in response to the applied forces and constraints. Lagrangian mechanics and a finite-element-like discretization were employed to derive equations for the dynamic behavior of the D-NURBS. Very recently, Qin and Terzopoulos [79] developed a physically based model relying on triangular B-spline geometry and prin-

ciples of physical dynamics. The dynamic behavior of this model, resulting from the numerical integration of differential equations of motion, produces physically meaningful and highly intuitive shape variation. Similar to their previous work, Lagrangian mechanics are used to formulate the equations of motion and finite element analysis is employed to reduce these equations to efficient numerical algorithms. Dynamic triangular B-spline was applied in fitting of structured 2D data.

The class of *triangulation-based methods* are local methods and can yield smooth curves or surfaces. A possible disadvantage of these methods is that they are generally implemented in a procedural approach by cubic Hermite operators and a weighted combination of three partial interpolants is formed. This lack of a simple closed form expression also makes it somewhat complex to compute partial derivatives and surface normals that are needed in most geometry processing applications.

Contour Based Methods

In a different approach, *homotopy* techniques using continuous deformation based on a toroidal graph allow surface reconstruction from lower-dimensional data, such as contours, see in Kunii and Shinagawa [59]. A toroidal graph represents correspondence between points on adjacent contours in order to prevent unwanted folds or twists in the surface. As an extension of this work, a new type of toroidal graph is used in Ikeda *et al.* [58] as a data structure for terrain topology to represent the relationship between multiple contours including critical points where contours split or merge. It is unclear how the method might extend to higher dimensions.

Spline Based Methods

In a manner similar to physically-based methods, many methods for surface reconstruction have used techniques from the field of *regularization.* In Delingette [22], Sinha [88], and Terzopoulos [94], a surface is constructed by minimizing an energy functional that characterizes geometric properties of the original surface. The method constructs a surface $Z(x, y)$ such that

$$z_i = L_i[Z(x, y)] + \varepsilon_i \tag{1.10}$$

for all i where L_i denotes *measurement functionals*, i.e., linear operators, of $Z(x, y)$

$$L_i[Z] = \left.\frac{\partial^k Z}{\partial x^j \partial y^{k-j}}\right|_{(x_i, y_i)} \qquad j = 0, 1, 2, \ldots, k \tag{1.11}$$

and z_i is the function value of the data and ε_i denotes measured errors.

The regularized solution to the surface reconstruction problem seeks a minimum to the

Tikhonov general form:

$$|v^2|_k = \int\int_D \sum_{j=0}^{k} \binom{k}{j} (L_j(Z))^2 \, dx \, dy. \tag{1.12}$$

where D is the 2D rectangular domain of interest.

In a similar approach, Dierckx [23] formulates the problem as finding the optimal knot vector for a least squares fitting problem using splines. It defines a measure of smoothness of the approximating spline as the high order discontinuity across the knots of the spline and seeks a minimum subject to the constraint that the weighted sum of the squared residuals cannot exceed a given value. This problem is solved using the method of Lagrange multiplier.

Tuohy and Bardis [97] described a surface approximation method which can convert a high degree Bézier or B-spline surface to a lower degree representation to facilitate the exchange of surface geometry between different geometric modeling systems. It uses adaptive sampling to compute approximation error and lofting of isoparametric curves to efficiently produce the approximating surface. The method is incapable of approximating scattered data in an irregular domain. The 3D version of B-spline (volumetric B-spline) modeling is a natural extension of B-spline surfaces and works well, but applies to modeling uniformly distributed data only. See Tuohy *et al.* [100, 101] and Yoon [106] for further description.

Attempts to model data in irregular domains include application of simplicial B-splines. Auerbach *et al.* [2] developed an approximation and geometric modeling system with simplicial B-splines associated with irregular triangles. They employed bivariate quadratic simplicial B-splines defined by their corresponding set of knots which are derived from a (suboptimal) constrained Delaunay triangulation of the domain and use the simplicial B-spline to fit a set of data to obtain a C^1-smooth surface. The simplicial B-splines they used are defined and developed in Dahmen and Micchelli [18, 19] and Höllig [56]. By using simplicial B-splines associated with irregular triangles for the purpose of approximation and geometric modeling, the shortcomings of tensor product B-splines requiring a grid of knots topologically equivalent to a rectangular grid are overcome, while at the same time many of their features are preserved. However, it is unclear how appropriate is the selection of the domain of the 2D scattered data in their approach and how they subdivided the domain if the resulting simplicial surface is not satisfactory after the first round of fitting. One of the other weaknesses of this modeling system is in the calculation of the simplicial B-spline. Incorrect results may arise when the recurrence relation is used for points that happen to be on knot lines (lines connecting any two knots), see Micchelli [68] and Grandine [48] for further explanation.

Very recently, Pfeifle and Seidel [78] presented a scheme to fit 2D scattered data with bivariate simplex spline functions defined in [20, 85, 43]. The scheme first automatically triangulates a domain containing the data points and generates basis functions over the triangulation. Then it combines least squares and bending energy minimization into the approximation of 2D scattered data. In this manner, a smooth surface can be generated.

Therefore, this scheme is a good application of bivariate simplex splines, but is not extended into trivariate problems.

Based on the above reviewed work, in this thesis, we will develop a data representation system which includes an appropriate definition of the domain of 2D (3D) scattered data; the proper triangulation on the 2D (3D) data domain to form initial approximating functions and the application of simplex splines in modeling 2D (3D) data as defined in Dahmen, Micchelli, and Seidel [20] to iteratively refine the approximation accuracy.

1.2.2 Data Representation with Uncertainty

A specific objective of this thesis is to explore interval arithmetic methods in representing and processing *uncertainty in physical properties* using the concept of interval simplex splines (ISS), rather than uncertainty due to numerical inaccuracy alone. However in interrogation of these splines, both uncertainties will be correctly accounted for by using interval arithmetic methods.

Stewart [90] presented a model-based approach to underwater 3D imaging and mapping with uncertainty. Approaches to incorporating new sensor information – stochastic backprojection – are derived from an incremental adaptation of the summation method for image reconstruction. Error and ambiguity are accounted for by blurring a spatial projection of remote-sensor data before combining it stochastically with the model. By exploiting the redundancy in high-bandwidth sensing, model certainty and resolution are enhanced as more data accumulate. In the case of three-dimensional profiling, the model converges to a *fuzzy* surface distribution from which a deterministic surface may be extracted.

A different approach for uncertainty modeling is *kriging*, see [75, 26]. *Kriging*, a stochastic model of spatial variation, originated from geophysical information systems (GIS) about two decades ago. This approach considers the fact that spatially distributed data behave like random variables and regionalized variable theory provides a set of stochastic methods for analysing them. It is essentially a method of estimation by local weighted averaging:

$$z^*(x) = \sum_{i=1}^{n} \lambda_i z(\mathbf{x}_i) \tag{1.13}$$

where $z^*(x)$ is the kriging estimation, and λ_i are the weights which are chosen such that $z^*(x)$ is the best linear unbiased estimator based on the $z^*(x)$, i.e., the variance is minimized. It has been proven in [75, 26] that splines are special cases of kriging. However, kriging differs from other spline-bases methods in that its objective is not aesthetics but accuracy. With kriging it is possible to calculate the estimation variance at each point which tells us how well each point is estimated compared to the others. The negative aspect of kriging is that it is time consuming.

Tuohy and Patrikalakis [99], Tuohy *et al.* [98] and Tuohy [96] applied interval methods in the representation of functions with uncertainty, such as geophysical property maps. Tensor product interval B-splines (IBS) are introduced for the nonlinear representation of two dimensional geophysical parameters within rectangular domains. A method for the creation

of IBS surfaces from measured data is developed such that the topography and uncertainty in the data are represented in a manner that provides for high accuracy and resolution, data reduction, and efficient interrogation with a guaranteed bounded approximation error. Bounds of the error are represented by intervals instead of the more commonly used point approximations. Tuohy and Patrikalakis [99] focused on geophysical data with uncertainty and developed the method of tensor product interval B-spline surfaces (IBS). The reconstruction of IBS is achieved by minimizing (maximizing) the volume under the B-spline surface with the constraint that the upper (lower) B-spline surface must lie completely above (below) the upper (lower) bilinear surface representing limits of variation of original data. The method of IBS has been extended to trivariate cases, defined over 3D rectangular parameter boxes. A summary of this work on trivariate interval B-splines can be found in [100, 101, 106].

In this work, we introduce the concept of *interval explicit* bivariate and trivariate simplex splines that not only represent the topography of the data, but also the uncertainty of the data resulting from either sensor measurement error, measurement platform position uncertainty (e.g., from GPS positioning resolution), or from standard formats for data recording (e.g., proper documentation). Specifically, quadratic *interval explicit* (rather than parametric) bivariate and trivariate simplex spline functions will be fit to the data.

1.2.3 Review of Visualization Methods

A popular technique for visualizing a trivariate scalar function is by rendering a selected set of *isosurfaces*. The 2D analog is generating *contour lines*, e.g., topographical isolevel maps. Techniques for rendering isosurfaces produce surfaces in the domain of the scalar quantity on which the scalar quantity has the same value, the *isosurface value*. These isosurfaces can be colored according to the isosurface values, or they can be colored according to another scalar field using texture mapping, and multiple isosurfaces can be displayed using transparency rendering techniques available on state-of-the-art graphics workstations.

As described by Drebin *et al.* [25] and Levoy [61], rendering techniques can be divided into surface rendering techniques and (direct) volume rendering techniques. Surface rendering is an indirect geometry-based technique which is used to visualize structures in 3D scalar or vector fields by first converting these structures into surface representations and then using conventional computer graphics techniques to render their surfaces. Direct volume rendering is a technique for the visualization of 3D scalar data sets without conversion to surface representations. For a broad review, see Rogers and Earnshaw [80] and Brodlie *et al.* [7].

A direct volume rendering technique for creating isosurfaces in sampled data considers cells with sample points as corners and approximates the isosurface in each cell by one or more polygons whose vertices are obtained by (usually linear) interpolation of the sample data. Specifically, each polygon vertex is a point on a cell edge between two adjacent sample points, where the function is estimated to be equal to the desired isosurface value. The two sample points on that edge have values on opposite sides of the isosurface value, and the interpolated point is called the intersection point.

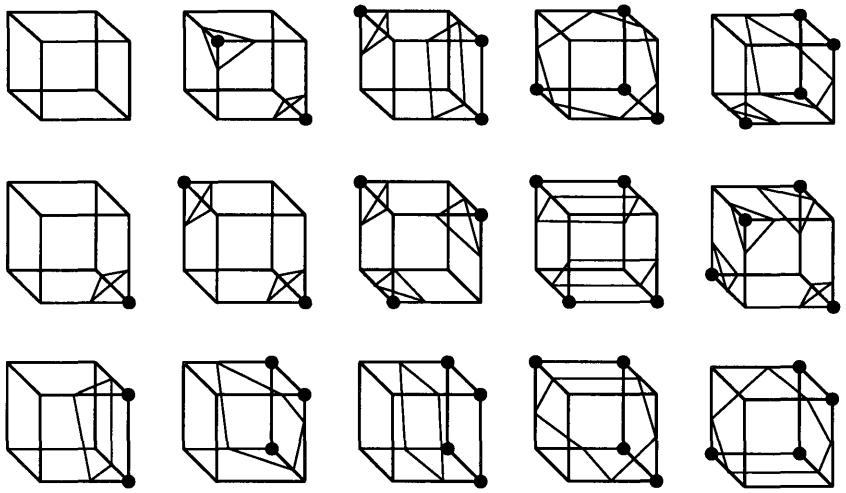


Figure 1-2: The look-up-table for marching cubes algorithm. There are fifteen distinct topologies for a partial surface in a grain cube, "•" means function value is +, otherwise the value is −.

Cube Based Methods

The *marching cubes algorithm* developed by Lorensen and Cline [64] is one such direct volume rendering method which constructs a polygonalization of an implicit (isovalued) surface within cube cells. Using a discretization of the domain with fixed sized cubes, polygons are computed for each cube from a look-up-table of possible facet topologies and a large number of polygons are produced. This algorithm uses a divide and conquer approach to create a table that defines fifteen possible facet topologies, see Figure 1-2. The algorithm processes the data in scan-line order and uses linear interpolation to calculate facet vertices.

One weakness of the standard marching cubes algorithm is that there are situations in which the choice of facet topologies is ambiguous. An incorrect choice can lead to erroneous topology in the rendered surface, and possible discontinuities on a continuous surface. A 2D example of such ambiguity is shown in Figure 1-3 where intersection points between the contour and the square element occur at all four edges and there are three possible topologies in which the intersection points can be connected. For an overview of such problems and strategies for disambiguation involving discrete data, see Ning and Bloomenthal [72]. Another weakness is resolution dependency of the surface evaluation. Here, features whose size is smaller than the finest grain discrete element may be missed entirely.

More recent extensions to the marching cubes algorithm include Schmidt's *cutting cubes algorithm* [82] and Wilhelms and Gelder's *octree method* [103]. The cutting cubes algorithm uses adaptive subdivision and makes few assumptions concerning the implicit surface and the representation of its defining function. The surface may contain singularities; for example, it may have self-intersections or be reducible. A user-defined domain is filled by a

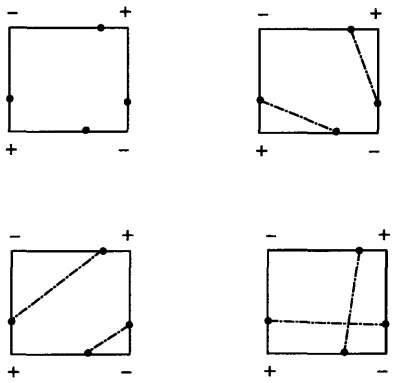


Figure 1-3: 2D example of ambiguities which occur when there are intersection points on all four edges of the square element and there are three possible topologies for the contour lines in the square.

set of cubes, some of which contain pieces of the surface. The set of cubes is controlled by an octree converging to the surface by adaptively subdividing cubes where ambiguities of possible surface topology appear. Although this method is capable of resolving some of the surface topology ambiguities in a cube and finding any isolated surface components within the domain of interest up to a specified minimum size, it may introduce inconsistencies between adjacent different-sized cubes in the octree. It also adds complexity since it requires a rescanning of the polygonalization to account for inconsistencies between polygonalization of adjacent cells that were introduced during the subdivision process.

If the function is defined by piecewise polynomials, then a method proposed by Tuohy *et al.* [100] and Yoon [106] uses the computation of critical points of the function to render reliably implicit (isovalued) surfaces. The method starts by computing the critical points of the function, that is, all points for the function $F(x, y, z)$ where $\nabla F = 0$. The solution to this system of nonlinear equations is found using the *interval projected polyhedron method* described by Sherbrooke and Patrikalakis [87] and Maekawa and Patrikalakis [66]. The original domain is subdivided at the critical point values and the subdomains rendered using marching cubes. In this manner, the algorithm will not miss singularities or small surface components regardless of relative size. Note that the algorithm computes the critical points only once, and thereafter, any implicit (isovalued) surface (i.e. $F = C$) can be rendered without additional computational penalty. The one time computation cost imposed by precomputing the critical points does not delay the interactivity of rendering multiple surfaces and is justified by the added reliability of surface rendering.

Interval methods are used in Suffern [91, 92] to robustly contour bivariate functions and render implicit surfaces based on the non-uniform quadtree decomposition. These interval methods have the advantage over their point-sampling counterparts that they are guaranteed not to miss parts of the contours or surfaces down to a specified plotting resolution.

All these methods, however, assume that the implicit (isovalued) surfaces are defined

in regular domains. Additionally, methods that attempt adaptive subdivision have added complexity to ensure topological consistency between adjacent cells. However, since there are regions of the domain for which the function is undefined, such as occurs with triangular Bézier patches [57], trimmed surfaces, and simplex spline surfaces, as described by Seidel and Vermeulen [85], *we seek an algorithm to render surfaces defined in domains described by simplices (triangulations) or, in other words, irregular domains and one which automatically encodes the topology of the resulting polygonalization.*

Here, we present the marching simplices algorithm for rendering and interrogating implicit (isovalued) surfaces defined in general irregular domains. The algorithm uses triangular (tetrahedral) cells (i.e., 2-simplices (3-simplices)) for the decomposition of a 2D (3D) domain. Similar to marching cubes, the marching simplices algorithm polygonalizes the isosurface in each simplex based on a look-up table of possible facet topologies. Adaptive subdivision techniques are used so as not to miss any fine scale features. By using a Topological Modeling Structure, based on the cell-tuple data structure developed by Brisson [6], for the original decomposition of the domain as well as the polygonalization, subdivision can be performed while automatically maintaining adjacency information. The result is a complete geometric model ready for complex interrogations (e.g., intersections, mesh generation, simulation, etc.).

1.3 Problem Statement

Modern technology has enabled the scientific community to collect a vast amount of physical data. There is currently no general and robust solution to the n-dimensional nonlinear continuous, uncertain data management problem supporting computer visualization and interrogation. These physical data exhibit certain important characteristics listed below:

- they are defined in irregular domains;
- they involve uncertainty;
- they describe continuous physical phenomena;
- they exist in huge amounts.

To efficiently and robustly represent such data, a modern scientific database should be able to

- reduce the amount of data; [3]
- evaluate (including function evaluation, differentiation, and integration), interrogate, and visualize the data to respond to user queries;

[3] Since scientific data describes continuous physical phenomena, we need to use highly accurate approximations of continuous functions.

- capture uncertainties, i.e., numerical inaccuracy and uncertainty by measurement error.

In summary, the thesis is essentially based on the following insight:

1. Continuous nonlinear higher dimensional geometry is the core of any sophisticated scientific or engineering database.
2. Data representation and data interrogation are intimately related and should not be treated as distinct issues because data representation should strongly support data interrogation, which is the purpose and reason for the existence of the database.

Thus, the goal for this work is the development of a scientific database system that efficiently represents, visualizes, and interrogates vast amount of physical data in irregular domains with uncertainty.

1.4 Methodology of the Thesis

The physical data we are interested in this work is called *scattered data*, i.e., non-grided data with non-uniform population. The domains of the definition of the data are often unknown. They are generally irregular, i.e., they cannot be accurately described by 2D rectangle or 3D rectangular box; in addition, when represented by a polygon (2D) or polyhedron (3D), the domains may not be convex. Thus such data segments differ from those that are in grided format with a uniform population.

To represent data segments, usually tensor or triple product B-spline expressions were adopted in previous work, (see Tuohy [96] and Tuohy *et al.* [100]), which are capable of producing smooth bivariate and trivariate functions. Since the domain over which the tensor or triple product B-splines are defined is a rectangle or a box in higher dimensional space, it is difficult to model irregular objects or shapes with tensor or triple product B-spline expressions. Attempts to overcome this difficulty have been based on subdivision [8, 24], interpolation [77], and on the use of multi-sided patches [62, 63, 105] and trimmed surfaces [39]. All of the above efforts work to some extent but often cause other problems. For example, a trimmed surface is not suitable to all surface problems [39]; when trying to join two trimmed surfaces together smoothly by joining along trimmed curves, there is no easy way to ensure exact tangent plane continuity between them, as was the case for standard tensor or triple product patches. Such questions of smoothness are usually handled on a case by case basis, which is clearly not desirable.

Bivariate and trivariate functional representation may also be modeled using Bézier triangles [39] which are defined on irregular triangular domains. However, *stitching* together many patches to form a large smooth surface requires a considerable amount of computation. To make the final surface smooth, i.e., to have some degree of continuity, control vertices of the patches must satisfy sets of constraints. Altering the position of these control points will destroy the continuity that has been meticulously achieved. So it is essential that a modeling scheme be developed that has both the advantages of Bézier triangles and

tensor/triple product surfaces and eliminates their drawbacks, i.e., a non-tensor or non-triple product spline function that can model surfaces or volumes in irregular domains and can produce smooth bivariate and trivariate functions *automatically* by its underlying basis functions.

Using simplex spline basis functions recently developed by Dahmen, Micchelli and Seidel [85, 20, 42, 43] can lead us to a spline expression with high order continuity while eliminating the above weakness of tensor product B-splines and Bézier triangles. Originally simplex splines [19, 56] were defined over arbitrary triangulations of the parameter space but they were lacking a geometric formulation relying on control vertices. The development of the simplex spline function by Dahmen, Micchelli and Seidel [20], is based on the combination of simplex splines with polar forms of polynomials formed by blending functions, control points and knots. Thus the simplex spline function exhibits several important properties that facilitate computation and modeling and are helpful for the development of high dimensional scientific databases. The new simplex spline allows construction of degree n bivariate (trivariate) functions over arbitrary 2D (3D) triangulations with C^{n-1} continuity. These functions are made of piecewise polynomials defined over the sub-triangulation induced by the knot net of the parametric space; thus they can be manipulated by changing the control vertices and the knots. Moving control points does not change the smoothness of the surfaces or hypersurfaces (in a manner similar to B-splines). The major properties and characteristics of the simplex spline function are [85]:

- The topology of the definition domain on which simplex splines are defined is a triangulation in n-dimensional Euclidean space. Thus it can model both surfaces and hypersurfaces in higher dimensional spaces within regular or irregular subdomains.

- The simplex spline is a piecewise polynomial of degree n over a triangulation and could be continuous up to C^{n-1} if the knots are in general position.

- The simplex spline is locally supported, i.e., it is greater than zero within part of the triangulation and vanishes outside.

- The simplex spline surfaces and their control points are affinely invariant.

The basic theory for bivariate simplex splines is available [43, 50, 20, 85], based on a combination of multivariate spline theory [12] and approximation theory [13]. In [85], the polar forms and the refinement of bivariate simplex splines are introduced. In [50], bivariate simplex splines are introduced and utilized to model solid objects. The general mathematical description for n-dimensional simplex splines can be found in [20] where the concept of multivariate simplex spline functions is described based on polar forms of polynomials. In [43], an implementation of bivariate triangular B-spline surfaces (i.e., bivariate simplex spline surfaces) over arbitrary triangulation is presented including the evaluation, subdivi-

sion and refinement. All the above works are contributions to the definition, processing and implementation of simplex splines.

However, efficient algorithms for generating and implementing multivariate n-dimensional simplex splines are still left open (e.g., trivariate simplex splines) and need to be developed and applied in scientific data representation and interrogation. In this thesis, we shall investigate not only bivariate simplex spline forms but also trivariate ones within domains defined by simplices. Combining *interval arithmetic* and *multivariate simplex splines*, we shall create *interval (multivariate) simplex splines* which have properties inherited from both and, hence, have broader applications than their predecessors.

The major contributions in this thesis are the investigation and applications of multivariate simplex splines and the development of a database system that efficiently represents, interrogates, visualizes, and manipulates huge collections of multidimensional data. Methods of efficient evaluation of degree n, bivariate and trivariate simplex splines are presented and implemented in the system. In this manner, data storage is dramatically reduced; yet, in contrast to discrete data representation used in traditional database systems, all possible data locations will be covered and higher-level interrogation can be performed.

Interval simplex splines (ISS) are presented and a method for creating bivariate and trivariate ISS functions from measured data is developed to capture all possible behavior of physical data with uncertainties in irregular domains. In this way, the uncertainties in the data set are represented in a manner that provides for high accuracy and resolution, data reduction and efficient interrogation with a guaranteed bounded approximation error.

A generalization of the convex hull of unstructured planar and spatial data is introduced and applied in defining domains of physical data such that the intuitive shape of the data can be captured in a realistic and practical manner. The α-shape technique can be used to detect the boundary of the domain of a finite set of points and thus to generate an appropriate domain for a set of planar or space points for fitting and approximation.

Based on a topological modeling structure (TMS) and adaptive subdivision, *marching simplices*, a method of visualizing bivariate and trivariate simplex spline functions as well as other general scalar functions in 2D (3D) irregular domains is presented and developed in this work. This algorithm has the following advantages: (1) it is capable of extracting implicit functions in irregular domains; (2) it needs fewer special cases for determining grain partial surface topology; (3) it eliminates ambiguity cases; (4) it creates automatic consistency between pieces of the surface in adjacent simplices; and most importantly, (5) it generates isosurfaces or implicit surfaces which are ready for further interrogation since their adjacency topology is encoded in the surface generation.

Figure 1-4 shows a conceptual flow chart of this work. We start by introducing 2D (3D) physical data preprocessing. Then we will explore (interval) bivariate (trivariate) simplex spline representation and evaluation to facilitate storage and reduction of the amount of data. Finally, we will introduce the marching simplices algorithm to effectively visualize and interrogate the (interval) representations.

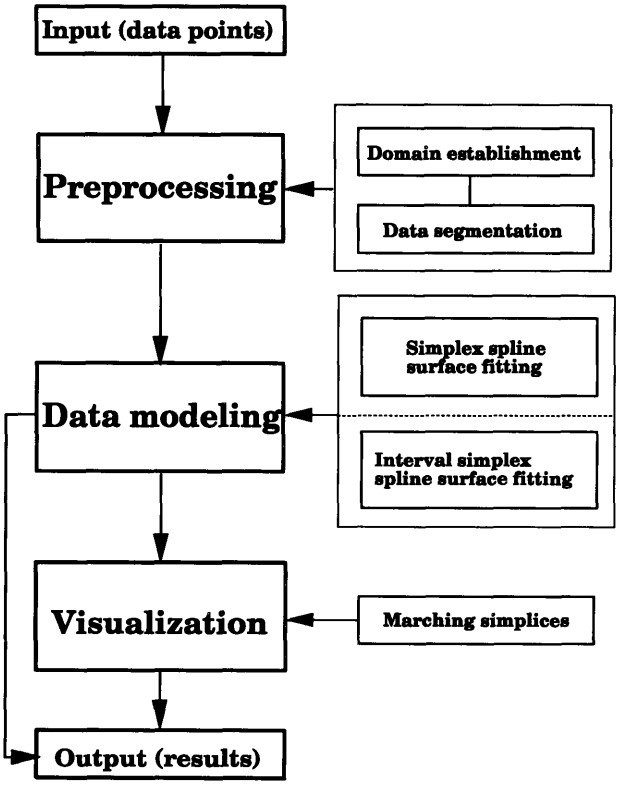


Figure 1-4: Flow chart of the methodology.

1.5 Organization of the Thesis

Chapter 2 will address the preprocessing procedure for constructing our scientific database. This preprocessing procedure includes domain establishment and data segmentation. Chapter 3 will introduce the mathematical background and explore efficient evaluation schemes for bivariate and trivariate simplex spline functions. Using simplex splines, the shortcomings of tensor (triple) product B-splines and Bézier triangles are overcome, while in the mean time, many of their favorable features are preserved. Chapter 4 will present the data modeling process with bivariate and trivariate simplex spline forms. Chapter 5 will introduce Interval Simplex Splines (ISS) to handle the uncertainty associated with physical data to ensure the bounded error in the databases. Based on the generated databases, Chapter 6 will present and develop the marching simplices algorithm for visualizing the data and facilitating further interrogation on the data to help understand the physical phenomena encapsulated in the data. Chapter 7 will present conclusions and recommendations for future research. Finally, Appendix A will discuss Delaunay triangulation and constrained Delaunay triangulation which will be used to form the initial triangulation for data modeling; Appendix B will present a method to determine which simplex in a 2D (3D) triangulation contains a given point, which is frequently used in preprocessing, evaluation, and data modeling procedures; and finally Appendix C will discuss the original definition of simplex splines and their geometrical interpretation.

Chapter 2

Preprocessing for Data Modeling

2.1 Introduction

Before we proceed to model a set of scattered physical data by simplex splines, we need to perform effective preprocessing on the data such that our data modeling process is not only successful, but also efficient and able to represent the data with high fidelity and reduce data storage significantly. This preprocessing procedure includes two major steps: (1) domain establishment; (2) data segmentation.

The domain of a finite set of scattered points plays an important role in 2D (3D) data representation, interrogation and visualization. Scientists and engineers often use the convex hulls as the domain of finite sets of 2D (3D) data points or simply use a 2D (3D) rectangular box enclosing all the points as the domains. Unfortunately, both convex hulls and boxes cannot capture the intuitive boundary of the shape formed by the definition of physical data points. For example, a set of physical points could be measured in a concave region with holes, while the convex hull of the set of points cannot express the concave features and distinguish the holes from the region.

Measured data are typically represented as raw data (e.g., densities) with no prior definition of the objects to be visualized. The domain establishing step creates the domain for the data and a classification/segmentation step that identifies the regions describing an object must therefore accompany the representation process.

2.2 The Domain of Data

A generalization of the convex hull concept of a finite set of planar points introduced by Edelsbrunner *et al.* [29] is described in this section and is applied in defining the domains of sets of planar points. This generalization leads to a family of straight-line graphs – *α-shapes* which appear to be able to capture the intuitive notions of *fine shape* and *crude shape* of point sets.

For convenience and clarity, part of this subsection restates and explains the definitions of α-shape from Edelsbrunner *et al.* [29]. For theoretical and in-depth coverage, readers are

referred to [29, 28, 30, 31].

2.2.1 2D α-Shape

The α-shape of a set of points is closely related to the Delaunay triangulation. Appendix A introduces the definition of Delaunay triangulation and describes the relation between α-shape and the Delaunay triangulation of a set of points.

Definition of α shape

Definition 2.1 *Given a set S of n planar points, let α be a sufficiently small but arbitrary positive real number, i.e., $\alpha \in \mathbf{R}^+$, the α-hull of S is the intersection of all closed discs with radius $r = \frac{1}{\alpha}$ that contain all the points of S. The α-hull of S is also called the generalized convex hull of S.*

In order to achieve an intersection of discs, it has to be guaranteed that there exists at least one disc of the chosen size that contains all points. This implies that the smallest possible value for $1/\alpha$ is equal to the radius of the smallest enclosing circle. In fact, it has been shown that $1/\alpha$ is no less than $3^{-1/2}$ times the diameter of S suffices, no matter how the points are distributed. See Edelsbrunner [29] and references therein.

In Figure 2-1(a), the α-hull for a particular sufficiently small positive α is depicted. Intuitively, a large (but still sufficiently small) α gives rise to hulls that have only in some sense *essential extreme points* on their boundary. As α approaches zero, the α-hull approximates the common convex hull.

Definition 2.2 *For an arbitrary negative real number α, the α-hull is defined as the intersection of all closed complements of discs (where these discs have radii -1/α) that contain all the points of S.*

Figure 2-1(b) displays such a hull for the same point set as used in Figure 2-1(a). For convenience, let us define the 0-hull as being the usual convex hull of the points and let us agree that the intersection of no discs (which may occur for a large positive α) is equal to the entire plane.

If we define a generalized disc of radius $1/\alpha$ as a disc of radius $1/\alpha$ if $\alpha > 0$, the complement of a disc of radius -$1/\alpha$ if $\alpha < 0$, and a halfplane if $\alpha = 0$, then the preceding definitions could be combined to the following definition.

Definition 2.3 *For an arbitrary real α and a set S of points in the plane, the α-hull of S is the intersection of all closed generalized discs of radius $1/\alpha$ that contain all points of S.*

Thus we have a family of α-hulls for α ranging from $-\infty$ to $+\infty$. Sample members of this family are the entire plane (for α sufficiently large), the smallest enclosing circle of S (when $1/\alpha$ equals its radius), the convex hull of S (for $\alpha = 0$), and S itself (for α sufficiently small).

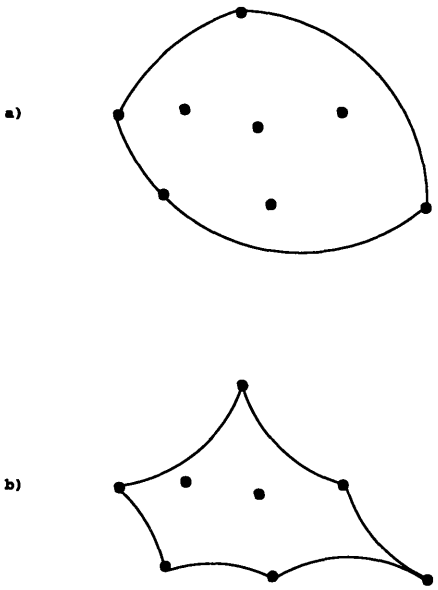


Figure 2-1: α-hull for a) $\alpha > 0$ and b) $\alpha < 0$.

Since negative α-hulls denote the domain of the points, in this thesis, we focus only on negative α-hulls. Therefore, in the following context, we discuss only negative α-hull by using $r = -\frac{1}{\alpha}$.

Definition 2.4 *Given a set S of points in the plane and an arbitrary real α, the α-shape of S is the straight line graph of α-hull.*

To understand the α-shape intuitively and procedurally, let us look at Figure 2-2(a) which depicts a set of fourteen planar points and the Delaunay triangulation applied on the points. For a given α ($\alpha < 0$), there is a disc whose radius equal to $r = -\frac{1}{\alpha}$. Imagine that the disc is an eraser which will remove any edge larger than $2r$. Figure 2-2(b) shows the resulting triangulation after the erasing. Note that the area enclosed by the boundary of the α-shape of the points is the area that is considered as the domain of the points in this thesis.

Definition 2.5 *The domain of a set of planar points is the area enclosed by the boundary of the α-shape of a set of points.*

The α_1-hull of a set of points is contained in the α_2-hull if $\alpha_1 \leq \alpha_2$. The set of α_1-extreme points in S is a subset of the α_2-extreme points if $\alpha_1 \geq \alpha_2$. See [29] for further details. Intuitively, *relatively large* α tend to produce a rather crude shape of the points (the extreme being a chord or an inscribed triangle of the smallest enclosing circle), whereas smaller α reveal more and more details, until, as α approaches $-\infty$, all points are isolated extreme points of the shape.

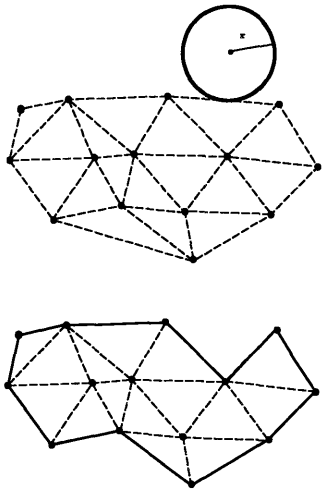


Figure 2-2: α-shape is illustrated procedurally from Delaunay triangulation. In this triangulation, two edges with lengths larger than the diameter ($2\times\frac{-1}{\alpha}$) of the disc are removed.

In our application of α-shape, we form the domain of a set of scattered data from the α-shape. Therefore, it is not desirable for the domain formed from α-shape to contain any non-manifold topology[1], e.g., a triangle adjacent to a dangling straight line segment emanating from a vertex of a triangle.

Implementation of α-Shape

As mentioned in previous sections, only "negative" α-shapes are used in this thesis in forming domains of points. Therefore, we can simplify the implementation. In the following text on domain creation, we omit the definition and implementation of (constrained) Delaunay triangulation which are addressed in Appendix A.

1. Input: (1) the number of points; (2) the x and y coordinates of the points; (3) radius R $(-\frac{1}{\alpha})$.
2. Remove redundant points from the point array.
3. Apply Delaunay triangulation on the points.
4. For each edge in the triangulation, if the length of the edge is larger than $2R$ then remove this triangle.
5. If any isolated point or incomplete triangle appears, the program will ask the user to increase R and go to Step 4.

[1]Non-manifold is also called non-two manifold.

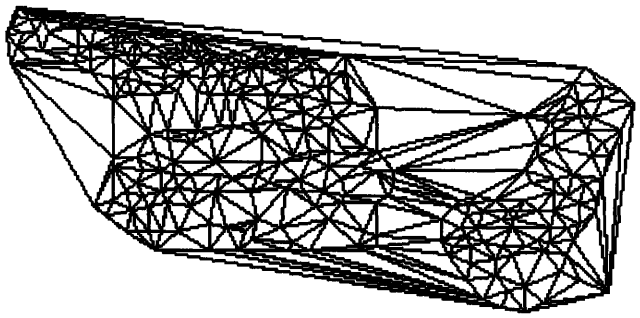


Figure 2-3: The Delaunay triangulation of a set of 303 planar points.

6. If users are satisfied with the α-shape, then proceed; otherwise, users may change R and go to Step 4.

7. Output the boundary points of α-shape and their topology.

Examples of α-Shape

In this section, we present examples of the α-shape at different values R for a set of 303 planar points. The x and y coordinates for this set of data are taken from data by Edelsbrunner *et al.* available by anonymous ftp the World Wide Web at ftp.ncsa.uiuc.edu. The program is implemented in C++ and runs on a Silicon Graphics workstation with OpenGL and Open Inventor. Figure 2-3 shows the Delaunay triangulation of the set of planar points. Figure 2-4 shows the α-shapes at $R = 60(\alpha = -\frac{1}{60})$, $R = 100(\alpha = -0.01)$ and $R = 4000(\alpha = -0.00025)$, respectively. It can be seen that as R increases, the shape of the set of points becomes cruder. Finally, Figure 2-5 shows the constrained Delaunay triangulation on the planar points; the constraints in this case are the boundary of the α-shape. The constrained Delaunay triangulation (described in Appendix A) based on the boundary points of α-shapes and some interior points will be further developed and used in the next section for data segmentation which is a necessary step for efficient data fitting.

The computational procedure for the 2D α-shape involves the construction of a 2D Delaynay triangulation over a set of n points, which needs $\mathcal{O}(nlogn)$ time; the time complexity of the computation needed for removing the triangles is $\mathcal{O}(n)$. Thus, the overall asymptotic time complexity of the 2D α-shape is $\mathcal{O}(nlogn)$.

2.2.2 3D α-Shape

3D α-shape is the natural extension of 2D α-shape into 3D Euclidean space. See Edelsbrunner [28, 30] for a detailed description of 3D α-shape.

Let S be a finite set of points in $\mathbf{R}^3$ and α a real number with $0 \leq \alpha \leq \infty$. The

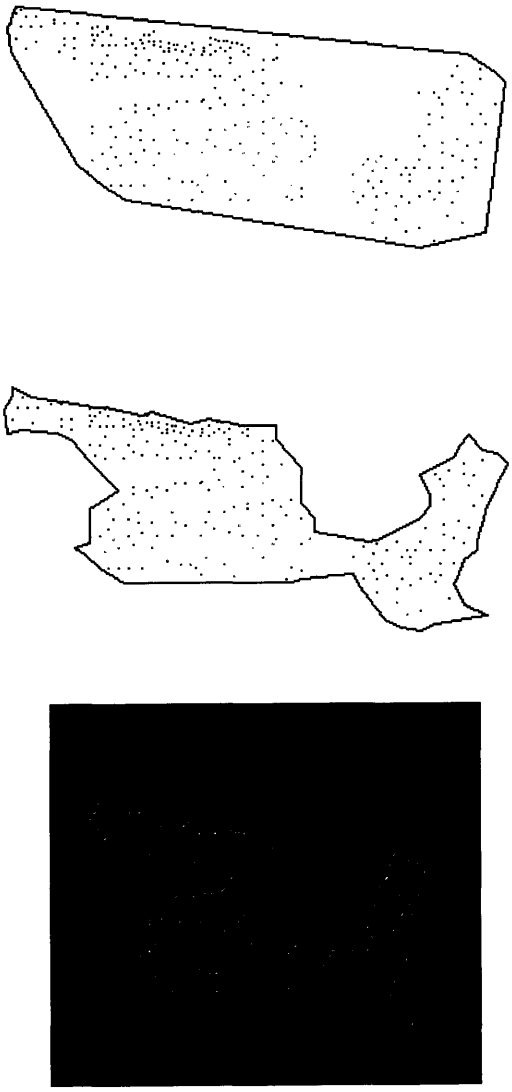


Figure 2-4: The α-shapes for a set of 303 points at $R = 4000$ ($\alpha = -0.00025$), $R = 60$ ($\alpha = -\frac{1}{60}$), and $R = 10$ ($\alpha = -0.1$), from top to the bottom.

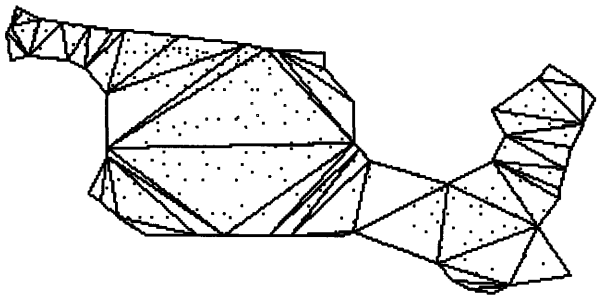


Figure 2-5: The constrained Delaunay triangulation of a set of planar points.

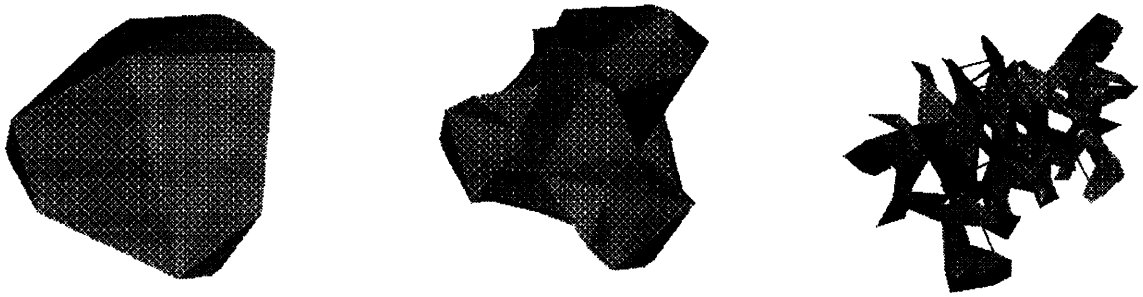


Figure 2-6: The α-shapes of a set of 3D data at different α values. The left picture shows the convex hull of the data when α is large, in which case the convex hull is equal to the α-shape; the middle and right pictures show the α-shapes when α decreases (adapted from [30]).

α-shape of S is a polytope [2] which is neither necessarily convex nor necessarily connected. For $\alpha = \infty$, the α-shape is identical to the convex hull of S. However, as α decreases, the α-shape shrinks by gradually developing cavities. These cavities may join to form tunnels, and even holes may appear (see Figure 2-6).

Intuitively, a piece of the polyhedron disappears when α becomes small enough so that a sphere with radius α, or several such spheres, can occupy its space without enclosing any of the points of S. Think of $\mathbf{R}^3$ filled with *styrofoam* and the points of S made of more solid material, such as a rock. Now imagine a spherical eraser with radius α. It is omnipresent in the sense that it carves out *styrofoam* at all positions where it does not enclose any of the

[2] Polytope is the analog in n-dimensional space of point, segment, polygon and polyhedron in a space of dimension 0, 1, 2 and 3. Later in this thesis, we simply use the term *polyhedron* to replace a 3D *polytope*.

sprinkled rocks, that is, points of S. The resulting object will be called the α-hull. To make things more feasible, we straighten the surface of the object by substituting straight edges for the circular ones and triangles for the spherical caps. The resulting object is the α-shape of S (see Figure 2-6). It is a polyhedron in a fairly general sense: it can be concave and even disconnected, it can contain two-dimensional patches of triangles and one-dimensional strings of edges, and its components can be as small as single points. (Since in this thesis, we forbid the presence of non-manifold geometry for the sake of a meaningful domain for data points, therefore, α cannot be too small to create isolated points, edges or faces.)

For $0 < \alpha < \infty$, let an α-ball be an open ball with radius α. For completeness, a 0-ball is a point and an ∞-ball is an open half-space. An α-ball b is empty if $b \cap S = 0$, where S is the set of 3D points. Any subset $T \subseteq S$ of size $|T| = k + 1$, with $0 \leq k \leq 3$, defines a k-simplex that is the convex hull of T. The general position assumption assures that all k-simplices σ_T are properly k-dimensional. For $0 \leq k \leq 2$, a k-simplex σ_T is said to be α-exposed if there is an empty α-ball b when $T = \partial b \subset S$, where ∂b is the sphere or plane bounding b. A fixed α thus defines sets $F(k, \alpha)$ of $\alpha - exposed$ k-simplices for $0 \leq k \leq 2$. The α-shape of S, denoted by S_α, is the polyhedron whose boundary consists the triangles in $F(2, \alpha)$, the edges in $F(1, \alpha)$, and the vertices in $F(0, \alpha)$. The k-simplices in $F(k, \alpha)$ are also called the k-faces of S_α.

Definition 2.6 *The domain of a set of 3D points is the volume enclosed by the boundary faces of the* α**-shape** *of a set of points.*

The implementation of the 3D α-shape and the relation between 3D α-shape and Delaunay triangulation are analogous to the 2D context and thus are omitted here. Readers are referred to references [29, 30] and software available through anonymous ftp on the World Wide Web at ftp.uiuc.edu.

The time complexity for construction of the 3D Delaunay triangulation over a set of n points is $\mathcal{O}(n^2)$ in the worst case. On average, it seems to be roughly proportional to $\mathcal{O}(n(logn)^2)$, see [30]. The overall time complexity for 3D α-shape is $\mathcal{O}(n^2 + mlogm)$ where m is the number of simplices formed by the Delaunay triangulation.

2.3 Data Segmentation

Data segmentation is an important process to extract critical features from physical data. It is a necessary preprocessing step towards data modeling for scientific databases. This preprocessing step extracts data by identifying the features of the underlying surface and is of potential interest to physical data interpretation. The data segmentation technique in this thesis subdivides the domain into triangles or tetrahedra such that (1) in each simplex, there are enough number of data points for fitting, e.g., at least six data points in each triangle when using quadratic simplex spline function in 2D data modeling; (2) the shape of the domain triangles is nice, i.e., the minimum angle in the triangulation is no less than a predefined real number and the maximum is no greater than $\frac{\pi}{2}$ [67]; (3) points representing

function values in the same simplex do not have large difference in distance to a linear faceted triangular approximating surface.

The data modeling operations (presented in the next Chapter) will be based on the triangulation formed by these triangles or tetrahedra and will be facilitated by the data segmentation process.

Automatic segmentation systems have emerged only for very specific tasks and data. At present, there are no general methods that could handle, for example, unstructured data with arbitrary population distribution. Adaptive and efficient methods for data segmentation are an important step towards good fitting results. Blindly segmented data may result in unnecessarily complex fitting procedures and infidelity in fitting results. Figure 2-7 shows a set of 1D data in an interval $[a, b]$ along x axis and different y coordinates. If we segment the data with binary sections on the interval [a, b] and use a linear function to fit the data in each subinterval, it can be seen that many unnecessary pieces are created; if we adaptively segment the data at some significant (critical) points, a smaller number of segments are created and less coefficients are needed; thus, a good fitting result with decreased memory requirements can be achieved.

For a 2D (3D) problem, if a real valued function is defined over a large planar (spatial) domain we can access geometrically interesting areas on the function's graph represented in a piecewise linear fashion. This computation for the approximation can be performed rapidly. It can be used efficiently to detect areas of interest requiring refined representation. Therefore, those areas may need the allocation of higher data density implying deeper subdivision levels if we employ hierarchical techniques to subdivide the data into appropriate region(s) within which nonlinear simplex spline fitting methods can be profitably used, (see also Samet [81]).

2.3.1 2D Data Segmentation

2D data segmentation can be achieved by combining linear approximation and Delaunay triangulation on the boundary nodes and interior nodes. Two triangulations are used in data segmentation, one is the triangulation w.r.t. data points, in which each data point is a vertex on the triangulation, called point triangulation; the other one is the domain triangulation, also called current triangulation, which is the one over which we define the fitting function.

The basic steps in 2D data segmentation include: filtering, inserting nodes, and smoothing.

Filtering 2D Raw Data

The input data ready for data segmentation is raw data which inevitably has uncertainties, high frequency noise and outliers. It is desirable to use filters to remove high frequency noise so that features are captured which are representation of the function and not the measurement error.

To remove high frequency noise, for every data point $\mathbf{p}_i = (x_i, y_i, f_i)$, we calculate the

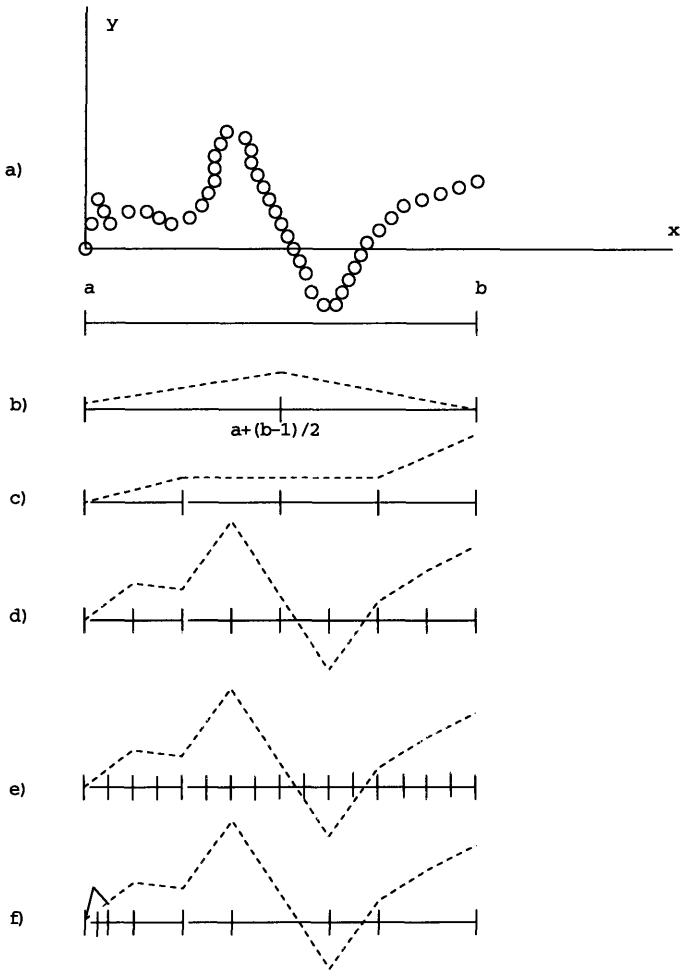


Figure 2-7: 1D data segmentation: a) – e) show when using straight line segments to fit a set of planar points, bisecting the domain causes unnecessarily many segments; f) shows adaptively subdividing the domain results in reasonable sub-intervals for the fitting

Figure 2-8: The bathymetric data from an area of Charles river, Boston, MA, USA.

weighted average of the data point and the points in the triangle neighborlet[3] for this data point. Suppose the number of points in the triangle neighborlet is m, and the points in the neighborlet are denoted as $\mathbf{p}_j$, where $j = 1, ..., m$, then the weighted average of $\mathbf{p}_i$ is:

$$\tilde{f}_i = \frac{2}{m+2} f_i + \frac{1}{m+2} \sum_{j=1, j \neq i}^{m} f_j \tag{2.1}$$

Finally, we use $\tilde{f}_i$ to replace f_i in data segmentation process shown in the following subsection.

Figure 2-9 shows a map of the Charles river (Boston, Massachusetts, USA) bathymetryic data before filtering and after the filtering. The data is measured by commercially available GPS (Global Positioning System) and depth sounder. The data was gathered by Ms. J. Brener and Ms. M. Frey during the 1993 MIT Sea Grant Summer Undergraduate Workshop in AUVs under the supervision of Dr. D. K. Atwood and Dr. S. T. Tuohy. The number of data points is 2201 and is shown in Figure 2-8.

Inserting nodes

The following steps illustrate the segmentation procedure.

1. Input the polygonal boundary of the domain of the data.
2. Place nodes at all boundary vertices.

[3] A triangle neighborlet includes the current triangle and the triangles sharing at least one vertex with the current triangle.

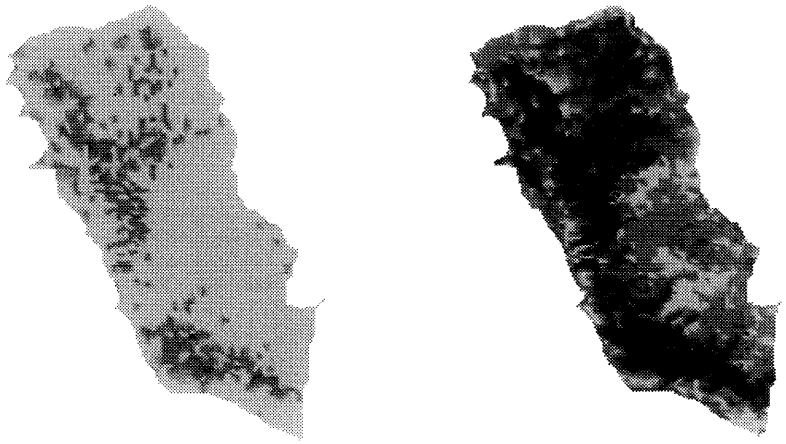


Figure 2-9: The filtered and unfiltered data of the Charles river. The figure at the right side is the raw data of an area of the Charles river; the figure at the left side is the same set of data after filtering.

3. Apply constrained Delaunay triangulation (introduced in Appendix A) on the nodes. The constraints are the boundary of the domain.

4. The vertices of the triangles are themselves data points with coordinates (x_i, y_i) and associated with scalar values f_i. Thus we can form a piecewise linear function $f(x, y)$ over the triangulated nodes by interpolating function values at the vertices of every triangle.

5. Calculate the absolute distance between every data point and the piecewise linear interpolation function in the direction of the function value axis, i.e., $d_i = \tilde{f}_i - f(x_i, y_i)$, and find the maximum absolute distance. For efficiently determining which triangle contains a certain data point, an algorithm presented in Appendix B may be used.

6. If the maximum absolute distance is less than a predefined threshold value, stop segmentation process; otherwise proceed.

7. Place a node at the planar point with maximum absolute distance and go to Step 4.

Smoothing Boundary

The α-shape as the boundary of the domain of data is composed of line segments. This boundary may have more vertices than necessary. For example, three consecutive nodes may be nearly coincident (too close) and/or collinear or a set of consecutive nodes may form a zig-zag shape on the boundary. (See Figures 2-10 and 2-11.) Without smoothing these nodes, the triangles formed on these nodes may be too small to contain data points and slivers may appear in the triangulation.

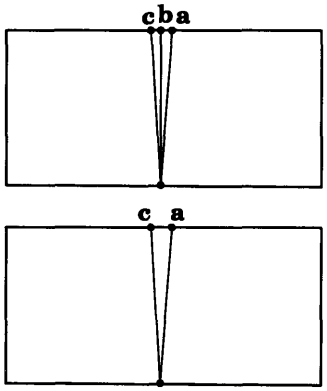


Figure 2-10: The smoothing of the boundary of a domain. The upper figure shows three consecutive nodes a, b, and c which are too close and collinear. The lower figure shows the mid-node b is removed from the node list.

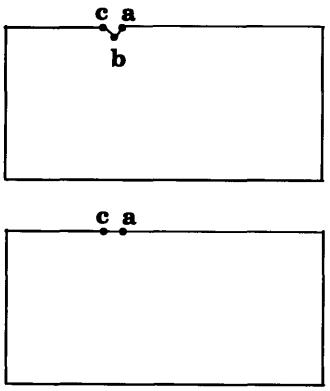


Figure 2-11: The smoothing of the boundary of a domain. The upper figure shows three consecutive nodes a, b, and c which are too close and they form a concave kink. The lower figure shows the mid-node b is removed from the node list.

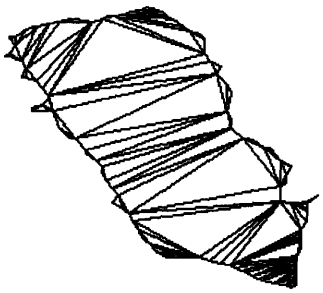


Figure 2-12: The constrained Delaunay triangulation on the boundary nodes of the α shape.

The following illustrates the criteria and methodologies for smoothing the boundary:

For a predefined threshold σ, if the summation of distances between two consecutive of three consecutive nodes $d_1 + d_2$ is less than σ, and if

1. the three nodes are collinear, or

2. the three nodes form a concave kink on the domain,

remove the mid-node from node list.

If the three nodes form a convex kink on the domain, we then remove the first and the third nodes from the node list.

The triangulation formed is the segmented triangulation for the data modeling process.

Figure 2-12 shows an example of Charles river data set and a triangulation (resulting from Step 3) based on the domain boundary nodes provided by the procedure of domain establishment in Section 2.2. Figure 2-13 shows the triangulation after one node is inserted into the interior of the domain at a point with the maximum absolute distance to the underlying linear surface. Figure 2-14 shows the triangulation after four nodes are inserted into the interior of the domain; and Figure 2-15 shows the triangulation we will use for data modeling later in this thesis.

2.3.2 3D Data Segmentation

3D data segmentation can be achieved by combining linear approximation and Delaunay triangulation on the boundary nodes and interior nodes of the domain. The basic steps in 3D data segmentation are analogous to those in 2D segmentation, including filtering, inserting nodes, smoothing boundary.

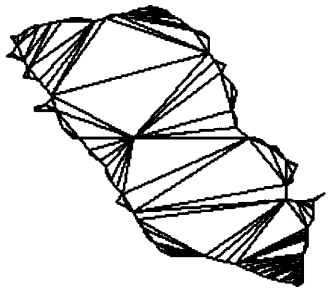


Figure 2-13: The constrained Delaunay triangulation after one interior node is inserted.

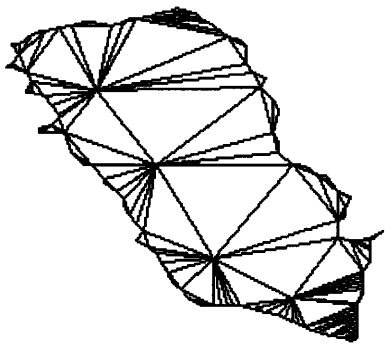


Figure 2-14: The constrained Delaunay triangulation after four interior nodes are inserted.

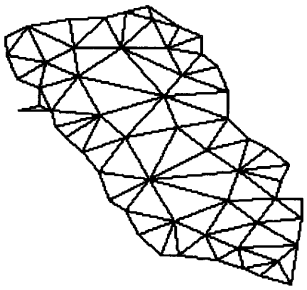


Figure 2-15: The triangulation ready for data modeling.

Filtering 3D Raw Data

To remove high frequency noise, for every data point $\mathbf{p}_i = (x_i, y_i, z_i, f_i)$, we calculate the weighted average of the data point and the points in the tetrahedron neighborlet[4] for this data point. Suppose the number of points in the tetrahedron platelet is m, points in the platelet are denoted as $\mathbf{p}_j$, where $j = 1, ..., m$, then the weighted average of $\mathbf{p}_i$ is:

$$\tilde{f}_i = \frac{2}{m+2} f_i + \frac{1}{m+2} \sum_{j=1, j\neq i}^{m} f_j \tag{2.2}$$

Finally, we use $\tilde{f}_i$ to replace f_i in data segmentation process shown in the following subsection.

Inserting nodes

The following steps illustrate this segmentation procedure.

1. Input the polyhedral boundary of the domain of the data.
2. Place nodes at all boundary vertices.
3. Apply 3D constrained Delaunay triangulation on the nodes. The constraints are the boundary facets of the domain.
4. Construct a piecewise linear trivariate interpolation function $f(x, y, z)$ over each tetrahedron based on the function values of the four data points.
5. Calculate the absolute distance between every data point and the piecewise linear interpolating function in the direction of the function value axis, i.e., $d_i = \tilde{f}_i - f(x_i, y_i, z_i)$ and find the maximum absolute distance. In determining which tetrahedron contains a certain data point, an algorithm presented in Appendix B may be used.
6. If the maximum absolute difference is less than a predefined a threshold value, stop segmentation; otherwise proceed.
7. Insert a node at the point with maximum absolute distance and go to Step 3.

Smoothing Boundary

The boundary of the domain resulting from α-shape may have more vertices than necessary. The following illustrates the criteria and methodology for smoothing the boundary.

For a predefined threshold σ, we check every boundary node v_i and the boundary triangular facets incident to v_i. If the area of one such facet is smaller than σ, and if

[4] A tetrahedron neighborlet includes the current tetrahedron and the tetrahedra sharing at least one vertex with the current tetrahedron.

1. the nodes of these triangular facets are co-planar, or

2. these facets form a non-convex kink at node v_i,

then we remove the node v_i from the node list.

If the facets form a convex kink on the domain, we then keep the node v_i and remove all the other boundary nodes which connect to node v_i.

The 3D constrained Delaunay triangulation on rest of the the nodes in the node list will be used for data segmentation. The definition of 3D constrained Delaunay triangulation can be found in Appendix A.

The triangulation formed will serve as the initial segmented triangulation for the 3D data modeling process.

2.4 Topological Modeling Structure

In this section, we introduce a data structure called Topological Modeling Structure (TMS) and apply it in representing the 2D (3D) triangulation domains to facilitate data modeling, visualization and interrogation. Without a topological data structure, it is difficult to maintain the adjacency information between domain simplices, especially for higher dimensional problems. Additionally, a TMS can facilitate the solid and surface representation and interrogation process. Later, in this thesis we will also use this data structure to represent polygonalized implicit or isovalued surfaces and contour lines to help gain insight into the data.

Brisson [6] presents a data structure for describing subdivisions of n-dimensional manifolds. This subdivision concept is called the *cell-tuple structure* because it contains a collection of topological cells of different dimension. A cell of dimension n is homeomorphic to the solid n-dimensional unit disc. The quad-edge structure for subdivisions of two-manifolds pioneered by Guibas and Stolfi [52] may be represented by an appropriate specialization of the cell-tuple structure. No restrictions are posed to the underlying manifold C. In other words, C can be embedded in the $n+k$-dimensional Euclidean space. C also can be a manifold with boundary. The cell-tuple structure uses an incidence graph whose nodes are cells of the set C. This structure is easy to implement in practice because of its hierarchical nature, conceptual recursive simplicity, and small number of key processing algorithms. See Bardis and Patrikalakis [3] for a recent detailed review of the relevant literature in this general area.

To obtain the topological relation between facets in different cells, we use the cell-tuple data structure to represent the cells in the domain and the generated polygonalized implicit surfaces or isosurfaces and contour lines.

The three main modules of the TMS are:

- *Topological data structure*:
  The whole structure is organized in incidence graphs of topological manifolds, illustrated here by the example shown in Figure 2-16. This incidence graph is rooted,

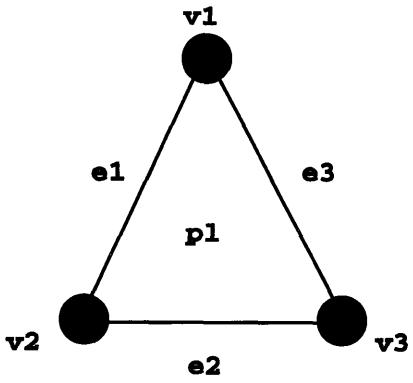


Figure 2-16: A triangle.

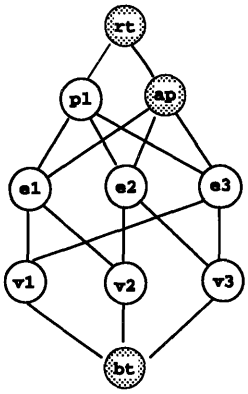


Figure 2-17: The triangle's incidence graph.

and each node represents a topological cell. Nodes at the same level represent cells of the same dimension. Cells represented by nodes on different levels are connected according to their incidence relationships. The incidence graph of a triangle is shown in Figure 2-17. In this graph, **rt** indicates the *root* of the graph; **p1** is the triangular face (of dimension 2) and **ap** is the complement of the face; edge nodes (of dimension 1) and point nodes (of dimension 0) are at corresponding levels; the lowest level **bt** is of dimension -1 which indicates the *bottom* of the graph.

- *Topological operators*:

  These can be classified into two groups: one group contains low-level operators, while the other group contains high-level operators. Low-level operators create and interrogate the incidence graph, while high-level operators manipulate the models represented by incidence graphs.

- *Geometric and attribute data structure*:

The topological data structure (that is, the incidence graph) only specifies the relationships between cells, not the geometry information of each cell, for example, its geometric shape or attribute information (e.g., geophysical properties). Therefore, we also need geometric and attribute data structures to store such information for each cell. Such structures can be effectively connected to the topological structure.

The theoretical foundation of the TMS is Brisson's work [6]. We employ the concept of the cell-tuple structure in this work. This structure yields a subdivision of a manifold in cells of different dimensions. The incidence relation between those cells is reflected in the incidence graph representation of the respective manifold.

Chapter 3

Simplex Spline Representation and Evaluation

In this chapter, we introduce the definition, evaluation, and properties of bivariate and trivariate simplex spline functions. Simplex spline functions will be used to model physical data in our database system because of their appropriate properties, for instance:

- capability in modeling irregular domains;
- capability in achieving optimal[1] and automatic smoothness.

3.1 Bivariate Simplex Splines

This section first introduces the mathematical background of bivariate simplex splines, see also [42, 50, 85, 20, 43] and the references therein. We will then develop efficient evaluation schemes for simplex spline surfaces.

3.1.1 Mathematical Background

Basic Definitions

Definition 3.1 *Given* $\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2 \in \mathbf{R}^2$, *we denote by* $\triangle = [\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2]$ *their closed convex hull, and by* $\triangle^o = int[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2]$ *the interior of the convex hull. The* **half-open 2-simplex** $[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)$ *is then defined as follows: Let* ξ *be the horizontal unit vector in* $\mathbf{R}^2$. *A point* $\mathbf{u} \in R^2$ *belongs to the half-open 2-simplex* $[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)$ *if and only if there exists a vector* η *with positive slope, and a positive scalar* $\epsilon > 0$, *such that the set*

$$A_{(\xi,\eta,\epsilon)}(\mathbf{u}) = \{s\xi + t\eta | 0 < s, t, 0 < s + t < \epsilon\} \tag{3.1}$$

is completely contained in $[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)$.

[1] A simplex spline of degree n may be up to C^{n-1} continuous everywhere.

Figure 3-1: Two half-open 2-simplices. All interior points are in $[\mathbf{t}_i, \mathbf{t}_j, \mathbf{t}_k)$; points on dark boundary edges are in $[\mathbf{t}_i, \mathbf{t}_j, \mathbf{t}_k)$; other boundary points do not belong to $[\mathbf{t}_i, \mathbf{t}_j, \mathbf{t}_k)$.

According to this definition, all interior points in a triangle belong to the triangle's $[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)$; making a ray cast from a boundary point toward the positive x axis, if this ray does not hit any point on the triangle, then this boundary point is not in $[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)$; otherwise, this boundary point is in $[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)$. Although this inconsistent treatment for points on the triangle boundary is dissatisfying from a purist's point of view, researchers in graphics often prefer this behavior, because when partitioning a region into many triangles, *every point will be in exactly one triangle.* This definition ensures that the simplex spline basis function (which will be introduced in the following definitions) cannot be calculated multiple times at a point and thus avoids incorrect results.

Definition 3.2 *Every polygon P (of m vertices) may be partitioned into simplices by the addition of (zero or more) diagonals such that the intersection of any two simplices in the* **triangulation** *T is either an empty set, a common vertex, or a common edge. The direction of T is defined as clockwise/counterclockwise if every triangle's vertices $[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2]$ in T are organized clockwise/counterclockwise.*

Definition 3.3 *The barycentric coordinates $(\lambda_0, \lambda_1, \lambda_2)$ of a point $\mathbf{u}$ with respect to a triangle $\triangle = (\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)$, where $\mathbf{t}_i \in \mathbf{R}^2$, and points $\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2$ are affinely independent, are defined as:*

$$\lambda_0(\mathbf{u}|\triangle) = \frac{area(\mathbf{u}, \mathbf{t}_1, \mathbf{t}_2)}{area(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)}; \quad \lambda_1(\mathbf{u}|\triangle) = \frac{area(\mathbf{t}_0, \mathbf{u}, \mathbf{t}_2)}{area(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)}; \quad \lambda_2(\mathbf{u}|\triangle) = \frac{area(\mathbf{t}_0, \mathbf{t}_1, \mathbf{u})}{area(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)}. \tag{3.2}$$

It can be proven that

$$\mathbf{u} = \sum_{j=0}^{2} \lambda_j(\mathbf{u}|\triangle)\mathbf{t}_j, \quad \sum_{j=0}^{2} \lambda_j(\mathbf{u}|\triangle) = 1. \tag{3.3}$$

$$area(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2) = \frac{1}{2} det \begin{pmatrix} 1 & 1 & 1 \\ \mathbf{t}_0 & \mathbf{t}_1 & \mathbf{t}_2 \end{pmatrix} \tag{3.4}$$

provided that the three points $\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2$ are non-collinear, i.e. $area(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2) \neq 0$.

Barycentric coordinates play an important role in evaluation of simplex spline basis functions. Note: the points inside the triangle $\triangle$ have positive barycentric coordinates.

Definition of Simplex Splines

Definition 3.4 *For a set of m+1 planar points (knots)* $\mathbf{t}_0, ..., \mathbf{t}_m$, *the simplex spline is*

$$M(\mathbf{u}|\mathbf{t}_0, ..., \mathbf{t}_m) = \sum_{j=0}^{2} \lambda_j(\mathbf{u}|\triangle_{\mathbf{t}_{i_0},\mathbf{t}_{i_1},\mathbf{t}_{i_2}}) M(\mathbf{u}|\mathbf{t}_0, ..., \hat{\mathbf{t}}_{i_j}, ..., \mathbf{t}_m) \tag{3.5}$$

Note: $\hat{\mathbf{t}}_{i_j}$ *means* $\hat{\mathbf{t}}_{i_j}$ *is removed from the knot set.*

$$M(\mathbf{u}|\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2}) = \frac{\mathcal{X}_{[\mathbf{t}_{i_0},\mathbf{t}_{i_1},\mathbf{t}_{i_2})}(\mathbf{u})}{|\ det(\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2})\ |} \tag{3.6}$$

where

$$\mathcal{X}_{[\mathbf{t}_{i_0},\mathbf{t}_{i_1},\mathbf{t}_{i_2})}(\mathbf{u}) = \begin{cases} 1 & \text{if } \mathbf{u} \in [\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2}) \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

is the characteristic function on the half-open 2-simplex $[\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2})$.

$\triangle_{\mathbf{t}_{i_0},\mathbf{t}_{i_1},\mathbf{t}_{i_2}} = \{\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2}\} \subseteq \{\mathbf{t}_0, ..., \mathbf{t}_m\}$, *called* **reference triangle**, *is any subset of affinely independent 3 points and* $\lambda_j(\mathbf{u}|\triangle_{\mathbf{t}_{i_0},\mathbf{t}_{i_1},\mathbf{t}_{i_2}})$ *are the barycentric coordinates of* $\mathbf{u}$ *with respect to* $\triangle_{\mathbf{t}_{i_0},\mathbf{t}_{i_1},\mathbf{t}_{i_2}}$.

Equation (3.5) is independent of the specific choice of $\triangle_{i_0,i_1,i_2}$ [84].

The geometrical interpretation and the properties of simplex splines is shown in Appendix C.

Definition of Simplex Spline Surfaces

Definition 3.5 *Let* $T = \{\triangle(I) = [\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2}] | I = (i_0, i_1, i_2) \in I \subseteq Z_+^3\}$ *be an arbitrary triangulation of the parameter plane* $\mathbf{R}^2$. *Assign a sequence of knots* $\mathbf{t}_{i,1}, ..., \mathbf{t}_{i,n}$ *to every vertex* $\mathbf{t}_i$ *in this triangulation and* $\mathbf{t}_{i,0} = \mathbf{t}_i$ *such that any three knots associated with one domain triangle form a proper triangle. A vertex and its associated knots are called a* **knot-cloud.**

Then **simplex spline surfaces** *are defined under the following two conditions:*

1. *The triangles in the domain triangulation have the same orientation, e.g. counter-clockwise.*

2. *Define region:*

$$\Omega_\beta^I = \cap_{\gamma \le \beta} |\ \triangle_\gamma^I\ |, \Omega_2^I = int\left(\cap_{|\beta|=n} \Omega_\beta^I\right) \neq \phi \tag{3.8}$$

where $\gamma = (\gamma_0, \gamma_1, \gamma_2) \le (\beta_0, \beta_1, \beta_2) = \beta$ *indicating* $\gamma_0 \le \beta_0, \gamma_1 \le \beta_1, \gamma_2 \le \beta_2$. *int(A) represents the interior of region A and* $\triangle_\gamma^I = [\mathbf{t}_{i_0,\gamma_0}, \mathbf{t}_{i_1,\gamma_1}, \mathbf{t}_{i_2,\gamma_2}]$. *This condition can be met by associating the three knot-clouds with the three vertices for a triangle in three separated circles, i.e.* $C_i \cap C_j = \phi$ *if* $i \neq j$ *for all i, j and* C_i *is a circle centered at vertex* $\mathbf{t}_i$. *Figure 3-2 illustrates this condition for one domain triangle.*

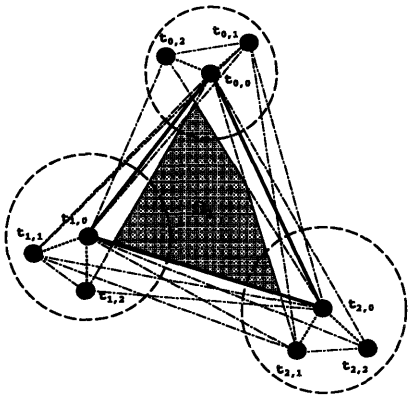


Figure 3-2: The region Ω_2 formed by intersections $int\left(\cap_{|\beta|=n}\Omega\beta^I\right)$ (adapted from [20]).

Finally, consider the sets:

$$V_\beta^I = \underbrace{\{\mathbf{t}_{i_0,0},...,\mathbf{t}_{i_0,\beta_0},...,\mathbf{t}_{i_1,0},...,\mathbf{t}_{i_1,\beta_1},...,\mathbf{t}_{i_2,0},...,\mathbf{t}_{i_2,\beta_2}\}}_{n+3} \tag{3.9}$$

where $\beta = (\beta_0, \beta_1, \beta_2)$, $|\beta| = \beta_0 + \beta_1 + \beta_2 = n$ *and an arbitrary simplex spline surface of degree* n *over a given triangulation* T *is:*

$$F(\mathbf{u}) = \sum_{I \in T} \sum_{|\beta|=n} N_\beta^I(\mathbf{u}) c_\beta^I, \tag{3.10}$$

where $c_\beta^I \in R^3$ *are the control points for the shape of the surface and*

$$N_\beta^I(\mathbf{u}) = |det(\mathbf{t}_{i_0,\beta_0}, \mathbf{t}_{i_1,\beta_1}, \mathbf{t}_{i_2,\beta_2})| \cdot M(\mathbf{u} \mid V_\beta^I) \tag{3.11}$$

is called the normalized simplex splines basis function. These are the blending functions for the simplex spline scheme. It has been shown in [85, 20] that:

$$\sum_{I \in T} \sum_{|\beta|=n} N_\beta^I(\mathbf{u}) = 1. \tag{3.12}$$

So the normalized simplex spline basis functions form a partition of unity.

If c_β^I are scalar numbers, Equation 3.10 generates a **bivariate simplex spline function.** This function can be used in aplications, e.g., 2D data modeling.

In ensuring region $\Omega_2^I \neq \phi$, we can have:

$$\sum_{I \in P_\mathbf{u}} \sum_{|\beta|=n} N_\beta^I(\mathbf{u}) = 1. \tag{3.13}$$

where $P_\mathbf{u}$ denotes the triangle neighborlet of $\triangle$ which contains $\mathbf{u}$.

Number of items needed for one basis function : M(u)			
	base conditions	lower degree basis functions	bar. coords.
Quadratic	3^2	$3^1 + 3^2$	$3^1 + 3^2$
Degree n	3^n	$\sum_{i=1}^{n} 3^i$	$\sum_{i=1}^{n} 3^i$
Number of items needed for one point over one triangle : F(u)			
	base conditions	lower degree basis functions	bar. coords.
Quadratic	6×9	6×12	6×12
Degree n	$m_c 3^n$	$m_c \sum_{i=1}^{n} 3^i$	$m_c \sum_{i=1}^{n} 3^i$

Table 3.1: The number of items needed for evaluation of a bivariate simplex spline basis function and a point on a simplex spline function, where $m_c = \frac{(n+1)(n+2)}{2}$.

Thus, we have

$$F(\mathbf{u}) = \sum_{I \in P_{\mathbf{u}}} \sum_{|\beta|=n} N_\beta^I(\mathbf{u}) c_\beta^I. \tag{3.14}$$

This equation provides a linear system with a banded matrix when we apply simplex spline functions in approximating a set of data points in Chapter 4.

3.1.2 Evaluation of Bivariate Simplex Spline Functions

It is important to evaluate simplex spline basis functions efficiently since a substantial number of them (including lower degree ones) are computed in evaluating a single point on a bivariate simplex spline function. See Table 3.1 for a brief review, where "base condition" means the calculation the the characteristic function $\mathcal{X}$.

From Table 3.1, at the worst case, the evaluation of a quadratic simplex spline basis function needs nine base cases in calculating $\mathcal{X}$, i.e., determining if a point is in a half-open triangle; 12 lower degree basis functions; and 12 barycentric coordinates. The evaluation of a point over a domain triangle needs more operations. Therefore, it is desirable to have a scheme which can efficiently evaluate simplex spline functions.

Efficient evaluation of a simplex spline basis function includes the following approaches:

1. Efficiently determining the domain triangle in which the point $\mathbf{u}$ is enclosed.

2. Efficiently determining if a point $\mathbf{u}$ is in a half-open triangle; in addition, this operation is numerically sensitive for points on the boundaries of the triangle. It is necessary to develop a simple and numerically stable algorithm to perform these operations. The algorithm should also be able to serve in its higher dimensional counterparts, e.g., trivariate cases.

3. A single basis function over a set of knots is defined by recursive Equation 3.5. Some lower degree basis functions in the recursive formula are reusable and can serve as intermediate terms for several higher degree basis function evaluations. This reusability is of the same spirit as tensor product B-spline evaluation.

4. $\sum_{i=1}^{n} 3^i$ barycentric coordinates are needed during each basis function of degree n computation. It consumes much time if we compute all these barycentric coordinates especially in higher dimensional cases (e.g., trivariate basis function). In fact, some lower degree basis functions obtained are zero, in which cases it is not necessary to calculate the corresponding barycentric coordinates [41].

Determining the triangle enclosing a point

The triangle in which the point $\mathbf{u} = (x, y)$ is enclosed must be found. We take here a *bucket sorting* approach stimulated by the one presented in [11]. The detail of our algorithm is illustrated in Appendix B. The main advantage of this algorithm is that, as long as the buckets and the relation between the buckets and the triangles are determined, it can be used for processing many points, each with a limited number of *triangle-point* checks. Thus the algorithm takes $O(N)$ time, where N is the number of triangle vertices.

Determining a point in an half-open 2-simplex

The base case in Definition 3.4 needs to determine if a point is in a half-open 2-simplex. 3^n such operations are needed for calculating one basis function of degree n. Thus, how to determine efficiently if a point is in a half-open 2-simplex is presented here which utilizes lower dimensional half-open intervals. We will see later that this operation will be used for higher dimensional simplex spline evaluation.

The following pseudocode will take in an array of three planar points $\mathbf{v}[0]$, $\mathbf{v}[1]$, $\mathbf{v}[2]$ and a planar point $\mathbf{u}$ and determine if $\mathbf{u} \in [\mathbf{v}[0], \mathbf{v}[1], \mathbf{v}[2])$. The code first translates the triangle so that $\mathbf{u}$ will be the origin. (See Figure 3-3.) Then we project the edges of the triangle onto the y-axis. Thus, on the y-axis, the three edges correspond to three intervals. If the origin is not in any of the three half-open intervals, we conclude that $\mathbf{u}$ is not in the half-open triangle; otherwise, the projection of $\mathbf{u}$ must be in only two of the three intervals. Therefore, we can exclude one ray-edge intersection check immediately. The strategy of converting such a 2D problem into 1D problems can be extended into its 3D or even higher dimensional counterparts; and the benefit will be more obvious there.

In the pseudocode, a loop over all edges $e(\mathbf{v}[i], \mathbf{v}[j])$ checks whether the origin is in the half-open interval of e projected on the y-axis. If so, then we calculate the edge's function values f_0 and f_1 at (0,0) and (1,0) and check if f_0 and f_1 have opposite signs or if $|f_1|$ is closer to the edge e. If so, then we increase the number of crossings N_c by one; otherwise proceed to the next edge. At the end of the routine, if N_c is an odd number, we conclude $\mathbf{u} \in [\mathbf{v}[0], \mathbf{v}[1], \mathbf{v}[2])$; otherwise $\mathbf{u} \notin [\mathbf{v}[0], \mathbf{v}[1], \mathbf{v}[2])$.

IN_HO_TRIANGLE (**v**, **u**)

```
1   for i ← 0 to 3
2       v[i] ← v[i] - u;
3   N_c ← 0;
4   for i ← 0 to 3
5       j ← (i+1) mod 3;
6       if(v[i] and v[j] cross x-axis)
7           f_0 ← the function value of the edge at (0,0)
8           f_1 ← the function value of the edge at (1,0)
9           if(f_0 • f_1 < 0)
10              N_c ← N_c + 1;
11          else if(|f_0| > |f_1|)
12              N_c ← N_c + 1;
13  return N_c mod 2;
```

Here "x mod n" means that x modulo n.

Calculating Basis Functions Efficiently

Simplex spline basis functions are evaluated in recursive Equation 3.5. A bivariate simplex spline function of degree n needs $m_c = \frac{(n+1)(n+2)}{2}$ (normalized) basis functions $M_\beta, (|\beta| = 2)$ of the same degree; and each of them needs $\sum_{i=1}^{n} 3^i$ lower degree $M_\beta, (|\beta| < 2)$. For example, a quadratic bivariate simplex spline basis function needs $3 + 3^2 = 12$ lower degree M_β and a quadratic simplex spline function over a domain triangle needs $\frac{3\times4}{2} = 6$ $M_\beta, (|\beta| = 2)$. Straightforward computation for a point of the function over one triangle needs $(12+1)\times6 = 78$ $M_\beta, (|\beta| \in \{0,1,2\})$. If we assume that the knots associated with the domain triangle are in general position, then we can evaluate the simplex spline basis functions with less number of $M_\beta, (|\beta| \in \{0,1,2\})$ by adopting the same spirit as in evaluating tensor product B-spline basis functions.

For example, evaluating a quadratic simplex spline basis function $M_{\beta_0\beta_1\beta_2}$ needs only 9 lower degree M_β if we reuse some of its lower degree M_β. Figure 3-4 shows that, by fixing the reference triangles, $M_{\beta_0\beta_1\beta_2}$ can be obtained from 9 lower degree M_β instead of 12, where β_i means, in the i^{th} knot cloud, $\beta_i + 1$ knots are involved in computing $M_{\beta_0\beta_1\beta_2}$; these knots are $\mathbf{t}_{i,0}, ..., \mathbf{t}_{i,\beta_i}$. Applying this strategy to all the 6 basis functions of degree 2, more calculations on $M_\beta, (|\beta| \in \{0,1,2\})$ can be saved, see Figure 3-5.

The following rules are followed in choosing a reference triangle and computing the basis functions:

1. A reference triangle $\triangle$ for basis function evaluation is formed by three knots, each of which is from a distinct knot cloud if the knot cloud participates in the computation, i.e., $\beta_i \geq 0$. For example, to calculate M_{200}, all three knot-clouds participate in the computation and each will contribute one knot to form the reference triangle.

2. When choosing a knot from a knot-cloud to form the reference triangle, choose the one with largest index if possible. For example, in case of M_{200}, choose $\mathbf{t}_{0,2}, \mathbf{t}_{1,0}, \mathbf{t}_{2,0}$

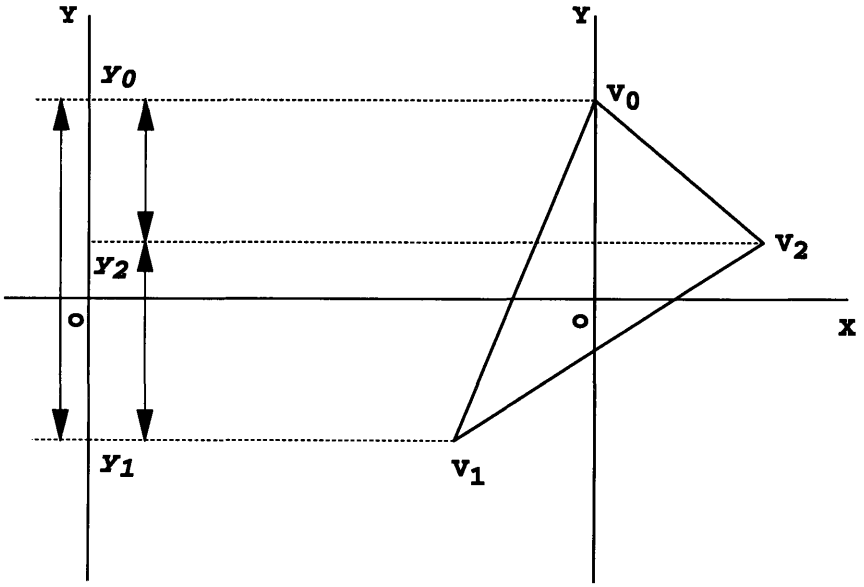


Figure 3-3: Determining if a point "o" is in the half-open triangle $[\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)$.

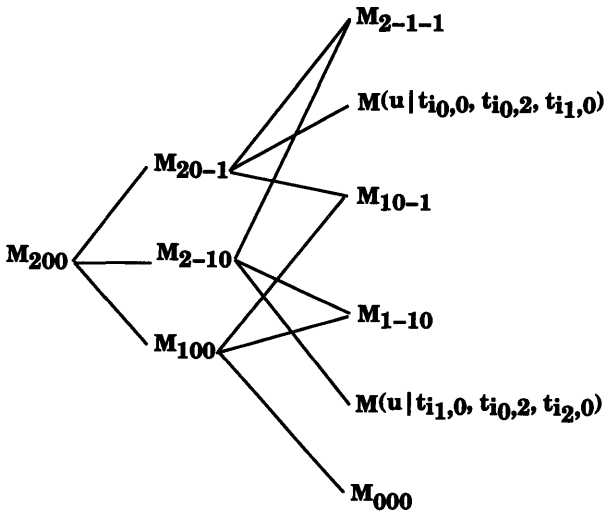


Figure 3-4: Efficient evaluation of a quadratic bivariate simplex spline basis function.

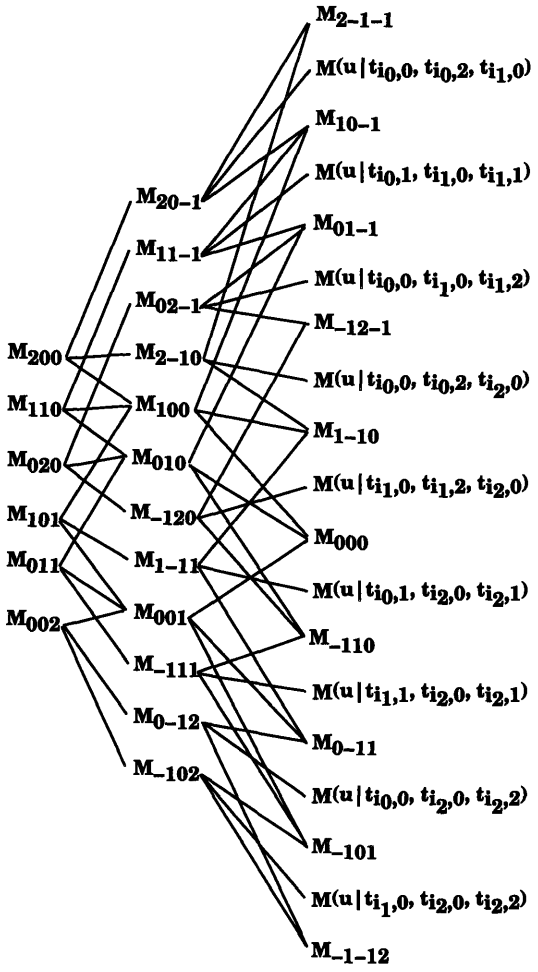


Figure 3-5: Efficiently evaluating a set of quadratic bivariate simplex spline basis functions.

as reference triangle.

3. If the i^{th} knot cloud does not contribute any knots ($\beta_i = -1$), choose knots from other knot clouds, keeping the above rules in mind. For example, in case of M_{20-1}, choose $t_{0,0}$, $t_{0,2}$, $t_{1,0}$ as the reference triangle.

For example, when calculating M_{200}, we choose $\triangle = (t_{0,2}, t_{1,0}, t_{2,0})$ as the reference triangle. Thus we can use lower degree basis functions as $M_{100}, M_{2-10}, M_{20-1}$, here index $'' - 1''$ indicates the corresponding knot cloud does not participate further in computation.

Avoiding barycentric coordinate computations

From Definition 3.4, calculating a basis function at point $\mathbf{u}$ needs $\sum_{i=1}^{n} 3^i$ (i.e., $\frac{3^{n+1}-1}{2}$) barycentric coordinates of the point $\mathbf{u}$ with respect to certain triangles, which is time consuming. In cases that the basis function is zero, then it is not necessary to calculate the corresponding barycentric coordinates λ_i, since the $\lambda_i M_\beta(\mathbf{u})$ is zero. See Fong [41].

3.1.3 Differentiation of Bivariate Simplex Spline Functions

A directional derivative along a given direction $\mathbf{d} = (d_x, d_y)$ for a parameter $\mathbf{u} \in \mathbf{R}^2$ may be computed for a bivariate simplex spline in the same fashion as evaluation. It can be shown that barycentric coordinates $\mu(\mathbf{d}|\triangle)$ for a vector $\mathbf{d}$ with respect to the triangle $\triangle = [\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2]$ sum to zero instead of one, i.e.,

$$\mathbf{d} = \sum_{j=0}^{2} \mu_j(\mathbf{d}|\triangle)\mathbf{t}_j; \quad \sum_{j=0}^{2} \mu_j(\mathbf{d}|\triangle) = 0. \tag{3.15}$$

The first order directional derivatives for degree n simplex spline basis function can then be computed as in [68, 95, 43]:

$$D_{\mathbf{d}} M(\mathbf{u}|V) = n \sum_{j=0}^{2} \mu_j(\mathbf{d}|\triangle) M(\mathbf{u}|V \setminus \{\mathbf{t}_{i_j}\}) \tag{3.16}$$

where V is the knot set.

For a simplex spline surface, we have:

$$D_{\mathbf{d}} F(\mathbf{u}) = \sum_{I \in P_{\mathbf{u}}} \sum_{|\beta|=n} c_\beta^I D_{\mathbf{d}} N_\beta^I(\mathbf{u}) \tag{3.17}$$

where $P_{\mathbf{u}}$ denotes the triangles in the neighborlet of the triangle enclosing point $\mathbf{u}$ defined in Section 3.1, and

$$D_{\mathbf{d}} N_\beta^I(\mathbf{u}) = |det(\triangle_\beta^I)| D_{\mathbf{d}} M(\mathbf{u}|\ V) \tag{3.18}$$

Up to q^{th} derivatives ($1 \leq q \leq n-1$) for simplex splines are given by:

$$D_{\mathbf{d}_1\mathbf{d}_2\ldots\mathbf{d}_q}M(\mathbf{u}) = (n-q+1)\sum_{j=0}^{2}\mu_j(\mathbf{d}_1|\triangle)D_{\mathbf{d}_2\ldots\mathbf{d}_q}M(\mathbf{u}|V\setminus\{\mathbf{t}_{i_j}\}) \tag{3.19}$$

3.1.4 Integration of Bivariate Simplex Spline Functions

It is obvious that, at the base case, an integral of simplex spline basis function over a region A is:

$$\int_A M(\mathbf{u}|\mathbf{t}_{i,0},\mathbf{t}_{i,1},\mathbf{t}_{i,2})d\sigma = A\cap\triangle[\mathbf{t}_{i,0},\mathbf{t}_{i,1},\mathbf{t}_{i,2}] \tag{3.20}$$

A degree n simplex spline basis function is:

$$\int_A M(\mathbf{u}|\mathbf{t}_0,\ldots,\mathbf{t}_m)d\sigma = \sum_{j=0}^{2}\int_A \lambda_j(\mathbf{u}|\triangle_{\mathbf{t}_{i_0},\mathbf{t}_{i_1},\mathbf{t}_{i_2}})M(\mathbf{u}\mid\mathbf{t}_0,\ldots\hat{\mathbf{t}}_{i_j},\ldots\mathbf{t}_m)d\sigma \tag{3.21}$$

The integral of a quadratic basis function is the summation of the integral of nine bounded polynomials over region A. For example,

$$\int_A M_{200}(\mathbf{u})d\sigma = \int_A\sum_{i=0}^{2}\sum_{j=0}^{2}\lambda_i(\mathbf{u}|\triangle_{00})\lambda_j(\mathbf{u}|\triangle_{1i})\mathcal{X}_{\triangle_{ij}}\frac{1}{det(\triangle_{ij})}d\sigma \tag{3.22}$$

$$= \frac{1}{det(\triangle_{ij})}\sum_{i=0}^{2}\sum_{j=0}^{2}\int_{A\cap\triangle_{ij}}\lambda_i(\mathbf{u}|\triangle_{00})\lambda_j(\mathbf{u}|\triangle_{1i})d\sigma \tag{3.23}$$

where $\triangle_{00}=\triangle_{200}=(\mathbf{t}_{0,2},\mathbf{t}_{1,0},\mathbf{t}_{2,0})$, $\triangle_{1i}=\triangle_{200-\mathbf{e}^i}$, $\triangle_{ij}=\triangle_{200-\mathbf{e}^i-\mathbf{e}^j}$, and $\mathbf{e}^j=(\delta_j), j\in\{0,1,2\}$ or explicitly $\mathbf{e}^0=(100),\mathbf{e}^1=(010),\mathbf{e}^2=(001)$. Accordingly, integration of other quadratic basis functions can be calculated in a similar manner.

3.2 Trivariate Simplex Splines

A trivariate simplex spline is a natural extension of bivariate simplex spline. Trivariate simplex spline functions possess the same properties as bivariate simplex spline surfaces. Therefore, in this section, we focus on the explicit definition and efficient evaluation of trivariate simplex splines. The evaluation scheme developed here will be utilized in trivariate data modeling in the next chapter. The generalization to n-variate simplex splines of arbitrary degree can be found in [20].

3.2.1 Mathematical Background

Basic Definitions

Half-open 3-simplex is a natural analog of half-open 2-simplex in 3D space. The definition for half-open 3-simplex is similar to that for half-open 2-simplex. Here we only give a

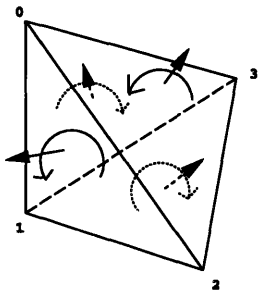


Figure 3-6: The direction "out" for a tetrahedron

intuitive description to define it.

Definition 3.6 *Given* $\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3 \in R^3$, *we denote by* $\triangle = [\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3]$, *the tetrahedron formed by these points. For a point* $\mathbf{u} \in \triangle$, *make a ray cast from* $\mathbf{u}$ *to the positive x-axis. If this ray hits more than one point on the 3-simplex, then* $\mathbf{u}$ *is in the* **half-open 3-simplex**, *denoted as* $[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3)$. *Otherwise,* $\mathbf{u}$ *is not in the half-open simplex.*

Definition 3.7 *Every polyhedron* P *(of* m *vertices) may be partitioned into 3-simplices such that the intersection of any two simplices in the* **triangulation** T *is either an empty set, a common vertex, a common edge, or a common face. The direction of* T *is defined as "in" or "out" if every tetrahedron* $[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3]$ *in* T *is defined as* pointing in or out. *Figure 3-6 shows an example of a tetrahedron pointing "out".*

Definition 3.8 *The barycentric coordinates* $(\lambda_0, \lambda_1, \lambda_2, \lambda_3)$ *of a point* $\mathbf{u}$ *with respect to a tetrahedron* $\triangle = (\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3), \mathbf{t}_i \in \mathbf{R}^3$, *are defined as:*

$$\lambda_0(\mathbf{u}|\triangle) = \frac{vol(\mathbf{u}, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3)}{vol(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3)} \quad \lambda_1(\mathbf{u}|\triangle) = \frac{vol(\mathbf{t}_0, \mathbf{u}, \mathbf{t}_2, \mathbf{t}_3)}{vol(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3)} \tag{3.24}$$

$$\lambda_2(\mathbf{u}|\triangle) = \frac{vol(\mathbf{t}_0, \mathbf{t}_1, \mathbf{u}, \mathbf{t}_3)}{vol(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3)} \quad \lambda_3(\mathbf{u}|\triangle) = \frac{vol(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{u})}{vol(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3)} \tag{3.25}$$

if points $\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3$ *are affinely independent.*

It can be proven that

$$\mathbf{u} = \lambda_0\mathbf{t}_0 + \lambda_1\mathbf{t}_1 + \lambda_2\mathbf{t}_2 + \lambda_3\mathbf{t}_3 = \sum_{i=0}^{3} \lambda_i\mathbf{t}_i \tag{3.26}$$

$$vol(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3) = \frac{1}{6} det \begin{pmatrix} 1 & 1 & 1 & 1 \\ \mathbf{t}_0 & \mathbf{t}_1 & \mathbf{t}_2 & \mathbf{t}_3 \end{pmatrix} \tag{3.27}$$

provided that the four point $\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3$ are not coplanar, i.e., $vol(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3) \neq 0$.

Note: Similar to bivariate simplex spline cases, the points inside a tetrahedron have positive barycentric coordinates.

Definition 3.9 *A* **trivariate simplex spline basis function** *is defined as*

$$M(\mathbf{u}|\mathbf{t}_0, ..., \mathbf{t}_m) = \sum_{j=0}^{3} \lambda_j(\mathbf{u}) M(\mathbf{u}|\mathbf{t}_0, ..., \hat{\mathbf{t}}_{i_j}, ..., \mathbf{t}_m) \tag{3.28}$$

$$M(\mathbf{u}|\mathbf{t}_{i,0}, \mathbf{t}_{i,1}, \mathbf{t}_{i,2}, \mathbf{t}_{i,3}) = \frac{\mathcal{X}_{[\mathbf{t}_{i,0}, \mathbf{t}_{i,1}, \mathbf{t}_{i,2}, \mathbf{t}_{i,3})}(\mathbf{u})}{\mid det(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3) \mid} \tag{3.29}$$

where

$$\mathcal{X}_{[\mathbf{t}_{i,0}, \mathbf{t}_{i,1}, \mathbf{t}_{i,2}, \mathbf{t}_{i,3})}(\mathbf{u}) = \begin{cases} 1 & \text{if } \mathbf{u} \in [\mathbf{t}_{i,0}, \mathbf{t}_{i,1}, \mathbf{t}_{i,2}, \mathbf{t}_{i,3}) \\ 0 & \text{otherwise} \end{cases} \tag{3.30}$$

Note: $\hat{\mathbf{t}}_{i_j}$ *means* $\hat{\mathbf{t}}_{i_j}$ *is removed from the knot set.*

$\triangle_{\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2}, \mathbf{t}_{i_3}} = \{\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2}, \mathbf{t}_{i_3}\} \subseteq \{\mathbf{t}_0, ..., \mathbf{t}_m\}$, *called* **reference tetrahedron**, *is any subset of affinely independent four points and* $\lambda_j(\mathbf{u})$ *are the barycentric coordinates of* $\mathbf{u}$ *with respect to* $\mathbf{t}_{i_j}$.

Equation 3.28 is independent of the specific choice of $\triangle_{\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2}, \mathbf{t}_{i_3}}$ [20].

Definition 3.10 *Let* $T = \{\triangle(I) = [\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2}, \mathbf{t}_{i_3}] | I = (i_0, i_1, i_2, i_3) \in I \subseteq Z_+^4\}$ *be an arbitrary triangulation of the parameter space* $\mathbf{R}^3$. *Assign a sequence of knots* $\mathbf{t}_{i,0}, ..., \mathbf{t}_{i,n}$ *to every vertex* $\mathbf{t}_i$ *in this triangulation. So* $\mathbf{t}_{i,0} = \mathbf{t}_i$ *and any four knots form a proper tetrahedron. Then consider the sets:*

$$V_\beta^I = \underbrace{\{\mathbf{t}_{i_0,0}, ..., \mathbf{t}_{i_0,\beta_0}, ..., \mathbf{t}_{i_3,0}, ..., \mathbf{t}_{i_3,\beta_3}\}}_{n+4} \tag{3.31}$$

where $\beta = (\beta_0, \beta_1, \beta_2, \beta_3)$, $|\beta| = \beta_0 + \beta_1 + \beta_2 + \beta_3 = n$ *then an arbitrary* **trivariate simplex spline hyper-surface** *of degree* n *over a given triangulation* T *is:*

$$F(\mathbf{u}) = \sum_{I \in T} \sum_{|\beta|=n} N_\beta^I(\mathbf{u}) c_\beta^I, \tag{3.32}$$

where $c_\beta^I \in \mathbf{R}^4$ *are the control points for the shape of the surface and*

$$N_\beta^I(\mathbf{u}) = |det(\mathbf{t}_{i_0,\beta_0}, \mathbf{t}_{i_1,\beta_1}, \mathbf{t}_{i_2,\beta_2}, \mathbf{t}_{i_3,\beta_3})| \cdot M_\beta^I(\mathbf{u}) \tag{3.33}$$

is called normalized trivariate simplex splines basis function.

If c_β^I are scalar numbers, Equation 3.32 generates a **trivariate simplex spline function**. This function can be used in aplications, e.g., 3D data modeling.

Similar to the 2D counterpart, in ensuring region $\Omega_3^I \neq \phi$, we can have:

$$\sum_{I \in P_{\mathbf{u}}} \sum_{|\beta|=n} N_\beta^I(\mathbf{u}) = 1. \tag{3.34}$$

where $P_{\mathbf{u}}$ denotes the tetrahedron neighborlet of $\triangle$ which contains $\mathbf{u}$.

Thus, we have

$$F(\mathbf{u}) = \sum_{I \in P_{\mathbf{u}}} \sum_{|\beta|=n} N_\beta^I(\mathbf{u}) c_\beta^I. \tag{3.35}$$

This equation provides a linear system with a banded matrix when we apply simplex spline functions in approximating a set of data points in Chapter 4.

3.2.2 Evaluation of Trivariate Simplex Spline Functions

For the same reason stated in Section 3.1.2, we emphasize the efficiency of evaluation of trivariate simplex spline functions. Evaluation of a point of a trivariate simplex spline function over one tetrahedron includes (1) determining if $\mathbf{u} \in \triangle_{\mathbf{t}_{i_0},\mathbf{t}_{i_1},\mathbf{t}_{i_2},\mathbf{t}_{i_3}}$; (2) computing barycentric coordinates; (3) multiplication of barycentric coordinates and the lower degree basis functions.

Analogously we consider the following four computational steps:

1. Determining the tetrahedron enclosing point $\mathbf{u}$.
2. Determining if a point is in a half-open 3-simplex.
3. Reducing the number of intermediate terms of lower degree basis functions.
4. Avoiding unnecessary barycentric coordinate computations.

Determining which domain tetrahedron encloses u

The tetrahedron in which the point $\mathbf{u} = (x, y, z)$ is enclosed must be found. We use here *bucket sorting* discussed in Section 3.1.2, see also Appendix B.

Determining if u is in a half-open 3-simplex

For the same reason as stated for bivariate simplex spline basis functions, we need to determine efficiently if a point $\mathbf{u} \in \mathbf{R}^3$ is in a half-open 3-simplex with numerical stability. Obviously, we do not want to use the barycentric coordinates of $\mathbf{u}$ because (1) it is inappropriate for boundary points, and (2) it needs to compute determinants of five matrices of order four. We do not want to use ray casting directly on the four faces of a 3-simplex either because that will check the ray intersecting four faces. However, by using the algorithm from 2D cases, we can simplify this situation based on the simple fact that, projecting $\mathbf{u}$ and a simplex onto y-z plane, we obtain $\mathbf{u}'$ and 4 triangles out of 4 vertices: $\mathbf{v}_0', \mathbf{v}_1', \mathbf{v}_2', \mathbf{v}_3'$. On the y-z plane, there are two possible cases.

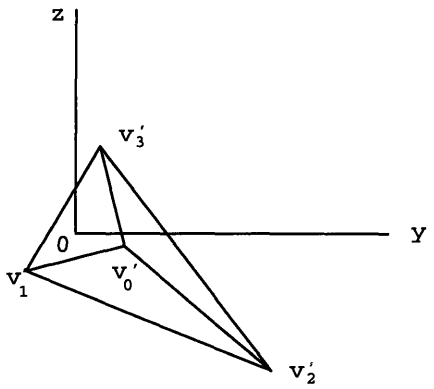


Figure 3-7: Project the 3-simplex onto y-z plane, we obtain four planar points: $\mathbf{v}_0', \mathbf{v}_1', \mathbf{v}_2', \mathbf{v}_3'$ and $\mathbf{u}$ =origin. Then (1) check which half-open triangles contain $\mathbf{u}$; (2) check the corresponding triangle faces to see if they intersect the positive x-axis; if yes increase N_c, the number of axis-face intersection by 1; (3) Finally, if N_c is odd, return TRUE; otherwise return FALSE.

- $\mathbf{u}'$ is not in the convex hull formed by the projected triangles. This indicates $\mathbf{u}$ is not in the 3-simplex.
- $\mathbf{u}'$ is in the convex hull. Then $\mathbf{u}$ is in at most two half-open 2-simplex. This indicates that we can exclude two 2-simplices by using the 2D process IN_HO_TRIANGLE. Thus we need to perform only two ray-face checks. If the result is that the ray intersects only one face, $\mathbf{u}$ is in the half-open 3-simplex, otherwise it is not. (Note: This procedure is extensible to a general half-open s-simplex ($s > 3$).)

The following pseudocode will take in an array of four space points v[0], v[1], v[2], v[3] and a space point $\mathbf{u}$ and determine if $\mathbf{u} \in [\mathbf{v}[0], \mathbf{v}[1], \mathbf{v}[2], \mathbf{v}[3])$, i.e., $\mathbf{u} \in [\mathbf{v})$, where $[\mathbf{v})$ denotes the half-open tetrahedron formed by the vertices v[0], v[1], v[2], v[3]. This routine first translates the origin to $\mathbf{u}$ and projects the vertices onto y-z plane. In the projected triangles, it uses IN_HO_TRIANGLE to determine half-open triangles containing $\mathbf{u}$; if none contains $\mathbf{u}$, then $\mathbf{u}$ is not in the half-open 3-simplex; otherwise, check which projected half-open triangles contain $\mathbf{u}$, and denote the corresponding triangle faces as $F_i, i \in \{0, 1\}$. Afterwards, check how many F_i intersect the positive x-axis, if the number is odd, then $\mathbf{u}$ is in $[\mathbf{v})$. See Figure 3-7.

```
IN_HO_TETRAHEDRON (v, u)
1   for i ← 0 to 4
2       v[i] ← v[i] - u;
3   N_c ← 0;
4   project u , v[i] to y-z plane obtaining u', v'[i], i = 0, ..., n.
5   for i ← 0 to 4
6       j ← (i+1) mod 4;
```

Number of items needed for one basis function : M(u)			
	base cases	lower degree basis functions	bar. coords.
Quadratic	4^2	$4^1 + 4^2$	$4^1 + 4^2$
Degree n	4^n	$\sum_{i=1}^{n} 4^i$	$\sum_{i=1}^{n} 4^i$
Number of items needed for one point over one tetrahedron: F(u)			
	base cases	lower degree basis functions	bar. coords.
Quadratic	10×4^2	$10 \times (4^1 + 4^2)$	$10 \times (4^1 + 4^2)$
Degree n	$m_c 4^n$	$m_c \sum_{i=1}^{n} 4^i$	$m_c \sum_{i=1}^{n} 4^i$

Table 3.2: The number of items needed for evaluation of a trivariate simplex spline basis function and a point on a trivariate simplex spline function, where $m_c = \frac{(n+1)(n+2)(n+3)}{6}$.

```
7       k ← (i+2) mod 4;
8       n ← IN_HO_TRIANGLE(v', u)
9       if(n > 0)
10          f0 ← the function value of the face at (0,0)
11          f1 ← the function value of the face at (1,0)
12          if(f0 • f1 < 0)
13              Nc ← Nc + 1;
14          else if(|f0| > |f1|)
15              Nc ← Nc + 1;
16  return Nc mod 2;
```

Evaluation of Simplex Spline Functions

Simplex spline basis functions are evaluated by a recursive formula, see Equation 3.28. A degree n trivariate simplex spline function needs $m_c = \frac{(n+1)(n+2)(n+3)}{6}$ basis functions of the same degree; and each of them needs $\sum_{i=1}^{n} 4^i$ lower degree basis functions. For example, straightforward computation for a point on a quadratic trivariate simplex spline function over one tetrahedron needs 10 basis functions of degree, i.e., $M_\beta, (|\beta| = 2)$. If we assume that the knots associated with a domain tetrahedron are in general position, then we can evaluate the $M_\beta, |\beta| = 2$ with less operations by a method analogous to the one in Section 3.1.2.

For example, evaluating a quadratic simplex spline basis function $M_{\beta_0\beta_1\beta_2\beta_3}$ needs only 20 lower degree basis functions, i.e., 4 $M_\beta, (|\beta| = 1)$ and 16 $M_\beta, (|\beta| = 0)$. Figure 3-8 shows that, by fixing the reference tetrahedra, $M_{\beta_0\beta_1\beta_2\beta_3}$ can be obtained from fourteen $M_\beta, |\beta| = \{0, 1\}$, where β_i means taht, in the i^{th} knot-cloud, $\beta_i + 1$ knots will be involved in computng $M_{\beta_0\beta_1\beta_2\beta_3}$. These knots are $\mathbf{t}_{i,0}, ..., \mathbf{t}_{i,\beta_i}$. Applying this strategy on all ten basis functions needed for evaluating a point on a quadratic simplex spline function over a tetrahedron, about 70% of lower degree M_β can be saved.

The following rules are followed in choosing a reference tetrahedron and computing the

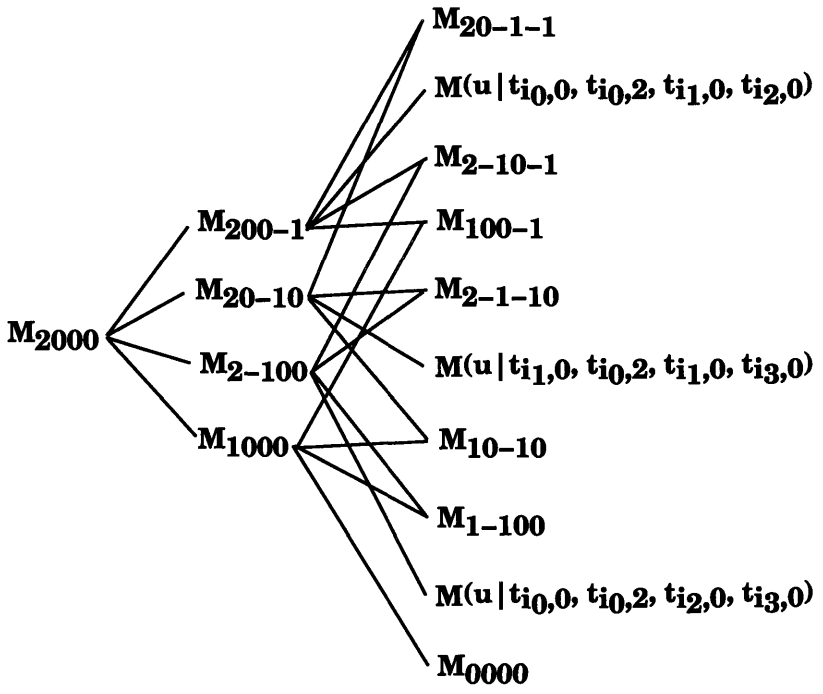


Figure 3-8: Efficiently evaluating a quadratic trivariate simplex spline basis function.

basis function:

1. A reference tetrahedron $\triangle$ for basis function evaluation is formed by four knots, each of the knots is from a distinct knot cloud if the knot cloud participates in the computation, i.e., $\beta_i \geq 0$.
2. When choosing a knot from a knot cloud, choose the one with largest index if possible.
3. If a knot cloud does not contribute any knots ($\beta_i = -1$), choose knots from other knot clouds, keeping the above rules in mind.

For example, when calculating M_{2000}, choose $\triangle = (t_{0,2}, t_{1,0}, t_{2,0}, t_{3,0})$ as the reference tetrahedron, thus we can have lower degree basis function as $M_{2000-\mathbf{e}^i}, i = 0, ..., 3$, or explicitly, $M_{1000}, M_{2-100}, M_{20-10}, M_{200-1}$, here "-1" indicating the corresponding knot cloud does not participate in further computing and $\mathbf{e}^j = (\delta_j), j \in \{0, 1, 2, 3\}$.

Avoiding barycentric coordinate computations

Similar to bivariate cases, if the basis function is zero, then it is not necessary to calculate the corresponding barycentric coordinate λ_i.

3.2.3 Differentiation of Trivariate Simplex Spline Functions

A directional derivative along a given direction $\mathbf{d} = (d_x, d_y, d_z)$ for a parameter $\mathbf{u} \in \mathbf{R}^3$ can be computed for a trivariate simplex spline in the following form:

$$\mathbf{d} = \sum_{j=0}^{3} \mu_j(\mathbf{d}|\triangle)\mathbf{t}_j; \quad \sum_{j=0}^{3} \mu_j(\mathbf{d}|\triangle) = 0. \tag{3.36}$$

Similar to bivariate simplex splines, the first order directional derivatives for a degree n simplex spline basis function can then be computed as [68, 20, 95]:

$$D_{\mathbf{d}}M(\mathbf{u}|V) = n\sum_{j=0}^{3} \mu_j(\mathbf{d}|\triangle)M(\mathbf{u}|V \setminus \{\mathbf{t}_{i_j}\}) \tag{3.37}$$

For a trivariate simplex spline function, we have:

$$D_{\mathbf{d}}F(\mathbf{u}) = \sum_{I \in T} \sum_{|\beta|=n} c_\beta^I D_{\mathbf{d}} N_\beta^I(\mathbf{u}) \tag{3.38}$$

$$D_{\mathbf{d}}N_\beta^I(\mathbf{u}) = |det(\triangle_\beta^I)| D_{\mathbf{d}}M(\mathbf{u}|V) \tag{3.39}$$

Up to q^{th} derivatives ($1 \leq q \leq n-1$) for simplex splines are given by:

$$D_{\mathbf{d}_1\mathbf{d}_2\ldots\mathbf{d}_q}M(\mathbf{u}) = (n-q+1)\sum_{j=0}^{3}\mu_j(\mathbf{d}_1|\triangle)D_{\mathbf{d}_2\ldots\mathbf{d}_q}M(\mathbf{u}|V\setminus\{\mathbf{t}_{i_j}\}) \tag{3.40}$$

3.2.4 Integration of Trivariate Simplex Spline Functions

It is obvious that, as in the base case, integral of simplex spline basis function over the spatial region V is:

$$\int_V M(\mathbf{u}|\mathbf{t}_0,\mathbf{t}_1,\mathbf{t}_2,\mathbf{t}_3)d\sigma = V\cap vol[\mathbf{t}_0,\mathbf{t}_1,\mathbf{t}_2,\mathbf{t}_3] \tag{3.41}$$

where $d\sigma = dxdydz$.

The integration of a degree n simplex spline basis function is:

$$\int_V M(\mathbf{u}|\mathbf{t}_0,\ldots,\mathbf{t}_m) = \sum_{j=0}^{3}\int_V \lambda_j(\mathbf{u}|\triangle_{\mathbf{t}_{i_0},\mathbf{t}_{i_1},\mathbf{t}_{i_2},\mathbf{t}_{i_3}})M(\mathbf{u}|\mathbf{t}_0,\ldots\hat{\mathbf{t}}_{i_j},\ldots\mathbf{t}_m)d\sigma \tag{3.42}$$

For example, a quadratic trivariate basis function is the summation of the integral of sisteen bounded quadratic polynomials over the spatial region V.

$$\int_V M_\beta d\sigma = \int_V \sum_{i=0}^{3}\sum_{j=0}^{3}\lambda_i(\mathbf{u}|\triangle_{00})\lambda_j(\mathbf{u}|\triangle_{1i})\mathcal{X}_{\triangle_{ij}}d\sigma = \sum_{i=0}^{3}\sum_{j=0}^{3}\int_{V\cap\triangle_{ij}}\lambda_i(\mathbf{u}|\triangle_{00})\lambda_j(\mathbf{u}|\triangle_{1i})d\sigma \tag{3.43}$$

where $\triangle_{00} = \triangle_\beta$, $\triangle_{1i} = \triangle_{\beta-\mathbf{e}^i}$, $\triangle_{ij} = \triangle_{\beta-\mathbf{e}^i-\mathbf{e}^j}$, and $\mathbf{e}^0 = (1000), \mathbf{e}^1 = (0100), \mathbf{e}^2 = (0010), \mathbf{e}^3 = (0001)$.

3.3 Properties of Simplex Splines

Simplex splines have the following properties:

1. **Modeling capability** in modeling irregular domains.
2. **Piecewise polynomials** of degree n. Therefore a simplex spline function can be subdivided into Bézier expressions although the resulting subdivided subdomains are numerous and multisided, see [85, 20] for details.
3. **Convex hull property**: A simplex spline surface lies in the convex hull of its control points. This is also a local property, i.e., a part of the surface lies in the convex hull of its local control points.
4. **Locality**: Movement of a control point influences the region of the (hyper)surface within the domain simplex's neighborlet and the simplices adjacent to the neighborlet.

5. **Continuity**: A degree n simplex spline function is C^{n-1} continuous everywhere if knots are in general position in every domain simplex.

6. **Affine invariance**: The relationship between the control points and the simplex spline (hyper)surface is invariant under affine coordinate transformations.

These properties can be utilized to model physical data over irregular domains with low degree simplex spline surfaces, and automatic continnuity and optimal smoothness can be achieved.

3.4 Discussion

Simplex spline, B-spline and composite triangular Bézier patches

Table 3.3 shows a comparison among simplex splines, tensor product B-splines and triangular Bézier patches. It is shown in the table that simplex splines can achieve optimal smoothness (C^{n-1}) automatically and model surfaces in irregular domains.

	Simplex spline	Tensor product B-spline	Triangular Bézier
Degree	Low (n)	High (n×n)	Low (n)
Domain	Irregular	Regular only	Irregular
Continuity	High order(C^{n-1})	High order(C^{n-1})	Low order($C^{\lfloor \frac{n-1}{4} \rfloor}$)[38]
Smoothness	Automatic	Automatic	Complex

Table 3.3: Comparison of simplex spline, B-spline, and composite Bézier triangles.

Evaluation complexity of simplex spline, and composite triangular Bézier patches

Quadratic simplex spline surfaces can achieve up to C^1 continuity contrary to the situation of composite triangular Bézier patches which need quintic surface patches to achieve the same degree of continuity. We compare the number of multiplications/divisions they need in the evaluation of a point on the corresponding surfaces. Table 3.4 shows a comparison between evaluation of a point on a quadratic simplex spline surface and a composite quintic triangular Bézier patch which are needed in modeling a C^1 surface. We show the numbers of multiplications/divisions under straightforward basis function computation for both surfaces and using the De Casteljau algorithm for the triangular Bézier patch. Notice that the evaluation using De Casteljau algorithm saves computational time dramatically. Unfortunately, for simplex spline surfaces, a De Casteljau-type algorithm is not yet available. However, the same magnitude of time-saving can be expected if such an algorithm could be developed.

	Straightforward	De Casteljau
Simplex splines	$444m_s$	-
Bézier patches	7631	371

Table 3.4: Complexity of evaluation of a quadraric simplex spline surface and a quintic Bézier triangular patch by straightforward basis function computation and De Casteljau algorithm, respectively. Here m_s is the number of triangles in the triangle neighborlet.

For the evaluation of a point on a quadratic simplex spline surface, from Definition 3.10, we notice that 6 normalized basis functions $N_\beta(\mathbf{u}) = det(\mathbf{t}_{i_0,\beta_0}, \mathbf{t}_{i_1,\beta_1}, \mathbf{t}_{i_2,\beta_2})\ M_\beta(\mathbf{u})$, where $|\beta| = 2$ are needed.

1. Each $M_\beta, (|\beta| = 2)$ needs 12 barycentric coordinates; every 3 barycentric coordinates involve 6 multiplications and 2 divisions, i.e., $\frac{12}{3}\times(6+2)=32$ multiplications/divisions.

2. In addition, each barycentric coordinate will multiply a lower degree basis function once, i.e., 12 multiplications.

3. At the base case in Equation 3.6, if we neglect the computational time for the characteristic function $\mathcal{X}(\mathbf{u})$, we need to calculate a determinant *det* of order 2, which needs 2 multiplications, and the value of $\frac{\mathcal{X}(\mathbf{u})}{|\bullet|}$, where "$|\bullet|$" means $|det|$, which needs 1 division and since there are 9 base cases for an $M_\beta(\mathbf{u})$ where $|\beta| = 2$, the total number of multiplications involved in the base cases is $(2+1)\times9=27$.

4. After $M_\beta(\mathbf{u})$, where $|\beta| = 2$ is computed, $a = det(\mathbf{t}_{i_0,\beta_0}, \mathbf{t}_{i_1,\beta_1}, \mathbf{t}_{i_2,\beta_2})$ needs 2 multiplications and $a \times M_\beta(\mathbf{u})$ should also be counted as 1 multiplication. Therefore, to compute a $N_\beta(\mathbf{u})$, where $|\beta| = 2$, we need $32 + 12 + 27 + 2 + 1 = 74$ multiplications/divisions.

5. Finally, the linear combination of 6 normalized basis functions and the corresponding control coefficients takes 6 multiplications.

Therefore the overall multiplication and divisions needed is: $74\times6=444$ for each involved triangle in the triangle neighborlet. Thus, at worst case, totally $444m_s$ multiplications/divisions are needed in computing a point on a quadratic bivariate simplex spline surface, where m_s is the number of triangles in the triangle neighborlet.

By using the De Casteljau algorithm, the computation of a point on a Bézier triangle needs 314 multiplications. Equation 3.44 shows the De Casteljau algorithm for the evaluation of a point with barycentric coordinates $\mathbf{u} = (u, v, w)$ on a degree n Bézier triangle [39]:

$$\mathbf{b}_\mathbf{i}^r(\mathbf{u}) = u\mathbf{b}_{\mathbf{i+e1}}^{r-1}(\mathbf{u}) + v\mathbf{b}_{\mathbf{i+e2}}^{r-1}(\mathbf{u}) + w\mathbf{b}_{\mathbf{i+e3}}^{r-1}(\mathbf{u}), \tag{3.44}$$

where $r = 1, ..., n$ and $|\mathbf{i}| = n - r$, $\mathbf{b_i} \in \mathbf{R}$ are control coefficients, and $\mathbf{e1} = (100)$, $\mathbf{e2} = (010)$, and $\mathbf{e3} = (001)$. To calculate the $\mathbf{b}(\mathbf{u}) = \mathbf{b}_i^5(\mathbf{u})$, $\sum_{i=1}^{5} 3^i = 363$ multiplications

are needed. In addition, 6 multiplications and 2 divisions are needed to determine the 3 barycentric coordinates of a point in the parametric space. Thus, we need 363+8 = 371 multiplications/divisions in the evaluation.

By straightforward basis function computation, the complexity of evaluation is shown below, see Equation 3.45.

$$\mathbf{b}^n(\mathbf{u}) = \sum_{|\mathbf{i}|=n-r} \mathbf{b}_{\mathbf{i}}^r(\mathbf{u}) B_{\mathbf{i}}^{n-r}(\mathbf{u}) \tag{3.45}$$

where the Bernstain polynomials $B_{\mathbf{i}}^n(\mathbf{u})$ are recursively defined by

$$B_{\mathbf{i}}^r(\mathbf{u}) = uB_{\mathbf{i}-\mathbf{e1}}^{r-1}(\mathbf{u}) + vB_{\mathbf{i}-\mathbf{e2}}^{r-1}(\mathbf{u}) + wB_{\mathbf{i}-\mathbf{e3}}^{r-1}(\mathbf{u}); \tag{3.46}$$

Analog to the complexity analysis for the computation of a $B_{\mathbf{i}}^5(\mathbf{u})$ using Equation 3.46 recursively, we need 363 multiplications. Since a quintic Bézier triangle has 21 control coefficients, totally, for the straightforward computation together with the computation of barycentric coordinates, we need $21 \times 363 + 8 = 7631$ multiplications/divisions.

The use of Clough-Toucher method instead of standard Bézier triangles can reduce the complexity in evaluation but increase the memory storage dramatically.

Issues on simplex spline functions

There are some issues that need to be explored extensively in the theoretical development of simplex splines.

- *Triangulation*: In scattered data fitting, a simplex spline needs certain number of data points in every domain simplex. Although such a requirement can be met by a data segmentation scheme, which is good for approximation, it is not ideal for interpolation. See Traas [95].
- *Knot configuration*: How to generate optimal knot configurations has not been thoroughly studied. See Traas [95].
- *De Casteljau-like Algorithm*: The development of a De Casteljau-like algorithm is highly desired to enhance the efficiency of the evaluation of simplex spline functions.

3.5 Examples of Bivariate/Trivariate Simplex Spline Functions

Our example shows a quadratic simplex spline surface defined over a 2D domain shown in Figure 3-9. At each vertex, we associate two knots. The domain triangulation and the knots are stored in a topological modeling structure introduced in Section 2.4. Control points can be accessed by corresponding triangle supports. Figure 3-10 shows the rendered surface in wireframe.

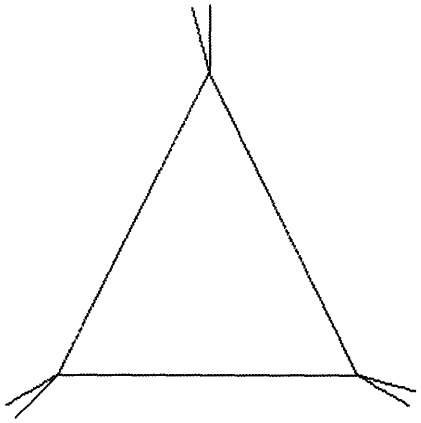


Figure 3-9: The domain of a simplex spline surface.

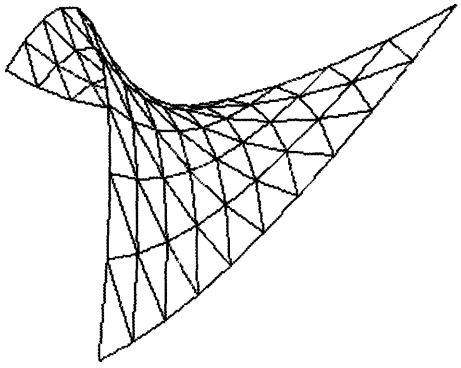


Figure 3-10: A quadratic simplex spline surface defined in domain shown in Figure 3-9.

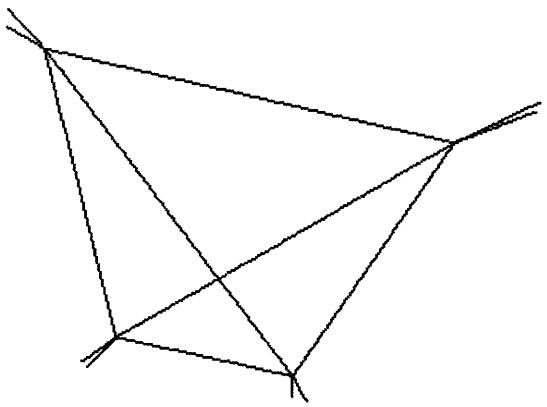


Figure 3-11: The domain tetrahedron and the knots of a trivariate simplex spline.

Our second example is a quadratic simplex spline hypersurface over a 3D tetrahedron. Similar to 2D cases, we associate two knots with each vertex. Figure 3-11 shows the domain tetrahedron and the knots; Figure 3-12 shows a isovalued surface of this hypersurface when the function value is 0.5 by using the visualization technique called *marching simplices*, which will be described in Chapter 6.

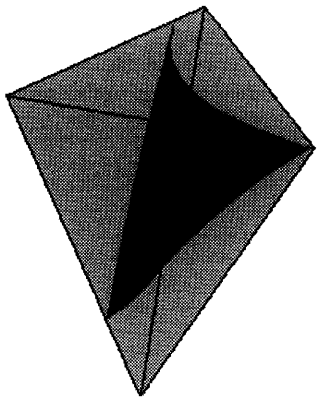


Figure 3-12: Isosurface of a trivariate simplex spline function in a tetrahedron.

Chapter 4

Modeling Physical Data in Irregular Domains

4.1 Introduction

Data modeling is the key in the construction of scientific databases. In this chapter, 2D (3D) data refers to the data points defined in a two (three)-dimensional Euclidean space and associated with scalar function values. Each function value has a certain physical meaning, e.g., terrain elevation, oil concentration, temperature, which can be described by continuous functions. The goal of data modeling and representation is to find a bivariate/trivariate continuous function defined in the same domain which can describe the behavior of the data in an accurate manner. This process is also called *(hyper) surface reconstruction.* The purpose of 2D (3D) data modeling is to interrogate and visualize the scientific data effectively to provide information on the data set. In this thesis, scattered data modeling involves approximating or fitting the data with low degree nonlinear piecewise polynomial functions.

In recent years, multivariate simplex splines have received much attention because of their attractive properties such as capability in modeling surfaces in irregular domains and achieving optimal smoothness. These properties are appropriate for fitting scattered 2D (3D) data.

However, applications of multivariate simplex splines in [85, 20, 42] are very rare, especially for the general simplex spline form, which is very likely due to their recent introduction, the difficulties arising in knot distribution derivation, and programming complexity. As demonstrated in Chapter 3, their use requires mathematical sophistication, and good software programming skills.

In this chapter, bivariate (trivariate) simplex spline functions of arbitrary degree are employed in modeling physical data in irregular domains. The modeling process includes knot generation, least-squares approximation, error estimation and domain subdivision. Figure 4-1 shows the flow chart of these operations. The resulting bivariate and trivariate functions can achieve optimal and automatic smoothness (if knots in one support are in

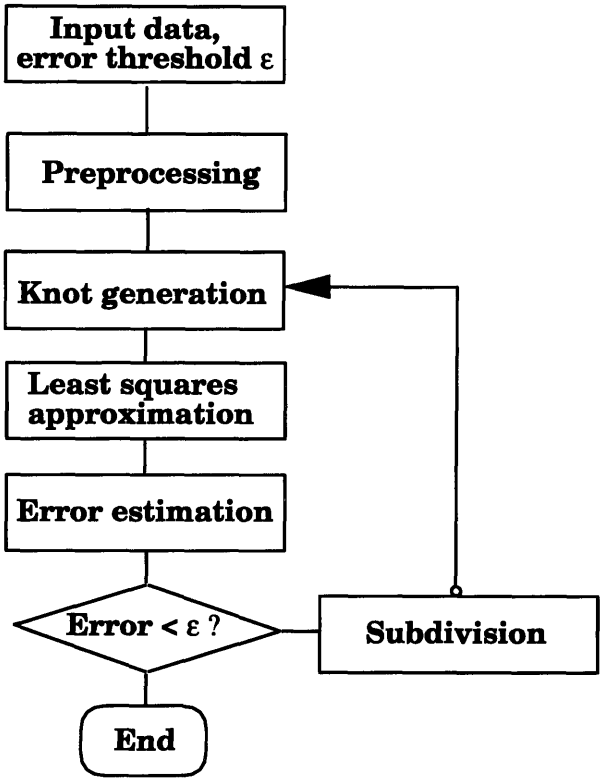


Figure 4-1: Flow chart for data modeling.

general position) to reflect the continuity of the physical phenomena encapsulated in the data.

4.2 2D Data Modeling

We concentrate on 2D data modeling in this section. Our objective is to create a bivariate simplex spline function $F : \mathbf{R}^2 \rightarrow \mathbf{R}^1$. The tasks of 2D data modeling include, apart from preprocessing (i.e., domain establishment and data segmentation discussed in Chapter 2),

1. knot generation;
2. least squares approximation;
3. error estimation;
4. subdivision.

These four processes can be automated in order to reduce the dependence on a sophisticated user of the modeling system.

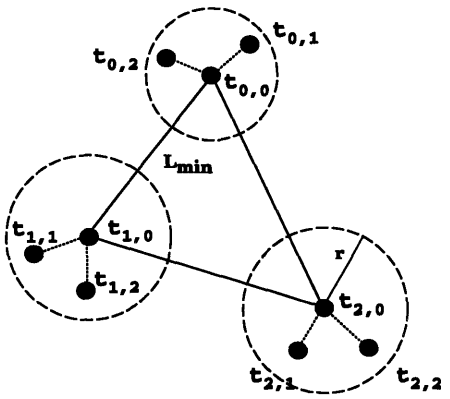


Figure 4-2: Knots associated with a vertex are in a circle centered at the vertex with radius $r \leq \frac{L_{min}}{2}$.

4.2.1 Knot Generation

Knots associated with each vertex in domain triangulation have much flexibility (freedom). Knot configurations associated with domain triangles play an important role in surface generation. For example, in a domain triangle, if there are q (> 2) knots collinear, the degree of continuity of the simplex spline surface will decrease by $(q-2)$ [85, 43, 20]. Knot configuration also influences the second order derivatives [95] of the surfaces [85, 43, 20, 95]. For physical data modeling, the knots have several constraints.

1. The distance between the knots $\mathbf{t}_{i,1}, ..., \mathbf{t}_{i,n}$ and the vertex $\mathbf{t}_{i,0}$, where n is the degree of the surface, needs to be considered. It is desirable to generate knots close to the vertex $\mathbf{t}_{i,0}$. A criterion to be used here is that the knots should be placed within a circle centered at the vertex with radius less than $\frac{1}{2}L_{min}$, where L_{min} is the length of the shortest edge of *all* the edges incident to the vertex $\mathbf{t}_{i,0}$ [95, 43]. (See Figure 4-2.)

2. To achieve optimal smoothness C^{n-1}, it is desirable that any three knots associated with a domain triangle are non-collinear. If we call a straight line between two knots a *knot line* or *forbidden line*, then we generate a new knot avoiding any existing knot lines (forbidden lines); in turn, the new knot with all the other knots form a new set of knot lines to serve as additional constraints for future knot generation. For example, Figure 4-3 shows knot generation for a quadratic simplex spline domain triangle. The vertices are knots and each vertex will be associated with two knots. Before we place any new knots, there are 3 knot lines (i.e., the three edges of the triangle). When creating a knot $\mathbf{t}_{0,1}$ for vertex $\mathbf{t}_0 (= \mathbf{t}_{0,0})$, we obtain three additional knot lines: $\overline{\mathbf{t}_{0,1}\mathbf{t}_0}$, $\overline{\mathbf{t}_{0,1}\mathbf{t}_1}$, and $\mathbf{t}_{0,1}\mathbf{t}_2$. The three knot lines, together with the initial three knot lines form constraints for the next knot generation. (See Figure 4-3.)

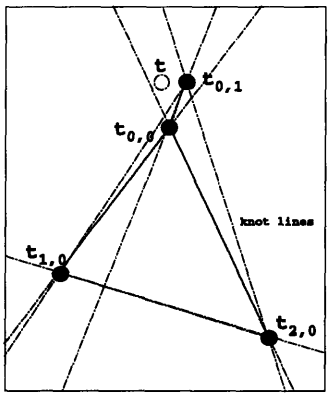


Figure 4-3: A new set of constraints is formed after introducing new knots in one domain triangle. Besides the constraints from $\overline{\mathbf{t}_{0,0}\mathbf{t}_{1,0}}$, $\overline{\mathbf{t}_{1,0}\mathbf{t}_{2,0}}$, $\overline{\mathbf{t}_{2,0}\mathbf{t}_{0,0}}$, new constraints are: $\overline{\mathbf{t}_{0,1}\mathbf{t}_{0,0}}$,$\overline{\mathbf{t}_{0,1}\mathbf{t}_{1,0}}$, $\overline{\mathbf{t}_{0,1}\mathbf{t}_{2,0}}$

3. Since three nearly collinear knots in one triangle make the jump in second derivatives of the surface across the knot lines of these knots very high [95], keeping the distance to the forbidden lines bounded from below assures that the second derivatives of the spline basis remain bounded. This fact is expressed by Equation 4.1, which describes the jump across the edge defined by the knots x^i, y^i and x^j, y^j [53].

$$D^\beta M|_1 - D^\beta M|_2 = 24\frac{(y^i - y^j)^{\beta_1}(x^i - y^j)^{\beta_2}}{\prod_{k=0,k\neq i,j}^{4} det(x^i, x^j, x^k)} \tag{4.1}$$

where $i,j \in (0,1,2,3,4)$, $\beta_1,\beta_2 \in (0,1,2), \beta_1+\beta_2 = 2, D^\beta = \frac{\partial^{|\beta|}}{\partial x^{\beta_1}\partial x^{\beta_2}}$. In case of collinearity, at least one of the determinants in the denominator of Equation 4.1 becomes zero. Therefore we generate a knot such that the absolute value of Equation 4.1 is minimized (at least approximately) over the edges in the domain triangles in which the knot is placed. The approach taken here is to avoid nearly three collinear knots in knot generation. Thus, it is desirable to generate a knot on the bisector of the two knot lines.

4. Knots associated with boundary vertices should be placed in the region K defined by the following criteria.

   (1) If the domain of definition is D, then the region for knots is outside of D, i.e., $K \cap D = \phi$.

   (2) For points $\mathbf{t} \in K \subset \mathbf{R}^2$, if we draw a line linking $\mathbf{t}$ and $\mathbf{t}_i$ and extend the line to obtain a symmetric point $\mathbf{t}'$ of $\mathbf{t}$ with respect to $\mathbf{t}_i$, $\mathbf{t}'$ should be in one of the boundary triangles incident to $\mathbf{t}_i$, i.e., $\mathbf{t}' \in \triangle_i \subset D$. See vertices $\mathbf{t}_i$ in Figure 4-4. The shaded region is the region for knots associated with vertex $\mathbf{t}_i$. In this way, the knot lines containing boundary knots are out of the domain D, which ensures the region

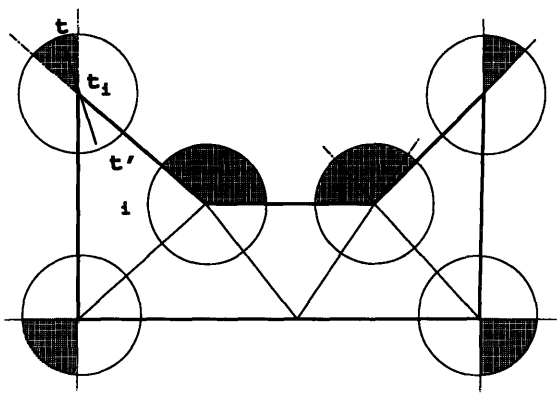


Figure 4-4: Boundary knot generation: knots associated with a boundary vertex should be in the shaded region.

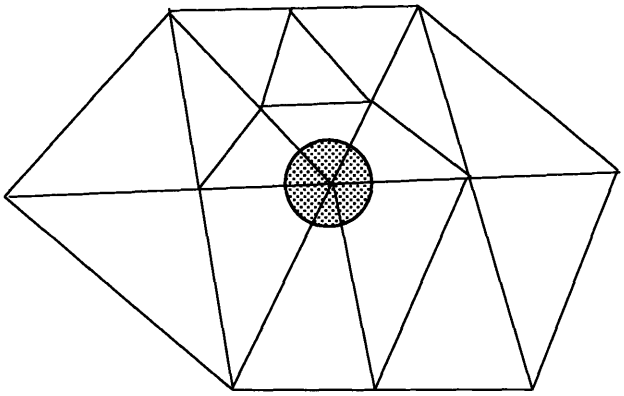


Figure 4-5: Inner knot generation: knots associated with an inner vertex can be anywhere in a circle centered at the vertex with a bounded radius, except on forbidden lines.

Ω defined by Equation 3.8 in the boundary triangle to be maximum. These criteria guarantee that the surface along the boundary of the domain is created properly[1].

5. Knots associated with interior vertices can be placed anywhere in the knot circles (except on the forbidden lines). (See Figure 4-5.)

Note:

1. Additional constraints with respect to knot generation may be imposed in case a designer wishes to use control points for manipulating surfaces.

[1]Otherwise, at some points in the domain near the boundary, the summation of the normalized basis functions may not be unity.

2. A further constraint with respect to knot generation may arise because of considerations of minimizing (on the average) the number of domain triangles which are incident to a given vertex. This is of importance for efficiency reasons, when computing values of linear combinations of simplex splines, see [95] for further details.

Based on the above constraints, we develop an automatic knot generator for bivariate simplex spline surfaces of arbitrary degree n. In the knot generator, T is the domain triangulation, ξ is a predefined positive real and $0 < \xi < 0.5$.

1. Input T, n and ξ.
2. For each vertex $\mathbf{v}_i$ of T, compute L_{min}, the shortest length of edges incident to the vertex.
3. Let $r = \xi L_{min}$.
4. If the vertex $\mathbf{v}_i$ is a boundary vertex, proceed; otherwise, go to Step 10.
5. For every new knot $\mathbf{t}_{i,j}$, j=1 to n, find all edges $\vec{e}$ that are incident to $\mathbf{v}_i$ and their symmetric edges $\vec{e}'$ with respect to the vertex $\mathbf{v}_i$ lie outside the domain.
6. Compute the angles between $\vec{e}$ and the x-axis and sort these angles in an increasing order; we then obtain an array of angles α_k, $k = 0, 1, ..., m$, where m is the number of edges incident to $\mathbf{v}_i$.
7. Find the largest difference between two consequent angles in the array $\delta_k = \alpha_{k+1} - \alpha_k$, i.e. $\delta_{max} = max(\delta_k)$.
8. Create a knot $\mathbf{t}_{i,j} = \mathbf{v}_i - r\mathbf{d}$, where $\mathbf{d} = (cos(\frac{\delta_{max}}{2}), sin(\frac{\delta_{max}}{2}))$.
9. If $i < n$, go to Step 5; otherwise, go to Step 2.
10. Find all edges $\vec{e}$ incident to v_i.
11. Repeat Steps 6 and 7;
12. Create a knot $\mathbf{t}_{i,j} = \mathbf{v}_i + r\mathbf{d}$.
13. If $i < n$, go to Step 5; otherwise, go to Step 2.

4.2.2 Least Squares Approximation

To achieve arbitrarily high order optimal continuity of a simplex spline function, the modeling system in this thesis is designed for arbitrary degree simplex spline functions. The goal is to calculate control coefficients based on given triangulation and knot configuration.

Given data points ($\mathbf{u}_i$, f_i), i=0 to N_p, where N_p is the number of data points $\mathbf{u}_i \in \mathbf{R}^2$, we use as our objective function for least squares approximation the following:

$$F(\mathbf{u}) = \sum_{i=0}^{N_p} [\, f_i - \sum_{j=0}^{m_i} \sum_{|\beta|=n} N_\beta^j(\mathbf{u}_i) c_\beta^j \,]^2 \tag{4.2}$$

where m_i is the number of triangles in the neighborlet of the triangle $\triangle$ enclosing $\mathbf{u}_i$ and c_β^j are unknown control coefficients. Since $N_\beta^j(\mathbf{u}_i)$ are evaluated constants and some of the control points are common control points, the matrix on the right side of Equation 4.2 is a banded matrix with bandwidth = $\max\{l_j - k_j\}$ for all $\triangle$ in T, where $\{l_j - k_j\}$ is the {maximum numbering - minimum numbering} of control coefficients of the neighborlets of the $\triangle$. We can solve the above system by standard numerical solver provided in NAG [74].

Note: Adding, deleting or slightly perturbing a data point in a triangle will change the fitting function in a local area (i.e., the triangle neighborlet and the triangles sharing at least one vertex with the neighborlet).

4.2.3 Error Estimation

For points in different domain triangles, the errors are different. Error in triangle $\triangle_i$, where $i = 0, ..., N_t$ and N_t is the number of triangles in the triangulation, is denoted as E_i. Suppose there are m data points in $\triangle_i$, then:

$$E_i = \sqrt{\frac{1}{m} \sum_{j=0}^{N} \left[f_j - \sum_{I \in T} \sum_{|\beta|=n} N_\beta^I(\mathbf{u}_j) c_\beta^I \right]^2} \tag{4.3}$$

We need to find the maximum error E_{max} over all the domain triangles and the corresponding triangle $\triangle_{E_{max}}$.

4.2.4 Adaptive Domain Subdivision

It is possible that the error resulting from a least squares approximation over the current domain triangulation is greater than a predefined threshold ϵ (> 0), i.e., $E_{max} > \epsilon$. Based on the fact that scientific data is usually dense, we can subdivide the domain and repeat the knot generation and approximation process to obtain a smaller approximation error. This process is repeated until a satisfying result is obtained.

There are four approaches to subdivide a domain. The following shows the four subdivision strategies on a triangulation. Suppose the slightly shaded triangle has maximum error and we need to subdivide it in a certain way. The subdivision may propagate to other triangles. The following four methods to subdivide the domain may be used.

1. Subdivide every domain triangle into 4 equal triangles; each of which is $\frac{1}{4}$ of its parent triangle. (See Figure 4-6.)

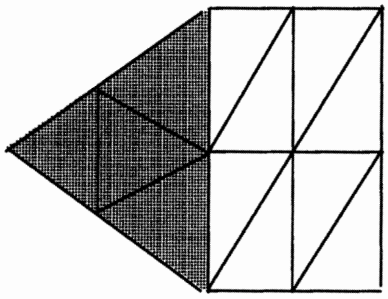


Figure 4-6: Subdivide every triangle into four.

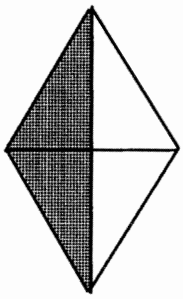


Figure 4-7: Subdivide the triangle with E_{max} at its longest edge into two and the adjacent triangle into two.

2. Subdivide the triangle with the largest error E_{max} into two triangles at the middle point of its longest edge and consequently the triangle that shares the edge. (See Figure 4-7.) This type of subdivition propagates to all the triangles.

3. Combine the above two methods and subdivide the shaded triangle $\triangle_i$ into two or four triangles depending on the shape of the triangle. If the triangle has an angle $\alpha > \frac{\pi}{2}$, then subdivide the $\triangle_i$ into two and the adjacent triangle(s) $\triangle_j$ ($j \in \{0,1,2,3\}$) into two or four, also depending on the shape of the triangle $\triangle_j$. The subdivsion is propagated to other triangles, see Figure 4-8.

4. Finally, the fourth approach can be taken to restrict the propagation of subdivisions, see Figure 4-9.)

The third and fourth approaches create less triangles and can make the triangles more equiangular.

4.3 3D Data Modeling

The analysis for 2D data modeling can be extended to 3D. In this section, we briefly address the major points in 3D data modeling.

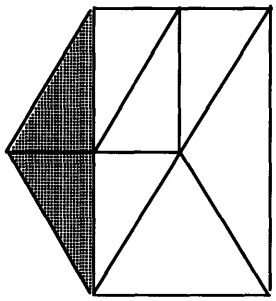


Figure 4-8: Subdivide the triangles into two or four depending on their shapes. Starting from the one with E_{max}, if it has an inner angle $> \frac{\pi}{2}$, then subdivide it into two; otherwise into four. Then subdivide the triangle(s) sharing the subdivided edge(s) into two or four depending on their shapes and the length of the subdivided edges.

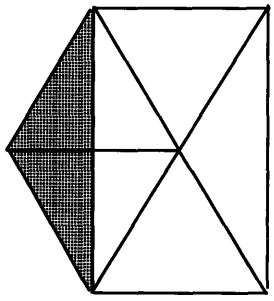


Figure 4-9: Subdivide the triangles into two or four depending on their shapes. This type of subdivision restricts the propagation.

Apart from preprocessing of the data addressed in Chapter 2, the tasks of 3D data modeling include:

1. knot generation;
2. least squares approximation;
3. error estimation; and
4. subdivision.

As for the 2D problem, these processes can be automated in order to reduce the dependence on a sophisticated user of the modeling system.

4.3.1 Knot Generation

Knot configurations associated with domain tetrahedra also are important in trivariate function generation for the same reasons stated in previous sections. The difference is that if four knots in one tetrahedron are co-planar then the order of continuity decreases by one; if they are nearly co-planar then the jump in (at least) one of the second order partial derivatives across any of the planes formed by three knots is very high. Knot configuration also influences the second order partial derivatives of the trivariate function [20]. Therefore constraints for 3D modeling are given below:

1. The knots $\mathbf{t}_{i,1}, ..., \mathbf{t}_{i,n}$ associated with the vertices $\mathbf{t}_{i,0}$ should be contained in a sphere centered at $\mathbf{t}_{i,0}$ with radius $r < \frac{1}{2}L_{min}$, where, similar to its counterparts in 2D, L_{min} is the shortest edge of all the edges incident to $\mathbf{t}_{i,0}$.

2. To achieve optimal smoothness C^{n-1} with degree n simplex spline, it is desirable that any four knots associated with a domain tetrahedron are non-coplanar. Denote a plane formed by three knots a *knot plane* or *forbidden plane*, then new knots should avoid being on any knot plane (forbidden plane). Also, the new knot with all the other knots form a new set of knot planes that impose additional constraints for future knot generation.

3. Keeping the distance to the forbidden planes bounded from below assures that the second derivatives of the spline basis remain bounded. Therefore, four nearly co-planar knots should be avoided in knot generation. It is better to generate a knot between two knot planes having equal distance to the two knot planes.

4. Knots associated with boundary vertices should be placed in the space $K \subset \mathbf{R}^3$ defined by the following criteria.

   (1) If the domain of definition is D, then the space for knots is out of D, i.e., $K \cap D = \phi$.

   (2) For points $\mathbf{t} \in K$, if we make a symmetric point $\mathbf{t}'$ of $\mathbf{t}$ with respect to the line linking $\mathbf{t}$ and $\mathbf{t}_i$, $\mathbf{t}'$ should be in one of the boundary tetrahedra incident at $\mathbf{t}_i$, i.e., $\mathbf{t}' \in \triangle_i \subset D$.

By this constraint, the knot planes containing boundary knots do not *cut* the domain tetrahedron D, which ensures the volume Ω in boundary tetrahedron will be maximum. These criteria make the function along the boundary of the domain to be created properly[2].

5. Knots associated with interior vertices can be placed anywhere in the spheres (except the forbidden planes).

Also, the number of tetrahedra incident at a vertex should not be large.

Based on these constraints, we developed an automatic knot generator for trivariate simplex spline functions of degree up to n ($n \geq 2$) similar to the 2D knot generator scheme.

4.3.2 Least Squares Approximation

The database in this thesis includes trivariate simplex spline functions with arbitrary degree n which can achieve up to C^{n-1} continuity. For a given tetrahedrization and knot configuration, the goal is to calculate control coefficients c_β^I.

Given data points $(\mathbf{u}_i, f_i)$, $i = 0$ to N_p, where N_p is the number of data points $\mathbf{u}_i \in \mathbf{R}^3$, we use as our objective function for least squares approximation the following:

$$F(\mathbf{u}) = \sum_{i=0}^{N_p} [\, f_i - \sum_{j=0}^{m_i} \sum_{|\beta|=n} N_\beta^j(\mathbf{u}_i) c_\beta^j \,]^2 \tag{4.4}$$

where m_i is the number of tetrahedra in the neighborlet of the tetrahedron enclosing $\mathbf{u}_i$, $j = 0, 1, ..., m_i$, and c_β^j are unknown control coefficients. Since $N_\beta^j(\mathbf{u}_i)$ are evaluated constants and some of the control coefficients are shared by several tetrahedra, the matrix at the right side of Equation 4.4 is a banded matrix with bandwidth = $\max\{l_j - k_j\}$ in all $\triangle$, where $\{l_j - k_j\}$ is the {maximum numbering - minimum numbering} of control coefficients of the neighborlets of the tetrahedron. We can solve the above system by standard numerical solver provided in NAG [74].

Note: similar to 2D data modeling, adding, deleting or slightly perturbing a data point in a tetrahedron will change the fitting function in a local region (i.e. the tetrahedron neighborlet and the tetrahedra sharing at least one vertex with the neighborlet).

[2]Otherwise, at some points in the domain near the boundary, the summation of the normalized basis functions may not be unity.

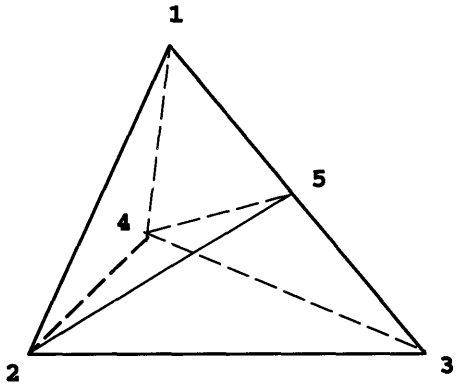


Figure 4-10: Subdivision of a tetrahedron into two on its longest edge.

4.3.3 Error Estimation

For data points in different domain tetrahedra, the errors are different. Error in tetrahedron $\triangle_i$ is denoted as E_i. Suppose there are m data points in $\triangle_i$, then:

$$E_i = \sqrt{\frac{1}{m}\sum_{j=0}^{N}\left[f_j - \sum_{I\in T}\sum_{|\beta|=n} N_\beta^I(\mathbf{u}_j)c_\beta^I\right]^2} \tag{4.5}$$

Finally, we can find the maximum error and the corresponding tetrahedron $\triangle_{E_{max}}$. If maximum error is greater than a given error threshold ϵ, we will subdivide certain domain tetrahedra and perform knot generation, least squares approximation, and error estimation repeatedly until the maximum error is smaller than the threshold ϵ.

4.3.4 Adaptive Domain Subdivision

Suppose $\epsilon > 0$ is the predefined threshold, then based on the same reasons stated in Section 4.2, if $E_{max} > \epsilon$, then the domain needs to be subdivided. The following three methods to subdivide the domain may be used:

1. Subdivide every domain tetrahedron into eight tetrahedra; see Figure 4-11.

2. Subdivide the tetrahedron with the largest error E_{max} into two tetrahedra at the middle point of its longest edge and consequently all the tetrahedra that share this edge; see Figure 4-10.

3. Combine the above two methods and subdivide the tetrahedron $\triangle_i$ with the largest error into two or four tetrahedra depending on the shape of the tetrahedron. Define the aspect ratio e of a tetrahedron as $e = \frac{l_{min}}{l_{max}}$, where l_{min} and l_{max} are the shortest and the longest edges of the tetrahedron. If the tetrahedron has an aspect ratio $e < 0.5$, then subdivide the $\triangle_i$ into two at the middle point of the longest edge,

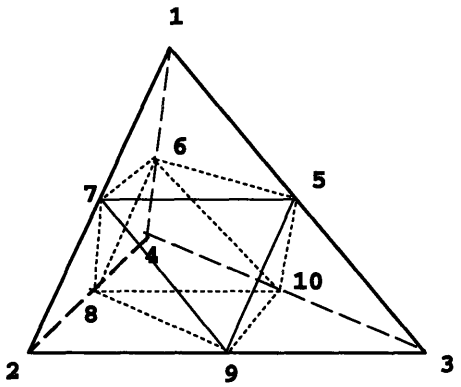


Figure 4-11: Subdivision of a tetrahedron into eight sub-tetrahedra: $\triangle_{(1,7,5,6)}$, $\triangle_{(2,9,7,8)}$, $\triangle_{(3,5,9,10)}$, $\triangle_{(4,8,10,6)}$, and the four additional tetrahedra created from the polyhedron: $P_{(5,6,7,8,9,10)}$.

otherwise subdivide the $\triangle_i$ into eight tetrahedra and all other tetrahedra incident to the edge into two or eight tetrahedra, also depending on the aspect ratio e of the tetrahedron $\triangle_i$.

4.4 Examples of Data Modeling

In this section, we present two examples. The first example shows a generated quadratic simplex spline surface by fitting 597 data points taken from a cubic polynomial function in a concave domain with a triangular hole. Figure 4-12 shows the domain of definition of this surface and Figure 4-13 illustrates the surface.

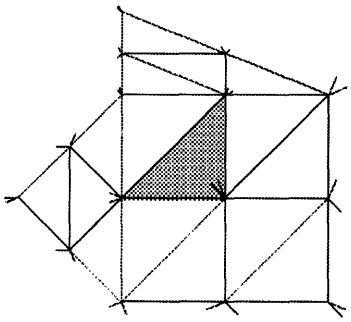


Figure 4-12: Concave domain with a hole (slightly shaded) and the knot configuration.

Our second example is a surface modeling a set of 2201 data points taken from a portion of the Charles river, Boston, MA, USA. Figure 4-14 shows the domain and the generated knots over the initial triangulation provided by the preprocessing steps in Chapter 2.

Figure 4-15 shows subdivision of the initial triangulation of Figure 4-14 and the generated knots on the new triangulation.

Figure 4-16 shows the wireframe of the surface resulting from least squares approximation on the data with maximum relative error of about 2%. Figure 4-17 shows the shaded surface resulting from least squares approximation on the data. The number of control points used in this surface is 193 and the total knots associated with the vertices in the triangulation is 59×3. Thus the total storage needed for representing the bivariate function is: 59×3×2+193×3=933 while the storage used for the measured data is 2201×3=6603, i.e., a reduction by a factor of 7.1.

Our third example applies the data modeling scheme over a set of 254 salinity data defined in a 3D domain and obtained from Mr. E. Dever of the Woods Hole Oceanographic Institution (WHOI), Woods Hole, MA, USA and consists of CTD observations made off the northern coast of California under the Shelf Mixed Layer Experiment (SMILE). The 3D domain, which is composed of 2 tetrahedra, and the three isosurfaces at isovalues $F = 9, F = 10$, and $F = 11$ (all in units of *psu*) are shown in Figure 4-18 and 4-19. The trivariate function approximation involves a maximum relative error of about 4%. The three isosurfaces are displayed with certain level of transparency. The number of control points involved is 14 and the number of knots is 5×3. Thus the total storage needed for representing the trivariate function is: 14×4+5×3×3=101, while the storage used for the measured data is: 254×4 = 1016, i.e., a reduction by a factor of 10.

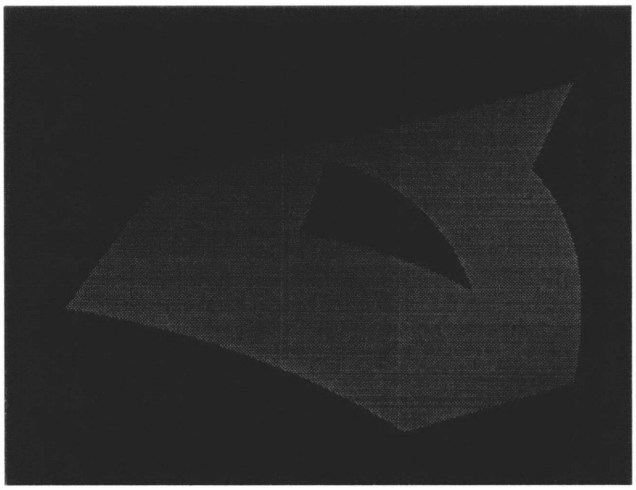


Figure 4-13: Surface defined on a 2D domain of Figure 4-12.

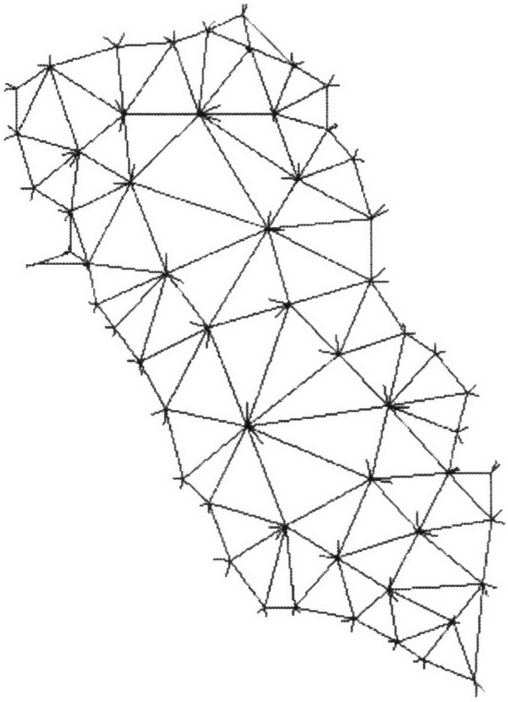


Figure 4-14: Domain and knots generated for the initial triangulation for the bathymetric data on a portion of the Charles river.

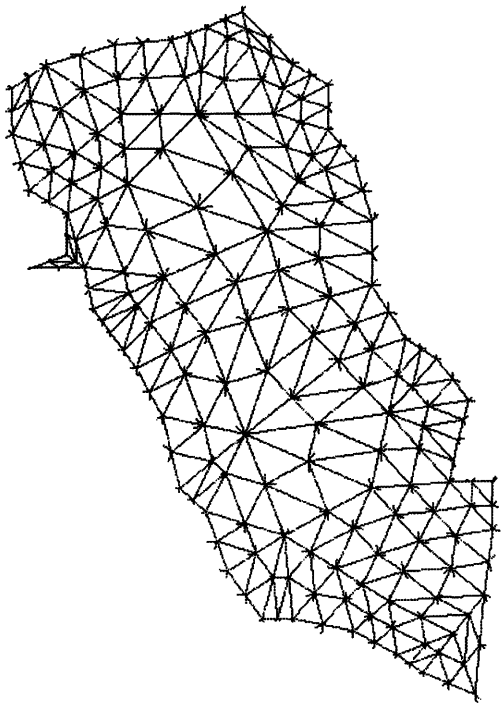


Figure 4-15: Subdivision on the initial triangulation in Figure 4-14.

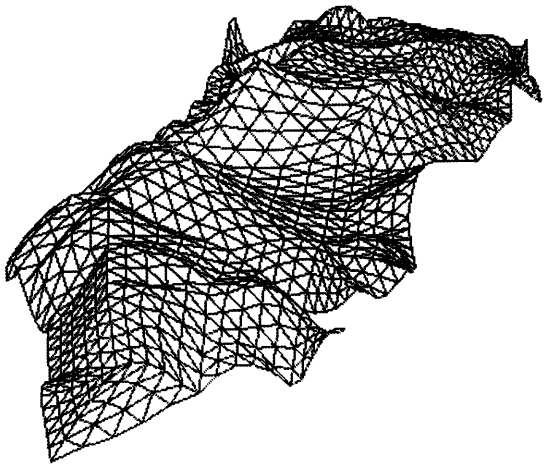


Figure 4-16: The wireframe of the topography surface of a portion of the Charles river.

Figure 4-17: The topography surface of a portion of the Charles river.

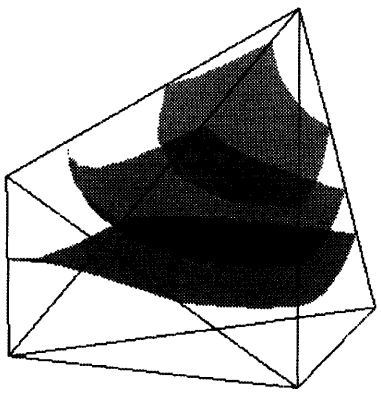


Figure 4-18: 3D domain and three isosurfaces of a trivariate simplex spline function obtained by fitting 254 salinity data. The data is obtained from WHOI under Shelf Mixed Layer Experiment (SMILE). The isovalues are $F = 9$, $F = 10$, $F = 11$, all in units of *psu*.

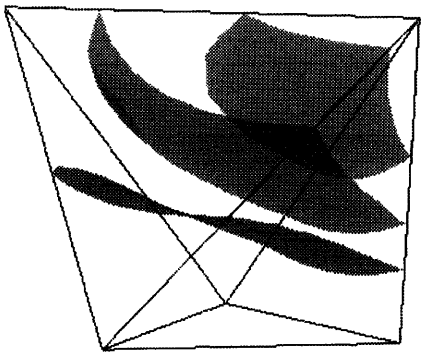


Figure 4-19: An other view of the 3D domain and three isosurfaces of a trivariate simplex spline function from Figure 4-18.

Chapter 5

Interval Simplex Spline Representations

5.1 Introduction

There is usually uncertainty in scientific data because of limitations in measurement precision, methods of measurement reporting, or difficulties in making observations and measurements in inaccessible or hostile regions. The measurement uncertainty in scientific data is typically much larger than that from floating point arithmetic. Hence, a new way to model the data in the presence of uncertainty over irregular domains needs to be introduced and studied.

The techniques developed for scientific data management have been driven by applications and are often case-specific. The tasks of a scientific database require:

1. compactness in storage to reduce data storage requirements;
2. ability to represent the data to a suitable accuracy (or resolution);
3. ability to represent uncertainty;
4. ability to visualize and interrogate in a robust manner.

These requirements have led us to use non-linear, low degree continuous piecewise polynomial functions with interval coefficients to represent the physical data to a suitable accuracy. In this chapter, we introduce the concept of Interval Simplex Spline (ISS) and apply it to represent physical data with measurement uncertainty in irregular domains for our scientific databases. In such cases, for a given set of points $\mathbf{x}_i$ in 2D or 3D and associated function values f_i, we wish to construct a bivariate (trivariate) $F(\mathbf{x})$ that satisfies certain criteria.

5.2 Interval Arithmetic

Interval arithmetic was developed three decades ago [69]. Here, we state the basic interval arithmetic operations. For further information, see Moore [69] and Alefeld [1].

- **Definition of an interval number**
  An interval number is a set of real numbers x such that
  $[a,b] = (x \mid a \leq x \leq b)$.

- **Interval arithmetic operations**
  For the usual algebraic operators $\circ \in \{+,-,*,/\}$,

$$[a,b] \circ [c,d] = \Box x \circ y | x \in [a,b], y \in [c,d] \tag{5.1}$$

  where $\Box$ corresponds to the convex hull operation. More explicitly,

$$[a,b] + [c,d] = [a+c, b+d] \tag{5.2}$$
$$[a,b] - [c,d] = [a-d, b-c] \tag{5.3}$$
$$[a,b] * [c,d] = [min(ac,ad,bc,bd), \tag{5.4}$$
$$\max(ac,ad,bc,bd)] \tag{5.5}$$
$$[a,b] / [c,d] = [a,b] \cdot [1/d, 1/c] \tag{5.6}$$

  where $0 \notin [c,d]$.

- **Midpoint of an interval**

$$m([a,b]) = \frac{a+b}{2} \tag{5.7}$$

- **Width of an interval**

$$w([a,b]) = b - a \tag{5.8}$$

Classical interval arithmetic operations still involve floating point arithmetic operations, so accuracy in interval arithmetic operations will gradually deteriorate as the floating point arithmetic operator does. *Rounded interval arithmetic* can solve this problem by performing operations in a conservative manner.

In addition, the use of bounded[1] interval operators permits the utilization of exact precision implementation on a computer for the performance of floating point operations. For example, we can redefine (5.1) such that all machine operations satisfy the *inclusion*

[1]The use of the term bounded here refers to the representation of number values by a pair of truncated real numbers such that the number value is bounded up to the last significant digit.

principle

$$[a,b] \circ [c,d] = [e,f] \in \left[e - \epsilon^l, f + \epsilon^u\right] \tag{5.9}$$

where ϵ^u or ϵ^l are associated with e or f, respectively, and are defined as $\epsilon = 2^{exp-53}$ for the IEEE standard double-precision having 64 bits, 8 bytes word size, and stored in a binary form as $(\pm)m \cdot 2^{exp}$, where m is the *mantissa* ($0.5 \leq m < 1$) and *exp* is the *exponent* [65].

Utilizing (5.9) results in the following redefinition of (5.1), namely

$$[a,b] + [c,d] = \left[a + c - \epsilon^l, b + d + \epsilon^u\right] \tag{5.10}$$

$$[a,b] - [c,d] = \left[a - c - \epsilon^l, b - d + \epsilon^u\right] \tag{5.11}$$

$$[a,b] * [c,d] = \left[\min(ac, ad, bc, bd) - \epsilon^l, \right. \tag{5.12}$$

$$\left. \max(ac, ad, bc, bd) + \epsilon^u\right] \tag{5.13}$$

Interval arithmetic is *commutative* and *associative.*

$$[a,b] + [c,d] = [c,d] + [a,b] \tag{5.14}$$

$$[a,b] \cdot [c,d] = [c,d] \cdot [a,b] \tag{5.15}$$

$$[a,b] + ([c,d] + [e,f]) = ([a,b] + [c,d]) + [e,f] \tag{5.16}$$

$$[a,b] \cdot ([c,d] \cdot [e,f]) = ([a,b] \cdot [c,d]) \cdot [e,f] \tag{5.17}$$

But it is not *distributive*, however, it is *subdistributive.*

$$[a,b] \cdot ([c,d] + [e,f]) \subseteq [a,b] \cdot [c,d] + [a,b] \cdot [e,f] \tag{5.18}$$

5.3 Interval Functions

In this work, we treat interval functions as functions represented by interval coefficients that are evaluated at real parameter values. For example, Figure 5-1 shows the function

$$f(t) = [f^l(t), f^u(t)] \tag{5.19}$$

evaluated at the parameter value $t = a$ such that

$$f(a) = [f^l(a), f^u(a)] \tag{5.20}$$

Finally, note that interval functions represent bounded values and are not uniquely defined in the interior. For example, the function shown in Figure 5-1 can represent uncertainty by either a uniform probability distribution (i.e., the function value is equally likely to occur anywhere inside the bounds), or by a normal (or Gaussian) probability where the bounding values represent the first standard deviation of the function. The different representations do not affect the method of surface reconstruction described in the following section but

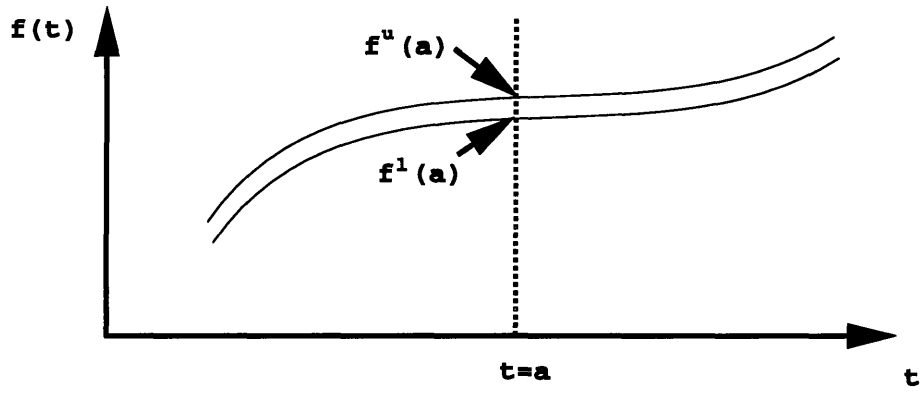


Figure 5-1: Interval function representation (adapted from Tuohy [96]).

depend upon the application.

5.4 Interval Simplex Spline Functions

In our approach, instead of using plain floating point representations of boundary spline functions, we explore a new representation based on *interval simplex splines* (ISS) geometries. ISS differs from classical simplex splines in that the real numbers representing control coefficients are replaced by *intervals*. Hence, the classical control coefficients are replaced by rectangular boxes. This implies that in 3D space, an ISS surface represents thin volumes typically visualized as shells; and in 4D space, an ISS hypersurface represents hyper-volumes embracing all possible behavior of the measured data. The ISS concept comes up naturally when formalizing computational accuracy on computing machinery. Namely, a floating point representation of a simplex spline patch is not a true 3D object over 2D domain, but rather an object enclosed by a shell containing a family of surface patches, whose representation fulfills the error bounds controlled by floating point arithmetic.

The ISS method can be used to represent uncertainty and to reduce data. In this section we want to illustrate with an example how the concept of ISS can be used to represent uncertainties in data. The first figure in Figure 5-2 represents a sequence of rectangles. This sequence of rectangles contains a discrete sequence of point data corresponding to values assigned to a continuous, univariate function. As those data have uncertainties, we do not know the precise function value assigned to the independent variable. Moreover it is also possible that we do not even know precisely the value of the independent variable to which the uncertain data value is attached. Therefore, we only know with certainty that a single data point (whose two coordinates are given by the uncertain independent variable value and the uncertain function value respectively) must be contained in a specific rectangular box. We may also have reasons to assume that the graph of the continuous function is contained in the area bounded by the bold drawn curves used to enclose the sequence of rectangles. The second picture in Figure 5-2 illustrates the same concept now with an example of a real

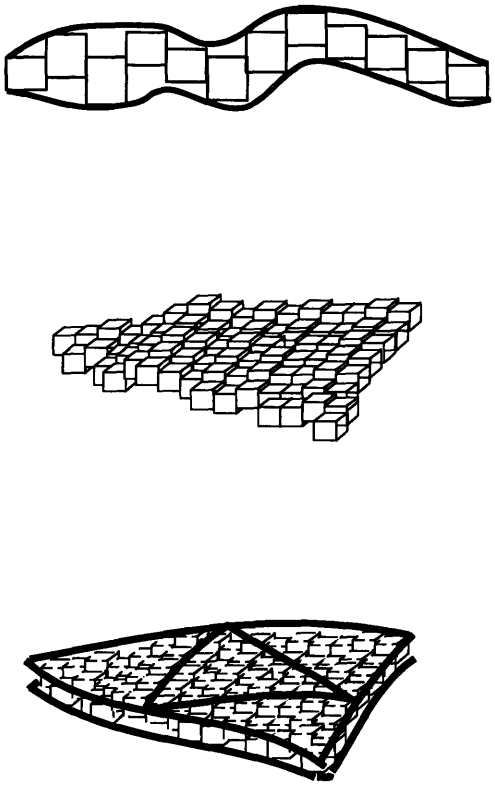


Figure 5-2: Interval simplex splines. Top: in one-dimensional space; middle: interval data set in two-dimensional space; bottom: interval surface over a 2D domain.

valued function defined over a two-dimensional domain. Finally, the last picture in Figure 5-2 represents the volume covered by an interval spline function (defined over a triangulation) used to approximate a collection of discrete uncertain data points given in a collection of 3-dimensional polyhedra similar to the middle picture of Figure 5-2. Theoretical estimates from approximation theory indicate that under reasonable assumptions, the approximating interval spline needs for its representation orders of magnitude less in storage than the data amount of the (uncertain) point data which it approximates [104].

For a bivariate (trivariate) simplex spline function $S(\mathbf{u})$, on a given 2D (3D) triangulation and knot configurations K, the control coefficients are defined by an interval $C = [C^l, C^u]$.

The definition for interval bivariate (trivariate) simplex spline functions $I_S(\mathbf{u})$ of degree n is:

$$I_S(\mathbf{u}) = [S^l(\mathbf{u}), S^u(\mathbf{u})] = \sum_{I \in T} \sum_{|\beta|=n} N_\beta^I(\mathbf{u}) C_\beta = \sum_{I} \sum_{|\beta|=n} N_\beta^I(\mathbf{u}) [C_\beta^l, C_\beta^u] \tag{5.21}$$

where T is the triangulation in 2D/3D space, $\mathbf{u} \in \mathbf{R}^s$, $s \in \{2, 3\}$.

Advantages of ISS are: (1) automated error control, (2) economical memory space, (3) machine representability, and (4) generality. ISS combined with rounded interval arithmetic can automatically control rounding errors induced by floating point arithmetic. The ISS will only use less than two times memory space as a simplex spline with real coefficients, so memory space of ISS is not too demanding. Importantly, ISS are represented and processed reliably in standard floating point arithmetic.

5.5 Construction of Bivariate (Trivariate) ISS Functions

The ISS construction procedure begins with bucket-sorting measured values into a (non)-uniformly distributed grid of cells. The cellular data is characterized (according to the measured values f_i) by a minimum and maximum measured value for a region $\mathbf{u} \in T \subset \mathbf{R}^s$, $(s \in \{2,3\})$. To derive vertices where groupings of empty cells occur, a linear interpolation of a faceted model obtained from a Delaunay triangulation of the vertices from non-empty cells is used as in Tuohy [96].

Upper and lower boundary functions, in the form of a quadratic simplex spline, are then fit by minimizing the difference between the bounding simplex spline functions with the constraint that the upper (lower) simplex spline function must lie completely above (below) the upper (lower) measured values f_i. Continuity is maintained automatically by making the knots associated with a triangle (tetrahedron) non-collinear (non-coplanar).

5.5.1 Preprocessing

In this section, we take an approach similar to that described in [96, 106].

Non-uniformly distributed cells $c_{q,r}$ $(c_{q,r,s})$ consist of the minimum $c^l_{q,r}$ $(c^l_{q,r,s})$ and the maximum $c^u_{q,r}$ $(c^u_{q,r,s})$ values of a given set of measured data for a particular region $[x_q + \triangle_x, y_r + \triangle_y]$ in 2D ($[x_q + \triangle_x, y_r + \triangle_y, z_s + \triangle_z]$ in 3D) where x, y and z are coordinates of the data points in Cartesian coordinate system. For scattered data, the sorting procedure $c^u_{q,r} = max(f_{q,r})$ and $c^l_{q,r} = min(z_{q,r})$ ($c^u_{q,r,s} = max(f_{q,r,s})$ and $c^l_{q,r,s} = min(z_{q,r,s})$) where

$$q = \left\lfloor \frac{x_k}{\triangle_x} \right\rfloor, \quad r = \left\lfloor \frac{y_k}{\triangle_y} \right\rfloor, \quad s = \left\lfloor \frac{z_k}{\triangle_z} \right\rfloor. \tag{5.22}$$

for $k = 0 \ldots n-1$ can be used to place the scattered data into the appropriate cells in $O(n)$ time.

After the cells have been constructed, we check each cell with respect to the domain. If a cell is completely outside of the domain, then we remove this cell from the cell list. For the remaining cells, we check if a cell is empty, i.e., if it contains no data points. If so, we use a quadratic polynomial to fit the cell with the points in the neighborlet of the cell and find the upper and lower bounds of the cell. Finally, the cells in the cell list are ready to serve in the interval simplex spline fitting.

5.5.2 Fitting 2D Data with ISS Bivariate Functions

The general goal for the ISS fitting algorithm is to construct a bivariate function $I_S(\mathbf{u}) = [S^l(\mathbf{u}), S^u(\mathbf{u})]$ which minimizes the approximation error and still conforms to the shape of the data. For interval bivariate functions, upper and lower bounding functions, in the form of quadratic simplex spline functions, are fit by minimizing (maximizing) the volume under the simplex spline function with the constraint that the upper (lower) simplex spline function must lie completely above (below) the upper (lower) the measured values $g = [g^l, g^u]$. Therefore, we define the objective functions for the two separate minimization problems as:

$$F = \int_T S^u dA = \sum_{\Delta} \sum_{|\beta|=n} C^u_{\Delta,\beta} \int_T N^{\Delta}_{\beta} dA \tag{5.23}$$

$$G = -\int_T S^l dA = -\sum_{\Delta} \sum_{|\beta|=n} C^l_{\Delta,\beta} \int_T N^{\Delta}_{\beta} dA \tag{5.24}$$

subject to the following constraints:

$$\sum_{\Delta} \sum_{|\beta|=n} N^{\Delta}_{\beta}(\mathbf{u}_i) C^u_{\Delta,\beta} \geq g^u_i \tag{5.25}$$

$$-\sum_{\Delta} \sum_{|\beta|=n} N^{\Delta}_{\beta}(\mathbf{u}_i) C^l_{\Delta,\beta} \geq -g^l_i \tag{5.26}$$

respectively, where $dA = dxdy$.

This minimization problem with linear constraints can be solved using techniques for linear programming [74, 46]. The integral of normalized simplex spline basis functionN^{Δ}_{β} is calculated by the scheme presented in Section 3.1.4.

5.5.3 Fitting 3D Data with ISS Trivariate Functions

The construction of interval trivariate simplex spline functions involves defining a function $I_F(x, y, z)$, where $I_F(x, y, z) \in \left[F^l(x, y, z), F^u(x, y, z)\right]$.

The goal of this trivariate function construction is to model a set of function values f_i measured at spatial data points: (x_i, y_i, z_i) where $i = 0, ..., N$ and $x_i \in [x^l_i, x^u_i], y_i \in [y^l_i, y^u_i], z_i \in [z^l_i, z^u_i], f_i \in [f^l_i, f^u_i]$, where N is the number of points.

Each data point (x_i, y_i, z_i, f_i) in 4D space forms a 4D box, or simply, 4-box. The 4D point cells can be represented by a trivariate function. The approach towards the interval trivariate function construction based on ISS is to fit spatial uncertain data by minimizing (maximizing) the hyper-volume under the ISS function with the constraint that the evaluated upper (lower) ISS trivariate function must be completely above (below) the upper (lower) measured values. That is, we seek a minimum to each of the following:

$$F = \int_V S^u dv = \sum_{\Delta} \sum_{|\beta|=n} C^u_{\Delta,\beta} \int N^{\Delta}_{\beta} dv \tag{5.27}$$

$$G = -\int_V S^l dv = -\sum_{\Delta} \sum_{|\beta|=n} C^l_{\Delta,\beta} \int N^{\Delta}_{\beta} dv \tag{5.28}$$

where $dv = dxdydz$, C^u_β and C^l_β are unknown control coefficients subject to:

$$\sum_{\Delta} \sum_{|\beta|=n} C^u_{\Delta,\beta} N^{\Delta}_{\beta}(x_i, y_i, z_i) \geq f^u_i \tag{5.29}$$

$$-\sum_{\Delta} \sum_{|\beta|=n} C^l_{\Delta,\beta} N^{\Delta}_{\beta}(x_i, y_i, z_i) \geq -f^l_i \tag{5.30}$$

respectively.

The integral of the normalized simplex spline basis function is defined in Section 3.2.4.

Analogous to Section 5.5.2, the minimization problem can be solved by using techniques for linear programming [74, 46].

5.6 Examples of Interval Simplex Spline Surfaces

In this section, we give an example to illustrate the concept of ISS. We use the set of data points taken from a portion of the Charles river. Figure 5-3 and 5-4 show the wireframes of upper and lower ISS surface. Figure 5-5 shows the resulting interval surface. The upper surface is shown in light wireframe and the lower is shaded.

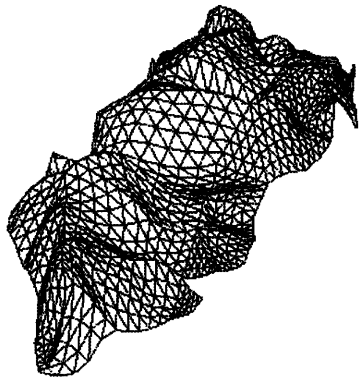


Figure 5-3: The upper surface for a set of data taken from a portion of Charles river.

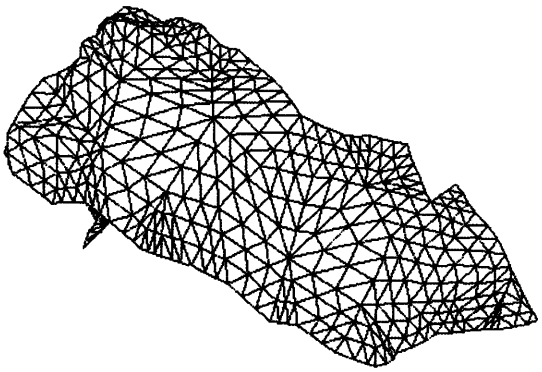


Figure 5-4: The lower surface for the same set of data as shown in Figure 5-3.

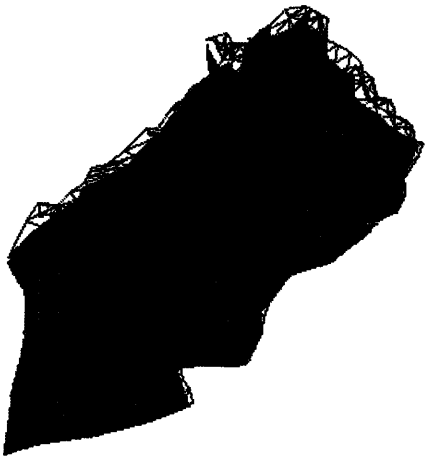


Figure 5-5: The ISS surface corresponding to Figures 5-3 and 5-4.

Chapter 6

Marching Simplices for Interrogation

6.1 Introduction

To help in understanding the physical phenomena represented by data, it is important and necessary to have an efficient method for visualization which also facilitates subsequent interrogation on the data. The domains in our data modeling system are based on triangulation. This feature requires our visualization method to be capable of visualize physical data defined in this type of domain and, more importantly, prepare for further interrogation on physical data over the domain.

In this chapter, we present a novel method for visualizing and evaluating implicit functions defined over irregular domains. Evaluation of implicit or isovalued surfaces is a powerful rendering technique for visualizing trivariate functions and there exist many mature methods to perform this operation (e.g. marching cubes). Most methods, however, assume the presence of a regular rectangular domain, i.e., a domain that can be decomposed easily into square or cubical elements (e.g. quadtree, octree, and voxel). To visualize functions based on real data (from finite element analysis, computational fluid dynamics, geographic information systems, etc.), which can be defined over irregular domains represented by triangulations in 2D (3D) Euclidean space, we develop the *marching simplices* algorithm. The marching simplices algorithm begins with discretizing the domain triangulation into small grain simplices. In each grain simplex, an implicit or isovalued surface is approximated by comparing the function values at the vertices of the grain simplex with the isosurface value and establishing an approximation of the topology of the surface in each grain. Adaptive subdivision methods are used to capture small scale features. To enhance the utility of the resulting polygonal representation, the topological relations between the facets of the polygonalized surface are emphasized and maintained in a cell-tuple topological modeling structure.

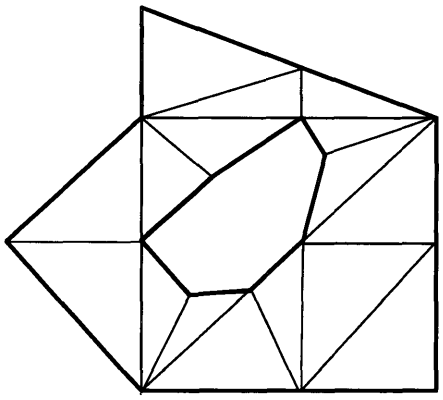


Figure 6-1: A domain in $\mathbf{R}^2$ represented by a triangulation with a polygonal hole.

6.2 Problem Definition

Assume a function $F : D \subset \mathbf{R}^n \rightarrow \mathbf{R}, n \in \{2,3\}$ is defined in a domain D which can be described by a triangulation T. Our goal is to extract implicit or isovalued surfaces in an irregular domain geometrically and topologically on which this function F is equal to a constant C. This is equivalent to rendering surfaces defined by points $\mathbf{x}_i$ such that $F(\mathbf{x}_i) = C$, where $\mathbf{x}_i \in D$.

The function F could be a piecewise polynomial, a transcendental function, or a procedurally defined function. The function is assumed to be continuous in the entire domain D.

In this work, we assume that the grain simplices are small. We use small simplices to decompose the domain, since, as stated in the previous section, the domain can be represented by triangulation. Figure 6-1 shows a 2D domain represented by a triangulation with a polygonal hole.

By using simplices as grains, we can model general irregular domains. Using simplices for determining the topology of pieces of surface in the grain, we have the following advantages:

- we eliminate the topological ambiguities that appear in marching cubes algorithms;
- in addition, using the cell-tuple topological modeling structure to represent both the triangulated domain and the polygonalized surface, we encode the adjacency information between facets of the generated surface;
- another benefit from the topological structure is that the generated surface can be used for further interrogation directly;
- by adaptive subdivision, we can describe small features of the surfaces;
- and finally, using simplices as grain elements requires fewer cases in the look-up-table which reduces programming complexity.

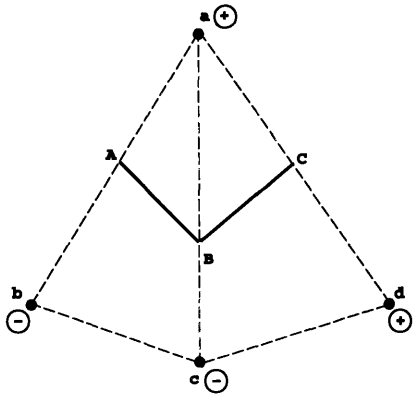


Figure 6-2: Two adjacent triangles in a domain (dashed lines). $\overline{AB}$ and $\overline{BC}$ represent an isovalued implicit curve $F = C$.

6.3 The Relation between Domains and Surfaces

In Chapter 2, we described the Topological Data Structure (TMS) and applied it into representing domain triangulations. In this chapter, we use TMS to represent both the domain triangulation and a polygonalized surface within the domain. Now let us take a closer look at how the polygonalized implicit or isovalued surfaces are represented in the TMS. Figures 6-2 and 6-3 show a 2D example: two triangles represented by the TMS which gives the topological relation among cells. First, three intersection points are found on edges at whose end points the function being contoured has opposite signs. These intersection points can be accessed by corresponding domain edges. Then, for each grain triangle, edges of the surface piece incident at intersection points are created. In this case, two edges $\overline{AB}$ and $\overline{BC}$ are created, both incident at B. Thus, the adjacency relations between the generated contour segments are built up automatically. In other words, this process automatically leads to a TMS for the contour in which $\overline{AB}$ and $\overline{BC}$ are adjacent at B.

Similarly in a 3D case, Figures 6-4 and 6-5 illustrate how the adjacency relation between two surface facets is represented automatically by the TMS. First, the intersection points on domain edges are calculated as A, B, C and D and their corresponding 0-dimensional nodes are created. These 0-dimensional nodes can be accessed through corresponding domain's edge nodes. The topologies of the facets in each domain tetrahedron are found from a *look-up-table* (which will be presented in the Section 6.4. Then a face node (2-dimensional) and three edge nodes (1-dimensional) are created for the implicit or isovalued surface in each grain simplex, all incident at corresponding point nodes and edge nodes. Finally, the facets, edges of the facets, and vertices of the facets' edges are now in a TMS at corresponding dimension levels related by incidence relations, see Figure 6-5. In this manner, the topological relations among the facets are formed naturally and efficiently and adjacency information between any two facets is embedded in the TMS and can be retrieved with little effort.

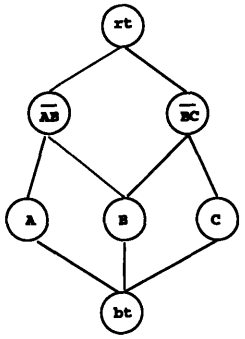


Figure 6-3: An incidence graph for a generated contour in Figure 6-2.

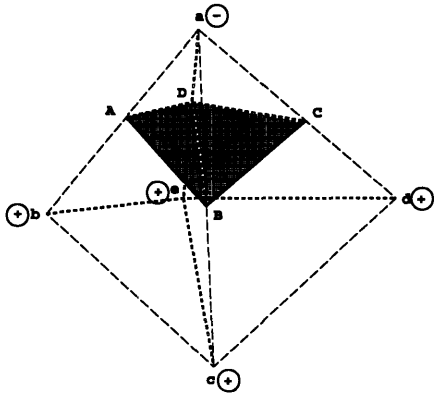


Figure 6-4: Two adjacent tetrahedra in a domain (dashed lines). Facets $\overline{ABD}$ and $\overline{DBC}$ represent an isovalued implicit surface $F = C$.

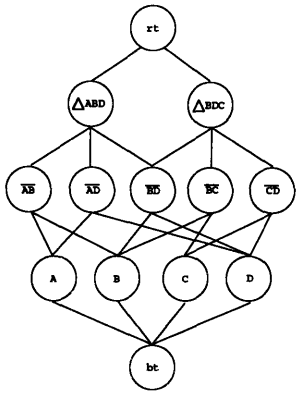


Figure 6-5: The incidence graph for the generated isovalued surface in Figure 6-4.

6.4 Approximation of Isovalued Functions

The approximation of isovalued functions with connected grid of points $\mathbf{x} \in D \subset R^s, s = \{2,3\}$ where $F(\mathbf{x}) = C$ defined in a 2D (3D) region D involves creation of straight line segments or triangular facets closely approximating the surface. These line segments or facets are calculated based on approximation and subdivision of the function domain into small grain simplices.

To generate a proper approximation for the surface piece within a grain simplex, we calculate good approximations of the intersection points between $F(\mathbf{x}) = C$ and the edges of the simplex connecting two vertices which have function values below and above the value C. (When the function values equal to C, it indicates that the surface passes through the corresponding vertices and thus are considered as a special case.) For other function values, we find the intersection points by either (1) linearly interpolating the function values; (2) binary search; or (3) *regula falsi* method, which uses two previous approximations and constructs the next approximation by making a linear interpolation between them. The interpolation point then serves as our next approximation to the root of $F(\mathbf{x}) = C$ on a given edge.

The calculated function values are associated with the corresponding nodes of vertices in the TMS representing the domain triangulation. The intersection points are defined as nodes in TMS and can be accessed through the corresponding edges in the data structure of the domain triangulation. Later, we will show how an implicit curve or surface is generated out of these intersection points.

6.4.1 Contour Lines

In this work, the domain of definition D of a scalar function $F : D \subset \mathbf{R}^2 \rightarrow \mathbf{R}$ is represented by a triangulation. For each triangle of the triangulation, function values at the three vertices are calculated. Since there are three vertices and two conditions for the sign of $F-C$ at each vertex, i.e., $F > C$ and $F < C$, there are eight cases. Eliminating topologically equivalent cases, there are only two cases left for consideration (see Figure 6-6). In the first case, since $F - C > 0$ at all vertices, (denoted as "+" in Figure 6-6), there are no contour lines in the simplex. In the second case, at one vertex the function value is greater than C and at the other two vertices the function values are less than C (denoted as "-" in Figure 6-6); thus, there exists a piece of contour line $\overline{ab}$ in the simplex. In these two cases, based on the assumption that the grain is small, there are no ambiguities. Thus, the topology of the contour line in a triangle can be easily created. Yet, in the second case, in order to detect possible small features on the contour line, we calculate the function value $F(\mathbf{p})$ at mid-point $\mathbf{p}$. Based on the function value at $\mathbf{p}$, the method will decide if the triangle needs to be further subdivided, see Section 6.5.

6.4.2 Implicit or Isovalued Surfaces

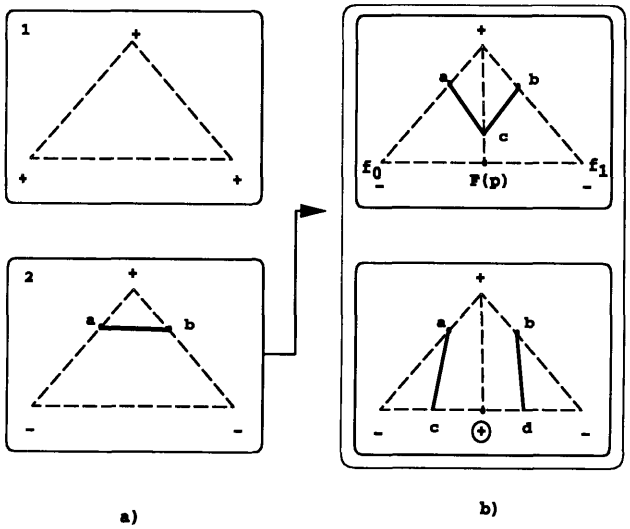


Figure 6-6: The look-up-table for marching 2-simplices and the subdivision to detect small features on the contour lines.

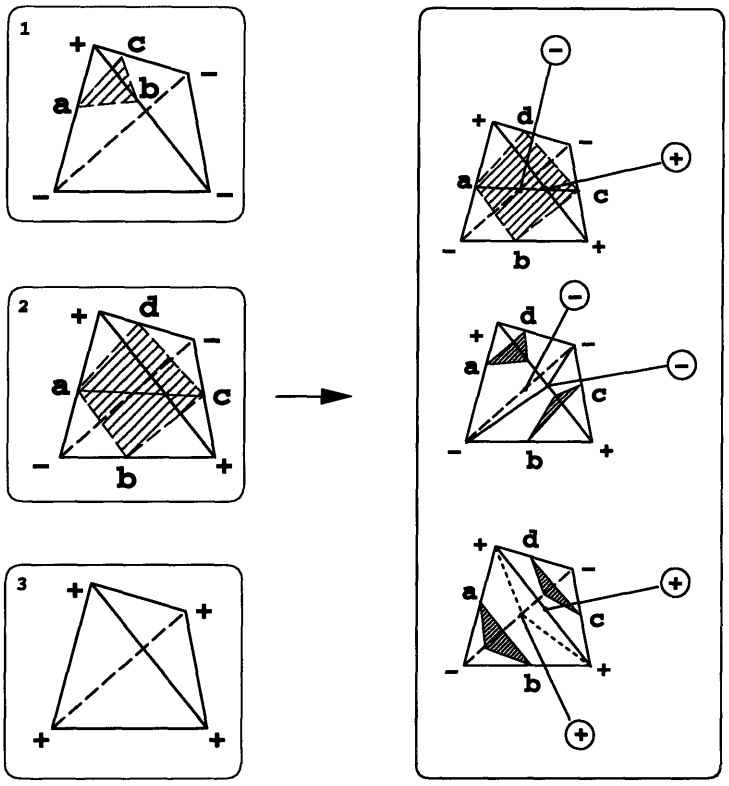


Figure 6-7: The look-up-table for marching 3-simplices and the subdivision to detect small features on the isosurface.

In this work, the domain of definition D of a scalar function $f : D \subset \mathbf{R}^3 \rightarrow \mathbf{R}$ is represented by a 3D triangulation (tetrahedrization). Function values at all vertices of the triangulation are calculated, and, for each tetrahedron there are only three cases for arrangements of the signs of $F - C$, i.e., $F > C$ or $F < C$, see Figure 6-7. In the first case, three function values are less than C, denoted at the corresponding vertices as "-" and one function value is greater than C, denoted at the vertex as "+". Thus the piece of surface in the grain is a triangle facet. In the second case, two vertices have signs "+" and two other vertices have signs "-", thus the surface in the grain can be decomposed into two triangular facets. In the third case, since all the sign of the four vertices are the same, we conclude that the grain contains no pieces of the surface. In order to capture possible small features on the surface, for the second case, we calculate the mid-points $\mathbf{p}$ and $\mathbf{q}$ on the two edges incident to the vertices possessing the same signs for $F - C$. The method will decide if the grain needs subdivision based on the function values at $\mathbf{p}$ and $\mathbf{q}$, see Section 6.5. In the three cases there are no ambiguities; thus, the topology of the implicit or isovalued surface in a tetrahedron can be created without considering and analyzing ambiguities.

6.5 Refinement of Isovalued Curves and Surfaces

Although uniformly discretizing a domain triangulation is easy to implement, it may unnecessarily create many simplices and may be time consuming in generating polygonalized implicit or isovalued surfaces. To save memory space and computation time, it is desirable to subdivide certain grain simplices adaptively. Another reason for adaptive subdivision is that certain singularities may appear in some grain simplices. A simplex under consideration is likely to contain a singularity of the surface if mid-point $\mathbf{p}$ of an edge lies in a half-space different from that of the edge's two vertices. We also use the absolute value of the function at each vertex which is called the *algebraic distance* between a point and the surface (see Patrikalakis and Kriezis [76]) in determining the condition for subdivision.

The criteria for subdivision include the following steps:

1. For a 2-simplex, if the three vertices have different signs for $F - C$ then calculate the function value at the mid-point $\mathbf{p}$ of the edge whose vertices have the same sign for $F - C$; if

   (a) the sign of $F - C$ at $\mathbf{p}$ is different from those of the two vertices, or

   (b) the absolute function value $|F|$ at $\mathbf{p}$ is less than both of the vertices' absolute function values,

   then subdivide the edge at $\mathbf{p}$ and consequently all the simplices which share the edge at point $\mathbf{p}$. (See right side of Figure 6-6.)

2. For a 3-simplex, if the simplex has two positive and two negative signs for $F - C$ then calculate the function values at the mid-points $\mathbf{p}$ and $\mathbf{q}$ of the two edges whose vertices are of the same sign, if

(a) the signs for $F - C$ at $\mathbf{p}$ or $\mathbf{q}$ are different from those of the corresponding two vertices, or

(b) the absolute function value $|F|$ at $\mathbf{p}$ or $\mathbf{q}$ is less than both of the absolute function values at the vertices, respectively,

then subdivide the edge at $\mathbf{p}$ or $\mathbf{q}$ and all tetrahedra which share the edge. (See right side of Figure 6-7).

6.6 Overview of Marching Simplices Algorithm

Our algorithm for generating a polyhedral approximation of an implicit or isovalued surface involves the following steps:

1. Input the domain triangulation and the minimum length of a grain edge. Any simplex whose longest edge is less than the minimum length will not be subdivided.
2. Use a cell-tuple topological modeling structure to represent the domain triangulation.
3. Decompose the domain into simplices.
4. Compute the function values at all vertices of the triangulation. We treat cases in which one or more function values at the vertices of the grain are equal to C as special cases.
5. Find the intersection points of $F(\mathbf{x}) = C$ with all edges of the triangulation and place the intersection points into another topological modeling structure, for the contour or surface. These points are represented as 0-dimensional nodes in the data structure for the contour or surface.
6. Adaptively subdivide simplices if needed; otherwise proceed to Step 7.
7. Find the contour or surface topology within each simplex from the look-up-tables.
8. In each face, the algorithm creates 1-dimensional nodes for edges of the surface TMS incident at those intersection-point nodes; in each tetrahedron, it creates 2-dimensional nodes for facets incident at the edges. For 2D cases, the algorithm creates 1-dimensional nodes for edge in each 2-simplex incident at those intersection-point nodes. Note: the algorithm does not make multiple nodes for the same geometric entities.
9. Output contour or surface TMS.

Note: The output of the algorithm is the polygonalized implicit (isovalued) contour or surface represented by the TMS, thus consistency between adjacent facets is resolved automatically and the surface model can be used for interrogation (e.g. Boolean operations on two implicit contour or (isovalued) surfaces, mesh generation, etc.).

So far, we described a marching simplices algorithm for 2D (3D) scalar function visualization and interrogation in irregular domains. We used TMS throughout the algorithm to represent both the domains and the generated implicit surfaces (isosurfaces) to maintain the topological information among the facets of the implicit surfaces (isosurfaces) not only for surface visualization but also for further interrogation. The algorithm can be applied to general functions as well as unstructured sample data.

Our marching simplices algorithm has several advantages over the generally used marching cubes algorithm, including:

- The ability to model irregular (and regular) domains. This is one of the most important advantages since it provides the power of modeling surfaces in domains with complex boundaries, holes and cavities.
- The generated polygonalized surface is stored in a topological data structure and can be used for further interrogation, e.g., for Boolean operations, meshing, etc.
- No ambiguity cases. This advantage makes the topology in a grain simplex clearly defined without ambiguity.
- Adaptive subdivision. This advantage is important since it enables the algorithm to efficiently resolve small features of the surfaces.
- Fewer cases for the surface topology in a grain simplex than in a cube, i.e., the look-up-table is smaller. This simplifies the design and implementation of the algorithm.

Nevertheless, marching simplices is not a panacea. Its main weakness is that, like other discrete methods, it may still miss some very small features of implicit surface, such as small isolated surface components, near self-intersection points, and other singularities.

The method presented here also can be applied in visualizing and interrogating a uniform or non-uniform sample of a function of two or three variables, i.e., direct volume rendering, in regular or irregular domains. This type of problem arises in computer-aided tomography (CAT), numerical simulations of fluid flows, etc. Constrained Delaunay triangulation can be used to triangulate the sample with a domain boundary as the constraints and the marching simplices algorithm can be utilized on the triangulation without the adaptive subdivision step.

6.7 Examples

In this section, we give a few examples to demonstrate the application of the marching simplices algorithm in visualizing and interrogating bivariate and trivariate functions over irregular domains.

6.7.1 Bivariate Functions

Our example for bivariate functions shows the contour lines for a real valued function. The domain for this problem is a square with a polygonal hole shown in Figure 6-8 as the area bounded by the outer square and the inner polygon.

The definition of the function, taken from Hoschek and Lasser[57], is:

$$F(x,y) = \frac{3}{4}e^{-\frac{1}{4}[(9x-2)^2+(9y-2)^2]} + \frac{3}{4}e^{-[\frac{1}{49}(9x+1)^2+\frac{1}{10}(9y+1)]} - \frac{1}{5}e^{-[(9x-4)^2+(9y-7)]} + \frac{1}{2}e^{-\frac{1}{4}[(9x-7)^2+(9y-3)^2]} \tag{6.1}$$

We can now represent the domain in a TMS and subdivide the domain to 1150 small triangles and generate contour lines by using our marching simplices algorithm for isovalues $C = 0.2$, $C = 0.4$, $f = 0.6$ and $C = 0.8$ shown in Figure 6-8 with 134, 333, 105, 64 segments, respectively. Note that contour lines may be topological loops entirely enclosed in the domain or they may be open curves terminating at the boundary of the domain.

6.7.2 Trivariate Functions

We present three examples for the isosurfaces of trivariate real valued functions defined in domains represented by 3D triangulations.

Our examples for isosurfaces of quadratic trivariate simplex spline functions have been shown in Figure 3-12 and 4-18 and 4-19.

Our first example in this section shows a saddle surface in a domain composed of five tetrahedra constructing a cube. The definition of the function for this example is:

$$F(x,y,z) = 4x^2 - 4y^2 + xy - 4x + 4y - z + 0.5; \tag{6.2}$$

Figure 6-9 shows the domain in red lines and the part of the saddle surface $C = 0.2101$ in the domain.

Our second example is an implicit function made up of the product of three cylinder equations. The function for this implicit surface is:

$$F(x,y,z) = (x^2+y^2-1)(y^2+z^2-1)(z^2+x^2-1) \tag{6.3}$$

Figure 6-10 shows the surface with $F = 1.0001$ in a cube. In this case, tetrahedra are discretized and some of the grain tetrahedra are further subdivided. The total number of tetrahedra in the final triangulation is 3006 and the generated surface contains 1357 facets. Figure 6-12 shows the surface defined in a domain with six faces of a cube as its exterior boundary and four faces of a tetrahedron (with four vertices A, B, C, and D) as its inner boundary. The isovalue for this case is $C = 1.0001$. The domain is shown in Figure 6-11 and Figure 6-13 shows the wireframe for the surface in the domain.

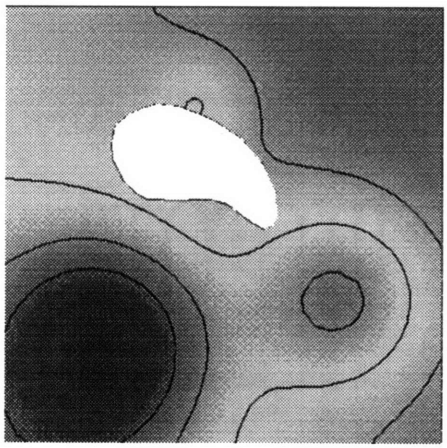


Figure 6-8: Contour lines of a bivariate function over a square with a hole.

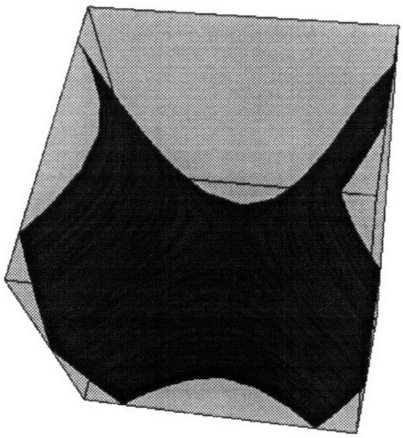


Figure 6-9: A saddle surface in a cube.

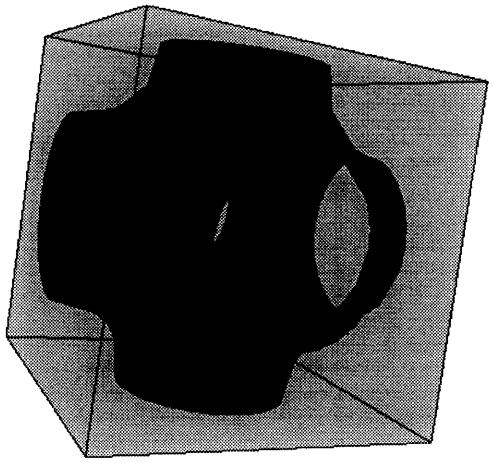


Figure 6-10: Surface of product of three cylinders in a cube.

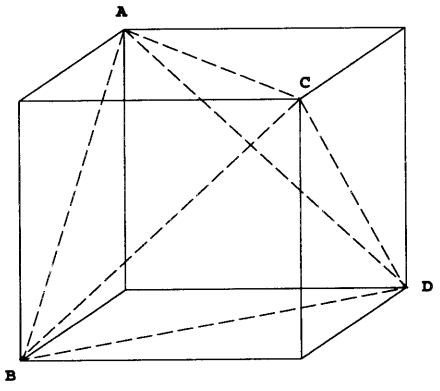


Figure 6-11: The domain of the cylinders in a cube with a tetrahedral cavity. The tetrahedron is depicted by the dotted edges and the vertices A, B, C, and D.

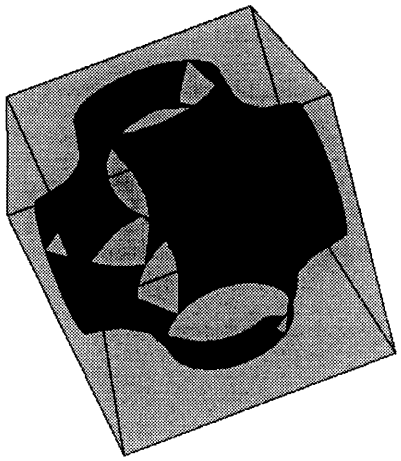


Figure 6-12: Surface of product of three cylinders in a cube with a tetrahedral cavity in the domain.

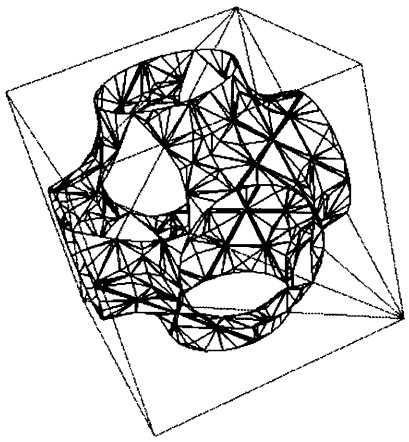


Figure 6-13: Wireframe of the implicit surface of Figure 6-12.

Chapter 7

Conclusions and Recommendations

7.1 Contributions

The major contributions in this thesis are the investigation and application of multivariate simplex splines and the development of a database system that efficiently represents, interrogates, visualizes, and manipulates large collections of multidimensional data. Methods of efficient evaluation of degree n, bivariate and trivariate simplex splines are developed and implemented in the system. In this manner, data storage is dramatically reduced; yet, in contrast to discrete data representation used in traditional database systems, all possible data locations will be covered and higher-level interrogation can be performed.

Interval simplex splines (ISS) are presented and the method for creating ISS surfaces from measured data is developed to capture all possible behavior of physical data with uncertainties in irregular domains. In this way, the uncertainties in the data set are represented in a manner that provides for high accuracy and resolution, data reduction and efficient interrogation with a guaranteed bounded approximation error.

The generalization of the convex hull of unstructured planar and spatial data is introduced and applied in defining domains of physical data such that the intuitive shape of the domain of the data can be captured in a realistic and practical manner. This α-shape can be used to detect the boundary of the domain of a finite set of points and thus to generate an appropriate domain for a set of points in 2D or 3D for least squares approximation.

Based on a cell-tuple topological modeling structure and adaptive subdivision, marching simplices, a method of visualizing bivariate and trivariate simplex spline functions as well as other general scalar functions in 2D and 3D regular and irregular domains is developed in this work. This algorithm has the following advantages: it is capable of modeling irregular domains; it needs fewer cases for determining grain partial surface topology; it eliminates ambiguity cases; it provides for automatic consistency between pieces of the surface in adjacent simplices; it generates isosurfaces which are easier to further interrogate since their topology is encoded in the surface generation.

7.2 Future Research Issues

In this thesis, we developed a database system for representing and visualizing 2D and 3D physical data with uncertainty in irregular domains. Methods for the evaluation and rendering of bivariate and trivariate simplex spline functions are developed, implemented and utilized in approximating scattered physical data. Interval simplex spline functions are defined and applied in physical data representation to capture the general form of the data with uncertainties in regular and irregular domains. In this way, the uncertainties in the data set are represented in a manner that provides for high accuracy and resolution, data reduction, and efficient interrogation with a guaranteed bounded approximation error.

Further theoretical study of simplex spline properties needs to be undertaken. In particular, more efficient evaluation algorithms (of the de Casteljau type) need to be developed. Study of properties like the linear precision property, subdivision/splitting and convex hull property needs to be undertaken. Finally, approximation of functions using simplex splines needs to be studied.

Based on this work, future research could emphasize efficient and robust interrogation schemes for simplex spline representations, for example, finding all the singular points of the surfaces using rounded interval arithmetic. A possible approach could be based on a divide-and-conquer method.

Complexity analysis of the algorithms developed in this thesis such as the marching simplices algorithm, needs to be developed.

The application of simplex spline functions in other engineering disciplines, e.g. finite element analysis, computational fluid dynamics, applied mechanics etc., should be studied.

In addition, to approximate isosurfaces robustly and accurately, a priori computation of singular points could be performed to capture small features on the surfaces such that domain discretization could be planned in a fashion similar to the way shown in Tuohy *et al.* [100]. Also visualization tools other than isosurfaces, e.g., ray casting, and glyphs, could also be developed and implemented for 2D and 3D scalar functions in irregular domains. A step beyond 3D scalar field visualization is the development of 3D vector and tensor field visualization in irregular domains.

Animation would be an interesting and important future topic, i.e., visualization of the 3D physical data with time dependence, to simulate continuous motion using rapid display of images, which is possible by modern graphics hardware. This technique can be applied to a wide range of scientific studies, for example, ocean and earthquake simulation, fluid flow, and forest growth.

Parallelism could be utilized for the database system on PVM (Parallel Virtual Machine) when rendering surfaces, and for knot generation and error estimation.

An area of potential future work could be the application of scientific database systems to Virtual Reality Environments to represent, interrogate and visualize complex real world physical phenomena in real time.

Appendix A

Delaunay Triangulation

A.1 General Delaunay Triangulation

The following description appears here for completeness and is adapted from [99].

The general problem of the Delaunay (or Thiessen) triangulation can be stated as follows: Given a set of vertices $N_i = (X_i, Y_i)$, $i = 0, \ldots, n-1$ distributed randomly (or scattered) in the $X - Y$ plane, construct a triangulation using all vertices that is as equiangular as possible. This triangulation can also be thought of as the dual to the Voronoi diagram, as shown in Figure A-1.

To construct a triangulation of the region S, for each vertex N_i define the Thiessen region associated with that vertex to be the closure of the set of points that are closer to N_i than to any other vertex. Further, a pair of vertices are defined as Thiessen neighbors if and only if their Thiessen regions share one (weak neighbors) or more (strong neighbors) points. A triangulation can thus be constructed by connecting all pairs of strong neighbors and connecting weak neighbors only when four or more vertices lie on a common circle [60].

Algorithms for triangulation can be found in Cline and Renka [14], Guibas and Stolfi [52], Sloan [89] and Lawson [60]. All algorithms use a sorting procedure of time $O(n \log_2 n)$

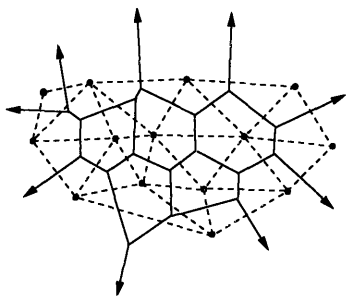


Figure A-1: The Voronoi diagram (solid) and the Delaunay diagram (dashed) (adapted from [52]).

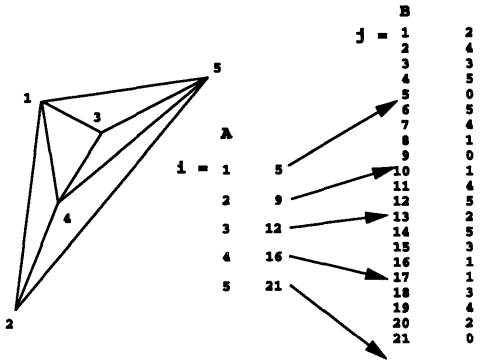


Figure A-2: Sample of triangulation and adjacency storage (adapted from [14]).

to preprocess the data. In the Guibas and Stolfi [52] algorithm, sorting is dominant whereas in the others, there is an additional $O(n^{5/4})$ time operation. For example, the algorithm by Cline and Renka [14] needs only $O(7n)$ integer type data records to record adjacency and ordering information and can be found in the NAG library [73].

The issue of insertion (or deletion) of a point in a Voronoi diagram of n points is addressed in [52] and [49]. There, it is shown that a point can be inserted in time $O(k)$ if k updates are made, which is worst case when $k = n$. This is always lower than $O(n \log n)$ which is the time needed initially to build the triangulation. If a point already included in the data set is to be updated (i.e. moved), then algorithms for this transition are outlined in [47].

Several data structures have been used in the literature to store adjacency and ordering information of the Delaunay triangulation. The data structure found in [14] has been implemented here. The structure consists of two arrays of integer type; one array $A[i]$ of length n contains pointers to the final adjacency pair for each vertex and an array $B[j]$ of length $L \leq 6n - 9$ contains a pointer to each adjacent vertex (where zero refers to a boundary adjacency). Therefore, using Figure A-2 as an example, the list of vertices adjacent to vertex 2 in the figure are $B[j]$ for $j = A[1] + 1, \ldots, A[2]$.

A.2 Constrained Delaunay Triangulation

Constrained Delaunay triangulation is a suboptimal Delaunay Triangulation with constraints imposed on predefined edges. Thus if a triangulation T is to preserve a predefined edge, it is generally not globally optimal. However, given the predefined edge, a new triangulation T^* can be constructed which is locally optimal for all strictly convex quadrangles involved. This uniquely defined triangulation T^* differs from the Delaunay triangulation T only locally.

A.3 α-Shape and Delaunay Triangulation

In this section we discuss the close relationship between α-shapes and Delaunay triangulations. Specifically, we review the fact that any α-shape of a set S of points is a subgraph of either the closest point or the furthest point Delaunay triangulation[1]. For theoretical and in-depth coverage, readers are referred to [29, 28, 30, 31].

First we present a few facts about Voronoi diagrams and Delaunay triangulations. Given a set S of n points in the plane, the closest point Voronoi diagram of $S, VD_c(S)$ is a covering of the plane by n regions V_p, $p \in S$, where

$$V_p = \{x | d(p,x) \leq d(q,x), p \neq q \in S\} \tag{A.1}$$

Similarly the furthest point Voronoi diagram of $S, VD_f(S)$ is defined by the regions

$$W_p = \{x | d(p,x) \geq d(q,x), p \neq q \in S\} \tag{A.2}$$

We will need the following properties of these diagrams.

1. The regions V_p and W_p are closed, convex, and bounded by straight line segments, called Voronoi edges, for all $p \in S$.

2. Each region V_p of $VD_c(S)$ contains p. Provided $n > 1$, each region W_p of $VD_f(S)$ does not contain p.

3. The regions V_p and W_p are unbounded if and only if p is a point on the convex hull of S. Otherwise V_p is a nonempty convex polygon and W_p is empty.

4. Two points p and q of S are said to be closest point Voronoi neighbors if V_p and W_p share a common point. Thus, two points p and q of S are closest and furthest point Voronoi neighbors if and only if (p,q) is a convex hull edge of S.

5. The closest point Delaunay triangulation of S, $DT_c(S)$, is defined as the straight line edge between p and q if and only if they are closest point Voronoi neighbors. Both V_p and W_p (as well as the respective Delaunay triangulation) of n points can be constructed in $O(nlogn)$ time and $O(n)$ space. Furthermore the closest or furthest point Voronoi diagram can be constructed from the respective Delaunay triangulation in $O(n)$ time, and vice versa.

For proofs leading to 1 – 5 and other properties of these constructions, see Edelsbrunner [29] and the references therein.

In the following we assume that our point set S is fixed. The relationship between the Delaunay triangulations and α-shapes is given by the following lemma [29].

Lemma A.1 *The α-shape of S is a subgraph of $DT_c(S)$ if $\alpha \leq 0$.*

[1] We will concentrate on negative α-shape and closest point Delaunay triangulation.

Appendix B

Determining the Simplex Enclosing a Point

B.1 Introduction

Suppose there is a 2D (3D) triangulation T and a set of 2D (3D) points. Our task here is to efficiently find a simplex enclosing a given point. The strategy is based on a *bucket sorting*, see Cho *et al.* [11]. That is, if we partition a box containing the triangulation into buckets and identify the bucket enclosing a point and certain triangulation vertices and edges, we can locate the point into a few triangles. The scheme consists of three parts: (1) construction of buckets; (2) putting edges into bucket; and (3) determining the simplex and repeating the above three parts for every point.

B.2 Construction of Buckets

Suppose a 2D (3D) triangulation falls into a rectangular (rectangular parallelepiped) frame with sides parallel to the xy (xyz) axes, respectively. The frame is partitioned into $n_x \times n_y$ ($n_x \times n_y \times n_z$) rectangular (rectangular parallelepiped) buckets of equal size as shown in Figure B-1. A 2D bucket which is the i^{th} from left and the j^{th} from bottom is denoted by B_{ij}. A 2D *bucket coordinate system* is the coordinate system obtained by *rescaling* the x and y axes such that the left-bottom corner of bucket B_{ij} has coordinates (i, j). In a similar manner, a 3D bucket B_{ijk} and the corresponding coordinates (i, j, k) can be defined. We note that a bucket to which an end point of an edge belongs is easily found by just taking the integer parts of the bucket coordinates of the end point. Given the number of edges n, corresponding n_x and n_y of the 2D frame are determined by the formulas :

$$n_x = \lfloor \alpha_x \sqrt{n} \rfloor, \qquad n_y = \lfloor \alpha_y \sqrt{n} \rfloor, \tag{B.1}$$

with properly chosen weighting parameters α_x and α_y, where $\lfloor \bullet \rfloor$ denotes the greatest integer less than or equal to $\bullet$. Parameters α_x and α_y can be appropriately obtained by the

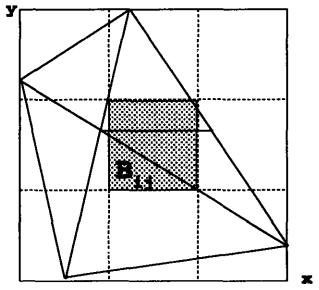


Figure B-1: A rectangle which bounds the edges of a triangulation in the coordinate system.

following two equations :

$$\frac{\alpha_y}{\alpha_x} = \frac{L_y}{L_x}, \qquad \alpha_x \alpha_y = 1, \tag{B.2}$$

where L_x and L_y are the width and height of a rectangle which exactly bounds the set of edges in xy coordinate system, as depicted in Figure B-1.

By solving Equation B.2, α_x and α_y are determined as

$$\alpha_x = \sqrt{\frac{L_x}{L_y}}, \qquad \alpha_y = \sqrt{\frac{L_y}{L_x}}. \tag{B.3}$$

Similarly, n_x, n_y and n_z of the 3D frame are determined by the formulas :

$$n_x = \lfloor \alpha_x n^{\frac{1}{3}} \rfloor, \qquad n_y = \lfloor \alpha_y n^{\frac{1}{3}} \rfloor, \qquad n_z = \lfloor \alpha_z n^{\frac{1}{3}} \rfloor, \tag{B.4}$$

where n is the number of edges and

$$\alpha_x = \left(\frac{L_x^2}{L_y L_z} \right)^{\frac{1}{3}}, \qquad \alpha_y = \left(\frac{L_y^2}{L_z L_x} \right)^{\frac{1}{3}}, \qquad \alpha_z = \left(\frac{L_z^2}{L_x L_y} \right)^{\frac{1}{3}}. \tag{B.5}$$

If n_x, n_y or n_z in Equations B.1 and B.4 happen to be zero, we consider them to be one. The number of buckets constructed is obviously $\mathcal{O}(n)$ and consequently the number of approximating segments in a bucket is $\mathcal{O}(1)$ on the average.

B.3 Putting Vertices into Buckets

After the bucket construction, we associate each vertex with a bucket containing the vertex. For each $\mathbf{p}$ in xy (xyz) coordinate system, we first perform a transformation with respect

to bucket coordinate system ij (ijk), respectively, i. e. :

$$\tilde{\vec{p}} = (\tilde{p}_i, \tilde{p}_j) \tag{B.6}$$

where vectors $\vec{p}$ are vertices of the triangles (tetrahedra). The tilde " ˜ " represents the corresponding transformed version. Those transformed points $\tilde{\vec{p}}$ in 2D case are obtained by the following relations:

$$\tilde{\vec{p}} = (\tilde{p}_i, \tilde{p}_j) \tag{B.7}$$

and

$$\tilde{p}_i = \frac{n_x}{L_x}(p_x - x_{min}), \qquad \tilde{p}_j = \frac{n_y}{L_y}(p_y - y_{min}), \tag{B.8}$$

where p_x, p_y are x, y components of the point $\vec{p}$ and (x_{min}, y_{min}) are coordinates of the left-bottom corner of a bounding rectangle shown in Figure B-1 and n_x, n_y, L_x, L_y are described in Equation B.2 and B.3. In a similar manner, the transformed point $\tilde{\vec{p}}$ in 3D case can be determined as :

$$\tilde{\vec{p}} = (\tilde{p}_i, \tilde{p}_j, \tilde{p}_k) \tag{B.9}$$

where

$$\tilde{p}_k = \frac{n_z}{L_z}(pz - z_{min}). \tag{B.10}$$

and $\tilde{p}_i$, $\tilde{p}_j$ are defined in Equations B.8.

B.4 Putting Edges into Buckets

After putting vertices into buckets, we associate each edge with the buckets constructed in Section B.2. For each edge $\vec{l}_p$ in xy (xyz) coordinate system, we now put each 2D (3D) transformed edge $\tilde{\vec{l}}_p$ into buckets B_{ij} (B_{ijk}), respectively. Buckets containing the end points $\tilde{\vec{b}}_p$ and $\tilde{\vec{e}}_p$ are easily determined by just taking the integer parts of the end points, as depicted in Figure B-2. To determine intermediate buckets through which the transformed edge $\tilde{\vec{l}}_p$ passes, a *line-type* of $\tilde{\vec{l}}_p$, associated with the *signs* of Δi, Δj and Δk, is identified, where

$$\Delta i = \tilde{e}_{pi} - \tilde{b}_{pi}, \quad \Delta j = \tilde{e}_{pj} - \tilde{b}_{pj}, \quad \Delta k = \tilde{e}_{pk} - \tilde{b}_{pk}. \tag{B.11}$$

Once the line-type is identified, we know the tracing direction, and can easily trace the segment $\tilde{\vec{l}}_p$. As an illustrative example, we consider a 2D edge $\tilde{\vec{l}}_p$ shown in Figure B-2. It is obvious that $\tilde{\vec{l}}_p$ has positive Δi and negative Δj. Therefore, the tracing direction is either bottom $(\Delta_i < 0)$ or right $(\Delta_j > 0)$ starting from the bucket B_{01} which contains an end point $\tilde{\vec{b}}_p$, we need to check whether $\tilde{\vec{l}}_p$ crosses the bottom edge or the right edge of each bucket it passes through. This procedure is repeated until an intermediate bucket, adjacent to B_{20} which contains the other end point $\tilde{\vec{e}}_p$, is determined. As illustrated in Figure B-3, $\tilde{\vec{l}}_p$ crosses the right edge of the bucket B_{01}. An intermediate bucket B_{11} is thus

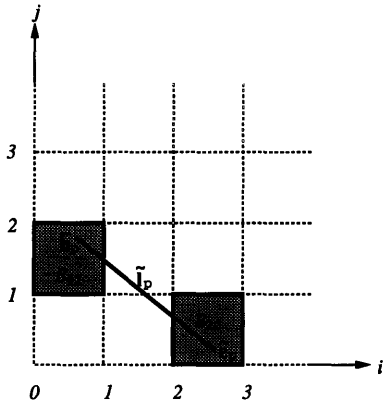


Figure B-2: Buckets containing the end points $\vec{b}_p$, $\vec{e}_p$ of a transformed edge $\vec{l}_p$ (adapted from Cho *et al.* [11]).

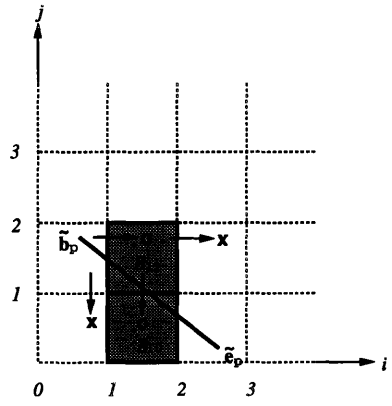


Figure B-3: Intermediate buckets through which the edge $\vec{l}_p$ passes (adapted from Cho *et al.* [11]).

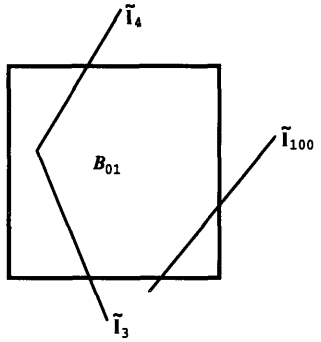


Figure B-4: A 2D bucket associated with three transformed edges (adapted from Cho [11]).

simply determined. Similarly, we next check if $\vec{\tilde{l}}_p$ crosses the bottom edge or the right edge of the bucket B_{11}, and consequently we obtain another intermediate bucket B_{10}. We can easily extend the idea of this procedure to a 3D transformed edge $\vec{\tilde{l}}_p$ and the corresponding intermediate buckets B_{ijk}. The only difference is that we have more possible line-types of $\vec{\tilde{l}}_p$ associated with Δk in Equation B.11 and we need to consider face-crossing of $\vec{\tilde{l}}_p$ instead of edge-crossing in 2D case. We note that in general, very few numbers of intermediate buckets exist due to the characteristics of bucketing technique – the number of uniformly distributed equal-sized buckets $\sim \mathcal{O}(n)$, where n is the number of edges.

Suppose the *preprocessing* step described above is completed for each 2D (3D) transformed edge $\vec{\tilde{l}}_p$, each bucket B_{ij} (B_{ijk}) is associated with the corresponding edges. In practical algorithms, a bucket B_{ij} (B_{ijk}) is no other than a *pointer* to the list of associated edges. For example, provided that a 2D bucket B_{01} is associated with three 2D transformed edges $\vec{\tilde{l}}_3$, $\vec{\tilde{l}}_4$ and $\vec{\tilde{l}}_{100}$ as shown in Figure B-4, then data for the bucket B_{01} can be represented as

$$B_{01} \longrightarrow \{\vec{\tilde{l}}_3, \longrightarrow \vec{\tilde{l}}_4, \longrightarrow \vec{\tilde{l}}_{100}\}. \tag{B.12}$$

These buckets can be constructed in $\mathcal{O}(n)$ time and space on the average.

B.5 Determining the Simplex Enclosing a Point

After we find the buckets containing vertices, edges, or both, the rest of the buckets are either completely inside a simplex or completely outside of any simplex.

By checking the bounding boxes of simplices, we can localize an undetermined bucket in the region covered by a few simplices and further check which simplex enclosing the bucket. Finally, the rest of buckets are those which do not have any intersection with the

triangulation.

At this stage, we can immediately find the bucket B_{ij} (B_{ijk}) enclosing the point $\mathbf{x}$. There are four possible situations here:

1. If B_{ij} (B_{ijk}) is completely in the interior of a simplex, then we immediately conclude that the simplex encloses the point.

2. If B_{ij} (B_{ijk}) does not contain any piece of any simplex, then we conclude that the point is outside of the triangulation.

3. If B_{ij} (B_{ijk}) encloses a vertex of the triangulation, we check every simplex incident at the vertex and point $\mathbf{x}$ by calculating the barycentric coordinates of $\mathbf{x}$ with respect to the simplex. If all the barycentric coordinates with respect to a simplex are greater than or equal to zero, we conclude that the simplex is enclosing point $\mathbf{x}$.

4. If B_{ij} (B_{ijk}) contains only partial edge(s) of the triangulation, we check every simplex incident to the edge(s). We then find the simplex enclosing the point in the same manner described above.

In this fashion, the simplex enclosing a point can be found in $\mathcal{O}(n)$ time on average where n is the number of edges. Once the above framework is formed, it can be used repeatedly for multiple points for enhanced efficiency of evaluation and visualization of simplex splines.

Appendix C

Geometrical Interpretation of Simplex Splines

In [19], the following relation defines the s-variate simplex spline $M(\mathbf{u}|\mathbf{u}^0,...,\mathbf{u}^m)$, which is a degree $m-s$ polynomial of $\mathbf{u} \in \mathbf{R}^s$ and depends on the $m+1$ parameters (knots) $\mathbf{u}^j \in \mathbf{R}^s, j = 0,1,...,m(m \geq s)$ and $s \in Z^+$:

$$\int_{\mathbf{R}^s} f(\mathbf{u}) M(\mathbf{u}|\mathbf{u}^0,...,\mathbf{u}^m) d\mathbf{u} = m! \int_{S^m} f(t_0\mathbf{u}^0 + \cdots + t_m\mathbf{u}^m) dt_1 \cdots dt_m \tag{C.1}$$

This definition is true for any integrable function $f : G \subset \mathbf{R}^s \rightarrow \mathbf{R}$, for some G containing the convex hull of the set of knots. The region of integration S^m is the standard m-simplex i.e., the region in $\mathbf{R}^m$ defined by

$$S^m = \{\mathbf{t}|\mathbf{t} = (t_1,...,t_m) \in \mathbf{R}^m; t_j \geq 0, j = 1,...,m; \sum_{j=1}^{m} t_j \leq 1\} \tag{C.2}$$

Given the values of $t_1,...,t_m$, the coefficient t_0 is determined by $\sum_{j=0}^{m} t_j = 1$. To interpret Equation C.1, we transform the integral over S^m to an integral over a general m-simplex σ which is based upon certain vertices $\mathbf{v}^0,...,\mathbf{v}^m, \mathbf{v}^j \in \mathbf{R}^m$ such that the points $\mathbf{u}^j$ are the orthogonal projections onto $\mathbf{R}^s$ of the $\mathbf{v}^j$. Then, after a series of mathematical operations shown in [68, 95], the following result can be obtained:

$$M(\mathbf{u}|\mathbf{u}^0,...,\mathbf{u}^m) = \frac{vol_{m-s}(A)}{vol_m(\sigma)}. \tag{C.3}$$

where $A = \sigma \cap \{\mathbf{v}|\mathbf{v} \in \mathbf{R}^m; (\mathbf{v}_1,...,\mathbf{v}_s) = \mathbf{u}\}$ and $vol_m(\sigma)$ is the m-dimensional volume of σ.

The geometrical interpretation is illustrated for cases $s = 1$, $m = 2$, and $s = 1$, $m = 3$, see Figure C-1.

The region in the parameter space in which the simplex spline value is non-zero is $[\mathbf{U}] = \{\mathbf{u}^0,...,\mathbf{u}^m\} \subset \mathbf{R}^s$, which is the convex hull of the collection of knots associated with

the spline. A hyperplane in $\mathbf{R}^s$ passing through any s of the knots separates polynomial pieces of degree $n = m - s$ within the non-zero region of the spline (polynomial cut region). If the knots are in *general position* (no $s+1$ knots are in one hyperplane in $\mathbf{R}^s$ or, any $s+1$ knots are affinely independent), then these polynomial pieces are C^{n-1} continuous. Some illustrations of simplex spline supports and of cut regions are given in Figures C-2 and C-3 for $s = 2$ and $m = 4$ (bivariate quadratic splines).

From this geometrical interpretation, we have the following for s-variate simplex spline:

1. **Piecewise polynomial** of degree $n = m - s$ over $[U]$;
2. **Optimal smoothness**: C^{n-1} continuous if any $s+1$ knots are not co-hyperplanar;
3. **Non-negativity**: $M \geq 0$;
4. **Normalization**: From Equation C.1 it follows that

$$\int_{\mathbf{R}^s} M(\mathbf{u}|U)d\mathbf{u} = 1 \tag{C.4}$$

   where $U = \mathbf{u}^0, ..., \mathbf{u}^m$.

5. **Locality**: M is zero for all $\mathbf{u} \notin [U]$.

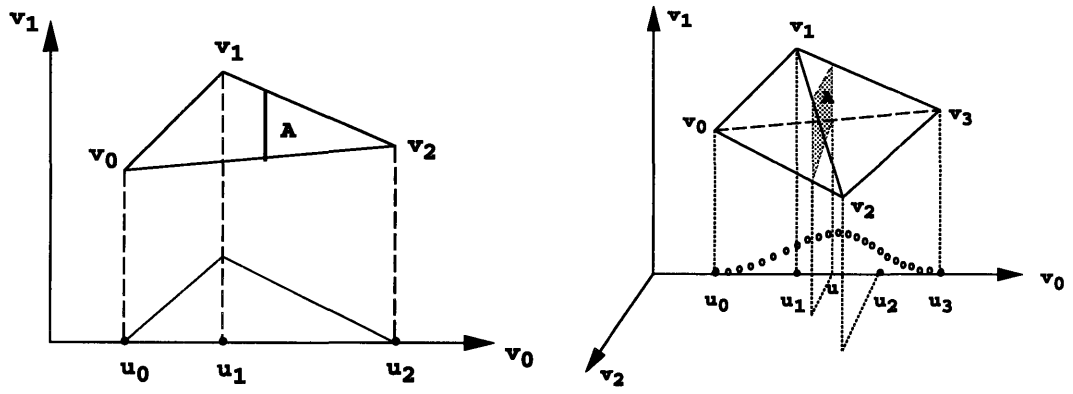


Figure C-1: Geometrical interpretation for cases $s = 1$, $m = 2$ and $s = 1$, $m = 3$. The vector $\mathbf{u}, \mathbf{u}^j$ are scalars in this illustration, having the *components* $\mathbf{u}_1(= \mathbf{u})$ and $\mathbf{u}_1^j(= \mathbf{u}^j)$, respectively.

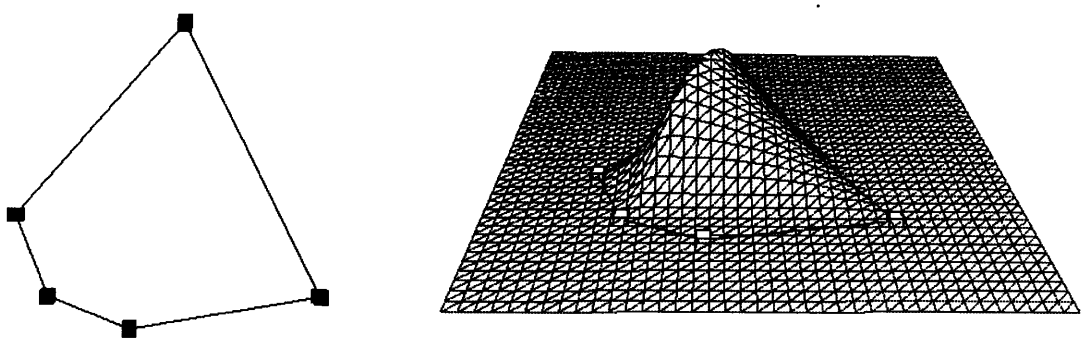


Figure C-2: Bivariate quadratic simplex spline basis function over 5 knots.

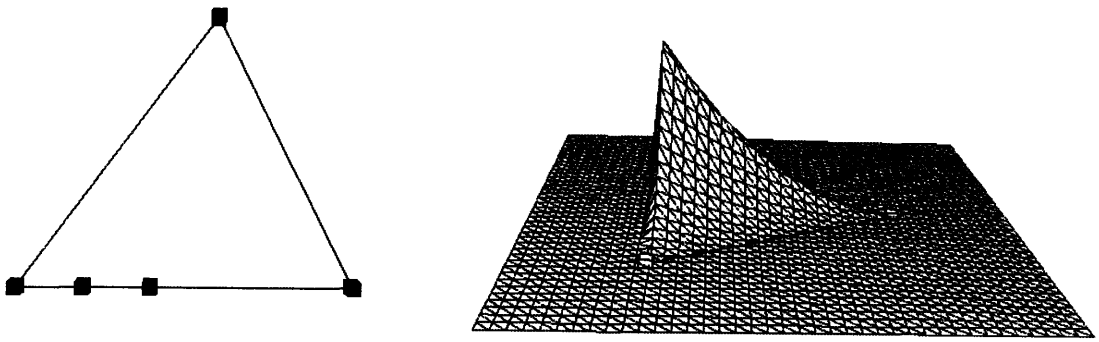


Figure C-3: Bivariate quadratic simplex spline basis function. There are four knots collinear in the five planar knots.

Bibliography

[1] G. Alefeld and L. Platzöder. A quadratically convergent Krawczyk-like algorithm. *SIAM Journal of Mathematical Analysis*, 20(1):210–219, February 1983.

[2] S. Auerbach, R. H. J. Gmelig Meyling, M. Neamtu, and H. Schaeben. Approximation and geometric modeling with simplex B-spline associated with irregular triangles. *Computer Aided Geometric Design*, (8):67–87, 1991.

[3] L. Bardis and N. M. Patrikalakis. Topological structures for generalized boundary representations. Technical Report MITSG 94-22, Cambridge, MA: MIT Sea Grant College Program, September 1994.

[4] R. E. Barnhill. Representation and approximation of surfaces. In J. R. Rice, editor, *Mathematical Software III*, pages 69–120, Madison, Wisconsin, March 1977. Mathematical Software Symposium, Academic Press.

[5] P. J. Besl. *Surfaces in Range Image Understanding.* Springer-Verlag, New York, 1988.

[6] E. Brisson. Representing geometric structures in d dimensions: Topology and order. *Discrete and Computational Geometry*, 9:387–426, 1993.

[7] K. W. Brodlie, L. A. Carpenter, R. A. Earnshaw, J. R. Gallop, R. J. Hubbold, A. M. Mumford, C. D. Osland, and P. Quarendon. *Scientific Visualization.* Springer-Verlag, Heidelberg, Berlin, 1992.

[8] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10:350–355, 1978.

[9] G. Celniker. *ShapeWright: Finite Element Based Free-Form Shape Design.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1990.

[10] G. Celniker and D. C. Gossard. Deformable curve and surface finite elements for free-form shape design. *Computer Graphics*, 25(4):257–266, 1991.

[11] W. Cho, T. Maekawa, and N. M. Patrikalakis. Topologically reliable approximation of composite Bézier curves. Design Laboratory Memorandum 94-11, MIT, Department of Ocean Engineering, Cambridge, Massachusetts, 1994.

[12] C. K. Chui. *Multivariate Splines.* SIAM, Albany, New York, 1988.

[13] C. K. Chui, L. L. Schumaker, and J. D. Ward. *Approximation Theory IV.* Academic Press: New York, 1983.

[14] A. K. Cline and R. L. Renka. A storage–efficient method for construction of a Thiessen triangulation. *Rocky Mountain Journal of Mathematics*, 14(1):119–139, Winter 1984.

[15] R. W. Clough and J. L. Tocher. Finite element stiffness matrices for the analysis of plate bending. In J. S. Przemieniecki, editor, *Proceedings of the 1st Conference on Matrix Methods in Structural Mechanics*, pages 515–545, Wright-Patterson Air Force Base, 1965. Washington, U.S. Dept. of Commerce.

[16] G. P. Crain and B. K. Bhattacharya. Treatment of non-equispaced two-dimensional data with a digital computer. *Geoexploration*, 5:173–194, 1967.

[17] G. P. Cressman. An operational objective analysis system. *Monthly Weather Review*, pages 367–374, 1959.

[18] W. Dahmen. Multivariate B-spline-recurrence relations and linear combinations of truncated powers. In W. Schempp and K. Zeller, editors, *Multivariate Approximation Theory*, pages 64–82, Birkhäuser, Basel, 1979.

[19] W. Dahmen and C. A. Micchelli. On the linear independence of multivariate B-splines I. Triangulations of simploids. *SIAM Journal of Numerical Analysis*, 19:993–1012, 1982.

[20] W. Dahmen, C. A. Micchelli, and H. P. Seidel. Blossoming begets B-spline bases built better by B-patches. *Mathematics of Computation*, 59(199):97–115, July 1993.

[21] L. De Floriani, B. Falcidieno, and C. Pienovi. Delaunay-based representation of surfaces defined over arbitrarily shaped domains. *Computer Vision, Graphics and Image Processing*, 32:127–140, 1985.

[22] H. Delingette, M. Hebert, and K. Ikeuchi. Energy functions for regularization algorithms. In B. C. Vemuri, editor, *Geometric Methods in Computer Vision*, pages 104–115, San Diego, CA, July 1991. SPIE. Volume 1570.

[23] P. Dierckx. An algorithm for surface-fitting with spline functions. *IMA Journal of Numerical Analysis*, 1:267–283, 1981.

[24] D. Doo and M. A. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10:356–360, 1978.

[25] R. A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *Computer Graphics*, 22(4):65–74, 1988.

[26] O. Dubrule. Comparing splines with kriging. *Computers and Geosciences*, 10:327–338, 1984.

[27] R. Earnshaw and N. Wiseman, editors. *An Introductory Guide to Scientific Visualization.* Springer-Verlag, London, 1992.

[28] H. Edelsbrunner. Weighted alpha shapes. Technical Report UIUCDCS-R-92-1760, Department of Computer Science, Univeristy of Illinois at Urbana–Champaign, Illinois, 1992.

[29] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, IT-29(4):551–559, July 1983.

[30] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. Technical Report UIUCDCS-R-92-1734, Department of Computer Science, University of Illinois at Urbana-Champaign, Illinois, 1992.

[31] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[32] B. Falcidieno and C. Pienovi. Natural surface approximation by constrained stochastic interpolation. *Computer Aided Design*, 22(3):167–172, April 1990.

[33] B. Falcidieno and M. Spagnuolo. Polyhedral surface decomposition based on curvature analysis. In T. L. Kunii and Y. Shinagawa, editors, *Modern Geometric Computing for Visualization, Proceedings of the Workshop on Modern Geometric Computing for Visualization*, pages 57–72, Kogakuin University, Japan, June 29-30 1992. Computer Graphics Society. Tokyo: Springer-Verlag, 1992.

[34] B. Falcidieno and M. Spagnuolo. Geometric reasoning for the extraction of surface shape properties. In N. M. Thalmann and D. Thalmann, editors, *Communicating with Virtual Worlds, Proceedings of CG International '93*, pages 166–178. Springer, Tokyo, June 1993.

[35] L. Fang. *Physically-Based Methods for Parametric Curve and Surface Reconstruction.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1994.

[36] L. Fang and D. C. Gossard. Reconstruction of smooth parametric surfaces from unorganized data points. In J. D. Warren, editor, *Proceedings of Curves and Surfaces in Computer Vision and Graphics III, Boston, MA, USA*, volume 1830, pages 226–236. SPIE–The International Society for Optical Engineering, Bellingham, Washington, USA, November 1992.

[37] L. Fang and D. C. Gossard. Multidimensional curve fitting to unorganized data points by nonlinear minimization. *Computer Aided Design*, 27(1):48–58, 1995.

[38] G. Farin. Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design*, 3(2):83–128, 1986.

[39] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide.* Academic Press, San Diego, CA, 3rd edition, 1993.

[40] T. A. Foley, H. Hagen, and G. M. Nielson. Visualizing and modeling unstructured data. *The Visual Computer*, 9:439–449, 1993.

[41] P. Fong. Shape control for multivariate B-spline surfaces over arbitrary triangulations. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada, 1992.

[42] P. Fong and H. P. Seidel. Control points for multivariate B-spline surfaces over arbitrary triangulations. *Computer Graphics Forum*, 10:309–317, 1991.

[43] P. Fong and H. P. Seidel. An implementation of triangular B-spline surfaces over arbitrary triangulations. *Computer Aided Geometric Design*, 10:267–275, 1993.

[44] R. Franke. Locally determined smooth interpolation at irregularly spaced points in several variables. *Journal of the Institute of Mathematics and its Applications*, 19:471–482, 1977.

[45] R. Franke. Scattered data interpolation: tests of some methods. *Mathematical Computation*, 38(157):181–200, 1982.

[46] P. E. Gill, W. Murray, and A. Wright. *Practical Optimization.* Academic Press, New York, 1981.

[47] C. M. Gold. Spatial data structures - The extension from one to two dimensions. In L. F. Pau, editor, *Mapping and Spatial Modeling for Navigation*, pages 11–39, Fano, Denmark, August 1989. NATO ASI, Springer-Verlag. Series F: Computer and Systems Science, Volume 65.

[48] T. A. Grandine. The computational costs of simplex spline functions. *SIAM Journal of Numerical Analysis*, 24:887–890, 1987.

[49] P. J. Green and R. Sibson. Computing Dirichlet tesselation in the plane. *Computer, Journal of the Computer Society of the IEEE*, 21(2):168–173, 1977.

[50] G. Greiner and H.-P. Seidel. Modeling with triangular B-splines. In J. Rossignac, J. Turner, and G. Allen, editors, *Proceedings of the Second Symposium on Solid Modeling and Applications*, pages 211–220, Montreal, Canada, May 1993. ACM SIGGRAPH in cooperation with the IEEE Computer Society.

[51] O. Guenther and A. Buchman. Research issues in spatial databases. *SIGMOD Record*, 19(4):61–68, December 1990.

[52] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, April 1985.

[53] H. A. Hakopian. Multivariate divided differences and multivariate interpolation of Lagrange and Hermite type. *Journal of Approximation Theory*, 34:286–305, 1982.

[54] R. W. Hamming. *Numerical Methods for Scientists and Engineers.* McGraw-Hill, New York, 1962.

[55] R. L. Hardy. Multiquadric equations of topography and other irregular surfaces. *Journal Geophysical Research*, 76:1905–1915, 1971.

[56] K. Höllig. Multivariate splines. *SIAM Journal of Numerical Analysis*, 19:1013–1031, 1982.

[57] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design.* A.K. Peters, Wellesley, MA, 1993. Translated by L.L. Schumaker.

[58] T. Ikeda, T. L. Kunii, Y. Shinagawa, and M. Ueda. A geographical database system based on the homotopy model. In T. L. Kunii and Y. Shinagawa, editors, *Modern Geometric Computing for Visualization*, pages 193–206. Tokyo: Springer-Verlag, 1992.

[59] T. L. Kunii and Y. Shinagawa. Visualization: New concepts and techniques to integrate diverse application areas. In N. M. Patrikalakis, editor, *Scientific Visualization of Physical Phenomena*, pages 3–25. Tokyo: Springer-Verlag, 1991.

[60] C. L. Lawson. Software for C^1 surface interpolation. In J. R. Rice, editor, *Mathematical Software III*, pages 161–194. Academic Press, New York, 1977.

[61] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988.

[62] C. Loop and T. D. DeRose. A multisided generalization of Bézier surfaces. *ACM Transactions on Graphics*, 8(3):204–234, 1989.

[63] C. Loop and T. D. DeRose. Generalized B-spline surfaces of arbitrary topology. *IEEE Computer Graphics and Applications*, 24(4):347–356, August 1990.

[64] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[65] T. Maekawa. *Robust Computational Methods for Shape Interrogation.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, June 1993.

[66] T. Maekawa and N. M. Patrikalakis. Interrogation of differential geometry properties for design and manufacture. *The Visual Computer*, 10(4):216–237, March 1994.

[67] E. A. Melissaratos. *Mesh Generation and Geometric Optimization.* PhD thesis, Rutgers, The State University of New Jersey, May 1995.

[68] C. A. Micchelli. On a numerically efficient method for computing multivariate B-splines. In W. Schempp and K. Zelker, editors, *Multivariate Approximation Theory*, pages 211–248, Basel, 1979. Birkhäuser.

[69] R. E. Moore. *Interval Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1966.

[70] G. M. Nielson. A method for interpolating scattered data based upon a minimum norm network. *Mathematics of Computation*, 40:199–205, 1983.

[71] G. M. Nielson, P. Brunet, M. Gross, H. Hagen, and S. K. Klimenko. Research issues in data modeling for scientific visualization. *IEEE Computer Graphics and Applications*, 14(2):70–73, March 1994.

[72] P. Ning and J. Bloomenthal. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 13(6):33–41, November 1993.

[73] Numerical Algorithms Group, Oxford, England. *NAG Fortran Library Manual, Mark 15, FLSG415D, Silicon Graphics 4D Double Precision User's Notes*, 1990.

[74] Numerical Algorithms Group, Oxford, England. *NAG Fortran Library Manual, Volumes 1-8*, Mark 14 edition, 1990.

[75] M. A. Oliver and R. Webster. Kriging: A method of interpolation for geographical information systems. *International Journal of Geographical Information Systems*, 4(3):313–332, 1990.

[76] N. M. Patrikalakis and G. A. Kriezis. Representation of piecewise continuous algebraic surfaces in terms of B-splines. *The Visual Computer*, 5(6):360–374, 1989.

[77] J. Peters. *Fitting smooth parametric surfaces to 3D data.* PhD thesis, University of Wisconsin, Madison, Wisconsin, 1990.

[78] R. Pfeifle and H. S. Seidel. Fitting triangular B-splines to functional scattered data. In *Proceedings of Graphics/Vision Interface '95, Quebec City, Canada, May 1995*, 1995.

[79] H. Qin and D. Terzopoulos. Dynamic manipulation of triangular B-splines. In C. Hoffmann and J. Rossignac, editors, *Proceedings of the Third ACM Solid Modeling Symposium*, pages 351–360, Salt Lake City, Utah, May 1995. ACM, NY.

[80] D. F. Rogers and R. A. Earnshaw, editors. *State of the Art in Computer Graphics.* Springer-Verlag, New York, 1991.

[81] H. Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS.* Addison-Wesley, Reading, MA, 1990.

[82] M. F. W. Schmidt. Cutting cubes - visualizing implicit surfaces by adaptive polygonalization. *The Visual Computer*, 10(2):101–115, 1993.

[83] L. L. Schumaker. Fitting surfaces to scattered data. In *Approximation Theory II Conference*, pages 203–269, Austin, TX, 1976. University of Texas.

[84] H.-P. Seidel. Representing piecesise polynomials as linear combinations of multivariate B-splines. In T. Lyche and Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design II.* Academic Press, Boston, 1992.

[85] H. P. Seidel and A. H. Vermeulen. Simplex splines support surprisingly strong symmetric structure and subdivision. In P. J. Laurent, A. Le Méhauté, and L. L. Schumaker, editors, *Curves and Surfaces II*, pages 1–13. Boston, AK Peters, 1991.

[86] D. Shepard. A two dimensional interpolation function for irregular spaced data. In *Proceedings of 23rd ACM National Conference*, pages 517–524, 1968.

[87] E. C. Sherbrooke and N. M. Patrikalakis. Computation of the solutions of nonlinear polynomial systems. *Computer Aided Geometric Design*, 10(5):379–405, October 1993.

[88] S. S. Sinha and B. G. Schunck. Surface approximation using weighted splines. In S. Negahdaripour *et al*, editor, *Proceedings of the 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 44–49, Los Alamitos, CA, June 1991. IEEE Computer Society Press.

[89] S. W. Sloan. A fast algorithm for constructing Delaunay triangulation in plane. *Advances in Engineering Software*, 9(1):34–55, January 1987.

[90] W. K. Stewart. A model-based approach to 3-D imaging and mapping underwater. *ASME Transactions, Journal of Offshore Mechanics and Arctic Engineering*, 111:352–356, 1990.

[91] K. G. Suffern. Quadtree algorithm for contouring functions of two variables. *The Computer Journal*, 33(5):402–407, 1990.

[92] K. G. Suffern. Interval methods in computer graphics. *Computers and Graphics*, 15(3):331–340, 1991.

[93] D. Terzopoulos and H. Qin. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136, April 1994.

[94] D. Terzopoulos and M. Vasilescu. Sampling and reconstruction with adaptive meshes. In S. Negahdaripour *et al*, editor, *Proceedings of the 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 70–75, Los Alamitos, CA, June 1991. IEEE Computer Society Press.

[95] C. R. Traas. Practice of bivariate quadratic simplicial splines. In W. Dahmen, editor, *Computation of Curves and Surfaces*, pages 383–422, Netherlands, 1990. Kluwer Academic.

[96] S. T. Tuohy. *Geophysical Map Representation, Abstraction and Interrogation for Autonomous Underwater Vehicle Navigation.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, October 1993.

[97] S. T. Tuohy and L. Bardis. Low degree approximation of high degree B-spline surfaces. *Engineering with Computers*, 9(4):198–209, Winter 1993.

[98] S. T. Tuohy, T. Maekawa, and N. M. Patrikalakis. Interrogation of geophysical maps with uncertainty for AUV micro-navigation. In *Engineering in Harmony with the Ocean, Proceedings of Oceans '93, Victoria, Canada.* IEEE Oceanic Engineering Society, October 1993.

[99] S. T. Tuohy and N. M. Patrikalakis. Representation of geophysical maps with uncertainty. In N. M. Thalmann and D. Thalmann, editors, *Communicating with Virtual Worlds, Proceedings of CG International '93*, pages 179–192. Springer, Tokyo, June 1993.

[100] S. T. Tuohy, J. W. Yoon, and N. M. Patrikalakis. Reliable interrogation of 3-D nonlinear geophysical databases. In J. A. Vince and R. A. Earnshaw, editors, *Computer Graphics: Developments in Virtual Environments, Proceedings of CG International '95*, Leeds, UK, June 1995. London, Academic Press. To appear.

[101] S. T. Tuohy, J. W. Yoon, and N. M. Patrikalakis. Volumetric splines for visualization of ocean data. In *Proceedings of Oceans '95.* IEEE Oceanic Engineering Society, 1995. To appear.

[102] M. Ueda, T. Ikeda, T. L. Kunii, and Y. Shinagawa. A case study for building a database and 3-d visualization in geomorphology. In T. L. Kunii and Y. Shinagawa, editors, *Modern Geometric Computing for Visualization*, pages 207–226. Tokyo: Springer-Verlag, 1992.

[103] J. Wilhelms and A. Van Gelder. Octtrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227, July 1992.

[104] F. E. Wolter and S. T. Tuohy. Approximation of high degree and procedural curves. *Engineering with Computers*, 8(2):61–80, 1992.

[105] X. Ye. *Construction and Verification of Smooth Free-Form Surfaces Generated by Compatible Interpolation of Arbitrary Meshes.* Verlag Dr. Köster, Berlin, Germany, July 1994.

[106] J. W. Yoon. A virtual environment for the visualization of geophysical ocean data sets. Master's of Engineering Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, August 1994.