GEOMETRIC MODELING

(Second Edition)

Michael E. Mortenson

WILEY COMPUTER PUBLISHING

John Wiley and Sons, Inc.
New York ▪ Chichester ▪ Weinheim ▪ Brisbane ▪ Singapore ▪ Toronto

PREFACE

Geometric modeling is now taught, studied, and practiced world-wide. It has come to affect the lives of us all. It makes possible the fast and accurate image construction of medical diagnostic devices. It is the basis of computer-aided design and manufacture of aircraft, automobiles, and consumer products—making possible safer, more efficient, and more pleasing designs, lowering costs, and providing more choices through customized production. Geometric modeling gives scientists graphic displays of phenomena beyond anyone's power to visualize just a few years ago. It even speeds the design and construction of America's Cup contenders and challenges our movie heroes and heroines with ever more fantastic virtual dragons to slay and worlds to conquer. The end is not in sight.

Many years have passed since publication of the first edition of *Geometric Modeling*. They have been productive and exciting years for those of us working in the field. During this time two compelling circumstances have motivated and affected the creation of this second edition. The first is the progress occurring within geometric modeling itself, as a result of vigorous ongoing research and of increasingly more sophisticated demands of client applications. The second is the comments from almost a generation of readers of the first edition—students, teachers, and professionals alike.

Although clearly a descendant of the first, this second edition is a substantial revision reflecting both new developments and a maturing discipline. With the increased capability, availability, and affordability of computers, geometric modeling applications have spread from mainframes and supercomputers to workstations and PCs. The usefulness of geometric modeling has

broadened from its initial applications in the aerospace and automotive industries and now includes such diverse areas as special effects in cinematography and television commercials and the design and control of type fonts. Considering all this—and with entirely new fields such as virtual reality, computer vision, and scientific visualization bringing us into the third millennium—the need is growing for practitioners in all the related fields and client applications to be knowledgeable and current in geometric modeling.

The second edition of *Geometric Modeling* offers the reader a comprehensive look at the indispensable core concepts of geometric modeling, describing and comparing all the important mathematical structures for modeling curves, surfaces, and solids, and showing how to shape and assemble those elements into more complex models. Like the first edition, it is intended for students, teachers, and professionals in the field of geometric modeling and its primary applications, including computer graphics, computer-aided design and manufacturing, robotics, and scientific visualization, among many others. While the first edition was intended to be used primarily as a textbook, the second edition will serve well as a professional tutorial and reference, teaching resource, and primary or supplementary textbook. Its language is as free as practical from the jargon of special applications so that it may focus on the geometry and be universally accessible across a wide range of disciplines. For these same reasons, application-oriented algorithms and programming techniques are omitted.

There are three important functions of geometric modeling: to represent elementary forms (for example, curves, surfaces, and solids), to shape and assemble those forms into more complex objects, and to process the necessary concomitant geometry (for example, computation of intersections, offsets, and fillets). The content and organization of this text reflect these functions. Chapter 1 attempts to answer the question, *What is geometric modeling?* It briefly discusses its history and the underlying mathematics. Chapters 2 through 5 are all about curves: mostly Hermite, Bézier, and B-Spline; their mathematical representation, modification, and analysis. Surfaces, including the quadric, bicubic Hermite, Bézier, and B-Spline, are the subjects of Chapters 6 through 9. Chapter 10 addresses the still esoteric world of trivariate parametric solids, as well as simple shape generation through transformations and instanciation of parameterized primitive shapes, sweep operations, and controlled nonlinear deformations. Much of the power of contemporary geometric modeling resides in its capacity to synthesize, allowing us to describe complex shapes as arrangements of much simpler ones; thus, the subject of Chapter 11 is complex model construction, beginning with the topology of models and proceeding with the theory and application of graph-based, Boolean, boundary, and space-partitioning models. Chapter 12

presents a discussion on the newly emerging methods of relational geometric synthesis. Appendices A and B review vectors and matrices, respectively. Appendix C summarizes various transformations of models. An extensive bibliography, frequently referenced in the text, completes this work.

Several subjects are not included. Although well-understood, they are either not yet widely accepted, too specialized, or mathematically beyond the scope of this text. For example, nontensor product patches, general offsets and blends, nonstandard parameterization schemes, and Euler models are not discussed here.

As a textbook, *Geometric Modeling* can be used for a one- or two-semester course at the upper division or graduate level in any of the disciplines mentioned, including engineering, physics, computer science, and applied mathematics. Some prior course work is required in elementary calculus, analytic geometry, and vector and matrix methods. Programming skills, while useful, are not necessary for understanding this text.

The importance of geometric modeling has long been recognized in its many fields of application. It is, of course, a primary ingredient of computer graphics and computer-aided design and manufacturing, as well as of such more recent applications as computer art, cinematographic special effects, robotics, computer vision, scientific visualization, and virtual reality. Progress in any of these areas now depends in large measure on how well we can create effective geometric models.

Finally, I must repeat a warning issued in the first edition: The geometric model of an object or the simulation of a process via geometric metaphor is not the object or the process. When we ask the model questions, when we analyze it to better understand or predict the behavior of what is being represented, we must remember that the answers and results pertain first to the model itself and only conditionally to its object. If and only if the model closely corresponds to the object or process can we safely infer the object's or process' behavior from the model's behavior. The accuracy of modeling methods is an art and science in itself.

ACKNOWLEDGMENTS

The history of geometric modeling is one of unique cooperation between industry and academia, and material presented here draws from both. Many individuals worked to develop and refine the theories and techniques of geometric modeling, and many of the most important contributions are referenced in this text. So I must repeat my acknowledgment in the first edition of all those both well-known and anonymous researchers and developers who have contributed to geometric modeling over the past 40 years. The authors whose works are listed in the bibliography have created a new discipline. Their work has made possible a growing family of scientific, industrial, and commercial applications whose very existence is not possible without a strong underlying geometric modeling capability.

I thank the many readers of the first edition who took the time to offer valuable comments and suggestions. I especially thank the directors and staff of the Port Townsend and Jefferson County public libraries, particularly Carol Cahill, Reference Librarian at the Port Townsend library, and Shery Hart, at the Jefferson County library. Although these two small rural libraries are not endowed with the resources of a large university library, the diligence and professional skills of their staffs are the equal of that found anywhere. I also thank Bob Aronds, Diane Cerra (now with Morgan Kaufmann), and Marjorie Spencer, my editors at Wiley, for all they have so thoughtfully and professionally done to make the production of this edition an enjoyable experience. Harry Meriwether and Art Eshleman I must especially thank for their example of how to think geometrically and for an understanding of the modeling process itself, both of which I had the pleasure to learn from them many years ago.

I thank Dr. John S. Letcher, Jr., president and founder of AeroHydro Inc., Southwest Harbor, Maine, for permission to use the panelization drawing of the keel and bulb structure of *Black Magic* as the cover artwork (see About the Cover). This geometric model was created using the methods of relational geometric synthesis (see Chapter 12) and AeroHydro's MultiSurf program. I also thank Dr. John S. Letcher, Jr., D. Michael Shook, Seth H. Porter, and George A. Dickson for permission to include their original article on relational geometric synthesis as Chapter 12 of this text. Their work is an important contribution to the field of geometric modeling.

Finally, thanks to my wife Janet, who will never forget the struggles we had preparing the manuscript for the first edition, using an Apple II and a now long-expired word processor that had no equation-editing capability. Even so, she contributed as much time and effort to this edition.

Michael E. Mortenson
Port Townsend
October 1996

CONTENTS

About the Cover vii
Preface ix
Acknowledgments xiii

1 Introduction 1

1.1 What Is Geometric Modeling? 1
1.2 History 6
1.3 The Mathematics 11
1.4 Conventions and Notation 16

2 Curves 19

2.1 Intrinsic Equations of Curves 19
2.2 Explicit and Implicit Equations of Curves 22
2.3 Parametric Equations of Curves 23
2.4 Conic Curves 31
2.5 Hermite, Bézier, and B-Spline Curves: An Overview 32
2.6 Points on a Curve 34

3 Hermite Curves **39**

3.1 Algebraic and Geometric Forms 39
3.2 Hermite Basis Functions 42
3.3 Matrix Form 47
3.4 Tangent Vectors 50
3.5 Truncating and Subdividing 54
3.6 Three-Point Interpolation 60
3.7 Four-Point Interpolation 61
3.8 Conic Hermite Curves 64
3.9 Composite Hermite Curves 74

4 Bézier Curves **81**

4.1 Bézier Basis Functions 83
4.2 Control Points 89
4.3 Truncating and Subdividing 97
4.4 Composite Bézier Curves 105
4.5 Rational Bézier Curves 108

5 B-Spline Curves **113**

5.1 Nonuniform B-Spline Basis Functions 113
5.2 Uniform B-Spline Basis Functions 126
5.3 Quadratic and Cubic B-Spline Basis Functions 128
5.4 Closed B-Spline Curves 130
5.5 Continuity 134
5.6 Conversion Between Basis Functions 137
5.7 Nonuniform Rational B-Spline Curves 139
5.8 Representing Conics with NURBS Curves 140
5.9 Cubic Beta Splines 142

6 Surfaces **143**

6.1 Explicit and Implicit Equations of Surfaces 144
6.2 Quadric Surfaces 145
6.3 Parametric Equations of Surfaces 149
6.4 Hermite, Bézier, and B-Spline Surfaces: An Overview 156

6.5 Points on a Surface 158
6.6 Curve Nets 159
6.7 Embedded Curves 161

7 The Bicubic Hermite Surface 169

7.1 Algebraic and Geometric Forms 169
7.2 Hermite Patch Basis Functions 179
7.3 Tangent and Twist Vectors 180
7.4 Normals 182
7.5 Sixteen-Point Form 184
7.6 Reparameterization of a Patch 186
7.7 Truncating and Subdividing a Patch 191
7.8 Composite Hermite Surfaces 192
7.9 Special Hermite Patches 203
7.10 Blend Surfaces 213

8 Bézier Surfaces 217

8.1 The Tensor Product Bézier Patch 217
8.2 The Bicubic Bézier Patch 218
8.3 A 3×5 Rectangular Array of Control Points 220
8.4 Converting Between Bicubic Bézier and Hermite Forms 222
8.5 Degree Elevation in a Bézier Surface 223
8.6 Composite Bézier Surfaces 224
8.7 Rational Bézier Patch 225

9 B-Spline Surfaces 227

9.1 The Tensor Product B-Spline Surface 227
9.2 Matrix Form 228
9.3 Open and Closed B-Spline Surfaces 228
9.4 Nonuniform Rational B-Spline Surfaces 235

10 Solids 237

10.1 Parametric Solids 237
10.2 The Tricubic Solid 240

10.3 Curves and Surfaces Embedded in a Solid 253
10.4 Generalized Notation Scheme and Higher-Dimension Elements 257
10.5 Instances and Parameterized Shapes 260
10.6 Sweep Solids 264
10.7 Controlled Deformation Solids 275

11 Complex Model Construction 281

11.1 Topology of Models 282
11.2 Graph-Based Models 310
11.3 Boolean Models 318
11.4 Boundary Models 356
11.5 Space-Partitioning Models 367

12 Relational Geometric Synthesis 373

12.1 Introduction 373
12.2 Relational Model Structure 378
12.3 Model Evaluation 383
12.4 Serialization 387
12.5 Relational Entities 388
12.6 Detailed Example 401
12.7 Applications 404
12.8 Open Issues in Relational Geometry 411
12.9 Conclusions 415

APPENDIX A Vectors 417

APPENDIX B Matrices 431

APPENDIX C Transformations 439

C.1 Translation 441
C.2 Rotation in the Plane 444
C.3 Rotation in Space about the Principal Axes 447
C.4 Rotation in Space about an Arbitrary Axis 453

C.5 Reflection 457
C.6 Dilation and Shear 462
C.7 Multiple Sequential Transformations 467

Bibliography **473**

Index **509**

1

INTRODUCTION

This chapter introduces geometric modeling in the context of the forces motivating and shaping its development, and tries to answer the question, *What is geometric modeling?* The history of geometric modeling is itself growing in importance because of what it tells us about the interplay of theory and application, academia and industry, as well as ideas and personalities, perhaps suggesting ways of managing these forces more effectively. This chapter reviews the highlights of this history. The mathematics of geometric modeling spans the disciplines of differential geometry, linear and Boolean algebra, topology, numerical methods, vectors, and matrices, and this chapter briefly discusses their use. The chapter concludes with a summary of the text's conventions and notation scheme.

1.1 WHAT IS GEOMETRIC MODELING?

Geometric modeling evokes a curious mix of the visual and the analytic in the eyes and minds of those of us who study and apply it. The visual response arises from its association with the simulation of shapes that define objects in the real world, from its association with the potential reality of the designer's work, and from its symbiosis with computer graphics. The analytical response arises from an awareness of the elegant and often subtle mathematics that is its foundation.

The term *geometric modeling* first came into use during the late 1960s and early 1970s, a time of rapidly developing computer graphics and computer-

aided design and manufacturing technologies. The discipline of geometric modeling is an interrelated, although somewhat loosely integrated, collection of mathematical methods that we use to describe the shape of an object or to express some physical process in terms of an appropriate geometric metaphor. These methods include computer-aided geometric design, solid modeling, algebraic geometry, and computational geometry. *Computer-aided geometric design* (*CAGD*) applies the mathematics of curves and surfaces to modeling, primarily using the parametric equations of differential geometry. It is here that we find the roots of contemporary geometric modeling. *Solid modeling,* usually encountered as *constructive solid geometry* (*CSG*), allows us to combine simple shapes to create complex solid models. CSG has its mathematical foundations in topology, algebraic geometry, and Boolean algebra. *Algebraic geometry* is the contemporary extension of classical analytic geometry, including differential geometry. *Computational geometry* is concerned with design and analysis of geometric algorithms, and has strong ties to numerical methods, computation theory, and complexity analysis. Computer-aided geometric design and constructive solid geometry are branches of geometric modeling, whereas algebraic and computational geometry tend other fields as well.

When we construct a model of some object, we create a substitute—a representation. The object may already exist physically, it may be the design for some as yet nonexistent object, it may be some virtual object never intended to be physically realized, or it may be the basis for the geometric interpretation of some physical process that we wish to visualize. An effective model is usually easier to test and analyze than the actual object, and it responds, within limits, in the same way as the actual object. To model something, then, is to give shape or form to it. In contemporary geometric modeling, we define a shape by a set of mathematical statements and logical relationships satisfying a set of axioms. These axioms we interpret as true statements about the model, and the consequent general properties of the model we analyze and evaluate as representative of the modeled object itself. The axioms, of course, correspond a priori to properties of the object that is modeled. Geometric modeling is the process of creating these statements and relationships.

Traditionally, models were made of clay or wood or rendered as sketches or engineering drawings. For large objects, the model was usually much smaller and easier to evaluate before committing to the construction of the real thing. Now we use geometric modeling to create a precise mathematical description of the shape of a real or hypothetical object, or to simulate a process or phenomenon that has a geometric analog. This description is analytical and abstract, in contrast to a physical model, which is literal and con-

crete. Physical models or mock-ups are still successfully used, and their story, in itself, is worthy of a separate study. We create a model, either physical or abstract, because it is a convenient and economical substitute for the real object or process, and because it is easier and more practical to analyze a model than to test, measure, or experiment with the real object or phenomenon. This may be because of size (the real object is too big or too small), complexity (the model is simpler relative to the characteristics of interest), or time (real phenomena are too fast, too slow, or uncontrollable). Beyond the *advantages of analysis, the mathematical model of an object's geometry is an* important way to convey information. For example, the abstract geometric model of a mechanical part is now commonly used to transmit design and production data between engineering and manufacturing operations. Through the medium of computer graphics and its rendering capabilities, a geometric model becomes the basis for exploring and evaluating aesthetic and functional qualities of some object. It is becoming more difficult, indeed, to overstate the importance of geometric modeling.

Geometric modeling, using differential and analytic geometry, vector and matrix methods, tensors, topology, set theory, and an arsenal of numerical computation methods to capture the potentially complex description of an object, requires the power of a computer. It is a computer-aided process, with the model stored in and analyzed by a computer. Using a computer is, in fact, central to the entire geometric-modeling process. Without computational power, we would be unable to construct and analyze models sophisticated or complex enough to be of any practical importance. So, even though the computer is not explicit in our studies, it is ever present, if only in the background as our implied "geometry engine."

Where we were once satisfied with two-dimensional representations of physical objects, we now demand and achieve topologically valid and analytically complete three-dimensional models. These models allow us to determine any geometric property or attribute. Obviously, the long association of geometric modeling with computer graphics demonstrates that visualization and appearance are important concerns, and we find that a complete model must satisfy the requirements of both rendering and analysis. In fact, we can now identify three important application categories of geometric modeling:

1. *Representation* of an existing object, where we must create a geometric model only once.
2. *Ab initio design,* where we must create a new object to satisfy some functional or aesthetic goals, defining and revising a hypothetical object's shape and performing appropriate analysis until we create a shape that fulfills the design criteria.

3. *Rendering*, where we must generate an image of the model in order to visually interpret and evaluate it.

These three categories are, of course, closely related. For example, when we first create the geometric model of a new, physically nonexistent product, it must be suitable for analysis and evaluation. After selecting a specific design, we can use its geometric model to guide the manufacture of the object. When production is complete, the model finally represents the physical shape of an existing object. At any point in this process, the geometric model provides information for rendering visual images of the object, including engineering drawings and computer graphics displays.

Computer graphics, computer-aided design, and computer-aided manufacturing have been and continue to be the driving forces behind the development of much of geometric modeling. Now robotics, computer vision, virtual reality, scientific visualization, and artificial intelligence are making new demands on geometric-modeling capabilities. Computer graphics systems can now routinely produce realistic two-dimensional color-shaded renderings of three-dimensional objects, including the appearance of texture, translucence, and the effects of multiple light sources. Art, animation, and virtual reality are each contributing their own impetus to the state of the art. Many visual special effects in feature films and advertising rely on these display systems and modeling techniques. Advances in solid modeling, polygonal, and sculptured-surface techniques have made much of this possible.

Development work continues on more sophisticated computer-aided design and manufacturing systems, with much of this effort aimed at creating a virtually paperless and seamless engineering and manufacturing process. In design and drafting, the engineer's activities may soon be more like sculpting than drafting. The new sculpting tools, using combinations of simple solid shapes and sophisticated shape transformations, will mimic not only those of the artist but also those forming processes available in manufacturing, extending a design engineer's ability to rapidly create and evaluate highly complex models.

Engineering analysis is an area undergoing change with the increasing sophistication of *solid modelers*. Solid modelers permit rapid construction of finite-element models. They also permit automatic static and dynamic structural analysis of mechanical parts subjected to a variety of loading conditions. The effects of these loads are then quickly and accurately displayed, using computer graphics.

Computer-aided *kinematic analysis* of complex mechanisms also becomes possible, because designers can move parts of a solid model independently and check clearances between those parts, either visually or analytically. It is

also now possible to ask other often-neglected questions, such as: Is there sufficient clearance for a wrench or screwdriver to reach and turn a fastener whose location allows only limited access? or Is it possible for a robotic spot-welder to reach a joint?

Perhaps one of the most fertile applications of geometric modeling is computer-aided manufacturing. For example, geometric modeling makes possible process planning and machine tool path-verification systems that are completely automatic. Engineering designers may furnish, via a computer database, complete and unambiguous geometric models of parts to be manufactured. Manufacturing programs then interpret these models and generate production and assembly instructions for the automated machine shop and assembly line. Automated inspection of finished parts is also now possible, improving quality assurance processes.

Developers of geometric-modeling systems now offer their products for application to robotics and computer vision, fields important to artificial intelligence systems. Advanced geometric-modeling systems will encourage a synthesis of these fields to produce intelligent robots that move in three dimensions and recognize objects from visual, tactile, and other sensory information. Ultimately, robots will be programmed to create, on their own, a geometric model of their immediate environment using artificial senses and adaptive powers of inference.

Geometric modeling is appropriately cast as an applied art and science. However, with strong stimulus from its client applications, the need for an overarching theory of geometric modeling is growing. As models become more complex, computer-aided design and manufacturing and computer graphics systems require automatic self-validation of the model, an understanding of geometric equivalence (both qualitative and quantitative), and more natural and powerful shaping procedures. Robotics and computer vision (Where am I? What is it?) require models amenable to spatial reasoning in configuration spaces. Virtual reality and scientific visualization require geometric models susceptible to dynamic influences. For all these the answer may lie in a new geometric metaphor encompassing both *shape* and *behavior*. The various conceptual fragments of contemporary geometric modeling offer partial solutions in these applications. However, some future integrative general theory of geometric modeling may yield more comprehensive results.

Much work remains to be done in geometric modeling on both fronts—theoretical and applied. We need a better understanding of why we model, of the kinds of models possible (physical and mathematical), and of the effectiveness of modeling and how it might best be measured. We need to explore the natural limits to visualization and how geometric modeling can work

within these limits and perhaps help to overcome them. We need a better understanding of the relationship between the object and its model, the strength of their correspondence, how to minimize ambiguities in a model, and how we choose a model.

1.2 HISTORY

The history of mathematical geometric modeling goes back over one thousand years, to the application of conic sections to naval architecture, when engineers and ship builders designed and lofted the lines of ships' hulls based on by then well-known methods for constructing conic curves. More recently, the shapes of ships and aircraft were manually lofted and stored in vast libraries of scribed sheet-metal sections. Then a revolution occurred. In the 1950s and early 1960s, the advent of the electronic computer changed forever the way we do modeling.

The roots of today's geometric modeling are found in the earliest computer graphic systems developed for computer-aided design and manufacturing. These, in turn, are descendants of the U.S. Air Force's SAGE computer system of the 1950s. Ivan Sutherland (1963), working at MIT, was among the earliest pioneers in this field with his *Sketchpad system*. This and other early graphics systems suggested wonderful possibilities for improving and extending the engineering design and drafting process, and they stimulated development of much of the mathematics that is the basis of today's geometric modeling. However, the modeling during that early period tended to emphasize appearance and rendition of design on the engineering side, with the notable exception of the finite-element models used in structural analysis and analogous models used in aerothermodynamic analysis.

On the manufacturing side, the history of geometric modeling begins with the simple subtractive modeling of numerical control processes. Computers were introduced into manufacturing to calculate and control the cutter motions of machine tools. This required a new way of understanding and extracting the shape information of an object's design from its engineering drawings. Such a task was not possible until special languages were developed to translate the shape information from the drawing into a computer-compatible format. D. T. Ross (1959), at the Illinois Institute of Technology, guided an early and highly successful effort to do just that, using classical analytic geometry as a basis for a modeling and machine control language. The *APT language* emerged from this work. Using APT, a parts programmer constructs a geometric model of an object to be machined from raw stock

and incorporates cutting instructions, which reference the model, for computer-controlled machine tools. This marked the beginning of the numerically controlled (NC) machinery revolution in industry.

In the mid-1960s, Ross (1967), then at MIT, developed an advanced compiler language for graphics programming and for generating problem-oriented languages. S. A. Coons (1963, 1965), also at MIT, and J. C. Ferguson (1964) at Boeing began important work in nonrational free-form curves and sculptured surfaces, using the cubic Hermite interpolation scheme. Coons' pioneering work was also important because it stimulated the development of other curve and surface representations. The early efforts of P. de Casteljau (1959, 1963) were quickly followed by the independent and insightful work of P. Bézier (1966, 1967, 1968), who produced a widely used method for curve and surface design.

At about this time, General Motors developed its DAC-1 system. Several other companies, notably Douglas, Lockheed, and McDonnell, also made significant developments. The Douglas effort was led by A. E. Eshleman and H. D. Meriwether (1966, 1967), who extended considerably the understanding and usefulness of the cubic and bicubic Hermitian forms of curves and surfaces. To Eshleman and Meriwether, the idea of a mathematical model describing and recording the geometry of the end product of a design was important, but the real prize was the process—the use and evolution of the model through all phases of design and analysis, and on into the manufacturing process itself. Of course, this put an emphasis on the universality and versatility of the model's representational form; a philosophical approach still bearing fruit, as witness the popular and widely used nonuniform rational B-Spline or NURBS form. The success of the NURBS form arose out of the simple fact that design and manufacturing processes still rely on standard analytic shapes, such as straight lines, circles, conics, and quadric surfaces. Rational curve and surface representation forms, like NURBS, are capable of incorporating both standard and free-form shapes.

Early limitations of the NC programming languages further stimulated work on the mathematics and applications of sculptured surfaces and solid modeling. This was reinforced by a growing need in the automotive and aircraft industries. Detroit was interested in the geometry of styling—body shapes, aesthetics, and highlights. Meanwhile, in Seattle, St. Louis, and Southern California, aircraft engineering departments were looking for better structural and aerodynamic shape-modeling tools. At this time, new ground was broken in the mathematics of parametric geometry, including Coons' bicubic patches and Bézier's (1974) special surfaces, with the notation scheme introduced by A. R. Forrest (1971). From a productive decade of research and development ending in the mid 1970s, there emerged the idea

of piecewise curves and surfaces, the joining together of many individual curve segments or surface patches to form more complex shapes.

Meanwhile, some workers in computer graphics and certain areas of computer-aided design (including architecture) began work in two areas of geometric modeling now called *wireframe* and *polygonal schemes*. These were initially two-dimensional and were intended as drafting tools or methods for reducing and interpreting digitized data. A *wireframe model* is composed of lines and curves defining the edges of an object, and it is usually constructed interactively. This is done in much the same way as an engineering drawing, where each line or curve element is constructed separately and independently. The model is stored in the computer as a list of lines and curves.

Current wireframe modeling systems are capable of constructing three-dimensional representations and have considerably enhanced interactive procedures. However, all pure wireframe systems exhibit well-known, often severe, deficiencies. For example, three-dimensional wireframe models are frequently ambiguous. Figure 1.1 is a good example of this ambiguity, because there are three equally likely passages through the object. It is also easy to create nonsense objects, since these systems usually have no internal logical tests to prohibit this. Figure 1.2a shows a classical example. Implied faces interpenetrate in a way that makes the interpretation of a physically realizable solid impossible. The wireframe in Figure 1.2b is a different kind of nonsense object, because it cannot be constructed or interpreted in three-dimensional space. Another deficiency is the lack of contour or profile information for surfaces inferred between the wireframe lines and curves. Figure 1.3 illustrates this limitation.

Figure 1.1 Wireframe ambiguity.

(a)

(b)

Figure 1.2 Wireframe nonsense objects.

Polygon modeling schemes were initially developed to create pictures or renderings, but there is a considerable gray area between polygon and wireframe systems, and the distinction is not always clear. The polygonal data structure is, however, straightforward, consisting of topologically cross-referenced lists of vertices, edges, and faces. The picture-generating and picture-manipulating algorithms associated with polygonal modeling systems are often highly sophisticated. This is true in part because polygonal schemes have been extensively used as research tools for computer graphics display technology, with animation and hidden-surface or ray-tracing techniques at the leading edge.

Figure 1.3 Missing contour information.

Coons, Ferguson, and others developed sculptured surfaces to replace the traditional lofting techniques of the shipbuilding, automotive, and aircraft industries with something faster, more accurate, and versatile. They were after forms that were easy to modify and that were compatible with analysis and manufacturing processes. The well-known parametric cubic patches of Coons and Ferguson and the formulations of Bézier, as mentioned previously, were early successes. Not much later, in the 1970s, W. J. Gordon and R. F. Riesenfeld (1974b) introduced and applied B-Spline curves and surfaces to computer-aided geometric design. Barsky (1981) introduced an important variation, the beta-spline. Rational formulations of the Bézier and B-Spline followed soon after, culminating in the very powerful and popular NURBS curves and surfaces (nonuniform rational B-Splines). I. Schoenberg (1946), M. G. Cox (1972), and C. de Boor (1972) laid the mathematical foundation for much of this work. More recently J. S. Letcher (1995) led the development and successful application of a new object-oriented framework for computer-aided geometric design, called *relational geometry* (see Chapter 12). Taken together these methods, grounded in the parametric curves and surfaces of differential geometry, form the core of computer-aided geometric design.

Solid modeling, a relative newcomer, is intended to overcome some of the limitations of the other schemes when representing and analyzing three-dimensional objects. The goal of solid modeling is to create unambiguous and complete geometric representations of objects. There have been several different approaches, including boundary representations (B-reps) and constructive solid geometry (CSG), pioneered by A. A. G. Requicha and H. B. Voelker (1977). Perhaps at the farthest frontier of geometric modeling and

still awaiting a wider recognition of its potential is true solid modeling, where the internal properties of an object are as fully represented as its surface properties. E. L. Stanton and L. M. Crain (1974, 1977) began work in this area over twenty years ago. Later chapters discuss these areas of solid modeling in much greater detail.

It is tempting to assume that the evolution of geometric modeling followed the line of descent from wireframe representations to sculptured surfaces, culminating in solid models. However, this is not the case. The record seems to show concurrent, but largely independent, development in all three areas. Even the seemingly later development of constructive solid geometry has a few tenuous roots in the APT technology of the 1960s. There have been and there continue to be efforts made to combine and hybridize the various modeling techniques, including the marrying of parametric and implicit geometries. Furthermore, the possibilities of interactive alteration and optimization of shapes has gained increasingly more attention, stimulating a reconsideration of all current modeling methods. The future of geometric modeling looks bright, and it is bound to bring many surprises.

1.3 THE MATHEMATICS

At first glance, there appears to be a large menagerie of unrelated curves and surfaces, but this is misleading. All parametric polynomial representations have an equivalent implicit form. The Hermite and Bézier representations are special cases of the B-Spline form, and the nonrational representations are special cases of the rational forms. This text proceeds from the more special and restricted forms to the more general and inclusive forms, and for curves and surfaces focuses particularly on parametric representations.

This study of geometric modeling makes extensive use of several mathematical techniques, in addition to the parametric equations of curves and surfaces from differential geometry. The most important of these techniques are *linear algebra, vectors, matrix methods, set theory* (including Boolean operations), *polynomial interpolation*, and *numerical methods*. Here is a brief review of some of them.

The *vector* is perhaps the single most important mathematical device used in geometric modeling. We can think of a vector itself as a geometric object, because it seems to fit our geometric intuition of displacement. This notion of objectivity is easily obscured if we are limited to working with vector components only individually and separately. Vector geometry offers a distinct advantage over classical analytic geometry by minimizing

dependence on a specific coordinate system. At the very least, vectors often allow the choice of a particular coordinate system to be postponed until the later stages of a computational process. Vectors, of course, carry more inherent geometric meaning than just displacement and direction. Simple vector operations reveal geometric relationships, such as perpendicularity and parallelism. These operations support algebraic methods while retaining geometric meaning. Last, but not least, vector equations handle several component equations at once. Appendix A presents a review of vectors.

Matrix algebra is another powerful tool of geometric modeling. The array of elements that makes up a matrix may represent simply an orderly way of storing numbers pertinent to some problem or perhaps a set of polynomial equation coefficients. The rules of matrix algebra define allowable operations on these arrays. Another use of a matrix is as an *operator*. Here, the matrix performs a geometric transformation on a set of points by operating on the position vectors that define those points. The interpretation of a matrix as a geometric operator is the foundation of many geometric-modeling computations. Because we will also encounter determinants in many operations and expressions in this book, it is a good idea to become reacquainted with their special properties. Appendix B reviews both matrices and determinants.

Approximation and *interpolation* describe two ways of fitting a curve or surface to a set of data points or curves. Under an approximation-fitting scheme, a curve must pass reasonably close to the data points but is not required to pass through, or interpolate, them. Under an interpolation-fitting scheme, however, the curve must pass through each data point. Here we use the powerful but simple theorem that says a straight line exactly interpolates (is defined by and passes through) two given points, a conic curve interpolates three points, a cubic interpolates four points, and so on. This suggests a polynomial of degree n: that is, a function of the form

$$f(x) = a_0 + a_1 x + \cdots + a_n x^n$$

These polynomials are widely used for interpolation because they can be evaluated, differentiated, and integrated easily and in a finite number of steps by using just the basic arithmetic operations of addition, subtraction, and multiplication.

Using polynomials of the same order as the number of points is very sensitive to the choice of interpolation points. At appropriately chosen points, they may produce a shape differing very little from that of the best approximation method. However, if the interpolating function is badly behaved,

producing an unacceptable shape anywhere in the sequence of interpolation points, then it may be poor everywhere. For example, the distribution of data points may lead us to expect a smooth, convex interpolating curve. However, the result may instead be a curve with many inflection points, that is, one that wiggles. Using *piecewise polynomials* usually allows us to sidestep this problem. That is, we can construct a *composite curve* consisting of a string of curve segments by fitting successive low-degree polynomial curves to corresponding successive groups of data points. In fact, as Farin (1993a) points out, high-degree polynomial interpolation is now mostly of theoretical value, because the piecewise approach is faster, more accurate, and reliably produces more acceptable shapes.

Numerical methods and *analysis* are necessary because the algorithms of geometric modeling are not processed on ideal computing machines with unlimited precision and capacity. This means that computations cannot be executed or carried out exactly. Horner's rule to evaluate a general polynomial of degree n and the forward difference method to evaluate a polynomial at equal intervals of the parametric variable are both discussed in Section 2.6.

An iterative technique called *Newton's method* (since he was the likely inventor of it) finds the roots of a given function $f(u) = 0$. A better name might be *the sliding tangent method*. This method works in certain controlled conditions. We isolate the roots u_r such that $u_j < u_r < u_k$, where there is only

Figure 1.4 Newton's method.

one root in this interval and the function (or curve) is well-behaved, with no discontinuities or inflections (see **Figure** 1.4). We let

$$f'(u) = \frac{df(u)}{du}$$

Then

$$u_{i+1} = u_i - \frac{f(u_i)}{f'(u_i)}$$

where $|f'(u_i)| > 0$ and $u_r = u_i$ when $u_{i+1} - u_i = 0$. If $f'(u_i)$ is zero or very small, then the shift or increment is very large, and convergence to a solution is doubtful. Here is an example:

$$f(u) = 30u^3 - 3u^2 + 9u - 15$$

$$f'(u) = 90u^2 - 6u + 9$$

$$u_{i+1} = u_i - \frac{30u^3 - 3u^2 + 9u - 15}{90u^2 - 6u + 9}$$

u_i	u_{i+1}
0.416667	→ 0.850596
0.850596	→ 0.720948
0.720948	→ 0.698190
0.698190	→ 0.697539
0.697539	→ 0.697539

$$u_r = 0.697539$$

Quadrature formulas evaluate integrals. The underlying assumption is that an integral represents the area under a curve defined by some function, say $f(u)$. Thus

$$dA = \int_a^b f(u)\, du$$

where $f(u)$ is an explicit formula that can be evaluated at any u in the range (a,b) (see Figure 1.5).

The *Newton-Cotes quadrature formulas* assume equally spaced ordinates, including the bounding ones (see **Figure** 1.6). The first four formulas are:

Figure 1.5 Quadrature.

$$A_2 = \frac{w}{2}[f(u_a) + f(u_b)] \qquad \text{(Trapezoidal rule)}$$

$$A_3 = \frac{w}{3}[f(u_a) + 4f(u_b) + f(u_c)] \qquad \text{(Simpson rule)}$$

$$A_4 = \frac{3w}{8}[f(u_a) + 3f(u_b) + 3f(u_c) + f(u_d)]$$

$$A_5 = \frac{4w}{90}[7f(u_a) + 32f(u_b) + 12f(u_c) + 32f(u_d) + 7f(u_e)]$$

$$\vdots$$

$$\vdots$$

where A_i is the total area and w is the width of a single quadrature division. For example, A_5 is the total area between a,e. The sequence continues for higher-order formulas.

Figure 1.6 Quadrature divisions.

Another approach, *Gaussian quadrature*, abandons equal spacing and allows the function to be sampled (evaluated) at more optimal locations. There are, of course, many other quadrature formulas. Each should be evaluated and selected for any particular problem based on its applicability.

1.4 CONVENTIONS AND NOTATION

Here is a list of the more common symbols and notations used in this text. In most cases, they will also be explained when first used. As for conventions, the right-handed Cartesian coordinate system is assumed unless noted otherwise.

$a, b, c, \ldots$	Scalars, constant coefficients: lowercase letters
$\mathbf{a}, \mathbf{b}, \mathbf{p}, \mathbf{r}, \mathbf{s}, \ldots$	Vectors: boldface, lower-case letters
$\lvert\mathbf{a}\rvert$	Magnitude, length of **a**, or norm of a vector
$\mathbf{A}, \mathbf{B}, \mathbf{M}, \ldots$	Matrices: boldface, uppercase letters
$\mathbf{A}^{-1}, \mathbf{B}^{-1}, \mathbf{M}^{-1}, \ldots$	Matrix inverse
$\mathbf{A}^T, \mathbf{B}^T, \mathbf{M}^T, \ldots$	Matrix transpose
$a_{ij}, b_{ij}, m_{ij}, \ldots$	Matrix elements or other array elements
$\mathbf{a}', \mathbf{p}', \mathbf{A}', \mathbf{P}', \ldots$	Transformed vector or matrix
$\lvert\mathbf{A}\rvert$ or $\lvert A\rvert$	Determinants
$A, B, Q, \ldots$	Quaternions
$s, t, u, v, w, \ldots$	Parametric variables
x, y, z or x_1, x_2, x_3	Cartesian coordinates
$\cup$	Union
$\cap$	Intersection
$-$	Difference
$a \in b$	a is an element of b
$\forall$	For all
$[a,b]$	Closed interval, continuous values
$[a : b]$	Closed interval, integral or discrete values
(a,b)	Open interval, continuous values

Differentiation:

$$y^{x} = \frac{dy}{dx}$$

$$y^{xx} = \frac{d^2y}{dx^2}$$

$$x^{u} = \frac{dx}{du}$$

$$x^{uw} = \frac{\partial^2 x}{\partial u\, \partial w}$$

and so forth.

$$\binom{n}{i} \textit{means } \frac{\mathrm{n}!}{i!(n-i)!}$$

The notation for Hermite, Bézier, and B-Spline curves and surfaces is far from uniform and standardized in the literature. The notation used in this text conforms to generally accepted, although not universal, forms. They are presented here for comparison and reference in both canonical and matrix form, including the rational B-Spline forms. The numbers in parentheses are the equation numbers of these expressions where they are first found in the text.

Hermite curve:

$$\mathbf{p}(u) = \sum_{i=0}^{n} \mathbf{b}_i F_{i,n}(u) \tag{3.11}$$

$$\mathbf{p}(u) = \mathbf{U}\mathbf{M}_F\mathbf{B} \tag{3.20}$$

Bézier curve:

$$\mathbf{p}(u) = \sum_{i=0}^{n} \mathbf{p}_i B_{i,n}(u) \tag{4.2}$$

$$\mathbf{p}(u) = \mathbf{U}\mathbf{M}_B\mathbf{P} \tag{4.7}$$

B-Spline curve:

$$\mathbf{p}(u) = \sum_{i=0}^{n} \mathbf{p}_i N_{i,K}(u) \tag{5.1}$$

$$\mathbf{p}_s(u) = \mathbf{U}\mathbf{M}_S\mathbf{P}_K \tag{5.16}$$

$$\mathbf{p}(u) = \frac{\sum_{i=0}^{n} h_i \mathbf{p}_i N_{i,K}(u)}{\sum_{i=0}^{n} h_i N_{i,K}(u)} \tag{5.49}$$

Hermite surface:

$$\mathbf{p}(u, w) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{b}_{ij} F_{i,m}(u) F_{j,n}(w)$$

$$\mathbf{p}(u,w) = \mathbf{U}\mathbf{M}_F\mathbf{B}\mathbf{M}_F^T\mathbf{W}^T$$

Bézier surface:

$$\mathbf{p}(u,w) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{p}_{ij} B_{i,m}(u) B_{j,n}(w)$$

$$\mathbf{p}(u,w) = \mathbf{U}\mathbf{M}_B\mathbf{P}\mathbf{M}_B^T\mathbf{W}^T$$

B-Spline surface:

$$\mathbf{p}(u,w) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{p}_{ij} N_{i,K}(u) N_{j,L}(w)$$

$$\mathbf{p}_{st}(u,w) = \mathbf{U}\mathbf{M}_S\mathbf{P}_{KL}\mathbf{M}_S^T\mathbf{W}^T$$

$$\mathbf{p}(u,w) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} h_{ij}\mathbf{p}_{ij} N_{i,K}(u)\, N_{j,L}(w)}{\sum_{i=0}^{m} \sum_{j=0}^{n} h_{ij} N_{i,K}(u)\, N_{j,L}(w)}$$

2

CURVES

Curves are often defined as *the locus of a point moving with one degree of freedom*. Another definition describes a curve as *the locus of a one-parameter family of points*. History records many variations on the idea of a curve as a path of one dimension, having length only. Such definitions may help us to visualize a curve and to improve our intuitive sense of its behavior, but they are not explicitly analytical. Of more importance to geometric modeling is that these definitions lead to useful analytical expressions. Ways to mathematically describe curves for geometric modeling include *intrinsic equations*, *explicit* and *implicit equations*, and *parametric equations*. The latter category is expressed powerfully in the Hermite, Bézier, and B-Spline forms. This chapter briefly reviews and compares these methods for representing curves. Many modeling problems require us to determine a series of points on a curve; so this chapter concludes with a discussion of the direct- and inverse-point solution.

2.1 INTRINSIC EQUATIONS OF CURVES

An *intrinsic property* is one that depends on only the figure in question, and not its relation to a coordinate system or other external frame of reference. The fact that a rectangle has four equal angles is intrinsic to the rectangle, but the fact that a particular rectangle has two vertical sides is *extrinsic*, because an external frame of reference is required to determine which direction is vertical.

Intrinsic descriptions of a figure are easy to construct and understand. Imagine that you are standing in a large open space, free of any obstructions or convenient reference points, and that you want to do a simple experiment in intrinsic geometry. First, you decide to walk in a square path; this is easy, because the intrinsic definition of a square is simple. From any initial position and orientation, walk ahead, say, 10 paces; turn to your right 90 degrees, and walk 10 more paces; turn right 90 degrees again, and walk 10 more paces. Repeat this one more time, and you should be back at your starting position. The 90-degree turns certainly do not require any external reference system, since you can easily imagine various "local" methods to determine these turns, and counting to 10 is obviously a purely local operation. You have just exercised a procedural intrinsic definition of a square, and what works for a square works equally well with other figures. Try this with a circle. Again, start anywhere. How many different intrinsic procedures can you find for walking in a circle? This demonstrates that intrinsic definitions are more local than traditional Cartesian coordinate definitions. Intrinsic definitions deal with geometry a little piece at a time.

These exercises may strengthen our intuitive appreciation of an intrinsic description, but they are not rich in analytical potential. For this we must look at an example of an intrinsic equation. A curve requires two intrinsic equations, one expressing its curvature $1/\rho$ and one its torsion τ as functions of its arc length s:

$$\frac{1}{\rho} = f(s) \qquad \text{and} \qquad \tau = g(s) \tag{2.1}$$

Torsion is a measure of how much a space curve deviates from a plane curve (how much it tries to twist out of the osculating plane), and arc length is, of course, the length measured along the curve. The theory of curves proceeds from these intrinsic equations.

It is interesting to make a distinction between intrinsic equations, as just defined, and the so-called *natural equations*. A natural equation of a curve is any equation connecting its curvature $1/\rho$, torsion τ, and arc length s:

$$\tilde{f}\left(\frac{1}{\rho}, \tau, s\right) = 0 \tag{2.2}$$

This equation imposes a condition on the curve, so that it has certain special properties, but many curves may have these properties. For example, $\tau = 0$ is a natural equation characterizing all plane curves, and $1/\rho = 0$ is also a natural equation characterizing all straight lines. An additional independent natural equation $\tilde{g}(1/\rho, \tau, s) = 0$ of the curve determines the curve still more. Solving the two natural equations $\tilde{f}(1/\rho, \tau, s) = 0$ and $\tilde{g}(1/\rho, \tau, s) = 0$ simultaneously for

$1/\rho$ and τ as functions of s produces intrinsic equations. Two natural equations determine a curve uniquely, except for its position in space.

Here is a slightly different approach, using Figure 2.1 and limited to plane curves: Given the initial point of a curve, the variation of θ with arc length s will completely define the curve. θ is the angle subtended by the tangent to the curve with the x axis. The relationship between s and θ is an intrinsic equation of the curve.

Curvature, $\kappa = 1/\rho$, emerges from the intrinsic equation

$$\kappa = \frac{d\theta}{ds} \tag{2.3}$$

A parametric description of a curve in terms of its arc length begins with the equations $x = x(s)$ and $y = y(s)$. The functions $x(s)$ and $y(s)$ are related by the equations

$$\frac{dx}{ds} = \cos\theta \qquad \text{and} \qquad \frac{dy}{ds} = \sin\theta \tag{2.4}$$

Differentiating these equations with respect to s and substituting κ for $d\theta/ds$, dx/ds for $\cos\theta$ and dy/ds for $\sin\theta$ yields the simultaneous differential equations

$$\begin{aligned} \frac{d^2x}{ds^2} + \frac{\kappa(s)dy}{ds} &= 0 \\ \frac{d^2y}{ds^2} - \frac{\kappa(s)dx}{ds} &= 0 \end{aligned} \tag{2.5}$$

Figure 2.1 Intrinsic definition of a curve.

Using appropriate numerical procedures, we solve these two second-order equations for $x(s)$ and $y(s)$ for any given curvature function $\kappa(s)$.

2.2 EXPLICIT ANO IMPLICIT EQUATIONS OF CURVES

In the plane, the explicit equation of a curve takes the general form

$$y = f(x) \tag{2.6}$$

In this form, there is only one y value for each x value. Consequently, it cannot represent closed or multiple-valued curves. This limitation is avoided by using an implicit equation of the general form

$$f(x, y) = 0 \tag{2.7}$$

Both explicit and implicit forms are axis dependent. Therefore, the choice of the coordinate system affects the ease of modeling the curves and calculating their properties. In spite of this limitation, implicit equations prove to be useful, usually as an adjunct to a larger, parameter-based modeling scheme. In fact, mathematicians have demonstrated that every parametric form has a corresponding implicit form, but there are implicit forms that have no known parametric representation or are not amenable to parameterization. The implicit formulation expedites the computation of intersections and the classification of points. The later characteristic allows us to determine if a given point is on the curve or on one side of it or the other. Both these procedures are more difficult with a parametric representation. Sederberg and Goldman (1986) and Hoffmann (1993), among others, discuss methods of conversion between parametric and implicit forms.

The implicit equations of the straight line and conic curve are

$$Ax + By + C = 0$$

and

$$Ax^2 + 2Bxy + Cy^2 + Dx + Ey + F = 0$$

respectively. An analysis of the coefficients of these equations yields information about special geometric characteristics of a line or curve. Section 2.4 discusses more about this analysis for the conic.

Even a very simple analysis of the equation of an implicitly defined curve reveals characteristics important to the geometric modeling process. For example, the following conditions determine the possible symmetries of a plane curve:

1. If $f(x, y) = \pm f(-x, -y)$, the curve is symmetric about the origin (or other point, given an appropriate transformation); for example, a circle $x^2 + y^2 = r^2$.
2. If $f(x, y) = \pm f(x, -y)$, the curve is symmetric about the x axis; for example, the parabola $x = y^2$.
3. If $f(x, y) = \pm f(-x, y)$, the curve is symmetric about the y axis; for example, the parabola $y = x^2$.
4. If $f(x, y) = \pm f(y, x)$, the curve is symmetric about the line $x = y$; for example, the hyperbola $xy = k^2$.
5. If $f(x, y) = \pm f(-y, -x)$, the curve is symmetric about the line $x = -y$; for example, the hyperbola $xy = -k^2$.

2.3 PARAMETRIC EQUATIONS OF CURVES

Ordinary, single-valued explicit functions, such as $y = f(x)$, cannot represent most shapes used in geometric modeling. There are many reasons for this. First, the shapes of most objects are intrinsically independent of any coordinate system. It is the relationship between the points themselves that determines the resulting shape of a curve or surface fit through a set of points, not the relationship between these points and some arbitrary coordinate system. In fact, most modeling applications require that the choice of a coordinate system should not affect the shape. Second, any solid, closed object will have some tangent lines or planes parallel to the principal axes or planes of any chosen coordinate system. For shapes defined by explicit functions, this results in values of infinity for some slopes on the model, or results in some other ill-defined mathematical properties. Finally, the curves and surfaces of geometric modeling are often nonplanar and bounded, and are not easily represented by an ordinary nonparametric function.

For these reasons and many others related to ease of programming and computability, the preferred way to represent shapes in geometric modeling is with parametric equations. For example, a two-dimensional curve is defined not by a single ordinary function like $y = f(x)$, but by a set of two functions $x = x(u)$, $y = y(u)$ of a parameter u. A three-dimensional curve is defined by $x = x(u)$, $y = y(u)$, and $z = z(u)$. These curves have a natural vector representation. For a space curve,

$$\mathbf{p}(u) = [x(u) \quad y(u) \quad z(u)] \tag{2.8}$$

and, by simple extension, for a surface,

$$\mathbf{p}(u, w) = [x(u, w) \quad x(u, w) \quad z(u,w)] \tag{2.9}$$

Parametric equations avoid many of the problems associated with nonparametric functions. They also best describe the way curves are drawn by a plotter or some computer graphics display screens. Here two time functions $x(t)$ and $y(t)$ control the servo system of a plotter or the electron beam deflection system of a vector display, causing the pen or electron beam to move on the appropriate curve.

The *point* is the basic element of parametric geometry and geometric modeling. It is an ordered set of real numbers, whose implied coordinate nature defines a position in three-dimensional Euclidean space. A vector may be defined in much the same way, so that frequently the terms *point* and *vector* are interchangeable. For example, the radius vector (or position vector) defines a point.

Parametric equations generate the sets of points defining most of the curves, surfaces, and other geometric elements discussed in this text. The following equations are an example of the parametric representation of a curve (see Figure 2.2). Of course, it is not possible to plot the curve for all values of u from $-\infty$ to $+\infty$. We must select an interval that has some significance to the modeling situation and that has computational convenience. The curve in the figure is plotted for an interval on u of $u \in [-1,1]$, for the equations

Figure 2.2 A parametric curve.

$$x = 3u^2$$
$$y = u^3 - u + 1 \tag{2.10}$$
$$z = 2u + 3$$

Here, u is the parametric variable and also the independent variable. Substituting a specific value of u into each of these equations produces specific values of x, y, and z. Each value of u generates a point on the curve. This produces three other curves as well. These are the curves (x,u), (y,u), and (z,u) in parameter space (see below), also shown in the figure.

It is usually convenient to normalize the domain of the parametric variable, which means restricting its value to the closed interval between 0 and 1, inclusive. We express this condition symbolically as $u \in [0,1]$, establishing the curve bounding points and creating a *curve segment*. Some examples of parametric equations of curves using this convention follow. The first example is a set of linear parametric equations

$$x = a + lu \qquad y = b + mu \qquad z = c + nu \tag{2.11}$$

where a,b,c, and l,m,n are constants and $u \in [0,1]$. This particular curve is a straight line starting at point $\mathbf{p}(0) = [a \quad b \quad c]$ and ending at $\mathbf{p}(1) = [(a + l) \quad (b + m) \quad (c + n)]$, with direction cosines proportional to l,m,n.

Another example is

$$x = u \qquad y = u^2 \qquad z = u^3 \tag{2.12}$$

This curve is a cubical parabola and also one form of a twisted cubic. Because $u \in [0,1]$, it is bounded by $\mathbf{p}(0) = [0 \quad 0 \quad 0]$ and $\mathbf{p}(1) = [1 \quad 1 \quad 1]$.

The final example is

$$x = a \cos u \qquad y = a \sin u \qquad z = bu \tag{2.13}$$

This curve is a left-handed circular helix, or machine screw. It is the locus of a point that revolves around the z axis at a constant distance a from it (in other words, a helix of radius $= a$). At the same time, the helix moves parallel to the z axis at a rate proportional to the angle of revolution u. If $b < 0$, then the helix is right-handed.

The versatility of the parametric form is apparent in these examples, and it is interesting to see how it adapts to curve-defining functions expressing a direct relationship between the Cartesian variables x,y,z. The following discussion reveals this not-always-obvious relationship. For example, one equation in x,y,z represents a surface. Two independent simultaneous equations in x,y,z, say,

$$F(x,y,z) = 0 \qquad G(x,y,z) = 0 \tag{2.14}$$

represent the intersection of two surfaces, which is locally a curve. They are the implicit equations of a curve in three dimensions. A curve defined this way is inherently unbounded; however, only a bounded part of it may he of interest.

Solving the implicit equations for two of the variables in terms of the third, say for y and z in terms of x, produces

$$y = y(x) \qquad z = z(x) \tag{2.15}$$

These equations represent the same curve as do Equations (2.14), and they, or the equations similarly expressing any two of the coordinates of a variable point on the curve as functions of the third coordinate, are the explicit equations of the curve. Each of Equations (2.15) separately represents a cylinder projecting the curve onto one of the principal planes, so they are a special form of Equations (2.14) for which the two surfaces are projecting cylinders.

Solving the first of the three parametric equations $x = x(u)$, $y = y(u)$, and $z = z(u)$ of a curve for u as a function of x, that is, $u(x)$, and substituting the result into the two remaining expressions, produces the explicit Equations (2.15). From one point of view, these explicit equations, when supplemented by the identity $x = x$, are also parametric equations of the curve; that is,

$$x = x \qquad y = y(x) \qquad z = z(x) \tag{2.16}$$

Now the parameter is also the coordinate x. The difficulty with this approach is the obviously unacceptable limitation on the range of x, because the parametric variable must be normalized to the interval $x \in [0,1]$, or a similar finite domain, to define a curve segment. Introducing a parametric function of the following form easily resolves this problem:

$$u = \frac{x - x_0}{x_1 - x_0} \tag{2.17}$$

This expression allows an explicit range of x from x_0 to x_1, provided $u \in [0,1]$, through the parametric functions $x(u)$, where

$$x = x(u) = x_0 + (x_1 - x_0)u \tag{2.18}$$

It satisfies the normalization condition on the parametric variable without compromising the range of x. Substituting this relationship into Equations (2.16) yields $x = x(u)$, $y = y[x(u)]$, and $z = z[x(u)]$, which, of course, simplify to $x = x(u)$, $y = y(u)$, and $z = z(u)$. These are the parametric forms introduced earlier, which means that a large class of explicit functions easily converts into parametric form.

All of this leads to the following definition: A *curve segment* is a bounded collection of points whose coordinates are given by continuous, one-parameter, single-valued polynomials of the form

$$x = x(u) \qquad y = y(u) \qquad z = z(u) \tag{2.19}$$

where the parametric variable u is ordinarily constrained to the closed interval $u \in [0,1]$, and the positive sense on the curve is the direction in which u increases. The curve is point-bounded because it has two definite endpoints, one at $u = 0$ and the other at $u = 1$. The $x(u)$, $y(u)$, and $z(u)$ functions may be of a fixed polynomial form, defining basis functions that generate curves with a characteristic set of properties. Special basis functions generate the Hermite, Bézier, and B-Spline curves, discussed in Chapters 3, 4, and 5, respectively.

We can treat the coordinates of any point on a parametric curve as the components of a vector $\mathbf{p}(u)$. Figure 2.3 illustrates this and other important vector elements of a curve. Unless noted otherwise, the tangent vectors are not drawn to scale. Here $\mathbf{p}(u)$ is the vector to the point $x(u)$,$y(u)$,$z(u)$, and $\mathbf{p}^u(u)$ is the tangent vector to the curve at that point, obtained by differentiating $\mathbf{p}(u)$:

$$\mathbf{p}(u) = \frac{d\mathbf{p}(u)}{du} \tag{2.20}$$

The vector components of $\mathbf{p}^u(u)$ are

$$x^u = \frac{dx(u)}{du} \qquad y^u = \frac{dy(u)}{du} \qquad z^u = \frac{dz(u)}{du} \tag{2.21}$$

These are the parametric derivatives. When u appears as a superscript, it indicates differentiation with respect to u. The relationship between the parametric derivatives and the ordinary derivatives of Cartesian space is

$$\frac{dy}{dx} = \frac{dy/du}{dx/du}$$

and similarly for dy/dz and dz/dx.

Figure 2.3 Elements of a parametric curve.

Model space is the three-dimensional space defined by the familiar Cartesian coordinates x,y,z. It is the space in which a geometric model is fully developed and expressed. The *parameter space* of a curve is the set of three two-dimensional spaces defined by (x,u), (y,u), and (z,u). Any parametric curve can be decomposed into its three components in parameter space. Plots of a curve in terms of its parameter space components, called *cross plots*, are sometimes useful in understanding its behavior in model space.

Figure 2.4 demonstrates the decomposition of a specific curve whose algebraic representation is

$$
\begin{aligned}
x(u) &= -6.5u^3 + 9u^2 + 0.5u + 1 \\
y(u) &= -12u^3 + 18u^2 - 2 \\
z(u) &= -10.25u^3 + 14.75u^2 + 1.5u - 2
\end{aligned}
\tag{2.22}
$$

Figure 2.4 Parameter space of a curve.

where

$$x'' = -19.5u^2 + 18u + .5u$$
$$y'' = -36u^2 + 36u \tag{2.23}$$
$$z'' = -30.75u^2 + 29.5u + 1.5$$

This figure shows the parametric components plotted in the interval $u \in [0,1]$. These are mathematically convenient delimiting values of the parametric variable, although any interval on the u line can be used without affecting the shape or position of the curve if appropriate adjustments are made to the algebraic or geometric coefficients. Later, we will discuss how to change this interval using a procedure called *reparameterization*. Different horizontal and vertical scales on these plots affect the apparent slope of the tangent vectors. We replot them so that the x-, y-, z-, and u-axis scales are equal.

The algebraic coefficients control the position of a curve in space as well as its size and shape. Each value of u (for example, u_i in Figure 2.4) defines a unique point on the curve in model space. Plotting the successive sets of x,y,z values generated by successive u values creates the entire curve. Each variable x, y, and z is independently controlled, and a different curve results even if only one coefficient is changed.

The graphs in parameter space show the behavior of each Cartesian coordinate from one end of the curve to the other as the parametric variable u varies over its domain from 0 to 1. The graphs also show the behavior of the parametric derivatives as represented by the slopes at points on these curves (for example, dx/du at $u = u_i$). Each Cartesian coordinate varies continuously over its range (that is, from $\mathbf{p}_0$ to $\mathbf{p}_1$) as the parametric variable u varies continuously over its domain. Here x, y, and z are the dependent variables whose values are determined by the independent variable u.

We can infer many characteristics of a curve by inspecting the graphs of the component curves. Figure 2.5 shows several special cases that are easily verified by sketching the resulting curves in model space. Here we see a curve contracted to a point (a degenerate curve of zero length), a straight line, a plane curve, or a space curve, depending on the values assigned to its coefficients. The Cartesian coordinates x,y,z are stacked in the figure, and another informative graphic technique is to superimpose them. Thus, we plot the value of the parametric variable along the horizontal axis and plot and superimpose the model-space variables along a common vertical axis.

Parametric equations have many advantages over other forms of representation. Here are the most important ones:

Figure 2.5 Special curves shown in parameter space.

- They allow separation of variables and direct computation of point coordinates.
- It is easy to express parametric equations as vectors.
- Each variable is treated alike.
- There are more degrees of freedom to control curve shape.
- Transformations may be performed directly on them.
- They accommodate all slopes without computational breakdown.
- Extension or contraction into higher or lower dimensions is direct and easy without affecting the initial representation.
- The curves they define are inherently bounded when the parameter is constrained to a specified finite interval.
- The same curve often can be represented by many different parameterizations. Conversely, a parameterization scheme is sometimes chosen because of its effect on curve shape.

See Ball (1984), Filip and Ball (1989), and others for in-depth discussion of the effects of parameterization on shape.

2.4 CONIC CURVES

A *conic curve* is defined by a second-degree implicit equation. Conversely, any second-degree equation defines a conic. The standard form is

$$Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0 \tag{2.24}$$

In matrix form, this equation becomes

$$\mathbf{PQP}^T = 0 \tag{2.25}$$

where

$$\mathbf{Q} = \begin{bmatrix} A & B & D \\ B & C & E \\ D & E & F \end{bmatrix} \tag{2.26}$$

and

$$\mathbf{P} = [x \quad y \quad 1] \tag{2.27}$$

Here **P** is given by homogeneous coordinates.

Table 2.1 Conic curve characteristics

k	$\lvert\mathbf{Q}\rvert$	Other conditions	Type
0	$\neq 0$		Parabola
0	0	$C \neq 0, E^2 - CF > 0$	Two parallel real lines
0	0	$C \neq 0, E^2 - CF = 0$	Two parallel coincident lines
0	0	$C \neq 0, E^2 - CF < 0$	Two parallel imaginary lines
0	0	$C = B = 0, D^2 - AF > 0$	Two parallel real lines
0	0	$C = B = 0, D^2 - AF = 0$	Two parallel coincident lines
0	0	$C = B = 0, D^2 - AF < 0$	Two parallel inaginary lines
<0	0		Point ellipse
<0	$\neq 0$	$-C\lvert\mathbf{Q}\rvert > 0$	Real ellipse
<0	$\neq 0$	$-C\lvert\mathbf{Q}\rvert < 0$	Imaginary ellipse
<0	$\neq 0$		Hyperbola
<0	0		Two intersecting lines

Certain characteristics of the conic equation are invariant under translation and rotation transformations. These include $A + C$, $k = AC - B^2$ and the determinant $|\mathbf{Q}|$. The values of k and $\mathbf{Q}$ indicate the type of conic curve. Table 2.1 classifies conic curves using these characteristics.

Common parametric forms of conic curves are

Parabola:

$$x = au^2$$
$$y = 2au \tag{2.28}$$

Hyperbola:

$$\begin{aligned} x &= a \sec u \\ y &= b \tan u \end{aligned} \qquad u \in [-\pi, \pi] \tag{2.29}$$

Ellipse:

$$\begin{aligned} x &= a \cos u \\ y &= b \sin u \end{aligned} \qquad u \in [-\pi, \pi] \tag{2.30}$$

2.5 HERMITE, BÉZIER, AND B-SPLINE CURVES: AN OVERVIEW

The next three chapters discuss the Hermite, Bézier, and B-Spline parametric curves, including both the nonrational and rational forms of the latter

two. Each nonrational form is a special case of a rational form. The cubic Hermite curve and bicubic Hermite patch were among the earliest forms investigated and used in geometric modeling. Their relative simplicity and natural versatility makes them good starting points for studying curves and surfaces. The Hermite curves are a special case of the Bézier and B-Spline forms, and the Bézier form is itself a special case of the B-Spline. The Hermite, Bézier, and B-Spline curves form the basic elements of splines: curves that consist of individual parametric curve segments joined to form a single composite curve whose continuity is controlled at the segment joints. We are primarily interested in the cubic polynomial representations, because this is the lowest degree capable of describing nonplanar curves. A brief comparison of their most important characteristics follows.

Hermite interpolation is not restricted to points. In its standard form for curves it uses both points and derivative data. For example, a cubic Hermite curve is defined by its two end points and the tangent vectors at those points. It interpolates all its control points and is fairly easy to subdivide. However, its lack of invariance under affine transformations can be troublesome if not accounted for. This curve's name is identified with Charles Hermite (1822–1901), a French mathematician of considerable accomplishments in the areas of cubic and quintic polynomials.

The *Bézier curve* is defined by a set of control points, the number of which determines the degree of the Bernstein polynomial basis functions that describe its shape. This curve interpolates its first and last control points and is tangent to the first and last sides of the open polygon defined by these control points. It exhibits the convex hull property and is the easiest of the curve forms to subdivide. However, it does not offer local control, because changing the position of any control point produces shape change throughout the curve.

The *B-Spline curve* is a piecewise polynomial curve defined by a set of control points which the curve ordinarily does not interpolate. The degree of its polynomial basis function is defined independently of the number of control points. Local control of curve shape is possible because changes in control point location do not propagate shape change globally, and control points influence only a few of the nearby curve segments. A valuable property of B-Splines is that they are invariant under affine transformations. Filip and Ball (1989) and others point out that some types of curves cannot be practically represented as B-Splines. For example, this is true for a procedurally defined intersection curve, which is likely to require a very high degree polynomial for its representation.

Barsky (1981) introduced a generalization of the uniform cubic B-Spline called the *beta-spline*. This form permits parametric discontinuities while

preserving continuity of the unit tangent and curvature vectors at joints. This provides two additional parameters, called *bias* and *tension*, that are independent of the control vertices and by which the shape of a curve or surface can be manipulated. A rational formulation of the beta-spline is reported in Barsky (1993) and earlier by Joe (1987).

B-Spline curves and Bézier curves have many advantages in common. Control points influence curve shape in a predictable, natural way, making them good candidates for use in an interactive environment. Both types of curve are variation diminishing, axis-independent, and multivalued, and both exhibit the convex hull property. However, it is the local control of curve shape possible with B-Splines that gives this form an advantage over the Bézier technique, as does the ability to add control points without increasing the degree of the curve.

2.6 POINTS ON A CURVE

To plot a curve $\mathbf{p}(u)$, we calculate the coordinates of a series of points on it for different values of the parametric variable, which requires that we evaluate a polynomial at predetermined increments of u. This process is known as the *direct-point solution*. Other modeling tasks require that we determine the value of the parameter u corresponding to a point $\mathbf{p}(u)$ which is given in terms of its coordinates x, y, z. This is the *inverse-point solution*.

Direct-Point Solution

Most methods for modeling curves and surfaces, including Hermite, Bézier, and B-Spline methods, express the x, y, z coordinates of points on these elements as parametric polynomial functions. The problem of calculating the coordinates of a specific point on a curve becomes one of calculating a polynomial, and a simple, straightforward method for calculating polynomials is *Horner's rule*. Although the development that follows is in terms of scalar polynomials, it applies equally well to vector polynomials. For the cubic polynomial $p(u) = au^3 + bu^2 + cu + d$, we rewrite it so that only three multiplications and three additions are required to compute the solution for a given value of u:

$$p(u) = [(au + b)u + c]u + d \tag{2.31}$$

To evaluate the polynomial of degree n

$$p(u) = a_n u^n + a_{n-1} u^{n-1} + a_{n-2} u^{n-2} + \cdots + a_1 u + a_0 \tag{2.32}$$

we generalize Horner's rule as follows:

$$p(u) = a_1u + a_0 \qquad \text{for} \qquad n = 1$$

$$p(u) = (a_2u + a_1)u + a_0 \qquad \text{for} \qquad n = 2$$

$$p(u) = [(a_3u + a_2)u + a_1]u + a_0 \qquad \text{for} \qquad n = 3$$

$$p(u) = \{[(a_4u + a_3)u + a_2]u + a_1]u + a_0 \qquad \text{for} \qquad n = 4$$

and so on. For any n, we develop a straight-line program of $2n$ steps to evaluate a general nth-degree polynomial. Clearly, n multiplications and n additions are necessary for each of the three coordinates of a point. Combining Horner's rule with a straight-line program yields:

$$\left.\begin{aligned} t &\leftarrow a_1u \\ p &\leftarrow t + a_0 \end{aligned}\right\} \text{for} \qquad n = 1$$

$$\left.\begin{aligned} t &\leftarrow a_2u \\ t &\leftarrow t + a_1 \\ t &\leftarrow tu \\ p &\leftarrow t + a_0 \end{aligned}\right\} \text{for} \qquad n = 2$$

$$\left.\begin{aligned} t &\leftarrow a_3u \\ t &\leftarrow t + a_2 \\ t &\leftarrow tu \\ t &\leftarrow t + a_1 \\ t &\leftarrow tu \\ p &\leftarrow t + a_0 \end{aligned}\right\} \text{for} \qquad n = 3$$

To evaluate points on a cubic Bézier curve using a variation on Horner's nested multiplication scheme, we find that

$$\mathbf{p}(u) = \left\{\left[\binom{3}{0}(1-u)\mathbf{p}_0 + \binom{3}{1}u\mathbf{p}_1\right](1-u) + \binom{3}{2}u^2\mathbf{p}_2\right\}(1-u) + \binom{3}{3}u^3\mathbf{p}_3$$

where

$$\binom{3}{0}, \binom{3}{1}, \ldots,$$

are binomial coefficients. Farin (1993a) describes de Casteljau's algorithm that recursively computes points on a Bézier curve.

Alternatively, we may use incremental methods to compute coordinates of points on a curve for successive values of the parametric variable. The polynomial format of the parametric functions lends itself to these methods.

We will investigate one method, the *forward difference method,* to evaluate a polynomial at equal intervals of the parametric variable.

We begin with the simple case of a linear equation $p(u) = cu + d$ and evaluate $p(u)$ for $n + 1$ equally spaced values of u. To find a set of values p_i where $p_i = p(i/n), i \in [0 : 1]$, and $u \in [0,1]$, we observe that the difference between two successive values of $p(u)$ is constant: $p_{i+1} = p_i + c/n$. Thus, we find successive values of the polynomial $p(u)$ by adding the constant c/n to the previous value. This method is used effectively in the incremental plotting and displaying of curves.

To apply this method to polynomials of any degree, we first consider the cubic polynomial $p(u) = au^3 + bu^2 + cu + d$. Here, the forward difference $p_{i+1} - p_i = d_{1,i}$ is not a constant, but a quadratic polynomial in i. If we can evaluate $d_{1,i}$ easily, then we can also evaluate p_i easily. Applying the same process (that is, forward difference) to evaluate $d_{1,i}$, we find $d_{1,i+1} - d_{1,i} = d_{2,i}$, a linear equation in i. This equation has a constant forward difference and is easily evaluated.

Here is an algorithm for evaluating a cubic polynomial, carrying out the algebra to combine the three levels of forward differences:

$$
\begin{aligned}
d_{1,i} &= (3i^2 + 3i + 1)\frac{a}{n^3} + (2i + 1)\frac{b}{n^2} + \frac{c}{n} \\
d_{2,i} &= 6(i + 1)\frac{a}{n^3} + \frac{2b}{n^2} \\
d_{3,i} &= \frac{6a}{n^3}
\end{aligned}
\tag{2.33}
$$

We initialize these variables and p at $u = 0$, where $i = 0$ and

$$
\begin{aligned}
p &= d \\
d_1 &= \frac{a}{n^3} + \frac{b}{n^2} + \frac{c}{n} \\
d_2 &= \frac{6a}{n^3} + \frac{2b}{n^2} \\
d_3 &= \frac{6a}{n^3}
\end{aligned}
\tag{2.34}
$$

Thus, $p_0 = d$. To calculate p at the next and each successive increment, we require three additions:

$$
\begin{aligned}
d_2 &\leftarrow d_2 + d_3 \\
d_1 &\leftarrow d_1 + d_2 \\
p &\leftarrow p + d_1
\end{aligned}
$$

We must repeat this three-step algorithm n times, generating values for p_1 to p_n. Other methods for evaluating cubic polynomials require several multiplications for each evaluation. However, after the computations to initialize the four variables p, d_1, d_2, and d_3, we require only three additions for each point. We can apply this technique to polynomials of any degree, resulting in a loop that uses n additions for polynomials of degree n.

One difficulty with the forward difference method is that we must carefully select the number of equal increments to ensure that we evaluate enough points to produce a curve smooth enough for display or to provide sufficient data for analysis. It is a good idea to subdivide the parameter range by halves and develop simple expressions for the location of the curve at its midpoint as we demonstrate for Hermite curves (Section 3.5). Another difficulty is that round-off errors accumulate as more points on the curve are computed. Using sufficient precision often mitigates this problem. See Shantz and Chang (1988) for further discussion of this subject.

Effects of Parameterization

Frequently, the direct-point solution for a succession of equal intervals of the parameter on a curve produces unsatisfactory results. Equal parameter values rarely produce equal intervals in model space. For a theoretical discussion of this problem and possible solutions, see Epstein (1976), Hartley and Judd (1978, 1980), and Farin (1993a).

Inverse-Point Solution

Many modeling algorithms require that we find the value of the parametric variable corresponding to a given point. For example, given the x, y, z coordinates of a point on (or very near) a cubic curve in model space, we must find the corresponding value of u, the *inverse-point solution*. This problem implies calculating the roots of three cubic equations. These equations are, of course,

$$x = a_x u^3 + b_x u^2 + c_x u + d_x$$

$$y = a_y u^3 + b_y u^2 + c_y u + d_y$$

$$z = a_z u^3 + b_z u^2 + c_z u + d_z$$

A difficulty arises, though, because it is unlikely that the u values (roots) from each of these three cubic equations will be exactly equal. Computation methods contribute to the potential disparity, and the source and significance of the point coordinates themselves add to it. In most cases, it is good practice to assume, for example, that a given point $\mathbf{q}$ is not on a curve. Instead, we find the point $\mathbf{p}$ on the curve closest to the given point $\mathbf{q}$.

Figure 2.6 Vector interpretation of the inverse-point solution.

We will solve this problem for a single point $\mathbf{q}$ and a parametric cubic curve (see Figure 2.6). We determine the point $\mathbf{p}$ on a curve closest to a point $\mathbf{q}$ in space by finding the vector $(\mathbf{p} - \mathbf{q})$ from the point $\mathbf{q}$ to the curve that is perpendicular to the tangent of the curve at point $\mathbf{p}$. The problem is to find $\mathbf{p}$ such that this condition is satisfied.

For $(\mathbf{p} - \mathbf{q})$ to be perpendicular to the tangent $\mathbf{p}''$ at $\mathbf{p}$, their scalar product must be zero. This means that

$$(\mathbf{p} - \mathbf{q}) \cdot \mathbf{p}'' = 0 \tag{2.35}$$

For cubic curves, this results in a quintic polynomial in u. Various numerical methods are available to solve such an equation. Of course, only roots in the interval $u \in [0,1]$ are of interest. Each root value in this interval determines a point $\mathbf{p}$ for which to compute $|\mathbf{p} - \mathbf{q}|$. The point $\mathbf{p}$ yielding the minimum value for $|\mathbf{p} - \mathbf{q}|$ is the distance between them.

3

HERMITE CURVE

The algebraic description of a curve as a set of parametric cubic polynomial equations and their corresponding vector equation begins this chapter. We see that expressing the algebraic coefficients in terms of boundary conditions leads directly to the more convenient geometric form. The Hermite basis functions emerge as the mathematical link between the algebraic and geometric forms. We then find that both forms are expressed concisely as matrix equations, which prove useful when converting between Hermite and Bézier basis functions as well as in other transformation operations. Varying the magnitude of the tangent vectors offers a way to modify the interior shape of a curve without changing the position of the endpoints or the Cartesian slopes at these points. Truncating and subdividing a curve is accomplished by reparameterizing the basis functions and adjusting certain curve boundary conditions. Additional refinements of the Hermite form allow a curve to interpolate three or four given points. Some conic curves are represented exactly by a Hermite curve, while others are closely approximated. Finally, in this chapter we see that Hermite curve segments joined end-to-end form a composite curve, and if certain continuity conditions are met, it will have the properties of a spline.

3.1 ALGEBRAIC AND GEOMETRIC FORMS

The algebraic form of a parametric cubic curve is given by the following three polynomials:

$$
\begin{aligned}
x(u) &= a_x u^3 + b_x u^2 + c_x u + d_x \\
y(u) &= a_y u^3 + b_y u^2 + c_y u + d_y \\
z(u) &= a_z u^3 + b_z u^2 + c_z u + d_z
\end{aligned}
\tag{3.1}
$$

We usually restrict the parameter u to values in the interval 0 to 1, inclusive, or $u \in [0,1]$. This restriction bounds the curve, creating a curve segment. The 12 coefficients, called *algebraic coefficients*, determine a unique curve, including its size, shape, and position in space. Therefore, two curves of the same shape and size have different algebraic coefficients if they occupy different positions in space.

Writing these three equations in the more compact form of a vector equation is not only less cumbersome to read and write but also allows us to denote an arbitrary number of dimensions. Thus, in vector notation, Equations (3.1) become

$$\mathbf{p}(u) = \mathbf{a}u^3 + \mathbf{b}u^2 + \mathbf{c}u + \mathbf{d} \tag{3.2}$$

where $\mathbf{p}(u)$ is the position vector of any point on the curve and $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ are the vector equivalents of the scalar algebraic coefficients. The components of $\mathbf{p}(u)$ correspond to the Cartesian coordinates of the point.

Equation (3.2) suggests yet an even more simplified and shortened form, using the summation notation:

$$\mathbf{p}(u) = \sum_{i=0}^{3} \mathbf{a}_i u^i$$

Finally, using the Einstein convention and adjusting the indices as necessary produces

$$\mathbf{p}(u) = \mathbf{a}_i u^i \qquad \text{or} \qquad x_j = a_{ij} u^i$$

where the a_{ij} are the algebraic coefficients and x_j is the jth component. This is the *power basis* representation of a curve.

The algebraic coefficients are not the most convenient way of controlling the shape of a curve in typical modeling situations, nor do they contribute much to an intuitive sense of a curve. However, the Hermite form offers a practical alternative, allowing us to define a curve segment in terms of conditions at its endpoints, or boundaries. For a cubic Hermite curve, these conditions are usually the endpoint coordinates and the tangent directions at these points. Using the endpoints $\mathbf{p}(0)$ and $\mathbf{p}(1)$ and the corresponding tangent vectors $\mathbf{p}'(0)$ and $\mathbf{p}'(1)$, we obtain the following four equations:

$$\mathbf{p}(0) = \mathbf{d}$$
$$\mathbf{p}(1) = \mathbf{a} + \mathbf{b} + \mathbf{c} + \mathbf{d}$$
$$\mathbf{p}''(0) = \mathbf{c}$$
$$\mathbf{p}''(1) = 3\mathbf{a} + 2\mathbf{b} + \mathbf{c}$$

where substituting $u = 0$ into Equation (3.2) obtains $\mathbf{p}(0)$, and substituting $u = 1$ into the equation obtains $\mathbf{p}(1)$. Finally, differentiating $\mathbf{p}(u)$ with respect to u yields $\mathbf{p}''(u) = 3\mathbf{a}u^2 + 2\mathbf{b}u + \mathbf{c}$, and substituting $u = 0$ and $u = 1$ into this obtains $\mathbf{p}''(0)$ and $\mathbf{p}''(1)$. Solving this set of four simultaneous equations in four unknowns yields the algebraic coefficients in terms of the boundary conditions:

$$\mathbf{a} = 2\mathbf{p}(0) - 2\mathbf{p}(1) + \mathbf{p}''(0) + \mathbf{p}''(1)$$
$$\mathbf{b} = -3\mathbf{p}(0) + 3\mathbf{p}(1) - 2\mathbf{p}''(0) - \mathbf{p}''(1)$$
$$\mathbf{c} = \mathbf{p}''(0)$$
$$\mathbf{d} = \mathbf{p}(0)$$

Substituting these expressions for the algebraic coefficients into Equation (3.2) and rearranging terms produces:

$$\mathbf{p}(u) = (2u^3 - 3u^2 + 1)\mathbf{p}(0) + (-2u^3 + 3u^2)\mathbf{p}(1) + (u^3 - 2u^2 + u)\mathbf{p}''(0) + (u^3 - u^2)\mathbf{p}''(1) \quad (3.3)$$

We simplify this equation using the following substitutions:

$$\begin{aligned} F_1(u) &= 2u^3 - 3u^2 + 1 \\ F_2(u) &= -2u^3 + 3u^2 \\ F_3(u) &= u^3 - 2u^2 + u \\ F_4(u) &= u^3 - u^2 \end{aligned} \quad (3.4)$$

Now we can rewrite Equation (3.3) as

$$\mathbf{p}(u) = F_1(u)\mathbf{p}(0) + F_2(u)\mathbf{p}(1) + F_3(u)\mathbf{p}''(0) + F_4(u)\mathbf{p}''(1) \quad (3.5)$$

We further simplify this equation by using subscripts to represent the endpoint u values. Equation (3.5) then becomes:

$$\mathbf{p}(u) = F_1(u)\mathbf{p}_0 + F_2(u)\mathbf{p}_1 + F_3(u)\mathbf{p}_0'' + F_4(u)\mathbf{p}_1'' \quad (3.6)$$

This is the geometric form, and the vectors $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_0''$, $\mathbf{p}_1''$ are the *geometric coefficients*. The F terms are the *Hermite basis functions* (see Section 3.2).

There are 12 degrees of freedom (4 vectors × 3 components per vector) that we must specify to define fully and unambiguously the equations of the cubic Hermite curve. As we will see later in more detail, it is useful to specify these shape-controlling variables at the endpoints of a curve, where $u = 0$ and $u = 1$. First, it is computationally easy to evaluate the polynomials at $u = 0$ and $u = 1$. Second, when joining many curves together end-to-end to form more complex curves, we must control what is going on at those ends.

These curves, like all parametric curves, are directional in the sense that values of the parameter u increase monotonically when progressing from point to point along the curve. We will discuss this characteristic in more detail later, but it is important to understand that reversing the sequence of endpoints and tangent vector directions generates an identical curve. The choice of which end of a curve to define as $\mathbf{p}_0$ and which as $\mathbf{p}_1$ is arbitrary unless there is some additional constraint dictating a preferred direction, such as position in a composite string of curves. There are no intrinsic mathematical constraints.

3.2 HERMITE BASIS FUNCTIONS

The Hermite basis functions first appear in the derivation of the geometric form from the algebraic form. They are the functions defined by Equations (3.4). Figure 3.1 shows each function as a curve over the domain of the parameter in the unit interval $u \in [0,1]$.

Three important characteristics of these basis functions are:

1. *Universality*—they hold for all cubic Hermite curves.
2. *Dimensional independence*—they are identical for each of the three model-space coordinates, because they are dependent only on u.
3. *Separation of boundary condition effects*—they allow the constituent boundary condition coefficients to be decoupled from each other.

This means that we can selectively modify a single specific boundary condition to alter the shape of a curve without affecting the other boundary conditions.

These functions blend the effects of the endpoints and tangent vectors to produce the intermediate point coordinate values over the domain of u. For example, we can decompose a curve into its parametric components $x(u), y(u), z(u)$, then decompose each component curve into four *orthogonal* curves corresponding to $F_1(u)\mathbf{p}_0$, $F_2(u)\mathbf{p}_1$, $F_3(u)\mathbf{p}_0'$, and $F_4(u)\mathbf{p}_1'$. (An expla-

Figure 3.1 Cubic Hermite basis functions.

nation of orthogonal curves and functions follows shortly.) Figure 3.2 shows an example where

$$\mathbf{p}_0 = [5 \quad 5 \quad 0]$$

$$\mathbf{p}_1 = [25 \quad 5 \quad 0]$$

$$\mathbf{p}_0'' = [66 \quad 44 \quad 0]$$

$$\mathbf{p}_1'' = [8 \quad -6 \quad 0]$$

The dashed curve shows the effect of multiplying y_0'' by 0.5. Notice how each basis function contributes to the total shape of the curve.

A situation analogous to that described for the position vector $\mathbf{p}(u)$ also holds for the tangent vector and second derivative vector $\mathbf{p}''(u)$ and $\mathbf{p}'''(u)$, respectively, where

$$\mathbf{p}''(u) = \frac{d\mathbf{p}(u)}{du} \qquad \text{and} \qquad \mathbf{p}'''(u) = \frac{d^2\mathbf{p}(u)}{du^2}$$

Figure 3.2 A curve decomposed into four orthogonal curves shaped by the basis functions.

Differentiating Equations (3.4) yields

$$F_1^u(u) = 6u^2 - 6u$$
$$F_2^u(u) = -6u^2 + 6u$$
$$F_3^u(u) = 3u^2 - 4u + 1$$
$$F_4^u(u) = 3u^2 - 2u$$

and

$$F_1^{uu}(u) = 12u - 6$$
$$F_2^{uu}(u) = -12u + 6$$
$$F_3^{uu}(u) = 6u - 4$$
$$F_4^{uu}(u) = 6u - 2$$

We compute the vectors $\mathbf{p}^u(u)$ and $\mathbf{p}^{uu}(u)$ at any point on a curve by using the appropriate set of basis functions operating on the geometric coefficients, so that

$$\mathbf{p}^u(u) = F_1^u(u)\mathbf{p}_0 + F_2^u(u)\mathbf{p}_1 + F_3^u(u)\mathbf{p}_0^u + F_4^u(u)\mathbf{p}_1^u \tag{3.9}$$

and

$$\mathbf{p}^{uu}(u) = F_1^{uu}(u)\mathbf{p}_0 + F_2^{uu}(u)\mathbf{p}_1 + F_3^{uu}(u)\mathbf{p}_0^u + F_4^{uu}(u)\mathbf{p}_1^u \tag{3.10}$$

The basis functions $F_i^u(u)$ and $F_i^{uu}(u)$ are graphed in Figures 3.3 and 3.4, respectively.

The point coefficient functions $F_i(u)$ are orthogonal because they have the property $F_1(u) + F_2(u) = 1$, so that when $F_1(u) = 1$ then $F_2(u) = 0$, and when $F_1(u) = 0$ then $F_2(u) = 1$. (The term *orthogonal* is borrowed to describe this condition.) Furthermore, the tangent vector coefficients have the property $F_3(u) = F_4(u) = 0$ at $u = 0$ and $u = 1$. This means that only $\mathbf{p}_0$ contributes to the curve position at $u = 0$ and only $\mathbf{p}_1$ contributes at $u = 1$, as expected.

The orthogonality of the $F_i^u(u)$ coefficients requires that $F_1^u(u) = F_2^u(u) = F_4^u(u) = 0$ and $F_3^u(u) = 1$ and that $F_1^u(u) = F_2^u(u) = F_3^u(u) = 0$ and $F_4^u(u) = 1$ at $u = 1$. The orthogonality of the $F_i^{uu}(u)$ cannot be demonstrated, because these coefficients blend the standard boundary conditions $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_0^u$, and $\mathbf{p}_1^u$. Instead we would have to derive other basis functions, say $H_i^{uu}(u)$, that blend $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_0^{uu}$, and $\mathbf{p}_1^{uu}$. Here the orthogonality of the $H_i^{uu}(u)$ requires that $H_1^{uu}(u) = H_2^{uu}(u) = H_4^{uu}(u) = 0$ and $H_3^{uu}(u) = 1$ at $u = 0$, and that $H_1^{uu}(u) = H_2^{uu}(u) = H_3^{uu}(u) = 0$ and $H_4^{uu}(u) = 1$ at $u = 1$. This is the second derivative form.

The symmetry between related pairs of basis function curves, for example $F_1(u)$ and $F_2(u)$, or $F_3(u)$ and $F_4(u)$, allows us to reverse the direction of parameterization and preserve a curve's shape, provided that the endpoints are interchanged and the tangent vector directions reversed. There are similar basis functions for other odd-degree Hermite polynomials, such as the quintic. Even-degree polynomial basis functions are seldom used, because they have an odd number of degrees of freedom. This prohibits establishing symmetrical boundary conditions at the curve segment endpoints.

Using the summation notation, we generalize Equation (3.6) as follows:

$$\mathbf{p}(u) = \sum_{i=0}^{n} \mathbf{b}_i F_{i,n}(u) \tag{3.11}$$

where n is the degree of the polynomial and the $\mathbf{b}_i$ are the geometric coefficient vectors. For $n = 3$, we have $\mathbf{b}_0 = \mathbf{p}_0$, $\mathbf{b}_1 = \mathbf{p}_1$, $\mathbf{b}_2 = \mathbf{p}_0^u$, and $\mathbf{b}_3 = \mathbf{p}_1^u$. Shifting the range of the index i down one step makes it conform to the common notation schemes of Bézier and B-Spline basis functions. For example, $F_1(u)$ of the cubic Hermite basis functions becomes $F_{0,3}(u)$.

Figure 3.3 First derivative basis function curves.

For the quintic Hermite curve, $n = 5$ and

$$\mathbf{p}(u) = \mathbf{b}_0 F_{0,5}(u) + \mathbf{b}_1 F_{1,5}(u) + \mathbf{b}_2 F_{2,5}(u) + \mathbf{b}_3 F_{3,5}(u) + \mathbf{b}_4 F_{4,5}(u) + \mathbf{b}_5 F_{5,5}(u) \tag{3.12}$$

where, for example, we might make the following assignments:

$$\mathbf{b}_0 = \mathbf{p}_0$$

$$\mathbf{b}_1 = \mathbf{p}_1$$

$$\mathbf{b}_2 = \mathbf{p}_0^u$$

Figure 3.4 Second derivative basis function curves.

$$\mathbf{b}_3 = \mathbf{p}_1^u$$

$$\mathbf{b}_4 = \mathbf{p}_0^{uu}$$

$$\mathbf{b}_5 = \mathbf{p}_1^{uu}$$

Clearly, the order of the six terms on the right side of Equation (3.12) is not important. However, care is required in constructing the $\mathbf{b}_i$ and $F_{i,n}(u)$ sets, so that their terms are appropriately linked by a common index.

These basis functions are a powerful modeling device. Using them, we can calculate the coordinates and the first and second derivatives of any point on any cubic Hermite curve. We will see similar basis functions when we study Bézier and B-Spline curves, and, in fact, under certain conditions we can convert from one basis to another.

3.3 MATRIX FORM

Matrix algebra and its notation scheme offer a compact and powerful mathematical form for representing a curve. Otherwise complex geometric oper-

ations, transformations, and analysis often become simple matrix operations. We can rewrite Equation (3.2) as the product of two matrices:

$$\mathbf{p}(u) = [u^3 \quad u^2 \quad u \quad 1]\,[\mathbf{a} \quad \mathbf{b} \quad \mathbf{c} \quad \mathbf{d}]^T \tag{3.13}$$

Letting

$$\mathbf{p}(u) = [x(u) \quad y(u) \quad z(u)]^T$$
$$\mathbf{U} = [u^3 \quad u^2 \quad u \quad 1]$$
$$\mathbf{A} = [\mathbf{a} \quad \mathbf{b} \quad \mathbf{c} \quad \mathbf{d}]^T$$

we can then rewrite Equation (3.13) as

$$\mathbf{p}(u) = \mathbf{UA} \tag{3.14}$$

and similarly for the geometric form

$$\mathbf{p}(u) = [F_1(u) \quad F_2(u) \quad F_3(u) \quad F_4(u)][\mathbf{p}_0 \quad \mathbf{p}_1 \quad \mathbf{p}_0^u \quad \mathbf{p}_1^u]^T \tag{3.15}$$

If we now let

$$\mathbf{F} = [F_1(u) \quad F_2(u) \quad F_3(u) \quad F_4(u)]$$

and

$$\mathbf{B} = [\mathbf{p}_0 \quad \mathbf{p}_1 \quad \mathbf{p}_0^u \quad \mathbf{p}_1^u]^T$$

then we can rewrite Equation (3.15) as

$$\mathbf{P} = \mathbf{FB} \tag{3.16}$$

Given that **A** is the matrix of algebraic coefficients and **B** is the Hermite geometry matrix containing the geometric coefficients, then using matrix algebra we can develop a relationship between the algebraic and geometric forms. From Equation (3.4), we obtain

$$\mathbf{F} = [(2u^3 - 3u^2 + 1) \quad (-2u^3 + 3u^2) \quad (u^3 - 2u^2 + u) \quad (u^3 - u^2)] \tag{3.17}$$

By inspection, we see that we can express the right side of this equation as

$$\mathbf{F} = [u^3 \quad u^2 \quad u \quad 1]\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.18}$$

The 4 × 4 matrix is the *Hermite basis transformation matrix,* which we will denote by $\mathbf{M}_F$. This matrix controls the basis transformation between the Hermite basis polynomials $F_i(u)$ and the monomial basis u^i or **U**, so that

$$\mathbf{F} = \mathbf{UM}_F \tag{3.19}$$

Substituting Equation (3.19) for (3.16) produces

$$\mathbf{p}(u) = \mathbf{U}\mathbf{M}_F\mathbf{B} \tag{3.20}$$

This means that

$$\mathbf{A} = \mathbf{M}_F\mathbf{B} \tag{3.21}$$

and, conversely, that

$$\mathbf{B} = \mathbf{M}_F^{-1}\mathbf{A} \tag{3.22}$$

where

$$\mathbf{M}_F^{-1} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \tag{3.23}$$

We see that Equations (3.21) and (3.22) allow ready conversion between algebraic and geometric forms, and also that matrices $\mathbf{U}$, $\mathbf{F}$, and $\mathbf{M}_F$ are identical for all cubic Hermite curves. Only the $\mathbf{A}$ and $\mathbf{B}$ matrices vary from curve to curve, depending on shape and position. This means that it is convenient and efficient to denote a specific curve by simply giving its matrix of algebraic or geometric coefficients.

For a given polynomial degree, different species of basis functions are interchangeable through a matrix transformation. For example, we let $\mathbf{B}_F$ represent a matrix of control points defining a curve $\mathbf{p}(u) = \mathbf{U}\mathbf{M}_F\mathbf{B}_F$ using basis functions $\mathbf{F} = \mathbf{U}\mathbf{M}_F$. Then, for another system of basis functions we let $\mathbf{G} = \mathbf{U}\mathbf{M}_G$ and find a set of control points $\mathbf{B}_G$ that reproduces the same curve. This means it must be true that $\mathbf{U}\mathbf{M}_G\mathbf{B}_G = \mathbf{U}\mathbf{M}_F\mathbf{B}_F$, so that $\mathbf{B}_G = \mathbf{M}_G^{-1}\mathbf{M}_F\mathbf{B}_F$. Later, we will see that under certain conditions we can convert between Hermite and Bézier curves using this approach.

Rewriting Equations (3.9) and (3.10) in the more compact matrix form yields

$$\mathbf{p}^{u}(u) = \mathbf{F}^{u}\mathbf{B} \qquad \text{or} \qquad \mathbf{p}^{u}(u) = \mathbf{U}\mathbf{M}_F^{u}\mathbf{B}$$

and

$$\mathbf{p}^{uu}(u) = \mathbf{F}^{uu}\mathbf{B} \qquad \text{or} \qquad \mathbf{p}^{uu}(u) = \mathbf{U}\mathbf{M}_F^{uu}\mathbf{B}$$

where

$$\mathbf{F}^{u} = [(6u^2 - 6u) \quad (-6u^2 + 6u) \quad (3u^2 - 4u + 1) \quad (3u^2 - 2u)]$$

and

$$\mathbf{F}^{uu} = [(12u - 6) \quad (-12u + 6) \quad (6u - 4) \quad (6u - 2)]$$

The basis matrices $\mathbf{M}_F^u$ and $\mathbf{M}_F^{uu}$ are

$$\mathbf{M}_F^u = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 6 & -6 & 3 & 3 \\ -6 & 6 & -4 & -2 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{M}_F^{uu} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 12 & -12 & 6 & 6 \\ -6 & 6 & -4 & -2 \end{bmatrix}$$

3.4 TANGENT VECTORS

The tangent vector at any point on a curve is given by the first derivative of the parametric equations defining the curve. The magnitude or length of the tangent vector depends on the parameterization and affects the interior shape of the curve.

Here is how tangent vectors work to define the shape of a curve. Specifying the coordinates and slopes at endpoints of a cubic Hermite curve accounts for only 10 of the 12 degrees of freedom implied by Equations (3.1). Six are supplied by the coordinates x_0, y_0, z_0 and x_1, y_1, z_1, directly from the components of $\mathbf{p}_0$ and $\mathbf{p}_1$. Four more are supplied by the direction cosines, two from each end. Remember that there are only two independent direction cosines at any point on a three-dimensional curve. The third direction cosine is a function of the other two, because the sum of the squares of the three direction cosines equals one. This means that there are two more degrees of freedom available to control the shape of a curve.

The slopes represented by tangent vectors $\mathbf{p}_0^u$ and $\mathbf{p}_1^u$ are those in parameter space. The components are $x_0^u = dx(0)/du$ and $x_1^u = dx(1)/du$, and similarly for y and z. These are directly related to the Euclidean slopes $y^x = dy/dx$ and $z^x = dz/dx$ from $y^x = y^u/x^u$ and $z^x = z^u/x^u$.

The tangent vector resolves the problem of the extra two degrees of freedom in the following way: A tangent vector operates at each end of the cubic Hermite curve, as shown in Figure 3.5. Not only do these vectors define the slopes at each end, but each vector also has a magnitude associated with it. The magnitudes supply the eleventh and twelfth degrees of freedom and contribute to the control of the shape of a curve's interior.

The direction cosines of the tangent to a point u on a curve comprise the components of a unit vector $\hat{\mathbf{t}}(u)$, where $\hat{\mathbf{t}}(u) = \mathbf{p}^u(u)/|\mathbf{p}^u(u)|$, where the direction cosines are $t_x(u), t_y(u), t_z(u)$. From elementary geometry, we know that

$$|\hat{\mathbf{t}}(u)| = \sqrt{\hat{t}_x^2(u) + \hat{t}_y^2(u) + \hat{t}_z^2(u)} = 1$$

where $\hat{t}_x(u), \hat{t}_y(u), \hat{t}_z(u) \in [0,1]$. Furthermore, if $k = |\mathbf{p}^u(u)|$, then $\mathbf{p}^u(u) = k\hat{\mathbf{t}}(u)$, where k represents the magnitude of $\mathbf{p}^u(u)$. We add subscripts to indicate a specific point, corresponding to its u value, giving the tangent vectors for the

Figure 3.5 Tangent vectors.

two endpoints of a curve as $\mathbf{p}_0^u = k_0\hat{\mathbf{t}}_0$ and $\mathbf{p}_1^u = k_1\hat{\mathbf{t}}_1$. Because the three direction cosines are related by $|\hat{\mathbf{t}}(u)| \equiv 1$, only two are independent variables. So now we can express the matrix of geometric coefficients as

$$\mathbf{B} = [\mathbf{p}_0 \quad \mathbf{p}_1 \quad k_0\hat{\mathbf{t}}_0 \quad k_1\hat{\mathbf{t}}_1]^T \tag{3.24}$$

This is a very interesting state of affairs, because we see that it is a property of cubic Hermite curves that we can obtain many different curves (in fact, an infinite family of them), all of which have the same endpoints and slopes yet entirely different interior shapes, depending on the tangent vector magnitudes k_0 and k_1. By varying k_0 and k_1, we can control the curvature at each end, or we can fix the location of some intermediate point. Figure 3.6 shows the effect of varying k_0 and k_1 while holding constant all the other geometric coefficients. The coefficients have the following values:

$$\begin{array}{lll} x_0 = 4.0 & y_0 = 4.0 & z_0 = 0 \\ x_1 = 24.0 & y_1 = 4.0 & z_1 = 0 \\ \hat{t}_{0x} = 0.8320 & \hat{t}_{0y} = 0.5547 & \hat{t}_{0z} = 0 \\ \hat{t}_{1x} = 0.8320 & \hat{t}_{1y} = -0.5547 & \hat{t}_{1z} = 0 \end{array}$$

Figure 3.6 Effect of tangent vector magnitude on curve shape. (a) Symmetrical slopes and equal tangent vector magnitudes. (b) Symmetrical slopes and unequal tangent vector magnitudes.

The curves in Figure 3.6a are symmetrical around a common vertical axis, and their respective values of k_0 and k_1 are equal. In Figure 3.6b, k_0 is varied while k_1 is constant, which results in the asymmetrical family of curves. Usually, when we model a curve we must meet some specific design criteria. However, the illustrations here are meant to explain curve behavior and do not represent particular examples of applications to real design problems.

When the magnitudes of the tangent vectors exceed some multiple of the chord length between the endpoints of the curve segment, the curve begins to exhibit undesirable characteristics, such as loops and cusps. Conversely, very low magnitudes produce a relatively flat curve. The chord length is $|\mathbf{p}_1 - \mathbf{p}_0| = 20$ for each curve in Figure 3.6, and a loop is produced in the curve when $k_0 = k_1 = 80$ (that is, four times the chord length).

At first it appears that by manipulating magnitudes of $\mathbf{p}_0''$ and $\mathbf{p}_1''$ we can force points on the curve to any location. But let us see what happens to the point corresponding to $u = 0.5$. The values of the basis functions at this point are $F_1(0.5) = 0.5$, $F_2(0.5) = 0.5$, $F_3(0.5) = 0.125$, and $F_4(0.5) = -0.125$, so that

$$\mathbf{p}(0.5) = 0.5(\mathbf{p}_0 + \mathbf{p}_1) + 0.125(\mathbf{p}_0'' - \mathbf{p}_1'')$$

Substituting $\mathbf{p}_0'' = k_0\hat{\mathbf{t}}_0$ and $\mathbf{p}_1'' = k_1\hat{\mathbf{t}}_1$, we obtain

$$\mathbf{p}(0.5) = 0.5(\mathbf{p}_0 + \mathbf{p}_1) + 0.125(k_0\hat{\mathbf{t}}_0 - k_1\hat{\mathbf{t}}_1)$$

The curves shown in Figure 3.7 differ only in the values of k_0 and k_1. In two of the curves, g and h, the direction of $\mathbf{p}_1''$ is reversed by using negative values for k_1. By manipulating the magnitudes and signs of $\mathbf{p}_0''$ and $\mathbf{p}_1''$ while preserving their lines of action, we are able to force $\mathbf{p}(0.5)$ to any location. This is true for any plane curve whether or not it lies in a plane coincident with, or parallel to, a principal plane of the coordinate system. There is, however, a restriction on curve shape readily apparent only in nonplanar curves. If the magnitudes (including signs) of the tangent vectors are varied, then $\mathbf{p}(0.5)$ will always lie in a plane that is parallel to both $\mathbf{p}_0''$ and $\mathbf{p}_1''$ and that passes through the midpoint of the line joining $\mathbf{p}_0$ and $\mathbf{p}_1$ (i.e., $0.5(\mathbf{p}_0 + \mathbf{p}_1)$). Here is a casual proof. Treating k_0 and k_1 as independent variables, we write the equation for $\mathbf{p}(0.5)$ as follows:

$$\mathbf{p}_{0.5}(k_0,k_1) = 0.5(\mathbf{p}_0 + \mathbf{p}_1) + (0.125\hat{\mathbf{t}}_0)k_0 - (0.125\hat{\mathbf{t}}_1)k_1$$

This is the vector equation of a plane through the point $0.5(\mathbf{p}_0 + \mathbf{p}_1)$ and containing $\hat{\mathbf{t}}_0$ and $\hat{\mathbf{t}}_1$. Of course, a similar restriction applies to any other given point $\mathbf{p}(u_i)$ on the curve.

Figure 3.7 Effect of tangent vector magnitude on the location of p(0.5).

3.5 TRUNCATING AND SUBDIVIDING

We can break a curve down into smaller segments by *truncating* or *subdividing* it. There are many reasons for doing this. For example, we may truncate to isolate and extract that part of a curve surviving a model modification process, or subdivide it to compute points for displaying it. To truncate, subdivide, or change the direction of parameterization of a curve ordinarily requires a mathematical operation called *reparameterization*. Ideally, this operation produces a change in the parametric interval so that neither the shape nor the position of the curve is changed. This effect is often referred to as *shape invariance under parameterization and reparameterization*. The function $v = f(u)$ describes the precise way this interval is changed. For example, reversing the direction of parameterization is the simplest form of reparameterization. To do this we use $v = -u$, where v is the new parametric variable. This transformation does not change the shape or position of the curve, nor does it change any of the curve's analytic properties.

Figure 3.8 illustrates a Hermite curve segment, showing its endpoints and tangent vectors for the two possible directions of parameterization. The

transformation from the condition illustrated in Figure 3.8a into that in 3.8b requires interchanging the position of $\mathbf{p}_0$ and $\mathbf{p}_1$ in the matrix of geometric coefficients and similarly interchanging the tangent vectors and reversing their signs. So, given the geometry matrix $\mathbf{B} = [\mathbf{p}_0 \quad \mathbf{p}_1 \quad \mathbf{p}_0^u \quad \mathbf{p}_1^u]^T$ before reversing the direction of parameterization, after reversing it becomes $\mathbf{B}'$, where $\mathbf{B}' = [\mathbf{p}_1 \quad \mathbf{p}_0 \quad -\mathbf{p}_1^u \quad -\mathbf{p}_0^u]^T$.

Figure 3.9 shows a more general form of reparameterization. Here we see two possible parameterizations for a single curve, where the curve is initially parameterized from u_i to u_j. To change this so that the parametric variable ranges from v_i to v_j, we let the geometric coefficients in the first case be $\mathbf{B}_1 = [\mathbf{p}_i \quad \mathbf{p}_j \quad \mathbf{p}_i^u \quad \mathbf{p}_j^u]^T$, and in the second case $\mathbf{B}_2 = [\mathbf{q}_i \quad \mathbf{q}_j \quad \mathbf{q}_i^v \quad \mathbf{q}_j^v]^T$. The endpoint coordinates are related in a straightforward way. They must be invariant to any change of parameterization so that $\mathbf{q}_i = \mathbf{p}_i$ and $\mathbf{q}_j = \mathbf{p}_j$; otherwise, the reparameterized curve does not satisfy the constant position requirement. The tangent vectors are another matter. Because they are defined by the first derivative of the parametric functions, they are sensitive to the function relating u and v, that is, $v = f(u)$. A linear relationship is required to pre-

Figure 3.8 Two possible directions of parameterization of a curve.

Figure 3.9 Reparameterization.

serve the cubic form of the parametric equations and the direction of the tangent vector; so again we must insist that $v = au + b$. This means that $dv = adu$. Because we know that $v_i = au_i + b$ and $v_j = au_j + b$, we can easily find a and b and, subsequently, the relationship between the tangent vectors, which is simply

$$\mathbf{q}^v = \frac{u_j - u_i}{v_j - v_i}\,\mathbf{p}^u \tag{3.25}$$

Now we can state the complete relationship between the two sets of geometric coefficients as

$$\begin{aligned}
\mathbf{q}_i &= \mathbf{p}_i \\
\mathbf{q}_j &= \mathbf{p}_j \\
\mathbf{q}_i^v &= \frac{u_j - u_i}{v_j - v_i}\,\mathbf{p}_i^u \\
\mathbf{q}_j^v &= \frac{u_j - u_i}{v_j - v_i}\,\mathbf{p}_j^u
\end{aligned} \tag{3.26}$$

This result tells us that the tangent vector magnitudes must change to accommodate a change in the range of the parametric variable. We see that these magnitudes are simply scaled by the ratio of the ranges of the parametric variables. This preserves the directions of the tangent vectors and the shape of the curve.

If u_i and u_j are successive pairs of integers, then $u_j - u_i = 1$, and the same holds true for v_i and v_j. This is very useful when dealing with segments of composite curves, because individual segments originally parameterized to the interval $u_i = 0$ and $u_j = 1$ may be cardinally reparameterized to allow relatively simple curve segment identifying schemes. For example, the nth curve

segment can be parameterized from $u_i = n - 1$ to $u_j = n$, with the result that $u_j - u_i = 1$. This means that the **B** matrix for the segment is unchanged by this form of reparameterization. In fact, there are many different parameterizations that describe the same curve.

These reparameterization formulas allow us to compute a new **B** matrix for a truncated or extended curve. Later sections that discuss composite curves and solid-modeling techniques will demonstrate the need for these capabilities as modeling elements are intersected, trimmed, and joined together to form more complex objects. Figure 3.10 illustrates a curve truncated at u_i and u_j. That is, the segments from u_0 to u_i and from u_j to u_1 are eliminated. We can represent the remaining segment as a cubic Hermite curve, parameterized from $v = 0$ to $v = 1$, by proceeding as follows: We compute $\mathbf{p}_i$ and $\mathbf{p}_j$ using $\mathbf{p}(u) = \mathbf{U}\mathbf{M}_F\mathbf{B}$, and $\mathbf{p}_i''$ and $\mathbf{p}_j''$ using $\mathbf{p}''(u) = \mathbf{U}\mathbf{M}_F''\mathbf{B}$. The ratio of parametric interval lengths $(u_j - u_i)/(v_j - v_i)$ given by Equation (3.26) reduces to $u_j - u_i$, because $v_j - v_i = 1$. If $\mathbf{q}(v)$ represents the truncated curve, then

$$\begin{aligned} \mathbf{q}_0 &= \mathbf{p}_i \\ \mathbf{q}_1 &= \mathbf{p}_j \\ \mathbf{q}_0'' &= (u_j - u_i)\mathbf{p}_i'' \\ \mathbf{q}_1'' &= (u_j - u_i)\mathbf{p}_j'' \end{aligned} \tag{3.27}$$

There is a generalization of the preceding parametric transformation. It applies to any polynomial or rational polynomial parametric curve. For this, we again require a linear relationship between initial and transformed parametric variables to preserve the degree of the polynomial, so that $v = au + b$. If $v_i = 0$ and $v_j = 1$ at u_i and u_j, respectively, then by solving for a and b we have

$$u = u_i + (u_j - u_i)v$$

or

$$u = u_i + \Delta u_i v$$

Figure 3.10 Reparameterization of a truncated curve.

where $\Delta u_i = u_j - u_i$. If $v_i \neq 0$ and/or $v_j \neq 1$, then

$$u = \left(\frac{u_i v_j - u_j v_i}{v_j - v_i}\right) + \left(\frac{u_j - u_i}{v_j - v_i}\right) v$$

Applying the binomial theorem to $u = u_i + \Delta u_i v$, the transformation equations are

$$u^n = \sum_{k=0}^{n} \binom{n}{k} u_i^k (\Delta u_i v)^{n-k} \tag{3.28}$$

where

$$\binom{n}{k} = \frac{n!}{k!(n-k)!},$$

the binomial coefficient. We extract a transformation matrix **T** from Equation (3.28) so that

$$\mathbf{U} = \mathbf{VT}$$

For a cubic Hermite curve, we have $\mathbf{p}(u) = \mathbf{UM}_F\mathbf{B}$. Applying the parametric transformation **T** produces the transformed geometric coefficients $\mathbf{B}'$, corresponding to the curve segment between u_i and u_j in the initial parameterization scheme and subsequently renormalized to the unit interval, so that

$$\mathbf{p}(u) = \mathbf{VTM}_F\mathbf{B}$$

$$\mathbf{TM}_F\mathbf{B} = \mathbf{M}_F\mathbf{B}'$$

$$\mathbf{B}' = \mathbf{M}_F^{-1}\mathbf{TM}_F\mathbf{B}$$

where

$$\mathbf{U} = [u^3 \quad u^2 \quad u \quad 1]$$

$$\mathbf{V} = [v^3 \quad v^2 \quad v \quad 1]$$

$$\mathbf{T} = \begin{bmatrix} \Delta u_i^3 & 0 & 0 & 0 \\ 3u_i\Delta u_i^2 & \Delta u_i^2 & 0 & 0 \\ 3u_i^2\Delta u_i & 2u_i\Delta u_i & \Delta u_i & 0 \\ u_i^3 & u_i^2 & u_i & 1 \end{bmatrix}$$

To subdivide a curve into n successive segments of arbitrary length and generate n new curves, we assume that u_i, $\mathbf{p}_i$, and $\mathbf{p}_i'$ at the segment boundaries are given or readily computed. Then, the geometric coefficients of **B** matrix elements of the ith segment are

$$\mathbf{B}_i = [\mathbf{p}_{i-1} \quad \mathbf{p}_i \quad (u_i - u_{i-1})\mathbf{p}_{i-1}' \quad (u_i - u_{i-1})\mathbf{p}_i']^T \tag{3.29}$$

This is clearly an application of Equation (3.27), where $v_j - v_{j-1} = 1$.

If a curve is divided into n equal segments (that is, segment boundaries occur at equal intervals of the parametric variable), then the $\mathbf{B}$ matrix of the ith segment is

$$\mathbf{B}_i = \begin{bmatrix} \mathbf{p}_{(i-1)/n} & \mathbf{p}_{i/n} & \frac{1}{n}\mathbf{p}^u_{(i-1)/n} & \frac{1}{n}\mathbf{p}^u_{i/n} \end{bmatrix}^T \tag{3.30}$$

Of course, we must evaluate each

$$\mathbf{p}_{i/n} \qquad \text{and} \qquad \frac{1}{n}\mathbf{p}^u_{i/n}$$

using $\mathbf{p}(u) = \mathbf{U}\mathbf{M}_F\mathbf{B}$ and $\mathbf{p}^u(u) = \mathbf{U}\mathbf{M}_F^u\mathbf{B}$, respectively.

A quick subdivision is possible by taking advantage of curve properties at $\mathbf{p}(0.5)$. We recursively subdivide in order to generate a set of points on the curve at 2^n equal parametric intervals, each of length $1/2^n$. Given $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}^u_0$, and $\mathbf{p}^u_1$, we procced as follows:

for $n = 1$

$$\mathbf{p}\left(\frac{1}{2}\right) = \frac{1}{2}[\mathbf{p}(0) + \mathbf{p}(1)] + \frac{1}{8}[\mathbf{p}^u(0) - \mathbf{p}^u(1)]$$

for $n = 2$

$$\mathbf{p}\left(\frac{1}{4}\right) = \frac{1}{2}\left[\mathbf{p}(0) + \mathbf{p}\left(\frac{1}{2}\right)\right] + \frac{1}{16}\left[\mathbf{p}^u(0) - \mathbf{p}^u\left(\frac{1}{2}\right)\right]$$

$$\mathbf{p}\left(\frac{3}{4}\right) = \frac{1}{2}\left[\mathbf{p}\left(\frac{1}{2}\right) + \mathbf{p}(1)\right] + \frac{1}{16}\left[\mathbf{p}^u\left(\frac{1}{2}\right) - \mathbf{p}^u(1)\right]$$

for $n = 3$

$$\mathbf{p}\left(\frac{1}{8}\right) = \frac{1}{2}\left[\mathbf{p}(0) + \mathbf{p}\left(\frac{1}{4}\right)\right] + \frac{1}{16}\left[\mathbf{p}^u(0) - \mathbf{p}^u\left(\frac{1}{4}\right)\right]$$

$$\mathbf{p}\left(\frac{3}{8}\right) = \frac{1}{2}\left[\mathbf{p}\left(\frac{1}{4}\right) + \mathbf{p}\left(\frac{1}{2}\right)\right] + \frac{1}{16}\left[\mathbf{p}^u\left(\frac{1}{4}\right) - \mathbf{p}^u\left(\frac{1}{2}\right)\right]$$

$$\mathbf{p}\left(\frac{5}{8}\right) = \frac{1}{2}\left[\mathbf{p}\left(\frac{1}{2}\right) + \mathbf{p}\left(\frac{3}{4}\right)\right] + \frac{1}{16}\left[\mathbf{p}^u\left(\frac{1}{2}\right) - \mathbf{p}^u\left(\frac{3}{4}\right)\right]$$

$$\mathbf{p}\left(\frac{7}{8}\right) = \frac{1}{2}\left[\mathbf{p}\left(\frac{3}{4}\right) + \mathbf{p}(1)\right] + \frac{1}{16}\left[\mathbf{p}^u\left(\frac{3}{4}\right) - \mathbf{p}^u(1)\right]$$

. . . and so on. We let $m = 1, n$ and $i = 1, 2^n - 1$ (incrementing by 2), and compute

$$\mathbf{p}\left(\frac{i}{2^m}\right) = \frac{1}{2}\left[\mathbf{p}\left(\frac{i-1}{2^m}\right) + \mathbf{p}\left(\frac{i+1}{2^m}\right)\right] + \frac{1}{16}\left[\mathbf{p}^u\left(\frac{i-1}{2^m}\right) - \mathbf{p}^u\left(\frac{i+1}{2^m}\right)\right] \tag{3.31}$$

If $m \neq n$, then we must compute tangent vectors at $u = i/2^m$, so that

$$\mathbf{p}^u\left(\frac{i}{2^m}\right) = \frac{3}{2}\left[\mathbf{p}\left(\frac{i+1}{2^m}\right) - \mathbf{p}\left(\frac{i-1}{2^m}\right)\right] + \frac{1}{4}\left[\mathbf{p}^u\left(\frac{i-1}{2^m}\right) + \mathbf{p}^u\left(\frac{i+1}{2^m}\right)\right] \quad (3.32)$$

We renormalize the intervals of length $i/2^{m+1}$, which requires the replacement:

$$\mathbf{p}^u\left(\frac{i}{2^m}\right) \leftarrow \frac{1}{2}\mathbf{p}^u\left(\frac{i}{2^m}\right)$$

Reparameterization back to the unit interval introduces the factor of 1/2 applied to the tangent vectors, so that

$$\mathbf{p}^u \leftarrow \frac{1}{2}\mathbf{p}^u$$

It is this quick and easy renormalizing of the new intervals after each subdivision that makes this work.

Ball (1984), Farin (1993a), and many others offer additional discussion on the effects of parameterization and reparameterization on curve shape.

3.6 THREE-POINT INTERPOLATION

There is a three-point interpolation method for defining a cubic Hermite curve given the following conditions: the two endpoints $\mathbf{p}_0$ and $\mathbf{p}_1$, an intermediate point $\mathbf{p}_i$ and its corresponding but unspecified parametric variable u_i, and the unit tangent vectors $\hat{\mathbf{t}}_0$ and $\hat{\mathbf{t}}_1$. The unspecified parametric variable u_i prevents the problem from being over-constrained. To determine the tangent vectors $\mathbf{p}_0^u$ and $\mathbf{p}_1^u$, we let $\mathbf{p}_0^u = k_0\hat{\mathbf{t}}_0$ and $\mathbf{p}_1^u = k_1\hat{\mathbf{t}}_1$ and attempt to find k_0 and k_1 such that the curve passes through $\mathbf{p}_i$. We have, from our earlier work,

$$\mathbf{p}(u_i) = F_1(u_i)\mathbf{p}_0 + F_2(u_i)\mathbf{p}_1 + k_0F_3(u_i)\hat{\mathbf{t}}_0 + k_1F_4(u_i)\hat{\mathbf{t}}_1$$

Because $F_1 = 1 - F_2$, we obtain

$$\mathbf{p}(u_i) - \mathbf{p}_0 = F_2(u_i)(\mathbf{p}_1 - \mathbf{p}_0) + k_0F_3(u_i)\hat{\mathbf{t}}_0 + k_1F_4(u_i)\hat{\mathbf{t}}_1$$

Expanding this equation in terms of each of the coordinates yields

$$\begin{aligned} x_i - x_0 &= F_2(u_i)(x_1 - x_0) + k_0F_3(u_i)\hat{t}_{x0} + k_1F_4(u_i)\hat{t}_{x1} \\ y_i - y_0 &= F_2(u_i)(y_1 - y_0) + k_0F_3(u_i)\hat{t}_{y0} + k_1F_4(u_i)\hat{t}_{y1} \\ z_i - z_0 &= F_2(u_i)(z_1 - z_0) + k_0F_3(u_i)\hat{t}_{z0} + k_1F_4(u_i)\hat{t}_{z1} \end{aligned} \quad (3.33)$$

Solving the first two of these equations in terms of $k_0F_3(u_i)$ and $k_1F_4(u_i)$ and using matrix notation produces

$$\begin{bmatrix} k_0F_3(u_i) \\ k_1F_4(u_i) \end{bmatrix} = \begin{bmatrix} \hat{t}_{x0} & \hat{t}_{x1} \\ \hat{t}_{y0} & \hat{t}_{y1} \end{bmatrix}^{-1} \left\{ \begin{bmatrix} (x_i - x_0) \\ (y_i - y_0) \end{bmatrix} - \begin{bmatrix} (x_1 - x_0) \\ (y_1 - y_0) \end{bmatrix} F_2(u_i) \right\}$$

From the third equation, we have

$$[z_i - z_0] = [z_1 - z_0]F_2(u_i) + [\hat{t}_{z0} \;\; \hat{t}_{z1}][k_0F_3(u_i) \;\; k_1F_4(u_i)]^T$$

Substituting appropriately produces

$$[z_i - z_0] = [z_1 - z_0]F_2(u_i) + [\hat{t}_{z0} \;\; \hat{t}_{z1}] \begin{bmatrix} \hat{t}_{x0} & \hat{t}_{x1} \\ \hat{t}_{y0} & \hat{t}_{y1} \end{bmatrix}^{-1} \left\{ \begin{bmatrix} x_i - x_0 \\ y_i - y_0 \end{bmatrix} - \begin{bmatrix} x_1 - x_0 \\ y_1 - y_0 \end{bmatrix} F_2(u_i) \right\}$$

leaving only u_i as an unknown.

After working through some rather tedious, though straightforward, matrix algebra, we finally obtain

$$\begin{vmatrix} (x_1 - x_0) & (y_1 - y_0) & (z_1 - z_0) \\ \hat{t}_{x0} & \hat{t}_{y0} & \hat{t}_{z0} \\ \hat{t}_{x1} & \hat{t}_{y1} & \hat{t}_{z1} \end{vmatrix} F_2(u_i) = \begin{vmatrix} (x_i - x_0) & (y_i - y_0) & (z_i - z_0) \\ \hat{t}_{x0} & \hat{t}_{y0} & \hat{t}_{z0} \\ \hat{t}_{x1} & \hat{t}_{y1} & \hat{t}_{z1} \end{vmatrix}$$

where $F_2(u_i) = -2u_i^3 + 3u_i^2$ [from Equation (3.4)] and the initial constraints are expressed in determinant arrays. This is a key equation in the development.

When we determine the value of u_i by solving the indicated cubic equation, we can compute the values of k_0 and k_1 using two equations from, say, Equations (3.33). Then we readily determine $\mathbf{p}_0''$ and $\mathbf{p}_1''$. Note that an acceptable solution of the cubic equation must satisfy these constraints:

$$0 < u_i < 1$$

$$k_0 > 0$$

$$k_1 > 0$$

We can relax the conditions on k_0 and k_1 if we don't care which direction the tangent vectors point along their respective lines of action.

3.7 FOUR-POINT INTERPOLATION

A cubic Hermite curve that passes through four given points is a derivative of the geometric form, and under certain conditions it is comparable to it in usefulness. Figure 3.11 illustrates this situation.

We begin by specifying four distinct points in space $[\mathbf{p}_1 \;\; \mathbf{p}_2 \;\; \mathbf{p}_3 \;\; \mathbf{p}_4]$, assigning to each a successive u value, so that $u_1 < u_2 < u_3 < u_4$. Then, we find

Figure 3.11 Four-point form of a curve.

a 4×4 matrix of constants that, when it premultiplies the matrix of coordinates of the four points, produces the **B** matrix of geometric coefficients:

$$\mathbf{B} = \mathbf{K}[\mathbf{p}_1 \quad \mathbf{p}_2 \quad \mathbf{p}_3 \quad \mathbf{p}_4]^T$$

or

$$\mathbf{B} = \mathbf{KP}$$

where $\mathbf{P} = [\mathbf{p}_1 \quad \mathbf{p}_2 \quad \mathbf{p}_3 \quad \mathbf{p}_4]^T$ and **K** is a 4×4 matrix of constants, depending solely on the u values assigned to the four points. (For the moment, we will treat all four as having arbitrary values.) It then follows that

$$\begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \mathbf{p}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \\ \mathbf{U}_3 \\ \mathbf{U}_4 \end{bmatrix} \mathbf{M}_F \mathbf{B} \tag{3.34}$$

where $\mathbf{U}_1 = [u_1^3 \quad u_1^2 \quad u_1 \quad 1]$ and similarly for $\mathbf{U}_2, \mathbf{U}_3, \mathbf{U}_4$, so that

$$\begin{bmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \\ \mathbf{U}_3 \\ \mathbf{U}_4 \end{bmatrix} = \begin{bmatrix} u_1^3 & u_1^2 & u_1 & 1 \\ u_2^3 & u_2^2 & u_2 & 1 \\ u_3^3 & u_3^2 & u_3 & 1 \\ u_4^3 & u_4^2 & u_4 & 1 \end{bmatrix}$$

We solve this equation directly to obtain

$$\mathbf{B} = \mathbf{M}_F^{-1}\begin{bmatrix}\mathbf{U}_1\\ \mathbf{U}_2\\ \mathbf{U}_3\\ \mathbf{U}_4\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{p}_1\\ \mathbf{p}_2\\ \mathbf{p}_3\\ \mathbf{p}_4\end{bmatrix}$$

and then extract the **K** matrix

$$\mathbf{K} = \mathbf{M}_F^{-1}\begin{bmatrix}\mathbf{U}_1\\ \mathbf{U}_2\\ \mathbf{U}_3\\ \mathbf{U}_4\end{bmatrix}^{-1} \tag{3.35}$$

Conversely, given the **B** matrix, the equivalent four-point form is

$$\mathbf{P} = \mathbf{K}^{-1}\mathbf{B} \tag{3.36}$$

Taking a standard approach and choosing equally distributed values of the parametric variable, that is, for $u_1 = 0, u_2 = 1/3, u_3 = 2/3, u_4 = 1$, then for **K** and $\mathbf{K}^{-1}$ we find

$$\mathbf{K} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\\ -\frac{11}{2} & 9 & -\frac{9}{2} & 1\\ -1 & \frac{9}{2} & -9 & \frac{11}{2}\end{bmatrix} \tag{3.37}$$

$$\mathbf{K}^{-1} = \begin{bmatrix}1 & 0 & 0 & 0\\ \frac{20}{27} & \frac{7}{27} & \frac{4}{27} & -\frac{2}{27}\\ \frac{7}{27} & \frac{20}{27} & \frac{2}{27} & -\frac{4}{27}\\ 0 & 1 & 0 & 0\end{bmatrix} \tag{3.38}$$

We can compose a slightly different expression as follows: Given $\mathbf{p}(u) = \mathbf{U}\mathbf{M}_F\mathbf{B}$ and $\mathbf{B} = \mathbf{K}\mathbf{P}$, by substitution we obtain

$$\mathbf{p}(u) = \mathbf{U}\mathbf{M}_F\mathbf{K}\mathbf{P}$$

If $\mathbf{M}_F\mathbf{K} = \mathbf{N}$, then

$$\mathbf{p}(u) = \mathbf{U}\mathbf{N}_F\mathbf{P} \tag{3.39}$$

where

$$\mathbf{N}_F = \begin{bmatrix} -\frac{9}{2} & \frac{27}{2} & -\frac{27}{2} & \frac{9}{2} \\ 9 & -\frac{45}{2} & 18 & -\frac{9}{2} \\ -\frac{11}{2} & 9 & -\frac{9}{2} & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Expanding Equation (3.39) produces

$$\begin{aligned} \mathbf{p}(u) = {} & (-4.5u^3 + 9u^2 - 5.5u + 1)\mathbf{p}_1 \\ & +(13.5u^3 - 22.5u^2 + 9u)\mathbf{p}_2 \\ & +(-13.5u^3 + 18u^2 - 4.5u)\mathbf{p}_3 \\ & +(4.5u^3 - 4.5u^2 + u)\mathbf{p}_4 \end{aligned}$$

Notice that we have created new basis functions, say $G_i(u)$, so that

$$\mathbf{p}(u) = G_1(u)\mathbf{p}_1 + G_2(u)\mathbf{p}_2 + G_3(u)\mathbf{p}_3 + G_4(u)\mathbf{p}_4$$

These basis functions possess the *equal partition of unity* characteristic because $\sum G_i(u) = 1$.

3.8 CONIC HERMITE CURVES

Constructing a conic curve the traditional way is worth reviewing to better understand how to use Hermite curves to represent them. To begin we state the problem as: Given two endpoints A and B, corresponding tangent lines, and an intermediate point C, all in a common plane, find a set of intermediate points sufficient to plot a segment of a conic curve (see Figure 3.12). The tangent at A is represented by line AD and at B by BD, where D is the point of intersection of the tangent lines. C must lie within the triangle ABC. We construct a set of points on the conic curve as follows:

1. Draw lines AC and BC, extending both lines past C.
2. Draw any line through D that intersects both lines AC and BC; a and b are such intersection points.

Figure 3.12 Construction technique for a conic curve.

3. Draw a line through points A and b.
4. Draw a line through points a and B, extending it until it intersects line Ab. The point of intersection P is a point on the conic curve.

We repeat these four steps as many times as necessary, each time starting with a different line through D. If we construct a line through D intersecting line AB at its midpoint E, then we can demonstrate that the tangent to the conic at point F is parallel to AB. Furthermore, we can determine the type of conic by computing the ratio of the length of line segment EF to the length of line segment DE. We call this ratio *rho* (ρ). If ρ is less than 0.5, then the curve is a segment of an ellipse. If $\rho = 0.5$, the curve is a parabolic segment. If $0.5 < \rho \leq 1$, the curve is a segment of a hyperbola.

Alternatively, we can construct line DE (see Figure 3.13) so that it intersects AB at its midpoint, and locate point C to yield ρ appropriate to the type of conic required. For example, if the segment of a parabola has endpoints at A and B and end slopes corresponding to lines AD and BD, then we locate C to make $\rho = 0.5$, that is, at the midpoint of DE. Next, we proceed with the steps previously outlined in order to generate additional points on the curve. Notice that if the perpendicular distance from point D to line AB is denoted as H, then the perpendicular distance from point C to this line is ρH.

We can define a Hermite curve whose endpoints and slopes correspond to A and B, and AD and BD, respectively, and that passes through the

Figure 3.13 More on the construction technique for a conic curve.

intermediate point C. We derive only the case where point C lies on the line passing through D and the midpoint E of line AB; Figure 3.14 illustrates this. Keep in mind that all these elements are coplanar. (Mathematicians can show that a polynomial of degree n is always contained in an n-dimensional space.) In Figure 3.14a, we see a relationship derived earlier, namely,

$$\mathbf{p}_{0.5} = 0.5(\mathbf{p}_0 + \mathbf{p}_1) + 0.125(\mathbf{p}_1'' - \mathbf{p}_0'')$$

Point C is located on DE, a distance ρDE from point E. In Figure 3.14b, lines EG and GC are parallel to AD and BD, respectively. Triangle BEF is similar to BAD; because $BE = AB/2$, therefore $EF = AD/2$ and $BF = BD/2$. Triangle CEG is similar to DEF; because $CE = \rho DE$, therefore $CG = \rho DF/2$ and $EG = \rho EF/2$. Furthermore, because $\rho DF = \rho(BD - BF) = \rho BD/2$, we find that

$$0.125|\mathbf{p}_0''| = 0.5\rho AD$$

From Figure 3.14c, we obtain

$$AD = |\mathbf{p}_2 - \mathbf{p}_0|$$

Figure 3.14 Hermite approximation of the construction of a conic curve.

Therefore,

$$|\mathbf{p}_0''| = 4\rho|\mathbf{p}_2 - \mathbf{p}_0|$$

Similarly,

$$0.125|\mathbf{p}_1''| = 0.5\rho BD$$

where $BD = |\mathbf{p}_1 - \mathbf{p}_2|$ and $|\mathbf{p}_1''| = 4\rho|\mathbf{p}_1 - \mathbf{p}_2|$. But $\mathbf{p}_0''$ is in the direction of $\mathbf{p}_2 - \mathbf{p}_0$, and $\mathbf{p}_1''$ is in the direction of $\mathbf{p}_1 - \mathbf{p}_2$, so that

$$\mathbf{p}_0'' = 4\rho(\mathbf{p}_2 - \mathbf{p}_0) \qquad \text{and} \qquad \mathbf{p}_1'' = 4\rho(\mathbf{p}_1 - \mathbf{p}_2)$$

Therefore, the Hermite equation for a conic is

$$\mathbf{p}(u) = \mathbf{U}\mathbf{M}_F[\mathbf{p}_0 \quad \mathbf{p}_1 \quad 4\rho(\mathbf{p}_2 - \mathbf{p}_0) \quad 4\rho(\mathbf{p}_1 - \mathbf{p}_2)]^T \tag{3.40}$$

Figure 3.15 illustrates this and also shows that $\mathbf{p}''(0.5)$ is parallel to $\mathbf{p}_1 - \mathbf{p}_0$.

The accuracy of Equation (3.40) as it applies to various types of conics is an important modeling consideration. When $\rho = 0.5$ a parabola is generated, and this equation becomes

$$\mathbf{p} = \mathbf{U}\mathbf{M}_F[\mathbf{p}_0 \quad \mathbf{p}_1 \quad 2(\mathbf{p}_2 - \mathbf{p}_0) \quad 2(\mathbf{p}_1 - \mathbf{p}_2)]^T$$

A Hermite curve defined by these boundary conditions is a conic, and, moreover, it is exactly a segment of a parabola. Figure 3.16 illustrates the general conditions necessary to generate this type of curve. Figures 3.17 and 3.18 are

Figure 3.15 More on the Hermite approximation of the construction of a conic curve.

Figure 3.16 A Hermite curve exactly matching a segment of a parabola.

examples of Hermite curves that demonstrate the exact match between this formulation and a parabolic segment, in particular the parabola $x^2 - y = 0$. Although we have been working in the x, y plane, we can readily create or transform the curve into any plane.

If $0.5 < \rho < 1$, then Equation (3.40) generates a Hermite curve approximating a segment of a hyperbola. The approximation improves as the hyperbolic segment becomes smaller, that is, as the difference between the slopes at each end decreases. In another sense, the more Hermite curves strung end-to-end to approximate a given hyperbolic segment, the better the match—the smaller the error or deviation of the curve from the true hyperbola. Figures 3.19 and 3.20 demonstrate this for different segments of the hyperbola given by the equation $x^2 - y^2 + 1 = 0$.

In Figure 3.19, a Hermite curve approximates the segment of the hyperbola from $\mathbf{p}_0 = [-10 \quad 10.05 \quad 0]$ to $\mathbf{p}_1 = [10 \quad 10.05 \quad 0]$. This segment is symmetrical around the y axis. The point on this curve corresponding to $u = 0.5$ is $\mathbf{p}(0.5) = [0 \quad 1 \quad 0]$. The point $0.5(\mathbf{p}_0 + \mathbf{p}_1) = [0 \quad 10 \quad 0]$. We find $\mathbf{p}_2$ by determining the point of intersection of the slopes or tangents at $\mathbf{p}_0$ and $\mathbf{p}_1$. Because we know these points, we next calculate $\rho = 0.91$. Substituting these values into Equation (3.40) produces

$$\mathbf{B} = \begin{bmatrix} -10. & 10.05 & 0 \\ 10. & 10.05 & 0 \\ 36.4 & -36.22 & 0 \\ 36.4 & 36.22 & 0 \end{bmatrix}$$

Parabola: $x^2 - y = 0$
$\rho = 0.5$

$\pm x$	y	dy/dx
0	0	0
1	1	2
2	4	4
3	9	6
4	16	8

$\mathbf{p} = \mathbf{F}[\mathbf{p}_0 \quad \mathbf{p}_1 \quad 2(\mathbf{p}_2 - \mathbf{p}_0) \quad 2(\mathbf{p}_1 - \mathbf{p}_2)]^t$

u	$x(u)$	$y(u)$	y True	Error
0	−4.0	16.00	16.00	0
0.2	−2.4	5.76	5.76	0
0.4	−0.8	0.64	0.64	0
0.5	0	0	0	0
0.6	0.8	0.64	0.64	0
0.8	2.4	5.76	5.76	0
1.0	4.0	16.00	16.00	0

Figure 3.17 An example of a **Hermite curve** representing a segment of a parabola.

We compute and plot the coordinates of points on the curve for a set of u values and substitute the values of $x(u)$ into the classical equation for the hyperbola, computing the corresponding true values of y. Then, we compare these to the $y(u)$ values and find a deviation or approximation error and tabulate it. The tabulated error is simply the difference between y values. This is greater than the perpendicular distance between the two curves, which is a more appropriate measure of the error.

Figure 3.20 shows a similar approach. Here we represent the right side of the hyperbola by two Hermite curves, computing the ρ values and geometric coefficients as before. The error or deviation of this curve from the true hyperbola is much less than that of the model in Figure 3.19.

Finally, we investigate a Hermite curve approximation of a circular arc. The match is not exact, but the smaller the angle subtended by the circular

Parabola, $x^2 - y = 0$
$\rho = 0.5$

x	y	$\frac{dy}{dx}$
−1.5	2.25	−3
−1	1	−2
0	0	0
1	1	2
2	4	4
2.5	6.25	5

$$\mathbf{p} = \mathbf{F}[\mathbf{p}_0 \quad \mathbf{p}_1 \quad 2(\mathbf{p}_2 - \mathbf{p}_0) \quad 2(\mathbf{p}_1 - \mathbf{p}_2)]^T$$

u	$x(u)$	$y(u)$	y True	Error
0	−1.5	2.25	2.25	0
0.2	−0.7	0.49	0.49	0
0.4	0.1	0.01	0.01	0
0.5	0.5	0.25	0.25	0
0.6	0.9	0.81	0.81	0
0.8	1.7	2.89	2.89	0
1	2.5	6.25	6.25	0

Figure 3.18 Another example of a Hermite curve representing a segment of a parabola.

arc, the better the approximation. Figure 3.21 shows that for angles less than 45°, $\delta R/R$ is less than 5×10^{-6}, more than adequate for most applications. There are two different approaches to this problem, both of which are based on the relationship expressed in Equation (3.40). Figure 3.22 shows the first approach; it puts $\mathbf{p}(0.5)$ exactly on the true circle. The geometry of this situation is very simple, and we can easily compute the value of ρ:

$$\rho = \frac{R(1 - \cos\theta)}{R\tan\theta\sin\theta} = \frac{1 - \cos\theta}{\tan\theta\sin\theta}$$

A few trigonometric substitutions quickly reduce this to

$$\rho = \frac{\cos\theta}{1 + \cos\theta} \tag{3.41}$$

This allows us to write directly the Hermite equation approximating a circular arc:

$$\mathbf{p}(u) = \mathbf{U}\mathbf{M}_F\left[\mathbf{p}_0 \quad \mathbf{p}_1 \quad \frac{4\cos\theta}{1 + \cos\theta}(\mathbf{p}_2 - \mathbf{p}_0) \quad \frac{4\cos\theta}{1 + \cos\theta}(\mathbf{p}_1 - \mathbf{p}_2)\right]^T$$

Hyperbola:

$$x^2 - y^2 + 1 = 0$$
$$\rho = 0.91$$

$\pm x$	y	$\pm \frac{dy}{dx}$
0	1	0
2	1.24	0.89
4	4.12	0.97
6	6.08	0.99
8	8.06	0.99
10	10.05	0.99

$$\mathbf{p} = \mathbf{F}[\mathbf{p}_0 \quad \mathbf{p}_1 \quad 4\rho(\mathbf{p}_2 - \mathbf{p}_0) \quad 4\rho(\mathbf{p}_1 - \mathbf{p}_2)]^T$$

u	$x(u)$	$y(u)$	y True	Error
0	−10	10.05	10.05	0
0.2	−4.43	4.26	4.54	0.28
0.4	−1.21	1.86	1.57	0.21
0.5	0	1	1.00	0
0.6	1.21	1.36	1.57	0.21
0.8	4.43	4.26	4.54	0.28
1	10	10.05	10.05	0

Figure 3.19 An example of a Hermite curve approximating a segment of a hyperbola.

The alternative approach is also shown in Figure 3.22. It uses the familiar relationship $\mathbf{p}(0.5) = 0.5(\mathbf{p}_0 + \mathbf{p}_1) + 0.125(\mathbf{p}_0^u - \mathbf{p}_1^u)$. Again, the geometry of the elements is very simple, and we find

$$0.125\mathbf{p}_0^u = \frac{R(1 - \cos\theta)}{2\sin\theta}\left(\frac{\mathbf{p}_2 - \mathbf{p}_0}{|\mathbf{p}_2 - \mathbf{p}_0|}\right)$$

$$\mathbf{p}_0^u = \frac{4R(1 - \cos\theta)}{\sin\theta}\left(\frac{\mathbf{p}_2 - \mathbf{p}_0}{|\mathbf{p}_2 - \mathbf{p}_0|}\right)$$

The same reasoning yields

$$\mathbf{p}_1^u = \frac{4R(1 - \cos\theta)}{\sin\theta}\left(\frac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|}\right)$$

We then assemble all the elements into the complete Hermite equation:

$$\mathbf{p}(u) = \mathbf{U}\mathbf{M}_F\left[\mathbf{p}_0 \quad \mathbf{p}_1 \quad \frac{4R(1 - \cos\theta)}{\sin\theta}\left(\frac{\mathbf{p}_2 - \mathbf{p}_0}{|\mathbf{p}_2 - \mathbf{p}_0|}\right) \quad \frac{4R(1 - \cos\theta)}{\sin\theta}\left(\frac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|}\right)\right]^T$$

Hyperbola: $x^2 - y^2 + 1 = 0$

$\pm x$	y	$\pm \frac{dy}{dx}$
0	1	0
2	1.24	0.89
4	4.12	0.97
6	6.08	0.99
8	8.06	0.99
10	10.05	0.99

Segment 1, $\rho = 0.5601$

$$\mathbf{p}_0 = \begin{pmatrix} 0.0 \\ 1.0 \end{pmatrix} \quad \mathbf{p}_1 = \begin{pmatrix} 2.0000 \\ 2.2361 \end{pmatrix} \quad \mathbf{p}_2 = \begin{pmatrix} 0.6180 \\ 1.0009 \end{pmatrix}$$

Segment 2, $\rho = 0.5679$

$$\mathbf{p}_0 = \begin{pmatrix} 2.0000 \\ 2.2361 \end{pmatrix} \quad \mathbf{p}_1 = \begin{pmatrix} 10.0000 \\ 10.0499 \end{pmatrix} \quad \mathbf{p}_2 = \begin{pmatrix} 3.4560 \\ 3.5384 \end{pmatrix}$$

	u	$x(u)$	$y(u)$	y True	Error
Segment 1	0	0	1	1	0
	0.2	0.2861	1.0400	1.0401	0.0001
	0.4	0.6062	1.1692	1.1694	0.0002
	0.5	0.7861	1.2719	1.2719	0
	0.6	0.9831	1.4022	1.4023	0.0001
	0.8	1.4400	1.7530	1.7532	0.0002
	1	2.0000	2.2361	2.2361	0
Segment 2	0	2.0000	2.2361	2.2361	0
	0.2	2.7796	2.9541	2.9540	0.0001
	0.4	3.8652	3.9926	3.9924	0.0002
	0.5	4.5552	4.6637	4.6637	0
	0.6	5.3609	5.4534	4.4534	0
	0.8	7.3710	7.4386	7.4385	0.0001
	1	10.0000	10.0499	10.0499	0.0001

Figure 3.20 Another example of a Hermite curve approximating a segment of a hyperbola.

Figure 3.21 Another Hermite curve approximation of a circular arc.

The maximum deviation of the Hermite curve from the true circle occurs at two places (see Figure 3.22). Adjusting the magnitudes of the tangent vectors so that $\mathbf{p}(0.5)$ falls slightly inside the circular arc reduces the deviation somewhat. This is equivalent to reducing the ρ value. Therefore, from Figure 3.23, we obtain

$$\rho = \frac{(1 - \cos\theta - \delta R/R)\cos\theta}{\sin^2\theta} \tag{3.42}$$

The magnitude of the deviation depends on θ (see Figure 3.21).

For any specific angle θ, an increase of $\delta R/R$ at $\mathbf{p}(0.5)$ causes a decrease in the maximum deviation in the interval $u = 0$ to $u = 0.5$ and in the interval $u = 0.5$ to $u = 1$. There is a value of $\delta R/R$ that makes the three maximum deviations equal. This value is different for every value of θ. The smaller the angle an arc subtends, the smaller the deviation of the curve from the true circular arc. This relationship is also plotted in Figure 3.21. If $\delta R/R = 0$, then Equation (3.42) reduces to Equation (3.41).

3.9 COMPOSITE HERMITE CURVES

Joining two or more curve segments together forms a *continuous composite curve*. We start the discussion of composite curves and their continuity by

$$\mathbf{p}_0'' = \frac{4\cos\theta}{1+\cos\theta}(\mathbf{p}_2 - \mathbf{p}_0) \quad \text{or} \quad \frac{4R(1-\cos\theta)}{\sin\theta}\frac{(\mathbf{p}_2 - \mathbf{p}_0)}{|\mathbf{p}_2 - \mathbf{p}_0|}$$

$$\mathbf{p}_1'' = \frac{4\cos\theta}{1+\cos\theta}(\mathbf{p}_1 - \mathbf{p}_2) \quad \text{or} \quad \frac{4R(1-\cos\theta)}{\sin\theta}\frac{(\mathbf{p}_1 - \mathbf{p}_2)}{|\mathbf{p}_1 - \mathbf{p}_2|}$$

Figure 3.22 Deviation of a Hermite curve from a true circular arc.

investigating how to blend a new curve between two existing curves, forming a composite curve consisting of three segments (see Figure 3.24). Given the geometric coefficients of two disjoint curves $\mathbf{B}_1$ and $\mathbf{B}_3$, we must find the geometric coefficients $\mathbf{B}_2$ of a curve joining them so that the two curve segments meeting at each of the joints are tangent to the same line. We let $\mathbf{B}_1 = [\mathbf{p}_1(0) \quad \mathbf{p}_1(1) \quad \mathbf{p}_1''(0) \quad \mathbf{p}_1''(1)]^T$ and $\mathbf{B}_3 = [\mathbf{p}_3(0) \quad \mathbf{p}_3(1) \quad \mathbf{p}_3''(0) \quad \mathbf{p}_3''(1)]^T$, modifying the notation scheme to permit the use of a subscript to identify a specific curve segment. Later, we modify this further so that each point is uniquely identified.

First, the appropriate endpoints must coincide; this means that $\mathbf{p}_2(0) = \mathbf{p}_1(1)$ and $\mathbf{p}_2(1) = \mathbf{p}_3(0)$. Furthermore, although the tangent line at each end of the new curve must match the tangent line of the adjoining curve, the magnitudes of adjoining tangent vectors can be different. In fact, an infinite num-

$$\rho = \frac{R(1-\cos\theta) - \delta R}{R\tan\theta\sin\theta} = \frac{[(1-\cos\theta - (\delta R/R)]}{\sin^2\theta}\cos\theta$$

Figure 3.23 A Hermite curve approximation of a circular arc.

ber of segments satisfies the tangency conditions. However, the unit tangent vectors must be equal. This means that

$$\mathbf{p}_2''(0) = a\frac{\mathbf{p}_1''(1)}{|\mathbf{p}_1''(1)|} \qquad \text{and} \qquad \mathbf{p}_2''(1) = b\frac{\mathbf{p}_3''(0)}{|\mathbf{p}_3''(0)|}$$

The $\mathbf{B}_2$ matrix, in terms of the adjacent curves, is

$$\mathbf{B}_2 = \left[\mathbf{p}_1(1) \quad \mathbf{p}_3(0) \quad a\frac{\mathbf{p}_1''(1)}{|\mathbf{p}_1''(1)|} \quad b\frac{\mathbf{p}_3''(0)}{|\mathbf{p}_3''(0)|}\right]^T$$

where a and b are positive scale factors. We are free to vary them to change the internal shape of the new curve.

The following expression describes any cubic Hermite curve smoothly and continuously blended with preceding and succeeding curves:

$$\mathbf{B}_i = \left[\mathbf{p}_{i-1}(1) \quad \mathbf{p}_{i+1}(0) \quad a_i\frac{\mathbf{p}_{i-1}''(1)}{|\mathbf{p}_{i-1}''(1)|} \quad b_i\frac{\mathbf{p}_{i+1}''(0)}{|\mathbf{p}_{i+1}''(0)|}\right]^T$$

Figure 3.24 Blending a curve between two existing curves.

With this small exercise complete, we can now elaborate on the general conditions of continuity at a joint between two cubic Hermite curves and the conditions arising when two or more curves are joined end-to-end forming a string of curves, called a *composite curve*. Joining two curve segments at a common point so that their first n parametric derivatives are equal at that point creates a condition of continuity called nth-order *parametric continuity*, denoted as C^n. A less restrictive form of nth-order continuity is nth-order *geometric continuity*, denoted as G^n. The curve defined previously has G^1 or first-order geometric continuity, because corresponding tangent vectors must be in the same direction but not of the same magnitude. If $\mathbf{p}_2''(0) = \mathbf{p}_1''(1)$ and $\mathbf{p}_2''(1) = \mathbf{p}_3''(0)$, then the continuity at both joints would be C^1, or first-order parametric continuity. Clearly, only one Hermite curve joining curves 1 and 3 satisfies C^1 continuity, whereas infinitely many satisfy G^1.

Geometric continuity is defined in terms of the intrinsic differential properties of a curve or surface, such as the unit tangent vector and curvature. These properties are independent of the parameterization. Parametric continuity absorbs more degrees of freedom than geometric continuity. If the conditions producing parametric continuity are relaxed while maintaining geometric continuity, then additional degrees of freedom become available to control the curve's shape. For example, Barsky's Beta-Splines offer extra shape control parameters by taking advantage of this distinction. Similar methods apply as well to surfaces.

In general, a curve is either continuous at all points, or it has one or more points of discontinuity. The simplest kind of continuity a curve can have is C^0 continuity, which ensures that there are no gaps or breaks between its begin-

ning and ending points. If what is alleged to be a single curve does not have C^0 continuity, then it must really be two or more curves, depending on the number of gaps.

Two cubic Hermite curves joined at a common endpoint have at least C^0 continuity at that joint. They comprise the simplest kind of composite curve. We note that the term *curve* alone will often mean a curve segment or composite curve. The meaning should be clear from the context. Figure 3.25 shows examples of composite curves and C^0 continuity. Figure 3.25a obviously consists of two distinct and separate curves, whereas in Figure 3.25b, they are joined and form a compound curve with C^0 continuity at the joint. Figure 3.25c is a compound curve consisting of three segments, again with C^0

Figure 3.25 Composite curves and geometric continuity.

Figure 3.26 Conditions required for G^0.

continuity at the joints. Figure 3.25d is an arbitrary shape defined with an arbitrary set of segments to form the closed composite curve in Figure 3.25e.

When two curves join end-to-end at a common point, as in Figure 3.26, the coordinates of that point are a common subset of the geometric coefficients of both curve segments. We observe that the C^0 continuity thus obtained does not affect the shape of either curve segment.

G^1 continuity between two curve segments requires a common tangent line at their joining point. Figure 3.27 is an example of G^1 continuity and indicates the mathematical conditions that must be met. The magnitudes of the tangent vectors at $\mathbf{p}_1(1) = \mathbf{p}_2(0)$ do not have to be equal, but the ratios of their components must be equal.

Figure 3.27 Conditions required for G^1 continuity.

It is apparent that we can join any two curves and then rotate them about their common point until they are tangent to each other while preserving the internal shape of each. Whereas G^1 continuity implies sharing a common point and tangent line at the junction of two curves, G^2 continuity also requires that the two curves possess equal curvature at their joint. To ensure this, the following conditions must exist:

$$\mathbf{p}_i(1) = \mathbf{p}_{i+1}(0)$$

$$\mathbf{p}_i''(1) = k_1\mathbf{p}_{i+1}''(0)$$

$$\mathbf{p}_i'''(1) = k_2\mathbf{p}_{i+1}'''(0)$$

The relationship between the parametric second derivatives ensures that $y_i^{xx} = y_{i+1}^{xx}$ at their common point. One more condition must be met for G^2 continuity: The osculating planes of the two curves must coincide, or their binormals must be collinear. However, the usefulness of G^2 continuity is limited. For example, most mechanical parts do not require it, since fillets or rounded edges usually blend directly into plane faces.

4

BÉZIER CURVES

Some curve-defining techniques, such as those using the Hermite basis functions, interpolate a given set of points. This means that the curve produced passes exactly through those points. Other techniques define a curve that only passes near or approximates the given points. Interpolation techniques like the Hermite basis have certain disadvantages when incorporated into an interactive geometric-modeling system. They usually do not give the user an intuitive sense of how to change or control the shape of a curve. For example, changing the shape of a spline-interpolated curve by moving one or more of the interpolating points may produce unexpected, if not undesirable, perturbations and inflections, both locally and globally. To control curve shape in a predictable way by changing only a few simple parameters is more desirable. The Bézier curve partially satisfies this need.

P. Bézier, who was familiar with the work of Ferguson and Coons and their parametric cubic curve and bicubic surface interpolating techniques, set out in the early 1960s to find what he hoped would be a mathematical format more amenable to the designer and the design process. The result of his work was the UNISURF system, used by the Renault company in the 1970s to design the sculptured surfaces of many of its automobile bodies. At the heart of the UNISURF system were the curves and surfaces that bear his name.

Bézier started with the principle that any point on a curve segment must be given by a parametric function of the following form:

$$\mathbf{p}(u) = \sum_{i=0}^{n} \mathbf{p}_i f_i(u) \qquad u \in [0,1] \tag{4.1}$$

Figure 4.1 Cubic Bézier curves.

where the vectors $\mathbf{p}_i$ represent the $n + 1$ vertices of a *characteristic polygon* (see Figure 4.1). These vertices are the *control points*. He then set forth the properties that the $f_i(u)$ basis functions must have and looked for specific functions to meet these requirements. Here is a review of these properties and the reasons for them.

1. The functions must interpolate the first and last vertex points; that is, the curve segment must start on $\mathbf{p}_0$ and end on $\mathbf{p}_n$. It is up to us to control the starting and ending points of a Bézier curve.
2. The tangent at $\mathbf{p}_0$ must be given by $\mathbf{p}_1 - \mathbf{p}_0$, and the tangent at $\mathbf{p}_n$ by $\mathbf{p}_n - \mathbf{p}_{n-1}$. This, of course, gives us direct control of the tangent to the curve at each end.
3. Requirement 2 is generalized for higher derivatives at the curve's endpoints. Thus, the second derivative at $\mathbf{p}_0$ must be determined by $\mathbf{p}_0$, $\mathbf{p}_1$, and $\mathbf{p}_2$. In general, the rth derivative at an endpoint must be determined by its r neighboring vertices. This allows us virtually unlimited control of the continuity at the joints between curve segments of a composite Bézier curve.

4. The functions $f_i(u)$ must be symmetric with respect to u and $(1 - u)$. This means that we can reverse the sequence of the vertex points defining the curve without changing the shape of the curve. In effect, this reverses the direction of parameterization.

4.1 BÉZIER BASIS FUNCTIONS

Bézier chose a family of functions called *Bernstein polynomials* to satisfy these properties. He originally chose a form of vector representation that used the sides of the characteristic polygon. However, we will use the notation introduced by Forrest (1971), which uses the vectors defining the polygon vertices. It is a more compact scheme, with greater intuitive appeal. It turns out that the functions Bézier selected depend on the number of vertices used to specify a particular curve. To indicate this, Equation (4.1) becomes

$$\mathbf{p}(u) = \sum_{i=0}^{n} \mathbf{p}_i B_{i,n}(u) \qquad u \in [0,1] \tag{4.2}$$

where the basis functions are

$$B_{i,n}(u) = \binom{n}{i} u^i (1-u)^{n-i} \tag{4.3}$$

and where

$$\binom{n}{i}$$

denotes the familiar binomial coefficient function, or binomial distribution from probability and statistics:

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

The following conventions apply: If i and u equal zero, then $u^i = 1$, and $0! = 1$. If there are $(n + 1)$ vertices, then the function $B_{i,n}(u)$ yields an nth-degree polynomial.

Expanding Equation (4.2) for curves defined by three, four, and five points produces the following polynomial forms:

For three points, $n = 2$, and

$$B_{0,2} = (1-u)^2$$

$$B_{1,2} = 2u(1-u)$$

$$B_{2,2} = u^2$$

so that

$$\mathbf{p}(u) = (1-u)^2\,\mathbf{p}_0 + 2u(1-u)\mathbf{p}_1 + u^2\mathbf{p}_2 \tag{4.4}$$

For four points, $n = 3$, and

$$B_{0,3} = (1-u)^3$$
$$B_{1,3} = 3u(1-u)^2$$
$$B_{2,3} = 3u^2(1-u)$$
$$B_{3,3} = u^3$$

so that

$$\mathbf{p}(u) = (1-u)^3\,\mathbf{p}_0 + 3u(1-u)^2\,\mathbf{p}_1 + 3u^2\,(1-u)\mathbf{p}_2 + u^3\mathbf{p}_3 \tag{4.5}$$

Using this cubic Bézier curve, a segment of a parabola with endpoints at $\mathbf{p}_0$ and $\mathbf{p}_3$ can be modeled exactly, if the end tangents intersect at $\mathbf{p}_1$ so that $\mathbf{p}_1 = (\mathbf{p}_0 + 2\mathbf{p}_1)/3$ and $\mathbf{p}_2 = (\mathbf{p}_3 + 2\mathbf{p}_1)/3$. These conditions produce $\mathbf{p}(u) = (1-u)^2\,\mathbf{p}_0 + 2(u-u^2)\mathbf{p}_1 + u^2\mathbf{p}_3$, which is exactly the form of Equation (4.4).

For five points, $n = 4$, and

$$\mathbf{p}(u) = (1-u)^4\,\mathbf{p}_0 + 4u(1-u)^3\,\mathbf{p}_1 + 6u^2(1-u)^2\,\mathbf{p}_2 + 4u^3\,(1-u)\mathbf{p}_3 + u^4\mathbf{p}_4 \tag{4.6}$$

Matrix Form

For the cubic Bézier curve, $n = 3$, we write Equation (4.5) in matrix form as follows:

$$\mathbf{p}(u) = \begin{bmatrix}(1-3u+3u^2-u^3) & (3u-6u^2+3u^3) & (3u^2-3u^3) & u^3\end{bmatrix}\begin{bmatrix}\mathbf{p}_0\\ \mathbf{p}_1\\ \mathbf{p}_2\\ \mathbf{p}_3\end{bmatrix}$$

or as

$$\mathbf{p}(u) = \begin{bmatrix}u^3 & u^2 & u & 1\end{bmatrix}\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 3 & 0 & 0\\ 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\mathbf{p}_0\\ \mathbf{p}_1\\ \mathbf{p}_2\\ \mathbf{p}_3\end{bmatrix}$$

Letting

$$\mathbf{U} = \begin{bmatrix}u^3 & u^2 & u & 1\end{bmatrix}$$
$$\mathbf{P} = \begin{bmatrix}\mathbf{p}_0 & \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3\end{bmatrix}^T$$

$$\mathbf{M}_B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

we can write this equation even more compactly as

$$\mathbf{p}(u) = \mathbf{U}\mathbf{M}_B\mathbf{P} \tag{4.7}$$

The subscript on the cubic Bézier basis transformation matrix $\mathbf{M}_B$ distinguishes it from the Hermite basis transformation matrix $\mathbf{M}_F$. The inverse of $\mathbf{M}_B$ for the cubic Bézier curve is

$$\mathbf{M}_B^{-1} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{3} & 1 \\ 0 & \frac{1}{3} & \frac{2}{3} & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Obviously, the composition of these matrices varies with the number of vertices, $n + 1$. So, for example, for $n = 4$ we have

$$\mathbf{U} = [u^4 \quad u^3 \quad u^2 \quad u \quad 1]$$

$$\mathbf{M}_B = \begin{bmatrix} 1 & -4 & 6 & 4 & 1 \\ -4 & -12 & -12 & 4 & 0 \\ 6 & -12 & 6 & 0 & 0 \\ -4 & 4 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \mathbf{p}_4 \end{bmatrix}$$

Bézier-Hermite Conversion

Equation (4.7) is similar to Equation (3.20) for Hermite curves, and at this point it may occur to you that we can also represent a cubic Hermite curve with a sequence of four control points. Figure 4.2 presents the details. These points are denoted as $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$, where $\mathbf{p}_0$ and $\mathbf{p}_3$ are the curve end-

Figure 4.2 The Hermite equivalent of a cubic Bézier curve.

points. The two interior points contribute to the required tangent vectors in the following way:

$$\mathbf{p}_0'' = k_0\,(\mathbf{p}_1 - \mathbf{p}_0) \qquad \text{and} \qquad \mathbf{p}_1'' = k_1\,(\mathbf{p}_3 - \mathbf{p}_2)$$

where k_0 and k_1 are arbitrary scale factors introduced to control the scale of the polygon and, thus, the magnitude of the tangent vectors, which are not drawn to scale in the figure. The matrix of geometric coefficients is

$$\mathbf{B} = [\mathbf{p}_0 \quad \mathbf{p}_3 \quad k_0\,(\mathbf{p}_1 - \mathbf{p}_0) \quad k_1\,(\mathbf{p}_3 - \mathbf{p}_2)]^T$$

For $k_0 = k_1 = 3$, the curve is identical to the cubic Bézier curve defined by $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$. We see that this is true from the following: The general polynomial expression for the tangent vector at any point on the cubic Bézier curve is

$$\mathbf{p}''(u) = (-3u^2 + 6u - 3)\mathbf{p}_0 + (9u^2 - 12u + 3)\mathbf{p}_1 + (-9u^2 + 6u)\mathbf{p}_2 + 3u^2\mathbf{p}_3$$

At $u = 0$, $\mathbf{p}''(u) = 3(\mathbf{p}_1 - \mathbf{p}_0)$, and at $u = 1$, $\mathbf{p}''(1) = 3(\mathbf{p}_3 - \mathbf{p}_2)$. Thus, if a cubic Bézier curve is given by the four control points $[\mathbf{p}_0 \quad \mathbf{p}_1 \quad \mathbf{p}_2 \quad \mathbf{p}_3]$, then the equivalent Hermite basis curve has geometric coefficients $[\mathbf{p}_0 \quad \mathbf{p}_3 \quad 3(\mathbf{p}_1 - \mathbf{p}_0) \quad 3(\mathbf{p}_3 - \mathbf{p}_2)]$. Similarly, if a cubic Hermite curve is given by the geometric coefficients $[\mathbf{p}_0 \quad \mathbf{p}_1 \quad \mathbf{p}_0'' \quad \mathbf{p}_1'']$, then the equivalent Bézier curve control points are

$$\left[\mathbf{p}_0 \quad \left(\mathbf{p}_0 + \frac{1}{3}\mathbf{p}_0''\right) \quad \left(\mathbf{p}_1 - \frac{1}{3}\mathbf{p}_1''\right) \quad \mathbf{p}_1\right]$$

We can express this conversion between the Hermite and Bézier forms using rather simple matrix algebra:

$$\mathbf{B} = \mathbf{M}_F^{-1}\mathbf{M}_B\mathbf{P} \tag{4.8}$$

or

$$\mathbf{P} = \mathbf{M}_B^{-1}\mathbf{M}_F\mathbf{B} \tag{4.9}$$

where

$$\mathbf{M}_F^{-1}\mathbf{M}_B = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \quad \text{and} \quad \mathbf{M}_B^{-1}\mathbf{M}_F = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & \frac{1}{3} & 0 \\ 0 & 1 & 0 & -\frac{1}{3} \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Invariance under Affine Transformations

A Bézier curve is invariant under an *affine transformation* (translation, rotation, scaling, or shear). This means that the following two methods must produce the same point:

1. Compute the point

$$\mathbf{p}(u_i) = \sum_{i=0}^{n} \mathbf{p}_i B_{i,n}(u_i)$$

   and then apply an affine transformation $\mathbf{A}$ such that $\mathbf{p}'(u_i) = \mathbf{A}\mathbf{p}(u_i)$.

2. Apply an affine transformation $\mathbf{A}$ to the control points such that $\mathbf{p}_i' = \mathbf{A}\mathbf{p}_i$, and then evaluate the curve represented by these transformed control points at u_i, where

$$\mathbf{p}'(u_i) = \sum_{i=0}^{n} \mathbf{p}_i' B_{i,n}(u_i)$$

Therefore

$$A\mathbf{p}(u_i) = \sum_{i=0}^{n} A\mathbf{p}_i B_{i,n}(u_i)$$

This property allows us to translate, rotate, and scale a Bézier curve while preserving the relationship between evaluated points and the corresponding value of the parameter.

A Bézier curve is also invariant under an *affine reparameterization,* or parameter transformation. The transformation from the unit interval [0,1] to the arbitrary interval $[a,b]$ is an affine transformation. If $u \in [0,1]$ and $v \in [a,b]$, then the transformation is $u = (v - a)/(b - a)$. We express this algebraically as

$$\sum_{i=0}^{n} \mathbf{p}_i B_{i,n}(u) = \sum_{i=0}^{n} \mathbf{p}_i B_{i,n}\left(\frac{v-a}{b-a}\right)$$

Bézier Curve by Geometric Construction

We can create a Bézier curve as the locus of points produced by a simple recursive geometric construction derived from work by de Casteljau. Bézier generalized this method and developed the explicit nonrecursive expression of Equation (4.2). We begin with the construction of a second degree Bézier curve (see Figure 4.3).

Given any three points A,B,C, we draw the sides of the open polygon AB and BC (see Figure 4.3a). For successive values of u in the closed interval [0,1] we construct points D and E so that $AD/AB = BE/BC = u$. Finally, on DE we construct F so that $DF/DE = u$. F is a point on the curve. Repeating this process for other values of u generates a sequence of points that defines the curve (in this case, a parabola). Notice that we must be directionally consistent in the way we subdivide the polygon edges.

The vector description of this process is (see Figure 4.3b):

$$\mathbf{p}(u) = \mathbf{p}_0 + u(\mathbf{p}_1 - \mathbf{p}_0) + u[\mathbf{p}_1 + u(\mathbf{p}_2 - \mathbf{p}_1) - \mathbf{p}_0 - u(\mathbf{p}_1 - \mathbf{p}_0)]$$

Rearranging terms yields

$$\mathbf{p}(u) = (1-u)^2\,\mathbf{p}_0 + 2u(1-u)\mathbf{p}_1 + u^2\mathbf{p}_2$$

Here we begin to see evidence of the Bernstein polynomials.

A similar construction applies to Bézier curves of any degree, and the control points are not constrained to a plane. Figure 4.4 illustrates the process as it applies to the construction of a cubic Bézier curve. Here we are given any four points A,B,C,D and draw the three sides of the open polygon AB,BC,CD (see Figure 4.4a). For successive values of the parameter, we construct points E,F,G so that $AE/AB = BF/BC = CG/CD = u$. Next, we construct points H,I so that $EH/EF = FI/FG = u$. Finally, we construct point J so

Figure 4.3 Geometric construction of a Bézier curve: $n = 2$.

that $HJ/HI = u$. Point J is on the curve. Continuing this process produces a sequence of points whose locus defines the curve. Figure 4.4b shows how this, too, can be described with vectors, from which the Bernstein polynomials emerge; thus, we find the following unwieldy expression

$$\begin{aligned}\mathbf{p}(u) = \mathbf{p}_0 &+ u(\mathbf{p}_1 - \mathbf{p}_0) + u[\mathbf{p}_1 + u(\mathbf{p}_2 - \mathbf{p}_1) - \mathbf{p}_0 - u(\mathbf{p}_1 - \mathbf{p}_0)] \\ &+ u\{\mathbf{p}_1 + u(\mathbf{p}_2 - \mathbf{p}_1) + u[\mathbf{p}_2 + u(\mathbf{p}_3 - \mathbf{p}_2) - \mathbf{p}_1 - u(\mathbf{p}_2 - \mathbf{p}_0)] \\ &- \mathbf{p}_0 - u(\mathbf{p}_1 - \mathbf{p}_0) - u[\mathbf{p}_1 + u(\mathbf{p}_2 - \mathbf{p}_1) - \mathbf{p}_0 - u(\mathbf{p}_1 - \mathbf{p}_0)]\}\end{aligned}$$

which reduces to

$$\mathbf{p}(u) = (1-u)^3\,\mathbf{p}_0 + 3u(1-u)^2\,\mathbf{p}_1 + 3u^2\,(1-u)\mathbf{p}_2 + u^3\,\mathbf{p}_3$$

4.2 CONTROL POINTS

The *control points* are the coefficients of the Bernstein polynomial basis functions (see Equation [4.2]). When we connect these points in the order of

Figure 4.4 Geometric construction of a Bézier curve: $n = 3$.

their numbering with straight lines, we form the *Bézier control polygon.* All points on a Bézier curve lie within the convex hull of the control polygon. The *partition of unity property* of the Bernstein basis functions, that is

$$\sum_i B_{i,n}(u) = 1$$

gives rise to the *convex hull property.* In more formal mathematical terms, a convex hull is the smallest convex set that contains a given set. A straight line connecting any two points in a convex set lies entirely within that set.

The polygon formed by the control points serves two functions: It establishes the initial shape of the curve and then furnishes a framework for altering the curve. Figure 4.5 shows two examples of cubic Bézier curves. In Figure 4.5a, the disposition of points generates a smooth, uninflected curve, while in Figure 4.5b, the disposition of points generates an inflected curve. In both cases, the curves are tangent to the lines defined by $\mathbf{p}_1 - \mathbf{p}_0$ and $\mathbf{p}_3 - \mathbf{p}_2$.

(a)

(b)

Figure 4.5 Cubic Bézier curves.

Figure 4.6 Bézier basis functions: (*a*) Three points, $n = 2$; (*b*) Four points, $n = 3$; (*c*) Five points, $n = 4$; (*d*) Six points, $n = 5$.

As is the case for Hermite curves, the basis functions are the key to the behavior of Bézier curves. Figure 4.6 shows basis-function curves $B_{i,n}$ for $n =$ 2,3,4, and 5. For the examples in Figure 4.5, the first control point $\mathbf{p}_0$, whose contribution to the curve's shape is propagated by $B_{0,3}(u)$, is the most influential when $u = 0$. The other control points do not contribute to $\mathbf{p}(u)$ for $u =$ 0, since their associated basis functions are each equal to zero. A symmetrical situation occurs for $\mathbf{p}_3$ when $u = 1$. Control points $\mathbf{p}_1$ and $\mathbf{p}_2$ are most influential when $u = 1/3$ and $2/3$, respectively.

For any Bézier curve, each control point $\mathbf{p}_i$ is weighted by its associated basis function. As we have seen, when $u = 0$, $\mathbf{p}_0$ is given a weight of 1.0, and $\mathbf{p}_1$ through $\mathbf{p}_n$ a weight of zero. Less weight is given to $\mathbf{p}_0$ and more to each succeeding $\mathbf{p}_i$ as u increases, reaching a maximum weight for each $\mathbf{p}_i$ when u becomes i/n. Then all other weights decay gradually to 0 as the weight of $\mathbf{p}_n$ reaches 1 when $u = 1$. We observe a shift in the influence of each point (each polygon vertex) as the parametric variable moves through its range from 0 to

1. This means that changing the position of control point $\mathbf{p}_i$ has the greatest influence on the curve's shape at and near the parameter value i/n.

The ability to modify a Bézier curve is demonstrated in Figure 4.7. In Figure 4.7a, moving point $\mathbf{p}_1$ to $\mathbf{p}_1'$ pulls the curve toward that vertex. The effect can be weaker or stronger depending on the distance (or direction) the point is moved. In Figure 4.7b, we see evidence of an interesting degree of freedom. By specifying multiply coincident control points at a vertex, we pull the curve in closer and closer to that vertex. To do this, a curve-generating program might simply require us to specify a positive integer for each vertex. The integer indicates the number of coincident points at a vertex. By adding points coincident with existing vertex points, we increase the degree of the polynomial representing the curve, in this case without changing the number of sides and shape of the characteristic polygon.

If the first and last points of the characteristic polygon coincide, the arrangement produces a closed curve. The closed curve in Figure 4.8a has C^1 continuity where the ends of the curve join together at $\mathbf{p}_0$ and $\mathbf{p}_5$. This continuity is produced by making points $\mathbf{p}_1$, $\mathbf{p}_0$, $\mathbf{p}_5$, and $\mathbf{p}_4$ collinear. A similar condition exists in Figure 4.8c. However, the curve in Figure 4.8b exhibits only C^0 continuity.

The shape of a Bézier curve is unaffected by the direction in which we traverse the control points. We are free to order them in two ways: $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n$ or $\mathbf{p}_n, \mathbf{p}_{n-1}, \ldots, \mathbf{p}_0$. The curve looks the same either way; only the direction of parameterization is reversed. We express this property as

$$\sum_{i=0}^{n} \mathbf{p}_i B_{i,n}(u) = \sum_{i=0}^{n} \mathbf{p}_{n-i} B_{i,n}(1-u)$$

The proof of this follows from the basis function identity $B_{i,n}(u) = B_{n-i,n}(1-u)$.

Fowler and Bartels (1993) describe a curve-shaping method that permits direct manipulation of certain geometric properties at any selection of points on a curve of arbitrary degree and basis. The method affects the shape of a curve at one or more points by establishing and solving an underdetermined system of constraint equations. The constraints correspond to position, tangency, or curvature conditions at the points.

The convex hull property expedites determination of the intersection of two Bézier curves. It is a simple matter to find the min-max box enclosing each control polygon and then to check for an intersection between the two boxes. If the boxes do not intersect, then neither do the curves. This preliminary test significantly increases the speed and efficiency of computing intersections.

Figure 4.7 Cubic Bézier curves and their modification.

Figure 4.8 Closed Bézier curves.

Degree Elevation

Adding another control point to the definition of a Bézier curve raises its degree by one. We are motivated to do this if we are not satisfied with the original curve and need more freedom to achieve an acceptable shape. It is best to add this point in a way that leaves the shape of the curve unchanged (see Figure 4.9). After adding the point, we may move any of the points, including the new addition, to alter the shape. Because the new set of control points ${}^1\mathbf{p}_i$ must initially generate the same curve as the original set, it must be true that

$$\sum_{i=0}^{n+1} {}^1\mathbf{p}_i B_{i,n+1}(u) = \sum_{i=0}^{n} \mathbf{p}_i B_{i,n}(u)$$

or

$$\sum_{i=0}^{n+1} {}^1\mathbf{p}_i \binom{n+1}{i} u^i (1-u)^{n+1-i} = \sum_{i=0}^{n} \mathbf{p}_i \binom{n}{i} u^i (1-u)^{n-i}$$

Multiplying the right side of this equation by $u + (1-u)$ produces

Figure 4.9 Degree elevation: $n = 3$ to $n = 4$.

$$\sum_{i=0}^{n+1} {}^1\mathbf{p}_i \binom{n+1}{i} u^i (1-u)^{n+1-i} = \sum_{i=0}^{n} \mathbf{p}_i \binom{n}{i} [u^i (1-u)^{n+1-i} + u^{i+1}(1-u)^{n-i}]$$

The summation on the right side of this equation produces $2(n+1)$ terms and on the left $n+2$ terms. We rearrange and group terms on the right side so that we can compare and equate the coefficients of $u^i (1-u)^{n+1-i}$ on both sides. Doing this, we find

$${}^1\mathbf{p}_i \binom{n+1}{i} = \mathbf{p}_i \binom{n}{i} + \mathbf{p}_{i-1} \binom{n}{i-1}$$

Expanding the binomial coefficients and simplifying the results yields

$${}^1\mathbf{p}_i = \left(\frac{i}{n+1}\right)\mathbf{p}_{i-1} + \left(1 - \frac{i}{n+1}\right)\mathbf{p}_i \qquad i = 0, \ldots, n+1 \tag{4.10}$$

This tells us that we can compute a new set of control points ${}^1\mathbf{p}_i$ using the original control points and piecewise linear interpolation at the parameter values $i/(n+1)$. Figure 4.9 shows the results of raising the degree of a cubic Bézier curve while maintaining the original shape. Clearly, after raising the degree and before moving any control points to adjust its shape, the curve is defined by more information than necessary.

It may be necessary to repeat this process several times to establish sufficient modeling flexibility. The general formula for producing a new set of control points for r degree elevations follows directly from the previous discussion:

$$ {}^{r}\mathbf{p}_i = \sum_{j=0}^{n} \mathbf{p}_j \binom{n}{j} \frac{\binom{r}{i-j}}{\binom{n+r}{i}} \tag{4.11} $$

Kocic (1991) presents a degree elevation technique that is a generalized form of this method. The technique is iterative and can be applied to tensor products and triangular Bézier patches, as well as to curves.

Degree reduction usually is not a good idea. However, approximations based upon a reversal of the process above are possible under special circumstances. Watkins and Worsey (1988) have developed a method that combines subdivision with degree reduction. An important use of such a procedure is in intersection computations and in computer graphics for use in rendering.

4.3 TRUNCATING AND SUBDIVIDING

Truncating a Bézier curve is necessary when the modeling process itself eliminates parts of a curve that no longer belong to the active model. Subdividing techniques are used to expedite the computation of points on a curve or to increase the number of control points by creating small curve segments without initially changing the shape. We will look at a recursive subdivision method and a subdivision by geometric construction.

Truncating a Second-Degree Curve

The truncation and subsequent reparameterization transformation for a second-degree Bézier curve proceeds as follows: Given the three control points defining the curve, we can find three new control points defining a segment of this curve in the parametric interval $u \in [u_i, u_j]$, where the interval is normalized to a new parameter v so that $v \in [0,1]$ (see Figure 4.10). In matrix form we write

$$ \mathbf{p}(u) = \mathbf{U}\mathbf{M}_B\mathbf{P} $$

where

$$ \mathbf{U} = [u^2 \quad u \quad 1] $$

$$ \mathbf{M}_B = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} $$

Figure 4.10 Truncating a second-degree Bézier curve.

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix}$$

Let $\mathbf{U} = \mathbf{VT}$, where $\mathbf{V} = [v^2 \quad v \quad 1]$, so that

$$\mathbf{p}(v) = \mathbf{VTM}_B\mathbf{P}$$

$$= \mathbf{VM}_B\mathbf{P}'$$

This means that

$$\mathbf{TM}_B\mathbf{P} = \mathbf{M}_B\mathbf{P}' \qquad \text{and} \qquad \mathbf{P}' = \mathbf{M}_B^{-1}\mathbf{TM}_B$$

where

$$\mathbf{T} = \begin{bmatrix} \Delta u_i^2 & 0 & 0 \\ 2u_i\Delta u & \Delta u_i & 0 \\ u_i^2 & u_i & 1 \end{bmatrix}$$

(see the development of Equation [3.28], et seq.) and where

$$\mathbf{M}_B^{-1} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & \frac{1}{2} & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

so that

$$\mathbf{M}_B^{-1}\mathbf{T}\mathbf{M}_B = \begin{bmatrix} u_i^2 - 2u_i + 1 & -2u_i^2 + 2u_i & u_i \\ u_i^2 - 2u_i + 1 + \Delta u_i(u_i - 1) & -2u_i^2 + 2u_i + \Delta u_i(-2u_i + 1) & u_i^2 + u_i\Delta u_i \\ u_i^2 - 2u_i + 1 + \Delta u_i^2 + 2\Delta u_i(u_i - 1) & -2u_i^2 + 2u_i - 2\Delta u_i^2 - 4u_i\Delta u_i + 2\Delta u_i & u_i^2 + 2u_i\Delta u_i + \Delta u_i^2 \end{bmatrix} \tag{4.12}$$

The control points for the truncated segment are

$$\mathbf{P}' = [\mathbf{p}'_0 \quad \mathbf{p}'_1 \quad \mathbf{p}'_2]^T$$

where

$$\mathbf{p}'_0 = (1 - u_i)^2\,\mathbf{p}_0 + 2u_i\,(1 - u_i)\mathbf{p}_1 + u_i^2\mathbf{p}_2$$

$$\mathbf{p}'_1 = (1 - u_i)(1 - u_j)\mathbf{p}_0 + (-2u_iu_j + u_j + u_i)\mathbf{p}_1 + u_iu_j\mathbf{p}_2$$

$$\mathbf{p}'_2 = (1 - u_j)^2\,\mathbf{p}_0 + 2u_j(1 - u_j)\mathbf{p}_1 + u_j^2\mathbf{p}_2$$

For the special case of dividing this second-degree curve into two segments at $u = 0.5$, we have for $u_i = 0$, $u_j = 0.5$

$$\mathbf{M}_B^{-1}\mathbf{T}\mathbf{M}_B = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 0.5 & 0 \\ 0.25 & 0.5 & 0.25 \end{bmatrix} \tag{4.13}$$

and for $u_i = 0.5$, $u_j = 1$

$$\mathbf{M}_B^{-1}\mathbf{T}\mathbf{M}_B = \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.14}$$

Thus, the control-point transformations are

$$\mathbf{P}'_{0\to 0.5} = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 0.5 & 0 \\ 0.25 & 0.5 & 0.25 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0 \\ 0.5(\mathbf{p}_0 + \mathbf{p}_1) \\ 0.25(\mathbf{p}_0 + 2\mathbf{p}_1 + \mathbf{p}_2) \end{bmatrix} = \begin{bmatrix} \mathbf{p}'_{0,0\to 0.5} \\ \mathbf{p}'_{1,0\to 0.5} \\ \mathbf{p}'_{2,0\to 0.5} \end{bmatrix}$$

and

$$\mathbf{P}'_{0.5\to 1} = \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix} = \begin{bmatrix} 0.25(\mathbf{p}_0 + 2\mathbf{p}_1 + \mathbf{p}_2) \\ 0.5(\mathbf{p}_0 + \mathbf{p}_1) \\ \mathbf{p}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{p}'_{0,0.5\to 1} \\ \mathbf{p}'_{1,0.5\to 1} \\ \mathbf{p}'_{2,0.5\to 1} \end{bmatrix}$$

As expected, we see that $\mathbf{p}'_{0,0.5\to 1} = \mathbf{p}'_{1,0\to 0.5}$ and that $\mathbf{p}'_{0,0\to 0.5} = \mathbf{p}'_0$ and $\mathbf{p}'_{2,0.5\to 1} = \mathbf{p}'_2$.

Truncating a Cubic Curve

Truncating and reparameterizing a cubic Bézier curve proceeds in the same way as outlined above for the second-degree curve. Given the four control points defining the curve, we find four new control points defining a segment of this curve in the parametric interval $u \in [u_i, u_j]$, where the interval is normalized to the new parameter v such that $v \in [0,1]$ (see Figure 4.11). Again, working in matrix form, we have the general expression $\mathbf{p}(u) = \mathbf{U}\mathbf{M}_B\mathbf{P}$, but now

$$\mathbf{U} = [u^3 \quad u^2 \quad u \quad 1]$$

$$\mathbf{M}_B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

If we let $\mathbf{U} = \mathbf{V}\mathbf{T}$, where

$$\mathbf{V} = [v^3 \quad v^2 \quad v \quad 1]$$

then, again,

$$\mathbf{p}(u) = \mathbf{V}\mathbf{T}\mathbf{M}_B\mathbf{P} = \mathbf{V}\mathbf{M}_B\mathbf{P}'$$

Figure 4.11 Truncating a cubic Bézier curve.

This means that

$$\mathbf{T}\mathbf{M}_B\mathbf{P} = \mathbf{M}_B\mathbf{P}' \qquad \text{and} \qquad \mathbf{P}' = \mathbf{M}_B^{-1}\mathbf{T}\mathbf{M}_B\mathbf{P}$$

where

$$\mathbf{T} = \begin{bmatrix} \Delta u_i^3 & 0 & 0 & 0 \\ 3u_i\Delta u_i^2 & \Delta u_i^2 & 0 & 0 \\ 3u_i^2\Delta u_i & 2u_i\Delta u_i & \Delta u_i & 0 \\ u_i^3 & u_i^2 & u_i & 1 \end{bmatrix}$$

(see the development of Equation [3.28] et seq.) and where

$$\mathbf{M}_B^{-1} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{3} & 1 \\ 0 & \frac{1}{3} & \frac{2}{3} & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

so that

$$\mathbf{M}_B^{-1}\mathbf{T}\mathbf{M}_B = \begin{bmatrix} (1-u_i)^3 & 3u_i(1-u_i)^2 & 3u_i^2(1-u_i) & u_i^3 \\ (1-u_i)^3 - (1-u_i)^2\Delta u_i & 3u_i(1-u_i)^2 + (1-3u_i)(1-u_i)\Delta u_i & 3u_i^2(1-u_i) + u_i(2-3u_i)\Delta u_i & u_i^2(u_i+\Delta u_i) \\ (1-u_i)^3 - 2(1-u_i)^2\Delta u_i + (1-u_i)\Delta u_i^2 & 3u_i(1-u_i)^2 + 2(1-3u_i)(1-u_i)\Delta u_i - (2-3u) & 3u_i^2(1-u_i) + 2u_i(2-3u_i)\Delta u_i + (1-3u_i)\Delta u_i^2 & u_i(u_i+\Delta u_i)^2 \\ (1-u_i)^3 - 3(1-u_i)^2\Delta u_i + 3(1-u_i)\Delta u_i^2 - \Delta u_i^3 & 3u_i(1-u_i)^2 + 3(1-3u_i)(1-u_i)\Delta u_i - 3(2-3u_i)\Delta & 3u_i^2(1-u_i) + 3u_i(2-3u_i)\Delta u_i + 3(1-3u_i)\Delta u_i^2 - 3\Delta u_i^3 & (u_i+\Delta u_i)^3 \end{bmatrix} \tag{4.15}$$

The control points for the truncated segment are

$$\mathbf{P}' = [\mathbf{p}'_0 \quad \mathbf{p}'_1 \quad \mathbf{p}'_2 \quad \mathbf{p}'_3]^T$$

where

$$\mathbf{p}_0' = (1-u_i)^3\,\mathbf{p}_0 + 3u_i(1-u_i)^2\,\mathbf{p}_1 + 3u_i^2(1-u_i)\mathbf{p}_2 + u_i^3\mathbf{p}_3$$

$$\mathbf{p}_1' = (1-u_i)^2\,(1-u_j)\mathbf{p}_0 + (1-u_i)(2u_i + u_j - 3u_iu_j)\mathbf{p}_1$$
$$+\,u_i(2u_j + u_i - 3u_iu_j)\mathbf{p}_2 + u_i^2u_j\mathbf{p}_3$$

$$\mathbf{p}_2' = (1-u_i)(1-u_j)^2\,\mathbf{p}_0 + (1-u_j)(2u_i + u_j - 3u_iu_j)\mathbf{p}_1$$
$$+\,u_j(2u_i + u_j - 3u_iu_j)\mathbf{p}_2 + u_iu_j^2\mathbf{p}_3$$

$$\mathbf{p}_3' = (1-u_j)^3\,\mathbf{p}_0 + 3u_j\,(1-u_j)^2\,\mathbf{p}_1 + 3u_j^2(1-u_j)\mathbf{p}_2 + u_j^3\mathbf{p}_3$$

Recursive Subdivision

Recursively subdividing a Bézier curve s times produces a set of points on the curve at 2^s intervals. The parameter length of each is approximately $1/2^s$. For computer-graphics applications subdivision continues until the segments are small enough to plot as straight line segments or to display as a single pixel. We begin by finding $\mathbf{p}(0.5)$ on the initial curve, dividing it into two smaller curves. Figure 4.12 shows the results of the first subdivision of a cubic curve. This gives us only the end control points. To find all the control points for each of these two smaller curves, we perform the necessary parameter transformations, repeating the subdivision as often as necessary. Here is an algebraic description of this process:

Step 1. First, we compute $\mathbf{p}(0.5) = \mathbf{p}_0/8 + 3\mathbf{p}_1/8 + 3\mathbf{p}_2/8 + \mathbf{p}_3/8$.

Step 2. Then we use the general parameter transformation for segment $u \in [0,0.5]$ to compute $\mathbf{P}_a'$. We have

$$\mathbf{P}_a' = \mathbf{M}_B^{-1}\mathbf{T}_a\mathbf{M}_B\mathbf{P}$$

where, from Equation (4.15),

$$\mathbf{M}_B^{-1}\mathbf{T}_a\mathbf{M}_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 \\ \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{bmatrix}$$

Figure 4.12 Recursive subdivision of a cubic Bézier curve.

so that

$$\mathbf{P}'_a = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 \\ \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0 \\ \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1) \\ (\mathbf{p}_0 + 2\mathbf{p}_1 + \mathbf{p}_2) \\ \frac{1}{8}(\mathbf{p}_0 + 3\mathbf{p}_1 + 3\mathbf{p}_2 + \mathbf{p}_3) \end{bmatrix}$$

Step 3. Finally, we use the general parameter transformation again, this time for segment $u \in [0.5,1]$, to compute its $\mathbf{P}'_b$. We have

$$\mathbf{P}'_b = \mathbf{M}_B^{-1}\mathbf{T}_b\mathbf{M}_B\mathbf{P}$$

and

$$\mathbf{M}_B^{-1}\mathbf{T}_b\mathbf{M}_B = \begin{bmatrix} \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \\ 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Therefore,

$$\mathbf{P}'_b = \begin{bmatrix} \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \\ 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{8}(\mathbf{p}_0 + 3\mathbf{p}_1 + 3\mathbf{p}_2 + \mathbf{p}_3) \\ \frac{1}{4}(\mathbf{p}_1 + 2\mathbf{p}_2 + \mathbf{p}_3) \\ \frac{1}{2}(\mathbf{p}_2 + \mathbf{p}_3) \\ \mathbf{p}_3 \end{bmatrix} \tag{4.16}$$

All that really interests us computationally in Steps 2 and 3 are the intermediate control points, because we already know the endpoints of each segment. Now we can find the midpoints of each of these two curve segments by repeating this process.

Subdivision by Geometric Construction

Bézier devised a subdivision technique by geometric construction to find a point on a curve corresponding to a given value of the parameter u_i (see Figure 4.13). First, we find the point on each edge of the control polygon that subdivides it proportionally according the value of u_i and connect these points to form a set of line segments. We subdivide each of these new line segments proportionally according to u_i and repeat the subdivision and line segment construction until we can construct only one line segment. The point that proportionally subdivides this line is the point on the curve corresponding to u_i. The curve in Figure 4.13 constructs the point at $u = 0.25$. Points labeled 0 are the initial control points. Those labeled 1, 2, 3, 4 are subsequent sequential subdivisions. In general there are n cycles of subdivision, and the number of line segments decreases by one each cycle. Figure 4.14

Figure 4.13 Subdivision by geometric construction.

Figure 4.14 Construction of the point $u = 0.5$.

shows the construction of $u = 0.5$ for a cubic Bézier curve. This construction is described by some vector algebra and a simple recursion algorithm. The first cycle of recursion operates on the initial control points, as shown in Figure 4.15.

4.4 COMPOSITE BÉZIER CURVES

Bézier curves may be joined end-to-end to form a composite curve. This allows us to create a more complex curve without having to raise the degree of an equivalent single curve. The geometric relationship of the control points adjacent to the joints determines the continuity conditions at those

Figure 4.15 A recursion algorithm.

joints. To explore these ideas, we again focus on the cubic Bézier curve and extend the continuity criteria discussed earlier (see Figure 4.16).

First, we differentiate Equation (4.5) and rearrange terms to obtain

$$\mathbf{p}^u(u) = (-3 + 6u - 3u^2)\mathbf{p}_0 + (3 - 12u + 9u^2)\mathbf{p}_1 + (6u - 9u^2)\mathbf{p}_2 + 3u^2\mathbf{p}_3$$

Next, we evaluate this function at $u = 0$ and $u = 1$, producing

$$\mathbf{p}_0^u = 3(\mathbf{p}_1 - \mathbf{p}_0) \qquad \text{and} \qquad \mathbf{p}_1^u = 3(\mathbf{p}_3 - \mathbf{p}_2)$$

These expressions demonstrate that the tangent at $u = 0$ is determined by the line between $\mathbf{p}_0$ and $\mathbf{p}_1$, and at $u = 1$ by the line between $\mathbf{p}_2$ and $\mathbf{p}_3$. They also reveal that certain continuity conditions between adjacent Bézier segments of a composite curve are relatively simple to specify. Thus, if we define one segment by control points $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{m-1}, \mathbf{p}_m$ and the adjacent segment by control points $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{n-1}, \mathbf{q}_n$, then G^1 continuity is established at $\mathbf{p}_u = \mathbf{q}_0$ by making $\mathbf{p}_{m-1}, \mathbf{p}_m, \mathbf{q}_0$, and $\mathbf{q}_1$ collinear.

Figure 4.17 shows two cubic Bézier curves joined with G^1 continuity (we do not address parametric continuity here). We are free to move point D so long as we also move C and E in a way that preserves the collinearity of C,

Figure 4.16 Composite Bézier curves.

Figure 4.17 Two cubic Bézier curves joined with G^1 continuity.

D, and *E*. Within this constraint, we may change the lengths of *C*,*D*, and *E*. The effects do not propagate beyond points *A* and *G* in the event that other curve segments are present or are added to this string.

A condition for G^2 continuity at a joint between curves is that the five vertices $\mathbf{p}_{m-2}$, $\mathbf{p}_{m-1}$, $\mathbf{p}_m = \mathbf{q}_0$, $\mathbf{q}_1$, and $\mathbf{q}_2$ must be coplanar. To achieve this might require increasing the number of control points in one or more segments. The equations for the curvature κ_0 and κ_1 at each end of a cubic Bézier curve follow without derivation.

$$\kappa_0 = \frac{2|(\mathbf{p}_1 - \mathbf{p}_0) \times (\mathbf{p}_2 - \mathbf{p}_1)|}{3|\mathbf{p}_1 - \mathbf{p}_0|^3}$$

$$\kappa_1 = \frac{2|(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_2)|}{3|\mathbf{p}_3 - \mathbf{p}_2|^3}$$

We can easily generalize these equations for Bézier curves with $n > 3$.

The advantage of higher-order Bézier curves is that they provide correspondingly higher orders of continuity between segments of composite curves. For example, a fifth-order, or quintic, Bézier curve permits us to specify endpoints, end tangents, and curvature at each end. However, we must consider how higher-degree polynomial functions affect the computation of geometric properties and relationships. If we have quintic curves and the

related biquintic surfaces, for example, the intersection computations for these are more complex than for cubic curves and bicubic surfaces.

4.5 RATIONAL BÉZIER CURVES

Equation (4.2) defines a Bézier curve using an integral or nonrational polynomial. We now discuss a Bézier curve defined by a rational polynomial. We can best study this curve by thinking of it as a projection of a four-dimensional nonrational curve onto three-dimensional space. To understand the underlying geometry and representational power of this formulation, we must review some elementary properties of homogeneous coordinates and projective geometry. Riesenfeld (1981) discusses the use of homogeneous coordinates and projective planes in computer graphics. Roberts (1965) was one of the first to observe that homogeneous coordinates are a useful way to represent geometric transformations and projections.

Every point x,y,z in three-dimensional Cartesian space has a corresponding set of homogeneous coordinates hx,hy,hz,h in four-dimensional projective space. There are an infinite number of points in this four-dimensional space corresponding to each point in ordinary space, because $x = hx/h$, $y = hy/h$, and $z = hz/h$ for all real h except $h = 0$. All the points on a line through the origin in this four-dimensional homogeneous space have identical ordinary 3D space coordinates. Thus, the point x,y,z is the projection of the point hx,hy,hz,h onto the hyperplane $h = 1$, where the origin is the center of projection and the hyperplane is ordinary three-dimensional space. Figures 4.18a and 4.18b illustrate homogeneous coordinate geometry in one and two dimensions.

We define a Bézier curve in the homogeneous coordinate space of four dimensions and find its projection onto the $h = 1$ hyperplane, that is, onto ordinary three-dimensional space. The equation of this curve is

$$\tilde{\mathbf{p}}(u) = \sum_{i=0}^{n} \tilde{\mathbf{p}}_i B_{i,n}(u) \tag{4.17}$$

where $\tilde{\mathbf{p}}(u)$ is the four-component vector of a point on the curve, and the $\tilde{\mathbf{p}}_i$ are the control points, also given in four-dimensional homogeneous coordinates. We know that

$$\tilde{\mathbf{p}}(u) = \begin{bmatrix} \tilde{x}(u) \\ \tilde{y}(u) \\ \tilde{z}(u) \\ h(u) \end{bmatrix} = \begin{bmatrix} h(u)x(u) \\ h(u)y(u) \\ h(u)z(u) \\ h(u) \end{bmatrix} \tag{4.18}$$

Figure 4.18 Homogeneous coordinate geometry in one and two dimensions.

so that now we can write, for the homogeneous coordinates of any point on the curve,

$$\tilde{x}(u) = \sum \tilde{x}_i B_{i,n}(u)$$
$$\tilde{y}(u) = \sum \tilde{y}_i B_{i,n}(u)$$
$$\tilde{z}(u) = \sum \tilde{z}_i B_{i,n}(u)$$
$$h(u) = \sum h_i B_{i,n}(u) \tag{4.19}$$

For the projection of this point onto x,y,z space we write

$$x(u) = \frac{\tilde{x}(u)}{h(u)}$$

$$= \frac{\sum \tilde{x}_i B_{i,n}(u)}{\sum h_i B_{i,n}(u)} \tag{4.20}$$

But $\tilde{x}_i = h_i x_i$, so that

$$x(u) = \frac{\sum h_i x_i B_{i,n}(u)}{\sum h_i B_{i,n}(u)} \tag{4.21}$$

Because analogous expressions apply to $y(u)$ and $z(u)$, we represent the rational Bézier curve in vector form as

$$\mathbf{p}(u) = \frac{\sum h_i \mathbf{p}_i B_{i,n}(u)}{\sum h_i B_{i,n}(u)} \tag{4.22}$$

where the $\mathbf{p}_i$ are the control points in three-dimensional space and the h_i are the corresponding projective space coordinates; the h_i are also known as the *weights*. Equation (4.22) reduces to an integral or nonrational Bézier curve when all $h_i = 1$, which follows from: If all $h_i = 1$, then $\sum h_i B_{i,n}(u) = 1$, because $\sum B_{i,n}(u) = 1$ and

$$\frac{\sum h_i \mathbf{p}_i B_{i,n}}{\sum h_i B_{i,n}} = \sum \mathbf{p}_i B_{i,n}(u) \tag{4.23}$$

We immediately see that for a rational Bézier curve the same set of control points can produce many different shapes, interpolating the same endpoints and with the same end tangents. Changing one or more of the weights changes the shape of the curve. This means, for example, that a rational cubic Bézier curve has four more degrees of freedom than the nonrational cubic.

Let us see how this works for a second-degree rational Bézier curve in the plane with control points $\mathbf{p}_0,\mathbf{p}_1,\mathbf{p}_2$ and weights h_1,h_2,h_3. In Figure 4.19, we see plotted in the $h = 1$ plane the rational Bézier curve for all $h_i = 1$. It is identical to the nonrational Bézier curve with the same control points. All the points on the line OA have the same x,y coordinates in the $h = 1$ plane, as do the points on OB and OC. This is because of the relationship we established earlier between x,y and h. When we change a weight, the corresponding control point slides along its projection line in homogeneous space to a position whose h coordinate matches the new weighted value. For the case of h_1, shown in the figure, the control point moves along line OB to the position

Figure 4.19 Rational Bézier curve in the plane.

h_1x_1, h_1y_1, h. We can think of this control point in its new position together with the two unchanged end control points as defining a new Bézier curve in the homogeneous or projective space. It is the projection of this curve onto the $h = 1$ plane that in turn produces a new curve in that plane. This is the curve of interest. It has, of course, the same control points as the $h_i = 1$ curve and the same end slopes. Only one weight has been changed, and the effect of this is to pull the curve toward the control point $\mathbf{p}_i$, whose weight we changed. All of this applies equally to three-dimensional rational Bézier curves of any degree.

A rational Bézier curve is invariant under a perspective transformation. If its control points are subjected to this transformation, then all other points are produced as usual from the transformed control points. The resulting curve is the accurate perspective transformation of the original. This is not true for the perspective transformation of a nonrational curve, where every point on the curve must be subjected to the transformation in order to produce an accurate perspective image.

Using the rational Bézier formulation, we can accurately represent conic curves. The Hermite basis and nonrational Bézier forms cannot do this.

Because conics are second-degree curves, only three control points and their associated weights are required to represent them. Thus, from

$$\mathbf{p}(u) = \frac{h_0\mathbf{p}_0 B_{0,2}(u) + h_1\mathbf{p}_1 B_{1,2}(u) + h_2\mathbf{p}_2 B_{2,2}(u)}{h_0 B_{0,2}(u) + h_1 B_{1,2}(u) + h_2 B_{2,2}(u)} \tag{4.24}$$

we can generate segments of circles, ellipses, hyperbolas, and parabolas. If the $h_i = 1$, then segments of parabolas are produced. Other weight values produce the other conics. Space does not permit a full discussion of this topic. However, these representations are also discussed in the section on rational B-Splines (see Section 5.7).

Let us review the properties of a Bézier curve that make it an unusually effective modeling tool. First, the curve has endpoints in common with the polygon (the other vertices are usually not on the curve). Second, the slope of the tangent vectors at the endpoints equals the slope of the first and last segments of the polygon. Third, the curve lies entirely within the convex hull defined by the extreme points of the polygon and generally mimics the gross features of the polygon. Fourth, Bézier curves are variation-diminishing. This means that they never oscillate wildly away from their defining control points (they are smoother than the data on which they are based). Fifth, compared to conventional polynomials or splines, Bézier curves do not require us to input slopes, just data points. Finally, the parametric formulation allows a curve to represent multiple-valued shapes. In fact, as we have seen, if the first and last points coincide (that is, if $\mathbf{p}_0 = \mathbf{p}_n$), then the curve is closed. Bézier curves are special cases of the more general B-Spline curves that we consider next.

5

B-SPLINE CURVES

A B-Spline curve differs from a Hermite or Bézier curve in that it usually consists of more than one curve segment. Each segment is defined and influenced by only a few control points, which are the coefficients of the B-Spline basis function polynomials. The degree of the curve is independent of the total number of control points. These characteristics allow changes in shape that do not propagate beyond one or only a few local segments. Most curve-defining techniques do not provide for local control of shape. Consequently, local changes (for example, a small change in the position of a point on a spline curve or of a vertex of a characteristic polygon of a Bézier curve) tend to be strongly propagated throughout the entire curve. This is sometimes described as a *global propagation of local change*. The B-Spline curve avoids this problem by using a special set of basis functions that has only local influence and depends on only a few neighboring control points. Connecting the control points $\mathbf{p}_i$ (see Equation [5.1]) in the order of their numbering with straight lines produces the B-Spline control polygon. The B-Spline curve is contained within the convex hull of its control polygon. In general, B-Splines do not necessarily interpolate their endpoints. However, the nonuniform B-Spline basis functions allow this. We consider here both nonrational and rational forms of the B-Spline basis functions.

5.1 NONUNIFORM B-SPLINE BASIS FUNCTIONS

We begin with the most general nonrational B-Spline curves, those defined by nonuniform basis functions. That is, the basis function defining one seg-

ment may differ from those defining another. This allows us to interpolate one or more of the control points, depending on the modeling situation. We do this by specifying a nondecreasing set of knot values (explained in the following). A B-Spline curve is similar to a Bézier curve in that a set of basis functions combines the effects of $n+1$ control points $\mathbf{p}_i$. The nonrational form is given by

$$\mathbf{p}(u) = \sum_{i=0}^{n} \mathbf{p}_i N_{i,K}(u) \tag{5.1}$$

Compare Equation (5.1) to Equation (4.2). For a Bézier curve, the number of control points determines the degree of the basis function polynomial. For a B-Spline curve, a parameter K controls the degree $(K-1)$ of the basis polynomials, and it is usually independent of the number of control points, except as it is limited by Equation (5.6), following.

The nonuniform B-Spline basis functions are defined recursively by the following expressions:

$$\begin{aligned} N_{i,1}(u) &= 1 \quad \text{if} \quad t_i \le u < t_{i+1} \\ &= 0 \quad \text{otherwise} \end{aligned} \tag{5.2}$$

and

$$N_{i,k}(u) = \frac{(u-t_i)N_{i,k-1}(u)}{t_{i+k-1}-t_i} + \frac{(t_{i+k}-u)N_{i+1,k-1}(u)}{t_{i+k}-t_{i+1}} \tag{5.3}$$

for integer values of k: $k = 2, \ldots, K$ where K, the order of the B-Spline, controls the degree $(K-1)$ of the resulting polynomial in u and also the continuity of the curve. The t_j are called *knot values*, and a set of knot values comprises a *knot vector*. They relate the parametric variable u to the $\mathbf{p}_i$ control points where $i = 0, \ldots, n$. For an open nonuniform curve that interpolates the endpoints, the t_j are calculated once using K:

$$\begin{aligned} t_j &= 0 && \text{if} \quad j < K \\ t_j &= j - K + 1 && \text{if} \quad K \le j \le n \\ t_j &= n - K + 2 && \text{if} \quad j > n \end{aligned} \tag{5.4}$$

for integer values of j: $j = 0, \ldots, n+K$.

Note that once the t_j are calculated for K, they are used to compute $N_{i,k}$ for all $k = 1, \ldots, K$. The index j on knot values t_j ranges from 0 to $(n+K)$. The knot values t_j themselves take on values from 0 to $(n-K+2)$. The index i on blending function $N_{i,k}$ ranges from 0 to n. There are always $n+1$ blending functions for each k.

The parameters determining the number of control points, knots, and the degree of the polynomial are related by

$$n + K + 1 = T \tag{5.5}$$

where T is the number of knots. For nonuniform and open B-Spline curves, the knot vector $\mathbf{T}$ is characterized by

$$\mathbf{T} = [\alpha,\alpha,\ldots,\alpha,t_K,\ldots,t_{T-K-1},\beta,\beta,\ldots,\beta]$$

where end knots α and β repeat with multiplicity K.

If the entire curve is parameterized over the unit interval, then, for most situations, $\alpha = 0$ and $\beta = 1$. However, if we assign nondecreasing integer values of the parameter to the knots, then $\alpha = 0$ and $\beta = n - K + 2$.

Spacing the knots at equal intervals of the parameter describes a uniform nonrational B-Spline curve; otherwise it is nonuniform. Usually we assign integer values to the parametric variable at the knots, although this is by no means a requirement. However, it greatly simplifies the discussion of this subject. So, we will define the range of the parametric variable u to be

$$0 \le u \le n - K + 2 \tag{5.6}$$

Because the denominators in Equation (5.3) ean be zero, we must define $0/0 = 0$.

Multiple or repeated knot-vector values, or multiply coincident control points, induce discontinuities. For a cubic curve, a double knot defines a join point with curvature discontinuity. A triple knot produces a corner point in the curve.

The basis functions $N_{i,1}(u)$, $N_{i,2}(u)$, and $N_{i,3}(u)$ corresponding to $K = 1$, $K = 2$, and $K = 3$, given six control points ($n = 5$), we compute as follows. For the $N_{i,1}(u)$ basis functions with $n = 5$ and $K = 1$, we find that $0 \le u \le 6$ and the knot vector is [0,1,2,3,4,5,6], or

$$\begin{aligned} t_0 &= 0 & t_4 &= 4 \\ t_1 &= 1 & t_5 &= 5 \\ t_2 &= 2 & t_6 &= 6 \\ t_3 &= 3 \end{aligned} \tag{5.7}$$

This is a uniform knot spacing. From Equation (5.2), we obtain directly

$$\begin{aligned} N_{0,1}(u) &= 1 \quad \text{if} \quad 0 \le u < 1 \\ &= 0 \quad \text{otherwise} \\ N_{1,1}(u) &= 1 \quad \text{if} \quad 1 \le u < 2 \\ &= 0 \quad \text{otherwise} \\ N_{2,1}(u) &= 1 \quad \text{if} \quad 2 \le u < 3 \\ &= 0 \quad \text{otherwise} \end{aligned}$$

$$N_{3,1}(u) = 1 \quad \text{if} \quad 3 \le u < 4$$
$$= 0 \quad \text{otherwise}$$

$$N_{4,1}(u) = 1 \quad \text{if} \quad 4 \le u < 5$$
$$= 0 \quad \text{otherwise}$$

$$N_{5,1}(u) = 1 \quad \text{if} \quad 5 \le u \le 6$$
$$= 0 \quad \text{otherwise}$$

These six basis functions are plotted in Figure 5.1. They are obviously identical in form and limited to a different successive unit interval of the parametric variable. Applying these basis functions to any set of six control points using Equation (5.2), we see that $\mathbf{p}(u)$ is given by a different equation for each unit interval, so that

$$\mathbf{p}(u) = \mathbf{p}_0 \qquad \text{for} \qquad 0 \le u < 1$$

Figure 5.1 B-Spline basis functions for $n = 5$, $K = 1$.

$$\mathbf{p}(u) = \mathbf{p}_1 \quad \text{for} \quad 1 \le u < 2$$
$$\mathbf{p}(u) = \mathbf{p}_2 \quad \text{for} \quad 2 \le u < 3$$
$$\mathbf{p}(u) = \mathbf{p}_3 \quad \text{for} \quad 3 \le u < 4$$
$$\mathbf{p}(u) = \mathbf{p}_4 \quad \text{for} \quad 4 \le u < 5$$
$$\mathbf{p}(u) = \mathbf{p}_5 \quad \text{for} \quad 5 \le u < 6$$

The resulting curve is clearly not a curve in the ordinary sense of the word, but merely the initial set of control points. (We could also interpret this to represent six disjoint curve segments of zero length, each segment concentrated at a control point.)

Next, for the $N_{i,2}(u)$ basis functions with $n = 5$ and $K = 2$, we find that $0 \le u \le 5$ and the knot vector is [0,0,1,2,3,4,5,5], or

$$\begin{aligned} t_0 &= 0 & t_4 &= 3 \\ t_1 &= 0 & t_5 &= 4 \\ t_2 &= 1 & t_6 &= 5 \\ t_3 &= 2 & t_7 &= 5 \end{aligned} \tag{5.8}$$

This is a nonuniform knot spacing.

Equation (5.3) indicates that the $N_{i,2}(u)$ depend on the $N_{i,1}(u)$, so before we can compute the $N_{i,2}(u)$, we must first compute another set of $N_{i,1}(u)$ corresponding to the knot values in Equation (5.8). This computation produces

$$\begin{aligned} N_{0,1}(u) &= 1 \quad \text{if} \quad u = 0 \\ &= 0 \quad \text{otherwise} \\ N_{1,1}(u) &= 1 \quad 0 \le u < 1 \\ &= 0 \quad \text{otherwise} \\ N_{2,1}(u) &= 1 \quad 1 \le u < 2 \\ &= 0 \quad \text{otherwise} \\ N_{3,1}(u) &= 1 \quad 2 \le u < 3 \\ &= 0 \quad \text{otherwise} \\ N_{4,1}(u) &= 1 \quad 3 \le u < 4 \\ &= 0 \quad \text{otherwise} \\ N_{5,1}(u) &= 1 \quad 4 \le u < 5 \\ &= 0 \quad \text{otherwise} \end{aligned}$$

Substituting this set as appropriate into Equation (5.3) produces

$$N_{0,2}(u) = (1-u)N_{1,1}(u)$$
$$N_{1,2}(u) = uN_{1,1}(u) + (2-u)N_{2,1}(u)$$
$$N_{2,2}(u) = (u-1)N_{2,1}(u) + (3-u)N_{3,1}(u)$$
$$N_{3,2}(u) = (u-2)N_{3,1}(u) + (4-u)N_{4,1}(u)$$
$$N_{4,2}(u) = (u-3)N_{4,1}(u) + (5-u)N_{5,1}(u)$$
$$N_{5,2}(u) = (u-4)N_{5,1}(u)$$

The $N_{i,1}(u)$ functions act like switches, taking on a value of 1 or 0, depending on the value of u. They define the region over which the complete basis function is nonzero, or the interval of support for a curve's expression. These basis functions are plotted in Figure 5.2; they are identical in size and shape, except at the end segments spanning $0 \le u < 1$ and $4 \le u \le 5$. Applying these basis functions to any set of six control points by means of Equation (5.2), we find that $\mathbf{p}(u)$ is given by a different equation for each unit interval in u, through the action of the $N_{i,1}(u)$ switches, so that

$$\mathbf{p}(u) = (1-u)\mathbf{p}_0 + u\mathbf{p}_1 \qquad \text{for} \qquad 0 \le u < 1$$
$$\mathbf{p}(u) = (2-u)\mathbf{p}_1 + (u-1)\mathbf{p}_2 \qquad \text{for} \qquad 1 \le u < 2$$
$$\mathbf{p}(u) = (3-u)\mathbf{p}_2 + (u-2)\mathbf{p}_3 \qquad \text{for} \qquad 2 \le u < 3$$
$$\mathbf{p}(u) = (4-u)\mathbf{p}_3 + (u-3)\mathbf{p}_4 \qquad \text{for} \qquad 3 \le u < 4$$
$$\mathbf{p}(u) = (5-u)\mathbf{p}_4 + (u-4)\mathbf{p}_5 \qquad \text{for} \qquad 4 \le u \le 5$$

Here the resulting curve is a sequence of straight line segments connecting the control points.

Finally, for the $N_{i,3}(u)$ basis functions with $n = 5$ and $K = 3$, we find that $0 \le u \le 4$ and the knot vector is [0,0,0,1,2,3,4,4,4], or

$$\begin{aligned} t_0 &= 0 & t_5 &= 3 \\ t_1 &= 0 & t_6 &= 4 \\ t_2 &= 0 & t_7 &= 4 \\ t_3 &= 1 & t_8 &= 4 \\ t_4 &= 2 \end{aligned} \tag{5.9}$$

This is a nonuniform knot spacing, and the knot values correspond to the joints between the curve segments.

Again, because of the recursive nature of the basis function equations, we must first compute the $N_{i,1}(u)$ and then the $N_{i,2}(u)$, strictly in that order,

Figure 5.2 B-Spline basis functions for $n = 5$, $K = 2$.

before computing the $N_{i,3}(u)$. For the $N_{i,1}(u)$ we obtain

$$
\begin{aligned}
N_{0,1}(u) &= 1 \quad && \text{for} \quad u = 0 \\
&= 0 && \text{otherwise} \\
N_{1,1}(u) &= 1 \quad && \text{for} \quad u = 0 \\
&= 0 && \text{otherwise} \\
N_{2,1}(u) &= 1 \quad && \text{for} \quad 0 \le u < 1 \\
&= 0 && \text{otherwise}
\end{aligned}
$$

$$
\begin{aligned}
N_{3,1}(u) &= 1 \quad && \text{for} \quad 1 \le u < 2 \\
&= 0 && \text{otherwise} \\
N_{4,1}(u) &= 1 \quad && \text{for} \quad 2 \le u < 3 \\
&= 0 && \text{otherwise} \\
N_{5,1}(u) &= 1 \quad && \text{for} \quad 3 \le u \le 4 \\
&= 0 && \text{otherwise}
\end{aligned}
$$

Substituting these equations as required into Equation (5.3) produces

$$
\begin{aligned}
N_{0,2}(u) &= 0 \\
N_{1,2}(u) &= (1-u)N_{2,1}(u) \\
N_{2,2}(u) &= uN_{2,1}(u) + (2-u)N_{3,1}(u) \\
N_{3,2}(u) &= (u-1)N_{3,1}(u) + (3-u)N_{4,1}(u) \\
N_{4,2}(u) &= (u-2)N_{4,1}(u) + (4-u)N_{5,1}(u) \\
N_{5,2}(u) &= (u-3)N_{5,1}(u)
\end{aligned}
$$

Notice how the new knot values affect the $N_{i,2}(u)$. Now we substitute as appropriate into Equation (5.3) to obtain the $N_{i,3}(u)$:

$$
\begin{aligned}
N_{0,3}(u) &= (1-u)^2 N_{2,1}(u) \\
N_{1,3}(u) &= \frac{1}{2}u(4-3u)N_{2,1}(u) + \frac{1}{2}(2-u)^2 N_{3,1}(u) \\
N_{2,3}(u) &= \frac{1}{2}u^2 N_{2,1}(u) + \frac{1}{2}(-2u^2+6u-3)N_{3,1}(u) + \frac{1}{2}(3-u)^2 N_{4,1}(u) \\
N_{3,3}(u) &= \frac{1}{2}(u-1)^2 N_{3,1}(u) + \frac{1}{2}(-2u^2+10u-11)N_{4,1}(u) + \frac{1}{2}(4-u)^2 N_{5,1}(u) \\
N_{4,3}(u) &= \frac{1}{2}(u-2)^2 N_{4,1}(u) + \frac{1}{2}(-3u^2+20u-32)N_{5,1}(u) \\
N_{5,3}(u) &= (u-3)^3 N_{5,1}(u)
\end{aligned}
$$

Here, again, the $N_{i,1}(u)$ functions act like switches, turning on and off the terms that they control. These basis functions are plotted in Figure 5.3. Notice the symmetries and congruencies of the shapes of these functions; we will explore this phenomenon in more detail a little later. Applying these basis functions to a set of six control points using Equation (5.1), we find that $\mathbf{p}(u)$ is given by a different equation for each unit interval in u through the action of the $N_{i,1}(u)$ switches, so that

$$\mathbf{p}_1(u) = (1-u)^2\mathbf{p}_0 + \frac{1}{2}u(4-3u)\mathbf{p}_1 + \frac{1}{2}u^2\mathbf{p}_2 \qquad \text{for} \qquad 0 \le u < 1$$

$$\mathbf{p}_2(u) = \frac{1}{2}(2-u)^2\,\mathbf{p}_1 + \frac{1}{2}(-2u^2+6u-3)\mathbf{p}_2 + \frac{1}{2}(u-1)^2\mathbf{p}_3 \qquad \text{for} \qquad 1 \le u < 2$$

$$\mathbf{p}_3(u) = \frac{1}{2}(3-u)^2\mathbf{p}_2 + \frac{1}{2}(-2u^2+10u-11)\mathbf{p}_3 + \frac{1}{2}(u-2)^2\mathbf{p}_4 \qquad \text{for} \qquad 2 \le u < 3$$

$$\mathbf{p}_4(u) = \frac{1}{2}(4-u)^2\mathbf{p}_3 + \frac{1}{2}(-3u^2+20u-32)\mathbf{p}_4 + \frac{1}{2}(u-3)^2\mathbf{p}_5 \qquad \text{for} \qquad 3 \le u < 4$$

The resulting curve is a composite sequence of four curve segments connected with C^1 continuity; an example is shown in Figure 5.4. We notice that the curve passes through only the first and last points, $\mathbf{p}_0$ and $\mathbf{p}_5$, and it is tangent to $\mathbf{p}_1 - \mathbf{p}_0$ and $\mathbf{p}_5 - \mathbf{p}_4$ at these same points. We also notice that this curve is tangent to each successive side of the characteristic polygon (only for $K = 3$ curves). This tangency occurs at the joints between curve segments (that is, at integral values of u). These joints are indicated by tick marks. In the figure, point $\mathbf{p}_1$ is moved to $\mathbf{p}'_1$, and the effect on the curve is plotted. This local

Figure 5.3 Nonuniform B-Spline basis functions for $n = 5$, $K = 3$.

Figure 5.4 Nonuniform B-Spline curves: $n = 5$, $K = 3$.

change affects only two segments of the curve; it is also evident in the preceding set of equations, because $\mathbf{p}_1$ occurs in only the first and second equations.

From these same equations, we infer that only three control points influence each curve segment. Conversely, a control point can influence the shape of at most only three curve segments. Similar observations apply to $K = 1$ and $K = 2$ curves. In fact, we can generalize this observation: Each segment of a B-Spline curve is influenced by only K control points, and, conversely, each control point influences only K curve segments. A nonuniform cubic B-Spline requires at least four control points, $n \geq 3$, and at least eight nondecreasing knot values.

What happens if we increase the number of control points? To answer this, we must review Figures 5.1 to 5.3. In Figure 5.1, we see no evidence whatso-

ever to indicate an influence on the shape of the basis-function curves that we can attribute to the number of control points. In both Figures 5.2 and 5.3, we also correctly infer that, except for the basis functions influenced by the endpoints or those very near them (depending on K), the interior basis functions are independent of n. For example, if $n = 7$ and $K = 3$, we would find that $N_{2,3}(u) = N_{3,3}(u) = N_{4,3}(u) = N_{5,3}(u)$.

Let us pursue this idea of basis-function independence of n and at the same time try to develop a more convenient matrix notation. We will do this for $K = 3$. First, we compute $N_{i,3}(u)$, $N_{i+1,3}(u)$, and $N_{i+2,3}(u)$. The reason for selecting these three will become clear as our investigation progresses. We choose an interval in i so that $K \le i \le n$. This anticipates the uniform B-Spline and, for now, simplifies the calculation of the t_j knot values so that $t_j = j - K + 1 = j - 2$. Using the recursive formulas of Equations (5.2) and (5.3), we then compute

$$
\begin{aligned}
N_{i,1}(u) &= 1 && \text{for} \quad i-2 \le u < i-1 \\
&= 0 && \text{otherwise} \\
N_{i+1,1}(u) &= 1 && \text{for} \quad i-1 \le u < i \\
&= 0 && \text{otherwise} \\
N_{i+2,1}(u) &= 1 && \text{for} \quad i \le u < i+1 \\
&= 0 && \text{otherwise} \\
N_{i+3,1}(u) &= 1 && \text{for} \quad i+1 \le u < i+2 \\
&= 0 && \text{otherwise} \\
N_{i+4,1}(u) &= 1 && \text{for} \quad i+2 \le u < i+3 \\
&= 0 && \text{otherwise}
\end{aligned}
$$

Next we compute

$$
\begin{aligned}
N_{i,2}(u) &= (u - i + 2)N_{i,1}(u) + (i - u)N_{i+1,1}(u) \\
N_{i+1,2}(u) &= (u - i + 1)N_{i+1,1}(u) + (i + 1 - u)N_{i+2,1}(u) \\
N_{i+2,2}(u) &= (u - i)N_{i+2,1}(u) + (i + 2 - u)N_{i+3,1}(u) \\
N_{i+3,2}(u) &= (u - i - 1)N_{i+3,1}(u) + (i + 3 - u)N_{i+4,1}(u)
\end{aligned}
$$

And, finally, we compute

$$
\begin{aligned}
N_{i,3}(u) = \frac{1}{2}[&(u - i + 2)^2 N_{i,1}(u) + (u - i + 2)(i - u)N_{i+1,1}(u) \\
&+ (i + 1 - u)(u - i + 1)N_{i+1,1}(u) + (i + 1 - u)^2 N_{i+2,1}(u)]
\end{aligned}
$$

$$N_{i+1,3}(u) = \frac{1}{2}[(u-i+1)^2 N_{i+1,1}(u) + (u-i+1)(i+1-u)N_{i+2,1}(u) + (i+2-u)(u-i)N_{i+2,1}(u) + (i+2-u)^2 N_{i+3,1}(u)]$$

$$N_{i+2,3}(u) = \frac{1}{2}[(u-i)^2 N_{i+2,1}(u) + (u-i)(i+2-u)N_{i+3,1}(u) + (i+3-u)(u-i-1)N_{i+3,1}(u) + (i+3-u)^2 N_{i+4,1}(u)]$$

Now we are ready to find an expression for $\mathbf{p}(u)$ over an arbitrary segment of the curve, say, for the interval $i \le u < i+1$. We again use the $N_{i,1}(u)$ as switches. Only the functions $N_{i+2,1}(u) = 1$ in this interval. Collecting those terms from the set of equations above that $N_{i+2,1}(u)$ switches on, we obtain from Equation (5.1)

$$\begin{aligned}\mathbf{p}(u) = {} & \frac{1}{2}(i+1-u)^2\mathbf{p}_i \\ & + \frac{1}{2}[(u-i+1)(i+1-u) + (i+2-u)(u-i)]\mathbf{p}_{i+1} \\ & + \frac{1}{2}(u-i)^2\mathbf{p}_{i+2}\end{aligned} \tag{5.10}$$

There are computational advantages to reparameterizing the interval so that $0 \le u < 1$ and identifying the interval in some useful way; for example, by subscripting $\mathbf{p}(u)$ as $\mathbf{p}_i(u)$ for the ith interval. To reparameterize Equation (5.10), we replace u by $u+1$, so that

$$\mathbf{p}_i(u) = \frac{1}{2}[(1-u)^2\mathbf{p}_i + (-2u^2+2u+1)\mathbf{p}_{i+1} + u^2\mathbf{p}_{i+2}] \tag{5.11}$$

where $0 \le u < 1$. Note that as long as we maintain the function notation, we will not confuse the function $\mathbf{p}_i(u)$ with the control point $\mathbf{p}_i$.

Matrix Form

We now rewrite Equation (5.11) using matrix notation, replacing i by $i-1$ (i now denotes the curve segment number). So, for a B-Spline with $K = 3$, we obtain

$$\mathbf{p}_i(u) = \frac{1}{2}[u^2 \quad u \quad 1]\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \end{bmatrix} \qquad \text{for} \qquad i \in [1 : n-1] \tag{5.12}$$

The analogous form for cubic B-Splines, $K = 4$, is

$$\mathbf{p}_i(u) = \frac{1}{6}[u^3 \quad u^2 \quad u \quad 1]\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}\begin{bmatrix} \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \mathbf{p}_{i+2} \end{bmatrix} \qquad \text{for} \qquad i \in [1 : n-2] \tag{5.13}$$

Simplifying these equations for a second-degree curve, where $K = 3$, $\mathbf{U} = [u^2 \quad u \quad 1]$, and

$$\mathbf{M}_S = \frac{1}{2}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

we have

$$\mathbf{p}_i(u) = \mathbf{U}\mathbf{M}_S\begin{bmatrix} \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \end{bmatrix} \qquad i \in [1 : n-1] \qquad \text{for open curves} \tag{5.14}$$

For the cubic curve, where $K = 4$, $\mathbf{U} = [u^3 \quad u^2 \quad u \quad 1]$, and

$$\mathbf{M}_S = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & 6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

we have

$$\mathbf{p}_i(u) = \mathbf{U}\mathbf{M}_S\begin{bmatrix} \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \mathbf{p}_{i+2} \end{bmatrix} \qquad i \in [1 : n-2] \qquad \text{for open curves} \tag{5.15}$$

The number of segments i is determined in Equations (5.14) and (5.15) for open curves. We investigate closed B-Spline curves later. The sizes of $\mathbf{U}$ and $\mathbf{M}_S$ depend on the degree of the curve.

Equations (5.14) and (5.15) are for specific K values. However, the general formulation is

$$\mathbf{p}_i(u) = \mathbf{U}\mathbf{M}_S\mathbf{P}_K \qquad i \in [1 : n+2-K] \tag{5.16}$$

where

$$\mathbf{U} = [u^{K-1} \quad u^{K-2} \quad \cdots \quad u \quad 1] \tag{5.17}$$

and

$$\mathbf{P}_K = [\mathbf{p}_j] \qquad j \in [i-1 : i+K-2] \qquad \text{for open curves} \tag{5.18}$$

We have now defined the basis-function transformation matrices for the Hermite, Bézier, and B-Spline curves. They are $\mathbf{M}_F$, $\mathbf{M}_B$, and $\mathbf{M}_S$, respectively.

5.2 UNIFORM B-SPLINE BASIS FUNCTIONS

Many geometric-modeling situations do not require the curve to pass through the endpoints. In those cases, we apply a formulation such as that in Equation (5.12) or (5.13) to the set of control points. For example, given the set of control points in Figure 5.4 and a uniform knot vector, we find for $n = 5$ and $K = 3, i \in [1:4]$

$$\begin{aligned}
\mathbf{p}_1(u) &= \mathbf{U}\mathbf{M}_S\begin{bmatrix}\mathbf{p}_0\\ \mathbf{p}_1\\ \mathbf{p}_2\end{bmatrix}\\
\mathbf{p}_2(u) &= \mathbf{U}\mathbf{M}_S\begin{bmatrix}\mathbf{p}_1\\ \mathbf{p}_2\\ \mathbf{p}_3\end{bmatrix}\\
\mathbf{p}_3(u) &= \mathbf{U}\mathbf{M}_S\begin{bmatrix}\mathbf{p}_2\\ \mathbf{p}_3\\ \mathbf{p}_4\end{bmatrix}\\
\mathbf{p}_4(u) &= \mathbf{U}\mathbf{M}_S\begin{bmatrix}\mathbf{p}_3\\ \mathbf{p}_4\\ \mathbf{p}_5\end{bmatrix}
\end{aligned} \tag{5.19}$$

and for $n = 5$ and $K = 4, i \in [1:3]$

$$\begin{aligned}
\mathbf{p}_1(u) &= \mathbf{U}\mathbf{M}_S\begin{bmatrix}\mathbf{p}_0\\ \mathbf{p}_1\\ \mathbf{p}_2\\ \mathbf{p}_3\end{bmatrix}\\
\mathbf{p}_2(u) &= \mathbf{U}\mathbf{M}_S\begin{bmatrix}\mathbf{p}_1\\ \mathbf{p}_2\\ \mathbf{p}_3\\ \mathbf{p}_4\end{bmatrix}\\
\mathbf{p}_3(u) &= \mathbf{U}\mathbf{M}_S\begin{bmatrix}\mathbf{p}_2\\ \mathbf{p}_3\\ \mathbf{p}_4\\ \mathbf{p}_5\end{bmatrix}
\end{aligned} \tag{5.20}$$

The B-Spline curves in Figure 5.5 result when we complete the remaining computations. Notice that neither the $K = 3$ curve nor the $K = 4$ curve passes through any of the control points. This method produces a uniform or periodic B-Spline curve. It is periodic because the basis function repeats itself identically over successive intervals of the parametric variable. On the other hand, the curves in Figure 5.4 are nonperiodic or nonuniform B-Spline curves, and the basis functions differ from one segment to the next. Again, in both cases, of course, the knot values correspond to the joints between the curve segments. We make use of the nonuniform B-Spline characteristic that allows us to have multiple knot values by inserting knots at selected locations, a convenient way to reduce the continuity at a segment joint without creating straight line segments (as in the case of the Bézier or uniform *B*-Spline

Figure 5.5 Uniform B-Spline curves: $n = 5$, $K = 3$ and $n = 5$, $K = 4$.

curves). See Farin (1993a) for descriptions of the effects of knot insertion on the shape of a B-Spline curve. In fact, we can force the nonuniform B-Spline curve to interpolate a control point, including the first and last control points. Equations (5.19) and (5.20) show which control points influence a particular curve segment, and they show the influence of a single control point.

Figure 5.6 presents the basis functions for $K = 2$, $K = 3$, and $K = 4$ uniform B-Spline curves. These basis functions span two-, three-, and four-unit intervals on the parametric line, respectively. Comparing these functions to those in Figures 5.2 and 5.3, we see that they are invariant from span to span.

5.3 QUADRATIC AND CUBIC B-SPLINE BASIS FUNCTIONS

To compare the uniform B-Spline basis functions to the Hermite and Bézier basis functions, we now look at the second-degree and cubic basis functions

Figure 5.6 Uniform B-Spline basis functions: $N_{i,2}(u)$, $N_{i,3}(u)$, and $N_{i,4}(u)$.

in more detail. We define $\mathbf{N}_3 = \mathbf{U}_3\mathbf{M}_3$ and $\mathbf{N}_4 = \mathbf{U}_4\mathbf{M}_4$, where $\mathbf{N}_3$ and $\mathbf{N}_4$ are the basis-function matrices for $K = 3$ and $K = 4$, respectively. Performing the matrix multiplications for $K = 3$ produces

$$\mathbf{N}_3 = [N_{1,3}(u) \quad N_{2,3}(u) \quad N_{3,3}(u)]$$

where

$$\begin{aligned} N_{1,3}(u) &= \frac{1}{2}(u^2 - 2u + 1) \\ N_{2,3}(u) &= \frac{1}{2}(-2u^2 + 2u + 1) \\ N_{3,3}(u) &= \frac{1}{2}u^2 \end{aligned} \tag{5.21}$$

and for $K = 4$,

$$\mathbf{N}_4 = [N_{1,4}(u) \quad N_{2,4}(u) \quad N_{3,4}(u) \quad N_{4,4}(u)]$$

where

$$\begin{aligned} N_{1,4}(u) &= \frac{1}{6}(-u^3 + 3u^2 - 3u + 1) \\ N_{2,4}(u) &= \frac{1}{6}(3u^3 - 6u^2 + 4) \\ N_{3,4}(u) &= \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1) \\ N_{4,4}(u) &= \frac{1}{6}u^3 \end{aligned} \tag{5.22}$$

In both cases, we see that $\sum N_{i,k}(u) = 1$. Figure 5.7 plots these basis functions.

Rewriting Equations (5.14) and (5.15) for $K = 3$ yields

$$\begin{aligned} \mathbf{p}_i(u) &= N_{1,3}(u)\mathbf{p}_{i-1} + N_{2,3}(u)\mathbf{p}_i + N_{3,3}(u)\mathbf{p}_{i+1} \\ i &\in [1 : n-1] \qquad \text{for open curves} \end{aligned} \tag{5.23}$$

and for $K = 4$ yields

$$\begin{aligned} \mathbf{p}_i(u) &= N_{1,4}(u)\mathbf{p}_{i-1} + N_{2,4}(u)\mathbf{p}_i + N_{3,4}(u)\mathbf{p}_{i+1} + N_{4,4}(u)\mathbf{p}_{i+2} \\ i &\in [1 : n-2] \qquad \text{for open curves} \end{aligned} \tag{5.24}$$

Figure 5.7 Uniform B-Spline basis functions on the unit interval for $K = 3$ and $K = 4$.

5.4 CLOSED B-SPLINE CURVES

The uniform B-Spline curves are particularly well suited to produce closed curves. Equations (5.14) and (5.15) are easily adapted for this by simple modifications of the segment number range and the subscripts on the control points. For uniform closed curves, we rewrite these equations as follows for $K = 3$:

$$\mathbf{p}_i(u) = \mathbf{U}_3\mathbf{M}_3\begin{bmatrix}\mathbf{p}_{(i-1) \bmod (n+1)} \\ \mathbf{p}_{i \bmod (n+1)} \\ \mathbf{p}_{(i+1) \bmod (n+1)}\end{bmatrix} \qquad i \in [1 : n+1] \qquad \text{for closed curves} \qquad (5.25)$$

Figure 5.8 Closed B-Spline curve with $n = 5$, $K = 4$.

The subscripts on the matrices are for convenience of reference. For $K = 4$,

$$\mathbf{p}_i(u) = \mathbf{U}_4\mathbf{M}_4\begin{bmatrix}\mathbf{p}_{(i-1) \bmod (n+1)}\\ \mathbf{p}_{i \bmod (n+1)}\\ \mathbf{p}_{(i+1) \bmod (n+1)}\\ \mathbf{p}_{(i+2) \bmod (n+1)}\end{bmatrix} \qquad i \in [1 : n+1] \qquad \text{for closed curves} \tag{5.26}$$

where mod(n + I) is the remaindering operator (that is, for example, 5mod4 = 1, 8mod3 = 2, 4mod4 = 0, and so on). The curve in Figure 5.8 is a good example to work through. Here, $n = 5$ and $K = 4$; using these values in Equation (5.26), we obtain

$$\mathbf{p}_i(u) = \mathbf{U}_4\mathbf{M}_4\begin{bmatrix}\mathbf{p}_{(i-1) \bmod 6}\\ \mathbf{p}_{i \bmod 6}\\ \mathbf{p}_{(i+1) \bmod 6}\\ \mathbf{p}_{(i+2) \bmod 6}\end{bmatrix} \qquad i \in [1 : 6] \tag{5.27}$$

Expanding this equation produces

$$
\begin{aligned}
\mathbf{p}_1(u) &= \mathbf{U}_4\mathbf{M}_4[\mathbf{p}_0 \quad \mathbf{p}_1 \quad \mathbf{p}_2 \quad \mathbf{p}_3]^T \\
\mathbf{p}_2(u) &= \mathbf{U}_4\mathbf{M}_4[\mathbf{p}_1 \quad \mathbf{p}_2 \quad \mathbf{p}_3 \quad \mathbf{p}_4]^T \\
\mathbf{p}_3(u) &= \mathbf{U}_4\mathbf{M}_4[\mathbf{p}_2 \quad \mathbf{p}_3 \quad \mathbf{p}_4 \quad \mathbf{p}_5]^T \\
\mathbf{p}_4(u) &= \mathbf{U}_4\mathbf{M}_4[\mathbf{p}_3 \quad \mathbf{p}_4 \quad \mathbf{p}_5 \quad \mathbf{p}_0]^T \\
\mathbf{p}_5(u) &= \mathbf{U}_4\mathbf{M}_4[\mathbf{p}_4 \quad \mathbf{p}_5 \quad \mathbf{p}_0 \quad \mathbf{p}_1]^T \\
\mathbf{p}_6(u) &= \mathbf{U}_4\mathbf{M}_4[\mathbf{p}_5 \quad \mathbf{p}_0 \quad \mathbf{p}_1 \quad \mathbf{p}_2]^T
\end{aligned}
\tag{5.28}
$$

We evaluate each of these curve segments for some sequence of u values and, of course, specific control-point coordinates to produce the plot of the resulting B-Spline curve. Notice the subscript sequences. Figure 5.9 is another example of a closed curve where the same conditions hold. It is clearly possible to generate a self-intersecting curve.

In Figure 5.10, moving $\mathbf{p}_2$ to $\mathbf{p}'_2$ produces a more elongated closed curve. Since only four control points define this curve, the perturbation of $\mathbf{p}_2$ affects the entire curve, although the most drastic change occurs near $\mathbf{p}'_2$, while the original curve is only slightly disturbed near $\mathbf{p}_0$.

Defining B-Spline curves with one or more multiply coincident control points produces an effect similar to that for Bézier curves without increasing the degree of the polynomials. Each group of multiple control points pulls the curve closer to itself. The example in Figure 5.11 shows the effect of one, two, and three control points at $\mathbf{p}_4$ (that is, at location $\mathbf{p}_4$, we have $\mathbf{p}_4$ alone; $\mathbf{p}_4$ and $\mathbf{p}_5$; or $\mathbf{p}_4$, $\mathbf{p}_5$, and $\mathbf{p}_6$, where $\mathbf{p}_4$, $\mathbf{p}_5$, and $\mathbf{p}_6$ have identical coordinate values).

Figure 5.9 Closed, self-intersecting B-Spline curve with $n = 5$, $K = 4$.

Figure 5.10 Closed B-Spline curve with $n = 3$, $K = 4$.

Figure 5.11 Closed B-Spline curves with multiply coincident control points, $K = 4$.

This makes it possible to define B-Spline curves with sharp corners, an advantage in many geometric-modeling situations.

5.5 CONTINUITY

To demonstrate the continuity between segments of a B-Spline curve, we will use the example in Figure 5.12. Because $K = 4$, the curve is a cubic B-Spline, and we can expect second-order parametric continuity throughout the curve. To examine the first- and second-derivative parametric continuity of this curve at point $u = 1$ of segment i corresponding to point $u = 0$ of segment $(i + 1)$, first we compute $\mathbf{N}_4^u$ and $\mathbf{N}_4^{uu}$:

$$\mathbf{N}_4^u = [N_{1,4}^u(u) \quad N_{2,4}^u(u) \quad N_{3,4}^u(u) \quad N_{4,4}^u(u)] \tag{5.29}$$

where

$$\begin{aligned} N_{1,4}^u(u) &= \frac{1}{6}(-3u^2 + 6u - 3) \\ N_{2,4}^u(u) &= \frac{1}{6}(9u^2 - 12u) \\ N_{3,4}^u(u) &= \frac{1}{6}(-9u^2 + 6u + 3) \end{aligned} \tag{5.30}$$

Figure 5.12 Four segments of a cubic B-Spline curve, $K = 4$.

$$N_{4,4}^{u}(u) = \frac{1}{2}u^2$$

and

$$\mathbf{N}_4^{uu} = [N_{1,4}^{uu}(u) \quad N_{2,4}^{uu}(u) \quad N_{3,4}^{uu}(u) \quad N_{4,4}^{uu}(u)] \tag{5.31}$$

where

$$\begin{aligned} N_{1,4}^{uu}(u) &= -u + 1 \\ N_{2,4}^{uu}(u) &= 3u - 2 \\ N_{3,4}^{uu}(u) &= -3u + 1 \\ N_{4,4}^{uu}(u) &= u \end{aligned} \tag{5.32}$$

Now we can write

$$\mathbf{p}_i^{u}(u) = N_{1,4}^{u}(u)\mathbf{p}_{i-1} + N_{2,4}^{u}(u)\mathbf{p}_i + N_{3,4}^{u}(u)\mathbf{p}_{i+1} + N_{4,4}^{u}(u)\mathbf{p}_{i+2} \tag{5.33}$$

and

$$\mathbf{p}_i^{uu}(u) = N_{1,4}^{uu}(u)\mathbf{p}_{i-1} + N_{2,4}^{uu}(u)\mathbf{p}_i + N_{3,4}^{uu}(u)\mathbf{p}_{i+1} + N_{4,4}^{uu}(u)\mathbf{p}_{i+2} \tag{5.34}$$

Next, we evaluate Equation (5.24) for $\mathbf{p}_i(1)$ and $\mathbf{p}_{i+1}(0)$ to obtain

$$\mathbf{p}_i(1) = \frac{1}{6}(\mathbf{p}_i + 4\mathbf{p}_{i+1} + \mathbf{p}_{i+2}) \tag{5.35}$$

and

$$\mathbf{p}_{i+1}(0) = \frac{1}{6}(\mathbf{p}_i + 4\mathbf{p}_{i+1} + \mathbf{p}_{i+2}) \tag{5.36}$$

Equation (5.36) shows that $\mathbf{p}_i(1) = \mathbf{p}_{i+1}(0)$, as we would expect. We examine first-derivative continuity by evaluating Equation (5.33) for $\mathbf{p}_i^{u}(1)$ and $\mathbf{p}_{i+1}^{u}(0)$. This yields

$$\mathbf{p}_i^{u}(1) = \frac{1}{2}(-\mathbf{p}_i + \mathbf{p}_{i+2}) \tag{5.37}$$

and

$$\mathbf{p}_{i+1}^{u}(0) = \frac{1}{2}(-\mathbf{p}_i + \mathbf{p}_{i+2}) \tag{5.38}$$

We see that $\mathbf{p}_i^{u}(0) = \mathbf{p}_{i+1}^{u}(1)$, demonstrating first-derivative continuity at the joint. Finally, we examine second-derivative continuity by evaluating Equation (5.34) for $\mathbf{p}_i^{uu}(1)$ and $\mathbf{p}_{i+1}^{uu}(0)$. This yields

$$\mathbf{p}_i''''(1) = \mathbf{p}_i - 2\mathbf{p}_{i+1} + \mathbf{p}_{i+2} \tag{5.39}$$

and

$$\mathbf{p}_{i+1}''''(0) = \mathbf{p}_i - 2\mathbf{p}_{i+1} + \mathbf{p}_{i+2} \tag{5.40}$$

We see that $\mathbf{p}_i''''(1) = \mathbf{p}_{i+1}''''(0)$, demonstrating second-derivative continuity at the joint. Of course, we can easily extend this analysis for higher-degree B-Spline curves.

The effects of multiply coincident control points and multiple knot values on the continuity at segment joints are worth some further discussion. Here is a summary of the continuity conditions:

Control point multiplicity = 1	C^2 and G^2
Control point multiplicity = 2	C^2 and G^1, with knots restricted to a reduced convex hull.
Control point multiplicity = 3	C^2 and G^0. The curve interpolates the triple control point, and the segments at each side of the joint are straight lines.
Control point multiplicity = 4	G^2 and G^0. The curve segments on both sides of the joint are straight lines and interpolate the control points on both sides.
Knot multiplicity = 1	C^2 and G^2
Knot multiplicity = 2	C^1 and G^1, with knots restricted to a reduced convex hull.
Knot multiplicity = 3	C^0 and G^0. The curve interpolates the control point. Curve segment shapes at the joint are free and not constrained to straight lines.
Knot multiplicity = 4	The curve is discontinuous, ending on one control point and resuming at the next. The shapes of the curve segments adjacent to the discontinuity are unconstrained.

Veltkamp (1992) surveys and discusses the various kinds of continuity conditions of curves and surfaces, including parametric, geometric, Frénet frame, and tangent surface continuity. Pegna and Wolter (1992) present results of work on the control of curvature continuity in the design of blend

surfaces, and Barsky and DeRose (1989) discuss the geometric continuity of parametric curves. These reports also reference other literature on the subject of continuity.

5.6 CONVERSION BETWEEN BASIS FUNCTIONS

The conversion between Bézier and B-Spline basis functions proceeds by equating appropriate matrix forms. For the cubic Bézier curve and the uniform cubic B-Spline, we have $\mathbf{U}\mathbf{M}_B\mathbf{P}_B = \mathbf{U}\mathbf{M}_S\mathbf{P}_S$, or $\mathbf{M}_B\mathbf{P}_B = \mathbf{M}_S\mathbf{P}_S$, so that to convert from B-Spline to Bézier control points, we have

$$\mathbf{P}_B = \mathbf{M}_B^{-1}\mathbf{M}_S\mathbf{P}_S \tag{5.41}$$

which expands to

$$\mathbf{P}_B = \frac{1}{6}\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{3} & 1 \\ 0 & \frac{1}{3} & \frac{2}{3} & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}\mathbf{P}_S$$

or

$$\mathbf{P}_B = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix}\mathbf{P}_S \tag{5.42}$$

To convert from Bézier to B-Spline control points, we use

$$\mathbf{P}_S = \mathbf{M}_S^{-1}\mathbf{M}_B\mathbf{P}_B \tag{5.43}$$

which expands to

$$\mathbf{P}_S = \begin{bmatrix} 0 & \frac{2}{3} & -1 & 1 \\ 0 & -\frac{1}{3} & 0 & 1 \\ 0 & \frac{2}{3} & 1 & 1 \\ 6 & \frac{11}{3} & 2 & 1 \end{bmatrix}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\mathbf{P}_B$$

or

$$\mathbf{P}_S = \begin{bmatrix} 6 & -7 & 2 & 0 \\ 0 & 2 & -1 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & 2 & -7 & 6 \end{bmatrix} \mathbf{P}_B \tag{5.44}$$

A nonuniform B-Spline equivalent of a cubic Bézier curve is produced by $K = 4$ and the knot vector [0,0,0,0,1,1,1,1].

Of course, we can also convert between the Hermite and B-Spline basis curves. For the cubic Hermite and uniform cubic B-Spline curves, we have

$$\mathbf{B} = \mathbf{M}_F^{-1} \mathbf{M}_S \mathbf{P}_S \tag{5.45}$$

which expands to

$$\mathbf{B} = \frac{1}{6} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \mathbf{P}_S$$

or

$$\mathbf{B} = \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ -3 & 0 & 3 & 0 \\ 0 & -3 & 0 & 3 \end{bmatrix} \mathbf{P}_S \tag{5.46}$$

Similarly, to go the other way, we have

$$\mathbf{P}_S = \mathbf{M}_S^{-1} \mathbf{M}_F \mathbf{B} \tag{5.47}$$

which expands to

$$\mathbf{P}_S = \begin{bmatrix} 0 & \frac{2}{3} & -1 & 1 \\ 0 & -\frac{1}{3} & 0 & 1 \\ 0 & \frac{2}{3} & 1 & 1 \\ 6 & \frac{11}{3} & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \mathbf{B}$$

or

$$\mathbf{P}_S = \begin{bmatrix} -1 & 2 & -\frac{7}{3} & -\frac{2}{3} \\ 2 & -1 & \frac{2}{3} & \frac{1}{3} \\ -1 & 2 & -\frac{1}{3} & -\frac{2}{3} \\ 12 & -11 & \frac{22}{3} & \frac{17}{3} \end{bmatrix} \mathbf{B} \tag{5.48}$$

5.7 NONUNIFORM RATIONAL B-SPLINE CURVES

One of the most versatile tools for modeling curves is the nonuniform rational B-Spline, or *NURBS curve,* as it is popularly called. It has become so widely used that it is almost an industry standard. Its basis is the ratio of two nonrational B-Spline basis functions, making it a vector-valued piecewise rational polynomial. As a rational form, it exhibits the same invariance properties as the rational Bézier basis, making use of homogeneous coordinates and characteristics of projective geometry. This means that the NURBS curves are invariant under translation, rotation, scaling, shear, and parallel and perspective projection. Like the rational Bézier curve, we can use a nonuniform rational B-Spline curve to represent exactly the conic curves, with the added advantages of local control. Nonrational B-Spline curves and rational and nonrational Bézier curves are special instances of the rational B-Splines.

Points on a NURBS curve are given by

$$\mathbf{p}(u) = \frac{\sum_{i=0}^{n} h_i \mathbf{p}_i N_{i,K}(u)}{\sum_{i=0}^{n} h_i N_{i,K}(u)} \tag{5.49}$$

where the h_i are the weights.

If weights $h_i = 1$ for all i, then the basis function of Equation (5.49) reduces to the nonrational B-Spline form $N_{i,k}$. However, if the weights $h_i = 1$ for all i and if the knot vector has the form $\mathbf{T} = \{0,0,\ldots,0,1,1,\ldots,1\}$, where 0 and 1 each have multiplicity K, then the basis function of this equation reduces to the nonrational Bernstein polynomial basis of degree $K - 1$ for the Bézier curve $B_{i,k-1}(u)$. If a weight $h_i = 0$, then control point $\mathbf{p}_i$ does not contribute to the curve shape.

There are three ways to modify the shape of a NURBS curve: Change the knot vector, move control points, and change the weights. Because it is relatively difficult to determine how a curve will respond to changes in the knot vector, this is not the best way to change curve shape. On the other hand, the effect of changing a control point is predictable and intuitive. If a weight h_i is increased or decreased in value, then the curve is pulled toward or pushed away from, respectively, the control point $\mathbf{p}_i$. Weight changes will move any point on the curve along a straight line through its corresponding control point. See Piegl and Tiller (1987) for more on curve and surface constructions using rational B-Splines, Piegl (1989c) for a discussion of shape modification, and Laurent-Gengoux and Mekhilef (1993) for a study of the optimization of a NURBS representation.

Choi, Yoo, and Lee (1990) present a method for obtaining the coefficient matrix of a NURBS curve of arbitrary degree and the matrix representation of a NURBS surface as a tensor product of NURBS curves. Joe, Wang, and Cheng (1994) discuss reduced knot NURBS representations of rational composite Bézier curves with G^1 continuity. Piegl (1991) and Farin (1992) survey the NURBS characteristics that led to the wide acceptance of this form.

5.8 REPRESENTING CONICS WITH NURBS CURVES

The NURBS curve is often the first choice of designers because they can use it to represent a conic curve exactly. For example, see Qin, Sun, and Wang (1992). A conic is a second-degree, or quadratic, curve, so we begin with the second-degree NURBS curve:

$$\mathbf{p}(u) = \frac{\sum_{i=0}^{2} h_i \mathbf{p}_i N_{i,K}(u)}{\sum_{i=0}^{2} h_i N_{i,K}(u)} \tag{5.50}$$

where $K = 3$ and the knot vector is $\{0,0,0,1,1,1\}$. This actually produces a rational Bézier curve. When expanded, the equation becomes

$$\mathbf{p}(u) = \frac{(1-u)^2 h_0 \mathbf{p}_0 + 2u(1-u)h_1 \mathbf{p}_1 + u^2 h_2 \mathbf{p}_2}{(1-u)^2 h_0 + 2u(1-u)h_1 + u^2 h_2} \tag{5.51}$$

We assert without proof that this is indeed the equation of a conic curve. Moreover, the ratio

$$\tilde{\rho} = \frac{h_1^2}{h_0 h_2} \tag{5.52}$$

is a constant, depending on the specie of conic represented. The following ratios determine a speeific type of conic:

$$\begin{aligned} \tilde{\rho} &< 1 \quad \text{for an ellipse} \\ \tilde{\rho} &= 1 \quad \text{for a parabola} \\ \tilde{\rho} &> 1 \quad \text{for a hyperbola} \end{aligned} \tag{5.53}$$

where $\tilde{\rho}$ is sometimes ealled the *conic shape factor.*

For a circular arc the following conditions must prevail:

1. The control points $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$ must define an isosceles triangle, where $\angle \mathbf{p}_0 = \angle \mathbf{p}_2$.
2. $h_0 = h_2 = 1$
3. $h_1 = \dfrac{|\mathbf{p}_2 - \mathbf{p}_0|}{2|\mathbf{p}_1 - \mathbf{p}_0|}$

We can join a set of four 90° curve segments to define a full circle, where each isosceles triangle has vertex angles of 45°, 90°, and 45°. Also, a seven-point square-based NURBS exists (see Figure 5.13), where

$$\mathbf{T} = \{0,0,0,\frac{1}{4},\frac{1}{2},\frac{1}{2},\frac{3}{4},1,1,1\}$$

Of course, one way to obtain any ellipse is by applying an appropriate affine transformation to the circle's eontrol points.

Figure 5.13 A seven-point square-based NURBS circle.

5.9 CUBIC BETA SPLINES

An interesting variation on the B-Spline basis function is the Beta-Spline, defined over a uniform knot sequence. It adds two more parameters, β_1 and β_2, which provide for global control over certain characteristics of curve shape. These two extra degrees of freedom are created when we relax the requirement of parametric continuity while preserving geometric continuity. Following the formulation of Equation (5.15), we have for the cubic beta-spline

$$\mathbf{p}_i(u) = \mathbf{U}\mathbf{M}_\beta \begin{bmatrix} \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \mathbf{p}_{i+2} \end{bmatrix} \qquad i \in [1 : n-2] \tag{5.54}$$

where

$$\mathbf{M}_\beta = \frac{1}{\delta} \begin{bmatrix} -2\beta_1^3 & 2(\beta_2 + \beta_1^3 + \beta_1^2 + \beta_1) & -2(\beta_2 + \beta_1^2 + \beta_1 + 1) & 2 \\ 6\beta_1^3 & -3(\beta_2 + 2\beta_1^3 + 2\beta_1^2) & 3(\beta_2 + 2\beta_1^2) & 0 \\ -6\beta_1^3 & 6(\beta_1^3 - \beta_1) & 6\beta_1 & 0 \\ 2\beta_1^3 & \beta_2 + 4(\beta_1^2 + \beta_1) & 2 & 0 \end{bmatrix} \tag{5.55}$$

and $\delta = \beta_2 + 2\beta_1^3 + 4\beta_1^2 + 4\beta_1 + 2$.

β_1 and β_2 are the so-called *bias* and *tension parameters,* respectively. Both affect the flatness of the curve. Manipulation of β_1 affects the tangent vector on the parameter-increasing side of each control point, and increasing β_2 pulls the curve closer to the sides of the control polygon. As we increase β_1 the curve asymmetrically shifts to one side of the control polygon. When we increase β_2 the curve uniformly flattens and more closely approaches and approximates the control polygon. A thorough discussion of this interesting curve (and surface) is found in Barsky (1988).

6

SURFACES

Our intuitive notion of a *surface* is a continuous set of points approximating a plane in the neighborhood of each of the points. Mathematically more succinct, we also may think of a surface as a two-parameter family of points. Yet another conception, analogous to our notion of a curve, is that of a surface as the locus of a point moving with two degrees of freedom. In addition, we can describe surfaces with special properties; for example, as the locus of points of a moving line or curve. These notions become useful to geometric modeling when we express them analytically. The intrinsic equations and characteristics of a surface are a subject of differential geometry and are far more subtle and complex than those of a curve. Where the theory of curves centers on curvature and torsion as functions of arc length, nothing quite so straightforward applies to surfaces. Although we will discuss many properties of surfaces that arise from their differential geometry in the chapters that follow, we will not pursue the fundamental forms or more rigorous derivations. In this chapter we review the *explicit* and *implicit equations* of surfaces and introduce the *bivariate parametric equations*. These topics are the basis for later discussions of quadric surfaces expressed implicitly, and the Hermite, Bézier, and B-Spline surfaces expressed parametrically. The implicit and parametric surfaces are the most commonly used surfaces in geometric modeling, and most of our studies here are particularly about the *tensor product parametric surfaces*.

We construct, evaluate, and analyze points and curves on a surface to display it, to reveal special properties, or to demonstrate its relationship to other geometric objects, which may then become the basis for subsequent

modeling processes or definitions. So this chapter also discusses evaluating points on a surface, curve nets, and embedded curves on surfaces.

6.1 EXPLICIT AND IMPLICIT EQUATIONS OF SURFACES

An equation of the form

$$f(x,y,z) = 0 \tag{6.1}$$

is the *implicit equation* of a surface, where $f(x,y,z)$ is a polynomial in x, y, and z such that

$$\sum_{i,j,k} a_{ijk} x^i y^j z^k = 0$$

From elementary algebra, we know that the degree of any term in this equation is simply the sum of the exponents appearing in it. In other words, the degree of term $a_{ijk}x^iy^jz^k$ is $i + j + k$. The degree of the polynomial expression itself is the maximum degree of its individual terms. If this equation is linear in all variables, then the surface is an *unbounded plane*. If it is a second-degree equation, then the surface is a *quadric*, of which the *sphere* is a special case. Finally, if one of the variables is absent from the equation, the surface must be a *cylinder* whose generators are parallel to the axis of the absent variable.

When we solve the implieit equation for one of the variables as a function of the other two, say, for z as a function of x and y, we obtain

$$z = f(x,y) \tag{6.2}$$

This represents the same surface as Equation (6.1). However, this form is the *explicit equation* of the surface.

It is also possible to express every rational and nonrational parametric surface with an implicit equation. The process of converting from a parametric to implicit form is called *implicitization*. Although the implicit and explicit forms are useful, in representing surfaces they suffer from an inherent weakness: their inability to represent an easily transformed and bounded surface. However, we can either direetly adapt or closely approximate most, if not all, of these implicit and explieit forms by parametric equations. On the other hand, they do have interesting strengtbs associated with intersection computations, point classifieation, and the inverse point eomputation.

Here is a relatively simple surface-fitting procedure, based on an explicit equation (see Figure 6.1)

Figure 6.1 An explicit surface.

$$z(x,y) = \sum_{i=0}^{m} \sum_{j=0}^{n} a_{ij} x^i y^j \tag{6.3}$$

where we determine the a_{ij} coefficients from a specified set of data points. We must specify 16 points for a bicubic polynomial (that is, $m = n = 3$). If m and n are large, then it becomes increasingly possible to inadvertently generate surfaces with undesirable oscillations.

This surface, as well as the nonrational Hermite, Bézier, B-Spline, and similar surfaces, are mathematically expressed as a *tensor product* (a generalization of the scalar product for vectors). A tensor is a multiply subscripted array that transforms in a certain way. For a variety of practical reasons, the indexing scheme used here and in most of the modeling literature does not follow strictly the more rigorous conventions of mathematical physics.

QUADRIC SURFACES

A *quadric surface* is one whose implicit representation is given by $f(x,y,z) = 0$ where $f(x,y,z)$ is a quadratic polynomial. The algebraic form of a quadric surface is given by the equation

$$Ax^2 + By^2 + Cz^2 + 2Dxy + 2Eyz + 2Fxz + 2Gx + 2Hy + 2Jz + K = 0 \tag{6.4}$$

For example, if $A = B = C = -K = 1$ and $D = E = F = G = H = J = 0$, then the equation produces a unit sphere at the origin.

In matrix form, we may write Equation (6.4) as

$$\mathbf{PQP}^T = 0 \tag{6.5}$$

where $\mathbf{P} = [x \quad y \quad z \quad 1]$ and

$$\mathbf{Q} = \begin{bmatrix} A & D & F & G \\ D & B & E & H \\ F & E & C & J \\ G & H & J & K \end{bmatrix} \tag{6.6}$$

The ten coefficients of $\mathbf{Q}$ have no direct geometric or physical meaning. Therefore, a geometric form of representation is often more practical. We convert to the algebraic form only when computationally necessary. However, computational problems may arise when using either form. The source of these problems is the very nature of floating-point data representation and computation, including imperfect accuracy and lack of robustness of intrinsic surface characteristics subjected to repeated transformations. This may slowly degrade the defining data. These small changes in the coefficients may cause a radical change in computed surface type.

We can represent every quadric surface by a data set consisting of vectors, scalars, and some kind of type identifier. For example, two vectors and a scalar represent a *right circular cylinder*, where one vector defines an arbitrary point on its axis, another vector defines the direction of the axis, and a scalar gives the radius (see Figure 6.2).

A rigid-body motion is produced simply by pre- and postmultiplying $\mathbf{Q}$ by a 4×4 transformation matrix $\mathbf{T}$:

$$\mathbf{Q}' = \mathbf{T}^{-1}\mathbf{Q}[\mathbf{T}^{-1}]^T \tag{6.7}$$

Certain properties of the matrix of quadric equation coefficients $\mathbf{Q}$ are invariant under rigid-body transformations, including the determinants $|\mathbf{Q}|$ and $|\mathbf{Q}_u|$. This allows computation of surface type, center, vertex, and axes (see Table 6.1). The tests to determine surface type are critically dependent on computing whether certain invariants are positive, negative, or zero. Again, this is often computationally impractical or unrealizable with floating-point arithmetic.

Using the following definitions and tests, we easily determine the type of quadric surface:

$$\mathbf{Q}_u = \begin{bmatrix} A & D & F \\ D & B & E \\ F & E & C \end{bmatrix}$$

Figure 6.2 A right circular cylinder.

$r_1 = \text{rank } \mathbf{Q}$

$r_2 = \text{rank } \mathbf{Q}_n$

s = signature of $\mathbf{Q}_n$, where the signature s of a quadratic form is defined as the difference between the number of positive and negative characteristic roots.

$\tau_1 = A + B + C$

$\tau_2 = AB + AC + BC - D^2 - E^2 - F^2$

$\delta_1 = \tau_2 + \tau_1 K - G^2 - H^2 - J^2$

$$\delta_2 = ABC + ACK + ABK + BCK + 2(DEF + FCJ + DGH + EHJ) - D^2(C + K) - E^2(A + K) - F^2(B + K) - G^2(B + C) - H^2(A + C) - J^2(A + B)$$

α: $\tau_2 > 0$; $|\mathbf{Q}_n|\tau_1 \leq 0$ or $\tau_2 \leq 0$

β: $\tau_2 > 0$; $|\mathbf{Q}_n|\tau_1 > 0$

A quadric surface of revolution is produced by rotating a conic curve about its axis. This surface is in its canonical position if and only if its center

Table 6.1 Classification of quadric surfaces

r_1	r_2	s	Other Conditions	Surface Type
1	0	0		Invalid
1	1	1		Coincident planes
2	0	0		Single plane
2	1	1	$\delta_1 > 0$	Invalid
2	1	1	$\delta_1 < 0$	Two parallel planes
2	2	0	$\tau_2 < 0$	Two intersecting planes
2	2	2	$\tau_2 > 0$	Line
3	1	1		Parabolic cylinder
3	2	0	$\tau_2 < 0$	Hyperbolic cylinder
3	2	2	$\tau_2 > 0; \tau_1\delta_2 < 0$	Elliptic cylinder
3	2	2	$\tau_2 > 0; \tau_1\delta_2 > 0$	Invalid
3	3	1	α	Cone
3	3	3	β	Point

$\lvert Q_4 \rvert$	$\lvert Q \rvert$	s	Other Conditions	Surface Type
0	+	0	$\tau_2 < 0$	Hyperbolic paraboloid
0	−	2	$\tau_2 > 0$	Elliptic paraboloid
±	+	1	α	Hyperboloid of one sheet
±	−	1	α	Hyperboloid of two sheets
±	+	3	β	Invalid
±	−	3	β	Ellipsoid

or vertex is at the origin and its axes of symmetry coincide with the coordinate axes. In this position there are no cross terms in the equation. Thus, $D = E = F = 0$, so that

$$Ax^2 + By^2 + 2Gx + 2Hy + 2Jz + K = 0 \tag{6.8}$$

where $AG = BH = CJ = 0$, since there is only one term in each variable if the center or vertex is at the origin. An additional condition is necessary. The axis of rotation must be along the z axis. This means that the intersection of the x, y plane ($z = 0$) and a quadric surface of revolution in canonical position is a circle whose center is at the origin; thus,

$$Ax^2 + By^2 + 2Gx + 2Hy + K = 0 \tag{6.9}$$

For a circle, $A = B \neq 0$ and $G = H = 0$. Applying these conditions to the general quadratic equation and dividing by A, we obtain the canonical equation of a quadric surface of revolution:

$$x^2 + y^2 + Lz^2 + Mz + N = 0 \tag{6.10}$$

where $LM = 0$. There are eight families in this general class (see Table 6.2).

6.3 PARAMETRIC EQUATIONS OF SURFACES

The simplest and most common mathematical element we use to model a surface is a *patch*. It is a convenient way to break down a complex surface into manageable units. In this way the patch is analogous to a segment of a spline curve. A patch is a curve-bounded collection of points whose coordinates are given by continuous, two-parameter (bivariate), single-valued polynomials of the form

$$x = x(u,w) \qquad y = y(u,w) \qquad z = z(u,w) \tag{6.11}$$

where the parametric variables u and w are constrained to the intervals $u,w \in [0,1]$. This, of course, generates a rectangular patch. There are other formulations that produce triangular or other nonrectangular patches.

Fixing the value of one of the parametric variables results in a curve on the patch in terms of the other variable, which remains free. By continuing this process first for one variable and then the other for any number of arbitrary values in the allowed interval, we form a parametric net of two one-parameter families of curves on the patch so that just one curve of each family passes through each point $\mathbf{p}(u,w)$. Again, the positive sense on any curve is the sense in which the parameter increases.

Associated with every patch is a set of boundary conditions (see Figure 6.3). The most obvious of these are the four corner points and the four curves defining its edges. Others of importance are the *tangent vectors*, *normal vectors*, and *twist vectors* (for the cubic Hermite patch), which we will discuss later. For an ordinary rectangular patch, there are always four and only four

Table 6.2 Quadric surfaces of revolution

Surface	Canonical Equation
Sphere	$x^2 + y^2 + z^2 - N = 0$
Cylinder	$x^2 + y^2 - N = 0$
Cone	$x^2 + y^2 - Lz^2 = 0$
Paraboloid	$x^2 + y^2 + Mz = 0$
Prolate ellipsoid	$x^2 + y^2 + Lz^2 - N = 0, L < 1$
Oblate ellipsoid	$x^2 + y^2 + Lz^2 - N = 0, L > 1$
Hyperboloid of one sheet	$x^2 + y^2 - Lz^2 - N = 0$
Hyperboloid of two sheets	$x^2 + y^2 - Lz^2 + N = 0$

Figure 6.3 A parametric surface patch.

corner points and edge curves. This follows from the possible combinations of the two limits of the two parametric variables. We find the corner points by substituting these four combinations of 0 and 1 into $\mathbf{p}(u,w)$ to obtain $\mathbf{p}(0,0)$,$\mathbf{p}(0,1)$,$\mathbf{p}(1,0)$, and $\mathbf{p}(1,1)$. On the other hand, the edge or boundary curves are functions of one of the two parametric variables. We obtain these by allowing one of the variables to remain free while fixing the other to its limiting values. This procedure results in four and only four possible combinations yielding the functions of the four parametric boundary curves $\mathbf{p}(u,0)$,$\mathbf{p}(u,1)$,$\mathbf{p}(0,w)$, and $\mathbf{p}(1,w)$.

Here are some examples of general parametric surface patches and their defining equations. The simplest one is a plane. The following parametric equations represent a rectangular segment of the x,y plane.

$$x = (c-a)u + a \qquad y = (d-b)w + b \qquad z = 0 \tag{6.12}$$

where $u,w \in [0,1]$. Figure 6.4 illustrates this patch, showing both the parametric coordinates and the x,y coordinates of each corner point. The curves of constant w are straight lines running parallel to the x axis. The curves are functions of u. An analogous situation exists for the lines of constant u, which

Figure 6.4 Parametric and x, y coordinates of a plane.

are straight lines parallel to the y axis. The boundaries of this plane are $x = a$, $x = c$, $y = b$, and $y = d$.

The next simplest surface is the *sphere* (see Figure 6.5), the locus of points at a constant distance from a given point. The parametric equations of a sphere of radius r, centered at point (x_0, y_0, z_0), are

Figure 6.5 A parametric sphere.

$$\begin{aligned} x &= x_0 + r\cos u\cos w \\ y &= y_0 + r\cos u\sin w \\ z &= z_0 + r\sin u \end{aligned} \qquad u \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right], \quad w \in [0,2\pi] \tag{6.13}$$

where u is analogous to latitude and w to longitude, with both angles in radians. Obviously, curves of constant u are parallels of latitude, so that a circle is generated in a plane parallel to the x,y plane as w varies. Curves of constant w are meridians of longitude, generating semicircles as u varies.

An ellipsoid centered at (x_0,y_0,z_0) has the following parametric equations:

$$\begin{aligned} x &= x_0 + a\cos u\cos w \\ y &= y_0 + b\cos u\sin w \\ z &= z_0 + c\sin u \end{aligned} \qquad u \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right], \quad w \in [0,2\pi] \tag{6.14}$$

Equations (6.14) are illustrated in Figure 6.6.

Figure 6.7 shows a *parametric surface of revolution*. For clarity, only part of the surface is shown. Notice that it is the curve defined by the function $x(u),z(u)$ that is revolved around the z axis. The equations are

$$\begin{aligned} x &= x(u)\cos w \\ y &= x(u)\sin w \\ z &= z(u) \end{aligned} \qquad u \in [0,1], \quad w \in [0,2\pi] \tag{6.15}$$

The parameter space of a patch differs from that of a curve, because we must account for two parametric variables. Thus, we have a set of hybrid three-dimensional parameter spaces defined by (u,w,x), (u,w,y), and (u,w,z)

Figure 6.6 A parametric ellipsoid.

Figure 6.7 A parametric surface of revolution.

coordinates. As with cubic Hermite curves, decomposing a bicubic patch into its parameter-space components often helps to elaborate its behavior in model space.

The surface in Figure 6.8 has the following properties: $\mathbf{p}_{00}$ and $\mathbf{p}_{01}$ lie in the y,z plane; $\mathbf{p}_{10}$ and $\mathbf{p}_{11}$ lie in the x,y plane; each boundary curve is planar. The graphs on the left show the patch's components in parameter space. Shown at the top left in the u,w,x coordinate system is a component patch given by the equation $x = x(u,w)$; the u,w,y and u,w,z patches are given by similar equations. Again, notice that the parametric variables are constrained to the unit square $u,w \in [0,1]$. Holding one of the parametric variables constant defines a specific parametric curve on the patch. The figure shows the curve $\mathbf{p}(u_i,w)$. Every point on this curve has the same u value, and in parameter space we see this curve decomposed into its components. These parametric curves are revealed by passing the plane $u = u_i$ through the component patches.

Figure 6.8 The parameter space of a surface.

As with curves, we can observe many characteristics of a patch by a careful inspection of the parameter-space component patches. Figure 6.9 shows three special cases. We call Figures 6.9a and 6.9b *degenerate patches*. In Figure 6.9a, a point is the result in model space; in Figure 6.9b, a straight line; and in Figure 6.9c, a plane. Figure 6.10 shows all three results in model space.

In addition to their inherent boundedness and ease of transformation, another advantage of the parametric representation of a surface is the complete control we have of the domain of a surface-modeling operation simply by the choosing an appropriate parameterization scheme. By carefully specifying subsets of a particular domain $[u_{min}, u_{max}] \times [w_{min}, w_{max}]$, we can readily define certain sections of a surface. This is useful whenever a surface is composed of several patches. We will frequently use this freedom of parameterization to choose $[0,1] \times [0,1]$ as the domain of a surface-interpolating

Figure 6.9 The parametric component surfaces of three special surfaces. Nine plane faces are shown in three groups of three.

operation. Any other domain can be normalized as necessary to this unit square in parameter space.

Two observations are necessary. First, we will often use the terms *patch* and *surface* interchangeably. Although, strictly speaking, a patch is a limited region of a larger surface, in the context of composite surfaces or patches the term *patch* has more significance and helps us distinguish these limited regions. Second, many classes of parametric surfaces have been developed.

Figure 6.10 Three special surfaces.

Most are interconvertable, their differences being primarily in the way each class defines and modifies its underlying piecewise parametric surface.

6.4 HERMITE, BÉZIER, ANO B-SPLINE SURFACES: AN OVERVIEW

In the next three chapters we consider the Hermite, Bézier, and B-Spline parametric surfaces, including rational Bézier and nonuniform rational B-Spline representations. As with curves, of course, the nonrational forms are special cases of the rational forms. The hierarchical relationship between the

curves applies equally to these surfaces, with the rational B-Spline surface being the most general. We will consider only the *tensor product,* or rectangular, forms of these surfaces. There are, however, well-known triangular or other multisided varieties of each of them. See, for example, Farin (1993a), Greiner and Seidel (1994), Kato (1991), and Warren (1992).

The bicubic Hermite patch is defined by its four corner points, tangent vectors of the four boundary curves, and the so-called *twist vectors* at the corner points. Composite arrays of these patches may be assembled to represent complex surfaces. This surface representation scheme exhibits advantages and disadvantages similar to those for the Hermite curves.

The Bézier patch is defined by a rectangular array of control points, whose dimensions determine the degrees of the curves comprising the isoparametric curve net. Its boundary curves interpolate the corner points of the array and are tangent to the plane defined by the two edges of the control polyhedron at each corner point. It has the convex hull property, and like Bézier curves it is the easiest surface to subdivide. Local control is not a characteristic, because the effect of moving any control point is propagated globally throughout the patch.

The B-Spline surface is also defined by a rectangular array of control points. This form permits local control of curve shape. The degrees of its basis functions are defined independently of the control points. Although our focus in the following three chapters on surfaces is on the effect of control points and weights on surface shape, there are other important approaches beyond the scope of this text. For example, see Georgiades and Greenberg (1992) and Fowler (1992), who discuss geometric manipulation of the shape of these and other tensor product surfaces. Also see Fowler and Bartels (1993) for similar work on curves.

Comparative advantages and disadvantages of these forms, including implicit and others, are well-known. For example, see Farin (1989b, 1993a), Piegl (1989b), and Letcher and Shook (1995a, 1995b). Letcher and Shook summarize the deficiencies of the NURBS surface form as applied to volume gridding for computational fluid dynamic analysis, but their comments are also relevant to the entire group of parametric forms. Six categories of concern are the following:

1. Division by zero (for the rational forms)
2. Nonuniform parameterization of arcs (particularly important in CAD/CAM)
3. Surfaces which cannot be accurately represented efficiently (for example, helicoidal surfaces are mentioned as common engineering and manufacturing surfaces having no exact NURBS representation)

4. Irregular isoparametric surface meshes resulting from the use of nonuniform weights (for the rational forms)
5. Ill-conditioned basis for surface fitting
6. Lack of closure under common geometric operations (for example, composition, projection, and intersection)

Other descriptive deficiencies have been pointed out by Requicha and Rossignac (1992): Fixed degree NURBS cannot describe offset surfaces exactly, and NURBS representation for quadrics or other canonical forms hide the nature of these surfaces. For example, it may be useful to know that the surface of a hole is a circular cylinder; the test to determine if a NURBS surface is indeed a cylinder is a numerically unstable computation. Of course, sophisticated modeling systems work around these problems, but their existence is a stimulus to search for alternative forms.

6.5 POINTS ON A SURFACE

We discussed different approaches to direct- and inverse-point evaluation on curves in Section 2.6. Point evaluation on a surface is an extension of those techniques, plus some additional steps. To evaluate directly a more or less regular distribution of points on a surface corresponding to a similarly regular distribution of parameter values, we first isolate a sequence of curves of constant parameter value on the surface and then evaluate points along each of these curves. Changing the parametric net by reparameterization often results in more efficient point evaluation (for example, increasing the density of points to be evaluated where curvature is greater). The inverse-point solution for surfaces is somewhat more subtle.

Because the inverse-point solution is one of the most important analytic operations, let us explore a somewhat more general strategy for surfaces than that used for curves. If we talk about N-dimensional space, then vectors constructed in it will have N components. To define a surface (or hypersurface) in these spaces requires $(N-1)$ parametric variables.

In the direct-solution mode, the parametric equations yield the N coordinate components of $\mathbf{p}$, as well as their various derivatives with respect to the $(N-1)$ parametric variables u_j, all at some point on the surface located by specifying a value for each of the parametric variables. However, in many problems of practical interest, a point is located by specifying values for J of the N coordinates of $\mathbf{p}$. For most problems $J \neq N-1$, so that the inverse may not be unique. In fact, when $J < N-1$, the inverse degenerates to a surface of $(N-J-1)$ dimensions. When $J = N$, the problem is somewhat overconstrained, so we simply ignore one of the coordinates to obtain a determinate system.

We let $\mathbf{p}'$ denote a specific point on the surface (or hypersurface) on which we must compute the first derivatives $\partial\mathbf{p}/\partial u_i$. We know the N coordinates or components of this point, but we do not know the values of the parametric variables u_i at the point; therefore, we cannot evaluate $\partial\mathbf{p}/\partial u_i$ directly. A related inversion problem develops if we are given only $(N-1)$ components of $\mathbf{p}'$ and must find the Nth component. The resulting matrix algebra required to invert and solve the system of equations defining these problems defies solution, and we are forced to use an iterative procedure. To illustrate the problem of finding the tangent vectors at a point $\mathbf{p}'$ on a surface $\mathbf{p}$, we consider the vector $\mathbf{r} = \mathbf{p}' - \mathbf{p}$. If $|\mathbf{r}|$ is small, then $\mathbf{r}$ approximates a vector tangent to the surface. A good guess is to start the iteration at $\mathbf{p}(1/2, 1/2, 1/2, \ldots, 1/2)$, the parametric center of the hypersurface. Now, we define a unit vector $\hat{\mathbf{t}} = \mathbf{r}/|\mathbf{r}|$ so that

$$\hat{\mathbf{t}} = \sum_{i=1}^{N-1} \frac{\partial\mathbf{p}/\partial u_i}{du_i/ds} \tag{6.16}$$

or in matrix form

$$[\hat{\mathbf{t}}] = \left[\frac{\partial\mathbf{p}}{\partial u_i}\right]\left[\frac{du_i}{ds}\right] = \left[\frac{\mathbf{r}}{|\mathbf{r}|}\right] \tag{6.17}$$

or

$$\left[\frac{du_i}{ds}\right] = \left[\frac{\partial\mathbf{p}}{\partial u_i}\right]^{-1}\left[\frac{\mathbf{r}}{|\mathbf{r}|}\right] \tag{6.18}$$

If $\Delta s \cong |\mathbf{r}|$, then the incremental changes Δu_i in the parametric variables are

$$[\Delta u_i] = \left[\frac{\partial\mathbf{p}}{\partial u_i}\right]^{-1}[\mathbf{r}] \tag{6.19}$$

Again, we compute the elements of $[\partial\mathbf{p}/\partial u_i]$ at the initial guess $\mathbf{p}(1/2, 1/2, 1/2, \ldots, 1/2)$, and $\mathbf{r}$ from $\mathbf{r} = \mathbf{p}' - \mathbf{p}$. Next, we appropriately increment each of the parametric variables by Δu_i to determine a new $\mathbf{p}$, and continue the procedure until we satisfy some convergence criterion $|\mathbf{r}| \le \varepsilon$.

6.6 CURVE NETS

Two related subjects are of interest here: the general formulation of curves embedded in a surface, and networks of curves on surfaces. The curve nets of interest are the *parametric, orthogonal,* and *conjugate nets,* all of which have special properties. Other types of curves on surfaces include lines of constant or equal curvature and geodesics. Our goal is to become aware of

certain systems of curves on surfaces we can use when applying surfaces to modeling problems.

Of course, the simplest and most obvious curve net on a patch consists of the defining *isoparametric curves*, that is, those curves on a patch where one of the parameters varies while the other is constant. If the parameter w is fixed while u varies, the locus of the point $\mathbf{p}(u,w)$ is a w = constant curve on the patch. By continuing this process, we can define a one-parameter family of curves that completely covers the patch. Next, we interchange the roles of u and w and define a one-parameter family of u = constant curves. Thus, these two families of curves together constitute the parametric curve net. A curve net on a patch $\mathbf{p}(u,w)$ is two one-parameter families of curves constructed so that through each point there passes just one of each family. Furthermore, the two tangents of the curves at the point must be distinct.

A net may be *covariant* to its sustaining patch; that is, it can be defined in a way that is invariant under rigid motion in space. However, this is not necessarily always the case. With a suitably chosen transformation of parameters, the parametric net can be made to coincide with any specified net on the surface. Although we will not develop the proofs here, we can transform any covariant net to a parametric net. This allows us to conclude that the geometry of a surface is independent of the analytical representation used and is, therefore, independent of the choice of parametric nets. If the parametric net is a covariant net, then we find that properties of the net are also properties of the surface itself.

We assert without proof that the only transformations of parameters that leave the parametric net invariant are

$$r = r(u) \qquad s = s(w) \tag{6.20}$$

and

$$r = r(w) \qquad s = s(u) \tag{6.21}$$

from u,w to r,s, or from u,w to s,r. Finally, we see that the normal vector at a point on a surface is invariant, except perhaps for its sense.

Next, we consider the more specialized orthogonal net of curves on a surface. An *orthogonal net* is a net such that at every point of the surface, the two curves of the net intersect at right angles. We immediately conclude that a necessary and sufficient condition for the parametric net on a patch $\mathbf{p}(u,w)$ to be orthogonal is

$$\mathbf{p}^u \cdot \mathbf{p}^w = 0 \tag{6.22}$$

An orthogonal net is useful in many analytical engineering applications of geometric modeling; for example, in structural analysis, fluid flow, and ther-

modynamics. Here the orthogonality of the parametric field of an object's or phenomenon's geometric characteristics can often be made compatible with the orthogonality of the driving and response functions, and thus facilitate analysis.

A net of curves on a surface is a *conjugate net* when the tangents of the curves of one family of the net at points along each fixed curve of the other family form a developable surface (see Figure 6.11). The two families of the net do not play symmetrical roles in this definition, but the two families are interchangeable. A necessary condition for the parametric net on a patch to be a conjugate net is

$$\mathbf{p}^{uw} \cdot \hat{\mathbf{n}} = 0 \tag{6.23}$$

6.7 EMBEOOEO CURVES

A surface provides a two-dimensional space suitable for supporting the vector analytic representation of curves. The two parametric variables defining the surface serve to supply the coordinate grid on which we may define *embedded curves*. This inherent characteristic of the parametric representation is an invaluable asset to many aspects of geometric modeling, including surfaces with irregular boundaries and the intersection and union of primitive shapes.

The curve on the surface in Figure 6.12 is defined in the u,w parameter space of the patch. Here we introduce a new parametric variable to distinguish the curve formulation, and we denote the curve as $\mathbf{c}(t)$. Note that in the u,w parametric plane, the curve is necessarily a plane curve. Also note that points on the curve have vector components in the u and w directions. Thus,

$$\mathbf{c}(t) = \mathbf{u}(t) + \mathbf{w}(t) \tag{6.24}$$

Figure 6.11 Conjugate net.

we can define this curve in many ways in the u,w plane: explicitly, implicitly, with Bézier or B-Spline forms, with the parametric cubic form, and so on. We define $\mathbf{c}(t)$ in Hermite form as

$$\mathbf{c}(t) = \mathbf{T}\mathbf{M}_F\mathbf{B}_C \tag{6.25}$$

where

$$\mathbf{T} = [t^3 \quad t^2 \quad t \quad 1]$$

$$\mathbf{B}_C = [\mathbf{c}_0 \quad \mathbf{c}_1 \quad \mathbf{c}_0' \quad \mathbf{c}_1']^T$$

and $\mathbf{M}_F$ is the Hermite basis transformation matrix. The parametric component curves are sketched in their respective u,t and w,t parameter spaces.

Figure 6.12 Curves on surfaces.

To determine the coordinates of points on this curve in model space, we first compute pairs of u,w values for successive values of t. We substitute these u,w values into the surface equation, say, $\mathbf{p}(u,w) = \mathbf{UMB}_S\mathbf{M}^T\mathbf{W}^T$, to obtain a sequence of x,y,z coordinates of points in object space. These points are necessarily on the patch. This process selects only those points on the patch that satisfy the curve equation. We can state this as

$$\mathbf{p}[u(t),w(t)] = \text{points on the curve, embedded in the patch} \tag{6.26}$$

We find a model-space tangent vector $\mathbf{p}_t'$ to the curve at any point t on it by first computing $\mathbf{c}_t'$, the tangent vector to the curve at $u(t),w(t)$ in the u,w plane. We determine the direction cosines of this vector (that is, with respect to the u,w coordinate system) and denote them k_u and k_w. Next we compute $\mathbf{p}_{uw}^u$ and $\mathbf{p}_{uw}^w$ at the u,w point determined by $\mathbf{c}(t)$ at t and find that

$$\mathbf{p}_t' = k_u \frac{\mathbf{p}_{uw}^u}{|\mathbf{p}_{uw}^u|} + k_w \frac{\mathbf{p}_{uw}^w}{|\mathbf{p}_{uw}^w|} \tag{6.27}$$

It is the direction of $\mathbf{p}_t'$ that interests us; the magnitude has no particular meaning.

A different approach requires only the control points of a Bézier or B-Spline curve to be embedded in a surface. We compute the corresponding model-space coordinates of the control points and then construct the curve itself as usual. However, the curve will lie only approximately in the surface. The advantage of this approach is that point evaluations do not require transformation from parameter space to model space via the surface equation.

We can decompose the total surface of a physical object into a collection of smaller surface regions (see Figure 6.13). Each of these regions is characterized by well-behaved contours, reaching practical limits at abrupt or discontinuous changes in shape. We establish the boundaries of a region at these discontinuities. In general, the curves where regions join do not coincide with the isoparametric curves of the individual patches comprising the surface. We may segment the total surface of a model in this way so that each region is itself then mathematically tractable, where often the total surface is not. We map each region and its boundary curves onto a unit square or $m \times n$ rectangle in parameter space.

Earlier we defined a surface in parametric form by specifying analytical relationships between the model-space coordinates and two parameters, u and w : $\mathbf{p}(u,w) = [x(u,w) \quad y(u,w) \quad z(u,w)]$. We can think of these parametric variables as surface coordinates. Every point on a surface has not only the familiar triplet of Cartesian coordinates x,y,z, but also a unique pair of parametric coordinates u,w. We have interpreted this as a mapping of the surface onto a

Figure 6.13 Decomposition of a complex shape.

unit square in the u,w coordinate plane, where we derived the parametric equations of the surface from certain boundary conditions or control points.

As mentioned above, the boundary curves of a surface are frequently irregular. For real objects, they may not correspond to the limits of the parametric surface established by the domain of the parametric variables. Usually by intent, the underlying parametric surface always equals or exceeds in extent the object's surface region that we are modeling. Thus, boundary curves may be necessary to delimit a subset of points on the patch. The points in this subset are in a one-to-one correspondence with points on the real object's surface.

In Figure 6.14, we see that two of the boundaries of a surface R are established by intersections with surfaces S_1 and S_2, which may or may not themselves contain model surface regions. We denote these boundary curves as $\mathbf{b}_1$ and $\mathbf{b}_2$, and map them onto the unit square of R in parameter space, where we express them as parametric equations of the form

$$\mathbf{b}_i = [u_i(t) \quad w_i(t)] \qquad t \in [0,1] \tag{6.28}$$

After we compute the intersections as sets of u,w points, we apply curve-fitting techniques to define these curves. To obtain the boundary curve in model space, we compute points on the u,w coordinate plane from $u_i(t)$ and $w_i(t)$. We then substitute the values of u,w into the appropriate surface equation to obtain x,y,z points on the curve. The resulting surface with these irregular boundaries is called a *trimmed patch*. The curves themselves are called *trimmed-patch boundary elements*. For an elegant entry into this sub-

ject see Casale (1987), and Casale, Bobrow, and Underwood (1992). Other references include Miller (1986), Farin (1993a), Shantz and Chang (1988), and Farouki and Hinds (1985).

We may need to decide if a given point on the unit square lies within a bounded region. This is called *point classification*. A somewhat brute-force way to do this is to specify a point **q** that we know to lie in this bounded region. Then, given any other point **p**, we can easily test whether or not it lies in the region as follows. We construct the straight line **p** − **q** in the u,w plane. We then compute the number of intersections this line makes with each of the boundary curves $\mathbf{b}_i$. If it does not intersect any of these curves, then **p** is in an active region. If the line intersects any boundary an odd number of

Figure 6.14 Surface with irregular boundary.

times, then **p** lies outside the region. All even intersections bring the **p** back into the region; Figure 6.14 shows several examples of this.

One way to symbolically represent a region R is as a function of the parametric surface S onto which it is mapped, the boundary curves $\mathbf{b}_i$, and a known interior point **q**.

$$\mathbf{R} = \Phi(S,\mathbf{b}_i,\mathbf{q}) \tag{6.29}$$

Here, S is an $m \times n$ rectangular array of parametric patches normalized on a unit square in the parametric plane u,w. $\mathbf{b}_i$ is a set of parametric curves mapped onto the same plane and expressed as functions of a parameter t; that is, $u = u(t)$ and $w = w(t)$. **q** is a point on the parametric plane inside the active region.

An alternative, and perhaps more elegant, way to indicate the active region is to specify the intersection of an appropriate set of two-dimensional half-spaces. Using this approach, we merely test each candidate point against the implicit functions defining the half-spaces (see Figure 6.15). More sophisticated point-classification methods are found in the references listed previously.

We stated earlier that we can decompose the total surface of any physical object into a collection of smaller surface regions. In Figure 6.16 we see an

Figure 6.15 Irregular surface defined by the intersection of half-spaces.

Figure 6.16 Surface of an object defined by a collection of smaller regions.

extrusion that has been slotted and truncated in a variety of ways, yet we easily decompose it into six regions, modeling each with a trimmed parametric surface.

There are six parametric surfaces. S_1 wraps around the entire cross section and defines the extrusion's longitudinal surface (before cuts and slots are

made). S_2 is a plane surface normal to S_1 that completely defines one end plane of the extrusion. S_3 is a closed cylindrical surface intersecting S_1 to form a slot in the extrusion. S_4 is an open cylindrical surface that takes a nick out of the bottom flange. S_5 is a plane normal to S_1 that establishes one facet of the near-end plane. S_6 is also a plane, but it takes an angular cut through S_1.

Each surface is a unique $m \times n$ array of parametric surface patches. We establish each region R on a surface by intersecting the surface with adjacent surfaces. We define the interiors by six points $\mathbf{q}_1 \cdots \mathbf{q}_6$. (Alternatively, we could use an expression defining the intersecting two-dimensional half-spaces, or we could use the right-left relationship to the direction of traversal of the bounding curve.) Figure 6.16 shows these points and region boundaries mapped onto their respective parametric surfaces in the unit square on the u,w plane.

To complete the definition of the object's total surface, we define a point $\mathbf{q}_v$, which we know to be inside the enclosed volume of the object. This point allows us to test (or classify) any other point $\mathbf{p}_v$ for the condition of being outside, on, or inside the volume. We do this by connecting $\mathbf{p}_v$ and $\mathbf{q}_v$ with a straight line. By counting intersections that occur between this line and each of the surface regions, we can determine the test results. If there are no intersections, then $\mathbf{p}_v$ is obviously inside the object. If the line intersects the bounding surface an odd number of times, then it lies outside the object. All even intersections bring the point back into the interior of the object. It is easy to determine if a point lies exactly on the surface of the object by testing the distance of $\mathbf{p}_v$ from each region.

7

THE BICUBIC HERMITE SURFACE

Two families of cubic Hermite curves forming a parametric net are the basis of the *bicubic Hermite surface.* A simple tensor product produces the 16-term polynomial, to which boundary conditions are applied, generating the geometric form. Four corner points, together with two tangent vectors and a twist vector at each of these points, are the necessary and sufficient conditions defining the bicubic Hermite patch. Sixteen points, four of which are interior, are also sufficient. Reparameterizing, truncating, and subdividing a patch proceed in much the same way as for curves. A large complex surface can be defined by a composite collection of simpler patches, while preserving certain levels of continuity.

7.1 ALGEBRAIC AND GEOMETRIC FORMS

The algebraic form of a bicubic Hermite patch is given by the tensor product

$$\mathbf{p}(u,w) = \sum_{i=0}^{3} \sum_{j=0}^{3} \mathbf{a}_{ij} u^i w^j \tag{7.1}$$

where

$$u,w \in [0,1] \tag{7.2}$$

The $\mathbf{a}_{ij}$ vectors are the *algebraic coefficients* of the patch. The reason for the term *bicubic* is obvious, because both parametric variables are cubic terms if

$\mathbf{a}_{33} \neq 0$. Notice that as with Hermite curves, the parametric variables u and w are usually restricted to values in the interval 0 to 1, inclusive. This makes the patch bounded in a regular way. We will discuss this restriction and the use of irregular boundaries later. The tensor product method allows us to describe a rectangular patch as a product of the curve-defining polynomials.

Expanding Equation (7.1) and arranging the $\mathbf{a}_{ij}$ terms in descending order produces a result similar to Equation (3.2) for curves:

$$\begin{aligned}\mathbf{p}(u,w) = {} & \mathbf{a}_{33}u^3w^3 + \mathbf{a}_{32}u^3w^2 + \mathbf{a}_{31}u^3w + \mathbf{a}_{30}u^3 \\ & + \mathbf{a}_{23}u^2w^3 + \mathbf{a}_{22}u^2w^2 + \mathbf{a}_{21}u^2w + \mathbf{a}_{20}u^2 \\ & + \mathbf{a}_{13}uw^3 + \mathbf{a}_{12}uw^2 + \mathbf{a}_{11}uw + \mathbf{a}_{10}u \\ & + \mathbf{a}_{03}w^3 + \mathbf{a}_{02}w^2 + \mathbf{a}_{01}w + \mathbf{a}_{00}\end{aligned} \tag{7.3}$$

This 16-term polynomial in u and w defines the set of all points lying on the patch, and it is the expanded algebraic form of the bicubic Hermite patch. Because each of the 16 vector coefficients $\mathbf{a}_{ij}$ has 3 independent components, there is a total of 48 algebraic coefficients, or 48 degrees of freedom. Thus, each vector component is

$$x(u,w) = a_{33x}u^3w^3 + a_{32x}u^3w^2 + a_{31x}u^3w + \cdots + a_{00x} \tag{7.4}$$

with similar expressions for $y(u,w)$ and $z(u,w)$.

In matrix notation the algebraic form is

$$\mathbf{p}(u,w) = \mathbf{UAW}^T \tag{7.5}$$

where

$$\mathbf{U} = [u^3 \quad u^2 \quad u \quad 1]$$

$$\mathbf{W} = [w^3 \quad w^2 \quad w \quad 1]$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_{33} & \mathbf{a}_{32} & \mathbf{a}_{31} & \mathbf{a}_{30} \\ \mathbf{a}_{23} & \mathbf{a}_{22} & \mathbf{a}_{21} & \mathbf{a}_{20} \\ \mathbf{a}_{13} & \mathbf{a}_{12} & \mathbf{a}_{11} & \mathbf{a}_{10} \\ \mathbf{a}_{03} & \mathbf{a}_{02} & \mathbf{a}_{01} & \mathbf{a}_{00} \end{bmatrix}$$

Because of the order of the elements in **U** and **W**, the subscripts of the vector elements in the **A** matrix correspond to those in Equation (7.3). They do not, in this instance, correspond to the normal indexing convention for matrices. Since the **a** elements are three-component vectors, the **A** matrix is actually a $4 \times 4 \times 3$ array.

As we found with the Hermite curves, the algebraic coefficients of a Hermite patch determine its shape and position in space. Patches of the same

size and shape have a different set of coefficients if they occupy different positions in space. Change any one of the 48 coefficients, and the result is a different patch. We generate a point on the patch each time we insert a specific pair of u,w values into Equation (7.5). Although the u,w values are restricted by Equation (7.2), the range of the variables x, y, and z is not restricted, because the range of the algebraic coefficients is not restricted.

A patch consists of an infinite number of points given by their x,y,z coordinates. An infinite number of pairs of u,w values also occurs in the corresponding parameter space. A unique pair of u,w values is associated with each point. Figure 7.1 shows a bicubic patch mapped into model space from its components in parameter space.

Figure 7.1 A bicubic Hermite patch mapped into model space from its components in parameter space.

A bicubic patch is bounded by four curves, and each boundary curve is a cubic Hermite curve. These curves are denoted as: $\mathbf{p}(u,0)$, $\mathbf{p}(u,1)$, $\mathbf{p}(0,w)$, and $\mathbf{p}(1,w)$, because they arise at the limit values of the parametric variables. Another way of denoting the boundary curves is by appropriately subscripting the vector $\mathbf{p}$. Thus, we may use $\mathbf{p}_{0w}$, $\mathbf{p}_{1w}$, $\mathbf{p}_{u0}$, $\mathbf{p}_{u1}$; their interpretation should be obvious. There are also four unique corner points $\mathbf{p}(0,0)$, $\mathbf{p}(1,0)$, $\mathbf{p}(0,1)$, and $\mathbf{p}(1,1)$, or $\mathbf{p}_{00}$, $\mathbf{p}_{10}$, $\mathbf{p}_{01}$, $\mathbf{p}_{11}$.

Reversing the sequence of the u,w parameterization does not change the shape of a surface. And, aside from any external modeling constraints (for example, the direction of surface normals), we have almost complete freedom to assign the parameterization scheme defining the boundary curves. Of course, $\mathbf{p}(0,w)$ must be opposite $\mathbf{p}(1,w)$ on the patch, and $\mathbf{p}(u,0)$ must be opposite $\mathbf{p}(u,1)$. Figure 7.2 shows a useful convention for the elements of a bicubic Hermite patch.

It is a simple exercise to show that a boundary curve *is* indeed a cubic Hermite curve. Let us investigate the curve $\mathbf{p}(u,0)$. The subscript indicates $w = 0$, so all terms containing w vanish, and Equation (7.3) becomes

$$\mathbf{p}(u,0) = \mathbf{a}_{30}u^3 + \mathbf{a}_{20}u^2 + \mathbf{a}_{10}u + \mathbf{a}_{00} \tag{7.6}$$

This, of course, is the expression for a cubic Hermite curve.

As we saw for curves, algebraic coefficients are not the most convenient way to define and control the shape of a patch, and they do not contribute to our understanding of surface behavior. Therefore, we must turn to the geo-

Figure 7.2 Elements of a bicubic Hermite patch.

metric form. We derive the geometric form from a relationship between certain boundary conditions and the algebraic form, in keeping with the approach used for the cubic Hermite curves. We will use the four patch corner points $\mathbf{p}_{00}, \mathbf{p}_{10}, \mathbf{p}_{01}, \mathbf{p}_{11}$ and the eight tangent vectors $\mathbf{p}^u_{00}, \mathbf{p}^w_{00}, \mathbf{p}^u_{10}, \mathbf{p}^w_{10}, \mathbf{p}^u_{01}, \mathbf{p}^w_{01}, \mathbf{p}^u_{11}, \mathbf{p}^w_{11}$. They, of course, define the boundary curves

$$\mathbf{p}(u,0) = \mathbf{F}[\mathbf{p}_{00} \quad \mathbf{p}_{10} \quad \mathbf{p}^u_{00} \quad \mathbf{p}^u_{10}]^T \tag{7.7}$$

$$\mathbf{p}(u,1) = \mathbf{F}[\mathbf{p}_{01} \quad \mathbf{p}_{11} \quad \mathbf{p}^u_{01} \quad \mathbf{p}^u_{11}]^T \tag{7.8}$$

$$\mathbf{p}(0,w) = \mathbf{F}[\mathbf{p}_{00} \quad \mathbf{p}_{01} \quad \mathbf{p}^w_{00} \quad \mathbf{p}^w_{01}]^T \tag{7.9}$$

$$\mathbf{p}(1,w) = \mathbf{F}[\mathbf{p}_{10} \quad \mathbf{p}_{11} \quad \mathbf{p}^w_{10} \quad \mathbf{p}^w_{11}]^T \tag{7.10}$$

These curves provide 12 of the 16 vectors needed to specify the 48 geometric coefficients. Four additional vectors are needed, and for these we will use the so-called *twist vectors*, one at each of the four corner points. Mathematically, we express the twist vectors as follows:

$$\mathbf{p}^{uw}_{00} = \frac{\partial^2 \mathbf{p}(u, w)}{\partial u\, \partial w} \qquad \text{at} \qquad u = 0, w = 0$$

$$\mathbf{p}^{uw}_{10} = \frac{\partial^2 \mathbf{p}(u, w)}{\partial u\, \partial w} \qquad \text{at} \qquad u = 1, w = 0$$

$$\mathbf{p}^{uw}_{01} = \frac{\partial^2 \mathbf{p}(u, w)}{\partial u\, \partial w} \qquad \text{at} \qquad u = 0, w = 1$$

$$\mathbf{p}^{uw}_{11} = \frac{\partial^2 \mathbf{p}(u, w)}{\partial u\, \partial w} \qquad \text{at} \qquad u = 1, w = 1$$

Calculating the mixed partial derivative of the function $\mathbf{p}(u,w)$ and evaluating the result at the indicated u,w values, from Equation (7.3) we obtain

$$\begin{aligned}\frac{\partial^2 \mathbf{p}(u, w)}{\partial u\, \partial w} &= 9\mathbf{a}_{33}u^2w^2 + 6\mathbf{a}_{32}u^2w + 3\mathbf{a}_{31}u^2 \\ &\quad + 6\mathbf{a}_{23}uw^2 + 4\mathbf{a}_{22}uw + 2\mathbf{a}_{21}u \\ &\quad + 3\mathbf{a}_{13}w^2 + 2\mathbf{a}_{12}w + \mathbf{a}_{11}\end{aligned}$$

When we evaluate this equation at the corner points, we obtain

$$\mathbf{p}^{uw}_{00} = \mathbf{a}_{11}$$

$$\mathbf{p}^{uw}_{10} = 3\mathbf{a}_{31} + 2\mathbf{a}_{21} + \mathbf{a}_{11}$$

$$\mathbf{p}^{uw}_{01} = 3\mathbf{a}_{13} + 2\mathbf{a}_{12} + \mathbf{a}_{11}$$

$$\mathbf{p}^{uw}_{11} = 9\mathbf{a}_{33} + 6\mathbf{a}_{32} + 3\mathbf{a}_{31} + 6\mathbf{a}_{23} + 4\mathbf{a}_{22} + 2\mathbf{a}_{21} + 3\mathbf{a}_{13} + 2\mathbf{a}_{12} + \mathbf{a}_{11}$$

Having gone this far with the twist vectors, we now do the same for the other 12 vectors. Thus, when we evaluate Equation (7.3), we obtain

$$\mathbf{p}_{00} = \mathbf{a}_{00}$$

$$\mathbf{p}_{10} = \mathbf{a}_{30} + \mathbf{a}_{20} + \mathbf{a}_{10} + \mathbf{a}_{00}$$

$$\mathbf{p}_{01} = \mathbf{a}_{03} + \mathbf{a}_{02} + \mathbf{a}_{01} + \mathbf{a}_{00}$$

$$\begin{aligned}\mathbf{p}_{11} = {} & \mathbf{a}_{33} + \mathbf{a}_{32} + \mathbf{a}_{31} + \mathbf{a}_{30} + \mathbf{a}_{23} + \mathbf{a}_{22} + \mathbf{a}_{21} + \mathbf{a}_{20} \\ & + \mathbf{a}_{13} + \mathbf{a}_{12} + \mathbf{a}_{11} + \mathbf{a}_{10} + \mathbf{a}_{03} + \mathbf{a}_{02} + \mathbf{a}_{01} + \mathbf{a}_{00}\end{aligned}$$

$$\mathbf{p}^{u}_{00} = \mathbf{a}_{10}$$

$$\mathbf{p}^{w}_{00} = \mathbf{a}_{01}$$

$$\mathbf{p}^{u}_{10} = 3\mathbf{a}_{30} + 2\mathbf{a}_{20} + \mathbf{a}_{10}$$

$$\mathbf{p}^{w}_{10} = \mathbf{a}_{31} + \mathbf{a}_{21} + \mathbf{a}_{11} + \mathbf{a}_{01}$$

$$\mathbf{p}^{u}_{01} = \mathbf{a}_{13} + \mathbf{a}_{12} + \mathbf{a}_{11} + \mathbf{a}_{10}$$

$$\mathbf{p}^{w}_{01} = 3\mathbf{a}_{03} + 2\mathbf{a}_{02} + \mathbf{a}_{01}$$

$$\mathbf{p}^{u}_{11} = 3\mathbf{a}_{33} + 3\mathbf{a}_{32} + 3\mathbf{a}_{31} + \mathbf{a}_{30} + 2\mathbf{a}_{23} + 2\mathbf{a}_{22} + 2\mathbf{a}_{21} + 2\mathbf{a}_{20} + \mathbf{a}_{13} + \mathbf{a}_{12} + \mathbf{a}_{11} + \mathbf{a}_{10}$$

$$\mathbf{p}^{w}_{11} = 3\mathbf{a}_{33} + 2\mathbf{a}_{32} + \mathbf{a}_{31} + 3\mathbf{a}_{23} + 2\mathbf{a}_{22} + \mathbf{a}_{21} + 3\mathbf{a}_{13} + 2\mathbf{a}_{12} + \mathbf{a}_{11} + 3\mathbf{a}_{03} + 2\mathbf{a}_{02} + \mathbf{a}_{01}$$

A geometric interpretation of the twist vectors is shown in Figure 7.3. We use the fact that a bicubic patch consists of two mutually orthogonal sets of curves—orthogonal in parameter space. They are the cubic curves defined by constant values of the parametric variables u_i and w_i. One set contains the $\mathbf{p}(0,w)$ and $\mathbf{p}(1,w)$ and an infinite number of intermediate $\mathbf{p}(u_i,w)$ curves, sweeping across a surface from one boundary to the other. A similar set of $\mathbf{p}(u,w_i)$ curves overlays these, forming an *orthogonal net.* Equations (7.7) to (7.10) are sufficient to define the boundary curves, but how do we define these intermediate curves, such as $\mathbf{p}(u,w_i)$?

We proceed as follows. First, we find the endpoints of $\mathbf{p}(u,w_i)$ by computing $\mathbf{p}(0,w_i)$ and $\mathbf{p}(1,w_i)$ at $w = w_i$ from Equations (7.9) and (7.10). Next, we must find the tangent vectors $\mathbf{p}^u(0,w_i)$ and $\mathbf{p}^u(1,w_i)$. To do this, we observe that, for the general case, $\mathbf{p}^u_{00}$ and $\mathbf{p}^u_{01}$ are not equal. Thus, $\mathbf{p}^u(0,w)$ must change as w changes from $w = 0$ through $w = w_i$ to $w = 1$. The same thing happens to the tangent vectors $\mathbf{p}^u(1,w)$ along curve $\mathbf{p}(1,w)$. Mathematically, we express this change in the tangent vectors as $(\partial/\partial w)(\mathbf{p}^u_{0w})$ and $(\partial/\partial w)(\mathbf{p}^u_{1w})$, but

$$\begin{aligned}\frac{\partial \mathbf{p}^u(0,w)}{\partial w} &= \frac{\partial^2 \mathbf{p}(0,w)}{\partial u\, \partial w} \\ &= \mathbf{p}^{uw}(0,w)\end{aligned} \tag{7.11}$$

Figure 7.3 Geometric interpretation of twist vectors.

and similarly

$$\frac{\partial \mathbf{p}^u(1,w)}{\partial w} = \frac{\partial^2 \mathbf{p}(1,w)}{\partial u\, \partial w} = \mathbf{p}^{uw}(1,w) \tag{7.12}$$

At the patch corners, these terms are $\mathbf{p}^{uw}_{00}, \mathbf{p}^{uw}_{01}, \mathbf{p}^{uw}_{10}, \mathbf{p}^{uw}_{11}$, the twist vectors of the set of boundary conditions we saw earlier.

There is an analogy between the way we handle a curve and the way we use the twist vectors to find the intermediate tangent vectors. Just as we specify points on a cubic Hermite curve by two endpoints and tangent vectors, we can specify intermediate tangent vectors along a boundary curve by the two end tangent vector values and corresponding twist vectors. We find points along a boundary curve $\mathbf{p}(0,w)$, for example, by using Equation (7.9)

$$\mathbf{p}(0,w) = \mathbf{W}\mathbf{M}_F[\mathbf{p}_{00} \quad \mathbf{p}_{01} \quad \mathbf{p}^w_{00} \quad \mathbf{p}^w_{01}]^T \tag{7.13}$$

We use an identical form to determine an intermediate tangent vector at a point on and orthogonally crossing a curve boundary:

$$\mathbf{p}^u(0,w) = \mathbf{W}\mathbf{M}^u_F[\mathbf{p}^u_{00} \quad \mathbf{p}^u_{01} \quad \mathbf{p}^{uw}_{00} \quad \mathbf{p}^{uw}_{01}]^T \tag{7.14}$$

Equation (7.14) describes a curve that we might visualize as being formed by the arrowhead endpoints of the intermediate tangent vectors as w varies

along the unit interval. Meriwether, who proposed this interpretation, calls this an *auxiliary curve.* Note that there are four auxiliary curves: $\mathbf{p}^u(0,w)$, $\mathbf{p}^u(1,w)$, $\mathbf{p}^w(0,u)$, and $\mathbf{p}^w(1,u)$. These curves are an abstract property of the general mathematical form of the bicubic Hermite patch. We cannot point to them on any physical model of a surface. The parametric components of the boundary curve $u = 1$ and its associated auxiliary curve are shown in Figure 7.3. Note that the ordinates are labeled $\mathbf{p}(1,w)$ and $\mathbf{p}^u_{1w}$, expediently denoting the x, y, or z components of these vectors.

Another interesting mathematical property of twist vectors is that at any point on the bicubic patch, $\mathbf{p}^{uw} = \mathbf{p}^{wu}$. That is, the order in which we take the mixed partial derivative is not important. This property means that u and w are completely interchangeable and no experiment on a physical manifestation of a bicubic patch can distinguish the u,w order.

Because the boundary conditions permit a complete and unambiguous definition of a bicubic patch as an orthogonal net of cubic Hermite curves, we can investigate any interior point or geometric property of a patch by selecting and analyzing an appropriate set of curves. Now we begin to see the patch in terms of Hermite curves in their geometric form. They, in turn, define the necessary patch boundary conditions.

We summarize these patch-defining curves by assembling them in a 4×4 matrix. We begin by filling the first two rows with the geometric coefficients of curves $\mathbf{p}(0,w)$ and $\mathbf{p}(1,w)$

$$\text{Row 1:} \quad \mathbf{p}(0,w) \rightarrow \mathbf{p}_{00} \quad \mathbf{p}_{01} \quad \mathbf{p}^w_{00} \quad \mathbf{p}^w_{01}$$

$$\text{Row 2:} \quad \mathbf{p}(1,w) \rightarrow \mathbf{p}_{1w} \quad \mathbf{p}_{11} \quad \mathbf{p}^w_{10} \quad \mathbf{p}^w_{11}$$

Next, we fill the first two columns with the geometric coefficients of curves $\mathbf{p}(u,0)$ and $\mathbf{p}(u,1)$, noting that some of these coefficients are already entered.

$$\begin{array}{llll}
\text{Column 1: } \mathbf{p}(u,0) & & & \\
\downarrow & \text{Column 2: } \mathbf{p}(u,1) & & \\
\downarrow & \downarrow & & \\
\mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}^w_{00} & \mathbf{p}^w_{01} \\
\mathbf{p}_{10} & \mathbf{p}_{11} & \mathbf{p}^w_{10} & \mathbf{p}^w_{11} \\
\mathbf{p}^u_{00} & \mathbf{p}^u_{01} & & \\
\mathbf{p}^u_{10} & \mathbf{p}^u_{11} & &
\end{array}$$

We have 12 of the 16 vectors entered into the matrix. Now, we must use the auxiliary curves to finish filling the matrix.

$$
\begin{array}{cccc|cc}
 & & \mathbf{p}(u,0) & \mathbf{p}(u,1) & \mathbf{p}^w(u,0) & \mathbf{p}^w(u,1) \\
 & & \downarrow & \downarrow & \downarrow & \downarrow \\
\mathbf{p}(0,w) & \rightarrow & \mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}^w_{00} & \mathbf{p}^w_{01} \\
\mathbf{p}(1,w) & \rightarrow & \mathbf{p}_{10} & \mathbf{p}_{11} & \mathbf{p}^w_{10} & \mathbf{p}^w_{11} \\
\hline
\mathbf{p}^u(0,w) & \rightarrow & \mathbf{p}^u_{00} & \mathbf{p}^u_{01} & \mathbf{p}^{uw}_{00} & \mathbf{p}^{uw}_{01} \\
\mathbf{p}^u(1,w) & \rightarrow & \mathbf{p}^u_{10} & \mathbf{p}^u_{11} & \mathbf{p}^{uw}_{10} & \mathbf{p}^{uw}_{11}
\end{array}
$$

Conversely, from this completed matrix, we can easily extract the geometric coefficients of the four boundary curves and the four auxiliary curves.

When we partition the matrix into quadrants, we observe that the four vectors in the upper left define the four corner points. The upper right quadrant contains the tangent vectors with respect to w at the corner points, while those with respect to u are in the lower left. Finally, the lower right quadrant contains the twist vector or cross-derivatives at the corner points. The pattern of these vector subscripts repeats in a regular way.

Using this matrix of geometric coefficients, we can evaluate a point on the patch at a specific pair of u,w values, say, u_i and w_j. Figure 7.4 illustrates the geometry of this situation. Here we see that the point of interest lies at the intersection of the curves $\mathbf{p}(u_i,w)$ and $\mathbf{p}(u,w_j)$. This problem is easily reduced to finding a point on a curve at a given value of a parametric variable. We can work with either curve, so we arbitrarily choose to begin with $\mathbf{p}(u_i,w)$. First, we determine the geometric coefficients of this curve, $\mathbf{p}(u_i,0)$, $\mathbf{p}(u_i,1)$, $\mathbf{p}^w(u_i,0)$, and $\mathbf{p}^w(u_i,1)$. With these coefficients, we can then determine the coordinates of the point $\mathbf{p}(u_i,w_j)$.

Figure 7.4 Determining the coordinates of a point on a bicubic Hermite patch.

We compute $\mathbf{p}(u_i,0)$ using curve $\mathbf{p}(u,0)$:

$$\mathbf{p}(u_i,0) = F_1(u_i)\mathbf{p}_{00} + F_2(u_i)\mathbf{p}_{10} + F_3(u_i)\mathbf{p}^u_{00} + F_4(u_i)\mathbf{p}^u_{10} \tag{7.15}$$

We compute $\mathbf{p}(u_i,1)$ using $\mathbf{p}(u,1)$:

$$\mathbf{p}(u_i,1) = F_1(u_i)\mathbf{p}_{01} + F_2(u_i)\mathbf{p}_{11} + F_3(u_i)\mathbf{p}^u_{01} + F_4(u_i)\mathbf{p}^u_{11} \tag{7.16}$$

We compute $\mathbf{p}^w(u_i,0)$ using $\mathbf{p}^w(u,0)$, an auxiliary curve:

$$\mathbf{p}^w(u_i,0) = F_1(u_i)\mathbf{p}^w_{00} + F_2(u_i)\mathbf{p}^w_{10} + F_3(u_i)\mathbf{p}^{uw}_{00} + F_4(u_i)\mathbf{p}^{uw}_{10} \tag{7.17}$$

Finally, we compute $\mathbf{p}^w(u_i,1)$ using $\mathbf{p}^w(u,1)$, another auxiliary curve:

$$\mathbf{p}^w(u_i,1) = F_1(u_i)\mathbf{p}^w_{01} + F_2(u_i)\mathbf{p}^w_{11} + F_3(u_i)\mathbf{p}^{uw}_{01} + F_4(u_i)\mathbf{p}^{uw}_{11} \tag{7.18}$$

Now that we have the geometric coefficients of the curve $\mathbf{p}(u_i,w)$, we evaluate it at w_j, substituting from Equations (7.15) to (7.18) as follows:

$$\begin{aligned}\mathbf{p}(u_i,w_j) = {} & F_1(w_j)[F_1(u_i)\mathbf{p}_{00} + F_2(u_i)\mathbf{p}_{10} + F_3(u_i)\mathbf{p}^u_{00} + F_4(u_i)\mathbf{p}^u_{10}] \\ & + F_2(w_j)[F_1(u_i)\mathbf{p}_{01} + F_2(u_i)\mathbf{p}_{11} + F_3(u_i)\mathbf{p}^u_{01} + F_4(u_i)\mathbf{p}^u_{11}] \\ & + F_3(w_j)[F_1(u_i)\mathbf{p}^w_{00} + F_2(u_i)\mathbf{p}^w_{10} + F_3(u_i)\mathbf{p}^{uw}_{00} + F_4(u_i)\mathbf{p}^{uw}_{10}] \\ & + F_4(w_j)[F_1(u_i)\mathbf{p}^w_{01} + F_2(u_i)\mathbf{p}^w_{11} + F_3(u_i)\mathbf{p}^{uw}_{01} + F_4(u_i)\mathbf{p}^{uw}_{11}]\end{aligned} \tag{7.19}$$

We could obtain the same result by using curve $\mathbf{p}(u,w_j)$ as the initial working curve. Equation (7.19) can certainly be reduced to matrix form. We will do this and at the same time generalize the parametric variables by dropping the subscripts. We cannot yet drop the function notation, but we can abbreviate the equation to read

$$\begin{aligned}\mathbf{p}(u,w) = {} & [F_1(u) \quad F_2(u) \quad F_3(u) \quad F_4(u)]\begin{bmatrix}\mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}^w_{00} & \mathbf{p}^w_{01} \\ \mathbf{p}_{10} & \mathbf{p}_{11} & \mathbf{p}^w_{10} & \mathbf{p}^w_{11} \\ \mathbf{p}^u_{00} & \mathbf{p}^u_{01} & \mathbf{p}^{uw}_{00} & \mathbf{p}^{uw}_{01} \\ \mathbf{p}^u_{10} & \mathbf{p}^u_{11} & \mathbf{p}^{uw}_{10} & \mathbf{p}^{uw}_{11}\end{bmatrix} \\ & \times [F_1(w) \quad F_2(w) \quad F_3(w) \quad F_4(w)]^T\end{aligned} \tag{7.20}$$

Denoting the 4×4 matrix of geometric coefficients as $\mathbf{B}$, and simplifying the $F_i(u)$ and $F_i(w)$ arrays to $\mathbf{F}(u)$ and $\mathbf{F}(w)$ [notice that $\mathbf{F}(u)$ and $\mathbf{F}(w)$ are row vectors] produces

$$\mathbf{p}(u,w) = \mathbf{F}(u)\mathbf{B}\mathbf{F}(w)^T \tag{7.21}$$

Now we eliminate the function notation and use the Hermite basis-function transformation matrix. Because it is also true that $\mathbf{F}(u) = \mathbf{U}\mathbf{M}_F$, and because u and w are symmetrical and interchangeable parametric variables, it is also

true that $\mathbf{F}(w) = \mathbf{W}\mathbf{M}_F$. From matrix algebra, we have $\mathbf{F}(w)^T = \mathbf{M}_F^T\mathbf{W}^T$. With appropriate substitution into Equation (7.21), we derive the conventional geometric form of a bicubic patch using matrix notation

$$\mathbf{p}(u,w) = \mathbf{U}\mathbf{M}_F\mathbf{B}\mathbf{M}_F^T\mathbf{W}^T \tag{7.22}$$

By comparing Equation (7.22) to Equation (3.12), we see the relationship between the algebraic and geometric forms and see that we can convert from one form to the other with the following formulas:

$$\mathbf{A} = \mathbf{M}_F\mathbf{B}\mathbf{M}_F^T \tag{7.23}$$

and

$$\mathbf{B} = \mathbf{M}_F^{-1}\mathbf{A}\mathbf{M}_F^{-T} \tag{7.24}$$

7.2 HERMITE PATCH BASIS FUNCTIONS

The basis functions for the bicubic Hermite patch have the same form and serve the same purpose as the basis functions for the cubic Hermite curve. Because the patch has two independent parametric variables, we use the basis functions twice. We see this in Equation (7.21) and, of course, in the development preceding it.

We apply the same tangent-vector basis functions of Equation (3.7) to patches and obtain the two principal tangent vectors and the twist vector:

$$\begin{aligned}
\mathbf{p}^u(u,w) &= \mathbf{F}^u(u)\mathbf{B}\mathbf{F}(w)^T \\
\mathbf{p}^w(u,w) &= \mathbf{F}(u)\mathbf{B}\mathbf{F}^w(w)^T \\
\mathbf{p}^{uw}(u,w) &= \mathbf{F}^u(u)\mathbf{B}\mathbf{F}^w(w)^T
\end{aligned}$$

or

$$\begin{aligned}
\mathbf{p}^u(u,w) &= \mathbf{U}\mathbf{M}_F^u\mathbf{B}\mathbf{M}_F^T\mathbf{W}^T \\
\mathbf{p}^w(u,w) &= \mathbf{U}\mathbf{M}_F\mathbf{B}\mathbf{M}_F^{wT}\mathbf{W}^T \\
\mathbf{p}^{uw}(u,w) &= \mathbf{U}\mathbf{M}_F^u\mathbf{B}\mathbf{M}_F^{wT}\mathbf{W}^T
\end{aligned} \tag{7.25}$$

Remember that $\mathbf{M}_F^u$ (and $\mathbf{M}_F^w$) is the Hermite basis transformation matrix for the tangent vector at a point on a cubic Hermite curve.

We can use a tensor product to express the geometric form as

$$\mathbf{p}(u,w) = \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{b}_{ij}F_{i,m}(u)F_{j,n}(w) \tag{7.26}$$

This is analogous to the algebraic form expressed in Equation (7.1). Again, we must take care in how we interpret the subscripts on $\mathbf{b}_{ij}$, $F_{i,m}(u)$, and $F_{j,n}(w)$, because we have shifted their limits. The consequences of $m \neq n$ are not explained here.

7.3 TANGENT AND TWIST VECTORS

Computing and understanding the significance of the parametric derivatives of the bicubic Hermite patch basis functions are similar to these same processes in the cubic Hermite curve. The difference is that the bicubic functions are expressed in *two* independent variables instead of just the one for curves. This means that we must deal with partial derivatives. Earlier, we developed expressions for the tangent and twist vectors and extended the notation system we used for curves. Let us briefly review these procedures.

The *tangent vectors* are the partial derivatives

$$\mathbf{p}^u(u,w) = \frac{\partial \mathbf{p}(u,w)}{\partial u} \qquad \text{and} \qquad \mathbf{p}^w = \frac{\partial \mathbf{p}(u,w)}{\partial w} \tag{7.27}$$

The *twist vector* is the mixed partial derivative

$$\mathbf{p}^{uw}(u,w) = \frac{\partial^2 \mathbf{p}(u,w)}{\partial u\, \partial w} \tag{7.28}$$

We can use these vectors to change the shape of a patch, taking the same approach here that we did with curves. We can think of the patch as being defined by a *curve net*. Changing the curves in the net changes the patch. To investigate the effect of manipulating the tangent and twist vectors, we recompose the **B** matrix by expressing these vectors in terms of unit vectors and scalar magnitudes:

$$\mathbf{B} = \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} & k_{00}\hat{\mathbf{t}}^w_{00} & k_{01}\hat{\mathbf{t}}^w_{01} \\ \mathbf{p}_{10} & \mathbf{p}_{11} & k_{10}\hat{\mathbf{t}}^w_{10} & k_{11}\hat{\mathbf{t}}^w_{11} \\ l_{00}\hat{\mathbf{t}}^w_{00} & l_{01}\hat{\mathbf{t}}^w_{01} & m_{00}\hat{\mathbf{t}}^{uw}_{00} & m_{01}\hat{\mathbf{t}}^{uw}_{01} \\ l_{10}\hat{\mathbf{t}}^w_{10} & l_{11}\hat{\mathbf{t}}^w_{11} & m_{10}\hat{\mathbf{t}}^{uw}_{10} & m_{11}\hat{\mathbf{t}}^{uw}_{11} \end{bmatrix}$$

Notice that we must now add a superscript u, w, or uw to the unit tangent vector $\hat{\mathbf{t}}$, because there are two independent variables and two distinct directions on the patch.

We change one or more of the boundary curves by changing the appropriate set of scalars. However, we notice that a change to any of the scalar multipliers of the twist vectors does not change any of the four boundary

curves. This is interesting. We can fix the four corner points and boundary curves and still change the interior shape of a patch through operations on the twist vector magnitudes (see Figure 7.5). The figure indicates the result of a change in the twist vector at $\mathbf{p}_{11}$, where $\mathbf{p}_{11}^{uw} \rightarrow (\mathbf{p}_{11}^{uw})'$. The dashed curve net on the patch highlights the effect of this change on the interior. We note that the boundary curves are unaffected.

A patch with all twist vectors equal to zero is called an F-patch, after Ferguson, its developer. Here, only C^1 continuity is possible across the boundaries of adjacent F-patches, because these patches are constrained to have zero cross-derivatives at their corners. This can lead to surfaces that are not smooth enough for some applications. However, it is easy to construct and modify F-patches, and they are adequate for many geometric-modeling situations. The matrix of geometric coefficients then becomes

Figure 7.5 The effect of twist vectors on the interior of a bicubic Hermite patch. The dashed curves on the patch show how it changes when boundary conditions change.

$$\mathbf{B} = \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}^{w}_{00} & \mathbf{p}^{w}_{01} \\ \mathbf{p}_{10} & \mathbf{p}_{11} & \mathbf{p}^{w}_{10} & \mathbf{p}^{w}_{11} \\ \mathbf{p}^{u}_{00} & \mathbf{p}^{u}_{01} & 0 & 0 \\ \mathbf{p}^{u}_{10} & \mathbf{p}^{u}_{11} & 0 & 0 \end{bmatrix} \tag{7.29}$$

7.4 NORMALS

The *unit normal* is almost indispensable in geometric modeling, and in most applications a consistent normal direction is required. For example, it is common practice for the normal to point outward from the surface of a solid model. Computing silhouette curves, hidden surfaces, shadows, and shading effects requires information about surface normals. At any point $\mathbf{p}(u,w)$ on a bicubic Hermite patch, we can construct a vector that is perpendicular to it. This is the *unit normal vector* $\hat{\mathbf{n}}(u,w)$, which is a function of the tangent vectors $\mathbf{p}^u$ and $\mathbf{p}^w$ at the point $\mathbf{p}(u,w)$. The unit normal is given by

$$\hat{\mathbf{n}}(u,w) = \frac{\mathbf{p}^u \times \mathbf{p}^w}{|\mathbf{p}^u \times \mathbf{p}^w|} \tag{7.30}$$

It is often convenient to express this as $k_n\hat{\mathbf{n}}(u,w) = \mathbf{p}^u \times \mathbf{p}^w$, where $k_n = |\mathbf{p}^u \times \mathbf{p}^w|$. The order in which the vector product is taken determines the direction of $\hat{\mathbf{n}}(u,w)$. It turns out that we can interpret $\hat{\mathbf{n}}(u,w)$ itself as a patch, the *normals patch*, and we will see later that this proves to be a very useful interpretation.

A convention for assigning identifying numbers to the corner points and boundary curves is the subject of Figure 7.6. Here we see that if the fingers of the right hand curl around the patch in the direction of ascending curve or corner-point numbering, then the thumb points in the direction of the *positive* or outward surface normal as defined by Equation (7.30). This convention gives a consistent algebraic sign when summing the area of several surface patches.

A more convenient way to write Equation (7.30), one that takes advantage of existing algorithms for the bicubic form, is

$$\begin{aligned} k_n\mathbf{n} &= [x^u \quad y^u \quad z^u] \times [x^w \quad y^w \quad z^w] \\ &= [(y^u z^w - y^w z^u) \quad (z^u x^w - x^u z^w) \quad (x^u y^w - y^u x^w)] \end{aligned}$$

The x component of $\hat{\mathbf{n}}$ is

$$\hat{n}_x = \frac{y^u z^w - y^w z^u}{k_n}$$

Figure 7.6 Normal vector to a patch.

To rewrite each of the terms y^u, z^w, y^w, and z^u in their matrix form, we make use of $\mathbf{p}^u(u,w) = \mathbf{U}\mathbf{M}_F^u\mathbf{B}\mathbf{M}_F^T\mathbf{W}^T$ and $\mathbf{p}^w(u,w) = \mathbf{U}\mathbf{M}_F\mathbf{B}\mathbf{M}_F^{wT}\mathbf{W}^T$ of Equations (7.26), and find that

$$\begin{aligned}\hat{n}_x = \frac{1}{k_n}[&(\mathbf{U}\mathbf{M}_F^u\mathbf{B}_z\mathbf{M}_F^T\mathbf{W}^T)(\mathbf{U}\mathbf{M}_F\mathbf{B}_z\mathbf{M}_F^{wT}\mathbf{W}^T)\\ &-(\mathbf{U}\mathbf{M}_F\mathbf{B}_y\mathbf{M}_F^{wT}\mathbf{W}^T)(\mathbf{U}\mathbf{M}_F^u\mathbf{B}_z\mathbf{M}_F^T\mathbf{W}^T)]\end{aligned} \tag{7.31}$$

and similarly for $\hat{n}_y$ and $\hat{n}_z$. This is a biquintic polynomial in u and w and is relatively expensive to compute. However, if the surface is well behaved (not self-intersecting, without undulations, and so on), we can instead approximate the normal at any point on the surface with a bicubic expression, so that $\hat{\mathbf{n}}(u,w)$ becomes

$$\hat{\mathbf{n}}(u,w) = \mathbf{U}\mathbf{M}_F\mathbf{B}_n\mathbf{M}_F^T\mathbf{W}^T \tag{7.32}$$

where

$$\mathbf{B}_{nx} = \left[\begin{array}{c|c}\hat{n}_x & \hat{n}_x^w\\ \hline \hat{n}_x^u & \hat{n}_x^{uw}\end{array}\right], \qquad \mathbf{B}_{ny} = \left[\begin{array}{c|c}\hat{n}_y & \hat{n}_y^w\\ \hline \hat{n}_y^u & \hat{n}_y^{uw}\end{array}\right], \qquad \mathbf{B}_{nz} = \left[\begin{array}{c|c}\hat{n} & \hat{n}_z^w\\ \hline \hat{n}_z^u & \hat{n}_z^{uw}\end{array}\right]$$

We have already derived $\hat{n}_x$, $\hat{n}_y$, and $\hat{n}_z$. We find the three remaining components of $\mathbf{B}_{nx}$ by appropriate differentiation of $\hat{n}_x$. The matrix algebra is straightforward but tedious, producing rather lengthy expressions. Nothing is gained by presenting them here. Nonetheless, evaluating each of the resulting expressions at $u,w \in [0:1]$, that is, at (0,0), (1,0), (0,1), (1,1), determines

the 16 elements of $\mathbf{B}_{nx}$. Similar matrices for $\mathbf{B}_{ny}$ and $\mathbf{B}_{nz}$ will determine the remaining elements of $\mathbf{B}_n$.

7.5 SIXTEEN-POINT FORM

It is not always easy, practical, or even possible to provide the tangent and twist vector values required to define a bicubic Hermite patch. Therefore, we now develop another way of representing this patch. Remember, we must specify 48 degrees of freedom or algebraic coefficients. A 4×4 grid of 16 points supplies and satisfies the 48 conditions (see Figure 7.7). However, we must also choose values for u and w at these points. There are several ways to do this: by estimating the u, w values for points other than the corner points using a ratio of line segment lengths to get values between 0 and 1, or by assuming u, w values at the one-third points.

Expanding the algebraic form of the patch $\mathbf{p}(u,w) = \mathbf{UAW}^T$ produces

$$\mathbf{p}(u,w) = \mathbf{a}_{33}u^3w^3 + \mathbf{a}_{32}u^3w^2 + \cdots + \mathbf{a}_{00}$$

This is, of course, Equation (7.3). We can generate 16 of these equations, one for each of the 16 points. Let us use the u_i, w_j values in Figure 7.7. Thus,

$$\mathbf{p}(0,0) = \mathbf{a}_{33}(0)^3(0)^3 + \mathbf{a}_{32}(0)^3(0)^2 + \cdots + \mathbf{a}_{00}$$

$$\mathbf{p}\left(\frac{1}{3},0\right) = \mathbf{a}_{33}\left(\frac{1}{3}\right)^3(0)^3 + \mathbf{a}_{32}\left(\frac{1}{3}\right)^3(0)^2 + \cdots + \mathbf{a}_{00}$$

Figure 7.7 Sixteen-point form of a bicubic Hermite patch.

$$\vdots$$

$$\vdots$$

$$\mathbf{p}\left(\frac{2}{3},\frac{1}{3}\right) = \mathbf{a}_{33}\left(\frac{2}{3}\right)^3\left(\frac{1}{3}\right)^3 + \mathbf{a}_{32}\left(\frac{2}{3}\right)^3\left(\frac{1}{3}\right)^2 + \cdots + \mathbf{a}_{00} \tag{7.33}$$

$$\vdots$$

$$\vdots$$

$$\mathbf{p}(1,1) = \mathbf{a}_{33}(1)^3(1)^3 + \mathbf{a}_{32}(1)^3(1)^2 + \cdots + \mathbf{a}_{00}$$

In matrix form, this set of equations becomes

$$\mathbf{p}(u_i,w_j) = \mathbf{E}\mathbf{a}$$

or

$$\mathbf{E}^{-1}\mathbf{p}(u_i,w_j) = \mathbf{a} \tag{7.34}$$

The result is a system of simultaneous linear equations whose unknowns are the elements of $\mathbf{a}$. Furthermore, $\mathbf{E}$ is a 16×16 matrix of uw products, $\mathbf{a}$ is a 16×1 vector of the unknown algebraic coefficients of the $\mathbf{A}$ matrix, and $\mathbf{p}(u_i,w_j)$ is a 16×1 vector of the given data points. Notice that the solution to this equation involves three $\mathbf{A}$ matrices—one for each of the coordinates x, y, and z.

Next, we investigate the 16-point solution for the geometric form, rewriting $\mathbf{p}(u,w) = \mathbf{U}\mathbf{M}_F\mathbf{B}\mathbf{M}_F^T\mathbf{W}^T$ as

$$\mathbf{p}(u,w) = \mathbf{U}\mathbf{N}_F\mathbf{P}\mathbf{N}_F^T\mathbf{W}^T \tag{7.35}$$

We replace the $\mathbf{B}$ matrix by a matrix $\mathbf{P}$, where

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}(0,0) & \mathbf{p}\left(0,\frac{1}{3}\right) & \mathbf{p}\left(0,\frac{2}{3}\right) & \mathbf{p}(0,1) \\ \mathbf{p}\left(\frac{1}{3},0\right) & \mathbf{p}\left(\frac{1}{3},\frac{1}{3}\right) & \mathbf{p}\left(\frac{1}{3},\frac{2}{3}\right) & \mathbf{p}\left(\frac{1}{3},1\right) \\ \mathbf{p}\left(\frac{2}{3},0\right) & \mathbf{p}\left(\frac{2}{3},\frac{1}{3}\right) & \mathbf{p}\left(\frac{2}{3},\frac{2}{3}\right) & \mathbf{p}\left(\frac{2}{3},1\right) \\ \mathbf{p}(1,0) & \mathbf{p}\left(1,\frac{1}{3}\right) & \mathbf{p}\left(1,\frac{2}{3}\right) & \mathbf{p}(1,1) \end{bmatrix} \tag{7.36}$$

Performing the indicated algebra, we find $\mathbf{N}_F$ to be

$$\mathbf{N}_F = \begin{bmatrix} -\frac{9}{2} & \frac{27}{2} & -\frac{27}{2} & \frac{9}{2} \\ 9 & -\frac{45}{2} & 18 & -\frac{9}{2} \\ -\frac{11}{2} & 9 & -\frac{9}{2} & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{7.37}$$

This is, of course, the same matrix we encountered earlier while investigating the four-point form for curves. With the 16-point form, the input data **P** are, parametrically speaking, uniformly distributed over the patch.

The **B** and **P** matrices are related as follows:

$$\mathbf{B} = \mathbf{L}\mathbf{P}\mathbf{L}^T \tag{7.38}$$

where

$$\mathbf{L} = \mathbf{M}_F^{-1}\mathbf{N}_F = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{11}{2} & 9 & -\frac{9}{2} & 1 \\ -1 & \frac{9}{2} & -9 & \frac{11}{2} \end{bmatrix} \tag{7.39}$$

which we also encountered earlier.

This approach, requiring only point data for input, is better for the surface-fitting problem, but certain characteristics remain that make its use somewhat tedious. For example, when using this approach in representing analytically known surfaces (spherical, parabolic sheets, and so on), a significant deviation arises because of nonuniform spacing of the input points. Also, deciding on a patch distribution over a composite surface requires considerable care. Finally, since each patch is fit independently, continuity across the boundaries is neither guaranteed nor likely. For many modeling applications, these flaws are unimportant; still, they motivate us to make further improvements.

7.6 REPARAMETERIZATION OF A PATCH

The reparameterization of a patch proceeds in much the same way as for a curve. The simplest form of reparameterization is a reversal of the direction

of one or both of the parametric variables u and w. Again as with curves, we can do this easily, and it does not change the shape of the patch. Figure 7.8 shows the initial parameterization of a patch and the three possible reversals. We will confine our attention to the $\mathbf{B}$ matrix, because this is the only matrix whose elements change.

The matrix of geometric coefficients for the initial reversals will be $\mathbf{B}_1$, where

$$\mathbf{B}_1 = \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}_{00}^{w} & \mathbf{p}_{01}^{w} \\ \mathbf{p}_{10} & \mathbf{p}_{11} & \mathbf{p}_{10}^{w} & \mathbf{p}_{11}^{w} \\ \mathbf{p}_{00}^{u} & \mathbf{p}_{01}^{u} & \mathbf{p}_{00}^{uw} & \mathbf{p}_{01}^{uw} \\ \mathbf{p}_{10}^{u} & \mathbf{p}_{11}^{u} & \mathbf{p}_{10}^{uw} & \mathbf{p}_{11}^{uw} \end{bmatrix} \tag{7.40}$$

First, we reverse the parametric variable u, as in Figure 7.8b. This means that all curves of constant w will change their direction of parameterization. To do this, we first simply interchange rows 1 and 2 of the $\mathbf{B}_1$ matrix. Next, we interchange rows 3 and 4 and multiply each coefficient in these rows by -1.

Figure 7.8 The effect of reparameterization on the patch normal.

$$\mathbf{B}_2 = \begin{bmatrix} \mathbf{q}_{00} = \mathbf{p}_{10} & \mathbf{q}_{01} = \mathbf{p}_{11} & \mathbf{q}^{w}_{00} = \mathbf{p}^{w}_{10} & \mathbf{q}^{w}_{01} = \mathbf{p}^{w}_{11} \\ \mathbf{q}_{10} = \mathbf{p}_{00} & \mathbf{q}_{11} = \mathbf{p}_{01} & \mathbf{q}^{w}_{10} = \mathbf{p}^{w}_{00} & \mathbf{q}^{w}_{11} = \mathbf{p}^{w}_{01} \\ \mathbf{q}^{u}_{00} = -\mathbf{p}^{u}_{10} & \mathbf{q}^{u}_{01} = -\mathbf{p}^{u}_{11} & \mathbf{q}^{uw}_{00} = -\mathbf{p}^{uw}_{10} & \mathbf{q}^{uw}_{01} = -\mathbf{p}^{uw}_{11} \\ \mathbf{q}^{u}_{10} = -\mathbf{p}^{u}_{10} & \mathbf{q}^{u}_{11} = -\mathbf{p}^{u}_{01} & \mathbf{q}^{uw}_{10} = -\mathbf{p}^{uw}_{00} & \mathbf{q}^{uw}_{11} = -\mathbf{p}^{uw}_{01} \end{bmatrix} \tag{7.41}$$

We notice two things: First, the coefficients of the auxiliary curves are interchanged and multiplied by −1; second, the directions of the patch normals are reversed.

Next, starting with $\mathbf{B}_1$, we reverse the parametric variable w, as in Figure 7.8c. This operation will change the direction of parameterization of curves of constant u. This time, we interchange columns 1 and 2 of $\mathbf{B}_1$ and also interchange columns 3 and 4, multiplying each coefficient in these rows by −1; thus, we obtain $\mathbf{B}_3$:

$$\mathbf{B}_3 = \begin{bmatrix} \mathbf{r}_{00} = \mathbf{p}_{01} & \mathbf{r}_{01} = \mathbf{p}_{00} & \mathbf{r}^{w}_{00} = -\mathbf{p}^{w}_{01} & \mathbf{r}^{w}_{01} = -\mathbf{p}^{w}_{00} \\ \mathbf{r}_{10} = \mathbf{p}_{11} & \mathbf{r}_{11} = \mathbf{p}_{10} & \mathbf{r}^{w}_{10} = -\mathbf{p}^{w}_{11} & \mathbf{r}^{w}_{11} = -\mathbf{p}^{w}_{10} \\ \mathbf{r}^{u}_{00} = \mathbf{p}^{u}_{01} & \mathbf{r}^{u}_{01} = \mathbf{p}^{u}_{00} & \mathbf{r}^{uw}_{00} = -\mathbf{p}^{uw}_{01} & \mathbf{r}^{uw}_{01} = -\mathbf{p}^{uw}_{00} \\ \mathbf{r}^{u}_{10} = \mathbf{p}^{u}_{11} & \mathbf{r}^{u}_{11} = \mathbf{p}^{u}_{10} & \mathbf{r}^{uw}_{10} = -\mathbf{p}^{uw}_{11} & \mathbf{r}^{uw}_{11} = -\mathbf{p}^{uw}_{10} \end{bmatrix} \tag{7.42}$$

Again, we notice that the directions of the patch normals are reversed.

Finally, we reverse the direction of both parametric variables (see Figure 7.8d), starting with $\mathbf{B}_1$. We can reverse either u or w first. Let us choose u, resulting in Equation (7.41), or $\mathbf{B}_2$. Next, we reverse w by operating on $\mathbf{B}_2$, interchanging columns 1 and 2, and 3 and 4, multiplying each coefficient in 3 and 4 by −1. Thus, we obtain $\mathbf{B}_4$:

$$\mathbf{B}_4 = \begin{bmatrix} \mathbf{s}_{00} = \mathbf{p}_{11} & \mathbf{s}_{01} = \mathbf{p}_{10} & \mathbf{s}^{w}_{00} = -\mathbf{p}^{w}_{11} & \mathbf{s}^{w}_{01} = -\mathbf{p}^{w}_{10} \\ \mathbf{s}_{10} = \mathbf{p}_{01} & \mathbf{s}_{11} = \mathbf{p}_{00} & \mathbf{s}^{w}_{10} = -\mathbf{p}^{w}_{01} & \mathbf{s}^{w}_{11} = -\mathbf{p}^{w}_{00} \\ \mathbf{s}^{u}_{00} = -\mathbf{p}^{u}_{11} & \mathbf{s}^{u}_{01} = -\mathbf{p}^{u}_{10} & \mathbf{s}^{uw}_{00} = \mathbf{p}^{uw}_{11} & \mathbf{s}^{uw}_{01} = \mathbf{p}^{uw}_{10} \\ \mathbf{s}^{u}_{10} = -\mathbf{p}^{u}_{01} & \mathbf{s}^{u}_{11} = -\mathbf{p}^{u}_{00} & \mathbf{s}^{uw}_{10} = \mathbf{p}^{uw}_{01} & \mathbf{s}^{uw}_{11} = \mathbf{p}^{uw}_{00} \end{bmatrix} \tag{7.43}$$

In this case, the directions of the patch normals are not reversed.

For a more generalized reparameterization, we consider the bicubic Hermite patch in Figure 7.9a. Here, the patch is parameterized from u_i to u_j and from w_k to w_l. Figure 7.9b shows the same patch, except that the parametric variables range from t_i to t_j and from v_k to v_l. We let $\mathbf{B}_1$ denote the matrix of geometric coefficients in the first case and $\mathbf{B}_2$ the matrix in the second:

$$\mathbf{B}_1 = \begin{bmatrix} \mathbf{p}_{ik} & \mathbf{p}_{il} & \mathbf{p}^{w}_{ik} & \mathbf{p}^{w}_{il} \\ \mathbf{p}_{jk} & \mathbf{p}_{jl} & \mathbf{p}^{w}_{jk} & \mathbf{p}^{w}_{jl} \\ \mathbf{p}^{u}_{ik} & \mathbf{p}^{u}_{il} & \mathbf{p}^{uw}_{ik} & \mathbf{p}^{uw}_{il} \\ \mathbf{p}^{u}_{jk} & \mathbf{p}^{u}_{jl} & \mathbf{p}^{uw}_{jk} & \mathbf{p}^{uw}_{jl} \end{bmatrix} \tag{7.44}$$

Figure 7.9 General reparameterization of a patch.

$$\mathbf{B}_2 = \begin{bmatrix} \mathbf{q}_{ik} & \mathbf{q}_{il} & \mathbf{q}_{ik}^{v} & \mathbf{q}_{il}^{v} \\ \mathbf{q}_{jk} & \mathbf{q}_{jl} & \mathbf{q}_{jk}^{v} & \mathbf{q}_{jl}^{v} \\ \mathbf{q}_{ik}^{t} & \mathbf{q}_{il}^{t} & \mathbf{q}_{ik}^{tv} & \mathbf{q}_{il}^{tv} \\ \mathbf{q}_{jk}^{t} & \mathbf{q}_{jl}^{t} & \mathbf{q}_{jk}^{tv} & \mathbf{q}_{jl}^{tv} \end{bmatrix} \tag{7.45}$$

The relationships between the elements of $\mathbf{B}_1$ and $\mathbf{B}_2$ are the same as those for curves, and the same reasoning applies. For patches, as for curves, the corner points are related directly.

$$\begin{aligned} \mathbf{q}_{ik} &= \mathbf{p}_{ik} \\ \mathbf{q}_{jk} &= \mathbf{p}_{jk} \\ \mathbf{q}_{il} &= \mathbf{p}_{il} \\ \mathbf{q}_{jl} &= \mathbf{p}_{jl} \end{aligned} \tag{7.46}$$

The tangent vectors are a different matter. To preserve the bicubic form of the patch equations, we assume a linear relationship between u and t and between w and v. When we do this, we obtain results similar to those expressed by the tangent vectors in Equations (3.29). In particular, we find

that

$$\mathbf{q}^t = \frac{u_j - u_i}{t_j - t_i}\mathbf{p}^u \qquad \text{and} \qquad \mathbf{q}^v = \frac{w_l - w_k}{v_l - v_k}\mathbf{p}^w \tag{7.47}$$

For the cross-derivatives, we obtain

$$\mathbf{q}^{tv} = \frac{(u_j - u_i)(w_l - w_k)}{(t_j - t_i)(v_l - v_k)}\mathbf{p}^{uw} \tag{7.48}$$

Here is an example. If we reparameterize a patch from $u,w \in [0,1]$ to $u',w' \in [0, \frac{1}{3}]$, we obtain

$$\mathbf{B}' = \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} & 3\mathbf{p}^w_{00} & 3\mathbf{p}^w_{01} \\ \mathbf{p}_{10} & \mathbf{p}_{11} & 3\mathbf{p}^w_{10} & 3\mathbf{p}^w_{11} \\ 3\mathbf{p}^u_{00} & 3\mathbf{p}^u_{01} & 9\mathbf{p}^{uw}_{00} & 9\mathbf{p}^{uw}_{01} \\ 3\mathbf{p}^u_{10} & 3\mathbf{p}^u_{11} & 9\mathbf{p}^{uw}_{10} & 9\mathbf{p}^{uw}_{11} \end{bmatrix}$$

If u_j and u_i are successive pairs of integers, then $u_j - u_i = 1$; similarly for $t_j - t_i$, $w_l - w_k$, and $v_l - v_k$. As we noted for curves, this relationship is useful when dealing with arrays of contiguous patches. In Figure 7.10, individual patches in the array are readily identified by the array indexing scheme of successive integers.

Figure 7.10 Parameterization of a rectangular array of patches.

7.7 TRUNCATING AND SUBDIVIDING A PATCH

We can now apply reparameterizing techniques to create a new patch from an existing one, stating the problem as follows: Given a patch whose geometric coefficients are $\mathbf{B}_1$, find the matrix $\mathbf{B}_2$ of a new patch that is a subpatch of the given patch and bounded by curves of constant u_i, u_j, w_k, and w_l. From Figure 7.11, we find the corner points of the new patch

$$\begin{aligned}
\mathbf{q}_{00} &= \mathbf{p}(u_i,w_j) \\
\mathbf{q}_{10} &= \mathbf{p}(u_j,w_k) \\
\mathbf{q}_{01} &= \mathbf{p}(u_i,w_l) \\
\mathbf{q}_{11} &= \mathbf{p}(u_j,w_l)
\end{aligned} \tag{7.49}$$

where the $\mathbf{p}$ vectors and $\mathbf{q}$ vectors are elements of $\mathbf{B}_1$ and $\mathbf{B}_2$, respectively.

Using Equations (7.47), the tangent vectors are

$$\begin{aligned}
\mathbf{q}^t_{00} &= (u_j - u_i)\mathbf{p}^u(u_i,w_k) & \mathbf{q}^v_{00} &= (w_l - w_k)\mathbf{p}^w(u_i,w_k) \\
\mathbf{q}^t_{10} &= (u_j - u_i)\mathbf{p}^u(u_j,w_k) & \mathbf{q}^v_{10} &= (w_l - w_k)\mathbf{p}^w(u_k,w_k) \\
\mathbf{q}^t_{01} &= (u_j - u_i)\mathbf{p}^u(u_i,w_l) & \mathbf{q}^v_{01} &= (w_l - w_k)\mathbf{p}^w(u_i,w_l) \\
\mathbf{q}^t_{11} &= (u_j - u_i)\mathbf{p}^u(u_j,w_l) & \mathbf{q}^v_{11} &= (w_l - w_k)\mathbf{p}^w(u_j,w_l)
\end{aligned} \tag{7.50}$$

Remember that $t_1 - t_0 = 1$ and $v_1 - v_0 = 1$.

Figure 7.11 Subdividing a patch.

We obtain the cross-derivatives or twist vectors by evaluating $\mathbf{p}^{uw}(u_i,w_k)$, $\mathbf{p}^{uw}(u_j,w_k)$, $\mathbf{p}^{uw}(u_i,w_l)$, and $\mathbf{p}^{uw}(u_j,w_l)$ and using Equation (7.48):

$$\begin{aligned}
\mathbf{q}_{00}^{tv} &= (u_j - u_i)(w_l - w_k)\mathbf{p}^{uw}(u_i,w_k) \\
\mathbf{q}_{10}^{tv} &= (u_j - u_i)(w_l - w_k)\mathbf{p}^{uw}(u_j,w_k) \\
\mathbf{q}_{01}^{tv} &= (u_j - u_i)(w_l - w_k)\mathbf{p}^{uw}(u_i,w_l) \\
\mathbf{q}_{11}^{tv} &= (u_j - u_i)(w_l - w_k)\mathbf{p}^{uw}(u_j,w_l)
\end{aligned} \tag{7.51}$$

7.B COMPOSITE HERMITE SURFACES

A composite surface is a collection of individual patches joined to form a continuous, more extensive, and more complex surface. Figure 7.12 shows a composite surface defined by a 4×3 array of 12 patches. There are two basic questions to consider. First, what factors control the shape and, particularly, the continuity of the composite surface? Second, how do we construct such surfaces? The *bicubic patch* is the basis for the investigation that follows.

We are interested in the continuity of two patches $\mathbf{p}(u,w)$ and $\mathbf{q}(u,w)$ having a common boundary such that the curves $\mathbf{p}(1,w)$ and $\mathbf{q}(0,w)$ are identical (see Figure 7.13), and in the effect of the twist vectors on the continuity along this shared patch boundary. To ensure G^1 continuity across this boundary, the coefficients of the respective auxiliary curves must be scalar multiples of

Figure 7.12 A composite surface.

Figure 7.13 The effect of the twist vector on continuity.

each other. Given the coefficients of the auxiliary curve of patch $\mathbf{p}(u,w)$ along $\mathbf{p}(1,w)$ (namely $\mathbf{p}_{10}^{u}$, $\mathbf{p}_{10}^{uw}$, $\mathbf{p}_{11}^{u}$, and $\mathbf{p}_{11}^{uw}$), then the corresponding coefficients of $\mathbf{q}(u,w)$ are related as follows: $\mathbf{q}_{00}^{u} = a\mathbf{p}_{10}^{u}$, $\mathbf{q}_{00}^{uw} = a\mathbf{p}_{10}^{uw}$, $\mathbf{q}_{01}^{u} = a\mathbf{p}_{11}^{u}$, $\mathbf{q}_{01}^{uw} = a\mathbf{p}_{11}^{uw}$. When these conditions prevail, we find that $\mathbf{q}^{u}(0,w) = a\mathbf{p}^{u}(1,w)$ at any point along $\mathbf{p}(1,w)$; that is, along $\mathbf{q}(0,w)$.

The geometric coefficients required to produce G^1 continuity across a common boundary between two patches are shown in Figure 7.14. This for-

Figure 7.14 The geometric coefficients affected by the requirement for G¹ continuity across a common boundary between two patches.

mat was introduced by Peters (1974). We conclude that adjacent patches have G^1 continuity only if common position-determining rows (or columns) of their **B** matrices are identical and if common auxiliary curve-determining rows (or columns) are multiples of each other. Thus, two patches joined to exhibit G^1 continuity have a total of 73 degrees of freedom, whereas two disjoint patches have a total of 96.

At a corner of any patch, four vectors contribute to its shape through the operation of the basis functions on them. Because the basis functions are invariant, all the shape-determining power lies in these four vectors, $\mathbf{p}(u,w)$, $\mathbf{p}^u(u,w)$, $\mathbf{p}^w(u,w)$, and $\mathbf{p}^{uw}(u,w)$. Four patches surround and share an interior corner point of a composite surface. In many modeling situations, it will help us to consider these mesh points or corner points as the centers or foci of the shaping parameters.

Figure 7.15 shows conditions at an arbitrary internal point $\mathbf{p}_{ij}$ of a composite surface where four patches meet. Each quadrant surrounding this point contains a list of the vectors applied to the patch in that quadrant. Clearly, all four patches share the common point $\mathbf{p}_{ij}$. Consider next the common boundary curve between patch I and patch II. The tangent vectors in the w direction must be identical to ensure that the respective boundary curves do match. The tangent vectors in the u direction must only be in the same direction, differing only by a scalar multiple k to ensure G^1 continuity

Figure 7.15 Continuity characteristics of a composite surface.

along the curve through the point. And since the cross-derivatives affect the slope along this boundary, as we have seen, they, too, must differ only by a scalar multiple.

By continuing this analysis for the other patch boundary combinations we generate the relationships between vectors, as the figure shows. We conclude that at an arbitrary point $\mathbf{p}_{ij}$, only 14 degrees of freedom are available to control the local shape of the composite surface (that is, four vectors and two scalars) if we are to preserve G^1 continuity.

Notice that the k and l scale factors are not strictly unique to each point. They do not change along a line of constant i or j (see Figure 7.16), thus preserving a constant scalar-multiple relationship between adjacent auxiliary

Figure 7.16 *Continuity-driven propagation of scale factors.*

curves along a curve of constant u_i or w_j. Also notice that when u and w take on integer values at intervals corresponding to the i,j indexing scheme, reparameterization is not necessary, because, for example, $u_{i+1} - u_i = 1$. In the figure, only the k and l scale factors are shown in three of the four quadrants at each i,j node. They multiply the corresponding vectors in the first quadrant of the node.

The distribution of scale factors for a 4×5 array of patches with seven distinct scale factors is shown in Figure 7.17. Although the arrows at each end of the band of constant scale factor are bidirectional, we must select a single consistent direction for each band to which the scalar relationship applies. Once we select a convention for associating the data at the points with the adjacent patches, we can then easily develop a scheme to fill in the **B** matrix for any specific patch. If we identify a patch by the lowest pair of indices of the four sets defining the patch corners, then we readily fill the **B** matrix of patch i,j as follows:

$$[\mathbf{B}]_{ij} = \begin{bmatrix} \mathbf{p}_{ij} & \mathbf{p}_{i,j+1} & \mathbf{p}^{w}_{ij} & l_{j+1}\mathbf{p}^{w}_{i,j+1} \\ \mathbf{p}_{i+1,j} & \mathbf{p}_{i+1,j+1} & \mathbf{p}^{w}_{i+1,j} & l_{j+1}\mathbf{p}^{w}_{i+1,j+1} \\ \mathbf{p}^{u}_{ij} & \mathbf{p}^{u}_{i,j+1} & \mathbf{p}^{uw}_{ij} & l_{j+1}\mathbf{p}^{uw}_{i,j+1} \\ k_{i+1}\mathbf{p}^{u}_{i+1,j} & k_{i+1}\mathbf{p}^{u}_{i+1,j+1} & k_{i+1}\mathbf{p}^{uw}_{i+1,j} & k_{i+1}l_{j+1}\mathbf{p}^{uw}_{i+1,j+1} \end{bmatrix} \tag{7.52}$$

Here the indices outside the brackets on **B** identify the patch itself and not an element of the matrix.

Now let us consider how to construct a composite surface. Applying parametric cubic forms to modeling problems once required the user to supply tangent and twist vectors as well as point coordinates at the patch corner points. Soon, there were schemes to define patches without having to specify tangent vectors. Some schemes replaced the required input tangent vectors with extra point coordinates other than the corner points. This approach was useful for creating individual patches, but most surfaces are sufficiently complex to require a number of patches joined together. Composite surfaces created this way usually lacked G^1 continuity, unsatisfactory for most applications.

Parametric spline interpolation came into vogue to generate a rectangular network of patches from a corresponding network of points, using numerical differentiation of the spline interpolation to calculate the derivatives required for the complete set of patch coefficients. *Cardinal splines*, a more elegant approach, were introduced by Gordon (1969).

Here is a summary of Timmer's (1976) application of Gordon's approach to defining a rectangular network of bicubic Hermite patches. We consider two families of intersecting curves $\mathbf{q}_i$ and $\mathbf{r}_j$ with $i \in [1:m]$ and $j \in [1:n]$,

Figure 7.17 Distribution of scale factors.

which combine to form the wireframe surface shown in Figure 7.18. Note that there are $m \times n$ intersection points.

We express the $\mathbf{q}_i$ family of m curves in terms of the parameter s_i and the $\mathbf{r}_j$ family of n curves in terms of the parameter t. Then, for a typical curve,

$$\mathbf{p}(s) = \mathbf{q}_i(s) \qquad s \in [0, S_i] \qquad \text{and} \qquad \mathbf{p}(t) = \mathbf{r}_j(t) \qquad t \in [0, T_j] \tag{7.53}$$

where $\mathbf{p}$ is the position vector to a point on the indicated curve.

The range of the parameter is not necessarily identical for all of the $\mathbf{q}_i$ or $\mathbf{r}_j$; therefore, we define a second set of parameters u, w such that

$$u_j = \frac{1}{m} \sum_{i=1}^{m} \frac{S_{ij}}{S_{in}} \qquad \text{and} \qquad w_j = \frac{1}{n} \sum_{j=1}^{n} \frac{t_{ij}}{t_{mj}} \tag{7.54}$$

The double subscript on s and t denotes their value at the intersection node indicated. The parameters u and w are normalized to map the entire surface

Figure 7.18 Rectangular network of intersecting curves.

into the unit square in u,w parameter space. Each s curve corresponds to a single value of w and each t curve corresponds to a single value of u. This procedure has the effect of defining n functions $t = T_j(w)$, $j \in [1:n]$ and m functions $s = S_i(u)$, $i \in [1:m]$. In practice, we express these as parametric spline-interpolating functions. As a result, we reparameterize the two families of curves to map them into the unit square, so that

$$\begin{aligned} \mathbf{p}(u) &= \mathbf{q}_i[S_i(u)] & & \mathbf{p}(w) = \mathbf{r}_j[T_j(w)] \\ i &\in [1:m] & \text{and} \quad & j \in [1:n] \\ u &\in [0,1] & & w \in [0,1] \end{aligned} \tag{7.55}$$

Next we consider the actual interpolation problem. We can treat this network interpolation as a combination of two problems, interpolating the $\mathbf{q}_i$ and $\mathbf{r}_j$ separately. For example, consider the m curves of $\mathbf{q}_i(u)$. Each of these curves corresponds to a distinct value of the parameter w; therefore, it is natural to interpolate these curves separately.

$$\mathbf{p}(u,w) = \sum_{i=1}^{m} F_i(w)\ \mathbf{q}_i(u) \tag{7.56}$$

The $F_i(w)$ are basis or interpolating functions. These have the obvious properties:

$$\begin{aligned} F_i(w_j) &= \delta_{ij} \\ &= 0 \quad \text{if} \quad i \neq j \\ &= 1 \quad \text{if} \quad i = j \end{aligned} \tag{7.57}$$

We are free to choose any functions that satisfy Equation (7.57) to act as basis functions for the indicated interpolation.

Let us consider the class of cubic spline-interpolating functions for the F_i, often referred to as *cardinal splines*. Figure 7.19 shows several of the F functions. Notice that the maxima damp rapidly as the curve moves away from the defining node. This is significant, because it is often possible to truncate Equation (7.56) using the F_i in only the immediate neighborhood of w_j.

Specifying values for the dependent variable at each of the interpolating mesh points results in two unspecified conditions. Usually these conditions are first- or second-derivative constraints applied at each end of the mesh.

For the splines defining the $\mathbf{q}_i(s)$, the constraints reflect the obvious slope and curvature considerations for a space curve. Selecting endpoint constraints for the other two spline fits is not so obvious. In the absence of any other constraints, we assume that a constant second derivative in each of the two end segments will be imposed on both $S_i(u)$ and $F_i(w)$. Clearly, we could have performed the interpolation

$$\mathbf{p}(u,w) = \sum_{j=1}^{n} \tilde{F}_j(u)\, \mathbf{r}_j(w) \tag{7.58}$$

The functions $\tilde{F}_j(u)$ assume the same role as the $F_i(w)$ in Equation (7.56). For our example, where we specify both $\mathbf{q}_i(u)$ and $\mathbf{r}_j(w)$, we achieve the optimum interpolation by using

$$\mathbf{p}(u,w) = \sum_{j=1}^{n} F_j(u)\, \mathbf{r}_j(w) + \sum_{i=1}^{m} F_i(w)\, \mathbf{q}_i(u) - \sum_{i=1}^{m}\sum_{j=1}^{n} \tilde{F}_i(w)\, F_j(u)\, \mathbf{p}_{ij} \tag{7.59}$$

where $\mathbf{p}_{ij}$ are the values of $\mathbf{p}$ at the i,j mesh points.

Figure 7.19 Cardinal spline-interpolating function.

It turns out that Equation (7.59) is an elegant but computationally inefficient way to interpolate two families of curves forming a rectangular mesh. However, we can use the equation to determine the geometric coefficients for a network of bicubic Hermite patches. Once we select the patch corner points, we can use the equation to determine values of $\mathbf{p}^{u}_{iow}$, $\mathbf{p}^{w}_{iow}$, and $\mathbf{p}^{uw}_{iow}$.

Finally, if $\mathbf{q}_i$ and $\mathbf{r}_j$ have G^0, G^1, and G^2 continuity, then Equation (7.59) guarantees this same continuity throughout the unit square, $u,w \in [0,1]$. If the unit square is subdivided into a rectangular array of bicubic Hermite patches, continuity is preserved, provided the patches are not arbitrarily reparameterized. For example, if we reparameterize each patch to the unit square, we create a discontinuous relationship between the original parameters and the new parameters, and we lose the G^2 continuity of Equation (7.59).

What are the practical implications of this surface-generating interpolation scheme? First, it is unlikely that the two families of curves will be completely defined at the outset of the modeling problem. It is likely that the initial data will be in the form of discrete points distributed in some semi-regular way over the prospective surface, perhaps including slope and curvature data. If this is the case, then the parametric representations of $\mathbf{q}_i(u)$ and $\mathbf{r}_j(w)$ remain to be defined. There are different approaches we can take, and the nature of the input data will determine the strategy.

Let us consider the case where the point data are in the form of an $m \times n$ array uniformly distributed over the surface, without slope or curvature preconditions. We may now decide to represent $\mathbf{q}_i(u)$ and $\mathbf{r}_j(w)$ as follows:

$$\mathbf{q}_i(u) = \sum_{j=1}^{n} \tilde{F}_j(u)\mathbf{p}_{ij} \qquad \text{and} \qquad \mathbf{r}_j(w) = \sum_{i=1}^{m} F_i(w)\mathbf{p}_{ij} \tag{7.60}$$

Substituting these equations into Equation (7.59) produces a simpler expression:

$$\mathbf{p}(u,w) = \sum_{i=1}^{m} \sum_{j=1}^{n} F_i(w)\tilde{F}_j(u)\mathbf{p}_{ij} \tag{7.61}$$

If we include slope or curvature constraints, this expression becomes more complex.

Equation (7.59) or (7.61) can be interpolated over the entire surface and mapped into the unit square in u,w parameter space, similar to the process we used for the bicubic Hermite patch. The obvious difference is that, for the immediate example, a single equation does not cover the entire unit square. However, each patch element formed by the two families of curves is the equivalent of a bicubic patch in the sense that, within any quadrilateral patch element, the interpolation may be expressed as the familiar

$$\mathbf{p}(u,w) = \sum_{i=0}^{3} \sum_{j=0}^{3} \mathbf{a}_{ij} u^i w^j \tag{7.62}$$

Any subelement can be parameterized to the unit square without danger of altering the composite surface. This means that interpolation by Equation (7.59) or (7.61) is equivalent to, and may be replaced by, a network of $(m-1) \times (n-1)$ bicubic Hermite patches without affecting the guaranteed continuity. A disadvantage of this approach is that it can lead to a large number of patches. A practical alternative would allow us to model the same surface using fewer patches and without changing the original input point data. Figure 7.20 illustrates a scheme to selectively reduce the number of patches used in a composite surface model. What was potentially an 11×15 array of patches is replaced by a 3×3 array.

Reducing the number of patches, however, could cause a loss of G^2 continuity. This loss becomes apparent when we consider a single curve; for example, the upper boundary curve in the figure. In the second segment, we use a single cubic Hermite curve to approximate seven curves. The single segment will match the original endpoints and tangents; however, the curvature may deviate somewhat. Freeing the curvature at the endpoints leads to the discontinuity. Because the original composite curve had continuous curvature,

Figure 7.20 Patch array on a composite surface.

the new single-curve approximation should be close and the resulting discontinuities barely discernible. Note that in the course of this reduction in the number of patches, the final network is still rectangular.

A final simplification is possible. In the preceding example, input data points form a rectangular mesh—each row has the same number of points in it. Now consider the problem of a surface with transitions from a complex cross section to a simple one (see Figure 7.21). The number of points we need to define the simple section is less than for the complex one. To preserve a rectangular mesh of initial data points, we would have to specify more data points for the simple sections than these shapes would otherwise warrant. We conclude that the rectangularity constraint requires us to supply too much data, and thus it is inefficient. To avoid this, we can use the following strategy.

We assume that in most cases we prefer to supply data for a series of cross-section curves rather than for a rectangular grid on a surface. We let n_0 denote the number of data points required to define the most complex cross-section curve and m the number of cross sections. A rectangular mesh requires $m \times n_0$ mesh points; however, instead of a rectangular mesh format, we now allow the model input to consist of the exact number of points necessary for each cross section individually. Thus, we could describe a linear cross-section curve using only two points. Using this approach, the total number of points required to describe the surface is

Figure 7.21 Transition from a complex to a simple cross section.

$$N = \sum_{i=1}^{m} n_i \tag{7.63}$$

Obviously, $N \leq mn_0$. Using this reduced input data, we interpolate each cross-section curve, using a conventional spline or other form of representation. Next, we compute a set of n_0 points for each curve, and we continue as for a rectangular mesh.

Insofar as composite surfaces are concerned, we have barely scratched the surface (pardon the pun). The literature abounds with a great variety of representational mathematics and construction techniques for most modeling situations.

7.9 SPECIAL HERMITE PATCHES

There are several ways to construct a plane patch, and we will investigate three of them here. Figure 7.22 shows the simplest construction, which we interpret as

$$\mathbf{p}(u,w) = \mathbf{p}_{00} + u\mathbf{r} + w\mathbf{s} \qquad u,w \in [0,1] \tag{7.64}$$

This equation defines a plane patch through the point $\mathbf{p}_{00}$ and parallel to the vectors r and s. It is a special case of the bicubic Hermite patch. We set all the

Figure 7.22 Vector equation of a plane.

algebraic coefficients in Equation (7.3) equal to zero except $\mathbf{a}_{00}$, $\mathbf{a}_{10}$, and $\mathbf{a}_{01}$, resulting in

$$\mathbf{p}(u,w) = \mathbf{a}_{00} + u\mathbf{a}_{10} + w\mathbf{a}_{01} \tag{7.65}$$

Equation (7.65) is identical to (7.64), and we find $\mathbf{a}_{00} = \mathbf{p}_{00}$, $\mathbf{a}_{10} = \mathbf{r}$, and $\mathbf{a}_{01} = \mathbf{s}$. We can determine the geometric coefficients for this construction as well:

$$\begin{aligned}
&\mathbf{p}_{00} = \mathbf{p}_{00} && \mathbf{p}_{01} = \mathbf{p}_{00} + \mathbf{s} \\
&\mathbf{p}_{10} = \mathbf{p}_{00} + \mathbf{r} && \mathbf{p}_{11} = \mathbf{p}_{00} + \mathbf{r} + \mathbf{s} \\
&\mathbf{p}^{u}_{00} = \mathbf{r} && \mathbf{p}^{u}_{01} = \mathbf{r} \\
&\mathbf{p}^{u}_{10} = \mathbf{r} && \mathbf{p}^{u}_{11} = \mathbf{r} \\
&\mathbf{p}^{w}_{00} = \mathbf{s} && \mathbf{p}^{w}_{01} = \mathbf{s} \\
&\mathbf{p}^{w}_{10} = \mathbf{s} && \mathbf{p}^{w}_{11} = \mathbf{s}
\end{aligned} \tag{7.66}$$

where the twist vectors are

$$\begin{aligned}
&\mathbf{p}^{uw}_{00} = 0 && \mathbf{p}^{uw}_{01} = 0 \\
&\mathbf{p}^{uw}_{10} = 0 && \mathbf{p}^{uw}_{11} = 0
\end{aligned} \tag{7.67}$$

We assemble the **B** matrix directly from these equations:

$$\mathbf{B} = \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{00} + \mathbf{s} & \mathbf{s} & \mathbf{s} \\ \mathbf{p}_{00} + \mathbf{r} & \mathbf{p}_{00} + \mathbf{r} + \mathbf{s} & \mathbf{s} & \mathbf{s} \\ \mathbf{r} & \mathbf{r} & 0 & 0 \\ \mathbf{r} & \mathbf{r} & 0 & 0 \end{bmatrix} \tag{7.68}$$

Figure 7.23 illustrates a second method of construction. Here we start with three points, $\mathbf{p}_{00}$, $\mathbf{p}_{10}$, $\mathbf{p}_{01}$. From elementary geometry we know that three points define a plane. We construct all the other geometric coefficients from these points in such a way as to guarantee that the patch lies in this plane. Using the elements from the figures, we easily assemble the **B** matrix:

$$\mathbf{B} = \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}_{01} - \mathbf{p}_{00} & \mathbf{p}_{01} - \mathbf{p}_{00} \\ \mathbf{p}_{10} & \mathbf{p}_{10} + \mathbf{p}_{01} - \mathbf{p}_{00} & \mathbf{p}_{01} - \mathbf{p}_{00} & \mathbf{p}_{01} - \mathbf{p}_{00} \\ \mathbf{p}_{10} - \mathbf{p}_{00} & \mathbf{p}_{10} - \mathbf{p}_{00} & 0 & 0 \\ \mathbf{p}_{10} - \mathbf{p}_{00} & \mathbf{p}_{10} - \mathbf{p}_{00} & 0 & 0 \end{bmatrix} \tag{7.69}$$

The final plane patch construction is considerably more generalized than the first two. It permits us to construct plane figures with curved boundaries. As we see in Figure 7.24, we do not restrict the boundary curves to being straight lines. We start with the point $\mathbf{p}_{00}$ and the two tangent vectors $\mathbf{p}^{u}_{00}$, $\mathbf{p}^{w}_{00}$.

Figure 7.23 Special bicubic Hermite patch form of a plane.

Figure 7.24 General bicubic Hermite patch form of a plane.

We use these tangent vectors as a basis in the plane of the patch and define the other geometric coefficients as linear combinations of them. We can define the cross-derivatives or twist vectors to be zero, or we can define them by some relationship, such as $\mathbf{p}_{10}^{uw} = (c_{11} - c_{10})\mathbf{p}_{00}^{u} + (d_{11} - d_{10})\mathbf{p}_{00}^{w}$. Once again, we can assemble the elements of a **B** matrix from the figure:

$$\mathbf{B} = \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{00} + a_{01}\mathbf{p}_{00}^{u} + b_{01}\mathbf{p}_{00}^{w} & \mathbf{p}_{00}^{w} & e_{01}\mathbf{p}_{00}^{u} + f_{01}\mathbf{p}_{00}^{w} \\ \mathbf{p}_{00} + a_{10}\mathbf{p}_{00}^{u} + b_{10}\mathbf{p}_{00}^{w} & \mathbf{p}_{00} + a_{11}\mathbf{p}_{00}^{u} + b_{11}\mathbf{p}_{00}^{w} & e_{10}\mathbf{p}_{00}^{u} + f_{10}\mathbf{p}_{00}^{w} & e_{11}\mathbf{p}_{00}^{u} + f_{11}\mathbf{p}_{00}^{w} \\ \mathbf{p}_{00}^{u} & c_{01}\mathbf{p}_{00}^{u} + d_{01}\mathbf{p}_{00}^{w} & 0 & 0 \\ c_{10}\mathbf{p}_{00}^{u} + d_{10}\mathbf{p}_{00}^{w} & c_{11}\mathbf{p}_{00}^{u} + d_{11}\mathbf{p}_{00}^{w} & 0 & 0 \end{bmatrix} \tag{7.70}$$

Cylindrical Surface

A *cylinder* is a surface generated by a straight line as it moves parallel to itself along a curve. Here, we will construct a cylindrical bicubic Hermite patch. In Figure 7.25, we see a curve whose geometric coefficients are $[\mathbf{p}_0 \quad \mathbf{p}_1 \quad \mathbf{p}_0^{u} \quad \mathbf{p}_1^{u}]^T$, and a straight line defined from $\mathbf{p}_0$ to $\mathbf{p}_2$. The geometric coefficients of the straight line in the cubic Hermite format are $[\mathbf{p}_0 \quad \mathbf{p}_2 \quad (\mathbf{p}_2 - \mathbf{p}_0) \quad (\mathbf{p}_2 - \mathbf{p}_0)]^T$. Using these elements, we readily find the **B** matrix for a patch defined by the given curve and line:

Figure 7.25 Cylindrical surface.

$$\mathbf{B} = \begin{bmatrix} \mathbf{p}_0 & \mathbf{p}_2 & \mathbf{p}_2 - \mathbf{p}_0 & \mathbf{p}_2 - \mathbf{p}_0 \\ \mathbf{p}_1 & \mathbf{p}_1 + \mathbf{p}_2 - \mathbf{p}_0 & \mathbf{p}_2 - \mathbf{p}_0 & \mathbf{p}_2 - \mathbf{p}_0 \\ \mathbf{p}_0^u & \mathbf{p}_0^u & 0 & 0 \\ \mathbf{p}_1^u & \mathbf{p}_1^u & 0 & 0 \end{bmatrix} \tag{7.71}$$

A more general expression is

$$\mathbf{p}(u, w) = \mathbf{p}(u) + w\mathbf{r} \tag{7.72}$$

where $\mathbf{p}(u)$ is any space curve and $\mathbf{r}$ is the direction vector of the straight-line rulings (see Figure 7.26). We do not restrict the function $\mathbf{p}(u)$ to the cubic polynomial form. A cylindrical surface is a special case of a ruled surface.

Ruled Surface

We now know that the locus of a moving point with one degree of freedom is a *curve*. We also know that the locus of a straight line moving with one degree of freedom is a *surface*. This is a special type of surface, called a *ruled surface*, defined such that through each point of it passes at least one straight line lying entirely in the surface. The simplest of all ruled surfaces are the *plane*, the *cone*, and the *cylinder*; all special cases because each is governed by a set of constraints that make it unique. The *hyperbolic paraboloid* and the *hyperboloid of one sheet* are also special ruled surfaces. Each of these surfaces has two mathematically distinct families of straight lines, called *rulings*.

Figure 7.26 Another cylindrical surface.

Every developable surface is ruled, but every ruled surface is not developable. A developable surface is one that can be unrolled onto a plane without stretching or distorting it; cones and cylinders are good examples. However, the hyperbolic paraboloid is not developable. Later, we will examine the special testable characteristics of developable surfaces. Figure 7.27 shows the geometry of a bicubic ruled surface. Given two cubic Hermite curves $\mathbf{p}(u)$ and $\mathbf{q}(u)$, we construct a ruled surface by joining with a straight line each point on $\mathbf{p}(u)$ to a point on $\mathbf{q}(u)$ having an equivalent u value. We fill in the **B**-matrix elements accordingly:

$$\mathbf{B} = \begin{bmatrix} \mathbf{p}_0 & \mathbf{q}_0 & \mathbf{q}_0 - \mathbf{p}_0 & \mathbf{q}_0 - \mathbf{p}_0 \\ \mathbf{p}_1 & \mathbf{q}_1 & \mathbf{q}_1 - \mathbf{p}_1 & \mathbf{q}_1 - \mathbf{p}_1 \\ \mathbf{p}_0^u & \mathbf{q}_0^u & 0 & 0 \\ \mathbf{p}_1^u & \mathbf{q}_1^u & 0 & 0 \end{bmatrix} \tag{7.73}$$

Each parametric curve of constant u is a straight line. Thus, a general ruled surface is a linear interpolation between two curves.

The *hyperbolic paraboloid* is a special case, because both families of parametric curves $\mathbf{p}(u,w_i)$ and $\mathbf{p}(u_i,w)$ are straight lines. Given the four corner points, we define the eight tangent vectors as straight lines and the four twist vectors as equal to zero (see Figure 7.28). From this, the **B** matrix follows directly:

Figure 7.27 Ruled surface.

Figure 7.28 Hyperbolic paraboloid.

$$\mathbf{B} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_3 & \mathbf{p}_3 - \mathbf{p}_1 & \mathbf{p}_3 - \mathbf{p}_1 \\ \mathbf{p}_2 & \mathbf{p}_4 & \mathbf{p}_4 - \mathbf{p}_2 & \mathbf{p}_4 - \mathbf{p}_2 \\ \mathbf{p}_2 - \mathbf{p}_1 & \mathbf{p}_4 - \mathbf{p}_3 & 0 & 0 \\ \mathbf{p}_2 - \mathbf{p}_1 & \mathbf{p}_4 - \mathbf{p}_3 & 0 & 0 \end{bmatrix} \tag{7.74}$$

Can we improve the representation if we use nonzero twist vectors? How and why? (Hint: Try $\mathbf{p}_{00}^{uw} = (\mathbf{p}_4 - \mathbf{p}_3) - (\mathbf{p}_2 - \mathbf{p}_1)$, and so on.)

Here are two more ways of representing a ruled surface with a parametric equation. The first is

$$\mathbf{p}(u,w) = \mathbf{g}(u) + w\mathbf{d}(u) \tag{7.75}$$

where $\mathbf{g}(u)$ is a curve and $\mathbf{d}(u)$ is the direction vector of a straight line at point u on the curve. The parameter w gives the distance ratio of the point $\mathbf{p}(u,w)$ from $\mathbf{g}(u)$ (see Figure 7.29). If $u,w \in [0,1]$, then $|\mathbf{d}(u)|$ equals the length of the straight line when $w = 1$. The vector functions $\mathbf{g}(u)$ and $\mathbf{d}(u)$ in this form are not necessarily restricted to the cubic polynomial form.

An alternative expression based on the straight-line rulings joining corresponding points on two space curves $\mathbf{d}(u)$ and $\mathbf{h}(u)$ is

$$\begin{aligned}\mathbf{p}(u,w) &= (1-w)\mathbf{g}(u) + w\mathbf{h}(u) \\ &= \mathbf{g}(u) + w[\mathbf{h}(u) - \mathbf{g}(u)]\end{aligned} \tag{7.76}$$

where $u,w \in [0,1]$. Again, in this form the vector functions $\mathbf{g}(u)$ and $\mathbf{h}(u)$ are not necessarily restricted to cubic polynomials. Figure 7.30 illustrates the vector construction of a ruled surface via Equation (7.76).

Degenerate Patches

Degenerate patches immediately attract our attention to their unusual shapes. Almost invariably, these strange shapes are the result of an array of geometric coefficients with equally unusual characteristics, which may include coincident control points or other unconventional values and relationships. Degenerate patches are possible through intention or accident. For now, the best source of these patches is our imagination. Any bizarre form we can conjure up is certain to appear unbidden and at an awkward moment in any sophisticated modeling system. The conditions we are about to explore can be created with any of the forms we have discussed thus far, as well as those yet to be discussed, including the Bézier and B-Spline surfaces. Some of these forms have obvious idiosyncrasies—superimposed vertices or, perhaps, jumbled vertices of their characteristic polyhedra.

Figure 7.29 Ruled surface.

Figure 7.30 Another ruled surface.

There are several ways to categorize degenerate patches; for example, those exhibiting degenerate boundary curves, those exhibiting superimposed corners, and those with strictly internal degeneracies. We can demonstrate them all using a single bicubic Hermite patch. The simplest such patch is a point where $\mathbf{p}_{00} = \mathbf{p}_{10} = \mathbf{p}_{01} = \mathbf{p}_{11}$ and all other coefficients are identically equal to zero. A patch degenerates to a straight line when $\mathbf{p}^u_{00} = \mathbf{p}^u_{10} = \mathbf{p}_{10} - \mathbf{p}_{00}$ and all other coefficients are zero.

The degenerate patches in Figure 7.31 have easily identifiable characteristics in their **B** matrices. For the three-sided patch in Figure 7.31a, if we assume that $\mathbf{p}_{00} = \mathbf{p}_{10}$, then the **B** matrix is

$$\mathbf{B}_{(a)} = \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}^w_{00} & \mathbf{p}^w_{01} \\ \mathbf{p}_{00} & \mathbf{p}_{11} & \mathbf{p}^w_{10} & \mathbf{p}^w_{11} \\ 0 & \mathbf{p}^u_{01} & \mathbf{p}^{uw}_{00} & \mathbf{p}^{uw}_{01} \\ 0 & \mathbf{p}^u_{11} & \mathbf{p}^{uw}_{10} & \mathbf{p}^{uw}_{11} \end{bmatrix} \tag{7.77}$$

A different kind of three-sided patch is shown in Figure 7.31b. Instead of a zero-length fourth side, one of the sides does double duty. Its **B** matrix might be as follows:

$$\mathbf{B}_{(b)} = \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}_{00}^{w} & \mathbf{p}_{01}^{w} \\ \mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}_{00}^{w} & \mathbf{p}_{01}^{w} \\ \mathbf{p}_{00}^{u} & \mathbf{p}_{01}^{u} & \mathbf{p}_{00}^{uw} & \mathbf{p}_{01}^{uw} \\ \mathbf{p}_{10}^{u} & \mathbf{p}_{11}^{u} & \mathbf{p}_{10}^{uw} & \mathbf{p}_{11}^{uw} \end{bmatrix} \tag{7.78}$$

When points diagonally opposite each other are coincident, the result may be as shown in Figure 7.31c, and the **B** matrix would be as follows:

$$\mathbf{B}_{(c)} = \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{10} & \mathbf{p}_{00}^{w} & \mathbf{p}_{01}^{w} \\ \mathbf{p}_{10} & \mathbf{p}_{11} & \mathbf{p}_{10}^{w} & \mathbf{p}_{11}^{w} \\ \mathbf{p}_{00}^{u} & \mathbf{p}_{01}^{u} & \mathbf{p}_{00}^{uw} & \mathbf{p}_{01}^{uw} \\ \mathbf{p}_{10}^{u} & \mathbf{p}_{11}^{u} & \mathbf{p}_{10}^{uw} & \mathbf{p}_{11}^{uw} \end{bmatrix} \tag{7.79}$$

Figure 7.31 Degenerate patches.

Finally, in Figure 7.31d, an interesting shape arises by using the results from Figure 7.31c and by joining the remaining diagonally opposing pair of points, with the following **B** matrix:

$$\mathbf{B}_{(d)} = \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{10} & \mathbf{p}_{00}^{w} & \mathbf{p}_{01}^{w} \\ \mathbf{p}_{10} & \mathbf{p}_{00} & \mathbf{p}_{10}^{w} & \mathbf{p}_{11}^{w} \\ \mathbf{p}_{00}^{u} & \mathbf{p}_{01}^{u} & \mathbf{p}_{00}^{uw} & \mathbf{p}_{01}^{uw} \\ \mathbf{p}_{10}^{u} & \mathbf{p}_{11}^{u} & \mathbf{p}_{10}^{uw} & \mathbf{p}_{11}^{uw} \end{bmatrix} \tag{7.80}$$

Perhaps the terms *degenerate* and *pathological* are too pejorative, at least for the examples we have just explored. Although some may have little practical value, they clearly have a certain aesthetic appeal.

7.10 BLEND SURFACES

Blend surfaces are secondary surfaces forming a locally smooth G^1 (at least) transition between two or more primary surfaces, which may or may not intersect. In the most general case, the curves of tangency between the blend and primary surfaces do not correspond to isoparametric curves or patch boundaries. A more rigorous definition seems to be lacking in the literature, in part because of widely different approaches to this subject. However, Rossignac and Requicha (1984) do identify four major categories of blends: blending surfaces generated by strong functional constraints, aesthetic blends, fairings, and rounds and fillets. Vida, Martin, and Varady (1994) take a somewhat different approach and classify methods of constructing parametric blends as rolling-ball based, spine based, trimline based, or polyhedral, among others.

The first example we will consider is a simple blend between two disjoint bicubic Hermite patches, where curves of tangency correspond to patch boundaries. Then we study a way to blend an arbitrary patch to the boundaries of another patch, where the first patch is interior to the second patch. Finally, we look at more general blend problems.

Blend between Two Disjoint Patches

Given two disjoint bicubic Hermite patches $\mathbf{p}(u,w)$ and $\mathbf{q}(u,w)$, we can find a third patch $\mathbf{r}(u,w)$ that blends smoothly between them, creating a G^1 continuous composite surface. We assume the common boundaries are such that $\mathbf{r}(0,w) = \mathbf{p}(1,w)$ and $\mathbf{r}(1,w) = \mathbf{q}(0,w)$ (see Figure 7.32). The modeling conditions are a direct extension of those shown in Figure 7.16 for continuity requirements across a common boundary between two patches. The blend

Figure 7.32 A patch blended between two disjoint patches.

patch $\mathbf{r}(u,w)$ has two degrees of freedom, k_1 and k_2, that allow us to vary its shape somewhat. There are no restrictions on $\mathbf{p}(u,w)$ and $\mathbf{q}(u,w)$.

Blend to the Boundaries of Another Patch

If we are given a bicubic Hermite patch $\mathbf{q}(u,w)$, a portion of whose interior we wish to modify, then we might begin by letting $\mathbf{p}(u,w)$ describe the shape of the modified section. We assume that $\mathbf{q}(u,w)$ itself is part of a G^1 composite surface and is surrounded by other patches. No generality is lost by proceeding this way. We "remove" $\mathbf{q}(u,w)$ from the larger composite surface; "insert" $\mathbf{p}(u,w)$ and create eight new subpatches that blend it with G^1 continuity to the patches surrounding the "hole" left by the removal of $\mathbf{q}(u,w)$. This is an extension of the procedure illustrated in Figure 7.32. The influence of $\mathbf{p}(u,w)$ on the geometric coefficients of the surrounding eight blend patches is shown in Figure 7.33. We assume that $\mathbf{p}(u,w)$ is blended to the boundaries of $\mathbf{q}(u,w)$ such that the bounding curves of $\mathbf{p}(u,w)$ correspond to the constant-parameter values u_1, u_2, w_1, and w_2 on $\mathbf{q}(u,w)$, as shown in the figure. The blank elements in the $\mathbf{B}$ matrices are determined by the surrounding patches, and the constant coefficients k_i are

$$
\begin{aligned}
k_1 &= \frac{w_1}{w_2 - w_1} \\
k_2 &= \frac{1 - w_2}{w_2 - w_1} \\
k_3 &= \frac{u_1}{u_2 - u_1} \\
k_4 &= \frac{1 - u_2}{u_2 - u_1}
\end{aligned}
\tag{7.81}
$$

General Blend Surfaces

The construction of constant-radius blends between two intersecting parametric surfaces are studied by Choi and Ju (1989) and others. Their method allows the construction of constant-radius blends for any type of rectangular parametric surfaces as long as their offset surfaces are smooth, without singularities and self-intersections. (The offset distance is equal to the blend radius.) Edge blends are created by sweeping rational quadratic curves. Corner blends are created by a convex combination of Taylor interpolants.

Figure 7.33 Blending a patch to the boundaries of another patch.

Representing blend surfaces as solutions to partial differential equations is proposed by Bloor and Wilson (1989). They demonstrate that this approach easily achieves the required degree of continuity between the blend and primary surfaces. More work in blend surfaces and related topics is reported by Chiyokura (1987); Choi and Lee (1990); Hoffmann and Hopcroft (1987); Holmström (1987), who discusses piecewise quadric blending of implicitly defined surfaces; Middleditch and Sears (1985); Rockwood and Owen (1987), who discuss blend surfaces in solid modeling; and Woodward (1987), among many others.

8

BÉZIER SURFACES

The *Bézier surface* is a direct extension of the Bézier curve. Many topics applicable to these surfaces, but discussed under Bézier curves, are not repeated here. This chapter does discuss the bicubic Bézier patch, conversion between bicubic Bézier and Hermite forms, degree elevation, the composite Bézier surface, and, briefly, the rational Bézier surface.

8.1 THE TENSOR PRODUCT BÉZIER PATCH

Points on a Bézier patch are given by the following tensor product, a simple extension of the general equation for points on a Bézier curve:

$$\mathbf{p}(u,w) = \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{p}_{ij}B_{i,m}(u)\,B_{j,n}(w) \qquad u,w \in [0,1] \tag{8.1}$$

The $\mathbf{p}_{ij}$ comprise an $(m + 1) \times (n + 1)$ rectangular array of control points defining the vertices of the characteristic polyhedron of the Bézier patch, which lies entirely within its convex hull. $B_{im}(u)$ and $B_{jn}(w)$ are the basis functions, defined in the same way as for Bézier curves.

The general matrix equation for a Bézier patch is

$$\mathbf{p}(u,w) = \mathbf{U}\mathbf{M}_B\mathbf{P}\mathbf{M}_B^T\mathbf{W}^T \tag{8.2}$$

or

$$\mathbf{p}(u,w) = \mathbf{U}_{1\times m}\mathbf{M}_{B,m\times m}\mathbf{P}_{m\times n}\mathbf{M}_{B,n\times n}^T\mathbf{W}_{n\times 1}^T$$

where the dimensions of the matrices depend on the dimensions of the control point array. The subscripts indicating the matrix sizes are not repeated in later expressions, and they are used here merely to illustrate their relationship to the control points. The Bézier patch need not be described by a square array of control points, although we will begin by exploring a 4 × 4 array of points, which defines the bicubic Bézier patch.

8.2 THE BICUBIC BÉZIER PATCH

We can express a *bicubic Bézier patch* in a form similar to the bicubic Hermite patch. Using the binomial representation of the cubic Bézier curve, the matrix equation for a patch defined by a 4 × 4 array of control points is

$$\mathbf{p}(u,w) = \mathrm{U}_{1\times 4}\mathbf{M}_{B,4\times 4}\mathbf{P}_{4\times 4}\mathbf{M}^T_{B,4\times 4}\mathbf{W}^T_{4\times 1} \tag{8.3}$$

again, where the subscripts on the matrices indicate their dimensions. Expanded, this equation becomes

$$\mathbf{p}(u,w) = [(1-u)^3 \quad 3u(1-u)^2 \quad 3u^2(1-u) \quad u^3]\mathbf{P}\begin{bmatrix}(1-w)^3\\ 3w(1-w)^2\\ 3w^2(1-w)\\ w^3\end{bmatrix} \tag{8.4}$$

where

$$\mathbf{P} = \begin{bmatrix}\mathbf{p}_{11} & \mathbf{p}_{12} & \mathbf{p}_{13} & \mathbf{p}_{14}\\ \mathbf{p}_{21} & \mathbf{p}_{22} & \mathbf{p}_{23} & \mathbf{p}_{24}\\ \mathbf{p}_{31} & \mathbf{p}_{32} & \mathbf{p}_{33} & \mathbf{p}_{34}\\ \mathbf{p}_{41} & \mathbf{p}_{42} & \mathbf{p}_{43} & \mathbf{p}_{44}\end{bmatrix} \tag{8.5}$$

The matrix **P** contains the points that define the characteristic polyhedron and, thereby, the Bézier patch.

Figure 8.1 illustrates the control points, the polyhedron, and the resulting patch. In the bicubic Bézier formulation, only the four corner points $\mathbf{p}_{11}$, $\mathbf{p}_{41}$, $\mathbf{p}_{14}$, and $\mathbf{p}_{44}$ actually lie on the patch. The points $\mathbf{p}_{21}$, $\mathbf{p}_{31}$, $\mathbf{p}_{12}$, $\mathbf{p}_{13}$, $\mathbf{p}_{42}$, $\mathbf{p}_{43}$, $\mathbf{p}_{24}$, and $\mathbf{p}_{34}$ control the end slopes of the boundary curves. The four interior points $\mathbf{p}_{22}$, $\mathbf{p}_{32}$, $\mathbf{p}_{23}$, and $\mathbf{p}_{33}$ control the cross slopes along the boundary curves in the same way as do the twist vectors of the bicubic Hermite patch. As this figure shows, the Bézier patch is completely defined by a net of control points describing two families of Bézier curves on the patch. Each boundary curve is defined by a polygon of four points or vertices—the greater the number of points used, the higher the degree of the polynomial.

Figure B.1 Cubic Bézier patch.

Consider the generation of an isoparametric curve on the bicuhic Bézier patch for $w = w_i$, a constant. The following matrix product yields four points defining the Bézier curve along the patch at w_i:

$$\begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \mathbf{p}_4 \end{bmatrix} = \mathbf{P} \begin{bmatrix} (1-w_i)^3 \\ 3w_i(1-w_i)^2 \\ 3w_i^2(1-w_i) \\ w_i^3 \end{bmatrix} \tag{8.6}$$

where the single subscripts on $\mathbf{p}_1$, $\mathbf{p}_2$, and so on indicate that these points are curve-defining. The point $\mathbf{p}_1$ lies on the patch boundary curve $\mathbf{p}_{0w}$, and the point $\mathbf{p}_4$ lies on curve $\mathbf{p}_{1w}$. Points $\mathbf{p}_2$ and $\mathbf{p}_3$ control the slopes at either end of the curve and do not lie on it. Now we can write an expression for a point on the patch along the curve $\mathbf{p}(u,w_i)$:

$$\mathbf{p}(u,w_i) = [(1-u)^3 \quad 3u(1-u)^2 \quad 3u^2(1-u) \quad u^3] \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \mathbf{p}_4 \end{bmatrix} \tag{8.7}$$

Here each value of u yields a point on the patch. We use a similar procedure to define the boundary curves. For example, the curve $u = 0$ is

$$\mathbf{p}(0,w) = [\mathbf{p}_{11} \quad \mathbf{p}_{12} \quad \mathbf{p}_{13} \quad \mathbf{p}_{14}] \begin{bmatrix} (1-w)^3 \\ 3w(1-w)^2 \\ 3w^2(1-w) \\ w^3 \end{bmatrix} \tag{8.8}$$

or

$$\mathbf{p}(0,w) = [(1-w)^3 \quad 3w(1-w)^2 \quad 3w^2(1-w) \quad w^3]\begin{bmatrix}\mathbf{p}_{11}\\ \mathbf{p}_{12}\\ \mathbf{p}_{13}\\ \mathbf{p}_{14}\end{bmatrix} \tag{8.9}$$

The boundary curve for $w = 0$ is

$$\mathbf{p}(u,0) = [(1-u)^3 \quad 3u(1-u)^2 \quad 3u^2(1-u) \quad u^3]\begin{bmatrix}\mathbf{p}_{14}\\ \mathbf{p}_{24}\\ \mathbf{p}_{34}\\ \mathbf{p}_{44}\end{bmatrix} \tag{8.10}$$

We define the other two boundary curves similarly.

Appropriately differentiating Equation (8.4) produces the equation for the twist vector at any point on the bicubic Bézier patch:

$$\mathbf{p}^{uw}(u,w) = [(-3u^2+6u-3) \quad (9u^2-12u+3) \quad (-9u^2+6u) \quad (3u^2)]\mathbf{P}\begin{bmatrix}(-3w^2+6w-3)\\ (9w^2-12w+3)\\ (-9w^2+6w)\\ (3w^2)\end{bmatrix} \tag{8.11}$$

For the twist vectors at the patch corner points we have

$$\begin{aligned}
\mathbf{p}_{00}^{uw} &= 9(\mathbf{p}_{11} - \mathbf{p}_{21} - \mathbf{p}_{12} + \mathbf{p}_{22})\\
\mathbf{p}_{10}^{uw} &= 9(\mathbf{p}_{31} - \mathbf{p}_{41} - \mathbf{p}_{32} + \mathbf{p}_{42})\\
\mathbf{p}_{01}^{uw} &= 9(\mathbf{p}_{13} - \mathbf{p}_{23} - \mathbf{p}_{14} + \mathbf{p}_{24})\\
\mathbf{p}_{11}^{uw} &= 9(\mathbf{p}_{33} - \mathbf{p}_{43} - \mathbf{p}_{34} + \mathbf{p}_{44})
\end{aligned} \tag{8.12}$$

This tells us that only the four control points at and adjacent to the corner point in question affect the twist vector at that corner (see Figure 8.2).

8.3 A 3 × 5 RECTANGULAR ARRAY OF CONTROL POINTS

Occasionally we want to create a patch that is described by an array of control points that is not square. This allows a relatively complex surface to be bounded by simple curves, or vice versa. For example, the patch shown in Figure 8.3 has a three-point Bézier curve along boundaries $u = 0$ and $u = 1$

Figure B.2 Control points influencing the twist vector at a patch corner.

Figure B.3 5 × 3 Bézier patch.

and a five-point curve along boundaries $w = 0$ and $w = 1$. The equation for the boundary along $u = 0$ is

$$\mathbf{p}(0,w) = [(1-w)^2 \quad 2w(1-w) \quad w^2]\begin{bmatrix}\mathbf{p}_{11}\\ \mathbf{p}_{12}\\ \mathbf{p}_{13}\end{bmatrix} \tag{8.13}$$

Along $w = 0$ we obtain

$$\mathbf{p}(u,0) = [(1-u)^4 \quad 4u(1-u)^3 \quad 6u^2(1-u)^2 \quad 4u^3(1-u) \quad u^4]\begin{bmatrix}\mathbf{p}_{11}\\ \mathbf{p}_{21}\\ \mathbf{p}_{31}\\ \mathbf{p}_{41}\\ \mathbf{p}_{51}\end{bmatrix} \tag{8.14}$$

For the patch itself,

$$\mathbf{p}(u,w) = [(1-u)^4 \quad 4u(1-u)^3 \quad 6u^2(1-u)^2 \quad 4u^3(1-u) \quad u^4] \times \begin{bmatrix}\mathbf{p}_{11} & \mathbf{p}_{12} & \mathbf{p}_{13}\\ \mathbf{p}_{21} & \mathbf{p}_{22} & \mathbf{p}_{23}\\ \mathbf{p}_{31} & \mathbf{p}_{32} & \mathbf{p}_{33}\\ \mathbf{p}_{41} & \mathbf{p}_{42} & \mathbf{p}_{43}\\ \mathbf{p}_{51} & \mathbf{p}_{52} & \mathbf{p}_{53}\end{bmatrix}\begin{bmatrix}(1-w)^2\\ 2w(1-w)\\ w^2\end{bmatrix} \tag{8.15}$$

The advantage of a five-point boundary eurve over a four-point curve is that a change in the third or middle point of the curve does not affect the slope at either end. Thus, patch shape can be changed without changing the end slopes, hence maintaining continuity with adjacent patches.

8.4 CONVERTING BETWEEN BICUBIC BÉZIER AND HERMITE FORMS

To convert from the bicubic Bézier to the bicubic Hermite form, or vice versa, we begin with the mathematical statement of this equivalence.

$$\mathbf{U}\mathbf{M}_F\mathbf{B}\mathbf{M}_F^T\mathbf{W}^T = \mathbf{U}\mathbf{M}_B\mathbf{P}\mathbf{M}_B^T\mathbf{W}^T \tag{8.16}$$

or more simply,

$$\mathbf{M}_F\mathbf{B}\mathbf{M}_F^T = \mathbf{M}_B\mathbf{P}\mathbf{M}_B^T \tag{8.17}$$

where the basis function transformation matrices $\mathbf{M}_F$ and $\mathbf{M}_B$ are the Hermite and Bézier, respectively. To convert from Bézier to Hermite we solve for $\mathbf{B}$, obtaining

$$\mathbf{B} = \mathbf{M}_F^{-1}\mathbf{M}_B\mathbf{P}\mathbf{M}_B^T[\mathbf{M}_F^T]^{-1}$$

After performing the required matrix algebra, we find that

$$\mathbf{B} = \begin{bmatrix} \mathbf{p}_{11} & \mathbf{p}_{14} & 3(\mathbf{p}_{12} - \mathbf{p}_{11}) & 3(\mathbf{p}_{14} - \mathbf{p}_{13}) \\ \mathbf{p}_{41} & \mathbf{p}_{44} & 3(\mathbf{p}_{42} - \mathbf{p}_{41}) & 3(\mathbf{p}_{44} - \mathbf{p}_{43}) \\ 3(\mathbf{p}_{21} - \mathbf{p}_{11}) & 3(\mathbf{p}_{24} - \mathbf{p}_{14}) & 9(\mathbf{p}_{11} - \mathbf{p}_{21} - \mathbf{p}_{12} + \mathbf{p}_{22}) & 9(\mathbf{p}_{13} - \mathbf{p}_{23} - \mathbf{p}_{14} + \mathbf{p}_{24}) \\ 3(\mathbf{p}_{41} - \mathbf{p}_{31}) & 3(\mathbf{p}_{44} - \mathbf{p}_{34}) & 9(\mathbf{p}_{31} - \mathbf{p}_{41} - \mathbf{p}_{32} + \mathbf{p}_{42}) & 9(\mathbf{p}_{33} - \mathbf{p}_{43} - \mathbf{p}_{34} + \mathbf{p}_{44}) \end{bmatrix} \tag{8.18}$$

Here we see that the tangent vectors and twist vectors are equivalently expressed in terms of points defining the vertices of the characteristic polyhedron. The tangent vectors are given solely in terms of points on the boundary of the polyhedron, while the interior points contribute only to the twist vectors.

A similar process applies to convert from Hermite to Bézier. Thus,

$$\mathbf{P} = \mathbf{M}_B^{-1}\mathbf{M}_F\mathbf{B}\mathbf{M}_F^T[\mathbf{M}_B^T]^{-1} \tag{8.19}$$

8.5 DEGREE ELEVATION IN A BÉZIER SURFACE

A way to extend our ability to manipulate the shape of a Bézier surface requires adding more control points. This, of course, raises the degree of the polynomial defining the patch. We can increase the number of control points defining the u curves, the w curves, or both. The process is analogous to that applied to Bézier curves (see Section 4.2). We begin by adding control points in a way that does not change the shape of the patch. To elevate the degree of the Bézier patch in the u direction, we begin by equating the two shapes

$$\sum_{i=0}^{m+1}\sum_{j=0}^{n} {}^{1,0}\mathbf{p}_{ij} B_{i,m+1}(u)\, B_{j,n}(w) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{p}_{ij} B_{i,m}(u)\, B_{j,n}(w)$$

Solving for ${}^{1,0}\mathbf{p}_{ij}$ produces

$${}^{1,0}\mathbf{p}_{ij} = \frac{i}{m+1}\mathbf{p}_{i-1,j} + \left(1 - \frac{i}{m+1}\right)\mathbf{p}_{ij} \qquad \begin{cases} i = 0, \ldots, m+1 \\ j = 0, \ldots, n \end{cases} \tag{8.20}$$

(see Equation [4.10]). The double index on the upper left superscript indicates the number of times the degree-elevation procedure has been applied to each parametric direction, respectively.

To elevate the degree in the w direction:

$${}^{0,1}\mathbf{p}_{ij} = \frac{j}{n+1}\mathbf{p}_{i,j-1} + \left(1 - \frac{j}{n+1}\right)\mathbf{p}_{ij} \qquad \begin{cases} i = 0, \ldots, m \\ j = 0, \ldots, n+1 \end{cases} \tag{8.21}$$

Doing both at once produces a patch of degree $(m+1)\times(n+1)$. Thus

$$
{}^{1,1}\mathbf{p}_{ij} = \begin{bmatrix} \dfrac{i}{m+1} & 1-\dfrac{i}{m+1} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{i-1,j-1} & \mathbf{p}_{i-1,j} \\ \mathbf{p}_{i,j-1} & \mathbf{p}_{ij} \end{bmatrix} \begin{bmatrix} \dfrac{j}{n+1} \\ 1-\dfrac{j}{n+1} \end{bmatrix} \qquad \begin{cases} i=0,\ldots,m+1 \\ j=0,\ldots,n+1 \end{cases} \tag{8.22}
$$

8.6 COMPOSITE BÉZIER SURFACES

A simple geometric constraint allows us to assemble Bézier patches into more complex composite surfaces with at least G^1 continuity. Consider the 4×7 rectangular array of control points defining a two-patch composite surface. The array is idealized as a rectangular grid in Figure 8.4. The common boundary curve is defined by control points $\mathbf{p}_{14}$, $\mathbf{p}_{24}$, $\mathbf{p}_{34}$, and $\mathbf{p}_{44}$. To ensure at least G^1 continuity across this boundary, all four sets of three points $[\mathbf{p}_{i,3}, \mathbf{p}_{i,4}, \mathbf{p}_{i,5}]$, $i \in [1:4]$ must be collinear. We easily generalize this to other arrangements of Bézier patches.

Consider a composite surface constructed by a 3×3 array of bicubic Bézier patches (see Figure 8.5). For G^0 continuity, adjacent patches must share control points along their joining boundary, shown as solid circles in

Figure 8.4 G^1 continuity across two Bézier patches.

Figure 8.5 Composite G^1 continuous 3×3 array.

the figure. For at least G^1 continuity throughout the surface, the control points defining tangents across boundaries must be collinear. The corresponding tangent vectors thus have the same direction, but not necessarily the same magnitude.

8.7 RATIONAL BÉZIER PATCH

The rational Bézier patch is expressed as

$$\mathbf{p}(u,w) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} h_{ij}\mathbf{p}_{ij}B_{im}(u)\,B_{jn}(w)}{\sum_{i=0}^{m}\sum_{j=0}^{n} h_{ij}B_{im}(u)\,B_{jn}(w)} \tag{8.23}$$

This is the central projection of a four-dimensional tensor product hyperpatch, but it is not itself a tensor product. The properties and effects of the weights h_{ij} on patch shape are direct analogs of those for the rational Bézier curve.

One way to evaluate curves and points on this patch proceeds as follows. For a curve of constant parameter value, say, $w = w_a$, we treat each column of control points (or each row, if $u = u_b$) in the characteristic polyhedron as control points defining the characteristic polygon of a rational Bézier curve. We evaluate each of these curves at $w = w_a$, including computing the corresponding weight. Thus,

$$\mathbf{p}_i = \frac{\sum_{j=0}^{n} h_{ij}\mathbf{p}_{ij}B_{j,n}(w_a)}{\sum_{j=0}^{n} h_{ij}B_{j,n}(w_a)} \qquad i = 0, \ldots, m \tag{8.24}$$

and the weight h_i corresponding to each $\mathbf{p}_i$ is simply

$$h_i = \sum_{j=0}^{n} h_{ij}B_{j,n}(w_a) \tag{8.25}$$

We interpret the $\mathbf{p}_i$ and h_i as the control points and weights describing the $w = w_a$ curve on the surface. In general, $h_0, h_m \neq 1$, so that we may need to reparameterize.

9

B-SPLINE SURFACES

The equation of a *B-Spline surface* follows directly from the equation of a B-Spline curve. This relationship is analogous to that between Bézier curves and surfaces. Furthermore, we define the B-Spline surface, like the Bézier surface, in terms of a characteristic polyhedron. The shape of the surface approximates the polyhedron. The approximation is weaker the higher the degree.

9.1 THE TENSOR PRODUCT B-SPLINE SURFACE

The tensor product equation of the B-Spline surface is

$$\mathbf{p}(u,w) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{p}_{ij} N_{i,K}(u)\, N_{j,L}(w) \tag{9.1}$$

The $\mathbf{p}_{ij}$ are control points and are the vertices of the characteristic polyhedron. The $N_{i,K}(u)$ and $N_{j,L}(w)$ are the basis functions, and they are the same as those for B-Spline curves. The degree of each of the basis-function polynomials is controlled by K and L, respectively.

For a nonperiodic B-Spline surface, we select the values of K, L, m, n, and compute the t_i and t_j knot values just as for curves. We compute $N_{i,K}(u)$ and $N_{j,L}(w)$ recursively, using Equations (5.2) and (5.3). Notice that two sets of knot values are required and that the control points form an $(m+1) \times (n+1)$ array.

9.2 MATRIX FORM

The matrix form for the B-Spline surface is similar to the form that we developed for the B-Spline curve. Recall that the B-Spline curve is computed in segments of a unit interval on the parametric variable u (see Equations [5.14], [5.15], and [5.23]). A unit square on the parametric variables u and w is used to compute patches on the B-Spline surface. The general matrix equation of an open, periodic B-Spline surface that approximates an $(m+1)\times(n+1)$ rectangular array of points is

$$\mathbf{p}_u(u,w) = \mathbf{U}\mathbf{M}_S\mathbf{P}_{KL}\mathbf{M}_S^T\mathbf{W}^T \qquad \begin{array}{l} s \in [1:m+2-K] \\ t \in [1:n+2-L] \\ u,w \in [0,1] \end{array} \tag{9.2}$$

where K and L denote the parameters that control the continuity of the surface, the degree of the basis-function polynomials, and, consequently, the dimensions of the matrices. s and t identify a particular patch in the surface. The range on s and t is a function of the parameters K and L and the dimensions of the rectangular array of control points. The matrix $\mathbf{U}$ is

$$\mathbf{U} = [u^{K-1} \quad u^{K-2} \quad \cdots \quad u \quad 1] \tag{9.3}$$

and W is

$$\mathbf{W} = [w^{L-1} \quad w^{L-2} \quad \cdots \quad w \quad 1] \tag{9.4}$$

Elements of the full $K \times L$ matrix of control points necessary to fill $\mathbf{P}_{KL}$ depend on the particular patch to be evaluated. Let $\mathbf{p}_{ij}$ denote these matrix elements; then

$$\mathbf{P}_{KL} = \mathbf{p}_{ij} \qquad \begin{array}{l} i \in [s-1:s+K-2] \\ j \in [t-1:t+L-2] \end{array} \tag{9.5}$$

The matrix $\mathbf{M}_S$ is identical to the transformation matrix for the B-Spline curve.

9.3 OPEN AND CLOSED B-SPLINE SURFACES

If the B-Spline surface is partially closed—that is, rolled into an open-ended tube—then the ranges on s,t and i,j must reflect this closure. For example, if the w = constant curves are closed, then

$$s \in [1:m+1]$$

$$t \in [1 : n + 2 - L]$$
$$i \in [(s - 1) \bmod (m + 1) : (s + K - 2) \bmod (m + 1)]$$
$$j \in [t - 1 : t + L - 2] \tag{9.6}$$

Similar expressions apply when the u = constant curves are closed.

In the following figures we idealize an array of control points and the patches they produce with a two-dimensional diagram, arranging the points on a rectangular grid. Solid lines represent the edges of the faces of the characteristic polyhedron. Dashed lines represent the patch boundaries. For some idealizations, these lines overlap; when this happens, the dashed lines

Figure 9.1 Open quadric B-Spline surface idealizations.

are shown. The patches are lightly shaded to highlight them; this technique is useful for exploring the effects of parameters on the relationship between the control points and patches. Let us look at some examples. All of the following examples are periodic B-Spline surfaces generated by specific instances of Equation (9.2).

Figure 9.2 Open cubic B-Spline surface idealizations.

When $K = L = 3$, a quadric B-Spline surface is produced. This surface exhibits C^1 continuity throughout and requires a minimum 3×3 array of control points (see Figure 9.1a). If $m = 4$ and $n = 5$, then a 3×4 array of patches results (see Figure 9.1b).

When $K = L = 4$, a bicubic B-Spline surface is produced. C^2 continuity exists at all points on this surface and requires a minimum 4×4 array of control points (see Figure 9.2a); compare this to the bicubic Hermite and Bézier surfaces. If $m = 4$ and $n = 5$, then a 2×3 array of patches is produced (see Figure 9.2b); compare this to the result obtained for the quadric surface in Figure 9.1b.

Figure 9.3 Open quartic B-Spline surface idealizations.

Quartic surfaces are shown in Figure 9.3 and quintic surfaces in Figure 9.4. Notice that as the degree of the basis-function polynomials increases, more control points define each patch. This is particularly evident along the patch and control-point array boundary regions.

It is often more efficient to define a B-Spline surface with $K = L$. For example, a ruled surface may require second-derivative continuity in one para-

Figure 9.4 Open quintic B-Spline surface idealizations.

metric direction, while first-derivative continuity suffices for the other. A cubic-quadric B-Spline surface is shown in Figure 9.5. With $m = n = 4$, a 2×3 patch array is produced.

A partially closed B-Spline surface results when we impose the conditions in Equation (9.6) (see Figure 9.6). Here, $K = 4$ and $L = 3$, with $m = 3$ and $n = 4$, to produce a 4×3 patch array. The w = constant curves are closed; the u = constant curves are open. A two-dimensional idealization is shown in Figure 9.6a and a three-dimensional one in Figure 9.6b. Only 20 control points produce the complex surface in Figure 9.6c. Notice the characteristic polyhedron whose edges and vertices (control points) are superimposed.

An open, periodic bicubic B-Spline surface ($K = L = 4$) defined over a 2×2 array of patches requires 25 control points. A 3×3 array of patches requires 36 control points, and a 4×4 array requires 49 control points. We find that $n_p = (n_a + 3)^2$, where n_p is the number of control points defining an $n_a \times n_a$ array of patches. As n_a increases, $(n_a + 3)^2$ approaches n_a^2. If the patches are biquintic, then we need 49 control points for a 2×2 array of patches; 64 for a 3×3 array; and 81 for a 4×4 array, where $n_p = (n_a + 5)^2$. The number of control points is related to the degree of the basis-function polynomials by

$$n_p = (n_a + K - 1)^2 \tag{9.7}$$

where $K = L$.

Figure 9.5 Open cubic-quadric B-Spline surface idealizations.

Figure 9.6 Partially closed B-Spline surface.

If the surface is partially closed, then

$$n_p = n_a(n_a + K - 1) \tag{9.8}$$

One advantage of the B-Spline formulation is its ability to preserve arbitrarily high degrees of continuity over complex surfaces. This follows directly from the properties of the B-Spline curves. Also, any change in the local shape of a B-Spline surface is not propagated throughout the entire surface. These characteristics make the B-Spline surface suitable for use in an interactive modeling environment.

9.4 NONUNIFORM RATIONAL B-SPLINE SURFACES

The NURBS surface is, like the rational Bézier surface, the central projection into three-dimensional model space of a four-dimensional tensor product hypersurface. The underlying geometric characteristics of the NURBS curve generally apply to the surface, which is expressed as

$$\mathbf{p}(u,w) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} h_{ij}\mathbf{p}_{ij} N_{i,K}(u)\, N_{j,L}(w)}{\sum_{i=0}^{m} \sum_{j=0}^{n} h_{ij} N_{i,K}(u)\, N_{j,L}(w)} \tag{9.9}$$

where the h_{ij} are weights. As with the rational curves, increasing the weight values increases the fullness of the surface, pulling it toward the corresponding control points.

Curve and surface constructions using rational B-Splines, including the circular cylinder, general ruled surface, sphere, and torus, are discussed in detail by Piegl and Tiller (1987). Piegl (1989d) discusses the problem of modifying the shape of a rational B-Spline surface directly from its definition, using the weights and control points.

This page intentionally left blank

10

SOLIDS

Many techniques for modeling solids are limited because they do not represent the interior of the solid. They cannot represent internal properties or offer ways of representing internal behavior. The techniques assume total internal homogeneity of the model. For many purposes they are adequate. However, in this chapter, we investigate the mathematics and geometry supporting a more complete solid-representation scheme.

There are several well-known methods for creating simple solid shapes, simple in the sense that the solids are not Boolean constructions or otherwise complex assemblies of even simpler solids. The discussion here includes instances and parameterized solids, sweeps, and solids generated by some form of nonlinear transformation. Appendix C and Mortenson (1995) provide valuable supporting discussions for solid shape transformations.

10.1 PARAMETRIC SOLIDS

Perhaps the simplest and most direct mathematical approach we can use to model a solid is by continuous, three-parameter, single-valued functions of the form

$$x = x(u,v,w) \quad y = y(u,v,w) \quad z = z(u,v,w) \tag{10.1}$$

where the parametric variables u,v,w are constrained to the interval $u,v,w \in [0,1]$. These functions define the coordinates of the set of points comprising the solid, exterior as well as interior. The terms *hyperpatch* and *para-*

metric solid are often used interchangeably, although historically the term *hyperpatch* has a narrower meaning.

We will develop the mathematics of these solids as we did for curves and surfaces. In doing this, we find that fixing the value of one of the parametric variables results in a surface within or on the boundary of the solid in terms of the other two variables, which remain free. We continue this process for any number of arbitrary values in the allowed interval, first for one variable and then for each of the others in turn, forming a parametric network of cells of three two-parameter families of surfaces throughout the solid. Through each point $\mathbf{p}(u,v,w)$ there passes just one surface of each family. These surfaces are sometimes called *isoparametric surfaces*, indicating a surface within the solid on which one of the three parametric variables is constant. As with curves and surfaces, the positive sense on any surface is the sense in which the free parameters increase.

Associated with a parametric solid is a set of *boundary elements* (see Figure 10.1). The set consists of the 8 corner points, the 12 curves defining the edges, and the 6 patches defining the faces. Others include the tangent vectors and twist vectors of the edge curves and face patches.

For an ordinary parametric solid, there are always eight and only eight *corner points:* $\mathbf{p}(0,0,0)$, $\mathbf{p}(1,0,0)$, $\mathbf{p}(0,1,0)$, $\mathbf{p}(0,0,1)$, $\mathbf{p}(1,1,0)$, $\mathbf{p}(1,0,1)$, $\mathbf{p}(0,1,1)$, and $\mathbf{p}(1,1,1)$.

The *edge curves* are functions of one of the three parametric variables. We obtain them by allowing one of the variables to remain free while succes-

Figure 10.1 Parametric solid boundary elements.

sively fixing combinations of the other two at their limiting values. This results in the 12 possible combinations yielding the defining functions of the 12 parametric edge curves $\mathbf{p}(u,0,0)$, $\mathbf{p}(u,1,0)$, $\mathbf{p}(u,0,1)$, $\mathbf{p}(u,1,1)$, $\mathbf{p}(0,v,0)$, $\mathbf{p}(1,v,0)$, $\mathbf{p}(0,v,1)$, $\mathbf{p}(1,v,1)$, $\mathbf{p}(0,0,w)$, $\mathbf{p}(1,0,w)$, $\mathbf{p}(0,1,w)$, and $\mathbf{p}(1,1,w)$.

The *bounding faces* are functions of two of the three parametric variables. We obtain them by allowing two of the variables to remain free while successively fixing the remaining variable to each of its two limiting values. This results in the six possible permutations yielding $\mathbf{p}(u,v,0)$, $\mathbf{p}(u,v,1)$, $\mathbf{p}(u,0,w)$, $\mathbf{p}(u,1,w)$, $\mathbf{p}(0,v,w)$, and $\mathbf{p}(1,v,w)$.

A rectangular solid is the simplest example (see Figure 10.2). The following parametric equations represent a rectangular solid in an x,y,z coordinate system:

$$\begin{aligned} x &= (b-a)u + a \\ y &= (d-c)v + c \qquad u,v,w \in [0,1] \\ z &= (f-e)w + e \end{aligned} \tag{10.2}$$

This example illustrates an important feature. These parametric equations define not only the points comprising the bounding elements of the solid, but also all the points interior to it.

Here we introduce the *tricubic Hermite solid*. It is a natural continuation of the cubic and bicubic forms for curves and surfaces. The literature on parametric solids in general, and the tricubic model in particular, is sparse, espe-

Figure 10.2 Rectangular parametric solid.

cially when compared to research on the lower-dimension forms of multivariate interpolation functions. However, it is a fertile area for development, where a greater analytic potential is required of a solid model. This potential has already been amply demonstrated by Stanton and Crain (1974) and Timmer, et al. (1970) in the areas of structural mechanics and ablation thermo- and aerodynamics. Also see Lasser's (1985) Bernstein-Bézier representation of solids. Earlier limits on research and development of solid-modeling techniques of this type were due to the then seemingly massive data requirements. However, current computing resources have nearly removed this constraint.

10.2 THE TRICUBIC SOLID

The algebraic form of a tricubic solid is given by the following equation:

$$\mathbf{p}(u,v,w) = \sum_{i=0}^{3} \sum_{j=0}^{3} \sum_{k=0}^{3} \mathbf{a}_{ijk} u^i v^j w^k \qquad u,v,w \in [0,1] \tag{10.3}$$

The $\mathbf{a}_{ijk}$ vectors are the algebraic coefficients of the solid. Notice the obvious source of the term *tricubic:* Each of the three parametric variables can appear as a cubic term. The parametric variables u,v,w are restricted by definition to values in the interval 0 to 1, inclusive. This restriction makes the solid bounded in a regular way by cubic patches. We will explore this situation later in this section.

When we perform the indicated summation and expansion of the cubic polynomials in u,v,w for $x(u,v,w)$, we obtain

$$x(u,v,w) = a_{333x} u^3 v^3 w^3 + a_{332x} u^3 v^3 w^2 + \cdots + a_{000x} \tag{10.4}$$

There are similar expressions for $y(u,v,w)$ and $z(u,v,w)$. Each expression consists of 64 terms, which means that the total number of algebraic coefficients available is three times this, or 192 coefficients.

Next, we write these coefficients in a more compact form using vector notation:

$$\begin{aligned} \mathbf{p}(u,v,w) = {} & \mathbf{a}_{333} u^3 v^3 w^3 + \mathbf{a}_{332} u^3 v^3 w^2 + \cdots + \mathbf{a}_{330} u^3 v^3 \\ & + \mathbf{a}_{323} u^3 v^2 w^3 + \cdots \\ & \vdots \\ & + \mathbf{a}_{000} \end{aligned} \tag{10.5}$$

This is an improvement, but perhaps we can do even better if we use matrix notation. We are now confronted with a modest complication, since we have an extra parametric dimension. With three-component vectors and

three independent parametric variables, the matrices are more cumbersome. We reduce the product of the first and second cubic polynomials in u and v to matrix form as $\mathbf{UA}_{uv}\mathbf{V}^T$, and reduce the third polynomial to $\mathbf{A}_w\mathbf{W}^T$, so that

$$\mathbf{p}(u,v,w) = \mathbf{UA}_{uv}\mathbf{V}^T\mathbf{A}_w\mathbf{W}^T \tag{10.6}$$

where

$$\begin{aligned} \mathbf{U} &= [u^3 \quad u^2 \quad u \quad 1] \\ \mathbf{V} &= [v^3 \quad v^2 \quad v \quad 1] \\ \mathbf{W} &= [w^3 \quad w^2 \quad w \quad 1] \end{aligned} \tag{10.7}$$

Equation (10.6) is considerably more complex than the analogous equation for surfaces.

Here is another way to index Equation (10.3):

$$\mathbf{p}(u,v,w) = \sum_{i=1}^{4}\sum_{j=1}^{4}\sum_{k=1}^{4} \mathbf{a}_{ijk}u^{4-i}v^{4-j}w^{4-k} \tag{10.8}$$

This equation is compatible with the one we will encounter for the geometric form, and it is a more acceptable form of tensor notation. We can further simplify this by dropping the summation signs and adopting the convention that for cubics the range of the indices is from 1 to 4, so that

$$\mathbf{p}_{uvw} = \mathbf{a}_{ijk}u^{4-i}v^{4-j}w^{4-k} \tag{10.9}$$

The same approach applies to the geometric form

$$\mathbf{p}_{uvw} = F_i(u)\ F_j(v)\ F_k(w)\mathbf{b}_{ijk} \tag{10.10}$$

where a repeated subscript implies summation.

The F terms are the Hermite basis functions F_1, F_2, F_3, F_4, as determined by the subscripted index and in terms of the specified parametric variable. $\mathbf{b}_{ijk}$ is the array of boundary conditions or geometric vectors. Notice that $\mathbf{a}_{ijk}$ and $\mathbf{b}_{ijk}$ are three-component vectors. Again, there are $4 \times 4 \times 4$ or 64 algebraic vectors and 64 geometric vectors, so that there are 192 algebraic coefficients and, of course, 192 geometric coefficients.

Consider the $\mathbf{b}_{ijk}$ terms. How do we interpret them? Although we will investigate more formal procedures for doing this in the last section of this chapter, for now we will apply empirical analysis to their interpretation. Because the indices on the $\mathbf{b}$ array are coordinated with those indices on the F terms, we can draw some conclusions from our knowledge of these basis functions as they apply to curves and surfaces.

First we must have a frame of reference, so from the unit cube in parameter space we establish some nomenclature (see Figure 10.3). At each of the

Figure 10.3 Geometric coefficients on the unit cube.

eight corners, we find the following boundary conditions: the corner point itself, three tangent vectors, three twist vectors, and a vector defined by the third-order mixed partial derivative of the function. There are 8 vectors defining boundary conditions at each of the 8 corners, resulting in a total of 64 vectors. Here is an example, arbitrarily choosing the corner defined by $\mathbf{p}(1,0,1)$ and listing the eight boundary condition vectors:

$\mathbf{p}_{101}$	corner point	
$\mathbf{p}^u_{101}$, $\mathbf{p}^v_{101}$, $\mathbf{p}^w_{101}$	tangent vectors	$\frac{\partial}{\partial u}, \frac{\partial}{\partial v}, \frac{\partial}{\partial w}$
$\mathbf{p}^{uv}_{101}$, $\mathbf{p}^{uw}_{101}$, $\mathbf{p}^{vw}_{101}$	twist vectors	$\frac{\partial^2}{\partial u \partial v}, \frac{\partial^2}{\partial u \partial w}, \frac{\partial^2}{\partial v \partial w}$
$\mathbf{p}^{uvw}_{101}$	triple mixed partial	$\frac{\partial^3}{\partial u \partial v \partial w}$

Now, we return to the problem of interpreting the elements of the **b** array. Similar representation and expansion for curves and surfaces offer more clues. Thus, for curves we have

$$\mathbf{p} = \Sigma F_i \mathbf{b}_i \tag{10.11}$$

which we expand to

$$\mathbf{p} = F_1(u)\mathbf{b}_1 + F_2(u)\mathbf{b}_2 + F_3(u)\mathbf{b}_3 + F_4(u)\mathbf{b}_4 \tag{10.12}$$

and interpret as

$$\mathbf{p} = F_1(u)\mathbf{p}_0 + F_2(u)\mathbf{p}_1 + F_3(u)\mathbf{p}'_0 + F_4(u)\mathbf{p}'_1 \tag{10.13}$$

Assigning curve boundary condition vectors to elements of the $\mathbf{b}_i$ array is obvious. Now look at this development for surfaces:

$$\mathbf{p} = F_i(u)\ F_j(w)\mathbf{b}_{ij} \tag{10.14}$$

which we expand to

$$\begin{aligned}\mathbf{p} = {} & F_1(u)\ F_1(w)\mathbf{b}_{11} + F_1(u)\ F_2(w)\mathbf{b}_{12} + F_1(u)\ F_3(w)\mathbf{b}_{13} + F_1(u)\ F_4(w)\mathbf{b}_{14} \\ & + F_2(u)\ F_1(w)\mathbf{b}_{21} + F_2(u)\ F_2(w)\mathbf{b}_{22} + F_2(u)\ F_3(w)\mathbf{b}_{23} + F_2(u)\ F_4(w)\mathbf{b}_{24} \\ & + F_3(u)\ F_1(w)\mathbf{b}_{31} + F_3(u)\ F_2(w)\mathbf{b}_{32} + F_3(u)\ F_3(w)\mathbf{b}_{33} + F_3(u)\ F_4(w)\mathbf{b}_{34} \\ & + F_4(u)\ F_1(w)\mathbf{b}_{41} + F_4(u)\ F_2(w)\mathbf{b}_{42} + F_4(u)\ F_3(w)\mathbf{b}_{43} + F_4(u)\ F_4(w)\mathbf{b}_{44}\end{aligned} \tag{10.15}$$

and interpret as

$$\begin{aligned}\mathbf{p} = {} & F_1(u)\ F_1(w)\mathbf{p}_{00} + F_1(u)\ F_2(w)\mathbf{p}_{01} + F_1(u)\ F_3(w)\mathbf{p}^{w}_{00} + F_1(u)\ F_4(w)\mathbf{p}^{w}_{01} \\ & + F_2(u)\ F_1(w)\mathbf{p}_{10} + F_2(u)\ F_2(w)\mathbf{p}_{11} + F_2(u)\ F_3(w)\mathbf{p}^{w}_{10} + F_2(u)\ F_4(w)\mathbf{p}^{w}_{11} \\ & + F_3(u)\ F_1(w)\mathbf{p}^{u}_{00} + F_3(u)\ F_2(w)\mathbf{p}^{u}_{01} + F_3(u)\ F_3(w)\mathbf{p}^{uw}_{00} + F_3(u)\ F_4(w)\mathbf{p}^{uw}_{01} \\ & + F_4(u)\ F_1(w)\mathbf{p}^{u}_{10} + F_4(u)\ F_2(w)\mathbf{p}^{u}_{11} + F_4(u)\ F_3(w)\mathbf{p}^{uw}_{10} + F_4(u)\ F_4(w)\mathbf{p}^{uw}_{11}\end{aligned} \tag{10.16}$$

Assigning boundary condition vectors to the elements of the $\mathbf{b}_{ij}$ array is obvious by simply comparing Equations (10.15) and (10.16) term for term.

Some generalizations based on these equations first lead us to observe that when the indices of the basis functions of a term are all less than 3, then the associated **b** vector denotes a point. Second, when one, and only one, of the indices of a term is 3 or 4, then the associated **b** vector denotes a tangent vector. Third, when two, and only two, of the indices are 3 or 4, then the associated **b** vector denotes a twist vector (second-order mixed partial derivative). Fourth, when, in the case of a tricubic solid, three indices of a term are 3 or 4, then the associated **b** vector denotes that vector defined by the third-order mixed partial derivative of the function $\mathbf{p}(u,v,w)$.

Next, we must find a rule to tell us where the **b** vector is applied. Our current method of constructing the arrays of geometric coefficients suggests the following empirical interpretation: For each odd index on **b**, assign a zero, and for each even index, assign a one.

Now we are ready to test these observations on some examples. For a curve, given $\mathbf{b}_2$, then because the index is less than 3, we find that it denotes a

point. And because the index is even, we assign a one to the geometric coefficient; the result is $\mathbf{p}_1$. For a surface, given $\mathbf{b}_{41}$, we obtain $\mathbf{p}^u_{10}$. For a tricubic solid, given $\mathbf{b}_{131}$, we find that it denotes $\mathbf{p}^v_{000}$.

The transformation from geometric to algebraic coefficients is given by

$$\mathbf{a}_{ijk} = \sum_{l=1}^{4} \sum_{m=1}^{4} \sum_{n=1}^{4} \mathbf{M}_{il}\mathbf{M}_{jm}\mathbf{M}_{kn}\mathbf{b}_{lmn} \tag{10.17}$$

or more compactly as

$$\mathbf{a}_{ijk} = \mathbf{M}_{il}\mathbf{M}_{jm}\mathbf{M}_{kn}\mathbf{b}_{lmn} \tag{10.18}$$

Here $\mathbf{M}$ is the Hermite basis transformation matrix, and the various subscripted indices denote specific elements of the matrix.

Stanton and Crain (1974) developed a slightly different approach, treating the 64 hyperpatch parameters as 4 sets of 16 parameters. This allows us to understand more easily the basis-function properties and the relationship between patches and hyperpatches. They begin by defining three distinct but equivalent forms: the familiar algebraic and geometric forms, and a point form where $\mathbf{p}(u,v,w)$ is given at tbe one-third points of the parametric variables. The point form thus requires 64 points to be complete. These forms are given as the following triple summations:

Algebraic form: $$\mathbf{p}(u,v,w) = F^a_i(u)\, F^a_j(v)\, F^a_k(w)\mathbf{a}_{ijk} \tag{10.19}$$

Geometric form: $$\mathbf{p}(u,v,w) = F^b_i(u)\, F^b_j(v)\, F^b_k(w)\mathbf{b}_{ijk} \tag{10.20}$$

Point form: $$\mathbf{p}(u,v,w) = F^p_i(u)\, F^p_j(v)\, F^p_k(w)\mathbf{p}_{ijk} \tag{10.21}$$

$F^a_i(u)$, $F^a_j(v)$, $F^a_k(w)$ are basis functions applied to the algebraic coefficients, defined by

$$\begin{aligned} F^a_1(u) &= u^3 \\ F^a_2(u) &= u^2 \\ F^a_3(u) &= u \\ F^a_4(u) &= 1 \end{aligned} \tag{10.22}$$

and similarly for v and w.

$F^b_i(u)$, $F^b_j(v)$, $F^b_k(w)$ are basis functions applied to the geometric coefficients, defined by

$$F^b_1(u) = 2u^3 - 3u^2 + 1$$

$$F^b_2(u) = -2u^3 + u^2$$

$$F_3^b(u) = u^3 - 2u^2 + u$$
$$F_4^b(u) = u^3 - u^2 \tag{10.23}$$

and similarly for v and w. Equations (10.23) are simply from $\mathbf{F} = \mathbf{UM}$.

Finally, $F_i^p(u)$, $F_j^p(v)$, $F_k^p(w)$ are basis functions applied to the point coefficients, defined by

$$F_1^p(u) = \left(-\frac{9}{2}\right)u^3 + 9u^2 - \left(\frac{11}{2}\right)u + 1$$
$$F_2^p(u) = \left(\frac{27}{2}\right)u^3 - \left(\frac{45}{2}\right)u^2 + 9u$$
$$F_3^p(u) = \left(-\frac{27}{2}\right)u^3 + 18u^2 - \left(\frac{9}{2}\right)u \tag{10.24}$$
$$F_4^p(u) = \left(\frac{9}{2}\right)u^3 - \left(\frac{9}{2}\right)u^2 + u$$

and similarly for v and w. Equations (10.24) are from $\mathbf{F} = \mathbf{UN}$.

We are free to contract any of the three parametric coordinates. If we do this for w, then we write the geometric coefficients in a sequence of four 16-element arrays. We denote these arrays as the matrices

$$\mathbf{B}_1 = [\mathbf{b}_{ij1}]$$
$$\mathbf{B}_2 = [\mathbf{b}_{ij2}]$$
$$\mathbf{B}_3 = [\mathbf{b}_{ij3}] \tag{10.25}$$
$$\mathbf{B}_4 = [\mathbf{b}_{ij4}]$$

We expand them as follows:

$$\mathbf{B}_1 = \begin{bmatrix} \mathbf{p}_{000} & \mathbf{p}_{010} & \mathbf{p}^v_{000} & \mathbf{p}^v_{010} \\ \mathbf{p}_{100} & \mathbf{p}_{110} & \mathbf{p}^v_{100} & \mathbf{p}^v_{110} \\ \mathbf{p}^u_{000} & \mathbf{p}^u_{010} & \mathbf{p}^{uv}_{000} & \mathbf{p}^{uv}_{010} \\ \mathbf{p}^u_{100} & \mathbf{p}^u_{110} & \mathbf{p}^{uv}_{100} & \mathbf{p}^{uv}_{110} \end{bmatrix}$$

$$\mathbf{B}_2 = \begin{bmatrix} \mathbf{p}_{001} & \mathbf{p}_{011} & \mathbf{p}^v_{001} & \mathbf{p}^v_{011} \\ \mathbf{p}_{101} & \mathbf{p}_{111} & \mathbf{p}^v_{101} & \mathbf{p}^v_{111} \\ \mathbf{p}^u_{001} & \mathbf{p}^u_{011} & \mathbf{p}^{uv}_{001} & \mathbf{p}^{uv}_{011} \\ \mathbf{p}^u_{101} & \mathbf{p}^u_{111} & \mathbf{p}^{uv}_{101} & \mathbf{p}^{uv}_{111} \end{bmatrix}$$

$$\mathbf{B}_3 = \begin{bmatrix} \mathbf{p}^{w}_{000} & \mathbf{p}^{w}_{010} & \mathbf{p}^{vw}_{000} & \mathbf{p}^{vw}_{010} \\ \mathbf{p}^{w}_{100} & \mathbf{p}^{w}_{110} & \mathbf{p}^{vw}_{100} & \mathbf{p}^{vw}_{110} \\ \mathbf{p}^{uw}_{000} & \mathbf{p}^{uw}_{010} & \mathbf{p}^{uvw}_{000} & \mathbf{p}^{uvw}_{010} \\ \mathbf{p}^{uw}_{100} & \mathbf{p}^{uw}_{110} & \mathbf{p}^{uvw}_{100} & \mathbf{p}^{uvw}_{110} \end{bmatrix} \tag{10.26}$$

$$\mathbf{B}_4 = \begin{bmatrix} \mathbf{p}^{w}_{001} & \mathbf{p}^{w}_{011} & \mathbf{p}^{vw}_{001} & \mathbf{p}^{vw}_{011} \\ \mathbf{p}^{w}_{101} & \mathbf{p}^{w}_{111} & \mathbf{p}^{vw}_{101} & \mathbf{p}^{vw}_{111} \\ \mathbf{p}^{uw}_{001} & \mathbf{p}^{uw}_{011} & \mathbf{p}^{uvw}_{001} & \mathbf{p}^{uvw}_{011} \\ \mathbf{p}^{uw}_{101} & \mathbf{p}^{uw}_{111} & \mathbf{p}^{uvw}_{101} & \mathbf{p}^{uvw}_{111} \end{bmatrix}$$

We can think of these equations as being organized in a three-dimensional array of vectors, as in Figure 10.4. We must keep in mind, however, that because the elements of each array are vectors, the array is more correctly $4 \times 4 \times 4 \times 3$, a four-dimensional array.

We take a similar approach with the point form, contracting the third parametric variable w and expressing the point-form coefficients as a sequence of four 16-element arrays. We denote these arrays as the matrices

$$\mathbf{p}_1 = [\mathbf{p}_{ij1}], \quad \mathbf{p}_2 = [\mathbf{p}_{ij2}], \quad \mathbf{p}_3 = [\mathbf{p}_{ij3}], \quad \mathbf{p}_4 = [\mathbf{p}_{ij4}] \tag{10.27}$$

and expand as follows

Figure 10.4 Three-dimensional array of geometric coefficients of a parametric solid.

$$
\mathbf{p}_1 = \begin{bmatrix} \mathbf{p}(0,0,0) & \mathbf{p}\left(0,\frac{1}{3},0\right) & \mathbf{p}\left(0,\frac{2}{3},0\right) & \mathbf{p}(0,1,0) \\ \mathbf{p}\left(\frac{1}{3},0,0\right) & \mathbf{p}\left(\frac{1}{3},\frac{1}{3},0\right) & \mathbf{p}\left(\frac{1}{3},\frac{2}{3},0\right) & \mathbf{p}\left(\frac{1}{3},1,0\right) \\ \mathbf{p}\left(\frac{2}{3},0,0\right) & \mathbf{p}\left(\frac{2}{3},\frac{1}{3},0\right) & \mathbf{p}\left(\frac{2}{3},\frac{2}{3},0\right) & \mathbf{p}\left(\frac{2}{3},1,0\right) \\ \mathbf{p}(1,0,0) & \mathbf{p}\left(1,\frac{1}{3},0\right) & \mathbf{p}\left(1,\frac{2}{3},0\right) & \mathbf{p}(1,1,0) \end{bmatrix}
$$

$$
\mathbf{p}_2 = \begin{bmatrix} \mathbf{p}\left(0,0,\frac{1}{3}\right) & \mathbf{p}\left(0,\frac{1}{3},\frac{1}{3}\right) & \mathbf{p}\left(0,\frac{2}{3},\frac{1}{3}\right) & \mathbf{p}\left(0,1,\frac{1}{3}\right) \\ \mathbf{p}\left(\frac{1}{3},0,\frac{1}{3}\right) & \mathbf{p}\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right) & \mathbf{p}\left(\frac{1}{3},\frac{2}{3},\frac{1}{3}\right) & \mathbf{p}\left(\frac{1}{3},1,\frac{1}{3}\right) \\ \mathbf{p}\left(\frac{2}{3},0,\frac{1}{3}\right) & \mathbf{p}\left(\frac{2}{3},\frac{1}{3},\frac{1}{3}\right) & \mathbf{p}\left(\frac{2}{3},\frac{2}{3},\frac{1}{3}\right) & \mathbf{p}\left(\frac{2}{3},1,\frac{1}{3}\right) \\ \mathbf{p}\left(1,0,\frac{1}{3}\right) & \mathbf{p}\left(1,\frac{1}{3},\frac{1}{3}\right) & \mathbf{p}\left(1,\frac{2}{3},\frac{1}{3}\right) & \mathbf{p}\left(1,1,\frac{1}{3}\right) \end{bmatrix}
$$

$$
\mathbf{p}_3 = \begin{bmatrix} \mathbf{p}\left(0,0,\frac{2}{3}\right) & \mathbf{p}\left(0,\frac{1}{3},\frac{2}{3}\right) & \mathbf{p}\left(0,\frac{2}{3},\frac{2}{3}\right) & \mathbf{p}\left(0,1,\frac{2}{3}\right) \\ \mathbf{p}\left(\frac{1}{3},0,\frac{2}{3}\right) & \mathbf{p}\left(\frac{1}{3},\frac{1}{3},\frac{2}{3}\right) & \mathbf{p}\left(\frac{1}{3},\frac{2}{3},\frac{2}{3}\right) & \mathbf{p}\left(\frac{1}{3},1,\frac{2}{3}\right) \\ \mathbf{p}\left(\frac{2}{3},0,\frac{2}{3}\right) & \mathbf{p}\left(\frac{2}{3},\frac{1}{3},\frac{2}{3}\right) & \mathbf{p}\left(\frac{2}{3},\frac{2}{3},\frac{2}{3}\right) & \mathbf{p}\left(\frac{2}{3},1,\frac{2}{3}\right) \\ \mathbf{p}\left(1,0,\frac{2}{3}\right) & \mathbf{p}\left(1,\frac{1}{3},\frac{2}{3}\right) & \mathbf{p}\left(1,\frac{2}{3},\frac{2}{3}\right) & \mathbf{p}\left(1,1,\frac{2}{3}\right) \end{bmatrix} \tag{10.28}
$$

$$
\mathbf{p}_3 = \begin{bmatrix} \mathbf{p}(0,0,1) & \mathbf{p}\left(0,\frac{1}{3},1\right) & \mathbf{p}\left(0,\frac{2}{3},1\right) & \mathbf{p}(0,1,1) \\ \mathbf{p}\left(\frac{1}{3},0,1\right) & \mathbf{p}\left(\frac{1}{3},\frac{1}{3},1\right) & \mathbf{p}\left(\frac{1}{3},\frac{2}{3},1\right) & \mathbf{p}\left(\frac{1}{3},1,1\right) \\ \mathbf{p}\left(\frac{2}{3},0,1\right) & \mathbf{p}\left(\frac{2}{3},\frac{1}{3},1\right) & \mathbf{p}\left(\frac{2}{3},\frac{2}{3},1\right) & \mathbf{p}\left(\frac{2}{3},1,1\right) \\ \mathbf{p}(1,0,1) & \mathbf{p}\left(1,\frac{1}{3},1\right) & \mathbf{p}\left(1,\frac{2}{3},1\right) & \mathbf{p}(1,1,1) \end{bmatrix}
$$

Figure 10.5 illustrates the point distribution within the unit cube in parameter space.

Because the algebraic coefficients $\mathbf{a}_{ijk}$ are too cumbersome to present this way, we express them in an unexpanded matrix format, determining them by operating on the geometric coefficients with the universal transformation matrix $\mathbf{M}$. Thus, we obtain

$$\begin{aligned}
\mathbf{a}_{ij1} &= \mathbf{M}_{11}\mathbf{M}\mathbf{B}_1\mathbf{M}^T + \mathbf{M}_{12}\mathbf{M}\mathbf{B}_2\mathbf{M}^T + \mathbf{M}_{13}\mathbf{M}\mathbf{B}_3\mathbf{M}^T + \mathbf{M}_{14}\mathbf{M}\mathbf{B}_4\mathbf{M}^T \\
\mathbf{a}_{ij2} &= \mathbf{M}_{21}\mathbf{M}\mathbf{B}_1\mathbf{M}^T + \mathbf{M}_{22}\mathbf{M}\mathbf{B}_2\mathbf{M}^T + \mathbf{M}_{23}\mathbf{M}\mathbf{B}_3\mathbf{M}^T + \mathbf{M}_{24}\mathbf{M}\mathbf{B}_4\mathbf{M}^T \\
\mathbf{a}_{ij3} &= \mathbf{M}_{31}\mathbf{M}\mathbf{B}_1\mathbf{M}^T + \mathbf{M}_{32}\mathbf{M}\mathbf{B}_2\mathbf{M}^T + \mathbf{M}_{33}\mathbf{M}\mathbf{B}_3\mathbf{M}^T + \mathbf{M}_{34}\mathbf{M}\mathbf{B}_4\mathbf{M}^T \\
\mathbf{a}_{ij4} &= \mathbf{M}_{41}\mathbf{M}\mathbf{B}_1\mathbf{M}^T + \mathbf{M}_{42}\mathbf{M}\mathbf{B}_2\mathbf{M}^T + \mathbf{M}_{43}\mathbf{M}\mathbf{B}_3\mathbf{M}^T + \mathbf{M}_{44}\mathbf{M}\mathbf{B}_4\mathbf{M}^T
\end{aligned} \tag{10.29}$$

where $\mathbf{M}_{lm}$ is a single specific element of the universal transformation matrix and $\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3, \mathbf{B}_4$ are given by Equations (10.26). If we let

$$\mathbf{A}_1 = [\mathbf{a}_{ij1}], \quad \mathbf{A}_2 = [\mathbf{a}_{ij2}], \quad \mathbf{A}_3 = [\mathbf{a}_{ij3}], \quad \mathbf{A}_4 = [\mathbf{a}_{ij4}] \tag{10.30}$$

or, in general,

$$\mathbf{A}_k = [\mathbf{a}_{ijk}] \tag{10.31}$$

then we rewrite Equation (10.29) more compactly as

$$\mathbf{A}_k = \mathbf{M}_{ki}\mathbf{M}\mathbf{B}_i\mathbf{M}^T \tag{10.32}$$

where we take the summation over the repeated subscript i.

Figure 10.5 Sixty-four-point form of a tricubic solid.

We would like to be able to transform equations from point form to geometric form. To do this, we introduce a patch function that we denote as $\mathbf{G}(w)$ and define as follows:

$$\mathbf{G}(w) = \mathbf{F}_i^b(w)\mathbf{B}_i \tag{10.33}$$

This defines a bicubic patch corresponding to a specific value of w, given the hyperpatch array $[\mathbf{b}_{ijk}]$ from which we extract the $\mathbf{B}_i$ arrays. Next, we evaluate this function at the one-third points to obtain

$$\mathbf{G}_j = \mathbf{M}^{-1}\mathbf{N}\mathbf{P}_j\mathbf{N}^T[\mathbf{M}^{-1}]^T \tag{10.34}$$

where $\mathbf{M}$ is the basis transformation matrix and $\mathbf{N}$ is the array of basis-function coefficients $\mathbf{F}_i^p$ from Equation (10.24). Thus,

$$\mathbf{N} = \begin{bmatrix} -\frac{9}{2} & \frac{27}{2} & -\frac{27}{2} & \frac{9}{2} \\ 9 & -\frac{45}{2} & 18 & -\frac{9}{2} \\ -\frac{11}{2} & 9 & -\frac{9}{2} & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{10.35}$$

We now substitute the results from Equation (10.34) into Equation (10.33) and solve for $\mathbf{B}_i$ to obtain the final transformation equation

$$\mathbf{B}_i = [F_{ij}^b]^{-1}\,\mathbf{M}^{-1}\mathbf{N}\mathbf{P}_j\mathbf{N}^T[\mathbf{M}^{-1}]^T \tag{10.36}$$

where $[\mathbf{F}_{ij}^b]$ is a matrix whose elements are the basis functions in Equation (10.23) evaluated at the one-third points. The expanded form of this matrix is

$$[\mathbf{F}_{ij}^b] = \begin{bmatrix} \mathbf{F}_1^b(0) & \mathbf{F}_2^b(0) & \mathbf{F}_3^b(0) & \mathbf{F}_4^b(0) \\ \mathbf{F}_1^b\left(\frac{1}{3}\right) & \mathbf{F}_2^b\left(\frac{1}{3}\right) & \mathbf{F}_3^b\left(\frac{1}{3}\right) & \mathbf{F}_4^b\left(\frac{1}{3}\right) \\ \mathbf{F}_1^b\left(\frac{2}{3}\right) & \mathbf{F}_2^b\left(\frac{2}{3}\right) & \mathbf{F}_3^b\left(\frac{2}{3}\right) & \mathbf{F}_4^b\left(\frac{2}{3}\right) \\ \mathbf{F}_1^b(1) & \mathbf{F}_2^b(1) & \mathbf{F}_3^b(1) & \mathbf{F}_4^b(1) \end{bmatrix} \tag{10.37}$$

When this matrix is evaluated, it becomes

$$[\mathbf{F}_{ij}^{b}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{11}{2} & 9 & -\frac{9}{2} & 1 \\ -1 & \frac{9}{2} & -9 & \frac{11}{2} \end{bmatrix} \tag{10.38}$$

and its inverse is

$$[\mathbf{F}_{ij}^{b}]^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{20}{27} & \frac{7}{27} & \frac{4}{27} & -\frac{2}{27} \\ \frac{7}{27} & \frac{20}{27} & \frac{2}{27} & -\frac{4}{27} \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{10.39}$$

We recognize them as the arrays from Equations (3.37) and (3.38), respectively.

There is no restriction on the basis used for a tricubic solid or any other single or mixed-degree solid. We could create a Bézier parametric solid, using a three-dimensional lattice of control points, or we could create a B-Spline solid.

Tangent Vectors and Twist Vectors

Computing the parametric derivatives of the tricubic solid involves processes similar to those for the bicubic patch. In fact, as we might suspect, the parallels continue with higher-dimension elements. We express the tricubic function as three independent variables, of course, and we must compute partial derivatives of this function.

We encountered these vectors earlier, because they are important constituents of the geometric coefficients. Let us summarize them briefly here.

We now have three species of tangent vectors,

$$\begin{aligned} \mathbf{p}_{uvw}^{u} &= \frac{\partial}{\partial u}\mathbf{p}(u,v,w) \\ \mathbf{p}_{uvw}^{v} &= \frac{\partial}{\partial v}\mathbf{p}(u,v,w) \\ \mathbf{p}_{uvw}^{w} &= \frac{\partial}{\partial w}\mathbf{p}(u,v,w) \end{aligned} \tag{10.40}$$

three of twist vectors,

$$\mathbf{p}_{uvw}^{uv} = \frac{\partial^2}{\partial u\, \partial v}\mathbf{p}(u,v,w)$$

$$\mathbf{p}_{uvw}^{uw} = \frac{\partial^2}{\partial u\, \partial w}\mathbf{p}(u,v,w) \tag{10.41}$$

$$\mathbf{p}_{uvw}^{vw} = \frac{\partial^2}{\partial v\, \partial w}\mathbf{p}(u,v,w)$$

and the singular triple-mixed partial,

$$\mathbf{p}_{uvw}^{uvw} = \frac{\partial^3}{\partial u\, \partial v\, \partial w}\mathbf{p}(u,v,w) \tag{10.42}$$

Just as these vectors change and control the shape of Hermite curves and surfaces, so do they also control the shape of a tricubic solid. These vectors control not only the shape of the edges and faces, but also the interior distribution of the parametric variables. Figure 10.6 illustrates some of the tangent vectors in place on a solid. Notice that the triple-mixed partial vectors $\mathbf{p}_{uvw}^{uvw}$ control the change of the twist vectors through the solid.

Figure 10.6 Parametric solid tangent vectors.

These vectors provide a variety of ways to control the exterior shape of a solid. By fixing the corner points of a solid, we can create concave and convex faces at will by manipulating the tangent and twist vectors. For simple shapes and constrained demands on interior behavior, we can set equal to zero some or all of the mixed vectors defined by mixed partial derivatives, in the manner of the F-patch for surfaces.

The Parameter Space of a Solid

The parameter space of a solid differs from that of a curve or surface, of course, because we must account for an additional parametric variable. We now have a set of four-dimensional parameter spaces defined by (u,v,w,x), (u,v,w,y), and (u,v,w,z) coordinates. Contrary to curves and surfaces, decomposing a tricubic solid into its parameter-space components is much more difficult to visualize and, therefore, less likely to help us understand its behavior in model space.

We can, however, apply this technique to the faces in exactly the same way as we did for bicubic patches. In fact, we can take parametric slices generating surfaces on which one of the parametric variables is constant and then proceed with the familiar bicubic decomposition.

Figure 10.7 shows an interesting approach. We start with the unit cube. At some convenient distance d from each of the six faces, we construct a unit square in a plane parallel to the face. This plane is designated as the $x = 0$ plane (or $y = 0$ or $z = 0$). Now we proceed with the bicubic decomposition of each of the six faces. We might produce some interesting effects, if not insights, from a dynamic computer-graphics display of the resulting phenomena.

Continuity and Composite Solids

We now consider the possible continuity requirements at the common boundary surface of two joined solids. Figure 10.8 gives the details of the conditions across the interior of this face. We will have satisfied all required continuity conditions across the adjoining external bounding surfaces of the two solids $\mathbf{p}(u,v,w)$ and $\mathbf{q}(u,v,w)$. Also, we will ignore curvature continuity and focus on G^1 parametric continuity across this surface. To ensure C^1 continuity, we must meet the following condition on the common surface:

$$\mathbf{p}_{1vw} = \mathbf{q}_{0vw} \tag{10.43}$$

For G^1 continuity, we must investigate the curves of constant v and w passing through each of these points. The tangent vectors of the curves at these points must be collinear; thus,

Figure 10.7 Parameter space of a parametric solid.

$$\mathbf{p}^{u}_{1vw} = k\mathbf{q}^{u}_{0vw} \tag{10.44}$$

The surface normals of the adjoining surfaces are in the same direction if

$$\mathbf{p}^{v}_{1vw} \times \mathbf{p}^{w}_{1vw} = a(\mathbf{q}^{v}_{0vw} \times \mathbf{q}^{w}_{0vw}) \tag{10.45}$$

If we modify this condition so that we now have

$$\mathbf{p}^{v}_{1vw} \times \mathbf{p}^{w}_{1vw} = b\mathbf{p}^{u}_{1vw} \tag{10.46}$$

then the tangent vectors indicated are also normal to the surface.

10.3 CURVES AND SURFACES EMBEDDED IN A SOLID

We found that a parametric surface supports a two-dimensional curvilinear coordinate system on which we can define curves. In a directly analogous way, a parametric solid supports a three-dimensional curvilinear coordinate system on which we can define both curves and surfaces, the simplest being the *isoparametric* curves and surfaces. An isoparametric curve, for example, results when we hold two of the three tricubic parametric variables fixed at some constant values. The curve $\mathbf{p}_{abw}$ in Figure 10.9 results when we assign $u = a$ and $v = b$, where a and b are constants. In the same figure, we find two isoparametric surfaces $\mathbf{p}_{avw}$ and $\mathbf{p}_{ubw}$.

Figure 10.8 Continuity conditions.

A *parametric cell* is a subelement of the solid bounded by six isoparametric surfaces. For example, in Figure 10.10, the six boundary surfaces are $\mathbf{p}_{avw}$, $\mathbf{p}_{bvw}$, $\mathbf{p}_{ucw}$, $\mathbf{p}_{udw}$, $\mathbf{p}_{uve}$, and $\mathbf{p}_{uvf}$. Such a cell is orthogonal if the curve nets of the three families of isoparametric surfaces are defined on orthogonal parametric curve nets.

We define a *nonisoparametric* curve by introducing an additional parametric variable t. The curve $\mathbf{c}(t)$ is shown mapped in a tricubic solid in both model space and parameter space in Figure 10.11. The curvilinear vector components of points on the curve are

$$\mathbf{c}(t) = \mathbf{u}(t) + \mathbf{v}(t) + \mathbf{w}(t) \tag{10.47}$$

In model space, the curve is expressed as

$$\mathbf{c}(x,y,z) = \mathbf{p}[u(t),v(t),w(t)] \tag{10.48}$$

A distorted cubical solid.

Figure 10.9 Isoparametric curves and surfaces in a parametric solid.

Figure 10.10 Parametric and orthogonal cells.

Figure 10.11 General curve in a parametric solid.

The procedures are identical to those we use for curves embedded in surfaces.

Because a parametric solid offers a three-dimensional curvilinear coordinate system, we have the additional option of defining nonisoparametric surfaces within the solid. In Figure 10.12, a surface patch is mapped in both the unit cube of parameter space and in a corresponding solid in model space. The vector components of the patch in parameter space are

$$\mathbf{r}(s,t) = \mathbf{u}(s,t) + \mathbf{v}(s,t) + \mathbf{w}(s,t) \tag{10.49}$$

In model space, the patch is expressed as

$$\mathbf{r}(x,y,z) = \mathbf{p}[u(s,t),v(s,t),w(s,t)] \tag{10.50}$$

Again we proceed as with curves in solids and on surfaces. We can choose a variety of surfaces to model in the parametric solid. What happens when a Bézier surface lies within a tricubic solid? How do we map such a surface in model space?

A variety of complex solids is possible if we permit them to have irregular or nonisoparametric boundaries, often called *trimmed boundaries*. A specific

Figure 10.12 Nonisoparametric surfaces in a parametric solid.

example is shown in Figure 10.13. The solid has one of its corners clipped and defined by a nonisoparametric surface. The surface is conveniently defined initially in either parameter space or model space, depending on the modeling situation.

10.4 GENERALIZED NOTATION SCHEME AND HIGHER-DIMENSION ELEMENTS

To introduce each subject, the preceding sections used a traditional algebraic notation scheme. Incorporating vector and matrix notation significantly reduces the complexity and awkwardness of these formulations. The next logical step is to introduce a more generalized summation scheme that extends to the representation of not only curves and surfaces, but also to solids and higher-order geometric forms. The notation convention we use

Figure 10.13 A parametric solid with a trimmed boundary.

here (largely attributable to Timmer) has these characteristics and others. It can extend to spaces of any dimension; it applies to polycubic interpolation involving an arbitrary number of independent parametric variables and suggests a system that can directly extend to any odd-ordered polynomial form. We will apply the following general symbol definitions:

$\mathbf{a}_{j_1 j_2 \cdots j_n} =$	vector elements of the algebraic coefficient array **A**
$a_{ij_1 j_2 \cdots j_n} =$	ith component of $\mathbf{a}_{j_1 j_2 \cdots j_n}$
$\mathbf{b}_{j_1 j_2 \cdots j_n} =$	vector elements of the geometric coefficient array **B**
$b_{ij_1 j_2 \cdots j_n} =$	ith component of $\mathbf{b}_{j_1 j_2 \cdots j_n}$
$\mathbf{p}(x_1, x_2, \ldots, x_n) =$	point vector in model space (dependent variables)
$u =$	parametric variable (independent variable)
$m_{jk} =$	elements of the transformation matrix **M**

The algebraic form is

$$x_i(u_1, u_2, \ldots, u_n) = \sum_{j_1=1}^{4} \sum_{j_2=1}^{4} \cdots \sum_{j_n=1}^{4} (a_{ij_1j_2\ldots j_n} u_1^{4-j_1} u_2^{4-j_2} \cdots u_n^{4-j_n}) \qquad (10.51)$$

where $i \in [1:m]$ and p_i is the ith component of a dependent-variable vector of m components, each defined by this n-cubic polynomial parametric function. We can omit the summation symbols. The geometric element described is of nth order, and the equation of each component is of $3n$th degree and has $4n$ terms. There are m dependent variables and n independent or parametric variables. Using the bicubic Hermite patch in three-space as an example, we have $m = 3$ and $n = 2$, and we rewrite Equation (10.51) as

$$x_i(u_1, u_2) = \sum_{j_1=1}^{4} \sum_{j_2=1}^{4} a_{ij_1j_2} u_1^{4-j_1} u_2^{4-j_2} \qquad i \in [1:3] \qquad (10.52)$$

The geometric form for arbitrary m and n is

$$x_i(u_1, u_2, \ldots, u_n) = \sum_{j_1=1}^{4} \sum_{j_2=1}^{4} \cdots \sum_{j_n=1}^{4} F_{j_1}(u_1) F_{j_2}(u_2) \cdots F_{j_n}(u_n) b_{ij_1j_2\ldots j_n} \quad i \in [1:m] \qquad (10.53)$$

The F-terms are Hermite basis functions. For the rth F-term, F_{j_r}, we have

$$\begin{aligned} j_r &= 1, \quad F_1(u_r) = 2u_r^3 - 3u_r^2 + 1 \\ j_r &= 2, \quad F_2(u_r) = -2u_r^3 + 3u_r^2 \\ j_r &= 3, \quad F_3(u_r) = u_r^3 - 2u_r^2 + u_r \\ j_r &= 4, \quad F_4(u_r) = u_r^3 - u_r^2 \end{aligned} \qquad (10.54)$$

The multisubscripted b array requires more care to interpret. If we use the following rules correctly, we can construct a correct and unambiguous set of geometric coefficients. The subscript i identifies the specific boundary on which the preceding product of basis functions is to operate. It also identifies a sequence of n operators applied to p_i, evaluated at the previously determined boundary. Table 10.1 is a summary of the information associated with the four values of the j_rth subscript.

The following examples demonstrate how to use this table:

Example 1. $n = 1$, cubic Hermite curves

$$b_{i2} = [1]x_i(1) = x_i(1)$$

$$b_{i3} = \left\{\frac{\partial}{\partial u_1}\right\} x_i(0) = \frac{\partial x_i(0)}{\partial u_i}$$

Table 10.1 Subscript interpretation

j_r	Boundary	Operator
$j_r = 1$	$u_r = 0$	1
$j_r = 2$	$u_r = 1$	1
$j_r = 3$	$u_r = 0$	$\frac{\partial}{\partial u_r}$
$j_r = 4$	$u_r = 1$	$\frac{\partial}{\partial u_r}$

Example 2. $n = 2$, bicubic Hermite patches

$$b_{i14} = [1]\left\{\frac{\partial}{\partial u_2}\right\}x_i(0,1) = \frac{\partial x_i(0,1)}{\partial u_2}$$

$$b_{i34} = \left\{\frac{\partial}{\partial u_1}\right\}\left\{\frac{\partial}{\partial u_2}\right\}x_i(0,1) = \frac{\partial^2 x_i(0,1)}{\partial u_1 \partial u_2}$$

Example 3. $n = 3$, tricubic Hermite solids

$$b_{i124} = [1][1]\left\{\frac{\partial}{\partial u_3}\right\}x_i(0,1,1) = \frac{\partial x_i(0,1,1)}{\partial u_3}$$

$$b_{i444} = \left\{\frac{\partial}{\partial u_1}\right\}\left\{\frac{\partial}{\partial u_2}\right\}\left\{\frac{\partial}{\partial u_3}\right\}x_i(1,1,1) = \frac{\partial^3 x_i(1,1,1)}{\partial u_1 \partial u_2 \partial u_3}$$

$$b_{i413} = \left\{\frac{\partial}{\partial u_1}\right\}[1]\left\{\frac{\partial}{\partial u_3}\right\}x_i(1,0,0) = \frac{\partial^2 x_i(1,0,0)}{\partial u_1 \partial u_3}$$

The braces [ ] indicate and separate the operators. The operations do not imply multiplication except in the case of [1].

Finally, the equation for the transformation from geometric to algebraic coefficients is

$$a_{ik_1k_2\ldots k_n} = \sum_{j1=1}^{4}\sum_{j2=1}^{4}\cdots\sum_{jn=1}^{4} m_{k_1j_1}m_{k_2j_2}\cdots m_{k_nj_n}b_{ij_1j_2\ldots j_n} \qquad i \in [1:m] \quad (10.55)$$

It is simple and quite straightforward to duplicate the above procedures for Bézier and B-Spline basis geometry.

10.5 INSTANCES AND PARAMETERIZED SOLIDS

A direct way of defining a new shape is as a simple linear transformation of an existing one. Let us consider a unit cube. There are several ways of trans-

forming it into a new shape by using scaling operators. Equal scaling of all three-dimensional components creates new different-size cubes. Differential scaling creates a variety of rectangular solids. Each new cube or rectangular solid is a particular instance of the initial cube.

Figure 10.14 shows examples of how simple scaling transformations create an unlimited variety of specific instances of an original shape. Notice that such transformations affect the geometry but not the topology of a shape. The shapes in this figure happen to be regular convex shapes. The original, or initial, shape is often called the *primitive* shape, and the result of its transformation produces an *instance*.

Figure 10.14 Instances.

Instancing a primitive shape is not limited to such simple primitives; for example, note the shape in Figure 10.15. We can transform this Z section by equal or differential scaling into an infinite number of instances. But, as specified here, there are certain built-in restrictions: The widths of the top and bottom flanges will always be equal, and the thickness of the section will always be one-tenth the height. And look at what happens to the circular arcs defined by the bend radii: They become elliptic arcs. This is not a fatal flaw when instancing, but restricting shape change to three orthogonal coordinate scale factors limits the prospects of a modeling system based solely on these transformations. Using fully parameterized shapes removes this limitation.

A few key dimensions are usually sufficient to define the shapes of simple objects. If each dimension is an independent variable, then we can produce a particular shape within a class of objects by specifying the key dimensions or parameters. Geometric-modeling algorithms then use these parameters to compute a more complete mathematical representation. In Figure 10.16, we see the same basic shape as that in Figure 10.15 but with much greater representational power.

Figure 10.15 *Instances of a Z section.*

Figure 10.16 Parameterized shape.

It is not difficult to verify the validity of data specifying a model. For the Z section in Figure 10.16, for example, we can apply the following restrictions and easily check them: $a, b, h, l, t > 0$, $b \leq a$, $a > 2t$, and $h > 4t$.

We can develop parameterized shapes with variable topologies, such as the n-celled structure in Figure 10.17, but most parameterized-shape procedures are restricted to a single topology.

A related scheme, called *group technology*, came into use in concert with certain computer-aided manufacturing techniques to encourage standardization in part design and production. The central thesis and main advantage of group technology is that many manufactured parts can be grouped into classes or families of similar shapes, where individual members are distin-

n = number of cells
t = wall thickness
a,b = cell dimensions
l = length

Figure 10.17 Parameterized shape of variable topology.

guished by a few parameters (key dimensions). A single family of shapes is a generic primitive, and individual members are primitive instances. Figure 10.18 shows examples of generic primitives.

The instances are not usually combined to form more complex shapes as they are in constructive solid geometry, although no theoretical barriers prevent it. For example, we could envision a modeling system that converts any primitive instance into a boundary representation (probably an envelope of surfaces), evaluates it to Boolean combinations of instances, and forms more complex shapes.

Parameterized-shape or group-technology models are easy to validate and use. They are unquestionably concise; however, the number of useful generic primitives, though large, is limited. Modeling systems built only on this type of representation are highly specialized; a large repertoire of generic primitives is required for them to have wide application.

10.6 SWEEP SOLIDS

Sweep representations are based on the notion of moving a curve, surface, or solid along some path. The locus of points generated by this process defines a new two- or three-dimensional object. Sweep representations for modeling solids are simple to understand and execute, and they offer opportunities for developing new methods. The relatively constrained geometry of a sweep shape means that only a small data set is needed to specify the shape. Procedures defining a particular type of sweep flesh out and develop the full pointwise shape representation. They are mathematically concise and often form the data and analytical framework for subsequent interpretation as a variety of analytical surfaces or solids, which, in turn, may be used in boundary or CSG representations. For more advanced work in this area that includes analysis and modeling of deformed swept volumes using sweep differential equation techniques, see the work of Blackmore, Leu, and Shih (1994), among others.

Figure 10.18 Typical generic primitives.

We define a *translational sweep* or *extrusion* by moving a planar curve or planar shape along a straight line normal to the plane of the curve, the former generating a surface and the latter a solid. We define a *rotational sweep* by rotating a planar curve or shape with finite area about an axis. Although this is a spatial rotation, the axis is usually in the plane of the figure—at least for the simplest and most common applications. A somewhat less-restrictive definition of a translation surface is produced by sweeping one curve along another curve, where the generating curve remains parallel to itself if it is a plane curve.

A *general sweep* is one whose generating shape follows some arbitrary curved path, and which itself may change size, shape, and orientation. Thus, for solid modeling we need two ingredients: an object to move and a path to move it along. The sweeping object must be a curve, surface, or solid, and the path must be analytically definable. The term *generator* denotes the sweeping object, and the term *director* denotes the path. Unfortunately, the word *sweep* is used as a noun, verb, or adjective. Fortunately, the meaning is usually clear from the context.

When does a sweep become so generalized that it is no longer a useful, intuitive concept in application? This is a difficult question with no easy answer. However, we cannot go wrong if we insist on the presence of a generator shape, a director path, and rules controlling the orientation of the generator at each point on the director.

Sweep shapes are important in geometric modeling because they accurately represent a large class of engineering and manufacturing objects and processes, and in contemporary modeling systems they prove to be practical and efficient for modeling constant-cross-section mechanical parts. We can use them to detect potential interference between parts of mechanisms; for example, a moving object A collides with a fixed object B if the volume swept by A intersects B. A related use is in simulating and analyzing material-removal operations in manufacturing; for example, in studying the cylindrical and spherical geometry of the volumes swept by a ball-shaped cutter moving along a predefined path intersecting and machining the raw stock for a part, where the intersection volume represents material removed from the part.

Figure 10.19 shows examples of sweep representations. The two principal types of trajectories are depicted—*translation* and *rotation*. Note that the director curve is not necessarily an element of the swept object. Also note that the director of a rotational sweep amounts to an algorithm to move each point in the generator along a circular arc in a plane perpendicular to the axis of rotation and with a radius defined as the perpendicular distance from the point to the axis. In all cases, the shape of the object, or generator, being swept along does not change. Later we will investigate more general, nonlinear sweep representations.

Figure 10.19 Examples of sweep shapes.

There are several obvious and some not-so-obvious ways of creating dimensionally nonhomogeneous objects. In Figure 10.20a, the translational sweep of a curve creating a surface also creates two dangling edges. In Figure 10.20b, two two-dimensional regions are connected by a one-dimensional structure. In Figures 10.20c and 10.20d, creating solids using invalid or nonhomogeneous generators results in dimensionally nonhomogeneous solids and ambiguities. The rotational sweep of a generator passing through the axis of rotation, as in Figures 10.20e and 10.20f, produces a surface or solid with a singularity. These conditions produce unacceptable results for most applications, but sometimes the results are expected and

Figure 10.20 Dimensionally nonhomogeneous sweep representations.

even desired. This is true for modeling systems requiring greater degrees of freedom—computer art, for example. By following criteria suggested in this figure and incorporating them in the design of sweep-representation generating algorithms, we can be sure to create dimensionally homogeneous models.

Translation and Generalized Sweeps

Lossing and Eshleman (1974) developed a powerful technique for representing constant-cross-section objects. Their approach emphasizes proce-

dures that minimize data storage requirements. They use a six-component trajectory and generator orientation curve called a *position and direction (PD) curve*. A closed planar curve defining the cross section of a mechanical part is translated along an axis to form the outline surface of a raw-stock model. This model is then trimmed by limiting planes to produce the bounding surfaces of the part. Figure 10.21 illustrates these features. If the finished part followed a curved axis, a nonlinear transformation on the model would describe this representation.

Lossing and Eshleman are able to define an almost unlimited variety of swept solids using their technique. They define a PD curve as a general form of a six-component curve, usually a cubic Hermite curve that continuously specifies position and an associated direction. The first three components define a continuous parametric cubic equation of position in three-dimensional space. The second three components define a corresponding continuous parametric cubic equation for an associated direction vector. A common parametric variable associates the direction vector with a specific position on the curve. The PD curve in Figure 10.22 defines a curved and twisting coordinate system.

We construct a local orthogonal system at $\mathbf{p}_1$ as follows: Compute $\mathbf{p}_i''$, the tangent vector. From $\mathbf{p}_i''$ and $\mathbf{d}_i$ (remember, $\mathbf{d}_i$ is given by another parametric equation), we find the orthogonal unit vectors $\mathbf{l}$, $\mathbf{m}$, and $\mathbf{n}$:

Figure 10.21 Outline surface of a constant-cross-section solid.

Figure 10.22 Characteristics of a PD curve.

$$\mathbf{l} = \frac{\mathbf{p}_i''}{|\mathbf{p}_i''|}$$

$$\mathbf{m} = \frac{\mathbf{d}_i \times \mathbf{p}_i''}{|\mathbf{d}_i \times \mathbf{p}_i''|} \tag{10.56}$$

$$\mathbf{n} = \mathbf{l} \times \mathbf{m}$$

The axes $\mathbf{l}$ and $\mathbf{n}$ define a direction plane in which $\mathbf{d}_i$ is located. This reference plane changes as the tangent vector and $\mathbf{d}_i$ change continuously along the PD curve. A PD curve defines continuous transformations for points on an outline curve or generator.

We can use a PD curve to produce generalized outline surfaces. Figure 10.23 illustrates a constant-cross-section part that is not only curved but also twists. Figure 10.24 illustrates how to coordinate the two curves comprising a PD curve through a common value of the parametric variable and how to extract the elements. Losing and Eshleman also point out that two or more cross-section curves can be used with one PD curve (for example, a pipe or tube with an inner and outer cross-section curve). Multiple cross sections with multiple PD curves can be used to generate a more complex or variable cross section. Positive and negative generator curves are used to create complex cross sections with high variability in the axial direction. Multiple PD

Figure 10.23 A constant-cross-section part that curves and twists.

curves and associated generator curves create, in effect, half-spaces with directed normals and can be used to add or subtract "material" to a raw-stock model. Some of these modeling procedures must depend on the user to verify model validity.

A six-component curve trajectory for controlling the movement of a generator curve has obvious natural extensions. For example, three more components may be added to yield scale factors to apply to the generator curve, differentially expanding or contracting it, and coordinating it with the motions imposed by the PD curve. This introduces nonrigid sweep representations, a field still in its formative stages, which we will not discuss here.

Choi and Lee (1990) describe sweep-modeling schemes based on conventional engineering drawing techniques. They address parallel and rotational sweeping, spined sweeping, and synchronized sweeping, and show that a

Figure 10.24 Components of a PD curve.

variety of surface shapes can be defined by specifying a set of section and control curves, with appropriate transformation and blending rules. Martin and Stephenson (1990) show how envelope theory from differential geometry can be used to find the volumes swept out by the surfaces of a solid body.

Rotational Sweeps

One way to generate a surface of revolution is by revolving a plane curve around an axis line in its plane (see Figure 10.25). The plane curve is called the *profile curve*, and, in its various positions around the axis, it creates meridians. The circles created by each point on this curve are called *parallels*. For the simplest case, we let the z axis be the axis of rotation and define the curve $\mathbf{p}(u) = \mathbf{x}(u) + \mathbf{z}(u)$ in the x, z plane. Then the surface of revolution has the equation

$$\mathbf{p}(u,\theta) = x(u)\cos\theta + x(u)\sin\theta + z(u) \tag{10.57}$$

Of course, the profile curve might be a general curve, elliptic segment, circular arc, and so on.

Figure 10.25 Surface of revolution.

For a more general surface of revolution, we consider the situation shown in Figure 10.26. The given or initial elements are a cubic Hermite curve defined by its geometric coefficients, a unit vector **a** defining the direction of the axis of revolution, a vector **b** locating a point through which the axis passes, and the angle ψ through which the curve rotates to sweep out the surface. We would like to find a bicubic Hermite patch that describes such a surface in terms of these elements. First, we must determine the scalars k_0 and k_1 and the vectors $\mathbf{r}_0$ and $\mathbf{r}_1$. We do this by solving the following sets of vector equations:

$$\mathbf{b} + k_0\mathbf{a} + \mathbf{r}_0 = \mathbf{p}_0 \qquad \text{where} \qquad \mathbf{a} \cdot \mathbf{r}_0 = 0 \tag{10.58}$$

and

$$\mathbf{b} + k_1\mathbf{a} + \mathbf{r}_1 = \mathbf{p}_1 \qquad \text{where} \qquad \mathbf{a} \cdot \mathbf{r}_1 = 0 \tag{10.59}$$

The condition asserted by $\mathbf{a} \cdot \mathbf{r}_0 = 0$ makes **a** and $\mathbf{r}_0$ mutually perpendicular; similarly for **a** and $\mathbf{r}_1$.

From Equation (10.58), we obtain

$$k_0 = \mathbf{a} \cdot (\mathbf{p}_0 - \mathbf{b}) \tag{10.60}$$

$$\mathbf{r}_0 = \mathbf{p}_0 - k_0\mathbf{a} - \mathbf{b} \tag{10.61}$$

From Equation (10.59), we obtain

$$k_1 = \mathbf{a} \cdot (\mathbf{p}_1 - \mathbf{b}) \tag{10.62}$$

$$\mathbf{r}_1 = \mathbf{p}_1 - k_1\mathbf{a} - \mathbf{b} \tag{10.63}$$

Figure 10.26 Another surface of revolution.

Now we can proceed to find the other two corner points $\mathbf{p}'_0$ and $\mathbf{p}'_1$. Clearly, $\mathbf{p}'_0 = \mathbf{b} + k_0\mathbf{a} + \mathbf{r}'_0$ and $\mathbf{p}'_1 = \mathbf{b} + k_1\mathbf{a} + \mathbf{r}'_1$. Thus, we must determine $\mathbf{r}'_0$ and $\mathbf{r}'_1$ (see Figure 10.27).

The unit tangents to the circular arc at $\mathbf{p}_0$ and $\mathbf{p}'_0$ are denoted by $\mathbf{t}_0$ and $\mathbf{t}'_0$, respectively. We will have to determine these vectors to find the final tangent vectors at these points. The unit vector in the direction of the vector product of $\mathbf{r}_0$ and $\mathbf{a}$ produces $\mathbf{t}_0$, so that

$$\mathbf{t}_0 = \frac{\mathbf{r}_0 \times \mathbf{a}}{|\mathbf{r}_0 \times \mathbf{a}|} \tag{10.64}$$

We can define the vector $\mathbf{r}'_0$ as the sum of two components, one in the direction of $\mathbf{r}_0$ and one in the direction of $\mathbf{t}_0$. We begin by observing that the magnitude of $\mathbf{r}'_0$ equals the magnitude of $\mathbf{r}_0$. Applying elementary trigonometry and vector arithmetic, we obtain

$$\mathbf{r}'_0 = |\mathbf{r}_0| \frac{\mathbf{r}_0}{|\mathbf{r}_0|} \cos\psi + |\mathbf{r}_0| \frac{\mathbf{r}_0 \times \mathbf{a}}{|\mathbf{r}_0 \times \mathbf{a}|} \sin\psi \tag{10.65}$$

or, simplified,

$$\mathbf{r}'_0 = \mathbf{r}_0 \cos\psi + |\mathbf{r}_0| \frac{\mathbf{r}_0 \times \mathbf{a}}{|\mathbf{r}_0 \times \mathbf{a}|} \sin\psi \tag{10.66}$$

Figure 10.27 Circumferential tangent vectors of a surface of revolution.

We find $\mathbf{r}'_1$ in a similar way:

$$\mathbf{r}'_1 = \mathbf{r}_1 \cos \psi + |\mathbf{r}_1| \frac{\mathbf{r}_1 \times \mathbf{a}}{|\mathbf{r}_1 \times \mathbf{a}|} \sin \psi \tag{10.67}$$

We can also define the unit tangent vector $\mathbf{t}'_0$ as the sum of two components in the directions of $\mathbf{t}_0$ and $\mathbf{r}_0$:

$$\mathbf{t}'_0 = \mathbf{t}_0 \cos \psi - \frac{\mathbf{r}_0}{|\mathbf{r}_0|} \sin \psi \tag{10.68}$$

or

$$\mathbf{t}'_0 = \frac{\mathbf{r}_0 \times \mathbf{a}}{|\mathbf{r}_0 \times \mathbf{a}|} \cos \psi - \frac{\mathbf{r}_0}{|\mathbf{r}_0|} \sin \psi \tag{10.69}$$

We know that $\mathbf{t}_1 = \mathbf{t}_0$ and $\mathbf{t}'_1 = \mathbf{t}'_0$.

We must now find the magnitudes of these tangent vectors. Recall how we developed the tangent vectors for circular arcs in Chapter 3 (Section 3.8). We used a point $\mathbf{p}_2$ lying on the intersection of the tangent lines at the two endpoints of the arc. We will do the same now. We find $\mathbf{p}_2$ by computing the intersection of the two straight lines $\mathbf{p}_0 + m\mathbf{t}_0$ and $\mathbf{p}'_0 + m\mathbf{t}'_0$. By symmetry, we see that $n = -m$. Furthermore,

$$\mathbf{p}_0 + m\mathbf{t}_0 = \mathbf{p}'_0 - m\mathbf{t}'_0$$

and

$$m = \frac{|\mathbf{p}'_0 - \mathbf{p}_0|}{|\mathbf{t}_0 + \mathbf{t}'_0|}$$

From these, we obtain

$$\mathbf{p}_2 = \mathbf{p}_0 + \frac{|\mathbf{p}'_0 - \mathbf{p}_0|}{|\mathbf{t}_0 + \mathbf{t}'_0|} \mathbf{t}_0$$

Using equations from Section 3.8 with appropriate substitutions, we find

$$\mathbf{p}^w_{00} = 4\,|\mathbf{r}_0| \left[\frac{1 - \cos(\psi/2)}{\sin(\psi/2)} \right] \left[\frac{\mathbf{p}_2 - \mathbf{p}_0}{|\mathbf{p}_2 - \mathbf{p}_0|} \right] \tag{10.70}$$

and

$$\mathbf{p}^w_{01} = 4\,|\mathbf{r}_0| \left[\frac{1 - \cos(\psi/2)}{\sin(\psi/2)} \right] \left[\frac{\mathbf{p}'_0 - \mathbf{p}_2}{|\mathbf{p}'_0 - \mathbf{p}_2|} \right] \tag{10.71}$$

We use this same procedure to find $\mathbf{p}^w_{10}$ and $\mathbf{p}^w_{11}$ at the other end of the patch. Notice that we have changed the nomenclature: We now denote point $\mathbf{p}'_0$ as $\mathbf{p}_{01}$, and, of course, $\mathbf{p}_0$ of the initial curve is $\mathbf{p}_{00}$ on the patch. It should help to keep in mind that we are creating a surface patch by sweeping a curve through an arc around an axis.

Next, we compute the tangent vector $\mathbf{p}_{01}^{u}$, which is, of course, the result of rotating $\mathbf{p}_{0}^{u}$ through the angle ψ around $\mathbf{a}$. First we see that $|\mathbf{p}_{01}^{u}| = |\mathbf{p}_{0}^{u}|$ (see Figure 10.28). Also, both vectors have identical projections on the unit vector $\mathbf{a}$, so $\mathbf{p}_{01}^{u} \cdot \mathbf{a} = \mathbf{p}_{0}^{u} \cdot \mathbf{a}$. We use the symmetry of the construction and realize that the projection of $\mathbf{p}_{01}{}^{u}$ onto the unit vector in the direction of $\mathbf{r}_0'$ has the same magnitude as the projection of $\mathbf{p}_{0}^{u}$ onto the unit vector in the direction of $\mathbf{r}_0$. Now we are ready to define $\mathbf{p}_{01}^{u}$ as the sum of its components along $\mathbf{a}$ and $\mathbf{r}_0'$

$$\mathbf{p}_{01}^{u} = (\mathbf{p}_{0}^{u} \cdot \mathbf{a})\mathbf{a} + \left(\mathbf{p}_{0}^{u} \cdot \frac{\mathbf{r}_0}{|\mathbf{r}_0|}\right)\left(\frac{\mathbf{r}_0'}{|\mathbf{r}_0'|}\right) \tag{10.72}$$

We find the tangent vector at the other end $\mathbf{p}_{11}^{u}$ in the same way. All of the required elements of the patch $\mathbf{B}$ matrix are determined except the twist vectors. Hint: What effect do the auxiliary curves have on the interior of the surfaces of revolution with $\mathbf{p}^{uw}(u, w) = 0$ or with $\mathbf{p}^{uw}(u, w) \neq 0$?

10.7 CONTROLLED DEFORMATION SOLIDS

We can use nonlinear transformations to deform shapes in a controlled way, thereby creating new shapes. To deform a curve $\mathbf{p}(u)$, we define a deformation of the x axis by $\mathbf{r}(t)$ (see Figure 10.29), and define a relationship between the parametric variables u and t:

Figure 10.28 Axial tangent vectors of a surface of revolution.

Figure 10.29 Axis deformation.

$$t = t(u)$$

Then the deformed curve is

$$\mathbf{p}' = \mathbf{r}[t(u)] + p_y\mathbf{m} \tag{10.73}$$

where $\mathbf{l} = \mathbf{r}'$, $\mathbf{m} = \hat{\mathbf{z}} \times \mathbf{l}$, and $\hat{\mathbf{z}}$ is the unit vector in the z direction.

Another way to deform a shape is by a simple basis deformation (see Figure 10.30). We define the initial shape in the coordinate system with $\mathbf{m}$ and $\mathbf{n}$ unit orthogonal basis vectors. We map the deformed shape with respect to a transformed basis $\mathbf{m}'$ and $\mathbf{n}'$, which is no longer orthogonal, and express it as

$$\mathbf{p}'_i = m_i\mathbf{m}' + n\mathbf{n}' \tag{10.74}$$

Axial Deformations

Axial deformations are a generalization of the methods discussed in the preceding sections. This technique controls shaping operations, such as scaling, bending, twisting, and stretching, by reference to some convenient axis in which we induce the desired deformations subsequently to be passed on to the model. Lazarus, Coquillart, and Jancéne (1994) have worked in this area, drawing on earlier work by Borrel and Bechmann (1991), Welch and Witkin (1992), Barr (1984), and others. Their approach to the axial-deformation technique makes use of an arbitrary, but convenient, three-dimensional axis (straight or curved) as the basis for deforming existing models. If the model itself is represented by a set of control points, as is the case for Bézier and B-

Figure 10.30 Simple basis deformation.

Spline curves and surfaces, then axis deformations are simply passed on to these points. Each control point is "attached" to a point on the axis and located within a local coordinate system defined at that point. This may be the Frenet frame, used as in the work of Lossing and Eshleman (1974), or a frame such as that proposed by Bishop (1975) or Klok (1986). Next, the axis is deformed to achieve the desired results in the model. If the axis itself is defined by control points, then the points may be moved to produce the axis deformation. Finally, transformed model control points are computed and used to represent and compute the deformed model.

Bivariate Deformation

We can create a deformed shape by defining it in an embedding space. Deformations of this space are passed on to the object. For example, we define a Hermite, Bézier, or B-Spline curve $\mathbf{c}(t)$ and embed it in a normalized parametric plane defining the domain of the parametric variables u,w (see Figure 10.31). We define a bivariate surface, $\mathbf{p}(u, w)$ in the x, y plane or in x, y, z space. Then we map the curve

$$\mathbf{p}_c = \mathbf{p}[u(t), w(t)] \tag{10.75}$$

Trivariate Deformation

Similarly, in three dimensions we can define a curve $\mathbf{c}(t)$ and embed it in a normalized parametric space u, v, w (see Figure 10.32). We define a trivariate solid $\mathbf{p}(u,v,w)$ and map the curve according to

$$\mathbf{p}_c = \mathbf{p}[u(t), v(t), w(t)] \tag{10.76}$$

(a) In parameter space

(b) In 2D

(c) In 3D

Figure 10.31 Shapes via deformed bivariate embedding spaces.

Figure 10.32 Shapes via deformed trivariate embedding spaces.

Using an analogous process we can embed and deform surfaces. This process is sometimes called *volume deformation*.

Deformable Surfaces

Celniker and Welch (1992) describe a method for sculpting B-Spline surfaces that leaves a set of local geometric constraints invariant, such as interpolated points and curves. Their method seeks to achieve a fair shape by minimizing a global energy function. They demonstrate how to constrain any parametric point to a fixed location; a parametric curve embedded in the surface to maintain fixed contours in the world coordinate system; and how to constrain surface normals along a parametric curve on the surface.

Related work in this area includes Fowler and Bartels (1993), whose method allows direct manipulation of geometric properties and relationships to control surface shape, rather than to control vertices or deformation lattices. Also see Bartels and Beatty (1989), Clark (1976), Coquillart (1990), Welch, Gleicher, and Witkin (1991), Kramer (1991), Light and Gossard (1982), Serano and Gossard (1987), and Witkin, Fleischer and Barr (1987), among others.

This page intentionally left blank

11

COMPLEX MODEL CONSTRUCTION

In geometric modeling, we combine simple shapes to construct complex models. The techniques we use must produce valid models. For example, a completed model must be dimensionally homogeneous. If it is a three-dimensional solid, it must have no dangling edges or surfaces. Model connectivity and homogeneity are topological properties, so consideration of topology is also an important part of the modeling process.

There are two approaches to representing complex models. One approach uses an implicit construction technique called *constructive solid geometry* (*CSG*), and the other uses an explicit boundary-based technique called *boundary representation* (*b-rep*). The CSG scheme defines complex solids as Boolean combinations of simpler solids. The complete representation is sometimes referred to as a *CSG tree,* because it uses a binary tree whose terminal nodes are simple solids and whose nonterminal nodes are so-called *regularized* Boolean combining operations. The data structure of a b-rep is a graph associating the curves and surfaces that comprise the boundary of a solid.

In the early chapters, we discussed the basic elements of a geometric model: The parametric and implicit curves, surfaces, and solids. Appendix C and Mortenson (1995) discuss the geometric transformations of these elements. In the previous chapter we discussed how to generate simple solid shapes using instances of parameterized shapes, sweeps, and deformations. All of these form the foundation on which we build the complex models that are the subject of this chapter. Here we will discuss topology, graphs, Boolean models, boundary representations, and space-partitioning schemes of complex solid models.

11.1 TOPOLOGY OF MODELS

Complex models require considerable attention to their topology. We must understand how simple elements are connected to form the complex model and how its topology is preserved when subjected to a variety of transformations. Topological properties are not metrical, but concern such things as connectivity and dimensional continuity. The properties of geometric shapes that are invariant under transformations that stretch, bend, twist, or compress a figure, without tearing, puncturing, or inducing self-intersection, are *topological properties.* The property of being an open or closed curve or surface is a *topological invariant.* One-sidedness and two-sidedness are topologically invariant properties of surfaces. Lines, parabolas, and the branches of a hyperbola belong to the class of topologically equivalent figures called *simple arcs.* We will address these and other topics here, including the topology of *closed paths, piecewise flat surfaces,* and *closed curved surfaces.*

The topology of a closed path is related to problems in geometric modeling, so we will develop the *closed-path theorem,* review the effect of *deformations* on the topology of curves and planes, present the *Jordan curve theorem,* and define the concept of *total curvature.* Hilbert and Cohn-Vossen (1952), Alexandroff (1961), and Abelson and Disessa (1981) wrote thoroughly and imaginatively on this subject. These authors are the primary sources for this discussion, and reference to their work is highly recommended. Abelson and Disessa, in particular, offer unique insights into fundamental topological principles relevant to geometric modeling.

The Topology of a Closed Path

The *closed-path theorem* states: The total turning along any closed path is an integer multiple of 360°. This integer, denoted N_R, is the *rotation number* of the path. It is an intrinsic property of the path and is independent of where the path starts or how it is oriented. Figure 11.1 shows several examples, including loops. For closed paths on a plane surface, simple polygons always appear to have total turning equal to ±360°, depending on the direction in which the path is traversed. Self-intersecting polygons and closed curved paths, however, always have total turning different from ±360°. There are two classes of paths: *simple* and *self-intersecting.* Self-intersecting polygons are *star polygons.*

The *simple-closed-path theorem* states that the total turning in a non-self-intersecting closed path is ±360° (clockwise or counterclockwise). In other words, the rotation number N_R of any simple closed path is ±1. Studying some examples of simple closed paths demonstrates the validity of this theorem, which, as simple and obvious as it appears, happens to be difficult to

Figure 11.1 Rotation number.

prove rigorously. The theorem implies that there is a relationship between the two properties of a closed path—the turning and the crossings or self-intersection points.

Curves defining closed paths that can be deformed into one another are curves that are topologically equivalent. Total turning is a topological invariant for closed paths, and any two closed paths that are topologically equivalent must have the same total turning. The converse is also true: If two paths have the same total turning, they can be deformed into one another. This was first proved in 1936 by H. Whitney and W. C. Graustein, who observed that two closed coplanar paths can be deformed into one another if, and only if, they have the same total turning.

The simple-closed-path theorem is more comprehensive than the closed-path theorem, and its proof is considerably more complicated. The simple-closed-path theorem is a link between local and global information. Since crossing points on a curve are nonlocal phenomena, there is no direct way for a traversal algorithm (that is, an algorithm proceeding stepwise around the path) to detect a crossing point. To do this requires looking at the entire curve at once. However, it is possible to determine total turning by local computation, and this theorem relates total turning to the existence of crossing points, a powerful capability.

A simple exercise demonstrates the link between local and global information contained in the theorem. We can walk around a closed path, perhaps as defined by a set of written instructions, carefully accumulating our total turning (taking clockwise as positive). If we complete the circuit and find on returning to the starting point that the total turning does not equal ±360°, then we can assert that somewhere the path must have at least one crossing point. Although we do not know where the crossing points are and we were unable to observe them while traversing the path, by applying the theorem we can determine that one or more crossing points must exist. Thus, the simple-closed-path theorem is an example of a powerful principle: It is often possible to determine global properties by accumulating local information. This theorem is useful in determining the validity of a model.

Deformation of Curves and Planes

To develop proofs of the simple-closed-path theorem and related theorems about the topology of simple closed curves, we turn our attention to the consequences of deformations of curves and planes. We must show that, given a simple closed path (a path with no crossing points), the total turning around the path is equal to ±360°. There are many paths for which this result is obvious; the simplest is the square. This suggests a strategy for proving the theorem by showing that any simple closed path can be deformed into a square. Since we already know that total turning is invariant under deformations, we can show that any simple closed path has the same total turning as a square—±360°—depending on the direction around the square. This reduces the problem of proving the simple-closed-path theorem to showing that any simple closed curve can be deformed into a square. In fact, we can go a bit further: Given any simple closed path, not only can the path itself be deformed into a square, but the entire plane can be deformed, pulled, and stretched, so that the path becomes a square. We can even consider the plane itself as being deformed into a curved surface.

Deformations of paths and planes are closely related. Imagine that the plane is an arbitrarily stretchable rubber sheet. Any kind of stretching or

shrinking that does not result in cutting or tearing is a valid deformation. If we draw a closed path on the rubber sheet plane, then any deformation of the plane results in a deformation of the path (see Figure 11.2). Next, imagine that the flat rubber sheet is arbitrarily stretched and distorted but must remain in the initial plane. Clearly, straight-line segments may curve under such a deformation. Many of the changes that are legitimate for the deformation of curves, such as the overlap phenomenon, do not and cannot happen with a deformation in the plane. We observe that crossing points are neither created nor destroyed during plane deformation.

Plane deformations are more conservative than *path deformations*. Every plane deformation is a path deformation, but a path deformation that introduces crossovers is too violent to be a plane deformation. The mathematical term for a path deformation is *regular homotopy*, while a rubber sheet deformation is an *ambient isotropy*. An important observation to make here is that any simple closed path can be deformed into a square, and, moreover, this can be done with a conservative deformation; that is, a plane deformation.

In summary, we can state the deformation theorem for simple closed curves as follows: For any simple closed curve in the plane, there is a rubber sheet deformation of the plane that reduces the curve to a square.

Figure 11.2 Rubber sheet deformation.

The Jordan Curve Theorem

Another result of the topology of simple closed paths that also follows from the deformation theorem is the *Jordan curve theorem*, which states: Any simple closed curve in the plane divides the plane into exactly two regions, an inside and an outside. Such properties as *dividing the plane into two regions* and *having an inside and an outside* are invariant under rubber sheet deformations of the plane. If the deformed curve has these properties, then so too must the original nondeformed curve.

The Jordan curve theorem probably seems obvious, but simple closed curves can be arbitrarily complex and convoluted. We note that the theorem is not true if we consider curves on surfaces other than the plane. For example, a simple closed curve can be drawn on a torus, yet it may not divide the torus into two distinct regions.

Not only does a closed curve have an inside and outside, but the inside itself is deformable, in the rubber sheet sense, into the interior of a circle in the plane. A region that can be deformed in this way is a *topological disk*. A topological disk has no holes or isolated points in it. In general, deforming a rubber sheet region does not require keeping the region a part of the flat plane (that is, the region may be planar or nonplanar). Figure 11.3 shows a curve deformed into a rectilinear polygon, where we see that the total turning is reduced to summing clockwise and counterclockwise 90° turns or angles.

Figure 11.3 Approximating a curve by a rectilinear path on a grid.

Angle Excess

The concept mathematicians call *angle excess,* or simply *excess,* is by definition the turning that some reference pointer undergoes when carried around a closed path. Excess is a rather general concept; it is an angle associated with a closed path on a curved surface. We can restate the closed-path theorem so that it holds for simple closed paths on arbitrary surfaces:

$$T + E = 360° \tag{11.1}$$

where T = total turning along the path and E = excess along the path.

Figure 11.4 illustrates the angle excess associated with a closed path on a sphere. A trip along this path might start at the equator, proceed to the pole, then return to the equator along a different meridian and continue along the equator to the starting point. Notice that the reference pointer must always be transported parallel to itself relative to the surface in which the path lies. For a fixed path, the excess is the same no matter where the trip begins.

Angle excess is additive, and we find that the following theorems apply.

1. If a triangle is subdivided into two subtriangles, then the excess of the triangle is the sum of the excesses of the pieces.
2. The excess of any polygon is the sum of the areas of the pieces in any polygonal subdivision.

Figure 11.4 Angle excess associated with a closed path on a sphere.

3. For any topological disk on an arbitrary surface, the angle excess around the boundary is equal to the total curvature of the interior.

Total Curvature and Curvature Density

We can show that the total curvature for a sphere is 4π and the angle excess is proportional to the area, or $E = kA$. Thus, for any polygon on a sphere of radius r, $E = k$, where $k = 1/r^2$ (if E is measured in radians). For a simple closed path on the surface of a sphere, total turning in radians is expressed as $T = 2\pi - A/r^2$, where A is the area enclosed by the path and r is the radius of the sphere.

In general, for an arbitrary surface, k is measurement taken at any point on the surface. The value of k at a point is the excess per unit area of a small patch of surface containing the point. We call k the *curvature density* of the surface at a point. We use the term *density* because k is "something per unit area" and the term *curvature density* rather than *excess density* because, although k is measured using excess, it tells us how curved the surface is at the point. To approximate a small patch of an arbitrary surface by a small patch of some surface we know very well, we observe that within a small enough area, almost any surface will appear planar.

We can make an even better approximation by using a small piece of a sphere. We will choose the approximating sphere to have the same excess per unit area as the patch of a surface. Therefore, if k does not change radically in the small patch, then all of the geometry there—angles, total turning, and so on—is very close to the geometry on a sphere whose radius is determined by $k = 1/r^2$. The smaller the radius of the approximating sphere (that is, the greater the curvature density), the more curved is the surface. A football, for example, is not very curved in the middle; k is small there. But at the pointed ends, the football is curved as much as a sphere of small radius. From the middle to the pointed ends, k gradually increases.

Curvature density is a local quantity that is measured in the vicinity of a point on an arbitrary surface. A global version of curvature density is called *total curvature*. We start with a region of an arbitrary surface and divide it into polygonal pieces. For each polygon, we compute the excess and sum the excesses for all the pieces. The result is the total curvature K of the region. Of course, in order for this definition to make sense, we must be sure that if we subdivide a region into polygons in two different ways, the sum of the excess of the pieces is the same in both cases.

If the initial region is itself a polygon, the additivity theorem implies that K is precisely equal to the excess around the boundary of the polygon. Therefore, K is a kind of excess over the region, with the condition that it is true for any region on the surface, not just for polygons.

Spheres, Tori, and Handles

Let us compute K for a region that has no boundary at all—the *sphere*. We divide a sphere into two pieces; for example, the northern and southern hemispheres. Each of them is bounded by the equator, which we know has excess 2π, yielding 4π for the total curvature of the spheres. Observe that the curvature density k of a sphere depends on the radius, but the total curvature K is the same for all spheres.

It turns out that the total curvature K is a topological invariant for closed surfaces. For surfaces with a boundary, the total curvature is unchanged by deformations that do not affect the vicinity of the boundary of the surface.

Any torus has zero total curvature. We are not asserting that a torus is flat like a cylinder or a plane, which would imply that the curvature density is zero everywhere. Instead, we observe that any torus has just as much negative curvature as positive curvature.

We now know the total curvature of two kinds of closed surfaces, spheres and tori. A sphere is not topologically the same as a torus, yet they are related. The *torus* can be described as a sphere with a handle attached. The process of handle attaching, though not a deformation, is a practical way of making new closed surfaces from old ones because it is easy to keep track of how total curvature changes in the process. Let us explore this process a little more.

We start with a sphere and flatten out two regions on it. From each region we cut out a flat disk and attach a *handle* in the holes left by the disks. The handle is topologically a cylinder, but the edges of the cylinder must be flared out to blend with the flat regions around the disks. The result is topologically a torus (see Figure 11.5). In other words,

$$\text{sphere} - 2\ \text{disks} + \text{handle} = \text{torus} \tag{11.2}$$

Next, we compute the total curvature on each side of the equation. The sphere has $K = 4\pi$; the disks were flat and thus have $K = 0$. The torus, as we

Figure 11.5 Sphere with a handle.

just asserted, also has $K = 0$. Then, for the equation to be balanced, the handle must have total curvature $= -4\pi$.

With this process of flattening, cutting, and gluing, we can attach a handle to any surface. We now know the curvature in the handle, and we see that it always decreases the total curvature by 4π. We find that the total curvature of a two-holed torus is the same as that of a torus with a handle attached or a sphere with two handles attached. This is expressed as

$$K_{\text{2-holed torus}} = 4\pi - 4\pi - 4\pi = -4\pi \tag{11.3}$$

In general, for a surface that is topologically the same as a sphere with g handles attached, total curvature is given by

$$K_{\text{sphere with } g \text{ handles}} = 4\pi(1 - g) \tag{11.4}$$

The important fact about spheres with handles is: Any closed surface in three-dimensional space is topologically equivalent to a sphere with some number of handles attached. For example, the surface in Figure 11.6a is equivalent to a sphere with six handles. What about the object in Figure 11.6b? (Hint: Topologically deform it as shown, then count handles.)

We see from Equation (11.4) that the total curvature of any closed surface in three-dimensional space is an integer multiple of 4π. This topological characteristic of closed surfaces is immediately applicable to the synthesis of solid models of complex objects, particularly boundary representations, and also as an analytic characteristic or determinant of topological type.

Figure 11.6 Topological equivalent of a sphere with: (*a*) six handles; (b) five handles.

Remember that total turning for a closed curve path is always an integer multiple of 2π, and we have now discovered an analogous property for closed surfaces in three-dimensional space—all that arbitrary denting and bending and all those excess angles must somehow combine to give precisely an integer multiple of 4π. Remarkably, in changing the total curvature of an ordinary closed surface, we must change it by a multiple of 4π or not at all.

There are closed surfaces that are too twisted to fit into three-dimensional space and, as a consequence, do not have a total curvature that is a multiple of 4π. As it turns out, any closed surface, including these twisted ones, must have total curvature equal to a multiple of 2π. This class of surfaces is not significant in geometric modeling.

Knots

Before ending this exploration of the topological properties of closed paths, we will take a last look at the Jordan curve theorem and curves forming knots. Consider a circle C in the plane. It is obvious that C divides the plane into two regions A and B, which are characterized by the property that any two points within any one of the regions (such as a,b and c,d) can be connected by a path that lies wholly within the region; whereas if we take a point from each of the regions (such as a and c), then every path between them will necessarily cut C (see Figure 11.7). The bounded region A can be called the inside of C, and B the outside. The first proof of the now-famous Jordan curve theorem, that every simple closed curve in the plane divides the plane into two regions, appeared in 1893 in the important book, *Cours d'Analyse,* by C. Jordan (1838–1922), although Jordan did not solve the problem completely. Even though Jordan's original proof has by now been greatly simpli-

Figure 11.7 Division of a plane into two regions by a closed curve.

fied, it is still not easy to prove this apparently obvious theorem. The well-defined notion of inside and outside for simple closed curves has direct bearing on many geometric-modeling problems. Incidentally, it is also useful for those who want to build or escape from mazes.

Jordan's theorem also applies to embedded circles in three-dimensional space. An embedded circle in three-dimensional space is a *knot*. The basic problem of knot theory is to determine when two knots are *equivalent;* that is, when they can be transformed into each other by twisting and pulling. Figure 11.8 shows an example of two nonequivalent knots. Try working out these ideas with pieces of string. To distinguish between different knot types in this context requires keeping the endpoints of the string fixed. Bringing the ends together to form a circle creates another familiar problem, but one that is easier to handle mathematically.

We call a knot *trivial* or *unknotted* if it can be deformed into a conventional circle. A special case of the general problem is to determine when a knot is unknotted. For example, if we are given a pile of string, how can we tell if a knot is present? We notice that we cannot distinguish between trivial knots intrinsically because they are all the topological equivalent of circles. Our problem here is to distinguish between various possible placements of the same circle in three-dimensional space, and thus this problem is different from other topological problems where we want to distinguish between objects.

Gauss made some brief remarks on knot theory at various times between 1823 and 1840. He attempted to classify singular curves in the plane, some of which arose from the projection of a knot, and in 1833 he found a formula connecting a certain double integral with the linking number of two knots, which is intuitively the number of times two knots are intertwined (see Figure 11.9).

However, the first mathematician to make a serious study of knots was J. B. Listing (1808–1882). His approach was to project the knot onto the plane and

Figure 11.8 Jordan curve theorem and knots.

Figure 11.9 Linking numbers of knots.

analyze it by considering the intersections, keeping track of over- and undercrossings. Such a method was also used later by P. G. Tait (1831–1901), T. P. Kirkman, and N. C. Little to classify the more elementary knots.

The first invariant characteristics of knots that were effectively computed were found in the 1920s by J. W. Alexander and K. Reidemeist. These invariants were obtained by studying the complement of the knot in three-dimensional Euclidean space and its associated *knot group*, which M. Dehn had defined in 1910. Knot theory, however, is very difficult, and many important unanswered questions remain. We have mentioned this topic only because it has another topological characteristic that might prove useful in geometric modeling.

Piecewise Flat Surfaces

Many of the existing and potential future representation schemes for modeling solid objects use polyhedra and their supporting topology, so there are many opportunities for applying the material presented here. The classical polyhedron is an excellent example with which to begin a discussion of piecewise flat surfaces. By *polyhedron* we mean an arrangement of polygons such that two and only two polygons meet at an edge; and that it is possible to traverse the surface of the polyhedron by crossing its edges and moving from one polygonal face to another until all polygons have been traversed by this continuous path.

The simple polyhedra are the most important, since they are historically the source of topology's contribution to geometric modeling. The term *simple polyhedra* refers to all polyhedra that can be continuously deformed into a sphere. *Regular polyhedra* are an example and subset of the simple polyhedra. Regular polyhedra have no reentrant edges; thus, they are convex. Furthermore, we apply the term *convex* to every polyhedron that lies entirely on one side of each of its polygonal faces. Although convexity is not

a topological property, it does imply one: Every convex polyhedron is a simple polyhedron. A *toroidal polyhedron* is a nonsimple polyhedron.

Euler's Formula

A well-known relationship between the number of vertices, edges, and faces V, E, and F, respectively, of a simple polyhedron is called *Euler's formula*. It states that

$$V - E + F = 2 \tag{11.5}$$

This formula provides a direct and simple proof that there are only five regular polyhedra.

To prove this, let us digress slightly and explore an application of Euler's theorem to geometric modeling of a more classical vintage. We will determine all possible regular polyhedra; that is, those polyhedra with every face having the same number of edges, say, h; every vertex having the same number of edges emanating from it, say, k; and every edge having the same length. Since every edge has two vertices and belongs to exactly two faces, it follows that $Fh = 2E = Vk$. Substitute this into Euler's formula:

$$\frac{2E}{k} - E + \frac{2E}{h} = 2 \tag{11.6}$$

or

$$\frac{1}{E} = \frac{1}{h} + \frac{1}{k} - \frac{1}{2} \tag{11.7}$$

For a polyhedron, we safely assume that $h, k \geq 3$. On the other hand, if both h and k were larger than 3, then the above equation would imply that

$$0 < \frac{1}{E} = \frac{1}{h} + \frac{1}{k} - \frac{1}{2} \leq \frac{1}{4} + \frac{1}{4} - \frac{1}{2} = 0 \tag{11.8}$$

which is obviously impossible. Therefore, either h or k equals 3. If $h = 3$, then

$$0 < \frac{1}{E} = \frac{1}{3} + \frac{1}{k} - \frac{1}{2} \tag{11.9}$$

implies that $3 \leq k \leq 5$. By symmetry, if $k = 3$, then $3 \leq h \leq 5$. Thus, $(h,k,E) =$ (3,3,6), (4,3,12), (5,3,30), and (3,5,30) are the only possibilities. They are, in fact, realized by the tetrahedron, the cube, the octahedron, the dodecahedron, and the icosahedron, respectively. Observe that we did not really use the fact that the edges of the polyhedron all have the same length. As long as the numbers h and k are constant, we still have only five possibilities (up to stretching or contracting).

Poincaré generalized Euler's formula to n-dimensional space. Instead of points, edges, and faces, he defined 0-, 1-, 2-, . . . , $n - 1$-dimensional elements. He denoted the numbers of each of these elements present in a hyperpolyhedron (or polytope) as $N_0, N_1, N_2, \ldots, N_{n-1}$, respectively, and expressed Euler's formula as

$$N_0 - N_1 + N_2 - \cdots = 1 - (-1)^n \tag{11.10}$$

For $n = 3$, this reduces to Euler's formula.

These regular polyhedra were known to Euclid, and he even claimed that there were no others. Euclid's definition of a *regular polyhedron* was, however, rather imprecise by today's standards, and his claim is not quite correct in the general sense in which he states it.

This is our cue to investigate *nonsimple polyhedra*. Nonsimple polyhedra are topological equivalents of any solid object with holes in it and are, therefore, of direct use to us in geometric modeling. In Figure 11.10 are some examples of nonsimple polyhedra. The simplest of these is a rectangular parallelepiped with a through hole, also in the shape of a parallelepiped whose

Figure 11.10 Examples of nonsimple polyhedra.

sides are parallel to the other faces. The two end faces are beveled, or faceted, merely to emphasize the properly interconnected grid of edges, vertices, and faces. The other faces are treated the same way but have more holes. None of them can be deformed into a sphere.

A *connectivity number* N is associated with every polyhedron. If the surface of a polyhedron is divided into two separate regions by every closed path (loop) defined by edges of the polygons making up its faces, we say that the polyhedron has connectivity $N = 0$. All simple polyhedra have $N = 0$, because the surface of a sphere is divided into two parts by any closed curve lying on it. Conversely, any polyhedron with $N = 0$ can be deformed into a sphere.

We can define any orientable unbounded surface, that is, any closed non-self-intersecting surface, as a sphere with G handles. Thus, $G = 0$ for a sphere or any simply connected surface, $G = 1$ for a torus, and $G = 2$ for the surface of a solid figure eight. The number G is called the *genus* of the surface. As we see in Figure 11.10, there are closed loops of edges that do not divide the surface of a nonsimple polyhedron into two parts. We assign these polyhedra a connectivity number greater than one. We define the connectivity number N as the maximum number of distinct loops with or without common points (intersections) to be found on a polyhedron that do not divide its surface into two separate regions. We define the genus G of a polyhedron as the maximum number of nonintersecting loops to be found that do not divide its surface into two regions. Furthermore, we can extend Euler's formula to polyhedra of any genus and connectivity. Thus,

$$V - E + F - 2(1 - G) = 0 \tag{11.11}$$

where $2G = N$, the connectivity number. This is the *Euler-Poincaré formula.*

Conversely, by counting the vertices, edges, and faces, we can determine the connectivity and genus

$$\begin{aligned} N &= -V + E - F + 2 \\ G &= \frac{(-V + E - F + 2)}{2} \end{aligned} \tag{11.12}$$

Thus, we can deform simple polyhedra into a sphere and the rectangular parallelepiped with a through hole into a torus. We can also replace more complicated structures by topologically equivalent polyhedra.

Topological Atlas and Orientation

We will now take a more general approach to developing additional topological properties and consider surfaces formed by taking a collection of planar pieces and gluing them together along their edges, creating piecewise flat surfaces. Any surface formed in this way will obviously be flat everywhere

except possibly along the edges where the pieces are glued together. In fact, if all the planar pieces have straight edges, then the glued surface can have curvature only at the vertices.

The crucial step in designing a modeling algorithm to explore piecewise flat geometry is in deciding how to represent these surfaces. The most straightforward representation is simply to describe each face separately and keep track of which edges are adjoining. This data structure is called an *atlas*. Figure 11.11 shows the atlas of a truncated pyramid. This topological atlas is similar to an

Figure 11.11 Topological atlas of a truncated pyramid.

ordinary road atlas, which is a collection of separate maps that each contain information directing the user to the next map.

Figure 11.12 illustrates two ways of joining a pair of edges. The first, shown on the left, identifies an *orientation-preserving construction*, while the other possibility, shown on the right, identifies an *orientation-reversing construction*. In the figure we have numbered each edge, using an arrow to indicate the ascending direction of the numbers, and have joined opposite edges so that the numbers match and the direction of the two arrows is the same. We can use either numbers or arrows alone to identify the way the two edges are joined. Orientation-reversing constructions have both arrows pointing in the same rotational sense—either clockwise or counterclockwise—around their respective faces. However, if one arrow points clockwise and the other counterclockwise, the construction is orientation-preserving. Including orientation-reversing joints in an atlas enables us to construct new surfaces, including some mathematical curiosities called *nonorientable surfaces*.

In Figure 11.13, we take the collection of matched-edge pairs as the atlas for a closed piecewise square surface. We label each edge with a pair of numbers specifying the face and edge of the face of interest. We number the faces sequentially starting from 1, numbering the four edges of each face clockwise from 0 to 3. For the cube shown, the atlas is

$$[(1,0)\quad(2,0)]\,[(1,1)\quad(5,0)]\,[(1,2)\quad(4,0)]\,[(1,3)\quad(3,0)]$$
$$[(2,1)\quad(3,3)]\,[(2,2)\quad(6,2)]\,[(2,3)\quad(5,1)]\,[(3,1)\quad(4,3)]$$
$$[(3,2)\quad(6,3)]\,[(4,1)\quad(5,3)]\,[(4,2)\quad(6,0)]\,[(5,2)\quad(6,1)]$$

We interpret this as edge 0 of face 1 matched with edge 0 of face 2, and so on.

Figure 11.12 Orientation.

Figure 11.13 Atlas of a cube.

Next, we take a square and identify a pair of opposite sides in an orientation-preserving way to produce a cylinder or, instead, identify the edges with the orientation reversed. Try this with a strip of paper. This identification amounts to making a half-twist in the strip before gluing the edges together, resulting in the Möbius strip.

As inhabitants of a Möbius strip we would observe a curious phenomenon. If we start out at some point and take a trip all the way around the strip, when we get back to our initial position, we find that left and right are reversed. This happens because right and left are not intrinsically defined. Externally, whether a turn appears left or right depends on the side of the surface we are looking from. It is an extrinsic property of the surface and depends on setting up an external reference. Thus, we must specify that the observation is from the top side or the bottom side.

Right and *left* are characteristics of the two-dimensional Möbius strip inhabitant's motions, not characteristics of the surface. As inhabitants of the Möbius strip, we can define right and left on the surface, but the definition works only locally. We cannot look at some point on the surface, then decide a priori that one direction is either right or left. We must first move around on the surface. There is nothing to ensure that, as we move around the sur-

face, the commands to go left or right will generate the same orientation each time we return to a given point. This potential reversal and confusion is exactly what happens on the Möbius strip.

A surface on which left and right are never reversed is *orientable,* and we are able to establish a consistent left and right definition on the surface. If we find a path that confuses left and right, the surface is nonorientable. We can easily demonstrate the orientability of a surface in terms of another property, too. For example, an orientable surface is one on which we can define clockwise and counterclockwise rotations in a consistent way: We choose a point p on the surface S and imagine standing at that point. Next, we decide which of the two possible rotations to call clockwise; this is called an orientation at p. Now, we let q be another point on S. (Note: q may equal p.) As we walk to q along some path, we keep track of which rotation we defined as clockwise. This induces an orientation at q; that is, a sense of clockwise or counterclockwise for rotations at q.

There are obviously many paths from p to q. Furthermore, there is not always a unique shortest path. Different paths may induce different orientations. If we obtain the same orientation no matter which path we take, then the surface is orientable. However, if we walk around the meridian of a Möbius strip, we will end up with the opposite orientation from our original one. Therefore, we again conclude that the Möbius strip is nonorientable, and our new definition agrees with the old. We conclude that orientability is an intrinsic property of surfaces.

Any closed surface that fits into three-dimensional space must be topologically the equivalent of a sphere with g handles, where g can equal zero. Now we add another condition, namely, that the surface must be orientable. A sphere is orientable, and it is not hard to show that adding a handle cannot change that. Note that the Möbius strip does fit into three-dimensional space, but it is not closed. To transform the Möbius strip into a closed surface, we make an orientation-preserving identification of the top and bottom edges on the atlas exactly as we did when producing a torus from a cylinder (see Figure 11.14). The resulting surface is a *Klein bottle,* and, like the Möbius strip, it is nonorientable.

Producing the Klein bottle involves sticking the bottle through itself—a rather drastic operation, but the best we can do in three dimensions. The Klein bottle will not fit into a three-dimensional space without self-intersections.

If we start with a Möbius strip and close it up by making a second orientation-reversing identification, we produce another type of surface, the *projective plane.* The projective plane is nonorientable and topologically distinct from a Klein bottle.

Figure 11.14 Atlas of: (*a*) a cylinder; (*b*) a torus; (*c*) a Möbius strip; and (*d*) a Klein bottle.

The atlas must specify not only which edges are identified, but also whether the orientation is reversed. We denote this by including a +1 or −1, the *transition parity*. Figure 11.15 shows several examples of a notation scheme, including the transition parity number. It should now be obvious that simple bookkeeping changes produce dramatic changes in the topological properties of a surface.

Curvature of Piecewise Flat Surfaces

We now turn our attention to a seemingly paradoxical subject: the curvature of piecewise flat surfaces. It so happens that all of the curvature in a piecewise flat surface is concentrated at the vertices, which makes total curvature easy to compute. We need only sum up the angle excesses of small paths around each of the vertices; thus,

Figure 11.15 Atlas and transition parity of: (*a*) a sphere; (*b*) a torus; (*c*) a Klein bottle; and (*d*) a projective plane.

$$K = \sum_{i=1}^{v} E_i \tag{11.13}$$

where E_i is the excess of a path around the vertex i.

To transform this expression into the simplest, most meaningful terms, we recall that excess is equal to 2π minus the total turning, and we rewrite the equation as

$$K = \sum_{i=1}^{v} (2\pi - T_i) \tag{11.14}$$

where T_i is the total turning of a path around the vertex i. Factoring out the 2π terms (one for each vertex) we obtain

$$K = 2\pi\, V - \sum_{i=1}^{v} T_i \tag{11.15}$$

where V is, of course, the number of vertices in the surface. We can further clarify this expression if we rewrite it using the fact that the total turning around a vertex is equal to the sum of all the interior angles meeting at that vertex. Summing over all the vertices gives us all the interior angles of all the faces in the surface. Regrouping these angles according to the faces they lie in, we compute total curvature as

$$K = 2\pi V - \sum_{i=1}^{F} f_i \tag{11.16}$$

where f_i is the sum of the interior angles of the face i.

This is a surprising result because we compute the second term without knowing how the edges are joined together. Therefore, if we have all the individual faces of a piecewise flat surface and we know V, we can compute total curvature without knowing anything about the atlas.

For piecewise square surfaces, the formula is even simpler, since the sum of interior angles of any face is 2π. Therefore, $K = 2\pi V - 2\pi F$ or $K = 2\pi(v - F)$. This equation for total curvature does not depend on angles at all! And we can do even better. Curiously enough, we can compute the total curvature of any closed piecewise flat surface without knowing any angles. We need only know the total number of faces and vertices and the number of edges. For a closed piecewise flat surface with V vertices, E edges, and F faces, the total curvature is

$$K = 2\pi(V - E + F) \tag{11.17}$$

Because we already know that $K = 2\pi V - \sum_{i=1}^{F} f_i$, to prove Equation (11.17) we must show that the sum $\sum_{i=1}^{F} f_i$ can be expressed independently of the particular values of the angles. We do not know much in general about the sum of the interior angles of a face, but we do know a closely related quantity—the sum of the exterior angles. The boundary of each face is a simple closed path, so we know that the sum of the exterior angles is 2π. To relate the exterior angles to the interior angles, we notice that each exterior angle pairs off with an interior angle to sum to $\pi = 180°$, and there are as many of these pairs as there are edges to a face. Therefore,

$$\begin{aligned} f_i &= [\text{sum of}(\pi - \text{exterior angles})] \\ &= \pi \times e_i - (\text{sum of exterior angles of the face}) \\ &= \pi \times e_i - 2\pi \end{aligned} \tag{11.18}$$

where e_i is the number of edges of the face i. Summing this quantity over all the faces gives

$$\sum_{i=1}^{F} f_i = \sum_{i=1}^{F} e_i - 2\pi F \tag{11.19}$$

Now we simplify $\sum_{i=1}^{F} e_i$, noting that in a closed surface each edge is shared by precisely two faces, and that summing the number of edges in a face over all the separate faces counts each edge exactly twice. This means that

$$\sum_{i=1}^{F} e_i = 2E \tag{11.20}$$

Combining Equation (11.20) with (11.19) yields

$$\sum_{i=1}^{F} f_i = 2\pi(E - F) \tag{11.21}$$

Substituting Equation (11.21) into (11.16) yields

$$K = 2\pi V - \sum_{i=1}^{F} f_i \tag{11.22}$$

or

$$K = 2\pi(V - E + F) \tag{11.23}$$

The quantity $(V - E + F)$ is the Euler characteristic of a surface and is denoted by the Greek letter χ (chi). Using this notation, we rewrite the curvature formula as

$$K = 2\pi\chi \tag{11.24}$$

We can think of this equation as a convenient way to calculate the total curvature of a piecewise flat surface. However, it also implies much more. As we have seen, K has special properties. First, K is defined for all surfaces, not only piecewise flat ones. Also, K is a topological invariant. Therefore, this equation leads us to suspect that the Euler characteristic also has these properties. This suggests the following questions: Can we show directly that $(V - E + F)$ is a topological invariant? Can we define $(V - E + F)$ for all surfaces, not just piecewise flat ones? If we achieve these goals, we provide powerful tools for geometric modeling.

Look at topological invariance. What happens to $(V - E + F)$ when we deform a piecewise flat surface? Absolutely nothing! After all, $(V - E + F)$ is just a counting algorithm, and unless we cut a face in two, remove an edge, or perform some other nontopological transformation, none of the numbers can change. This, then, is our introduction to the next topic.

The Topology of Closed Curved Surfaces

Let us now see how the sum $V - E + F$ works for an arbitrary surface. The image of a distorted piecewise flat surface suggests a new definition for the terms *vertices*, *edges*, and *faces* on a general surface. Therefore, we will define *faces* on the surfaces not as flat polygons but as topological disks, and *edges* not as straight lines but as simple arcs with a vertex at either end. We will define a *net* on a general surface as an arbitrary collection of simple arcs (terminated at each end by a vertex) that divide the surface everywhere into topological disks. For convenience, we will continue to call the elements of a net edges, vertices, and faces. Given a net, we define the *Euler characteristic* χ by the same formula $\chi = V - E + F$.

The definition of the Euler characteristic raises a question: If we can draw an infinite number of different nets on a surface, how do we decide which net to use in computing the Euler characteristic? The answer is that it does not matter which one we choose, because all valid nets on the same closed surface have the same Euler characteristic.

To prove this, we could imagine starting with a particular net on a surface, then transforming this net into a different one by adding or deleting vertices or edges. We can do this in several ways. We single out two elementary net transformations:

1. Adding (or deleting) a face by drawing in (or erasing) an edge between existing vertices.
2. Adding (or deleting) a vertex.

Let us see how these transformations affect the value of χ. If we add an edge, then E increases by 1. But F also increases by 1; hence $F - E$ is unchanged. And because V is unchanged, $\chi = V - E + F$ is unchanged by the first type of transformation. If we insert a new vertex into an edge, we produce not only a new vertex, but also a new edge. Thus, V and E each increase by 1, and F is unchanged; then $V - E + F$ is again unchanged. We conclude that χ is invariant under the two net transformations just defined. To complete the proof that any two nets on the same surface will give the same value of χ, we assert that, given any two nets, we can always get from one to the other by some sequence of these transformations.

Every surface has an Euler characteristic χ, and χ is a topological invariant. This means that now we have two topological invariants for surfaces—the Euler characteristic χ and the total curvature K. Furthermore, we know that they are related by $K = 2\pi\chi$ for piecewise flat surfaces. But does this hold

for any surface? It does, and we may now assert the following theorem: For any closed surface, the total curvature and the Euler characteristic are related by $K = 2\pi\chi$. This is the *Gauss-Bonnet theorem.*

There are two proofs of this theorem. The first is based on the fact that K and χ are both topological invariants. We start with an arbitrary surface and deform it into a piecewise flat surface. Since K and χ are both topological invariants, the deformation leaves them unchanged. Since we know that $K = 2\pi\chi$ for the piecewise flat surface, then it is true for the original surface as well. The only caveat in this proof is that we must show that any surface can be deformed into a piecewise flat surface. Intuitively, this seems possible. The approach is to flatten the surface piece by piece, pushing all the curvature into the edges between the pieces. Then we flatten and straighten the edges piece by piece, pushing all the curvature into the vertices. The details of this procedure are difficult to express more rigorously, and they are not necessary here. A second proof that $K = 2\pi\chi$ is based on a direct computation of K for any surface and is similar to the way we proved the theorem for piecewise flat surfaces.

The Gauss-Bonnet theorem is important in the geometry of surfaces because it produces a relationship between quantities defined purely in terms of topology (such as the Euler characteristic) and quantities defined purely in terms of distances and angles (such as total curvature).

Euler Operators

Euler objects always satisfy Euler's formula. The processes that add or delete faces, edges, and vertices to create a new Euler object are the *Euler operators.* These operators provide a rational method for constructing solid, polyhedra-like objects and ensure that they are topologically valid (that is, closed and oriented).

The connectedness of the boundary surface of a solid is a property distinct from, and independent of, the enclosed interior points. Connectivity, orientation, and the characteristic of being non-self-intersecting are global properties of the surface and depend on all of its parts. Euler's formula asserts a quantitative relationship between these parts that allows us to assign certain distinctive global characteristics: the number of handles or through holes, total curvature, connectivity, and so on.

Since we have just seen that Euler's formula is not restricted to plane-faced polyhedra but also applies to any closed surface on which we can construct a proper net, the formula becomes a useful check on the topological validity of any solid whose surface we can express as a net of patches, curve segments, and vertices. To apply the formula, however, certain other conditions must be met.

1. All faces must be simply connected (i.e., they are topological disks), with no holes in them, and bounded by a single ring of edges.
2. The solid object must be simply connected (i.e., its complement is connected) with no holes through it.
3. Each edge must adjoin exactly two faces and terminate at a vertex at each end.
4. At least three edges must meet at each vertex.

Let us look at some examples of the formula in action in Figure 11.16. It is easy to see that the objects in this figure satisfy Euler's formula and the nets are proper, that is, a collection of simple arcs (edges), terminated at each end by a vertex, that divide the surface into topological disks. Distorting these shapes by making the straight edges curves and the faces nonplanar surfaces does not change the applicability or validity of the formula. Figure 11.17 demonstrates this with two examples of spherical nets.

Vertices, edges, and faces added to a model must produce a result that satisfies both the Euler formula and the four conditions mentioned above. In Figure 11.18a and 11.18b, a cubical polyhedron is legitimately modified. In Figure 11.18c, the formula is certainly satisfied, but notice that edges (1, 5) and (2, 5) do not adjoin two faces. Furthermore, only two edges meet at vertex 5, and edge (1, 2) adjoins three faces. Therefore, as it stands, the object in Figure 11.18c is not a valid solid. We can remedy this situation by adding edges (3, 5) and (4, 5), resulting in a net gain of two edges and two faces. Edges (1, 2), (2, 3), (3, 4), and (4, 1) no longer define a face. Figure 11.19 shows valid modifications to the spherical nets.

An interesting modification of Euler's formula states that for a three-dimensional space divided into C polyhedral cells, the vertices, edges, faces,

Euler's formula: $V - E + F = 2$

Figure 11.16 Euler's formula and simple polyhedra.

Figure 11.17 Euler's formula applied to a spherical net.

and cells are related by

$$V - E + F - C = 1 \tag{11.25}$$

By adding a vertex, point 9, to the interior of a cube (see Figure 11.20) and joining it to each of the other eight vertices with edges, we create a six-celled polyhedron. It is easy to verify the vertex, edge, face, and cell count.

Look at what happens to the polyhedral object with a hole through it in Figure 11.21. Although the Euler formula is satisfied in Figure 11.21a, two of the faces are inadmissible according to our convention. The faces containing the entrance and exit to the hole are not topological disks, and therefore the surface net is not proper. We correct this situation in Figure 11.21b by adding the edges shown as dashed lines, but look at what happens to the formula. Modifying the formula and deleting conditions 1 and 2 accommodates such objects. Thus,

$$V - E + F - H + 2P = 2B \tag{11.26}$$

Figure 11.18 Euler operations on a cube.

Figure 11.19 Modification of an Euler net on a sphere.

where V, E, and F retain their usual meaning and denote the number of vertices, edges, and faces. H denotes the number of holes in faces; P denotes the number of passages (holes through the entire object); and B denotes the number of separate, disjoint bodies (objects). Another acceptable variation is shown in Figure 11.21c; here, the through passage in Figure 11.21a and 11.21b is now merely a concavity. If we add edges as in Figure 11.21d, then we satisfy the original Euler formula. (The individual faces, as well as the interior, are each simply connected entities.)

A polyhedron exhibits nine classes of topological relationships between pairs of the three types of elements: vertices, edges, and faces (see Figure 11.22). Thus, for example, a vertex is topologically characterized by adjacent

$$V - E + F - C = 1$$
$$9 - 20 + 18 - 6 = 1$$

Figure 11.20 Euler's formula and polyhedral cells.

Figure 11.21 Multiply connected polyhedra and a modified Euler formula.

vertices, edges, and faces. For certain graphics applications, it is a good idea to use one of the relationships $V:[V]$, $E:[V]$, or $F:[V]$ to know how vertices are joined, whereas Boolean shape operations are facilitated by the ring of faces around the vertex $V:[F]$, and the Euler operators usually require adjacency among faces $F:[F]$.

11.2 GRAPH-BASED MODELS

A geometric model emphasizing topological structure, with data pointers linking together an object's faces, edges, and vertices, is a *graph-based model*. We can represent a solid object as a list of the object's faces and their respective surface equations. We represent the edges of these faces as curve equations, with pointers to their endpoint vertices and adjoining faces. We represent the vertices as lists of coordinates, with pointers to the edges meeting at each vertex. Notice that there are two kinds of information:

1. The pointers defining the topology or connectivity between vertices, edges, and faces
2. Numerical data defining curve and surface equations and vertex coordinates

Subsequent modeling operations may alter just the pointers, just the numerical data, or both.

Figure 11.23 is an example of a graph-based model. The model contains redundant information, but in many situations this is desirable. Some redundancy can speed up searching algorithms, thereby eliminating the need for global searches. Scaling and rigid-body transformations (translation and rotation) alter only numerical data, leaving the pointers unaffected. In addition, wireframe-based computer-graphics displays can be derived from a graph-based model in an amount of time proportional to the number of edges.

There are interesting differences among graph-based systems. In addition to differences in degree of redundancy, there are differences in how the edges of a face are accommodated. Some graph-based systems require a face to have a single boundary with no hole. Others allow an outer boundary plus

Figure 11.22 Topological relationships between pairs of polyhedron elements.

Vertices	Edges	Faces
V_1	E_1	F_1
V_2, V_3, V_4 E_1, E_2, E_3 F_1, F_2, F_4	V_1, V_2 E_2, E_3, E_4, E_6 F_1, F_4	V_1, V_2, V_3 E_1, E_4, E_2 F_2, F_3, F_4
V_2	E_2	F_2
V_1, V_3, V_4 E_1, E_4, E_6 F_1, F_3, F_4	V_1, V_3 E_1, E_3, E_4, E_5 F_1, F_2	V_1, V_3, V_4 E_2, E_5, E_3 F_1, F_3, F_4
V_3	E_3	F_3
V_1, V_2, V_4 $E_2, \ldots$	V_1, V_4 $E_1, E_2, \ldots$	V_4, V_3, V_2 $E_4, \ldots$

Figure 11.23 A graph-based model.

disjoint inner-boundary loops representing holes. This characterizes the more highly structured systems and offers computational advantages for complex (multiply connected) solids.

For solids represented as planar-faced polyhedra, many economies are available with graph-based systems. Because all edges are straight-line segments, we can store them as vertex pairs, and face boundaries as ordered lists (chains) of vertices. Or we can treat a polyhedron as a simple graph, listing the vertices with their coordinates in one array and their connectivity in another. The latter is a *connectivity matrix* (see Figure 11.24). The connectivity matrix is a binary matrix. Zero-valued elements indicate that no connectivity exists, and one-valued elements indicate that connectivity exists between the pair of vertices. We can construct such a matrix for other elements as well, and one is shown in this figure for the connectivity between the faces of the polyhedron.

The connectivity matrix is also called an *adjacency matrix*. These matrices are convenient for algorithms that determine whether an edge exists between two vertices. If we let a_{ij} denote the element of the connectivity matrix describing the existence of an edge between vertex i and vertex j, then an algorithm accesses a_{ij} and reads its value. The time needed to do this is independent of the number of vertices and the number of edges. The main disadvantage of a connectivity matrix is that it requires V^2 storage (V = number of vertices) even though most $a_{ij} = 0$. If efficient packing is available the storage requirement can be reduced to V^2 bits.

A graph, then, is a set of nodes (or points) connected by branches (lines). The number of branches at a node determines its degree. If any branch has a direction associated with it, then the graph is a *directed graph* (see Figure 11.25). For directed graphs, we can assign an *in degree* and an *out degree* to each node by counting the branches pointing to it and away from it, respectively. A path from one node to another is the sequence of branches to be traversed. If the start and end nodes of a path are the same, the path is a *circuit*. If there is a path between any pair of nodes of a graph, the graph is *connected*. A *tree* is a connected graph without circuits. A graph containing some or all of the nodes and branches of a graph G and no other nodes or branches is a *subgraph* of G. A *spanning tree* of a connected graph G is a subgraph containing all its nodes and enough branches to maintain connectivity without creating any circuits.

1 = Connectivity
0 = No connectivity

Vertex

Vertex	1	2	3	4	5	6	7	8
1	0	1	0	1	1	0	0	0
2	1	0	1	0	0	1	0	0
3	0	1	0	1	0	0	1	0
4	1	0	1	0	0	0	0	1
5	1	0	0	0	0	1	0	1
6	0	1	0	0	1	0	1	0
7	0	0	1	0	0	1	0	1
8	0	0	0	1	1	0	1	0

Face

Face	A	B	C	D	E	F
A	0	1	1	1	0	1
B	1	0	1	1	1	0
C	1	1	0	0	1	1
D	1	1	0	0	1	1
E	0	1	1	1	0	1
F	1	0	1	1	1	0

Figure 11.24 Connectivity matrices for a polyhedron.

Figure 11.25 Examples of graphs.

We may represent a graph in many ways. One way uses a list of vertices (nodes) and their connectivity matrix (branches), as we saw earlier. Another interesting way uses bit vector matrix elements, which often results in considerable efficiency for graph algorithms. Lists may represent a graph by having a list of adjacent vertices associated with each vertex or node. Figure 11.25b shows a directed graph with six nodes. Its connectivity (or adjacency) matrix is

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Note that for a directed graph, $a_{ij} = 1$ only if there is a directed edge from vertex i to vertex j. The six adjacency lists are

vertex $A[B,F]$	vertex $D[E]$
vertex $B[C,F]$	vertex $E[C]$
vertex $C[D]$	vertex $F[E]$

Edges of undirected graphs are represented twice in an adjacency matrix. If there is an undirected branch (or edge) connecting vertex i to vertex j, then $a_{ij} = a_{ji} = 1$. When we delete an edge from an undirected graph, we delete both a_{ij} and a_{ji} from the matrix. In the case of adjacency lists, we create a supplementary array to link the double entries of edges and speed the search process.

Again, a tree is a directed graph with no circuits. A tree has the following additional properties:

1. There is one and only one node, called the *root*, which no branches enter.
2. Every node except the root node has one and only one entering branch.
3. There is a unique path from the root node to each other node (shown in bold outline in Figure 11.26).

The terms *entering branch, exiting branch, parent node,* and *descendant node* are all relative to the root node.

Although it may be disconcerting to see trees with roots at the top and leaves at the bottom, that is the convention. Given two adjacent nodes a and b, as in Figure 11.26, if the branch is directed from a to b, then a is a proper ancestor of b (sometimes called the father of b), and b is a proper descendant of a (a son of a). A leaf node has no proper descendants.

Figure 11.26 A tree graph.

Several quantitative parameters are associated with a tree, as in Figure 11.26. Looking at node b, we find that the *depth* of this node in the tree is the length of the path (number of branches traversed) from the root R to b. The *height* of node b is the length of the longest path from b to a leaf. We also notice that the height of the tree is the height of the root. The *level* of a node is the height of the tree minus the depth of the node.

If the descendants of each node are in order, say, from left to right, then the tree is ordered. When we study Boolean models in Section 11.3, we will use a special type of tree, called a *binary tree*, which is an ordered tree where each node has two descendants—a left descendant and a right one. The tree in Figure 11.26 is a binary tree.

The binary tree in Figure 11.27 is complete. A binary tree is complete if for some integer k every node of depth less than k has a left and right descendant and every node of depth k is a leaf. This is true for the nodes at depth $k = 3$ in the figure. The total number of nodes in a complete binary tree of height k is $2^{k+1} - 1$.

Algorithms must often traverse a binary tree, that is, visit each node to get information about it or the path on which it lies. We will investigate three systematic ways to do this: *preorder traversal*, *postorder traversal*, and *inorder traversal*. Figure 11.28 illustrates these traversals on the binary tree shown in Figure 11.27. First, we define a node n and all its descendants as a subtree t_n of tree T. The node n is the root of t_n. We let r denote the root of the tree and d_L and d_R its immediate descendants. Then we define the *preorder traversal* of T recursively as follows (see Figure 11.28a):

1. Visit the root r.
2. Visit in preorder the subtrees with roots d_L and d_R, in that order.

Figure 11.27 A complete binary tree.

Figure 11.28 Traversals of a binary tree.

We define the *postorder traversal* of T recursively as follows (see Figure 11.28b):

1. Visit in postorder the subtrees with roots d_L and d_R, in that order.
2. Visit the root r.

Finally, we define the *inorder traversal* recursively as follows (see Figure 11.28c):

1. Visit in inorder the left subtree of root r (if it exists).
2. Visit r.
3. Visit in inorder the right subtree of r (if it exists).

If we assign a number in sequence to each node we visit, we notice some interesting properties. The inorder traversal numbers of the nodes of a binary tree occur in such a way that each node in the left subtree of a root node n_R has a number less than n_R, while each node in the right subtree has a number greater than n_R.

A *wireframe model* is the simplest example of a graph-based model, consisting of a set of vertices defined by their coordinates and a connectivity matrix defining how the vertices are connected to form straight-line edges. Faces either are not explicitly defined or are given as a separate list, or circuit of vertices. The literature describing the basics of wireframe modeling methods is vast and accessible. Work in this area is by no means finished. This is demonstrated by the work of Agarwal and Waggenspack (1992) on decomposition methods for extracting face topologies from wireframe models; of Hojnicki and White (1988) on converting CAD wireframe data to surfaced representations; of Lequett (1988) on the automatic construction of curvilinear solids from wireframe views; of Brewer, Vicknair, and Courter (1989) on converting wireframes to solids; and by the work of many others.

Using a data structure equivalent to a directed graph, Letcher and Shook (1995a) present a new logical framework they call *relational geometric synthesis* (*RGS*) for constructing complex geometric models from points, curves, surfaces, and solids (see Chapter 12). Their graph structure is more complicated than the conventional tree, because most of their geometric entities require two or more support nodes. The RGS scheme incorporates embedded entities (points on curves, curves on surfaces, and so on) and relative entities. The data structure ensures durability of intersection curves, as well as durability of the results of compositional processes. Changes in any part of a complex geometric model are automatically reflected in contingent parts of the model.

11.3 BOOLEAN MODELS

The discussion of Boolean models begins with a brief review of set theory as it affects Boolean models and operators. In geometric modeling, when we

combine the simple shapes called *primitives* to form more complex shapes, set theory becomes useful. Solid-modeling techniques, in particular, have drawn considerably from the axioms of set theory. For example, important early work at the University of Rochester draws from point-set topology to establish more rigorous mathematical foundations for solid modeling (see Requicha and Voelker [1977] and Requicha and Tilove [1978]). We begin with some of the basic concepts of set theory and finish with a look at point-set topology, including set-membership classification and its relevance to solid modeling. (Incidentally, the first formal treatment of sets dates back to only the second half of the nineteenth century, when Cantor created the main body of the theory.)

The term *set* denotes any well-defined collection of objects. Objects belonging to a set are its elements or members. In geometric modeling, solid or otherwise, the basic element is the point. The symbols [|] indicate the set-builder notation and describe the set in terms of conditions on any arbitrary element of the set—conditions that every element of the set must meet. For example, $[x|2.5 < x < 3.5]$ is a set consisting of all real numbers in the specified interval. On the left-hand side of the vertical line we read, "the set of all x . . ." On the right-hand side we have the conditions for set membership.

A set that contains all the elements of all the sets under consideration is the *universal set*, denoted by E. A deck of 52 playing cards may be the universal set if we are investigating the probabilities of certain combinations of cards in a poker hand. Conversely, a *null set* is a set which has no elements at all. It is denoted by ϕ and is also referred to as the *empty set* or the *void set*.

In general, two sets A and B are equal, expressed as

$$A = B \tag{11.27}$$

whenever set A and set B contain exactly the same elements. Two sets A and B are in one-to-one correspondence if we can pair each element in A with exactly one element in B, and each element in B with exactly one element in A. Two sets A and B are equivalent sets if we can place their elements in one-to-one correspondence.

Any set A is a subset of set B only if every element in A is also an element in B. The symbol $\subset$ indicates the subset relationship. Thus,

$$A \subset B \tag{11.28}$$

means that A is a subset of B. A is a proper subset of B if every element in A is contained in B and if B has at least one element not contained in A. Every set is a subset of itself, but not a proper subset.

New sets are formed by combining the elements in two or more sets in some fashion. Given sets A and B, we can construct a third set C whose ele-

ments are all of the elements in A together with all of the elements in B; that is, the union of A and B. We write this as

$$C = A \cup B \tag{11.29}$$

By the union of two sets, we mean the set of all elements in one set or in the other set or in both sets. For example, if $A = [a,b,c]$ and $B = [c,d,e,f]$, then $C = A \cup B = [a,b,c,d,e,f]$. Notice that there is no repetition of elements in C, even though element c is in both A and B. We may read $A \cup B$ as A or B. Here, the *or* is what logicians call *the inclusive or* because $A \cup B$ includes the elements that belong to A as well as B.

If we form a set D comprised of elements common to both A and B, set D is the intersection of A and B, expressed as

$$D = A \cap B \tag{11.30}$$

By the intersection of two sets, we mean the set of all elements, and only those elements, that are in both sets. Thus, if $A = [a,b,c,d]$, $B = [c,d,e,f,g]$, and $D = A \cap B$, then $D = [c,d]$.

If A and B are sets, then their difference (or more formally, their Boolean difference) $A - B$ denotes the set of elements in A that are not also elements in B. Thus, if $A = [a,b,c,d]$, and $B = [d,e,f]$, then $A - B = [a,b,c]$ and $B - A = [e,f]$. Clearly, $A - A = 0$. This minus sign does not combine with $\cup$ in the way that a plus analogy might suggest. For example, $(A \cup A) - A = 0$, but $A \cup (A - A) = A$. The complement of a set A with respect to a universal set E is the set of all elements in E that are not elements in A, written cA. Furthermore, if we form complements with respect to a set E containing A and B as subsets, then $A - B \equiv A \cap cB$ and $A \cup B \equiv c(cA \cap cB)$. These two identities often allow us to simplify computations. In boundary evaluation algorithms, with appropriate pre- and postprocessing, only the intersection operator is required.

One way to study sets and subsets is with *Venn diagrams*, which are particularly useful in representing such set relationships as equality and such set operations as union and intersection. Figure 11.29 illustrates some set properties and operations using these diagrams. There are several ways we can combine sets, and certain well-established rules that govern these operations (see Table 11.1).

The geometric nature of Venn diagrams suggests useful ways for us to interpret the properties of sets and operations on them. In geometric modeling, sets consist of points, and the universal set E is the set of points defining a Euclidean space with a dimension of our choosing. Set theory suggests methods for operating on these points and classifying them according to such properties as inside, outside, or on the boundary of a geometric solid.

Figure 11.29 Venn diagrams and set theory.

Let us consider the real line defined by a continuous set of points in E^1, where E^1 is the universal set of points we are considering (see Figure 11.30). We define a subset X of E^1 by the line segment $a < X < b$, where a and b are limiting points of the set X. X is an open set, where the open circles at a and b symbolize that these points are not included in the set, and the solid circles in the lower figure indicate their inclusion. The closure of an open set is the union of the set with the set of all its limit points. These concepts extend to two- and three-dimensional spaces and point sets.

The *boundary* of a closed set is the set of all its limit points. Conversely, the *interior* of a closed set is the set of all points of a set not on its boundary. Thus,

$$X = bX \cup iX \tag{11.31}$$

where bX denotes the set of boundary points and iX denotes the set of interior points.

We are now ready to see how these concepts allow us to use simple shapes to create more complex ones. We will use the set operators union ($\cup$), intersection ($\cap$), and difference ($-$), also called *Boolean operators*, and the rules for their application and combination, called *Boolean algebra*.

Boolean Operators

A distinguishing feature of the geometric objects we will deal with here is that we define them as closed sets of points having a boundary subset and an

Table 11.1 Properties of operations on sets

Union Properties	
1. $A \cup B$ is a set.	Closure property.
2. $A \cup B = B \cup A$	Commutative property.
3. $(A \cup B) \cup C = A \cup (B \cup C)$	Associative property.
4. $A \cup \phi = A$	Identity property.
5. $A \cup A = A$	Idempotent property.
6. $A \cup cA = E$	Complement property.

Intersection Properties	
1. $A \cap B$ is a set.	Closure property.
2. $A \cap B = B \cap A$	Commutative property.
3. $(A \cap B) \cap C = A \cap (B \cap C)$	Associative property.
4. $A \cap E = A$	Identity property.
5. $A \cap A = A$	Idempotent property.
6. $A \cap cA = \phi$	Complement property.

Distributive Properties	
1. $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	Union is distributive over intersection.
2. $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	Intersection is distributive over union.

Complementation Properties	
1. $cE = \phi$	The complement of the universal set is the empty set.
2. $c\phi = E$	The complement of the empty set is the universal set.
3. $c(cA) = A$	The complement of a complement of a set A is A.
4. $c(A \cup B) = cA \cap cB$	DeMorgan's law.
5. $c(A \cap B) = cA \cup cB$	DeMorgan's law.

interior subset. We use Boolean operations similar to set intersection, union, and difference to combine simple objects to form more complex ones. The algorithms that perform these operations must produce as output objects that are also closed sets of points having boundary and interior subsets and preserve the dimensionality and homogeneity of the initial objects. The latter requirement means that in any Boolean operation, such as $A \cup B = C$, all the objects must be of the same spatial dimension. See also Requicha and Voelker (1985) and Putnam and Subrahmanyam (1986).

Figure 11.30 Open and closed sets.

Figure 11.31 demonstrates how the ordinary set-theoretic intersection of two well-defined two-dimensional objects produces a result that does not meet our requirements. First, A and B are well defined because each possesses a boundary set bA and bB and an interior set iA and iB. Second, the resulting intersection is mathematically correct according to set theory but geometrically incorrect, or at least inadmissible, because C has no interior. Thus, C is not like A and B. It is not a two-dimensional object and, clearly, this intersection operation did not preserve dimensionality. We would prefer an operator that recognizes this condition and produces a null set. Requicha (1977) and others early on proposed the use of *regularized set operators*, which preserve dimensionality and homogeneity (no dangling or disconnected parts of lower dimension). The importance of these regularized operators will soon become apparent.

We begin by combining two simple polygons A and B, as shown in Figure 11.32. Both A and B are two-dimensional objects with straight-line boundaries. The arrows indicate the direction of parameterization. Notice that both A and B are parameterized in a consistent direction. In this case, they are

Figure 11.31 Degenerate intersection of two well-defined two-dimensional objects.

Figure 11.32 Union, difference, and intersection of two simple polygons.

both counterclockwise. By maintaining consistency in orienting the polygons' edges, we can establish a convention defining which side contains points inside the polygon—on the left in our example.

We assume that we have already applied the appropriate scaling and positioning transformations A and B, so now we will look at an algorithm for finding $C = A \cup B$.

1. Find all the intersection points of the edges of A and B; points 1, 2, 3, and 4.
2. Segment the edges of A and B. Thus, if the boundary polygon of A is parameterized from $u = 0$ to $u = 1$ and B from $v = 0$ to $v = 1$, then the boundary of A in the example has four segments: $u \in [u_1; u_2]$, $u \in [u_2; u_3]$, $u \in [u_3; u_4]$, and $u \in [u_4; u_1]$. B also has four segments: $v \in [v_4; v_3]$, $v \in [v_3; v_2]$, $v \in [v_2; v_1]$, and $v \in [v_1; v_4]$.
3. Find a point on the boundary polygon of A that is outside B, say, $\mathbf{p}_0$. Then that segment is also outside B.
4. Starting at $\mathbf{p}_0$, trace the boundary of A to the next intersection point with B, point 1.
5. Find the intersecting segment of B, and trace along it in the direction of increasing v to its intersection with A, point 4. At this point, we discover that we have traced back to the starting segment on A, but we have not yet exhausted the list of segments. We have found only one loop.

6. Find a point on one of the remaining segments of A that is outside B. Then that segment, too, is outside B.
7. Trace this boundary segment to the next intersection point with B, point 2.
8. Repeat step 5, tracing B to point 3. We have completed another loop. Because there are no more qualifying segments to trace, the union operation is complete. Note that the second loop is parameterized in a clockwise direction, which means that it encloses a hole.

We label the boundary segments of A and B that comprise the loops as the *active* segments; conversely, the other boundary segments we label as *inactive*. If we no longer need representations of the initial objects A and B, we discard them and delete them from the modeling database. In any event, we represent object C by reparameterized concatenations of the appropriate active segments of A and B, forming two separate and distinct loops, as we have just seen.

The difference operation is the same as the union operation except that we trace the boundary segments of B clockwise. The intersection operation, too, is similar to union, except that segment tracing must start from a point on the boundary of A that is inside B.

The algorithm just outlined requires several supporting algorithms. The principal ones are a procedure for finding the intersection points; a procedure for determining whether a point is inside or outside a closed polygon; and procedures tracing segments, forming loops, and reparameterizing the loops. The list will grow as we investigate more sophisticated problems involving Boolean operators.

The example we have just explored is very simple—too simple, in fact, to illuminate most modeling problems we encounter when applying Boolean operators. However, the example demonstrates the basic approach: intersecting, testing, tracing, and sorting. Determining the topology of surfaces that are the product of Boolean combinations of other surfaces is a subtle and complex process and continues to be a challenging subject of research.

Consider the two-dimensional objects A and B in Figure 11.33. Each is well defined; that is, closed and dimensionally homogeneous. Thus, we express A and B as

$$A = bA \cup iA \qquad \text{and} \qquad B = bB \cup iB \tag{11.32}$$

where, as in Equation (11.31), b denotes the set of points on the boundary and i denotes the set of interior points.

Next, we translate A and B into position prior to combining them by the Boolean operation to form object C. First, we perform the set-theoretic

Figure 11.33 Set-theoretic and regularized Boolean intersections.

intersection, with the proper result shown in the figure. Notice the dangling edge; this result is obviously not dimensionally homogeneous, yet it is the correct set-theoretic intersection. The result we seek is shown to the right. It is the regularized Boolean intersection denoted $A \cap {}^*B$, closed and dimensionally homogeneous. Given the set-theoretic intersection, we have

$$C = A \cap B \tag{11.33}$$

We rewrite this as

$$C = (bA \cup iA) \cap (bB \cup ib) \tag{11.34}$$

which expands to

$$C = (bA \cap bB) \cup (iA \cap bB) \cup (bA \cap iB) \cup (iA \cap iB) \tag{11.35}$$

Figure 11.34 shows a geometric interpretation of each of the four expressions in parentheses, above. We notice, again, that the direction of parameterization is indicated by arrows in Figure 11.34a. Since $C = bC \cup iC$, we must find the subsets of bC and iC that form a closed, dimensionally homogeneous object C^*. We must derive candidates for C^* from the terms in Equation (11.35). In Figure 11.34d, we see the two-dimensional interior of C and correctly surmise that

$$iC = iC^* = iA \cap iB \tag{11.36}$$

Figure 11.34 Candidate components of a regularized Boolean intersection.

Next, we must determine bC^*, where $bC^* = \text{Valid}\ (bA \cup bB)$. Note that the boundaries of any new object will always consist of boundary segments of the combining elements. We can generalize this observation as follows. Boundary points can become interior points, whereas interior points cannot become boundary points. Furthermore, for regularized intersections, we assert without proof that

$$iA \cap bB \subset bC^* \tag{11.37}$$

and

$$bA \cap iB \subset bC^* \tag{11.38}$$

So far, we have successfully accounted for Figures 11.34b to 11.34d of the set-theoretic intersection in our effort to determine the regularized intersection. We must now analyze $(bA \cap bB)$ (see Figure 11.34a) to determine which of its subsets are valid subsets of the boundary of C^*. The isolated point is a valid member of bC^*, because it must always be a member of both $(iA \cap bB)$ and $(bA \cap iB)$. We are left with two apparently identical overlapping intersections of A and B. These intersections are distinguished because they are interior to neither A nor B. Let us find a test to differentiate between them.

At some point $\mathbf{p}_1$ on segment 1, we create a new point $\mathbf{p}_R$ offset perpendicularly from it a distance ϵ to the right (relative to the direction of parameterization), and a similar point $\mathbf{p}_L$ to the left (see Figure 11.35). We do the same at a point $\mathbf{p}_2$ on segment 2. Next, we construct a table for each segment to test whether each point $\mathbf{p}_R$ and $\mathbf{p}_L$ is inside A or B. For segment 1, neither test point is inside both A and B, whereas for segment 2, test point $\mathbf{p}_L$ is inside both A and B. This test determines that segment 2 is a valid boundary of C^*. This procedure analyzes the neighborhood of points on the segment so that we can properly classify it.

Another, simpler classification test is available if we adopt a consistent convention for the direction of parameterization. At $\mathbf{p}_1$ we compute the tangent vector $\mathbf{p}_1''$ from the A boundary representation and $\mathbf{p}_1'$ from the B boundary representation. In our example, they are in opposite directions. We do the same for segment 2, where the tangent vectors are in the same direction. From this we conclude: If the respective tangent vectors at a point of the overlapping boundaries of two intersection objects A and B are in the same direction, then the overlapping segment is a valid boundary of $C^* = A \cap^* B$; otherwise the segment is not a valid boundary.

Let us summarize the results. The valid regularized intersection of two objects A and B is

$$C^* = A \cap^* B \tag{11.39}$$

Segment 1	In A	In B
p_R	0	1
p_L	1	0

Segment 2	In A	In B
p_R	0	0
p_L	1	1

Note: 1 = yes
0 = no

Figure 11.35 Regularized boundary test.

where

$$\begin{aligned} C^* &= bC^* \cup iC^* \\ &= \text{Valid}_b\,(bA \cap bB) \cup (iA \cap bB) \cup (bA \cap iB) \cup (iA \cap iB) \end{aligned} \tag{11.40}$$

There is nothing in this expression that indicates dimensionality, so we conclude that it applies equally to one-, two-, three-, or n-dimensional objects. Later in this section, we will investigate the Boolean combination of three-dimensional solids. First, let us continue operations on the objects in Figure 11.33.

The components of the set-theoretic union of A and B are shown in Figure 11.36. Again, they are the full set of valid candidates for determining $C^* = A \cup^* B$. We begin, as in the preceding intersection, by expanding the set-theoretic expression

$$C = A \cup B \tag{11.41}$$

$$C = (bA \cup iA) \cup (bB \cup iB) \tag{11.42}$$

$$C = (bA \cup bB) \cup (iA \cup bB) \cup (bA \cup iB) \cup (iA \cup iB) \tag{11.43}$$

Equation (11.43) is just as correct and subject to interpretation without the parenthetic separation of terms. Thus,

$$C = bA \cup bB \cup iA \cup \underline{bB} \cup \underline{bA} \cup iB \cup \underline{iA} \cup \underline{iB} \tag{11.44}$$

Figure 11.36 Candidate components of a regularized Boolean union.

The redundant elements are underlined. We delete them so that Equation (11.44) becomes

$$\begin{aligned} C &= A \cup B \\ &= bA \cup bB \cup iA \cup iB \end{aligned} \tag{11.45}$$

which is identical to Equation (11.42) with parentheses deleted. From these components, we determine bC^*, iC^*, and, hence, C^*. First, we observe that

$$iC^* = iA \cup iB \cup [\text{Valid}_i(bA \cap bB)] \tag{11.46}$$

and note that some boundary points become interior points. If we do not include these points, we will have what amounts to a hole in iC^*. Also, we notice that it is redundant to add $\cup(bA \cap iB) \cap (bB \cap iA)$ to the right side of Equation (11.46). Why is this not the case for $\cup[\text{Valid}_i(bA \cap bB)]$?

Next, we observe that

$$bC^* = \text{Valid}_b(bA \cup bB) \tag{11.47}$$

where

$$\text{Valid } bA = bA \text{ not in } iB \text{ and part on } bB \tag{11.48}$$

or

$$\text{Valid } bA = bA - [(bA \cap iB) \cup \text{Valid}_b(bA \cap bB)] \tag{11.49}$$

Similarly,

$$\text{Valid } bB = bB - [(bB \cap iA) \cup \text{Valid}_b(bA \cap bB)] \tag{11.50}$$

An ambiguity exists in $(bA \cap bB)$ that we must resolve by a test similar to that discussed for the intersection operator. If we discard all of $(bA \cap bB)$, then bC^* is incomplete. Therefore, the boundary of the regularized set C^* is

$$bC^* = bA \cup bB - [(bA \cap iB) \cup (bB \cap iA) \cup \text{Valid}_b(bA \cap bB)] \tag{11.51}$$

In Equations (11.46) and (11.51), we see that $(bA \cap bB)$ is subdivided, a portion assigned to iC^* and a portion to bC^*. Furthermore, nothing is lost, since we assert $\text{Valid}_i(bA \cap bB) = \text{Valid}_b(bA \cap bB)$.

Finally, we consider Figure 11.37. Here are the components of the difference operator $(A - B)$. Again, we expand the set-theoretic expression

$$C = A - B \tag{11.52}$$

to yield

$$C = (bA - bB - iB) \cup (iA - bB - iB) \tag{11.53}$$

Figure 11.37 Candidate components of a regularized Boolean difference.

Two things are immediately clear from the figure. First, iC^* must equal $iA - bB - iB$; in the case of our example, two disjoint sets result. Second, $C^* \neq C$, because certain segments of bC^* are missing from C. If we add $iA \cap bB$ to C, as in Figure 11.37d, the boundary is still incomplete. The missing segment is a subset of $bA \cap bB$. Here, again, we must perform a test to determine the valid subset. For the case of the difference operator, Valid($bA \cap bB$) are those segments adjacent to only iC^* or ($iA - iB$). Thus,

$$\begin{aligned} bC^* &= bC \cup (\mathrm{iA} \cap \mathrm{bB}) \cup \mathrm{Valid}(bA \cap bB) \\ &= (bA - bB - iB) \cup (iA \cap bB) \cup \mathrm{Valid}(bA \cap bB) \end{aligned} \tag{11.54}$$

Therefore C^*, the regularized $(A - B)$, is

$$C^* = (bA - bB - iB) \cup (iA \cap bB) \cup \mathrm{Valid}(bA \cap bB) \cup (iA - bB - iB) \tag{11.55}$$

Other relationships between A and B are possible. Figure 11.38 shows a particularly useful example. Here, object A completely encloses B. The equations just developed verify the results. This is a way to model holes.

If we execute a sequence of two or more Boolean operations on a set of objects, then the result depends on the order of the sequence. In Figure

Figure 11.38 Examples of Boolean operations.

11.39, three objects are combined, and the results are obviously dependent on the order.

Boolean operators apply to three-dimensional solids in exactly the same way that two-dimensional objects do. The regularized Boolean combining operations are the same, and closure and dimensional homogeneity are also necessary. Figure 11.40 illustrates the effect of various combining operations on two simple three-dimensional objects. In Figures 11.40a to 11.40c, nothing unusual results, but in Figures 11.40d to 11.40f, nonregularized operations produce intersections that are not three-dimensional. Applying the regularized operators to these last three cases properly produces null results.

Set-Membership Classification

In order to regularize sets resulting from combining operations on other sets, we must determine whether a given point is inside, outside, or on the bound-

Figure 11.39 Order dependence of Boolean operations.

ary of a given set. Three important subsets of any regularized set A are: the set of all its interior points, denoted iA; the set of all points on its boundary, denoted bA; and all points outside it, denoted cA. Assignment of a specific point to one of these sets is *set-membership classification*. The early work in formalizing and making rigorous set-membership classification for geometric modeling was done by R. B. Tilove (1980) and others at the University of Rochester. They observed a similarity between four types of geometric problems related to modeling:

1. *Point inclusion.* Given a solid and a point, is the point inside, outside, or on the boundary of the solid?
2. *Line/polygon clipping.* Given a polygon and a line segment, what part of the line is inside the polygon? What part is outside the polygon? What part is on the boundary of the polygon?
3. *Polygon intersection.* Given two polygons A and B, what is the intersection polygon $A \cap B$?
4. *Solid interference.* Given two solids, do they interfere (that is, intersect unintentionally)?

In each problem, two geometric elements are involved, requiring us to determine some sort of inclusion relationship between them. Creating a set-

Figure 11.40 Boolean operations on a three-dimensional solid.

membership classification function unifies the approach to these problems. We denote this function as $M[\quad]$, which operates on two specified point sets—a reference set S and a candidate set X. Thus, $M[X,S]$ partitions X into subsets corresponding to their membership in bS, iS, or cS.

Let us look at several classification relationships and some classification methods. We will consider the possible relationships between sets taken from E^0, E^1, E^2, and E^3 universal sets; that is, points, lines or curves, surfaces, and solids.

Relationships involving points are shown in Figure 11.41. The most basic, yet easily overlooked, classification relationship is between two points: A test point may be the same as a given point, or it may be different. A simple algorithm to compare coordinates with some degree of precision will produce the classification.

A point may be on or off a given curve; if on the curve, three subclasses are possible—on the initial point, on the endpoint, or on an intermediate point.

For a straight line, an additional pair of classifications for a point exist—on the back extension of the line or on the forward extension of the line. Finally, a point can be to the right or left of a line or curve on a surface relative to its direction of parameterization. A simple test for right-left status is to first find the point on the given curve nearest the candidate point. Given a consistent convention, the sign of the resulting vector product indicates on which side of the curve we find the point.

Given a bounded, simply-connected topological disk (for example, a closed, non-self-intersecting curve lying on a surface, of which a convex polygon on a plane is a special case), then a point may be inside the disk, on its boundary, or outside of it. In the special case of a straight-edged plane polygon, there are additional classifications relative to the boundary—the point may be on a specific edge or a vertex.

One way to classify a point as inside or outside a disk or polygon is to compute the *winding number* (see Apostol [1978] or any other calculus textbook). The sign of the winding number depends on the direction of parameterization (see Figure 11.42). We can arrange the numerical integration of the appropriate line integral and the direction of parameterization to yield either the signs shown in the figure or the reverse.

Figure 11.41 Point classification.

Figure 11.42 The winding number and the inside-outside classification.

Given a solid, that is, a regularized three-dimensional set of points, a candidate point can again have the three basic relationships to it—inside, outside, or on its boundary. If the solid is a polyhedron, then we can further describe the point on its boundary as being on an edge or a vertex. There are several ways to determine the inside-outside status. Let us look at two approaches.

Figure 11.43a shows an arbitrarily shaped solid whose boundary we will assume is a parametric surface. Given **p**, we compute the closest point **q** to it on the surface. Next, we compute the surface normal **n** at **q** and compare its direction to the vector (**p** − **q**). Assuming the convention of an outward-

Figure 11.43 Inside and outside a solid.

pointing normal, then if $\mathbf{n}$ and $(\mathbf{p} - \mathbf{q})$ have the same sign, $\mathbf{p}$ is outside the solid; otherwise, it is inside (if not on the boundary, in which case $\mathbf{p} - \mathbf{q} = 0$).

Figure 11.43b presumes a solid defined as the Boolean intersection of half-spaces in E^3. Here, we simply test $\mathbf{p}$ against the set of inequalities defining the half-spaces. As we proceed through an ordered list of these inequalities, we update a status flag on $\mathbf{p}$. As long as $\mathbf{p}$ satisfies each successive half-space definition, it is *inside* the solid, and we so flag it. If $\mathbf{p}$ identically satisfies a half-space limit (for example, $\mathbf{p}_z = e$), then we change its status to *on the boundary*. If the point $\mathbf{p}$ fails any test, the algorithm terminates, and we flag the point's status as *outside* the solid.

Set-membership classification also extends to curves and straight-line segments and, perhaps most importantly, to straight lines and polygons. Figure 11.44 shows the principal relationships between two curves and two straight lines. Detection and computation in Figures 11.44a and 11.44b for curves, and Figures 11.44c, 11.44h, and 11.44i for straight lines are fairly straightforward. The remaining cases are examples of overlap. For straight lines, detecting two points in common or a common point and common absolute direction of tangent vectors at the point indicates overlap. With curves, the situation is more complex. Here, we must accept an approximation of overlap because, unless forced to do so, most incidental matches will not mate identically.

Line and polygon classifications are shown in Figure 11.45: the polygons represent regularized sets in E^2. A single line segment defining an edge of polygon B intersects polygon A in three ways. The membership-classification function $M[\quad]$ subdivides and classifies this line: X_1 is a subset of the boundary of A, X_2 is a subset of the interior of A, or X_3 is a subset of the outside or complement of A. Several membership tests for these conditions were suggested previously, so we will not discuss them again here.

In Figure 11.46, we see the tangent vector convention, also mentioned earlier. This convention permits us to locate the direction of the inside of an object from any point on the boundary. We then extract information about its immediate neighborhood from the boundary representation. An ambiguity arises when we try to classify point 2, say, with respect to $A \cap B$, given only the classification of the point with respect to A and B separately. We cannot express $M[X,A|OP|B]$ solely in terms of $M[X,A]$ and $M[X,B]$, and we need more information to resolve ambiguities. Here, the point is on bA and on bB, but not on bC, although $C = A \cap B$. We see in the figure that we can use information embedded in the boundary itself to remove ambiguities. Of course, this does presuppose an analytical representation of the boundary.

Figure 11.44 Curve and line-segment classification.

Finally, we consider the two intersecting solids in Figure 11.47. If the surface of each one is represented analytically, then we can compute normal vectors at any point on them. We can use this information in a set-membership classification algorithm. We assume that the surfaces of the solids are parameterized in such a way that the normal at any point on them always points outward, away from the interior or material of the object. Then, on overlapping surfaces, we can easily classify a neighborhood. If $\mathbf{n}_A = \mathbf{n}_B$, the iA and iB are on the same side, and $iA \cap iB$ is not a null set. If $\mathbf{n}_A = -\mathbf{n}_B$, then iA and iB are on opposite sides and do not intersect in the neighborhood.

Figure 11.45 Line and polygon classifications.

Figure 11.46 Tangent vector convention for two-dimensional objects.

Figure 11.47 Normal vector convention for three-dimensional solids.

We have taken a somewhat rigorous approach to solving the problems of set-membership classification. Later discussion will reveal the importance of other concepts that we touched on only briefly here (see Figure 11.48). These concepts pervade computer graphics and CAD/CAM modeling algorithms and other more abstract applications, offering ample opportunity for further study and research.

Figure 11.48 Problems for set-membership classification.

Boolean Models

If we represent a solid object by the Boolean combination of two or more simpler objects, then the representation is a *Boolean model.* If A, B, and C denote solids and if $C = A\langle OP\rangle B$, where $\langle OP\rangle$ is any regularized Boolean operator, then $A\langle OP\rangle B$ is a Boolean model of C. Remember that A, B, and C must be of the same spatial dimension. From now on, we will assume that all Boolean operations are regularized, so that, for conciseness, we can drop this term as a qualifier. The symbols $\cup$, $\cap$, and $-$ will denote the regularized Boolean operators: union, intersection, and difference, respectively.

A Boolean model is a *procedural model* (see Figure 11.49). For example, let us assume that we know the size, position, and orientation of A, B, and C, perhaps as a list of vertex coordinates. The Boolean model of D is $D = (A \cup B) - C$. This Boolean statement defining D says nothing quantitative about the new solid it creates. It only specifies the procedure for combining the primitive constituents. It does not tell us the coordinates of the vertices of the new solid or anything about its edges or faces. We may know all there is to know geometrically and topologically about A, B, and C, but all we know about D is how to assemble it. Therefore, this leads us to say that the Boolean model is a procedural representation, or an *unevaluated model.* If we want to know more, then we must evaluate the Boolean model, compute intersections to determine new edges and vertices, and analyze the connectivity of these new elements to determine the model's topological characteristics.

$D = (A \cup B) - C$

Figure 11.49 A simple procedural model.

The boundary of D has 32 vertices, 48 edges, and 18 faces. The undirected connectivity matrix of the vertices is a 32×32 array containing 1,024 elements. Since each vertex in this model is connected to exactly 3 others, there are only 96 nonzero elements in the matrix. A boundary evaluator routine uses this information to construct the boundary representation.

The binary tree for this model is shown in Figure 11.50. The leaf nodes are the primitive solids, with Boolean operators at each internal node and the root. Each internal node combines the two objects immediately below it in the tree and, if necessary, transforms the result in readiness for the next operation. The constructive representation of an object is a direct translation of the Boolean operators into a binary tree structure.

How are the primitives modeled? In many systems, they are stored as a graph-based model and become a unit template or parameterized shape to be scaled and positioned as a leaf node of the model's binary tree. Or a primitive may be a Boolean combination of directed surfaces or half-spaces. A

Figure 11.50 The binary tree for $D = (A \cup B) - C$.

directed surface is a surface whose normal at any point determines the inside and outside of the primitive solid. An unbounded surface divides Cartesian space into two unbounded regions; each region is a *half-space*. The Boolean intersection of an appropriate set of half-spaces can form a closed three-dimensional solid.

We may use directed surfaces defining half-spaces to construct an entire model. Each directed surface is given by an implicit equation of the form $f(x,y,z) = 0$. The function is zero at the surface and positive inside the object. Thus, we define a complex object by the union of the intersection of directed surfaces:

$$F = \bigcup_{j=1}^{m}\left(\bigcap_{i=1}^{n} f_{ij}\right) \tag{11.56}$$

where the f_{ij} are directed surfaces or half-spaces. The f_{ij} describing the surfaces are usually implicit functions of the form $f_{ij}(x,y,z) = 0$. These functions define surfaces that divide space into two half-spaces, and we can easily evaluate points as satisfying $f_{ij}(x,y,z) \geq 0$ or $f_{ij}(x,y,z) < 0$, which we then classify as belonging to the object's point set or to its complement, respectively. For a fuller treatment see Mortenson (1989). Parametric surfaces do not formally define half-spaces, because they do not divide space into two parts in a direct analytical way. They would first have to be converted into implicit surfaces.

Other systems operate with solid, bounded primitives. At any node of the binary tree, two valid solids combine to produce a third valid solid. The Boolean model of even relatively complex solids is generated quickly in these systems, and it has a very compact and concise data structure.

Figure 11.51 is another example of a Boolean model of a solid object. We let Π denote a primitive object. Here Π_1 is a rectangular parallelepiped and Π_2 is a right circular cylinder. T_1 denotes a transformation that scales and positions the primitives. First, the union operator combines $T_1\Pi_1$ and $T_2\Pi_2$ to produce S_1, a hub-like shape with rectangular ears. Next, $T_3\Pi_1$ and $T_4\Pi_2$ combine to produce S_2, similar to S_1 but smaller in size. The Boolean difference between S_1 and S_2 creates a circular hole and pair of keyways in S_1 to produce S_3. Then the intersection of S_3 with $T_4\Pi_2$ reforms the outer surface of the ears, giving them a radius equal to that of $T_4\Pi_2$'s, producing S_4. S_4 and S_5 combine to produce S_6 and add a further refinement to the geometry of the ears. S_6 and S_7 combine to produce S_8, the final object shape. Notice that S_7 defines a pair of holes; these are "drilled" in S_6 to produce S_8.

Although this model represents a modest accomplishment, we can appreciate the compactness and conciseness that a Boolean model achieves. To demonstrate this more explicitly, we express S_8 as a string of symbols

Figure 11.51 A Boolean model of a mechanical part.

$$
\begin{aligned}
S_8 &= S_6 - S_7 \\
&= (S_4 - S_5) - (T_8\Pi_2 \cup T_9\Pi_2) \\
&= (S_3 \cap T_2\Pi_2) - (T_6\Pi_2 \cup T_7\Pi_2) - (T_8\Pi_2 \cup T_9\Pi_2) \\
&= (S_1 - S_2) \cap T_5\Pi_2 - (T_6\Pi_2 \cup T_7\Pi_2) - (T_8\Pi_2 \cup T_9\Pi_2) \\
&= [(T_1\Pi_1 \cup T_2\Pi_2) - (T_3\Pi_1 \cup T_4\Pi_2)] \cap T_5\Pi_2 \\
&\quad - (T_6\Pi_2 \cup T_7\Pi_2) - (T_8\Pi_2 \cup T_9\Pi_2)
\end{aligned}
\tag{11.57}
$$

Then we use the distributive property of the intersect operator to obtain

$$
\begin{aligned}
S_8 = {} & (T_1\Pi_1 \cap T_5\Pi_2 \cup T_2\Pi_2 \cap T_5\Pi_2) - (T_3\Pi_1 \cap T_5\Pi_2 \cup T_4\Pi_2 \cap T_5\Pi_2) \\
& - (T_6\Pi_2 \cup T_7\Pi_2) - (T_8\Pi_2 \cup T_9\Pi_2)
\end{aligned}
\tag{11.58}
$$

This symbol string, including the nine transformations, is the procedural, unevaluated geometric model. We note that this symbol string is not unique. Although Π_1 and Π_2 are fixed primitive types, the transformations T_i vary depending on the size and shape of the object modeled. We also note that for this example, all operations involving $\cap T_2\Pi_2$ except $T_1\Pi_1 \cap T_5\Pi_2$ are trivial, since all the other bodies lie entirely within $T_5\Pi_2$. Thus,

$$T_2\Pi_2 \cap T_5\Pi_2 = T_2\Pi_2$$

$$T_3\Pi_1 \cap T_5\Pi_2 = T_3\Pi_1$$

$$T_4\Pi_2 \cap T_5\Pi_2 = T_4\Pi_2$$

A sophisticated modeling system will detect the condition and will not process these trivial operations.

Computing intersections is of central importance in evaluating the boundary of a Boolean model. If we define the complete boundaries of two primitives or subobjects, then the complete boundary of $A\langle OP\rangle B$ is the sum of the segments of A and B boundaries. These segments are the active regions on A and B and are themselves bounded by the intersection of the original boundaries of A and B.

Let us review the effect of the three Boolean operators on a set of two or three primitive objects. We consider an object, including primitives, to be a collection of points, some comprising the interior and some the boundary. The union of two objects, say, $A \cup B$, is a collection of points that belong to A, B, or A and B. The difference of two objects $(A - B)$ is a collection of points that belongs to the surface and interior of A but is not inside B, a collection of points that belongs to the surface of B and is inside A. Figure 11.52 shows two- and three-dimensional examples.

Figure 11.52 Examples of union and difference.

The operation $A \cap B$ is the set of points belonging to both A and B. These points are of three types: those interior to both A and B, those on boundary regions of A interior to B, and those on boundary regions of B interior to A (see Figure 11.53). Only points 1 and 2 of $A \cap B$ in Figure 11.53a are important, because they are the bounding points of the active regions of the bounding curves of A and B. There may be more than two points of intersection of the two-dimensional boundary curves, depending on their shapes.

The order in which we perform combining operations in a Boolean model is important; for example, in general

$$A \cup B - C \neq A - B \cup C \tag{11.59}$$

However, we can randomly mix the order within a sequence subset where all the operators are the same type *without* affecting the resulting shape. Thus,

Figure 11.53 The intersection operation.

$$A \cup B - C = B \cup A - C \tag{11.60}$$

or

$$\begin{aligned} A - B - C - D \cup E &= A - (B \cup C \cup D) \cup E \\ &= A - (B \cup D \cup C) \cup E \\ &= A - (D \cup B \cup C) \cup E \end{aligned} \tag{11.61}$$

and so on. To understand order dependence better, see again Figure 11.39.

A geometric-modeling system must accommodate many interesting, difficult, and potentially pathological situations when evaluating the boundary of a complex shape. Some of these situations are shown in Figure 11.54. The union of two disjoint primitives (that is, nonintersecting) appears in Figure 11.54a; the difference of two disjoint primitives appears in Figure 11.54b; the union of two primitives where one wholly contains the other appears in Figure 11.54c; the difference of two primitives where the positive primitive wholly contains the negative primitive appears in Figure 11.54d; the difference of two primitives where the negative primitive wholly contains the positive primitive appears in Figure 11.54e; the difference of two primitives that creates two or more new objects appears in Figure 11.54f; the union of two primitives that are tangent appears in Figure 11.54g; and the union of two primitives that creates inner loops or cavities (bubbles) appears in Figure 11.54h. Not shown in this figure are the operations $A \cup A$ or $A - A$. Although the preceding situations are two-dimensional, each also has a direct three-dimensional analog.

Some generalizations are in order (see Figure 11.55). Modeling systems use algorithmic equivalents to check the validity of results or to speed computations. If two closed planar curves intersect, they will intersect at an even number of points—a tangent condition is not counted as an intersection (see Figure 11.55a). If two closed coplanar curves A and B do not intersect, and if

Figure 11.54 A variety of Boolean modeling situations.

a point on B is inside curve A, then curve B is entirely inside curve A—the converse is also true (see Figure 11.55b). If a closed curve intersects the bounding surface of a three-dimensional shape, it will intersect it at an even number of points—again, tangents are not counted (see Figure 11.55c). If an unbounded plane P intersects the closed bounding surface of a three-dimensional shape S, then the intersection will consist of one or more closed nonintersecting curves (see Figure 11.55d).

Constructive Solid Geometry

Constructive solid geometry (*CSG*) is the name for a set of modeling methods that defines complex solids as compositions of simpler solids. Boolean operators are used to create a procedural model of a complex solid. The model is represented by a binary tree of Boolean operations, where the leaf nodes are simple primitive shapes, sized and positioned in space, or directed surfaces (half-spaces), and the branch nodes are the set operators (union, difference, and intersection). For a good entry into the literature of this method, see Requicha and Rossignac (1992). They survey and critique the field of solid modeling, including mathematical foundations, representations, algorithms, applications, user interfaces, and systems. Requicha, Voelcker,

and others on the Production Automation Project at the University of Rochester introduced many of the most important concepts of CSG, including regularized Boolean operators, primitives, boundary-evaluation procedures, and point-membership classification. Although we have already discussed many of these concepts, we will review them here in the context of CSG. Also see Mortenson (1989), Tilove (1977), among many others.

Requicha (1980) viewed CSG as a generalization of cell decomposition. In cell decomposition models, we combine individual cells using a *gluing operation*, a limited form of the union operator where we join components at only perfectly matched faces. Constructive solid geometry operators are more versatile, since boundaries of joined components (primitives) need not match, and interiors need not be disjoint. Furthermore, CSG uses all the reg-

Figure 11.55 Four general properties of Boolean models.

ularized Boolean operators—union, difference, and intersect—so that we can add material as well as remove it.

Constructive solid geometry representations of complex solids are ordered binary trees whose leaf or terminal nodes are either primitives or transformations. The nonterminal nodes are either regularized Boolean operators or transformations that operate on their two subnodes (or subsolids). Each subtree of a node (not a transformation leaf) represents a solid resulting from combining and transforming operations indicated below it. The root, of course, represents the final object. Let us look at a simple example.

In Figure 11.56, the four leaf nodes represent the primitives Π_1 and Π_2 and the translation Δx. The two internal nodes represent results of the operations $(\Pi_1 - \Pi_2)$ and $\Delta x \Pi_2$. The root node represents the final object. Note that the primitive and intermediate objects are valid bounded solids. Of course, transformations are not limited to rigid motions. The full range of scaling and symmetry transformations is theoretically possible. Such transformations are limited only by the capabilities of subsequent Boolean algorithms.

If the primitive elements of a modeling system are valid bounded solids and the combining operators are regularized, then the resulting solid models

Figure 11.56 Constructive solid geometry representation.

are valid and bounded. However, a modeling system is possible that allows user-defined primitives. Here, either the user or the system must verify the validity of a model.

The most common approach in contemporary modeling systems is to offer a finite set of concise, compact primitives whose size, shape, position, and orientation are determined by a small set of user-specified parameters. For example, most systems provide a block-type primitive, for which we can specify the length, width, height, and initial position. The modeling system then checks the validity of the parameters (that is, the length, width, and height values must all be positive real numbers). Other common primitives are the cylinder, sphere, cone, and torus. These and other examples are shown in Figure 11.57 with their corresponding parameters (excluding position and orientation).

The more sophisticated modeling systems can also generate quasi-primitives with sweeping- or extruding-type operators. Even here, the system controls validity by checking input parameters. The number of primitives, however, is not a sign of the descriptive power of a modeling system. For example, the block and cylinder alone have the same descriptive power as the primitive set consisting of the block, cylinder, wedge, fillet, cylindrical segment, and tetrahedron if both sets have the same combining and transforming operators.

The primitives themselves are usually represented by the intersection of a set of curved or planar half-spaces. For example, the primitive block is represented by the regularized intersection of six planar half-spaces and the cylinder by the intersection of a cylindrical half-space and two planar half-spaces (see Figure 11.58). Only three of the six half-spaces defining the primitive block are shown in Figure 11.58a. Arrows indicate the direction of the material and not necessarily the direction of the surface normals. (If normals are part of the representation, they can be assigned during the boundary-evaluation phase.) Purely graph-based primitives are also possible.

The Boolean operators used by CSG systems are the familiar threesome: union, difference, and intersect. However, some modelers include other operators, such as *glue*, which is a specialized version of the basic three. The complement operator is not used, because in an otherwise regularized scheme it produces nonregular objects. Also, recall from the discussion of set theory and Boolean operators that union and intersect are commutative (that is, $A \cup B = B \cup A$ and $A \cap B = B \cap A$), but the difference operator is not (that is, $A - B \neq B - A$).

More powerful modeling systems often generate two representations of a solid. The first is the *procedural* or *constructive representation* exemplified by a binary tree data structure linking primitives and successive subsolids by

Figure 11.57 Primitive solids.

using combining operators and transformations, as we just discussed. The second is the *boundary representation,* which describes the faces, edges, and vertices of the boundary of the solid. This description itself has two forms—a topological representation of the connectivity of the boundary elements and numerical data describing the shape geometry and position of these elements. The data defining the object's topology usually point to the geometric shape data.

The boundary representation is computed from the constructive representation by a set of algorithms called the *boundary evaluator.* The boundary

evaluator determines where component faces are truncated and new edges and vertices are created or deleted. Where boundary elements overlap or coincide, the evaluator merges them into a single element and thus maintains a consistent, nonredundant data structure representing a solid's boundary.

New edges are created where surfaces of two combined solids intersect. The boundary evaluator finds these intersections and then determines by set-membership classification which segments of the intersection are actual edges of the new solid. The new edges are terminated by new vertices, that is, where the edge intersects a surface. Again, the vertices must be classified. We

Figure 11.58 Primitives as intersections of half-spaces.

will look a little deeper into some of these concepts of boundary evaluation, using the example in Figure 11.59.

First, we intersect each surface of A with each surface of B; this operation assumes that both surfaces are unbounded. In our example, there are 36 intersection combinations, with only 10 producing intersection edges. (Note that simple tests or heuristic techniques can dramatically reduce the number of candidates for intersection.) We produce edge e_{ab} by intersecting face a and face b, and list the edges produced this way as tentative edges, or t-edges, for reasons that will soon be apparent. The list of t-edges is a superset of the actual edges of the new solid C. The intersecting pair of surfaces defining face a and face b are unbounded, making t-edge e_{ab} unbounded, also.

Next, we intersect all t-edges with all the faces to produce points (potential vertices) that segment the t-edges. Edge e_{ab} is divided into three segments by two of the vertical plane faces of B. We classify these segments as outside, inside, or on the boundary of C. Only segments on the boundary are real edges. Doing this in steps, we first classify each t-edge segment with respect to each primitive. Then we use the classifications instead of the primitives, processing them through the model's binary tree and reclassifying them at each successive node. At the root node, the segments of t-edges on the boundary are real edges of C.

We need to know which points near each segment of a t-edge are inside the solid and which are outside. Knowing only that a segment is on a primitive is inadequate for classifying it at higher levels in a model's binary tree. Using a

Figure 11.59 Boundary evaluation.

neighborhood model, consisting of points close to the segment, we can indicate which points are inside and which ones are outside the solid. Figure 11.60 shows a neighborhood model of segment 2 of e_{ab}. Here, a small disk symbolizes the neighborhood, and the shaded area indicates points inside the solid. Our perspective is from an end of the segment. The neighborhood model is defined by the faces that bound the model at this edge. We use neighborhood models of two primitives (or subsolids) and the corresponding operator to combine these neighborhoods to determine if the segment is outside, inside, or on the next highest level of the tree. More examples are presented in Figure 11.61. We create a neighborhood at any node in the tree by applying the indicated operator to the neighborhoods of the two subnodes.

We can now make some generalizations. First, faces of a new solid are a subset of the faces of the combining solids. We can modify but cannot create faces, unless we admit sweep operators. However, we can create new edges and vertices, and we can delete any element type.

We have seen that the procedural representation of a complex solid is compact and we can generate it rapidly. This is true of the solid created by the ROMULUS system (circa 1980s) in Figure 11.62. Extensive computations are necessary to generate the boundary representation from a procedural representation, but the boundary representation contains a wealth of useful information for computing global properties, graphic display models, and other applications. Again, the procedural form is an unevaluated model, and the boundary form is an evaluated model. The most versatile and powerful modeling systems use both.

Figure 11.60 Neighborhood model.

Figure 11.61 Combining neighborhood models.

The next section continues the discussion of boundary evaluation. A review of the sections on set-membership classification and Boolean operators will be helpful.

11.4 BOUNDARY MODELS

The objective of a boundary model (or *b-rep*) is to build a complete representation of a solid as an organized collection of surfaces. We ean represent a solid as a union of faces (surfaces), bounded by edges (curves), which in turn are bounded by vertices (points). Faces, then, lie on surfaces; edges lie on curves; and vertices are at edge endpoints. A boundary model stores the

Figure 11.62 A ROMULUS-created solid.

mathematical data of the surface geometry on which the face lies, the curve geometry on which the edge lies and which bounds the face, and the point geometry (the coordinates) of the vertices.

A simple b-rep method represents solids by means of a set of plane polygonal faces (sometimes called *polyhedral* or *tessellated* models). The face edges are straight lines. This is obviously a special case of boundary representation where curved surfaces and edges are approximated by planes and straight lines. We are assured of a closed boundary surface by certain topological relationships between the elements of a polyhedral net on the surface. Each edge is adjoined by two and only two faces and vertices, with an equal number of edges and vertices on the ring surrounding any face. Similarly, an equal number of edges and faces surround any vertex (with the exception of the apex of a conical-type surface). These conditions rule out open, shell-like surfaces. Finally, the boundary of a solid must satisfy certain conditions: It must be closed, orientable, non-self-intersecting, bounding, and connected.

Generalized Concept of a Boundary

To begin the discussion of boundaries of physical objects, let us review certain concepts. Cartesian space of any dimension n can be denoted by the symbol E^n. Thus, ordinary three-dimensional space is E^3, and two-dimensional space is E^2. Points in E^2 are defined by two real numbers, in E^3 by three real numbers, and in E^n by n real numbers. Coordinate space is unbounded, because the numbers defining coordinate points of the space can take any real values from $-\infty$ to $+\infty$.

A *region* R^n is a finite, bounded portion of space E^n. The points that comprise any region either lie entirely within the region or on its boundary. Thus, the set of points denoted by R can be divided conveniently into two subsets R_i and R_b, where R_i is the set of points in the interior of a region and R_b is the set of points on its boundary. We write this as

$$R = [R_i, R_b] \tag{11.62}$$

All physical objects are three-dimensional regions R^3. The boundary of an R^3 region is a closed surface. Given any point in space, it is obviously either outside the region, in the boundary set R_b, or in the interior set R_i, and this applies to regions of any dimension.

A curve is a one-dimensional region R^1. It has two points in the set R_b^1 unless it is a closed curve; then there are no points in R_b^1. All other points of the curve are in R_i^1.

A surface is a two-dimensional region R^2. An ordinary open surface is always bounded by a closed curve. On the surface within this curve may

occur one to n nonintersecting closed curves or loops. All the points on all the loops comprise the set R_b^2. All other points of the curve are in R_i^2 (see Figure 11.63).

Now we expand our horizons a little and also modify the notation scheme. Let $R^{m,n}$ be a region of E^n, where m = dimensionality of R and n = dimensionality of the space E in which R is located. Note that $m \leq n$. Then

$$R^{m,n} = [B^{m-1,n}, I^{m,n}] \tag{11.63}$$

where $B^{m-1,n}$ is the set of points on the boundary of $R^{m,n}$ and $I^{m,n}$ is the set of points in the interior of the region. The set $B^{m-1,n}$ is itself a proper region because it can be decomposed into a set of points on its boundary and a set on its interior. Table 11.2 defines allowable regions in E^3. Of course, we can construct a similar table for regions in spaces of any dimension. The general concept of a boundary is contained in the expression $B^{m-1,n}$.

Any point in space has one and only one of the following three properties with respect to any region $R^{m,n}$:

1. It is inside the region; that is, it is a member of the set $I^{m,n}$.
2. It is on the boundary of the region; that is, it is a member of the set $B^{m-1,n}$.
3. It is outside, not a member of the set $R^{m,n}$.

For a homogeneous region, when $m = n$ we can imply $I^{m,n}$ by an explicit formulation of $B^{m-1,n}$. Thus, for a homogeneous solid in E^3, the explicit definition of the $B^{2,3}$ of the solid is necessary and sufficient for the definition of $R^{3,3}$ of the solid. $B^{2,3}$ is the outline or boundary surface of the solid, where points on the inside (that is, in $I^{3,3}$) are implied by $B^{2,3}$.

Figure 11.63 A plane figure and its boundaries.

Table 11.2 Allowable regions in E^3

Order	Class	$B^{m-1,n}$	I^{mn}
$R^{0,3}$	Point	The point itself	No interior points
$R^{1,3}$	Curve	The two endpoints	The set of points of the curve except for the two boundary points in $B^{0,3}$
$R^{2,3}$	Surface	One or more closed curves that define the boundaries of the surface	The set of all points on the surface except those on the bounding curves in $B^{1,3}$
$R^{3,3}$	Solid	One or more closed surfaces that define the boundaries of the solid	The set of all points within the solid except those on the bounding surfaces in $B^{2,3}$

B-reps

The boundary of a solid separates points inside from points outside the solid. The boundary is the primary interface between the solid and the surrounding environment. Reflection of light from a solid and, therefore, the solid's appearance are determined by the surface properties of shape, color, and texture. Even the bounding surface of transparent objects influences light reflection. The bounding surface of a solid is where it contacts other solids. Manufacturing processes are concerned with the interaction of the surface envelope of a tool path with the surface of a part's raw-stock shape.

Boundary representations (*b-reps*) of solids must satisfy certain conditions. The conditions for a well-formed surface are that it must be closed, orientable, non-self-intersecting, bounded, and connected. We have already distinguished between the boundary of an object (including two-dimensional objects and objects of arbitrary dimensions) and its interior points, and we have discussed a boundary evaluator for CSG representations. Thus, we have most of the ingredients for boundary representations.

We will refine our notation of the bounding surface of a solid and then investigate a specific type of boundary representation. It is computationally convenient for us to segment the boundary surface of a solid into faces or patches, with each face or patch in turn bounded by a set of edges and vertices. Faces should satisfy the following minimal set of conditions:

1. A finite number of faces defines the boundary of a solid.
2. A face of a solid is a subset of the solid's boundary.
3. The union of all faces of an object defines its boundary.

4. A face is itself a subset or limited region of some more extensive surface.
5. A face must have a finite area and be dimensionally homogeneous.

These conditions are illustrated by the boundary surface of the simple cube and cylinder in Figure 11.64. Both objects have a finite number of faces. The cube's boundary is conveniently segmented into six planar faces, defined by the discontinuities at the edges, and it coincides with our intuitive notion of a boundary. The cylinder's boundary is more arbitrarily segmented, with the end planar faces having the only natural segmentation and the cylindrical surface requiring a division into faces in any number of ways. (That is, assuming segmentation by vertical edges, we can have any number of faces of equal or unequal size.)

Figure 11.64 Faces defining the boundary of a solid.

No single face of either the cube or the cylinder defines the entire object; therefore, every face is a subset of the complete boundary. Each face is also a subset or limited (bounded) region of a more extensive surface. Thus, each face of the cube is a bounded region of an infinite plane, and so are the circular disks or end faces of the cylinder. How about the cylindrical surface itself? Each face of the cube and cylinder has a finite area and is dimensionally homogeneous. (There are no dangling edges or isolated points.)

We can represent the planar faces simply by their bounding edges. However, curved faces require more information. For example, the curved face may be a region of a Bézier surface, where we define this surface by its characteristic polyhedron, and we similarly define the curve on this surface that delimits the actual face with its characteristic polygon. There are computational devices or conventions to indicate on which side of an edge or delimiting curve the face lies. We have investigated several, and Figure 11.65 illustrates one such convention. Here, the face-bounding curve is parameterized in a consistent direction so that the vector $\mathbf{n} \times \mathbf{t}$ points to the face side of the curve.

The data structure of a b-rep is best described by a hierarchical graph listing the faces, edges, and vertices that form the boundary of a solid. The complete bounding surface of a solid consists of one or more shells, depending on the presence of internal voids, or disconnected islands (if imaginary solids are admitted). Each *shell* consists of one or more faces, where a *face* is a connected subset of a surface bounded by a closed connected set of edges. An *edge* is a segment of a curve bounded by two points, the vertices.

We can usually segment the boundary of an object into faces, edges, and vertices in an unlimited number of ways, so there is no unique b-rep of an object. As Figure 11.66 shows, there are many valid boundary representa-

Figure 11.65 Face boundary convention.

tions for the cube and cylinder. The cylinder to the right of each cube is its topological equivalent, having the same boundary-segmentation scheme. Geometric data for each object are different, of course. Strictly speaking, a cylinder and cube have the same topology regardless of how their bounding surfaces are segmented. Sometimes, to avoid confusion when discussing an object's boundary segmentation, we speak of its *topological* or *combinatorial* structure. We use the term *metric information* to mean geometric information (for example, the coordinates of a point).

In general, topological structure and metric information are not independent of each other. For example, in Figure 11.67, if we make vertex 1 coincide with vertex 2, then the object is no longer valid. We conclude that a valid combinatorial or topological structure does not in itself guarantee a valid object. We must reestablish validity conditions for both metric and topological data. These conditions depend on characteristics of the representation

Figure 11.66 Boundary representations are not unique.

which the construction processes allow (for example, many schemes are limited to faces defined by three edges).

There are powerful b-rep schemes where validity must be determined by means other than computing face, edge, and vertex relationships. Ordinarily, these schemes describe a solid as the union of very general faces embedded in extended surfaces, where the solid's edges are defined by the intersections of these surfaces. These schemes are not true directed half-spaces, since their net Boolean intersection is usually not related to the object model, and surface behavior external to and remote from the face is not important. We can construct such models in two different ways—assemble surfaces to create solid primitives that then, as in CSG schemes, combine to create more complex shapes, or omit the primitive phase altogether and construct complex shapes directly by assembling and intersecting appropriate surfaces. The procedures using Boolean operations on b-reps are commonly known as *boundary evaluation* or *merging* algorithms. Let us look at some two- and three-dimensional examples.

The following statement implies the shape of a complex object b:

$$S = A \cup B \tag{11.64}$$

In boundary-representation schemes, this means that the bounding surface of S consists of a set of appropriate regions of the bounding surfaces of primitives A and B. These regions are the active regions, and we denote them by a_A and a_B. We determine these active regions with two conditions. First, the regions are bounded by the intersection of the boundary surfaces of A and B; that is, $bA \cap bB$. In the two-dimensional example in Figure 11.68, the intersection consists of points 1 and 2, which are obviously independent of the type of combining operation. Second, we select the active region of a spe-

Figure 11.67 Interdependence of topology and geometry.

cific primitive based on its classification being inside, outside, or on the boundary of other combining primitives.

The combining sequence of the three primitives represented by the binary tree (see Figure 11.69) reduces to the linear sequence $A \cup B - C$. Each primitive is in position for performing the Boolean operations (see Figure 11.69a). Because this is a two-dimensional example, the primitives are closed parametric curves. The zero point of the parametric variable is shown, along with an arrow indicating the direction of parameterization. Counterclockwise is the positive direction of parameterization for curve loops enclosing a *solid* region. Curve loops parameterized clockwise enclose or define *holes*.

The union of A and B in two steps is shown in Figure 11.69b. Here we find the intersection of the two bounding curves of A and B, points u_1 and u_2:

$$bA \cap bB = [u_1, u_2] \tag{11.65}$$

This operation divides both curves into two segments. For the union operation, the active region of bA is that segment of bA lying outside B. The active region of bB is that segment of bB lying outside A. To the right in this figure are active regions on the unit parametric intervals of the respective boundary curves. The active region on bA, denoted a_A, consists of two intervals—one from 0 to u_{1A} and the other from u_{2A} to 1. There is one active interval a_B on bB—from u_{1B} to u_{2B}.

We complete the shape definition by using the results of $A \cup B$ and taking its Boolean difference with C (see Figure 11.69c). Here we find the active regions on bC as well as the revised regions on bA and bB. Computing intersections is, again, the first step. In doing this, we find only the intersection of bC with the active regions of bA and bB. For our example,

$$(bC \cap a_A) \cup (bC \cap a_B) = [u_3, u_4] \tag{11.66}$$

Once we find u_3 and u_4, we select the proper interval a_C on bC and determine the effect on a_A and a_B. Because this is a difference operation, a_C is that part of bC that is inside the result of $A \cup B$. We modify a_A and a_B by retain-

Figure 11.68 Boundary intersection.

Figure 11.69 Two-dimensional boundary representation.

ing only those portions lying outside C. The effect on a_A, a_B, and a_C is shown by the bold sections on the unit parametric intervals at the right in Figure 11.69c. The union of the active intervals after any combining operation always defines one or more closed loops.

A similar situation occurs with three-dimensional objects. The active surface regions (faces) on all the primitives combined to form a solid will define

a closed surface. For example, in Figure 11.70 the shape consists of the union of a sphere with a skew-truncated cylinder. We denote the cylinder by Π_1 and the sphere by Π_2. We define the cylinder by the combination of three surfaces whose intersection curves outline and bound face regions. The union of these face regions defines the boundary surface of the cylinder. Points bound active regions on curves, and curves bound aetive regions on surfaces. The active regions of the surfaces in this figure are shaded. We decompose the primitive Π_1 (cylinder) into its constituent surfaces and map each of these onto a unit square in parameter space. Their respective active regions are outlined. (Remember that we parameterize loops defining holes in the clockwise direction.) We see also the primitive Π_2 (sphere). The intersection of $b\Pi_1$ and $b\Pi_2$ is a closed-space curve, which appears twice in the parametric space representations. First, the intersection curve is associated with Π_2 or surface D; second, it is associated with constituent surface A of Π_1. In each case, the curve is a function of a third parametric variable t. The functions of t map the curve into the unit square in u,w parameter space from the unit interval in t parameter space.

11.5 SPACE-PARTITIONING MODELS

We can take a common object—a coffee mug, for example—and imagine decomposing it into separate pieces so that each piece is easier to describe than the original (see Figure 11.71). As the first step, we detach the handle from the mug. This seems natural and topologically reasonable, because there are then two parts, each with a simply-connected topology, instead of one multiply connected object. Next, we detach the bottom of the mug. Now there are two parts with a simply connected topology and one with a multiply connected topology. If necessary, we continue to decompose any or all of these parts until we meet some predetermined describability criterion. This process is known as *cell decomposition*. We can represent any solid as the sum or union of a set of cells into which it is divided. Cell decomposition is useful where the total object may not be amenable to representation, but its cells are. There are many ways of decomposing a solid into constituent cells; none is unique, but all are unambiguous.

Spatial-occupancy enumeration is a special case of cell decomposition where cells are cubical in shape and located in a fixed spatial grid. As the size of the cube decreases, this method approaches the representation of a solid body as a set of contiguous points in space. To define a solid using spatial-occupancy enumeration requires a convenient way to represent this set of cubical cells. One way is simply to list the coordinates of the centers of the

Figure 11.70 Three-dimensional boundary representation.

Figure 11.71 Cell decomposition example.

cells; a solid object is thus a set of adjacent cells. Cell size determines the maximum resolution of the model.

Two advantages of representing a solid by spatial arrays are often cited: It is easy to access a given point, and spatial uniqueness is assured. There are also disadvantages: There is no explicit relationship between the parts of an object, and such schemes usually demand large amounts of data storage.

A cell in a spatial array is either occupied by a piece of the solid or it is not. A cell can be marked with the binary 1 or 0 to indicate spatial occupancy. Early schemes of this type were highly redundant, because all cells of an object were marked, even though it was highly likely that any particular cells had the same state as cells adjacent to it. Only close to an object's boundary do chances increase for a change of state.

Quadtrees and their three-dimensional analogs, *octrees*, suggest a way to use spatial-occupancy enumeration more efficiently. Let us look at quadtrees first. The quadtree representation of a two-dimensional object is based on recursive subdivision of a square array into quadrants. Each node represents a square region on a plane. In computer-graphic applications, this may be the

screen plane of the display. Whereas each node of a binary tree has two descendants, each node of a quadtree has four.

In Figure 11.72, we superimpose a square on an arbitrary two-dimensional object. If the object does not uniformly cover the square, then we subdivide the square into four equal quadrants. If any of the resulting quadrants is full or empty, then we do not need to subdivide it further, which is true for quadrant 3 in this example. If any of the resulting quadrants is partially full, we subdivide it again into quadrants. We continue to subdivide partially full quadrants until the resulting regions are either full or empty or until we reach some predetermined level of resolution. We can arbitrarily declare the partially full quadrants as either full or empty according to a convention we establish.

The root node of the quadtree represents the entire array. It is the *universe element*, and *leaf nodes* represent regions that require no further subdivision. Such regions have standard sizes and positions related to powers of 2. The number of nested subdivisions from a given node to the universe element determines the node's level in the quadtree and also the size of the region represented by the node. If the height of the tree is n, then the maximum potential array size is $2^n \times 2^n$. For the example in Figure 11.72, $n = 3$, and because of the efficiencies of quadtree representation, there are only 33 nodes instead of 64 (that is, $2^3 \times 2^3$). The resolution necessary to accurately represent an object depends on the difference in size of gross and fine features of the object and curvature of the boundary. The greater the magnitude of n, the greater the resolution. Reducing an object model into a quadtree representation is *quadtree encoding*.

Octree encoding is an extension into three dimensions of quadtree encoding. Meagher (1982b) developed a solid-modeling method based on octree encoding for the high-speed manipulation, analysis, and computer-graphic display of solid objects. His approach uses a spatially presorted 8-ary hierarchical tree structure, the octree, to represent solids. It also uses algorithms that grow only linearly with object complexity by taking advantage of the inherent data presorting in the tree structure. The octree encoding of a model proceeds similarly to quadtree encoding. Here we recursively subdivide a cubical region into *octants* or eight cubical regions (see Figure 11.73). Each node of an octree that is not a leaf node has eight descendants. If the height of the tree is n, then the maximum potential array size is $2^n \times 2^n \times 2^n$. For the example in the figure, $n = 3$.

Octree and quadtree encoding schemes suggest many interesting possibilities for representing, analyzing, and displaying complex solid objects. All computations on these models are based on integer arithmetic, making analysis algorithms fast and amenable to parallel processing. Meagher and

Figure 11.72 Quadtree representation.

Figure 11.73 Octree representation.

others have demonstrated algorithms that translate, rotate, and scale octree models; that combine them using Boolean operators; and that compute geometric properties and perform interference analysis. These schemes, and variations on them, offer heuristic tools for rapid preprocessing or sorting models based on other techniques.

12

RELATIONAL GEOMETRIC SYNTHESIS

Relational geometric synthesis is a new approach to geometric modeling and computer-aided geometric design. It offers an innovative, object oriented, relationally structured environment that greatly improves the efficiency of the design process and resolves many long-standing problems. This chapter begins with an introduction to the characteristics of relational geometric synthesis, followed by a discussion of relational model structure, model evaluation, serialization, and relational entities. A detailed example and several applications are presented. This chapter concludes with a discussion of open issues in relational geometry.

This chapter was co-authored by Dr. John S. Letcher, Jr., principal developer of relational geometric synthesis, and his associates D. Michael Shook, Seth H. Porter, and George A. Dickson. They can be contacted at: Aero-Hydro Inc., P.O. Box 684, Southwest Harbor, ME 04679, (207) 244-7347, http://www.aerohydro.com.

12.1 INTRODUCTION

Relational geometric synthesis (*RGS*)—or *relational geometry* (*RG*)—is an object-oriented framework for computer-aided geometric design. RG provides a new structure for CAD programs and CAD models that solves important outstanding problems in geometric modeling and confers profound benefits in practical design. Compared with conventional CAD, RG is characterized by:

1. A *synthetic* design approach, building up from simpler to more complex geometry—from points, to curves, to surfaces—through sequential constructions.
2. Representation and retention of a large variety of constructive relationships between the geometric elements (objects) comprising a geometric model.
3. Support of a rich and easily extensible set of methods for object construction.
4. Methods for construction of point, curve, and surface objects explicitly and durably embedded in other objects; for example, points and curves embedded in surfaces.
5. Methods for construction of accurate and durable joins between surface objects without performing surface-surface intersections.
6. Facility in the construction of parametrically variable models involving complex, free-form and sculptured surfaces combined with durable topology.
7. Compact serialization and storage of complex geometries, in the textual form of a geometric-representation language.

These characteristics lead to many advantageous applications for relational geometry throughout computer-aided design. In particular, its facility for generating parametrically varied geometries in forms suitable for visual evaluation and/or engineering analysis enables an unprecedented potential for refinement and optimization of designs.

In the balance of this Introduction, we briefly amplify each of these characteristics.

Synthesis

The RG approach to modeling builds upward from simpler to more complex objects. *Point objects* are the foundation for any model. *Curves* are typically constructed from points in a variety of ways or, in some cases, from other curves. *Surfaces* are typically constructed from curves in a variety of ways or, in some cases, from other surfaces. Further geometry can be constructed on and from surfaces.

Capture of Relationships

The conventional CAD design process requires the creation of a sequence of geometric objects by execution of user commands. Almost every object added to a CAD model is created in some definite relationship to one or

more objects already in the model; for example, size, position, shape, and orientation. Most of these relationships exist only in the mind of the designer; but even in the cases where the intended relationship is explicitly communicated to and utilized by the CAD program, as soon as the new object is constructed the relationship information is discarded. Thus, a conventional CAD model consists of a collection of essentially independent, unrelated, self-contained objects.

If the designer had perfect foresight and could envision the completed model and all its properties from the outset of the design, this loss of relationship information might be of little consequence. However, it is well known that design is seldom a simple forward process. It is far more often an iterative process: Design is carried forward to some stage, then analyzed and evaluated; problems are identified; the designer then has to retreat to some earlier stage, change some elements, and work forward again. It is typical that many iterative cycles are required, depending on the skills of the designer, the difficulty of the design specifications, and whatever constraints and optimization objectives may be present. In each forward stage, the designer will have to repeat many previously performed operations (updating), in order to restore relationships disrupted by the revision of earlier design elements.

Revision of a previously existing design to meet new requirements is a common situation in which similar problems are encountered. A change that alters an early stage of the design requires at least one forward pass through all the subsequent design stages to restore disrupted relationships. The updating process can be very difficult, error-prone, and time-consuming—particularly if the relationships, and the sequence of design stages to achieve them, have been lost.

RG provides a comprehensive framework for capturing, storing, and utilizing relationships between objects. Each relationship expresses a constructive *dependency* of one object upon another or, conversely, the *support* of the second object by the first. Since some objects may depend on two or more other objects, and some objects may support two or more other objects, the overall structure of dependencies in a relational model is that of a *relation* or *directed graph*.

The principal advantage to capturing and maintaining interobject relationships is that the set of affected objects can be identified and automatically updated in appropriate sequence following a change in any object, preserving all captured relationships. A qualitative property or quantitative relationship that is so captured and maintained is referred to as a *durable property* of the RG model.

Rich and Extensible Entity Set

RG is also characterized by a rich set of geometric object classes (*entities*), each of which represents one or more distinct constructions of a single object from a set of other objects. An RG *object* instantiated from one of these classes knows how it was constructed, and from what other objects; this information is part of its stored data. RG provides a unified framework on which many useful methods of construction of points, curves, surfaces, and solid objects can coexist and complement one another.

In contrast to the restrictiveness and homogeneity of NURBS-based modeling, in which all geometry is fundamentally represented as ratios of B-Spline tensor products, RG is a heterogeneous, inclusive, pluralistic framework. The most fundamental representation of an RG object is simply a precise statement of its method of construction plus those data values required to complete the construction, including the identities of any supporting objects. Many different methods of construction can be supported within RG, including NURBS. This pluralism has a cost in complexity, in two important ways:

1. The entities all require different types and quantities of data.
2. Essentially, each entity requires a different evaluator procedure.

However, the additional complexity is found to be manageable through object-oriented program design, and it does confer profound advantages in modeling flexibility and accuracy.

Embedded Geometry

Many of the entities in RG provide constructions of objects that are explicitly embedded in other objects of higher or same dimensionality, for example:

Beads—points embedded in a curve
Magnets—points embedded in a surface
Snakes—curves embedded in a surface

The embedding of an object is a durable qualitative property, captured through a dependency relationship of the embedded object (*guest*) on the supporting object (*host*).

Accurate Joins between Surfaces

In conventional surface and solid modeling, intersections between surfaces play a prominent and often troublesome role. Typically, when two surfaces must share a common edge each of the surfaces is made arbitrarily larger

than necessary so that they deliberately intersect; the solution for the curve of intersection between them is then determined, and the unwanted portions of each surface are trimmed off along the intersection. This process has many difficulties and pitfalls, which account for much of the complexity, slow response, and frequent lack of robustness found in these programs.

RG provides several relatively efficient and robust ways to construct multiple surface objects having precise, durable joins. Often these procedures are the inverse of the conventional process; that is, first design the intersection, then build the adjoining surface or surfaces to conform to it. These methods do not solve all surface-to-surface joining problems, so projection- and intersection-based methods are also supported, but they are needed for at most a small percentage of the junctions in typical RG models.

Parametric Modeling

Typically, RG models are constructed so as to have many durable properties, combined with many parametric degrees of freedom. The durable properties result from captured relationships and embedding. They can be qualitative, topological properties or quantitative, metrical properties, or a combination. The parametric degrees of freedom can often be expressed as coordinate values, parameter values, or coordinate and parameter offsets of various point objects which are acting as supports for other geometry. This tends to permit visual, interactive variations by dragging point objects. An alternative is to support numeric variables and expressions as relational objects on which geometric elements can depend. In either case, the ability to work at a high level with parametrically variable models enables rapid refinement and optimization of designs, as well as adaptation and customization of existing designs to satisfy new requirements.

Compact Storage

The serialization of an RG model expresses the complete model structure, including interobject relationships, in an extremely compact form. In part, the compactness results from the use of references to objects to eliminate redundsncies in the representation. More profoundly, the compactness results from the expression of the model in terms of constructive relationships and prescriptions—a sort of genetic code for recreating the model from scratch—rather than as all the elaborated detail.

Related Developments

Sutherland's Sketchpad project (Sutherland [1963]) utilized stored references to point objects to establish durable connections between lines. Reference to

objects by name or address, and geometry languages, were also used in the APT machine-tool programming system and in Armit's Multipatch, Multiobject, ICON, and TAG projects (Armit [1982]). Some elements of RG may underlie Boeing's aerodynamics grid and paneling system (AGPS), a proprietary grid generation system (Snepp and Pomeroy [1987]). AGPS has at least some entities which reference other objects for their data, achieving at least one level of automatic updating. Snyder (1992) presents a procedural-language approach to CAD which supports rapid regeneration following changes in underlying objects. Each of these systems allows a script to be executed in order to regenerate a complex geometry with parametric variations.

Further, there are some examples of storing a multilevel relational structure of interobject dependencies, permitting selective updates of only the portions of the model which are actually affected by a change. Armit (1982) discusses this capability in a surface modeler, TAG. PADL-2 is a CSG solid-modeling framework with this feature (Brown [1982]). Gossard, et al. (1988) describe a dimension-driven polygonal solid modeler.

Constraint-based and variational modeling accomplish some of the updating capabilities of RG, but we find their flavor to be quite different from RG, and their relevance to free-form three-dimensional modeling to be apparently relatively limited.

MultiSurf and RGKernel

MultiSurf is a Windows-based relational surface modeler available from AeroHydro, Inc. The working demo version of MultiSurf, available free from the address or web site noted previously, will permit the interested reader to learn about relational geometry in far greater depth than the subject can be treated in this chapter. RGKernel, the C library providing the data storage, maintenance, serialization, and geometry evaluation functions of MultiSurf, is also available from AeroHydro, Inc. as a software component. It is noted that the novel features of RGS are the subject of patents and patent applications in several countries.

12.2 RELATIONAL MODEL STRUCTURE

Object versus *Entity*

Any geometric-modeling program must provide means of creating objects from several object types or classes, such as point, line, polyline, circular arc, Bézier patch, rectangular solid. CAD nomenclature tends to apply the words *object* and *entity* more or less interchangeably to both the abstract object class

and the objects instantiated from it. In RG we experience the need to differentiate these concepts, and to avoid confusion we recommend adherence to the following convention:

The abstract object class is called an *entity*.
The instantiated object is called an *object*.

A suggestive way to think of the entity is as a blank form for specifying an object of a particular class. An object is created by filling in all required fields on one of these forms. From the programmer's point of view, the entity is an abstract data type, plus one or more methods for utilizing the data; an object is the allocated and filled-in data structure.

In RG, one of the data fields in any entity is for an *object name*. Within a model, every object has a unique object name. Object names have utility in identifying objects on the screen, and for selecting objects for editing, interrogation, and export operations. However, the more critical and characteristic use of object names in RG is for specifying *interobject dependencies*. Most RG entities represent the construction of an object from one or more other objects (supports); the identity of the supports is expressed by use of their names.

Example: RelPoint

A *Relative Point* object is instantiated from the RelPoint entity by filling in the name of a basis-point object, and the x,y,z coordinate offsets from the basis point.

Example: BCurve

A *B-Spline Curve* object is a parametric B-Spline curve of specified degree made from an ordered set of control points (vertices). The set of supporting points is specified by an ordered list of object names.

Example: CLoftSurf

A *C-Lofted Surface* object is a parametric surface that interpolates a set of curve objects in sequence. The set of supporting curves is specified by an ordered list of object names.

Entity Classes and Subclasses

Most RG entities can be arranged into a class hierarchy, primarily on the basis of dimensionality and embedding, as in Figure 12.1. The parent class is

RGObject; the member data at this level include an object name, an entity code, and such common attributes as color and visibility codes. From RGObject, the hierarchy is in descending order of dimensionality:

RGSolids are parametric solids in three dimensions, $\mathbf{x} = \mathbf{x}(u,y,w)$.
RGSurfaces are parametric surfaces in three-dimensions, $\mathbf{x} = \mathbf{x}(u,v)$.
RGCurves are parametric curves in three-dimensions, $\mathbf{x} = \mathbf{x}(t)$.
  RGSnakes are parametric curves embedded in an RGSurface.
RGPoints are points in three-dimensions.
  RGBeads are points embedded in an RGCurve.
  RGMagnets are points embedded in an RGSurface.
  RGRings are points embedded in an RGSnake.

The inheritance diagrammed in Figure 12.1 primarily indicates the existence of methods permitting objects of one class to serve as degenerate instances of another class. For example, any RGPoint, or any object subclassed from RGPoint, can be evaluated as a degenerate RGCurve; it just returns the same x,y,z coordinates for any value of the parameter t. (The converse is not true; evaluating an RGCurve as an RGPoint would not make sense.) Similarly, any RGMagnet can be evaluated as a degenerate RGSnake. As another example, an RGSnake requires a method to be evaluated as an RGCurve, returning x,y,z coordinates for given parameter value t; but it does not make sense to try to evaluate an RGCurve as an RGSnake, since an RGCurve in general does not lie in any surface.

There are some problems with the hierarchy in Figure 12.1. Clearly, it is rife with multiple inheritance which, although supported to some degree in many object-oriented programming systems, is nonetheless a rich source of conceptual and implementational difficulties. Embedding of solids, surfaces, curves, and points in solids presents more complex possibilities not representable in the diagram.

There are other entity classes that do not fit into these dimensionality/embedding classifications. RGPlanes are infinite planes specified in several ways and used in a variety of constructions. RGFrames are local Cartesian coordinate frames. RGGraphs are nongeometric entities that provide a uniform way to specify univariate functions, used to enhance the modeling flexibility of many entities. RGRelabels, also nongeometric entities, provide a versatile way to reparameterize curves and surfaces.

Entities can also be classified as *direct* versus *iterative*. Objects instantiated from a direct entity can be evaluated exactly and uniquely in a finite number of operations (assuming exact arithmetic, and that the supports can be evalu-

Figure 12.1 Class hierarchy of relational entities. Numbers in parentheses represent the current population of each class.

ated exactly). An AbsMagnet is a simple example; it just requires the supporting surface to be evaluated at a single specified u,v position. Iterative entities require a search or iterative solution. For example, an IntMagnet is a point on a surface at its intersection with a curve; locating it requires solution of three simultaneous (usually nonlinear) equations in the three unknowns t,u,v. The iterative solution may require many surface and curve evaluations; may fail to exist; may fail to converge; may have multiple solutions; may require provision of starting values for the unknowns; and may have limited accuracy because of truncation errors. Direct entities are highly preferred for their simplicity, efficiency, accuracy, and robustness, but iterative entities provide sometimes necessary constructions not available through the direct entities.

A recently developed class of entity is designated as *procedural*. Essentially, procedural entities permit the construction of a curve, surface, or solid by constructing one typical point, curve, or surface using any sequence or combination of point, curve, or surface objects, then automating the repetition of that construction over a one-, two-, or three-dimensional parametric domain. As essentially *user-defined* entities, the procedural entities vastly broaden the flexibility of relational modeling.

Model Relation and Digraph

A *logical model* consists of a set O of objects $o_1, o_2, \ldots o_N$, and a set R (the *relation*) of constructive dependency relationships $r_1, r_2, \ldots r_M$, each relationship comprising an ordered pair (o_i, o_j) of elements of O in the order (support, dependent). A relation can be depicted by a *directed graph* (*digraph*) in which the nodes are objects, and each edge is a directed relationship between two objects. The *in-degree* of an object is the number of its supports; the *out-degree* is the number of its dependents.

We speak of the set of all objects to which there is a *downward* path through the digraph as the *descendants* of the object, and the set of all objects to which there is an *upward* path as the object's *ancestors*.

Figure 12.2 shows the digraphs for three simple relational models. Figure 12.2a consists of just two objects, an absolute point and a relative point. (An absolute point has no supports; its data is simply its x,y,z coordinates, plus some attributes common to all RGPoints.) Figure 12.2b consists of five objects: four absolute points and a BCurve made from them. Figure 12.2c also consists of five objects and is similar to Figure 12.2b, in fact may form the identical absolute configuration. However, in the case of Figure 12.2c, three of the points are RelPoints, relative to the AbsPoint *P0*. This provides a quite different relational structure, with this possibly useful property: If *P0* is moved, the curve *BC* will translate to a new position, while retaining its original shape.

Figure 12.2 Digraphs for three simple relational models: (*a*) Two objects, an absolute point and a relative point; (*b*) Five objects, four absolute points and a BCurve made from them; (*c*) five objects, one absolute point, three relative points, and a BCurve made from the four points.

As cyclic dependencies evidently lead to infinite recursion and self-contradictory or indeterminate models, they are currently forbidden; that is, the digraph is required to be acyclic.

12.3 MODEL EVALUATION

The logical model is insufficient for display, interrogation, and manufacturing purposes. Although it presumably contains a fundamentally exact and complete specification of a desired geometric configuration, the logical model contains little that you can get your teeth into. For example, only the AbsPoint objects are available in a form suitable for immediate display. The actual configuration of any objects having one or more supports is only implied. The evaluation of absolute geometric representations—explicit representations in terms of x,y,z coordinates—from the corresponding logical objects is the principal computational task of RG.

Object Evaluation

What constitutes the *evaluation* of an object depends upon the object's class and may be implementation-dependent. Each object contains a state variable *evaluated* indicating whether its evaluation is currently valid. An evaluated object contains data structures holding some evaluated form appropriate to its entity class. For example, the evaluation of any RGPoint object will include the x,y,z coordinates of the point. An object in the RGBead or RGRing class will additionally include the object's t parameter location on the supporting curve; an RGMagnet or RGRing will additionally include the object's u,v parameter location on the supporting surface. The evaluation of an RGCurve might be a uniform tabulation (sampling) of points from the curve; the evaluation of an RGSurface might be the tabulation of a topologically rectangular mesh of points from the surface. Derivative values could also be present in curve and surface evaluations.

The evaluation of any object can be immediately utilized for display purposes and for export of absolute object data in various formats.

Various error conditions can arise during the evaluation of an object; for the most part, these are entity-specific. For example, the evaluation of an RGPlane class object consists of a point in the plane and a unit normal vector. One RGPlane class entity is a Plane3—the plane determined by three supporting points. If these three points happen to be collinear, they fail to determine a plane. Another possibility is that evaluation of one or more of the supports fails; for example, one of the supporting points may be an IntMagnet that failed to converge. Such failures must be detected and handled—the Plane3 object must not be marked as evaluated until the problem is corrected—or nonsense will ensue (assuming some other object depends on the plane).

Program Structure

Although relational geometry is presented as an object-oriented framework—suggesting the particular appropriateness of implementation in an object-oriented language—in fact, to date RGKernel is implemented entirely in ANSI C. The principal interface between an application requiring absolute geometric dsta and the RGKernel is a set of *class evaluators,* one for each entity class. One argument for any class evaluator is the object to evaluate. The other arguments depend on the class:

RGPoint: out- X,Y,Z
RGBead: out- identity of host curve; t parameter

RGMagnet: out- identity of host surface; u,v parameters
RGRing: out- identities of host snake and surface; t,u,v parameters
RGLine: out- X,Y,Z of one point; unit tangent vector
RGCurve: in- vector of t parameter values
out- tabulation of sampled points: X,Y,Z
RGSnake: in- vector of t parameter values
out- identity of host surface; tabulation of u,v parameters
RGSurface: in- vector of u parameter values; vector of v parameter values
out- tabulation of sampled points: X,Y,Z
RGSolid: in- vector of u values; vector of v values; vector of w values
out- tabulation of sampled points: X,Y,Z
RGPlane: out- X,Y,Z of one point; unit normal
RGFrame: out- transformation matrix

The return value of a primary math routine is an error code if an error occurs; else 0.

The essential structure of a class evaluator is simply (pseudo-C):

```
int RGClass ( object, other arguments)
[
        int err = 0;
        if( object->entity is not in this class) return 1;
        if( object->evaluated && LookupEnabled)
                fill output arguments from lookup table;
        else
                switch( object->entity)
                [
                case ENTITY1: err = Entity1( object, other arguments);
                break;
                case ENTITY2: err = Entity2( object, other arguments);
                break;
                .....
                case ENTITYN: err = EntityN( object, other arguments);
                break;
                ]
        return err;
]
```

In the above pseudocode, Entity1( ), Entity2( ), and so on are evaluator procedures for the individual entities in the class. If, as in almost all cases, the entity has supports, the entity evaluator will have an equivalent number of class evaluator calls to evaluate them. For example, the evaluator for a BCurve will first iterate through its point supports, calling the RGPoint class evaluator for each of them and storing the resulting control-point coordinates. Then it can iterate through its input list of *t* parameter values, evaluating each output point as the inner product of the control points with B-Spline basis functions.

Thus, the addition of a new entity to an existing class requires only:

1. Creation of the evaluator procedure for this entity
2. A new case in the class evaluator
3. Addition of user interface code for creation and editing

Only linear effort is required to indefinitely extend the supported entity set.

Recursive Model Evaluation

Provided that the evaluator functions are all programmed recursively (i.e., reentrant calls are permitted), the program structure previously outlined (with **LookupEnabled = FALSE**) can recursively evaluate any valid model (or identify the errors in a model that is incompletely valid). Accuracy of recursive evaluation is limited only by roundoff, and by controllable truncation error in the case of iterative entities. However, it is apparent that when there are deep levels of dependency many function calls are required, possibly repeating some evaluations many times, so the model evaluation process may be unacceptably slow.

Table Lookup

An alternative to recursive evaluation, included in the previous structure when **LookupEnabled** is **TRUE**, is to utilize the evaluation of an object as a lookup table. Table lookup short-circuits the recursion and prevents duplicate evaluations, enormously accelerating the evaluation process. In the case of curves, surfaces, and solids, some form of interpolation is needed, as the tabulated sampling will generally not include the particular parameter values that will be required in later evaluations. Truncation errors will generally be present, and will accumulate to a degree that is difficult to predict and monitor. Nevertheless, in our experience interpolation-based evaluation can provide ample accuracy for visual displays during interactive modeling phases, and in most cases for the output of actual manufacturing data.

Updating

Part of the data stored in an object is an ordered list of its supports, as required for the model evaluation process previously outlined. This is the primary representation of the model relation. For efficient updating, it is also expedient to store and maintain a list of the object's dependents. When an object is changed in an editing operation, it can notify its dependents, they can notify their dependents, and so on, in a recursive process that finally invalidates all descendants. (Invalidation is resetting the *evaluated* flag.)

During the evaluation of an object, as noted earlier, each of its supports must be evaluated. If a support is marked as invalid, it will be evaluated before a value is returned. Thus, evaluating any invalidated object will cause any of its ancestors that are invalid to reevaluate themselves, recursively, in the correct sequence to update this particular object. No object needs to be evaluated more than once in updating the entire model.

12.4 SERIALIZATION

File storage of a relational model requires a serialization of the object data for all of the model's objects. Besides the object data, the file must convey any model-level information. This can include a language version, model version, symmetry flags, identification of units, model-level comments, and so on.

The body of the file is a sequence of objects. Each entity has a method for writing its data to a string. An object pointer, that is, the identity of a support, is output as the name of the referenced object. It is not necessary to output the object's list of dependents, as this information can easily be reconstructed when the file is read back in.

The format for an object string is essentially as follows:

Entity-name object-name color visibility divisions [other object attributes] / (entity-specific data);

Entity-name: A keyword identifying the entity; for example, AbsPoint, RelPoint, BCurve.

object-name: A name for the object.

color: An integer color code for displaying the object.

visibility: An integer code controlling aspects of the object's display appearance.

divisions: Specifies the number of divisions and subdivisions for tabulation and display.

other object attributes: Include optional layer, unit weight, object remark.

/: Terminates the optional attributes.

entity-specific data: The sequence of integer, float, and supporting object names required to instantiate the object; for example:

AbsPoint object-name ..... / X Y Z;

RelPoint object-name ..... / point dX dY dZ;

BCurve object-name ..... / relabel type { point1 point2 ..... pointN };

A variable-length list of supports, such as the control points of a BCurve, is enclosed in braces.

The serialization of a relational model can be viewed as a geometry-representation language. Model files can be created by typing in this language, or can be automatically generated and modified by other programs, so long as correct syntax is observed. Entity names are commands to create, that is, calls to predefined procedures. The object's data values are like the parameters passed to the procedure. Object names are analogous to variables. MultiSurf functions as an interactive interpreter/compiler for the model langnage.

The model language does not currently support such typical programming-language constructs as subroutines, branching, and iterators, but these could be fruitful additions for the future.

Early versions of MultiSurf required top-down order in the model file (i.e., an object had to be defined before it could be used as a support for another object) as well as in the internal data storage. This condition was difficult to maintain when editing the model file manually. It was later found to require only a modest amount of bookkeeping during file read-in to dispense with top-down order.

12.5 RELATIONAL ENTITIES

In this section we specify for concreteness' sake about 80 relational entities representing all object classes. (This is about 75 percent of the current MultiSurf/ RGKernel entity population.)

The entities are presented in the form of a template for their model file descriptions. Some shorthand notation used in these templates requires explanation:

point, curve, surface, and so on: Entity class names signify the name of an object of the specified class.

{ .... }: Braces indicate an arbitrary-length list of the indicated objects; counted 1 to N.

.....: Ellipsis indicates position of some attributes that are class-specific—color, visibility, divisions for tabulation and display—plus some optional attributes such as layer and unit weight.

mirror: A plane, line, or point object used for mirror imaging

mirror/surface: For projection, a two-parameter family of projection lines; radials through a point, normals to a plane, line, or surface. For intersection, the dual one-parameter family of implicit surfaces; spheres around a point, cylinders around a line, planes parallel to a plane, or surfaces parallel to a surface.

bead/curve: Either a bead or a curve. This specifies a curve, and optionally a point on it, which is usually used as the starting value for a fixed-point iteration. Similarly **ring/snake** and **magnet/surface**.

knotlist: A knot vector object used for NURBS entities.

Three-Dimensional Points

AbsPoint name .... / X Y Z;
RelPoint name .... / point dX dY dZ;
PolarRelPt name .... / point lat lon radius;

In the Polar Relative Point, the distance and direction from **point** are specified in terms of spherical polar coordinates.

ProjPoint name .... / point mirror;

The Projected Point is located at the projection of **point** upon the **mirror**.

MirrPoint name .... / point mirror;

The Mirror Point is located at the mirror image of **point** with respect to the **mirror**, that is, two times the displacement of the ProjPoint.

RotatPoint name .... / point line angle;

The Rotated Point is located by rotating from the position of **point**, about the axis **line**, through an angular distance of **angle**.

TanPoint name .... / bead offset;

The Tangent Point is located along the tangent to the host curve of **bead**, starting at the location of **bead**, at signed distance **offset**.

OffsetPt name .... / magnet offset;

The Offset Point is located along the normal to the host surface of **magnet**, starting at the location of **magnet**, at signed distance **offset**.

FrameAbsPt name .... / frame x y z;

The Frame Absolute Point is located at x, y, z specified in **frame** coordinates.

FrameRelPt name .... / frame point dx dy dz;

The Frame Relative Point is located relative to **point** at the offsets dx, dy, dz specified in **frame** coordinates.

BlendPoint name .... / { point1 wt1 point2 wt2 .... pointN wtN };

The Blended Point is the weighted sum of the component **points**: $\mathbf{x} = \Sigma\, w_i \mathbf{x}_i$

Beads and Rings

A *bead* class entity constructs a single point embedded in a curve. A *ring* class entity constructs a single point embedded in a snake. Since snakes are a subclass of curves, rings are a subclass of beads. The difference is that a ring is logically also embedded in the snake's host surface, therefore can serve also as a magnet on that surface.

AbsBead name ..... / curve t ;

The Absolute Bead is the point on **curve** at parameter value t.

RelBead name ..... / bead dt;

The Relative Bead is a point on the host curve of **bead**, at the signed parameter offset dt.

ArcRelBead name ..... / bead/curve ds;

The Arclength Relative Bead is a point on the host curve of **bead**, at the signed arclength offset *ds*. When a **curve** is directly specified, rather than a **bead**, the offset is from $t = 0$.

IntBead name ..... / bead/curve mirror/surface point;

The Intersection Bead is a point on the curve designated by **bead/curve**, where it is intersected by the implicit cutting surface parallel to **mirror/surface** which passes through **point**. If **bead/curve** is a bead, its parameter location t is used as the starting value for the iterative search; if it is a curve, the search starts at $t = .5$.

The following three ring entities are parallel to the corresponding beads.

AbsRing name ..... / snake t ;
RelRing name ..... / ring dt ;
IntRing name ..... / ring/snake mirror/surface point;

IntRing2 name ..... / ring/snake1 ring/snake2 ;

The Intersection Ring, second kind, is the intersection of two snakes on the same surface (or of a snake with itself). Its host is the snake designated by **ring/snake1**. If either ring/snake is a ring, its t parameter value is used as the starting position for the search; otherwise the search starts at $t = .5$.

Magnets

Each magnet class entity constructs a point embedded in a surface.

AbsMagnet name ..... / surface u v ;

The Absolute Magnet is the point on **surface** at parameter values u, v.

RelMagnet name ..... / magnet du dv ;

The Relative Magnet is the point on the host surface of **magnet**, at parameter offsets du, dv.

IntMagnet name ..... / magnet/surface bead/curve ;

The Intersection Magnet is a point on the surface designated by **magnet/surface** where it is intersected by the curve designated by **bead/curve**. If **mag-**

net/surface is a magnet, its u, v position is used for starting values for the search; otherwise the search starts at $u = .5$, $v = .5$.

ProjMagnet name ..... / point magnet/surface mirror/surface;

The Projected Magnet is a point on the surface designated by **magnet/surface**, formed by the projection of **point** along a radiant of **mirror/surface**. If **magnet/surface** is a magnet, its u, v position is used for starting values for the search; otherwise the search starts at $u = .5$, $v = .5$.

Curves

An RG curve class entity is a parametric curve $\mathbf{x} = \mathbf{x}(t)$ with a nominal parametric range of [0, 1]. However, curves do extend beyond the nominal range, and can be evaluated at any parameter value, unless the evaluation results in a division by zero (possible with NURBCurve). The first support of any curve is a Relabel object, which performs an optional reparameterization. Specifying the default relabel, denoted by *, accepts the curve's *default* or *natural* parameterization.

Line name ..... / relabel point1 point2;

The Line is the straight line from **point1** to **point2**. The default parameterization is uniform. Besides qualifying as a curve, a line can serve as a mirror or as a rotation axis in some entity constructions.

Arc name ..... / relabel type point1 point2 point3;

The Arc is a circular arc constructed in one of six ways (**type** 1 to 6) from three **points**. The default parameterization is uniform with respect to arc length.

Conic name ..... / relabel type point1 point2 point3 s0 s1;

The Conic is a conic section or catenary, depending on **type**. **point1** is the center; **point2** and **point3** establish semimajor and semiminor axes. **s8** and **s1** are starting and ending conic parameter values corresponding to $t = 0$ and $t = 1$.

Helix name ..... / relabel point line graph angle ;

The Helix is a generalized helix permitting variable radius and pitch. If **line** is uniformly parameterized and **graph** is constant, the resulting curve is a standard helix.

BCurve name ..... / relabel type { point1 point2 ..... pointN };

The B-Spline Curve is the Bézier curve using the B-Spline basis functions with uniform knots, order k = **type** + 1, and N vertices: $\mathbf{x}(t) = \Sigma\, \mathbf{x}_i B_i(t)$.

CCurve name ..... / relabel type { point1 point2 ..... pointN };

The C-Spline Curve is an interpolating spline of **type** 2 (quadratic) or 3 (cubic) with chord-length parameterization and not-a-knot end conditions.

FoilCurve name ..... / relabel type { point1 point2 ..... pointN }

The Foil Curve implements several families of standard NACA airfoil sections. **type** selects the airfoil family. The number N of control **points** can be three, four or five. Three points create half of a symmetric profile; four a full symmetric foil; five a full cambered foil.

NURBCurve name ..... / relabel type knotlist { point1 wt1 point2 wt2 .... pointN wtN };

The NURBS Curve is the rational B-Spline with order k = **type** + 1, knots supplied by a KnotList object, and N control **points** $\mathbf{x}_i$ with weights w_i: $\mathbf{x}(t) = \Sigma\, w_i \mathbf{x}_i B_i(t) \,/\, \Sigma\, w_i\, B_i(t)$.

RelCurve name ..... / relabel curve point0 point1 graph;

The Relative Curve is a copy of its basis **curve** $\mathbf{x}_b(t)$, stretched and rotated to fit between new endpoints $\mathbf{x}_0, \mathbf{x}_1$: $\mathbf{x}(t) = \mathbf{x}_b(t) + (1 - f(t))\,[\mathbf{x}_0 - \mathbf{x}_b(0)] + f(t)\,[\mathbf{x}_1 - \mathbf{x}_b(1)]$. The graph supplies the blending function $f(t)$; in this context, the default graph * signifies a linear blend, $f(t) = t$.

SubCurve name ..... / relabel bead1 bead2;

The SubCurve is the portion of a basis curve $\mathbf{x}_b(s)$ between two **beads** at positions s_0 and s_1, reparameterized to the interval [0,1]: $\mathbf{x}(t) = \mathbf{x}_b[(1 - t)\, s_0 + t\, s_1]$.

PolyCurve name ..... / relabel { curve1 t1 curve2 t2 ..... curveN 1.) };

The PolyCurve is the concatenation of two or more **curves**, reparameterized to the interval [0,1]. The real value t_i associated with each **curve** specifies where that **curve** ends in the [0,1] parameter space of the PolyCurve.

ProcCurve name ..... / relabel point { bead/ring/graph1 .... bead/ring/graphN };

point is any point object, which generally has the **bead** and **ring** supports as ancestors. The Procedural Curve is the locus of **point** as the supporting **beads**, **rings**, and/or **graphs** are varied in unison over the t parameter interval [0,1]. Any **graphs** included as supports are evaluated at each parameter value t, and substituted in order for any real parameters in **point**.

Snakes

An RGSnake class object is a parametric curve with a nominal parametric range of [0, 1], embedded in a surface object. It needs to be emphasized that a snake by definition lies precisely on the host surface, and that three-dimensional absolute coordinates of points on the snake are obtained by a composition operation: The snake is evaluated to a location $\mathbf{w} = (u,v)$ in the surface's parameter space; then the surface is evaluated at this location through the surface equations.

LineSnake name .... / relabel magnet1 magnet2;

A straight line is constructed in the (u,v) parameter space, from **magnet1** to **magnet2:** $\mathbf{w}(t) = (1 - t)\,\mathbf{w}_1 + t\,\mathbf{w}_2$.

The LineSnake and the following snakes are highly parallel to the corresponding curve entities, requiring no further explanation:

ArcSnake name .... / relabel type magnet1 magnet2 magnet3 ;
BSnake name .... / relabel type { magnet1 magnet2 .... magnetN } ;
CSnake name .... / relabel type { magnet1 magnet2 .... magnetN } ;
FoilSnake name .... / relabel type { magnet1 magnet2 .... magnetN } ;
NURBSNake name .... / relabel type knotlist { magnet1 wt1 magnet2 wt2 .... magnetN wtN} ;
RelSnake name .... / relabel snake magnet1 magnet2 graph ;
SubSnake name .... / relabel ring1 ring2 ;
PolySnake name .... / relabel { snake1 t1 snake2 t2 .... snakeN tN };
ProcSnake name .... / relabel magnet { bead/ring/graph1 ..... bead/ring/graphN }

The following snakes have no direct counterpart among RGCurves:

ProjSnake name .... / relabel curve magnet/surface mirror/surface;

The Projected Snake is the projection of **curve** onto the surface identified by **magnet/surface**, using **mirror/surface** for the projection.

IntSnake name .... / relabel magnet mirror/surface point;

The Intersection Snake is an intersection of the host surface of **magnet** with the implicit surface parallel to **mirror/surface** and passing through **point**. **magnet** serves three purposes: to identify the host surface for the IntSnake, to identify which of multiple intersection curves is to be used for the IntSnake (the one whose end is closest to **magnet**), and to identify which end of the selected intersection curve is to be the $t = 0$ end of the IntSnake.

UVSnake name ..... / relabel type magnet;

The UVSnake is a parametric line u = constant or v = constant, according to **type**.

EdgeSnake name ..... / relabel type surface;

The EdgeSnake is one of the four edges of **surface**, according to **type**.

GeoSnake name ..... / relabel magnet1 magnet2;

The Geodesic Snake is a geodesic arc from **magnet1** to **magnet2**.

Surfaces

An RGSurface class object is a parametric surface patch $\mathbf{x} = \mathbf{x}(u,v)$ with a nominal parametric range of $[0,1] \times [0,1]$. Like curves, surfaces extend beyond the nominal range and can be evaluated at any parameter value, unless the evaluation results in a division by zero. The first support of some surfaces is a Relabel object, which performs an optional reparameterization in the v-direction. Specifying the default relabel, denoted by *, accepts the surface's default or natural parameterization.

Several surface entities belong to a subclass of *lofted surfaces*, supported by an ordered set of curves called *master curves*. Evaluation of a lofted surface at a position (u,v) is a three-stage process:

1. Each master curve is evaluated at parameter value u.
2. The resulting points $\mathbf{x}_i$, $i = 1, \ldots, N$ are used to construct a lofting curve of appropriate type.
3. The lofting curve is evaluated at parameter value v.

TranSurf name ..... / curve1 curve2;

The Translation Surface is generated by a translated copy of **curve2**, swept along **curve1:** $\mathbf{x}(u,v) = \mathbf{x}_1(u) + \mathbf{x}_2(v) - \mathbf{x}_2(0)$.

RevSurf name ..... / relabel curve line angle1 angle2;

The Revolution Surface is generated by revolving **curve** about **line**, from **angle1** to **angle2**.

RuledSurf name ..... / relabel curve1 curve2;

The Ruled Surface is generated by connecting corresponding points on **curve1** and **curve2** with straight lines: $\mathbf{x}(u,v) = (1 - v)\,\mathbf{x}_1(u) + v\,\mathbf{x}_2(u)$.

DevSurf name ..... / bead/curve1 bead/curve2;

The Developable Surface is also a ruled surface between **curve1** and **curve2**, but the rulings are skewed: $\mathbf{x}(u,v) = (1 - v)\,\mathbf{x}_1(u + \delta) + v\,\mathbf{x}_2(u - \delta)$, where $\delta(u)$ solves the tangency condition (scalar triple product).

$$\left| [\mathbf{x}_1(u + \delta) - \mathbf{x}_2(u - \delta)]\ \mathbf{x}_1'(u + \delta)\ \mathbf{x}_2'(u - \delta) \right| = 0$$

which ensures developability. This solution for δ requires iteration. If the supports are beads (at $t1$ and $t2$, say), they supply a starting value for δ at $u = (t1 + t2)/2$—a suggested first ruling—which helps to select between possible multiple solutions.

BlendSurf name ..... / graph1 graph2 curve1 curve2 curve3 curve4;

The BlendSurf is a Coons patch formed from the four boundary **curves**, which nominally join end-to-end in the given sequence: $\mathbf{x}(u,v) = \mathbf{U}(u,v) + \mathbf{V}(u,v) - \mathbf{W}(u,v)$, where:

$$\mathbf{U} = (1 - a)\,\mathbf{x}_1(u) + a\,\mathbf{x}_3(1 - u)$$

$$\mathbf{V} = (1 - b)\,\mathbf{x}_4(1 - v) + b\,\mathbf{x}_2(v)$$

$$\mathbf{W} = (1 - a)(1 - b)\,\frac{\mathbf{x}_1(0) + \mathbf{x}_4(1)}{2} + (1 - a)\,b\,\frac{\mathbf{x}_2(0) + \mathbf{x}_1(1)}{2}$$

$$+ a\,b\,\frac{\mathbf{x}_3(0) + \mathbf{x}_2(1)}{2} + a\,(1 - b)\,\frac{\mathbf{x}_1(0) + \mathbf{x}_4(1)}{2}$$

The **graphs** supply the optional blending functions $a(v)$, $b(u)$; the default graphs are the identities $a(v) = v$, $b(u) = u$. If the boundary curves do meet at the corners, the BlendSurf interpolates all four of them.

NURBSurf name ..... / u-type v-type knotlist1 knotlist2 ncu
{ point11 wt11 ..... pointNM wtNM };

The NURBSurf is the nonuniform rational B-Spline surface made from a rectangular array of control **points** $\mathbf{x}_{ij}$ with weights w_{ij}: $\mathbf{x}(u,v) = \Sigma\,\Sigma\, w_{ij}\, \mathbf{x}_{ij}\, B_i(u)B_j(v) / \Sigma\,\Sigma\, w_{ij}\, B_i(u)\, B_j(v)$.

ALoftSurf name ..... / relabel type curve1 curve2 curve3 ;

The Arc-lofted Surface is lofted with circular arcs (see Arc) of the specified **type**, 1 to 6.

BLoftSurf name ..... / relabel type { curve1 curve2 ..... curveN } ;

The B-Spline lofted surface is lofted with B-Spline curves: $\mathbf{x}(u,v) = \Sigma\, \mathbf{x}_i(u)\, B_i(v)$. It interpolates $\mathbf{x}_1$ and $\mathbf{x}_N$ and is "attracted" to its other master **curves** in the same way a B-Spline curve is attracted to its control points.

CLoftSurf name ..... / relabel type { curve1 curve2 ..... curveN } ;

The C-lofted surface is lofted with C-Spline curves (see CCurve) that interpolate all the master **curves** in sequence.

FLoftSurf name ..... / relabel type { curve1 curve2 ..... curveN } ;

The Foil-lofted surface is lofted with standard airfoil sections (see FoilCurve). There can be three, four, or five master **curves**. The first and fifth are the trailing edge, the third is the leading edge, and the second and fourth control thickness and camber.

SweepSurf name ..... / bead curve1 curve2 graph1 graph2 graph3;

The SweepSurf is generated by sweeping **curve1** (the *shape*) along the host curve of **bead** (the *path*), with optional scaling supplied by the graphs.

OffsetSurf name ..... / surface { offset1 { offset2 offset3 offset4} } ;

The OffsetSurf is generated from the base **surface** by offsetting in the normal direction by a signed variable amount. If only **offset1** is given, the result is a parallel surface. If **offsets** are given for the four corners, the normal offset is bilinearly blended.

RelSurf name ..... / surface { point1 [point2 point3 point4} } ;

The Relative surface is a copy of **surface**, rotated and stretched to span four new corner **points**. If only **point1** is given, the result is a parallel translation bringing the (0,0) corner to **point1**. If all four corners are given, the displacement is a bilinear blend of the corner displacements:

$$\mathbf{x}(u,v) = \mathbf{x}_b(u,v) + (1-u)(1-v)[\mathbf{x}_1 - \mathbf{x}_b(0,0)] + u(1-v)[\mathbf{x}_2 - \mathbf{x}_b(1,0)] + uv[\mathbf{x}_3 - \mathbf{x}_b(1,1)] + (1-u)v[\mathbf{x}_4 - \mathbf{x}_b(0,1)]$$

PolySurf name ..... / { surface1 u1 surface2 u2 ..... surfaceN uN } ;

The PolySurf is the concatenation of two or more **surfaces**, reparameterized in u to the interval [0,1]. The real value u_i associated with each **surface** specifies where that **surface** ends in the [0,1] u-parameter space of the PolySurf.

SubSurf name ..... / { snake1 snake2 { snake3 snake4 } } ;

The SubSurf is a portion of the host surface of the **snakes**, reparametrized to $[0,1] \times [0,1]$. Denote the parameters of the supporting surface by $\mathbf{w} = \{p,q\}$. If only two **snakes** are used, the SubSurf is a *ruling* between them in p,q space: $\mathbf{w}(u,v) = (1-v)\,\mathbf{w}_1(u) + v\,\mathbf{w}_2(u)$; if all four **snakes** are given, the SubSurf is a bilinear Coons blending between them in p,q space, analogous to the BlendSurf.

ProcSurf1 name ..... / point { magnet1 magnet2 ..... magnetN } ;

point is any kind of point object, which generally has the **magnets** as ancestors. The first kind of Procedural Surface is the locus of **point** as the **magnets** are varied in unison over their (u,v) spaces.

ProcSurf2 name ..... / curve { bead/ring/graph1 .... bead/ring/graphN } ;

curve is any curve object, which generally has the **bead** and **ring** supports as ancestors. The second kind of Procedural Surface is the locus of **curve** as the supporting **beads**, **rings**, and/or **graphs** are varied in unison over the t parameter interval [0,1]. Any **graphs** included as supports are evaluated at each parameter value t, and substituted in order for any real parameters in **curve**.

Solids

An RGSolid class object is a parametric solid hyperpatch $\mathbf{x} = \mathbf{x}(u,v,w)$ with a nominal parametric range of $[0,1] \times [0,1] \times [0,1]$. Solids are not currently implemented in the MultiSurf product, but are in a development version of

RGKernel. Currently supported construction methods are highly parallel to corresponding surface entities.

RuledSolid name ..... / relabel surface1 surface2;

The Ruled Solid is generated by connecting corresponding points on **surface1** and **surface2** with straight lines: $\mathbf{x}(u,v,w) = (1 - w)\,\mathbf{x}_1(u,v) + w\,\mathbf{x}_2(u,v)$.

RevSolid name ..... / relabel surface line angle1 angle2;

The Revolution Solid is generated by revolving **surface** about **line**, from **angle1** to **angle2**.

BLoftSolid name ..... / relabel type {surface1 surface2 ..... surfaceN };

The B-Spline lofted solid is lofted with B-Spline curves: $\mathbf{x}(u,v,w) = \Sigma\, \mathbf{x}_i(u,v)\, B_i(w)$.

BlendSolid name ..... / graph1 graph2 graph3 surface1 .... surface6;

The BlendSolid is a trilinear blend of its six supporting **surfaces**, which nominally surround a topologically hexahedral volume.

Contours

Each Contour object is a set of contours cut through one or more surfaces by a family of parallel implicit surfaces: *planes, cylinders,* or *spheres*.

Contours name ..... / first last q0 qint mirror { surfaces };

The cutting surfaces are implicit surfaces parallel to **mirror**, at signed distances $q = \mathbf{q8} + i\,\mathbf{qint}$, $i = \mathbf{first}, \ldots \mathbf{last}$.

XContours name ..... / first last X0 Xint { surfaces } ;
YContours name ..... / first last Y0 Yint { surfaces } ;
ZContours name ..... / first last Z0 Zint { surfaces } ;

The cutting surfaces are planes parallel to the coordinate planes; for example, the planes $X = \mathbf{X8} + i\,\mathbf{Xint}$, $i = \mathbf{first}, \ldots \mathbf{last}$.

CvContours name ..... / curve first last t0 tint { surfaces };

The cutting surfaces are planes normal to **curve**, at parameter values t = **tB** + i **tint**, i = **first**, ...**last**.

Graphs

A Graph object is a univariate function $f(t)$ over the parameter range [0,1]. Graphs are not visible objects, but are used to enrich the flexibility of other entities. Wherever a graph is required, the default graph * can always be used to accept some appropriate default behavior, for example, the linear graph $f(t) = t$, or a constant $f(t) = 1$ or $f(t) = 0$. The meaning of the default graph * is specific to the entity in which it is used.

BGraph name / type { f1 f2 ..... fN } ;

The B-Spline Graph is a univariate function specified as a linear combination of the B-Spline basis functions with uniform knots, order k = **type** + 1, and N vertices: $f(t) = \Sigma f_i B_i(t)$.

Relabel name / type { t1 t2 ..... tN } ;

The Relabel is a special BGraph with $t_1 = 0, t_N = 1$, and nondecreasing values, used for reparameterizing curves and surfaces. The parameter t or v is put through the relabel function before being used in the curve or surface definition.

PolyGraph name / { graph1 t1 graph2 t2 .... graphN 1.0 } ;

The PolyGraph is the concatenation of two or more **graphs**, reparameterized to the interval [0,1]. The real value t_i associated with each **graph** specifies where that **graph** ends in the [0,1] parameter space of the PolyGraph.

Planes

Each Plane class object defines an infinite implicit plane. Planes are not parameterized, so they cannot support magnets and snakes.

Plane2 name ..... / point1 point2 ;

The plane passes through **point1** and is normal to the line from **point1** to **point2**.

Plane3 name ..... / point1 point2 point3 ;

The Plane3 object is the plane determined by the three **points**.

XPlane name ..... / X ;
YPlane name ..... / Y ;
ZPlane name ..... / Z ;

These are planes parallel to the coordinate planes, at the specified X, Y, or Z positions.

Frames

Each Frame class object is a local right-handed orthogonal x,y,z coordinate system, used for location of FramePoints. The default frame * is the global coordinate system.

Frame3 name ..... / point1 point2 point3 ;

This frame is located at **point1**, with its x axis in the direction of **point2**, and its y axis in the plane of the three **points**.

EulerFrame name ..... / point frame angle1 angle2 angle3 ;

The Euler Frame is constructed by translating a copy of **frame** to **point**, followed by rotations through three Euler **angles**.

12.6 DETAILED EXAMPLE

Table 12.1 is the model file for an example model utilizing a variety of point, curve, and snake objects and six connected surfaces of various types, as defined and outlined previously. All of the surface connections are exact and durable. Figure 12.3 is a wireframe representation of the resulting absolute model. The example comprises hull, deck, and cabin surfaces for a 30-foot sailing-yacht design.

The example model has six surface objects: **hull** and **deck** are C-lofted surfaces; **cabin_fwd, cabin_side** and **cabin_aft** are ruled surfaces; and **cabin_top** is a blended surface. The surfaces all have visibility 1, which causes only the parameter lines u = constant to be displayed. Eleven transverse sections (XContours **stations**) through the model are also displayed for purposes of visualizing the shapes.

hull is a C-lofted surface with three B-Spline master curves **MCA**, **MCB**, **MCC**, each having four absolute points as supports. **deck** also has three mas-

Table 12.1 Example of relational model file

```
MultiSurf 1.08
Units: none
Extents: 0.000 0.000 -1.451 31.700 6.023 5.173
View: 10.00 50.00 0
Places: 3
BeginModel;
// 3x4 cloft hull, deck, cabin for GM example
AbsPoint A1 14 1 / 0.000 0.000 3.600 ;
AbsPoint A2 14 1 / 1.000 0.000 1.410 ;
AbsPoint A3 14 1 / 2.500 0.000 -0.840 ;
AbsPoint A4 14 1 / 3.000 0.000 -0.900 ;
BCurve MCA 11 1 8x4 / * 2 { A1 A2 A3 A4 } ;
AbsPoint B1 14 1 / 15.000 5.840 2.640 ;
AbsPoint B2 14 1 / 15.000 6.000 0.540 ;
AbsPoint B3 14 1 / 15.000 3.900 -1.200 ;
AbsPoint B4 14 1 / 15.000 0.000 -1.440 ;
BCurve MCB 11 1 8x4 / * 2 { B1 B2 B3 B4 } ;
AbsPoint C1 14 1 / 30.000 3.500 2.760 ;
AbsPoint C2 14 1 / 30.900 3.500 1.410 ;
AbsPoint C3 14 1 / 31.700 2.500 0.220 ;
AbsPoint C4 14 1 / 31.700 0.000 0.220 ;
BCurve MCC 11 1 8x4 / * 2 { C1 C2 C3 C4 } ;
CLoftSurf hull 10 1 8x2 10x3 0 / * 3 { MCA MCB
MCC } ;
AbsPoint transom0 14 1 / 29.800 0.000 3.000 ;
AbsPoint transom1 14 1 / 29.800 1.750 3.000 ;
BCurve transom_top 10 1 10x1 / * 2 { C1 transom1
transom0 } ;
AbsPoint deck_clr 14 1 / 15.000 0.000 3.450 ;
AbsPoint deck_mid 14 1 / 15.000 2.700 3.450 ;
BCurve deck_beam 10 1 10x1 / * 2 { B1 deck_mid
deck_clr } ;
CLoftSurf deck 7 1 4x3 10x3 0 / * 3 { A1 deck_beam
transom_top } ;
EdgeSnake n0 13 1 8x4 / * 2 deck ;
AbsRing r0 11 1 / n0 0.25500 ;
RelRing r1 11 1 / r0 0.42600 ;
RelMagnet dm2 12 1 / r0 -0.37000 0.0 ;
RelMagnet dm3 12 1 / r0 -0.65000 0.03000 ;
SubSnake n1 14 1 8x4 / * r0 r1 ;
AbsRing r2 11 1 / n1 0.50000 ;
```

Table 12.1 (*Continued*)

```
RelMagnet dm4 12 1 / r2 -0.80000 0.0 ;
RelMagnet dm5 12 1 / r1 -0.80000 0.0 ;
BSnake fp_fwd 11 1 10×1 / * 2 { r0 dm2 dm3 } ;
BSnake fp_side 11 1 20×1 / * 2 { dm3 dm4 dm5 } ;
LineSnake fp_aft 11 1 10×1 / * dm5 r1 ;
RelPoint rp1 11 1 / r0 2.000 0.000 1.300 ;
RelPoint rp3 11 1 / dm3 2.000 0.000 1.100 ;
RelPoint rp5 11 1 / dm5 -0.200 -0.500 1.400 ;
RelPoint rp6 11 1 / r1 -0.300 0.000 1.800 ;
RelPoint rp7 11 1 / r2 0.000 0.000 1.650 ;
RelCurve top_fwd 11 1 10×1 / * fp_fwd rp1 rp3 * ;
RelCurve top_side 11 1 20×1 / * fp_side rp3 rp5 * ;
RelCurve top_aft 11 1 10×1 / * fp_aft rp5 rp6 * ;
RuledSurf cabin_fwd 11 1 10×1 1×1 0 / * fp_fwd top_fwd ;
RuledSurf cabin_side 11 1 20×1 1×1 0 / * fp_side top_side ;
RuledSurf cabin_aft 11 1 10×1 1×1 0 / * fp_aft top_aft ;
CCurve top_ctr 11 1 10×1 / * 2 { rp1 rp7 rp6 } ;
BlendSurf cabin_top 14 1 4×2 5×2 0 / * *
  { top_fwd top_side top_aft top_ctr } ;
XContours stations 12 1 / 0 10 1.949 2.733
  { hull deck cabin_fwd cabin_side cabin_aft cabin_top } ;
RuledSurf transom 3 1 8×2 1×1 0 / * transom_top MCC ;
EndModel;
```

ter curves; the first is the single point **A1**, and the other two are three-vertex B-Spline curves **deck_beam** and **transom**. The join between **hull** and **deck** is exact and durable because the C-Splines at the adjoining edges on each surface use the same data points, namely, **A1**, **B1**, **C1**, and therefore are identical curves.

The three ruled surfaces **cabin_fwd**, **cabin_side**, **cabin_aft** are constructed in a similar fashion to one another: Each uses a snake on **deck** as one edge, providing an exact and durable join to the **deck** surface, and a relative curve dependent on that snake as the second (upper) edge. The three snakes on **deck** join each other exactly and durably because they share common endpoints, namely, magnets **dm3** and **dm5**. The three relative curves **top_fwd**, **top_side**, **top_aft** also join each other exactly and durably because they are constructed using common endpoints, namely, relative points **rp3** and **rp5**. **cabin_side** joins the other two surfaces exactly because its end rulings are the lines **dm3 - rp3** and **dm5 - rp5**, which are identical to end rulings on the adjoining surfaces.

Figure 12.3 Example of model of Table t2.t, comprising six accurately conjoined surface objects.

The blended surface **cabin_top** joins the three ruled surfaces accurately because it uses their upper edge curves **top_fwd**, **top_side**, **top_aft** as data, interpolating each of them. Its fourth side is a three-point C-Spline **top_ctr**, which lies exactly in the center plane because each of its vertices has a zero y coordinate.

Exact and durable surface joins are accomplished in this model by three relational techniques:

1. Two transfinite surfaces sharing a common curve as edges, for example, **cabin_side** to **cabin_top**.
2. A transfinite surface connected to a snake embedded in another surface, for example, **cabin_side** to **deck**.
3. Two identically ruled or lofted surfaces with master curves that share common endpoints, for example, **hull** to **deck** or **cabin_side** to **cabin_fwd**.

12.7 APPLICATIONS

Parametric Variations

The example model of Section 12.6 serves as a basis for illustrating the potential of RG for constructing parametrically variable models. This model can be transformed, with preservation of its topology, into an extremely wide variety of alternative shapes by changing the coordinates of absolute points, the offsets of relative points, and the parameters and parameter offsets of rings and magnets. An example modification which affects all six surfaces is to increase the y coordinate of AbsPoint **B1**. Following this change, the con-

nectivity and relative positioning of the several surfaces is automatically preserved as the absolute model is updated (see Figure 12.4).

A further example modification is to drag AbsRing **r0** to a new position (see Figure 12.5). **r8** belongs to an EdgeSnake along the centerline edge of **deck**. Because RelRing **r1** is relative to **r8**, and the several **dm** magnets are relative to these two rings, the entire cabin structure transports intact to a new location where it continues to join the deck precisely. Also, **r1** can be dragged to change the length of the cabin.

The parametric handles tend to be the floating-point data components of various kinds of points. This has the advantage that most of the handles are visible points which can be selected and dragged, as in the preceding examples.

Discretization

RG is very well adapted to creating surface and volume discretizations for both boundary integral and finite-element analysis methods. Discretizations can be constructed in such a way that they are topologically invariant with respect to parametric variations of the underlying geometry.

Boundary integral (panel) methods are widely used for problems governed by linear field equations and boundary conditions, and are especially advantageous for domains which are unbounded in some directions; for example, irrotational hydrodynamic and aerodynamic flows, acoustics, and electrostatic and electromagnetic problems. The discretized geometric representation consists of an assemblage of small triangular or quadrilateral *panels* tessellating the exposed surfaces.

When a surface patch is completely exposed, it is simple to partition it into quadrilateral or triangular panels along its parametric mesh lines. Relabeling of curves and surfaces provides a facility for parametric control of mesh size distribution, as may be required for solution accuracy. When a surface patch

Figure 12.4 Example model following a change in the *y* coordinate of point **B1**.

Figure 12.5 Example model following a change in the *t* parameter of ring **r0**.

is partially covered by one or more other patches, the SubSurf entity provides a way to panelize the exposed portion with a quadrilateral mesh that adheres to the underlying surface. Use of common edge curves or snakes for surface and SubSurf construction typically provides exact conforming joins between the panels on adjoining surfaces.

Figure 12.6 shows a propeller model panelized for potential flow analysis. The model has five-fold global symmetry, so only one blade and a helical portion (⅕) of the shaft and hub are explicitly modeled. The blade is a BLoftSurf with seven master curves: a snake on the shaft plus six FoilSnakes lying on concentric cylinders. A Fillet surface blends between a snake on the blade

Figure 12.6 Panelized model of a propeller for potential flow analysis.

and one on the shaft. One SubSurf on the blade, the complete Fillet, and four SubSurfs on the shaft/hub complete the panelization. Many variations in pitch, skew, rake, chord, thickness, and camber distributions are possible with the topology of the panelization being completely conserved.

Finite element methods are widely used for problems governed by nonlinear field equations, or in bounded domains, for example, viscous flows, heat conduction, and stress analysis. The discretized representation consists of a partition of the problem domain into subdomains (finite elements), typically tetrahedral or hexahedral cells. Parametric solids are easily divided into conforming hexahedral cells along parametric mesh surfaces; a hexahedral cell can be further divided into five tetrahedra.

Optimal Design

The ability to generate parametrically variable models with topologically invariant discretizations opens the way to automated analysis and optimization cycles. One of the most advanced applications of RG to date was the development of New Zealand's AC class sailing yacht *Black Magic,* winner of the 1995 America's Cup. Over 2,000 hull and appendage variations were simulated and evaluated in a systematic search for the optimum competitive configuration.

As in any simulation-based optimal design process, it is important in yacht design to simulate the complete system, rather than any particular subsystem (Oliver [1987]). In this case the simulation consisted of the following steps:

1. Choice of geometric model parameters.
2. Generation of discretized model.
3. Simulation of AC class rule measurement, yielding allowable sail areas.
4. Computational fluid dynamic (CFD) analysis, yielding coefficients for hydrodynamic lift, drag, and moments.
5. Equilibrium performance analysis for the complete yacht (Velocity Prediction Program [VPP]), yielding predicted speeds for different headings and wind speeds.
6. Race course traversal, yielding elapsed times versus wind speeds.
7. Simulation of two-yacht races versus other candidate designs in the predicted distribution of wind speeds (Race Model Program), yielding win/loss probabilities for pairs of yachts.
8. Simulation of scoring system for the entire race series, yielding win/loss probability for the series.

The win probability for the series is of course the single objective for optimization.

In a competition involving the world's top-level yacht designers, builders, and sailors from eight countries, and intense technical development on all sides, *Black Magic* was a spectacular success. She won 35 out of 36 elimination races, and swept the final contest with 5 decisive victories over the U.S. defender, taking the America's Cup away from the United States for only the second time in the 144-year history of the event.

Animation

Relational models having parameters which cause them to deform in mechanically realistic ways are a potentially powerful basis for creating animated scenes. Once a model is developed with a number of parametric handles, it can be controlled by programming the handles versus time, like a robot. Of course, arbitrary control programs may produce unphysical or unrealistic motions. The integration of physics and physiology with relational geometry models to produce realistic animations would be an interesting and possibly very fruitful topic.

Figure 12.7 shows a relational shark model which is able to "swim." Two parameters (*y* offsets of two RelPoints) control the first (half-cycle) and second (full-cycle) shape modes of the spine curve. The body is a SweepSurf using the spine as path. Each of the five fins is also a SweepSurf, sweeping a snake on the body along a path. The two handles are programmed with out-of-phase sinusoidal variations to produce the swimming action.

Developable Surfaces

The efficient welded or riveted metal construction of such large complex shell structures as ships and airplanes requires high precision in metal cutting and forming processes. Fabrication is greatly simplified when the surfaces are *developable,* that is, they can be formed from flat sheet material by bending alone, rather than with any degree of in-plane stretching.

MultiSurf provides several surface entities that can construct cylinders and cones, plus the DevSurf entity that generates explicitly developable surfaces spanning two supporting curves. Developable surfaces made by any of these methods can be accurately flattened (*developed*) onto the plane, with their dependent magnets, snakes, and contours intact. Figure 12.8 is an example of this capability.

Compound Plate Layout

Similarly, construction of nondevelopable (*compound-curved*) shell structures can greatly benefit from the modeling features and precision of RG.

Figure 12.7 A relational shark model for demonstrating potential of RG for animation studies.

Figure 12.9 shows an example of this application. First, a single fair surface is designed for the complete (nondevelopable) hull surface of a vessel. Snakes are used to represent weld seams dividing the surface up into 12 plates, each of which will be individually formed from a flat *blank* by some *compounding* process before assembly. Each plate is covered with a SubSurf. The three-dimensional geometry of each SubSurf is exported to an auxiliary program which simulates the forming process in reverse, yielding a flattened outline for accurate cutting of the blank from flat material, plus strain maps which can guide the application of compounding tools (Letcher [1993b]).

(a)

(b)

Figure 12.8 (*a*) MultiSurf model for hard-chine sailboat made up of 9 developable surface panels. (*b*) MSDEV expanded panels for hard-chine sailboat, with positions of stations, waterlines, and buttocks marked.

Figure 12.9 Subdivision of compound-curved surface into individual plates for expansion and fabrication.

CNC Machining

Numerical-controlled machining of sculptured parts requires a precise description of the geometry for generating tool paths. In the future, RG could well come into use as the fundamental geometry standard for tool-path generation. Today, NURBS surfaces transmitted through IGES files are the most common form of part description. In order to communicate with NURBS-based CAM and other applications, RG surfaces must be exported in the form of NURBS.

Since most RG surfaces are in fact not exactly representable by NURBS, in most cases approximation is required, with some loss of accuracy. This is done by least-squares fitting to a tabulated mesh. Figure 12.10 shows a CNC machined part derived from a MultiSurf model. The part is a custom marine hardware item (a bow chock) designed by Peter Smith for a production power yacht built by the Hinckley Co., Southwest Harbor, Maine.

12.8 OPEN ISSUES IN RELATIONAL GEOMETRY

Logical Completeness

During the first years of development of RG (1991 to 1992), the number of entities implemented grew rapidly, and there was always a long list of entities

Figure 12.10 Chock designed in two dimensions by Peter Smith (The Hinckley Co.). Model for NC milling made in MultiSurf by George Dickson (Aero-Hydro). Rendering made with POV-Ray by Ed Stanley (AeroHydro).

clamoring for inclusion. Sometimes it appeared that the growth in entity population might be exponential if we implemented every useful construction. In later years, the growth became much slower as we seemed to approach a state of having the bases covered; that is, few modeling problems came to our attention which could not be accomplished with some combination of the available entities, about 100 in number. In particular, the procedural entities covered a whole set of outstanding problems, and could obviate a fair number of existing specialized entities. Although we still have a list of proposed entities waiting, we also have a sense that some level of logical completeness may not be far away.

Modularity in Relational Models

The digraph model structure is potentially complex and hard to visualize. As the number of objects increases, relational models become progressively (evidently, much more than linearly) more difficult to comprehend. This is similar to problems addressed by the trend to modularity in programming

languages. It would appear very beneficial to be able to encapsulate groups of related objects into *modules,* using *local* intermediate objects and limited, well-defined interfaces to other modules. The representation and manipulation of such a new layer of structure appears practically essential to moving much beyond the 1,000- to 2,000-object level of complexity in relational models.

Subroutines

Many models are characterized by having similar, sometimes identical, constructions repeated in different locations in the model. The spars and sails of a square-rigged ship come to mind as a practical example. Currently, such constructions require duplication of groups of objects, with minor variations; often, we construct such models by writing a LISP, C, or Basic program that generates a model file fragment. The concept of a *subroutine* or *procedure* in the model representation that would encapsulate the construction of one or more objects, and be called with varying parameters, seems to offer a useful and highly attractive alternative.

Variables, Expressions, and Functions

Many entities, particularly in the point class, have data fields filled by real (floating-point) numbers. It is not hard to see potential advantage in allowing these fields to be filled with numeric variables or expressions, as well as constants. By adding a Float entity

Float name ..... / expression;

the dependency of an object on one or more numeric variables can be represented and maintained within the usual dependency relation. Besides a set of numeric functions such as sine and cosine, some numeric-valued functions of objects such as Distance(point1, point2) would be useful. We have tentatively implemented variables, expressions, and functions in a development version of RGKernel, but user-interface issues prevented inclusion in the current product release.

Constraint Equations

RG accomplishes some, but definitely not all, of the parametric capabilities typical of constraint-based modeling through its geometric constructions. With numeric variables and expressions it should be possible to add a quite general constraint equation and solution capability. This has not been explored.

B-Rep Solid Modeling

The predominant method of solid modeling today is a boundary representation, comprising a description of edge-face topology combined with geometry of faces, typically trimmed portions of parametric or implicit surfaces. Modeling proceeds primarily by Boolean operations and analogs of machining operations, operating on instances of a small set of primitive solids. While this approach appears to be quite successful in the design of mechanical parts, it has not shown much ability to incorporate free-form, sculptured surfaces in solid models.

Our current, short-term approach to connection with b-rep solid modeling is via NURBS approximation. MultiSurf serves as a relational, interactive front end in which complex surface geometries can be modeled and varied. These surfaces are exported as NURBS to a solid modeler where, provided the accuracy is high enough, they can be united into shells and subjected to Boolean operations and so forth. Desirable additions to facilitate this approach are two new entities:

BFitSurf name ..... / u-type nu v-type nv tolerance surface;

The B-Spline fitted surface is a least-squares approximation to **surface**, using B-Spline types **u-type** and **v-type** for the *u* and *v* directions, respectively. The number of control points *nu*, *nv* for each direction can be specified, or if specified as 0, will start at **type**+1 and be increased as necessary to meet the specified tolerance. Statistics on the quality of fit and the number of control points used are available by interrogating the BFitSurf. The fitted surface object can be viewed, of course, like any other surface, and can support other geometry. On export, no further fitting is performed.

TrimmedSurf name ..... / magnet { snakes };

The trimmed surface is the portion of the host surface of **magnet** that can be reached without crossing any of the **snakes**. Since the trimmed surface (unlike a SubSurf) does not possess a regular parametric domain, this entity is subclassed from RGObject instead of RGSurface.

It may also be advantageous to implement a class of implicit (nonparametric) surface entities comprising the typical solid-modeling primitives, so these can be constructed in a relational fashion and positioned and viewed in MultiSurf before being exported to the solid modeler.

In the longer range, solid modeling should benefit from adopting the RG approach to surface modeling and providing native support for a significant set of RG entities. This would greatly enrich the capability of solid modelers

to deal with free-form surfaces, and would solve many of the precision problems that arise from approximation-based NURBS modeling.

12.9 CONCLUSIONS

Relational geometric synthesis is a design framework that permits and encourages the capture of qualitative and quantitative relationships between the geometric elements of a CAD model. With roughly the same amount of effort as is required to create a single, static design in a conventional CAD program, RGS allows the creation of a relational model having many conserved topological and geometric properties, combined with many degrees of freedom for parametric variation and refinement—"live" versus "dead" geometry.

We propose RGS as a new standard framework for surface modeling, providing flexibility and accuracy far beyond NURBS-restricted methods. Promising connections to solid modeling, engineering analysis, CNC manufacturing, and animation are being developed and explored.

This page intentionally left blank

APPENDIX A

VECTORS

In this text, a vector is an ordered set of numbers from which we derive the properties of direction and magnitude or length. We denote vectors with boldface, lowercase letters, and visualize them as a directed line segments, or arrows. A vector with zero magnitude is a null vector, and we denote it by **0** (no direction is defined for it).

The traditional form is

$$\mathbf{p} = p_x\mathbf{i} + p_y\mathbf{j} + p_z\mathbf{k} \tag{A.1}$$

where the **i**, **j**, **k** are mutually orthogonal unit *basis vectors* (see Figure A.1). In general, basis vectors need not be mutually orthogonal nor of unit magnitude. The p_x, p_y, p_z are vector components representing displacements in the x, y, z directions, where **i**, **j**, **k** occur along the x, y, z axes, respectively. A *free vector* is not necessarily associated with any point in space, while a *position vector* or *radius vector* has its initial point fixed at the origin so that its components are equivalent to the coordinates of a point.

In matrix form, Equation (A.1) is

$$\mathbf{p} = [p_x \quad p_y \quad p_z]\begin{bmatrix}\mathbf{i}\\ \mathbf{j}\\ \mathbf{k}\end{bmatrix} \tag{A.2}$$

or, alternatively,

$$\mathbf{p} = [\mathbf{i} \quad \mathbf{j} \quad \mathbf{k}]\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{A.3}$$

To simplify the matrix form, we omit the $\mathbf{i}, \mathbf{j}, \mathbf{k}$, so that

$$\mathbf{p} = [x \quad y \quad z] \qquad \text{or} \qquad \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{A.4}$$

This is particularly appropriate for the position or radius vector, where we interpret the components, now matrix elements, as point coordinates.

In the *tensor* form, the superscript ($i = 1, 2,$ or 3) on x identifies the component as x, y, or z, respectively, where $\mathbf{e}_i$ is the corresponding basis vector.

$$\begin{aligned} \mathbf{x} &= \sum_{i=1}^{3} x^i \mathbf{e}_i \\ &= x^1\mathbf{e}_1 + x^2\mathbf{e}_2 + x^3\mathbf{e}_3 \end{aligned} \tag{A.5}$$

or simply

$$\mathbf{x} = x^i \mathbf{e}_i \tag{A.6}$$

Figure A.1 Traditional vector form.

The repetition of the index i indicates summation. This is the *Einstein convention* used in tensor analysis.

Two vectors **a** and **b** are equal if and only if their respective components are equal. Thus, if $a_x = b_x$, $a_y = b_y$, and $a_z = b_z$, then $\mathbf{a} = \mathbf{b}$. Fixed vectors (or radius or position vectors) are equal if and only if their respective end points coincide. Free vectors are equal if they can be made to coincide by a translation.

W. R. Hamilton (1805–1865) and H. Grassman (1809–1877) developed the foundations of vector analysis in the mid-nineteenth century, and J. W. Gibbs (1839–1903) made important contributions to the present form of vector analysis.

Vector Magnitude

The *magnitude* (also *length* or *norm*) of a vector $\mathbf{p} = [p_x \quad p_y \quad p_z]$ is the real number $|\mathbf{p}|$, where

$$|\mathbf{p}| = \sqrt{p_x^2 + p_y^2 + p_z^2} \tag{A.7}$$

It is always the case that $|\mathbf{p}| \geq 0$. Furthermore, $|\mathbf{p}| = 0$, if and only if $\mathbf{p} = 0$; that is, $\mathbf{p} = [0 \quad 0 \quad 0]$, the null vector.

Multiplying a vector by a scalar changes its magnitude. Thus

$$k\mathbf{p} = [kp_x \quad kp_y \quad kp_z] \tag{A.8}$$

and

$$\begin{aligned} |k\mathbf{p}| &= k\sqrt{p_x^2 + p_y^2 + p_z^2} \\ &= k|\mathbf{p}| \end{aligned} \tag{A.9}$$

If $k < 0$, the direction of $\mathbf{p}$ is reversed.

The magnitude of a vector is invariant under rigid-body transformations; that is, magnitude is independent of direction.

Vector Direction

The direction of a vector is given by its *direction cosines* (or *direction numbers*): $\cos\alpha$, $\cos\beta$, $\cos\gamma$ (see Figure A.2).

$$\begin{aligned} \cos\alpha &= \frac{p_x}{|\mathbf{p}|} \\ \cos\beta &= \frac{p_y}{|\mathbf{p}|} \\ \cos\gamma &= \frac{p_z}{|\mathbf{p}|} \end{aligned} \tag{A.10}$$

Figure A.2 Vector direction.

Since $|\mathbf{p}| = \sqrt{p_x^2 + p_y^2 + p_z^2}$, then

$$\left(\frac{p_x}{|\mathbf{p}|}\right)^2 + \left(\frac{p_y}{|\mathbf{p}|}\right)^2 + \left(\frac{p_x}{|\mathbf{p}|}\right)^2 = 1$$

so that

$$\cos^2\alpha + \cos^2\beta + \cos^2\gamma = 1 \tag{A.11}$$

This means that any two direction cosines are sufficient to determine the direction of a vector.

Unit Vector

The following notation convention is useful: If $\mathbf{p}$ is any vector, then $\hat{\mathbf{p}}$ denotes its *unit vector*. The unit vector $\hat{\mathbf{p}}$ in the direction of $\mathbf{p}$ is a vector having a magnitude equal to one; $|\hat{\mathbf{p}}| = 1$. It is given by

$$\hat{\mathbf{p}} = \frac{\mathbf{p}}{|\mathbf{p}|} \tag{A.12}$$

This means that

$$\hat{p}_x = \frac{p_x}{|\mathbf{p}|}, \qquad \hat{p}_y = \frac{p_y}{|\mathbf{p}|}, \qquad \hat{p}_z = \frac{p_z}{|\mathbf{p}|}$$

and

$$|\hat{\mathbf{p}}| = \sqrt{\hat{p}_x^2 + \hat{p}_y^2 + \hat{p}_z^2}$$
$$= 1$$

Obviously

$$\hat{p}_x^2 + \hat{p}_y^2 + \hat{p}_z^2 = 1$$

Using Equation (A.10) and substituting appropriately produces

$$\hat{p}_x = \cos\alpha$$
$$\hat{p}_y = \cos\beta$$
$$\hat{p}_z = \cos\gamma$$

This demonstrates that the components of a unit vector are also its direction cosines.

Basis Vectors

In a vector space the set of vectors $\mathbf{e}_i$ is a *basis* if we can express every vector in that space as a linear combination of the basis vectors and if the $\mathbf{e}_i$ are linearly independent. In three-dimensional Cartesian vector space any three linearly independent vectors form a basis. The unit vectors $\mathbf{i}, \mathbf{j}, \mathbf{k}$ form such a basis because $\mathbf{i} = [1 \quad 0 \quad 0]$, $\mathbf{j} = [0 \quad 1 \quad 0]$, and $\mathbf{k} = [0 \quad 0 \quad 1]$ are linearly independent. The components of a vector depend on the basis chosen, and, in general, the components change if the basis changes.

Vector Addition

Given two vectors $\mathbf{p} = [p_x \quad p_y \quad p_z]$ and $\mathbf{q} = [q_x \quad q_y \quad q_z]$, their sum is the vector (see Figure A.3a)

$$\mathbf{p} + \mathbf{q} = [(p_x + q_x) \quad (p_y + q_y) \quad (p_z + q_z)] \tag{A.13}$$

The difference of two vectors is (see Figure A.3b)

$$\mathbf{p} - \mathbf{q} = [(p_x - q_x) \quad (p_y - q_y) \quad (p_z - q_z)] \tag{A.14}$$

Order is not important. If $\mathbf{d} = \mathbf{a} + \mathbf{b} + \mathbf{c}$, then it is also true that $\mathbf{d} = \mathbf{a} + \mathbf{c} + \mathbf{b}$. This means that vector addition is *commutative*.

A graphic interpretation is: To add two or more vectors (drawn as arrows), join the tail of each succeeding vector to the head of the preceding one. The

Figure A.3 Vector addition.

resultant vector is represented by an arrow from the tail of the first vector to the head of the last.

Scalar Product

The *scalar product* (or *inner product*) of two vectors **p** and **q** is the sum of the products of their corresponding components:

$$\mathbf{p} \cdot \mathbf{q} = p_x q_x + p_y q_y + p_z q_z \tag{A.15}$$

If $\mathbf{p} = \mathbf{q}$, then $\mathbf{p} \cdot \mathbf{p} = |\mathbf{p}|^2$. If $\mathbf{p} \cdot \mathbf{q} = 0$, they are perpendicular. The various scalar products of the basis vectors **i**, **j**, and **k** produce the following:

$$\begin{aligned} \mathbf{i} \cdot \mathbf{i} &= \mathbf{j} \cdot \mathbf{j} = \mathbf{k} \cdot \mathbf{k} = 1 \\ \mathbf{i} \cdot \mathbf{j} &= \mathbf{j} \cdot \mathbf{k} = \mathbf{k} \cdot \mathbf{i} = 0 \\ \mathbf{i} \cdot \mathbf{k} &= \mathbf{j} \cdot \mathbf{i} = \mathbf{k} \cdot \mathbf{j} = 0 \end{aligned} \tag{A.16}$$

The scalar product written in tensor notation is

$$\mathbf{p} \cdot \mathbf{q} = p^j q^k \delta_{jk} \tag{A.17}$$

where δ_{jk} is the *Kronecker delta,* and

$$\delta_{jk} = \begin{cases} 1 & \text{if} \quad j = k \\ 0 & \text{if} \quad j \neq k \end{cases} \tag{A.18}$$

The angle θ between two vectors **p** and **q** satisfies the following equation:

$$\mathbf{p} \cdot \mathbf{q} = |\mathbf{p}||\mathbf{q}| \cos \theta \tag{A.19}$$

so that

$$\theta = \cos^{-1} \frac{\mathbf{p} \cdot \mathbf{q}}{|\mathbf{p}||\mathbf{q}|} \tag{A.20}$$

The scalar product of two vectors is independent of the coordinate system in which they are expressed.

Vector Product

The *vector product* of two vectors **p** and **q** is another vector **r**:

$$\mathbf{p} \times \mathbf{q} = \mathbf{r} \tag{A.21}$$

where

$$\mathbf{r} = [(p_y q_z - p_z q_y) \quad (p_z q_x - p_x q_z) \quad (p_x q_y - p_y q_x)] \tag{A.22}$$

It is easy to show that $\mathbf{p} \times \mathbf{p} = \mathbf{0}$. Furthermore, if $\mathbf{p} \times \mathbf{q} = \mathbf{0}$, then **p** and **q** are parallel.

The expansion of the following determinant also produces the vector product:

$$\mathbf{p} \times \mathbf{q} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ p_x & p_y & p_z \\ q_x & q_y & q_z \end{vmatrix} \tag{A.23}$$

The angle θ between **p** and **q** is a by-product of the vector product:

$$|\mathbf{p} \times \mathbf{q}| = |\mathbf{p}||\mathbf{q}| \sin \theta$$

so that

$$\theta = \sin^{-1} \frac{|\mathbf{p} \times \mathbf{q}|}{|\mathbf{p}||\mathbf{q}|} \tag{A.24}$$

The *right-hand rule* gives an intuitive sense of the direction of the vector **r** resulting from $\mathbf{p} \times \mathbf{q}$. Think of rotating **p** into **q**, curling the fingers of your right hand in this angular direction. Then the extended thumb of your right hand will point in the direction of **r** (see Figure A.4).

Figure A.4 Vector product.

Vector products of the i, **j**, **k** basis vectors produce

$$\begin{aligned} \mathbf{i} \times \mathbf{j} &= \mathbf{k} \\ \mathbf{j} \times \mathbf{k} &= \mathbf{i} \\ \mathbf{k} \times \mathbf{i} &= \mathbf{j} \\ \mathbf{i} \times \mathbf{i} = \mathbf{j} \times \mathbf{j} &= \mathbf{k} \times \mathbf{k} = \mathbf{0} \end{aligned} \tag{A.25}$$

The vector product is useful in finding the area of a triangle (see Figure A.5). The area of the triangle ABC is

$$\text{Area } \Delta ABC = \frac{1}{2}|\mathbf{a} \times \mathbf{b}| \tag{A.26}$$

Triple Product

The *triple scalar product* $\mathbf{p} \cdot \mathbf{q} \times \mathbf{r}$ given in terms of a determinant is

$$\mathbf{p} \cdot \mathbf{q} \times \mathbf{r} = \begin{vmatrix} p_x & q_x & r_x \\ p_y & q_y & r_y \\ p_z & q_z & r_z \end{vmatrix} \tag{A.27}$$

and it is easy to show that $\mathbf{p} \cdot \mathbf{q} \times \mathbf{r} = \mathbf{p} \times \mathbf{q} \cdot \mathbf{r}$.

An alternative tensor form is

$$\mathbf{p} \cdot \mathbf{q} \times \mathbf{r} = \xi_{ijk} p^i q^j r^k \tag{A.28}$$

where

$$\xi_{ijk} = \begin{Bmatrix} 1 \\ -1 \\ 0 \end{Bmatrix} \quad \text{if} \quad i,j,k \text{ is an } \begin{Bmatrix} \text{even} \\ \text{odd} \\ \text{other} \end{Bmatrix} \text{ permutation of } 1,2,3.$$

Figure A.5 Area of a triangle in vector form.

The *triple vector product* yields

$$\mathbf{p} \times (\mathbf{q} \times \mathbf{r}) = (\mathbf{p} \cdot \mathbf{r})\mathbf{q} - (\mathbf{p} \cdot \mathbf{q})\mathbf{r} \tag{A.29}$$

where the result of $\mathbf{p} \times (\mathbf{q} \times \mathbf{r})$ lies in the plane of $\mathbf{q}$ and $\mathbf{r}$.

Vector Projection

The scalar projection s of $\mathbf{p}$ onto $\mathbf{q}$ is (see Figure A.6)

$$s = \mathbf{p} \cdot \left(\frac{\mathbf{q}}{|\mathbf{q}|}\right) = |\mathbf{p}| \cos \theta \tag{A.30}$$

The vector projection $\mathbf{v}$ of $\mathbf{p}$ onto $\mathbf{q}$ is (see Figure A.7)

$$\mathbf{v} = s\left(\frac{\mathbf{q}}{|\mathbf{q}|}\right) \tag{A.31}$$

Vector Equation of a Line

The vector equation of a line $\mathbf{p}(u)$ through point $\mathbf{p}_0$ and parallel to vector $\mathbf{t}$ is (see Figure A.8)

$$\mathbf{p}(u) = \mathbf{p}_0 + u\mathbf{t} \tag{A.32}$$

or, in component form,

$$\begin{aligned} x(u) &= x_0 + ut_x \\ y(u) &= y_0 + ut_y \\ z(u) &= z_0 + ut_z \end{aligned} \tag{A.33}$$

An alternative form is

$$\mathbf{p}(u) = \mathbf{p}_0 + u(\mathbf{p}_1 - \mathbf{p}_0) \tag{A.34}$$

where if $u \in [0,1]$, then Equation (A.34) defines a line segment from $\mathbf{p}_0$ to $\mathbf{p}_1$ (see Figure A.9).

If $\mathbf{q}(w) = \mathbf{q}_0 + w\mathbf{s}$ and $\mathbf{s} = k\mathbf{t}$, then the lines through $\mathbf{p}_0$ and $\mathbf{q}_0$ are parallel.

Figure A.6 Scalar projection.

Figure A.7 Vector projection.

Figure A.8 Vector equation of a line.

Figure A.9 Vector equation of a line segment.

Vector Equation of a Plane

The vector equation of a plane $\mathbf{p}(u,w)$ through $\mathbf{p}_0$ and parallel to two independent vectors $\mathbf{s}$ and $\mathbf{t}$ is (see Figure A.10)

$$\mathbf{p}(u,w) = \mathbf{p}_0 + u\mathbf{s} + w\mathbf{t} \tag{A.35}$$

with the condition $\mathbf{s} \neq k\mathbf{t}$. In component form this is

$$\begin{aligned} x &= x_0 + us_x + wt_x \\ y &= y_0 + us_y + wt_y \\ z &= z_0 + us_z + wt_z \end{aligned} \tag{A.36}$$

The unit normal $\hat{\mathbf{n}}$ to the plane is the vector product

$$\hat{\mathbf{n}} = \hat{\mathbf{s}} \times \hat{\mathbf{t}} \tag{A.37}$$

Three noncollinear points $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$ are also sufficient to uniquely define a plane (see Figure A.11). Thus,

$$\mathbf{p}(u,w) = \mathbf{p}_0 + u(\mathbf{p}_1 - \mathbf{p}_0) + w(\mathbf{p}_2 - \mathbf{p}_1) \tag{A.38}$$

The unit normal for this formulation is

$$\hat{\mathbf{n}} = \frac{(\mathbf{p}_1 - \mathbf{p}_0) \times (\mathbf{p}_2 - \mathbf{p}_1)}{|(\mathbf{p}_1 - \mathbf{p}_0) \times (\mathbf{p}_2 - \mathbf{p}_1)|} \tag{A.39}$$

Figure A.10 Vector equation of a plane.

Figure A.11 A plane containing three noncollinear points.

Vector Equation of a Plane: Normal Form

One way to define a plane is by specifying a vector **d** from the origin to a point on it such that **d** is perpendicular to the plane (see Figure A.12). Then any point **p** on the plane must satisfy

$$(\mathbf{p} - \mathbf{d}) \cdot \mathbf{d} = 0 \tag{A.40}$$

Expanding this equation produces

$$(x - d_x)d_x + (y - d_y)d_y + (z - d_z)d_z = 0$$

or

$$d_x x + d_y y + d_z z - (d_x^2 + d_y^2 + d_z^2) = 0 \tag{A.41}$$

Figure A.12 Vector equation of a plane: normal form.

Because $d^2 = d_x^2 + d_y^2 + d_z^2$, $\cos\alpha = d_x/d$, $\cos\beta = d_y/d$, and $\cos\gamma = d_z/d$, then Equation A.41 reduces to

$$x\cos\alpha + y\cos\beta + z\cos\gamma - d = 0 \tag{A.42}$$

where $\cos\alpha$, $\cos\beta$, $\cos\gamma$ are the direction cosines of $\mathbf{d}$.

Solution of Vector Equations

The vector equation $\mathbf{a} + u\mathbf{b} + w\mathbf{c} = \mathbf{d} + t\mathbf{e}$ represents a system of three linear equations in three unknowns: u, w, and t. In component form these equations are

$$a_x + ub_x + wc_x = d_x + te_x$$

$$a_y + ub_y + wc_y = d_y + te_y$$

$$a_z + ub_z + wc_z = d_z + te_z$$

We use their vector form to solve for u, w, and t by isolating each in turn. For example, to isolate t, we apply $(\mathbf{b} \times \mathbf{c})$ as follows:

$$(\mathbf{b} \times \mathbf{c}) \cdot (\mathbf{a} + u\mathbf{b} + w\mathbf{c}) = (\mathbf{b} \times \mathbf{c}) \cdot (\mathbf{d} + t\mathbf{e})$$

Because $(\mathbf{b} \times \mathbf{c})$ is perpendicular to both $\mathbf{b}$ and $\mathbf{c}$, then

$$(\mathbf{b} \times \mathbf{c}) \cdot \mathbf{a} = (\mathbf{b} \times \mathbf{c}) \cdot \mathbf{d} + (t\mathbf{b} \times \mathbf{c}) \cdot \mathbf{e}$$

Solving this equation for t produces

$$t = \frac{(\mathbf{b} \times \mathbf{c}) \cdot \mathbf{a} - (\mathbf{b} \times \mathbf{c}) \cdot \mathbf{d}}{(\mathbf{b} \times \mathbf{c}) \cdot \mathbf{e}}$$

Continuing this method yields similar expressions for u and w:

$$u = \frac{(\mathbf{c} \times \mathbf{e}) \cdot \mathbf{d} - (\mathbf{c} \times \mathbf{e}) \cdot \mathbf{a}}{(\mathbf{c} \times \mathbf{e}) \cdot \mathbf{b}}$$

$$w = \frac{(\mathbf{b} \times \mathbf{e}) \cdot \mathbf{d} - (\mathbf{b} \times \mathbf{e}) \cdot \mathbf{a}}{(\mathbf{b} \times \mathbf{e}) \cdot \mathbf{c}}$$

Summary of Vector Properties

Given vectors $\mathbf{p}$, $\mathbf{q}$, $\mathbf{r}$ and scalars k and l, then

1. $\mathbf{p} + \mathbf{q} = \mathbf{q} + \mathbf{p}$
2. $\mathbf{p} + (\mathbf{q} + \mathbf{r}) = (\mathbf{p} + \mathbf{q}) + \mathbf{r}$
3. $k(l\mathbf{p}) = kl\mathbf{p}$

4. $(k + l)\mathbf{p} = k\mathbf{p} + l\mathbf{p}$
5. $k(\mathbf{p} + \mathbf{q}) = k\mathbf{p} + k\mathbf{q}$
6. $\mathbf{p} \cdot \mathbf{p} = |\mathbf{p}|^2$
7. $\mathbf{p} \cdot \mathbf{r} = \mathbf{r} \cdot \mathbf{p}$
8. $\mathbf{p} \cdot \mathbf{r} = |\mathbf{p}||\mathbf{r}| \cos \theta$
9. $\mathbf{p} \cdot (\mathbf{r} + \mathbf{q}) = \mathbf{p} \cdot \mathbf{r} + \mathbf{p} \cdot \mathbf{q}$
10. $(k\mathbf{p}) \cdot \mathbf{r} = \mathbf{p} \cdot (k\mathbf{r}) = k(\mathbf{p} \cdot \mathbf{r})$
11. If $\mathbf{p} \cdot \mathbf{r} = 0$, they are perpendicular.
12. If $\mathbf{p} \times \mathbf{q} = \mathbf{r}$, then $\mathbf{r}$ is perpendicular to both $\mathbf{p}$ and $\mathbf{q}$.
13. $\mathbf{p} \times \mathbf{r} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ p_x & p_y & p_z \\ q_x & q_y & q_z \end{vmatrix}$
14. $\mathbf{p} \times \mathbf{r} = -(\mathbf{r} \times \mathbf{p})$
15. $\mathbf{p} \times (\mathbf{q} + \mathbf{r}) = \mathbf{p} \times \mathbf{q} + \mathbf{p} \times \mathbf{r}$
16. $(k\mathbf{p}) \times \mathbf{r} = \mathbf{p} \times (k\mathbf{r}) = k(\mathbf{p} \times \mathbf{r})$
17. If $\mathbf{p} \times \mathbf{r} = \mathbf{0}$, they are parallel.
18. $\mathbf{p} \times \mathbf{p} = \mathbf{0}$

APPENDIX B

MATRICES

A matrix is a rectangular array of numbers or other mathematical elements arranged in m rows and n columns. For example:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}$$

where a_{ij} is the matrix element in the ith row and jth column.

The number of rows and columns determines the *order* of a matrix. A matrix of m rows and n columns is of order $m \times n$. Then $i \in [1 : m]$ and $j \in [1 : n]$. The order of the largest square submatrix whose determinant has a nonzero value is the *rank* of the matrix.

Two matrices **A** and **B** are equal if corresponding elements are equal. This means that $\mathbf{A} = \mathbf{B}$ if $a_{ij} = b_{ij}$ for all i, j.

A matrix is a *square matrix* if $m = n$, a *row matrix* if $m = 1$, and a *column matrix* if $n = 1$. Single row or column matrices are often called *vectors*.

Matrices have a unique algebra that governs operations on them, and it was the English mathematician Arthur Cayley (1821–1895) who developed and refined the algebra of matrices.

Special Matrices

A *diagonal matrix* is a square matrix that has zero elements everywhere except on the main diagonal. For example,

$$\mathbf{A} = \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ 0 & 0 & a_{33} & 0 \\ 0 & 0 & 0 & a_{44} \end{bmatrix}$$

Thus, $a_{ij} = 0$ if $i \neq j$. If all the a_{ii} are equal, then the diagonal matrix is a *scalar matrix*. A diagonal matrix that has unit elements on the main diagonal is a *unit matrix* or *identity matrix:*

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \delta_{ij}$$

where, again, δ_{ij} is the Kronecker delta.

A square matrix whose elements are symmetrical about the main diagonal is a *symmetric matrix,* so that $a_{ij} = a_{ji}$. For example,

$$\mathbf{A} = \begin{bmatrix} 5 & 3 & 7 \\ 3 & 1 & 4 \\ 7 & 4 & -2 \end{bmatrix}$$

A square matrix is *antisymmetric* or *skew symmetric* if $a_{ij} = -a_{ji}$. For example,

$$\mathbf{A} = \begin{bmatrix} 0 & -9 & -3 \\ 9 & 0 & 1 \\ 3 & -1 & 0 \end{bmatrix}$$

where $a_{ij} = -a_{ji}$ implies that $a_{ii} = 0$.

A *null matrix,* denoted by $\varnothing$, is one whose elements are all zero.

Matrix Transpose

Interchanging the rows and columns of a matrix $\mathbf{A}$ produces its *transpose* $\mathbf{A}^T$. For example, if

$$\mathbf{A} = \begin{bmatrix} 3 & 1 & 2 \\ 0 & -1 & -2 \\ -3 & 0 & 4 \end{bmatrix}$$

then

$$\mathbf{A}^T = \begin{bmatrix} 3 & 0 & -3 \\ 1 & -1 & 0 \\ 2 & -2 & 4 \end{bmatrix}$$

The product of any matrix and its transpose is a symmetric matrix, that is, $(\mathbf{AA}^T)^T = \mathbf{AA}^T$.

All matrices have a transpose, not just square ones. Thus, if $\mathbf{B} = [x \quad y \quad z]$, then

$$\mathbf{B}^T = \begin{bmatrix} x \\ y \\ z \end{bmatrix}.$$

Matrix Addition

Adding two matrices **A** and **B** produces a third matrix **C** whose elements are equal to the sum of the corresponding elements of **A** and **B**. Thus, $\mathbf{A} + \mathbf{B} = \mathbf{C}$, or $a_{ij} + b_{ij} = c_{ij}$. **A** and **B** must be of the same order, and the resulting matrix is of the same order as the original matrices.

The difference of two matrices **A** and **B** is another matrix **D** whose elements are equal to the difference of corresponding elements of **A** and **B**. Thus $\mathbf{A} - \mathbf{B} = \mathbf{D}$, or $a_{ij} - b_{ij} = d_{ij}$. Again, **A** and **B** must be of the same order, and the resulting matrix is of the same order as the original matrices.

Scalar Multiplication

Multiplying a matrix **A** by a scalar constant k produces a new matrix **B** of the same order as **A**. Thus, $k\mathbf{A} = \mathbf{B}$, or, in terms of the matrix elements, $ka_{ij} = b_{ij}$.

Matrix Multiplication

Multiplying two matrices **A** and **B** produces a third matrix **C**, $\mathbf{AB} = \mathbf{C}$, if and only if the number of columns of **A** is equal to the number of rows of **B**, in which case **A** and **B** are *conformable*. If **A** is of order $m \times n$ and **B** is $n \times p$, then **C** is $m \times p$. In this example, **A** premultiplies **B** and **B** postmultiplies **A**. Given matrices **P** and **M**, with $\mathbf{R} = \mathbf{MP}$ (where **M** premultiplies **P**), and $\mathbf{S} = \mathbf{PM}$ (where **M** postmultiplies **P**), then in general $\mathbf{R} \neq \mathrm{S}$, except for the identity matrix, where

$$\mathbf{AI} = \mathbf{IA} \tag{B.1}$$

The product of two matrices defined in terms of their elements is

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \tag{B.2}$$

where n is equal to the number of columns of $\mathbf{A}$ and the number of rows of $\mathbf{B}$.

It is possible to construct matrices $\mathbf{A}$ and $\mathbf{B}$ such that $\mathbf{A} \neq 0$, $\mathbf{B} \neq 0$, but $\mathbf{AB} = 0$. A *zero* or *null matrix* is one for which all elements are zero.

Partitioned Matrices

It is often computationally convenient to partition a matrix into submatrices and to treat it as a matrix whose elements are themselves these submatrices. Thus

$$\mathbf{M} = \left[\begin{array}{c|c} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \hline \mathbf{M}_{21} & \mathbf{M}_{22} \end{array}\right]$$

where $\mathbf{M}_{11}$ and $\mathbf{M}_{12}$ must necessarily have the same number of rows (and similarly for $\mathbf{M}_{21}$ and $\mathbf{M}_{22}$), and $\mathbf{M}_{11}$ and $\mathbf{M}_{21}$ must necessarily have the same number of columns (and similarly for $\mathbf{M}_{12}$ and $\mathbf{M}_{22}$). For example, if

$$\mathbf{A} = \left[\begin{array}{ccc} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \hline \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} \end{array}\right], \qquad \mathbf{B} = \left[\begin{array}{ccc} \mathbf{B}_{11} & \mathbf{B}_{12} & \mathbf{B}_{13} \\ \hline \mathbf{B}_{21} & \mathbf{B}_{22} & \mathbf{B}_{23} \end{array}\right]$$

and $\mathbf{C} = \mathbf{A} + \mathbf{B}$, then

$$\mathbf{C} = \left[\begin{array}{ccc} \mathbf{A}_{11} + \mathbf{B}_{11} & \mathbf{A}_{12} + \mathbf{B}_{12} & \mathbf{A}_{13} + \mathbf{B}_{13} \\ \hline \mathbf{A}_{21} + \mathbf{B}_{21} & \mathbf{A}_{22} + \mathbf{B}_{22} & \mathbf{A}_{23} + \mathbf{B}_{23} \end{array}\right]$$

where the $\mathbf{A}_{ij}$ and $\mathbf{B}_{ij}$ are conformable.

Here is an example of the multiplication of partitioned matrices. The matrices must be conformable for multiplication before and after partitioning.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \\ b_{51} & b_{52} & b_{53} \end{bmatrix} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \\ \mathbf{B}_{31} & \mathbf{B}_{32} \end{bmatrix}$$

$$\mathbf{AB} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{B}_{11} + \mathbf{A}_{12}\mathbf{B}_{21} + \mathbf{A}_{13}\mathbf{B}_{31} & \mathbf{A}_{11}\mathbf{B}_{12} + \mathbf{A}_{12}\mathbf{B}_{22} + \mathbf{A}_{13}\mathbf{B}_{32} \\ \mathbf{A}_{21}\mathbf{B}_{11} + \mathbf{A}_{22}\mathbf{B}_{21} + \mathbf{A}_{23}\mathbf{B}_{31} & \mathbf{A}_{21}\mathbf{B}_{12} + \mathbf{A}_{22}\mathbf{B}_{22} + \mathbf{A}_{23}\mathbf{B}_{32} \end{bmatrix}$$

Determining the products $\mathbf{A}_{11}\mathbf{B}_{11}$, $\mathbf{A}_{12}\mathbf{B}_{21}$, . . . , and so forth, and performing the indicated sums completes this computation.

Matrix Inversion

The inverse of a square matrix $\mathbf{A}$ is $\mathbf{A}^{-1}$. A matrix and its inverse satisfy the conditions

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I} \tag{B.3}$$

The elements of $\mathbf{A}^{-1}$ are $\tilde{a}_{ij}$, where

$$\tilde{a}_{ij} = \frac{(-1)^{i+j}|\mathbf{A}'_{ji}|}{|\mathbf{A}|} \tag{B.4}$$

where $|\mathbf{A}'_{ji}|$ denotes the determinant of the $(n-1) \times (n-1)$ matrix derived from $\mathbf{A}$ by deleting row j and column i from $\mathbf{A}$ (note the subscript order on $\mathbf{A}'_{ji}$). If $\mathbf{A}^{-1}$ exists, then $|\mathbf{A}| \neq 0$.

Scalar and Vector Products

If row matrices $\mathbf{P}$ and $\mathbf{Q}$ represent the vectors $\mathbf{p}$ and $\mathbf{q}$ where $\mathbf{P} = [p_1 \quad p_2 \quad p_3]$, and $\mathbf{Q} = [q_1 \quad q_2 \quad q_3]$, then the scalar product of the vectors is

$$\mathbf{p} \cdot \mathbf{q} = \mathbf{P}\mathbf{Q}^T \tag{B.5}$$

where

$$\mathbf{P}\mathbf{Q}^T = [p_1\, p_2\, p_3]\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = p_1q_1 + p_2q_2 + p_3q_3 \tag{B.6}$$

Using the components of $\mathbf{p}$ to form the following antisymmetric matrix, then a matrix multiplication represents the vector product of $\mathbf{p}$ and $\mathbf{q}$:

$$\begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix}$$

and

$$\mathbf{p} \times \mathbf{q} = \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix}\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{B.7}$$

Eigenvalues and Eigenvectors

The expression $\mathbf{p}' = \mathbf{A}\mathbf{p}$ represents the general transformation of a point $\mathbf{p}$, where $\mathbf{p}'$ is the transformed point and $\mathbf{A}$ is the transformation matrix. Under certain conditions it is possible to find a scalar λ such that $\mathbf{p}' = \lambda\mathbf{p}$ or

$$\mathbf{A}\mathbf{p} = \lambda\mathbf{p} \tag{B.8}$$

Every vector for which this is true is an *eigenvector* of $\mathbf{A}$, and λ is the *eigenvalue* of $\mathbf{A}$ corresponding to $\mathbf{p}$. Matrix $\mathbf{A}$ sends an eigenvector into a collinear vector, and the corresponding eigenvalue is equal to the ratio of the magnitudes of the two collinear vectors.

Equation (B.8) is equivalent to

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{p} = 0 \tag{B.9}$$

This equation has nontrivial solutions if $\mathbf{p} \neq 0$ and

$$|\mathbf{A} - \lambda\mathbf{I}| = 0 \tag{B.10}$$

This is the *characteristic equation*, and its solutions are eigenvalues of $\mathbf{A}$.

Summary of Matrix Properties

1. $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$
2. $\mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C}$
3. $k(\mathbf{A} + \mathbf{B}) = k\mathbf{A} + k\mathbf{B}$
4. $(k + l)\mathbf{A} = k\mathbf{A} + l\mathbf{A}$
5. $k(l\mathbf{A}) = (kl)\mathbf{A} = l(k\mathbf{A})$
6. $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$
7. $\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$
8. $(\mathbf{A} + \mathbf{B})\mathbf{C} = \mathbf{AC} + \mathbf{BC}$
9. $\mathbf{A}(k\mathbf{B}) = k\mathbf{AB} = (k\mathbf{A})\mathbf{B}$
18. $(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$
11. $(k\mathbf{A})^T = k\mathbf{A}^T$
12. $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$
13. If $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$ and $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$, then $\mathbf{A}$ is nonsingular.
14. If $|\mathbf{A}| = 0$, then $\mathbf{A}$ is singular.
15. If $\mathbf{A}\mathbf{A}^T = \mathbf{I}$, then $\mathbf{A}$ is orthogonal.

Determinants

A *determinant* is a square array of elements that reduces to a single value by following a well-defined procedure. Determinants facilitate many vector and

matrix operations. However, a matrix must be square to have a determinant, and not every square matrix is associated with a determinant. If

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

then

$$|\mathbf{A}| = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

If

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

then

$$|\mathbf{A}| = a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

The *cofactor* of element a_{ij} of determinant $|\mathbf{A}|$ is

$$c_{ij} = (-1)^{i+j}|\mathbf{A}'_{ij}|$$

where $|\mathbf{A}'_{ij}|$ denotes the determinant of the $(n-1) \times (n-1)$ matrix derived from $\mathbf{A}$ by deleting row i and column j from $\mathbf{A}$.

Determinant Properties

1. The determinant of a square matrix is equal to the determinant of its transpose: $|\mathbf{A}| = |\mathbf{A}^T|$.
2. Interchanging any two rows (or columns) of $\mathbf{A}$ changes the sign of $|\mathbf{A}|$.
3. If we obtain $\mathbf{B}$ by multiplying one row (or column) of $\mathbf{A}$ by a constant c, then $|\mathbf{B}| = c|\mathbf{A}|$.
4. If two rows (or columns) of $\mathbf{A}$ are identical, then $|\mathbf{A}| = 0$.
5. If we derive $\mathbf{B}$ from $\mathbf{A}$ by adding a multiple of one row (or column) of $\mathbf{A}$ to another row (or column) of $\mathbf{A}$, then $|\mathbf{B}| = |\mathbf{A}|$.
6. If $\mathbf{A}$ and $\mathbf{B}$ are both $n \times n$ matrices, then the determinant of their product is $|\mathbf{AB}| = |\mathbf{A}||\mathbf{B}|$.
7. If every element of a row (or column) is zero, then the value of the determinant is zero.

This page intentionally left blank

APPENDIX C

TRANSFORMATIONS

Geometric transformations are an integral part of the modeling process. Once we define a curve, surface, or solid, we can move it, duplicate it, and reshape it to either become part of a more complex model construction or simply to meet more refined design requirements. A transformation allows us to change the position, orientation, or shape of a model in ways that would otherwise be tedious, if not impossible, to achieve.

Linear transformations are the focus of this chapter. With the exception of translations and projective transformations, they are given by linear homogeneous equations of the form

$$x' = a_{11}x + a_{12}y + a_{13}z$$

$$y' = a_{21}x + a_{22}y + a_{23}z$$

$$z' = a_{31}x + a_{32}y + a_{33}z$$

A translation spoils the homogeneous character of these equations by requiring the addition of the constant terms a_{14}, a_{24}, and a_{34}, respectively. The projectivities require a rational form, where each of the three expressions on the right is divided by a fourth linear expression $a_{41}x + a_{42}y + a_{43}z + a_{44}$ (with or without the constant terms). Introducing homogeneous coordinates restores the homogeneous character of these equations.

We can determine a hierarchy of linear transformations by the invariance of certain geometric properties. An *invariant property* is one that does not

change, one that is preserved when we subject the geometric model to a transformation. The rigid-body transformations, or *isometries*, consisting of translations and rotations, are the most restrictive of the so-called *affine transformations*. These are the linear transformations that preserve, at least, parallel lines. For the isometries, all metric properties are invariant (distance, angle, area, volume, and so on).

An *isotropic* or uniform *dilation (scaling) transformation*, another linear transformation, preserves angles but not the other metric properties. Nonuniform, or *anisotropic dilation*, as well as *shear transformations*, preserve parallel lines but not angles and distance. *Projective transformations*, the least restrictive linear transformations, preserve straight lines but not parallels. This brings us to *nonlinear topological transformations*, where bending, stretching, and twisting are allowed and only the topology of the model is preserved. We addressed these in part in Chapter 10.

The transformations we consider here are translation, rotation, reflection, symmetry, scaling and shear. We also discuss how to find the product of multiple sequential transformations. For a more complete discussion of these and many other geometric transformations, including symmetry and group theory, see Mortenson (1995).

Transformation of Hermite representations is somewhat different from that of the Bézier and B-Spline representations. The Hermite basis curves and surfaces are defined by a mix of boundary condition types: namely, endpoints and tangent vectors. The Bézier and B-Spline curves and surfaces are simply defined by a set of control points. We already know that Bézier and B-Spline forms are affinely invariant, while this is not true of the Hermite form. For example, given a Bézier curve and an arbitrary point $\mathbf{p}(u_i)$ on it, we can perform an affine transformation on the control points and then, from the results, compute a $\mathbf{p}'(u_i)$, or we can apply the same transformation to $\mathbf{p}(u_i)$, producing $\bar{\mathbf{p}}(u_i)$, and finding that $\mathbf{p}'(u_i) = \bar{\mathbf{p}}(u_i)$. Using the Bézier form as representative of the modeling elements defined by control points, we can compare it to the Hermite form, depending upon the particular transformation imposed.

We will interpret transformations in an *active sense*. That is, we act on and transform the model within a fixed coordinate system. Although we might think of transformations as accomplishing some sort of movement, they are really just mappings of functions. There is no path or transformation trajectory. Only when we use a transformation to sweep an element through its space to produce a higher-order element is the idea of a path necessary, and then it is obvious and we explicitly incorporate it into the model.

Each specie of transformation is presented first in terms of linear Cartesian equations. From these we develop their vector and matrix representations. We use homogeneous coordinates to compute the product of multiple

sequential transformations that include translations and projections. Recall that in homogeneous coordinates we represent a point **p** in n-dimensional space by a vector of $n + 1$ components. In ordinary Cartesian coordinates, we represent a point by the three-component vector $[x \quad y \quad z]^T$, and in homogeneous coordinates by the four-component vector $[x \quad y \quad z \quad h]^T$. The Cartesian and homogeneous coordinates are related by

$$x = \frac{hx}{h}$$

$$y = \frac{hy}{h}$$

$$z = \frac{hz}{h}$$

These equations, of course, should be familiar from the earlier discussion of rational curves and surfaces. To simplify things we use $[x \quad y \quad z \quad 1]^T$ to represent the point $[x \quad y \quad z]^T$.

C.1 TRANSLATION

The translation of a point is given by

$$\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \\ z' &= z + t_z \end{aligned} \tag{C.1}$$

where t_x, t_y, and t_z represent components of the translation. Translation preserves the size and shape of an object and is, therefore, one of the rigid-body transformations.

The rigid-body translation of a geometric object requires that every point on it moves equally a given distance in a given direction (see Figure C.1). This means that we may specify a translation by a vector **t** so that

$$\mathbf{p}' = \mathbf{p} + \mathrm{t} \tag{C.2}$$

The prime mark denotes a transformed vector or matrix. In purely matrix form we write this as

$$\mathbf{P}' = \mathbf{P} + \mathbf{T} \tag{C.3}$$

Expanding these matrices produces

Figure C.1 Translation of a curve.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

$$= \begin{bmatrix} x + t_x \\ y + t_y \\ z + t_z \end{bmatrix} \tag{C.4}$$

If we are careful with our notation and its interpretation, we can mix vectors and matrices as we already have for curves and surfaces. Adopting the convention of treating points as column matrices, we may write Equation (C.3) as

$$\mathbf{p}' = \mathbf{p} + \mathbf{T} \tag{C.5}$$

A succession of translations is another translation equivalent to its vector sum. The order of the succession is not important. Thus,

$$\mathbf{p}' = \mathbf{p} + \mathbf{t}_1 + \mathbf{t}_2 + \cdots + \mathbf{t}_n$$

is equivalent to

$$\mathbf{p}' = \mathbf{p} + \sum_{i=1}^{n} \mathbf{t}_i \tag{C.6}$$

Vector geometry and matrices certainly expedite translation transformations. For example, if the vector equation of a line is $\mathbf{p}(u) = \mathbf{p}_0 + u\mathbf{a}$, then its

translation by $\mathbf{t}$ is $\mathbf{p}'(u) = (\mathbf{p}_0 + \mathbf{t}) + u\mathbf{a}$, or $\mathbf{p}'(u) = \mathbf{p}(u) + \mathbf{t}$. Similarly, if a plane is $\mathbf{p}(u,w) = \mathbf{p}_0 + u\mathbf{a} + w\mathbf{b}$, then its translation is $\mathbf{p}'(u) = (\mathbf{p}_0 + \mathbf{t}) + u\mathbf{a} + w\mathbf{b}$, or $\mathbf{p}'(u,w) = \mathbf{p}(u,w) + \mathbf{t}$.

It is not necessary to translate every point on a curve or surface. It is sufficient to translate the control points, from which we can then generate a new curve or surface. To translate a cubic Hermite curve, we must transform its matrix of geometric coefficients. Thus,

$$\mathbf{B}' = \mathbf{B} + \mathbf{T} \tag{C.7}$$

where $\mathbf{T} = [\mathbf{t} \quad \mathbf{t} \quad 0 \quad 0]^T$.

For a bicubic Hermite patch we have, again, the general form $\mathbf{B}' = \mathbf{B} + \mathbf{T}$, but now

$$\mathbf{T} = \begin{bmatrix} \mathbf{t} & \mathbf{t} & 0 & 0 \\ \mathbf{t} & \mathbf{t} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{C.8}$$

The four corner points are translated, and the tangent and twist vectors are unaffected.

For a Bézier curve or surface we translate all the control points by $\mathbf{t}$. Thus, for curves we have

$$\begin{aligned} \mathbf{p}'(u) &= \sum_{i=0}^{n} (\mathbf{p}_i + \mathbf{t}) B_{in}(u) \\ &= \sum_{i=0}^{n} \mathbf{p}'_i B_{in}(u) \end{aligned} \tag{C.9}$$

and for surfaces

$$\begin{aligned} \mathbf{p}'(u,w) &= \sum_{i=0}^{m} \sum_{j=0}^{n} (\mathbf{p}_{ij} + \mathbf{t})\, B_{im}(u)\, B_{jn}(w) \\ &= \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{p}'_{ij} B_{im}(u)\, B_{jn}(w) \end{aligned} \tag{C.10}$$

We translate B-Spline curves and surfaces in the same way, by translating the control points.

C.2 ROTATION IN THE PLANE

The geometry describing the rotation of a point (or vector) $\mathbf{p}$ through an angle ϕ about the origin is shown in Figure C.2. Because $\mathbf{p}$ and $\mathbf{p}'$ lie on the circumference of a circle centered at the origin and $|\mathbf{p}| = |\mathbf{p}'|$ (distance is preserved in a rigid-body transformation), from elementary trigonometry we find that

$$\cos(\omega + \phi) = \cos\omega\cos\phi - \sin\omega\sin\phi$$

$$\sin(\omega + \phi) = \sin\omega\cos\phi + \cos\omega\sin\phi$$

Furthermore, $\cos\omega = x/|\mathbf{p}|$ and $\sin\omega = y/|\mathbf{p}|$, so that substitution yields

$$\cos(\omega + \phi) = \frac{x'}{|\mathbf{p}|} = \frac{x}{|\mathbf{p}|}\cos\phi - \frac{y}{|\mathbf{p}|}\sin\phi$$

$$\sin(\omega + \phi) = \frac{y'}{|\mathbf{p}|} = \frac{x}{|\mathbf{p}|}\sin\phi + \frac{y}{|\mathbf{p}|}\cos\phi$$

which simplifies to

$$\begin{aligned} x' &= x\cos\phi - y\sin\phi \\ y' &= x\sin\phi + y\cos\phi \end{aligned} \tag{C.11}$$

Figure C.2 Rotating a point or vector about the z axis.

In matrix form this equation becomes

$$\mathbf{p}' = \mathbf{R}\mathbf{p} \tag{C.12}$$

where $\mathbf{R}$ is the rotation transformation matrix, and

$$\mathbf{R} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \tag{C.13}$$

We will use the convention of premultiplying the point matrix by the transformation matrix. Notice that $|\mathbf{R}| = 1$ is a property of all isometries.

Successive Rotations about the Origin

The product of two successive rotations ϕ_1 and ϕ_2 of a point about the origin is given by

$$\mathbf{p}' = \mathbf{R}_{\phi_2}\mathbf{R}_{\phi_1}\mathbf{p} \tag{C.14}$$

We interpret this as a rotation by ϕ_1 followed by ϕ_2, in that order. The matrix product $\mathbf{R}_{\phi_2}\mathbf{R}_{\phi_1}$ yields

$$\mathbf{R}_{\phi_2}\mathbf{R}_{\phi_1} = \begin{bmatrix} \cos\phi_1\cos\phi_2 - \sin\phi_1\sin\phi_2 & -\sin\phi_1\cos\phi_2 - \cos\phi_1\sin\phi_2 \\ \cos\phi_1\sin\phi_2 + \sin\phi_1\cos\phi_2 & -\sin\phi_1\sin\phi_2 + \cos\phi_1\cos\phi_2 \end{bmatrix}$$

Using elementary trigonometric identities we find

$$\mathbf{R}_{\phi_2}\mathbf{R}_{\phi_1} = \begin{bmatrix} \cos(\phi_1+\phi_2) & -\sin(\phi_1+\phi_2) \\ \sin(\phi_1+\phi_2) & \cos(\phi_1+\phi_2) \end{bmatrix} \tag{C.15}$$

which tells us that for successive rotations in the plane about the origin we simply add the angles and use the resulting sum in the rotation matrix of Equation (C.13). We easily generalize this to the product of n successive rotations about the origin, or

$$\mathbf{R}_{(\phi_1+\phi_2+\cdots+\phi_n)} = \begin{bmatrix} \cos(\phi_1+\phi_2+\cdots+\phi_n) & -\sin(\phi_1+\phi_2+\cdots+\phi_n) \\ \sin(\phi_1+\phi_2+\cdots+\phi_n) & \cos(\phi_1+\phi_2+\cdots+\phi_n) \end{bmatrix} \tag{C.16}$$

The order in which we perform the product is not important, so that, for example, $\mathbf{R}_{\phi_2}\mathbf{R}_{\phi_1} = \mathbf{R}_{\phi_1}\mathbf{R}_{\phi_2} = \mathbf{R}_{\phi_1+\phi_2}$. We conclude that a sequence of rotations in the plane about the origin is commutative. This is not the case for rotations in space, where order is important and different orders produce different results.

Rotation about an Arbitrary Point

A rotation $\mathbf{R}$ about an arbitrary point $\mathbf{p}_c$ in the plane requires first a translation $\mathbf{t}_1 = -\mathbf{p}_c$ that brings the point $\mathbf{p}_c$ to the origin. Next, we execute the rotation and then reverse the translation $\mathbf{t}_2 = \mathbf{p}_c$, obtaining

$$\mathbf{p}' = \mathbf{R}(\mathbf{p} - \mathbf{p}_c) + \mathbf{p}_c \tag{C.17}$$

The combination of matrix multiplication and addition quickly becomes cumbersome as we add more rotations and translations. We will see that using homogeneous coordinates reduces this problem to a more workable sequence of matrix multiplications only.

Rotation of Curves and Surfaces

Given the vector equations of a line and plane, we rotate them about the origin as follows:

Line: $$\mathbf{p}'(u) = \mathbf{R}\mathbf{p}_0 + u\mathbf{R}\mathbf{a} \tag{C.18}$$

Plane: $$\mathbf{p}'(u,w) = \mathbf{R}\mathbf{p}_0 + u\mathbf{R}\mathbf{a} + w\mathbf{R}\mathbf{b} \tag{C.19}$$

Rotating a planar cubic Hermite curve requires some attention to how we set up its $\mathbf{B}$ matrix. Here is one way to do this. We let

$$\mathbf{B} = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_0' \\ \mathbf{p}_1' \end{bmatrix} = \begin{bmatrix} x_0 & y_0 \\ x_1 & y_1 \\ x_0' & y_0' \\ x_1' & y_1' \end{bmatrix} \tag{C.20}$$

Then a rotation transforms the curve by operating on its matrix of geometric coefficients as follows:

$$\mathbf{B}' = \mathbf{R}\mathbf{B} \tag{C.21}$$

or, more completely

$$\mathbf{p}'(u) = \mathbf{U}\mathbf{M}\mathbf{R}\mathbf{B} = \mathbf{U}\mathbf{M}\mathbf{B}' \tag{C.22}$$

For a Bézier curve, we rotate the set of control points, so that

$$\mathbf{p}_i' = \mathbf{R}\mathbf{p}_i \tag{C.23}$$

and

$$\mathbf{p}'(u) = \sum_{i=0}^{n} \mathbf{p}_i' B_{in}(u) \tag{C.24}$$

Coordinate System Rotation

To compute the coordinates of a point or the components of a vector in a coordinate system rotated by ϕ_c with respect to the initial system and with a common origin, we use Equation (C.13) with $\phi = -\phi_c$. This asserts that a coordinate system rotation produces results identical to that of an equivalent but oppositely directed rotation of a point or vector in the fixed initial coordinate system, and leads to

$$\mathbf{R}_{\phi_c} = \begin{bmatrix} \cos(-\phi_c) & -\sin(-\phi_c) \\ \sin(-\phi_c) & \cos(-\phi_c) \end{bmatrix} = \begin{bmatrix} \cos\phi_c & \sin\phi_c \\ -\sin\phi_c & \cos\phi_c \end{bmatrix} \tag{C.25}$$

so that

$$\mathbf{p}' = \mathbf{R}_{\phi_c}\mathbf{p} \tag{C.26}$$

We also see that

$$\mathbf{R}_{\phi_c} = \mathbf{R}_{\phi}^T = \mathbf{R}_{\phi}^{-1} \tag{C.27}$$

C.3 ROTATION IN SPACE ABOUT THE PRINCIPAL AXES

Rotations in space are more challenging and complex than rotations in the plane. We let ψ, θ, and ϕ denote rotations about the x, y, and z axes, respectively, of a fixed global frame. Remember, we are taking the active point of view, where points and other geometric objects are repositioned in a fixed coordinate frame. We begin with a rotation about the z axis, because it looks much like a rotation about the origin in the x, y plane, and simply expand Equation (C.11) to account for the z coordinate, so that

$$\begin{aligned} x' &= x\cos\phi - y\sin\phi \\ y' &= x\sin\phi + y\cos\phi \\ z' &= z \end{aligned} \tag{C.28}$$

and make appropriate changes to the rotation matrix, so that

$$\mathbf{R}_{\phi} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{C.29}$$

The rotation matrices for the x and y axes are simple extensions, namely

$$\mathbf{R}_\psi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \tag{C.30}$$

and

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 1 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{C.31}$$

Obviously, it is still the case that $|\mathbf{R}_\phi| = |\mathbf{R}_\psi| = |\mathbf{R}_\theta| = 1$.

Successive Rotations about the Principal Axes

We can use successive rotations about each of the three principal axes to place a geometric object into any orientation. However, it is important that we establish a convention for doing this, because it turns out that, for successive rotations in space, order is important. Here is the first convention we will use:

1. First, if $\phi \neq 0$, rotate by $\mathbf{R}_\phi$ about the z axis.
2. Next, if $\theta \neq 0$, rotate by $\mathbf{R}_\theta$ about the y axis.
3. Finally, if $\psi \neq 0$, rotate by $\mathbf{R}_\psi$ about the x axis.

Using this convention to rotate a point we have

$$\mathbf{p}' = \mathbf{R}_\psi \mathbf{R}_\theta \mathbf{R}_\phi \mathbf{p} \tag{C.32}$$

The expanded product of these three rotation matrices is

$$\mathbf{R}_{\psi\theta\phi} = \begin{bmatrix} \cos\theta\cos\phi & -\cos\theta\sin\phi & \sin\theta \\ \cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi & \cos\psi\cos\phi - \sin\psi\sin\theta\sin\phi & -\sin\psi\cos\theta \\ \sin\psi\sin\phi - \cos\psi\sin\theta\cos\phi & \sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \cos\psi\cos\theta \end{bmatrix} \tag{C.33}$$

A different order, say, $\mathbf{R}_\theta \mathbf{R}_\psi \mathbf{R}_\phi$, yields an entirely different product matrix.

We know that $|\mathbf{R}_\phi| = |\mathbf{R}_\psi| = |\mathbf{R}_\theta| = 1$, and we can use matrix algebra to show that $|\mathbf{R}_{\psi\theta\phi}| = 1$. This raises the following comment and caution: After a sequence of rotations it is unavoidable that the determinant of the resultant rotation matrix will no longer precisely equal one. That is, it will happen that $|\mathbf{R}| = 1 \pm \epsilon$. The deviation will depend on the accuracy of the algorithm computing the sine and cosine functions, computational precision, round-off errors, and, of course, the number of individual rotation matrices contribut-

ing to the final product. In the absence of corrective measures, this deviation will produce non-rigid-body transformations and incorrect values for equivalent rotations.

Rotation of the Coordinate System

There are situations in which we must determine the coordinates of a point or components of a vector in a coordinate system that is rotated with respect to the initial system, but with a common origin or pivot point. When rotating a point or vector, the principal axes are fixed, but under a coordinate system rotation a succession of original and transformed axes leads to their final position. Figure C.3 shows what happens to the coordinate axes when they are rotated by ϕ_c, θ_c, and ψ_c, in that order, and when none of them is zero. Rotating the coordinate system by ϕ_c has the same effect as rotating the point or vector by $\phi = -\phi_c$.

Rotating ϕ_c about the z axis displaces the x and y axes to the x' and y' axes, while the z' axis remains coincident with the initial z axis. The next rotation, Θ_c about the y' axis, displaces the x' and z' axes to x'' and z'' and leaves the y'' axis coincident with the y' axis. The final rotation, ψ_c about the x'' axis, displaces the y'' and z'' axes to y''' and z''', leaving the x''' axis fixed and coincident with the x'' axis. We denote the first rotation as $\mathbf{R}(\phi_c)$, the second as $\mathbf{R}(\theta_c)$, and the third as $\mathbf{R}(\psi_c)$. Letting $\mathbf{R}(\phi_c, \theta_c, \psi_c) = \mathbf{R}(\psi_c)\mathbf{R}(\theta_c)\mathbf{R}(\phi_c)$, with $\phi = -\phi_c$, $\theta = -\theta_c$, and $\psi = -\psi_c$, we find from Equation (C.33)

Figure C.3 Rotation of the coordinate system.

$$\mathbf{R}(\phi_c, \theta_c, \psi_c)$$

$$= \begin{bmatrix} \cos\theta_c \cos\phi_c & \cos\theta_c \sin\phi_c & -\sin\theta_c \\ -\cos\psi_c \sin\phi_c + \sin\psi_c \sin\theta_c \cos\phi_c & \cos\psi_c \cos\phi_c + \sin\psi_c \sin\theta_c \sin\phi_c & \sin\psi_c \cos\theta_c \\ \sin\psi_c \sin\phi_c + \cos\psi_c \sin\theta_c \cos\phi_c & -\sin\psi_c \cos\phi_c + \cos\psi_c \sin\theta_c \sin\phi_c & \cos\psi_c \cos\theta_c \end{bmatrix} \quad \text{(C.34)}$$

As we might expect, it turns out that $\mathbf{R}(\phi_c, \theta_c, \psi_c)$ is neither the transpose nor the inverse of $\mathbf{R}(\phi, \theta, \psi)$. Simple matrix algebra suffices to demonstrate this assertion. Thus,

$$\mathbf{R}(\phi_c, \theta_c, \psi_c) = \mathbf{R}_\psi^{-1}\, \mathbf{R}_\theta^{-1}\, \mathbf{R}_\phi^{-1} = \mathbf{R}_\psi^T\, \mathbf{R}_\theta^T\, \mathbf{R}_\phi^T = [\mathbf{R}_\phi\, \mathbf{R}_\theta\, \mathbf{R}_\psi]^T$$

and

$$\mathbf{R}^T(\phi, \theta, \psi) = [\mathbf{R}_\psi\, \mathbf{R}_\theta\, \mathbf{R}_\phi]^T$$

However, in general

$$[\mathbf{R}_\phi\, \mathbf{R}_\theta\, \mathbf{R}_\psi]^T \neq [\mathbf{R}_\psi\, \mathbf{R}_\theta\, \mathbf{R}_\phi]^T$$

Euler Angle Rotations

So far, we have rotated points and other geometric objects with respect to a fixed global frame, or we have rotated the global frame. Now we will define a local frame that is rigidly attached to an object. The local and global frames initially coincide. We perform subsequent rotations with respect to this local frame, so that when we rotate an object its frame rotates with it. The rotation angles are the *Euler angles*.

The rotation sequence in

$$\mathbf{p}' = \mathbf{R}(z'', \psi)\, \mathbf{R}(x', \theta)\, \mathbf{R}(z, \phi)\mathbf{p}$$

is the *x convention* for Euler angle rotations, where x', y', and z' indicate the local frame position with respect to the global frame after the first rotation, and x'', y'', and z'' indicate its position after the second rotation. (Notice the change in notation: The first argument element in parenthesis on the rotation matrix indicates the *axis* of rotation, and the second element indicates the *angle* of rotation.) The premultiplying convention for combining rotations applies. The first rotation $\mathbf{R}(z, \phi)$ is about the z axis in the global frame. It leaves the z axis of the local frame unchanged but rotates the local x and y axes with the object. The second rotation $\mathbf{R}(x', \theta)$ is about an x' axis that does not now coincide with any of the global axes. This means that we must first rotate x' into coincidence with one of the global axes, so we choose to rotate through $-\phi$ about the global z axis, or $\mathbf{R}(z, -\phi)$, bringing the x and x' axes into

coincidence. We now perform $\mathbf{R}(x, \theta)$, followed by $\mathbf{R}(z, \phi)$, which returns x' to its proper position. This tells us that there is a sequence of global frame rotations equivalent to $\mathbf{R}(x', \theta)$:

$$\mathbf{R}(x', \theta) = \mathbf{R}(z, \phi)\, \mathbf{R}(x, \theta)\, \mathbf{R}(z, -\phi)$$

The first two rotations of the x convention produce

$$\mathbf{R}(x', \theta)\, \mathbf{R}(z, \phi) = \mathbf{R}(z, \phi)\, \mathbf{R}(x, \theta)\, \mathbf{R}(z, -\phi)\, \mathbf{R}(z, \phi)$$

Using $\mathbf{R}(z, -\phi)\, \mathbf{R}(z, \phi) = \mathbf{I}$ to simplify the above equation produces

$$\mathbf{R}(x', \theta)\, \mathbf{R}(z, \phi) = \mathbf{R}(z, \phi)\, \mathbf{R}(x, \theta) \tag{C.35}$$

When we perform the third rotation $\mathbf{R}(z'', \psi)$ and again simplify using the procedure outlined above, we find that

$$\mathbf{R}(z'', \psi)\, \mathbf{R}(x', \theta)\, \mathbf{R}(z, \phi) = \mathbf{R}(z, \phi)\, \mathbf{R}(x, \theta)\, \mathbf{R}(z, \psi) \tag{C.36}$$

From this we conclude that a sequence of three Euler angle rotations in the local frame is equivalent to three rotations through these angles applied in reverse order and with respect to the global frame. The details of this method and its generalization are given by Lee and Koh (1995).

The x convention is frequently used in applied mechanics, kinematics, and molecular and solid-state physics. The y convention's rotation sequence (using the premultiplying convention) is $\mathbf{R}(z'', \psi)\, \mathbf{R}(y', \theta)\, \mathbf{R}(z, \phi)$. Many other conventions are possible. The general guideline is that no two successive rotations should be about the same axis.

Equivalent Rotations about the Principal Axes

After a sequence of many rotations producing a net rotation $\mathbf{R}$, we often must solve the inverse problem: Given any proper rotation matrix $\mathbf{R}$, find a set of rotation angles ϕ, θ, and ψ that generates an equivalent matrix. This problem is somewhat loosely stated, and we must choose conditions that limit the problem. We first select a specific sequence of angles and corresponding axes. Using the sequence that produced $\mathbf{R}_{\psi\theta\phi}$ in Equation (C.33), we equate the arbitrary proper rotation matrix $\mathbf{R}$ to $\mathbf{R}_{\psi\theta\phi}$ to obtain the following relationship:

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\phi & -\cos\theta\sin\phi & \sin\theta \\ \cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi & \cos\psi\cos\phi - \sin\psi\sin\theta\sin\phi & -\sin\psi\cos\theta \\ \sin\psi\sin\phi - \cos\psi\sin\theta\cos\phi & \sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \cos\psi\cos\theta \end{bmatrix} \tag{C.37}$$

Because matrix equality implies element-by-element equality, the following relationships also apply:

$$r_{11} = \cos\theta\cos\phi \tag{C.38}$$

$$r_{21} = \cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi$$

$$r_{31} = \sin\psi\sin\phi - \cos\psi\sin\theta\cos\phi$$

$$r_{12} = -\cos\theta\sin\phi$$

$$r_{22} = \cos\psi\cos\phi - \sin\psi\sin\theta\sin\phi$$

$$r_{32} = \sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi$$

$$r_{13} = \sin\theta \tag{C.39}$$

$$r_{23} = -\sin\psi\cos\theta$$

$$r_{33} = \cos\psi\cos\theta \tag{C.40}$$

It is tempting to solve these equations for ϕ, θ, and ψ as follows:

$$\theta = \sin^{-1} r_{13} \qquad \text{from Equation (C.39)} \tag{C.41}$$

$$\psi = \cos^{-1}\left(\frac{r_{33}}{\cos\theta}\right) \qquad \text{from Equations (C.40) and (C.41)} \tag{C.42}$$

$$\phi = \cos^{-1}\left(\frac{r_{11}}{\cos\theta}\right) \qquad \text{from Equations (C.38) and (C.41)} \tag{C.43}$$

However, Equations (C.41) to (C.43) have the potential for serious computational problems. For example, computing an angle using the arcsine function makes the distinction between the angles θ and $\pi - \theta$ ambiguous. Furthermore, if $\theta = 0$ or π, then $\sin\theta = 0$. Equations (C.42) and (C.43) are undefined when $\theta = \pm\pi/2$, because $\cos(\pm\pi/2) = 0$. A more conservative approach is to use tangent and arctangent functions. One way to do this is to premultiply both sides of Equation (C.37) by $\mathbf{R}_\psi^{-1}$ to produce

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21}\cos\psi + r_{31}\sin\psi & r_{22}\cos\psi + r_{32}\sin\psi & r_{23}\cos\psi + r_{33}\sin\psi\cos\psi \\ -r_{21}\sin\psi + r_{31}\cos\psi & -r_{22}\sin\psi + r_{32}\cos\psi & -r_{23}\sin\psi + r_{33}\cos\psi \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\phi & -\cos\theta\sin\phi & \sin\theta \\ \sin\phi & \cos\phi & 0 \\ -\sin\theta\cos\phi & \sin\theta\sin\phi & \cos\theta \end{bmatrix} \tag{C.44}$$

The element in row 2 and column 3 is zero. Equating this element in the matrix on the right to that on the left produces

$$r_{23}\cos\psi + r_{33}\sin\psi = 0$$

$$\tan\psi = -\frac{r_{23}}{r_{33}}$$

$$\psi = \tan^{-1}\left(-\frac{r_{23}}{r_{33}}\right)$$

These two solutions are 180° apart. It is also possible that $r_{23} = r_{33} = 0$; then the angle is undefined, and we arbitrarily set $\psi = 0$. Otherwise, once we compute a value for ψ, we know all the elements of the left side of Equation (C.44) and we can compute θ and ϕ. Thus

$$\sin\theta = r_{13}$$

$$\cos\theta = -r_{23}\sin\psi + r_{33}\cos\psi$$

so that

$$\theta = \tan^{-1}\left(\frac{r_{13}}{-r_{23}\sin\psi + r_{33}\cos\psi}\right)$$

With the sine and cosine defined, we easily compute a unique value for θ.

For ϕ we have

$$\sin\phi = r_{21}\cos\phi + r_{31}\sin\phi$$

$$\cos\phi = r_{23}\cos\psi + r_{32}\sin\psi$$

$$\phi = \tan^{-1}\left(\frac{r_{21}\cos\phi + r_{31}\sin\phi}{r_{23}\cos\psi + r_{32}\sin\psi}\right)$$

We conclude from all this that there is no unique solution to the "given **R**, find ϕ, θ, and ψ" problem, unless we make a series of clarifying assumptions.

C.4 ROTATION IN SPACE ABOUT AN ARBITRARY AXIS

Rotation about an arbitrary axis in space is the most general form of a rotation transformation. The development that follows assumes that the axis of rotation passes through the origin because, if it does not, it requires only a simple translation transformation to produce the condition. We define the axis by a unit vector $\hat{\mathbf{a}}$ and the angle of rotation by α. The right hand rule applies to the sign of α: If the thumb of the right hand points in the direction of $\hat{\mathbf{a}}$, then the fingers of the right hand curl around $\hat{\mathbf{a}}$ in the positive sense of the angle (see Figure C.4).

Figure C.4 Rotation about an arbitrary axis.

Here is the strategy we will use to develop the rotation matrix $\mathbf{R}$ defining this transformation: Rotate the vector $\hat{\mathbf{a}}$ into alignment with the z axis, perform the rotation through the angle α, and then return $\hat{\mathbf{a}}$ to its original orientation. This process is expressed algebraically by the following equations:

$$\mathbf{p}' = \mathbf{R}_{-\phi}\,\mathbf{R}_{-\psi}\,\mathbf{R}_{\alpha}\,\mathbf{R}_{\psi}\,\mathbf{R}_{\phi}\,\mathbf{p} \tag{C.45}$$

$$\mathbf{R} = \mathbf{R}_{-\phi}\,\mathbf{R}_{-\psi}\,\mathbf{R}_{\alpha}\,\mathbf{R}_{\psi}\,\mathbf{R}_{\phi} \tag{C.46}$$

Reading terms from right to left in Equation (C.46), we first execute a rotation $\mathbf{R}_{\phi}$ on an arbitrary point $\mathbf{p}$ about the z axis and through the angle ϕ. This places the axis $\hat{\mathbf{a}}$ in the y, z plane. Then $\mathbf{R}_{\psi}$ rotates the point about the x axis through the angle ψ, which aligns $\hat{\mathbf{a}}$ with the z. Next, $\mathbf{R}_{\alpha}$ rotates the point about the z axis, through the angle α. Finally, we reverse the process, with $\mathbf{R}_{-\psi}$ and $\mathbf{R}_{-\phi}$ rotating the point by $-\psi$ and $-\phi$, in that order, all of which

returns $\hat{\mathbf{a}}$ to its initial position. We see that the matrix product $\mathbf{R}_{-\phi}\,\mathbf{R}_{-\psi}\,\mathbf{S}_{\alpha}\,\mathbf{R}_{\psi}\,\mathbf{R}_{\phi}$ appropriately rotates an arbitrary point $\mathbf{p}$ to produce $\mathbf{p}'$ and leaves $\hat{\mathbf{a}}$ and, of course, any points on it invariant.

Performing the matrix multiplication indicated in Equation (C.46) to produce $\mathbf{R}$ is a horrible task. Fortunately, it must be done only once. To proceed, we use the following matrices:

$$\mathbf{R}_{\phi} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{R}_{-\phi} = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_{\psi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix}, \qquad \mathbf{R}_{-\psi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{bmatrix}$$

$$\mathbf{R}_{\alpha} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where

$$\cos\phi = \frac{\hat{a}_y}{\sqrt{\hat{a}_x^2 + \hat{a}_y^2}}$$

$$\sin\phi = \frac{\hat{a}_x}{\sqrt{\hat{a}_x^2 + \hat{a}_y^2}}$$

$$\cos\psi = \hat{a}_z$$

$$\sin\psi = \sqrt{\hat{a}_x^2 + \hat{a}_y^2}$$

and, because $\hat{\mathbf{a}}$ is a unit vector, we have $\hat{a}_x^2 + \hat{a}_y^2 + \hat{a}_z^2 = 1$.

First we find the elements of $\mathbf{R}$ in terms of sine and cosine functions of the angles ϕ, θ, and ψ. Then we substitute appropriately to replace the $\sin\phi$, $\cos\phi$, $\sin\psi$, and $\cos\psi$ terms with their equivalents using the unit vector components $\hat{a}_x$, $\hat{a}_y$, and $\hat{a}_z$. Finally, we obtain

$$\mathbf{R} = \begin{bmatrix} \cos\alpha + \hat{a}_x^2(1-\cos\alpha) & -\hat{a}_z\sin\alpha + \hat{a}_x\hat{a}_y(1-\cos\alpha) & \hat{a}\sin\alpha + \hat{a}_x\hat{a}_z(1-\cos\alpha) \\ \hat{a}_z\sin\alpha + \hat{a}_x\hat{a}_z(1-\cos\alpha) & \cos\alpha + \hat{a}_y^2(1-\cos\alpha) & -\hat{a}_x\sin\alpha + \hat{a}_y\hat{a}_z(1-\cos\alpha) \\ -\hat{a}_y\sin\alpha + \hat{a}_x\hat{a}_z(1-\cos\alpha) & \hat{a}_x\sin\alpha + \hat{a}_y\hat{a}_z(1-\cos\alpha) & \cos\alpha + \hat{a}_z^2(1-\cos\alpha) \end{bmatrix} \tag{C.47}$$

Compare the r_{ii} matrix elements, and also the r_{ij} and r_{ji} elements. These and other patterns or regularities in the distribution of terms often indicate the correctness of a complex expression such as this.

If the axis represented by $\hat{\mathbf{a}}$ coincides with any of the principal coordinate axes, then $\mathbf{R}$ simplifies accordingly. Thus, for example, if $\hat{\mathbf{a}}$ lies along the z axis, then $\hat{a}_x = \hat{a}_y = 0$ and $\hat{a}_z = 1$, so that

$$\mathbf{R} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

as expected.

Rotation and Quaternions

Quaternions, invented by W. R. Hamilton in 1843, prove to be useful for computing the product of sequential rotations about arbitrary axes and subsequently extracting the equivalent rotation angle and axis. They are four-component hypernumbers, part scalar (one component) and part vector (three components), with special operating rules and interpretations. We will define a quaternion A as the formal sum $a + \mathbf{r}$ of a real number (scalar) and a vector. The formal derivation of the following quaternion properties appears in many readily accessible texts.

The sum of two quaternions $A = a + \mathbf{r}$ and $B = b + \mathbf{s}$ is another quaternion such that

$$A + B = (a + b) + (\mathbf{r} + \mathbf{s})$$

and the product of two quaternions is

$$\begin{aligned} A + B &= (a + \mathbf{r})(b + \mathbf{s}) \\ &= (ab - \mathbf{r} \cdot \mathbf{s}) + (a\mathbf{s} + b\mathbf{r} + \mathbf{r} \times \mathbf{s}) \end{aligned}$$

The result in both cases consists of a scalar part and a vector part. The presence of the term $\mathbf{r} \times \mathbf{s}$ in the vector product indicates that the vector product is not commutative, that is, $AB \neq BA$.

We apply quaternions to the rotation transformation as follows: Let unit vector $\hat{\mathbf{a}}$ define the axis of rotation through the origin, and let α define the angle of rotation. We assert without proof that we can describe this rotation by the quaternion

$$R = \cos(\alpha/2) + \sin(\alpha/2)\hat{\mathbf{a}} \tag{C.48}$$

This is the *normalized form* of a quaternion. The *conjugate* of R is

$$R^* = \cos(\alpha/2) - \sin(\alpha/2)\hat{\mathbf{a}}$$

To rotate a point $\mathbf{p}$ through α about $\hat{\mathbf{a}}$, we perform the quaternion multiplication given by

$$P' = RPR^{-1}$$

where P is the quaternion representing $\mathbf{p}$ (the scalar part of P is zero and the vector part is $\mathbf{p}$). P' is the rotated quaternion from which we extract $\mathbf{p}'$. For a normalized quaternion, the inverse is equal to the conjugate; thus, $R^{-1} = R^* = \cos(\alpha/2) - \sin(\alpha/2)\hat{\mathbf{a}}$. The inverse is found by simply reversing the sign of α. It is easy to prove that the product of two normalized quaternions is another normalized quaternion, a necessary characteristic when dealing with rotations.

This process extends to and greatly simplifies executing a sequence of rotations about arbitrary axes through the origin. For example, if R and S describe two distinct rotations, then R rotates $\mathbf{p}$ to RPR^{-1} and S takes this result to $S(RPR^{-1})S^{-1}$. Because quaternion multiplication is associative, we have

$$S(RPR^{-1})S^{-1} = (SR)P(R^{-1}S^{-1})$$

and

$$R^{-1}S^{-1} = (SR)^{-1}$$

If we let $T = SR$, then we find that

$$(SR)P(SR)^{-1} = TPT^{-1}$$

Clearly, the quaternion T describes a rotation that is equivalent to the sequence R followed by S. Thus, any sequence of rotations about axes through the origin has an equivalent single rotation. It is easy to extract the components of this equivalent rotation from the resultant quaternion T. The vector part of T defines the axis, and the scalar part is the cosine of one-half the rotation angle.

C.5 REFLECTION

Although the size and shape of an object do not change under a reflection transformation, its handedness may change, as in a mirror reflection. For this reason it is not strictly a rigid-body transformation. It is not a topological transformation either, because the topology of an object is not preserved under a reflection; that is, it is not possible to create a mirror image of an object by

bending, twisting, and stretching without cutting or piercing. Yet combinations of two or more reflections can produce all the rigid-body transformations.

Reflection transformations play an important part in constructing and analyzing the symmetry of models, and reflection lines and planes are lines and planes of symmetry of a model and its reflected image. Of course, the subject of symmetry is much deeper than this statement implies. To completely explore symmetry and its relationship to geometric modeling requires the aid of group theory, which is beyond the scope of this text.

Figure C.5 illustrates some of the ways to reflect a curve in the x, y plane. The geometric coefficients of a cubic Hermite curve are shown for each of four positions. The procedure for constructing a reflected image of any object in space is simple. For example, to reflect the curve in the first quadrant through the $y = 0$ plane, we transform each point $\mathbf{p}$ on it to its symmetric image $\mathbf{p}'$ located an equal distance from the plane but on the other side of it. The distance between point $\mathbf{a}$ and point $\mathbf{p}$ is equal to the distance between $\mathbf{a}$ and $\mathbf{p}'$. Point $\mathbf{a}$ lies on the straight line joining $\mathbf{p}$ and $\mathbf{p}'$ at its intersection with the x axis. Furthermore, the vector $\mathbf{p}' - \mathbf{p}$ is normal to this axis.

This transformation merely changes the algebraic signs of the y components of the curve coefficients. In fact, when the plane of symmetry is any one of the three principal planes, the reflection transformation is accomplished by

Figure C.5 Reflections of a curve in the x, y plane.

changing the sign of the curve coefficients of the component corresponding to the plane of symmetry. Although the example in Figure C.5 is a plane curve, the same procedures apply to a curve that twists through space.

Reflection may also take place across any of the principal axes or through the origin. A procedure analogous to that for reflection through a plane of symmetry applies to reflection across a principal axis or through the origin. Figures C.6 and C.7 are more elaborate examples of the reflection of a curve.

The determinant of a reflection transformation $\mathbf{R}_f$ is equal to -1:

$$|\mathbf{R}_f| = -1 \tag{C.49}$$

This tells us that a reflection is an orientation (or handedness) reversing transformation (for example, a right hand becomes a left hand). We will also investigate pseudoreflections, transformations that look like a reflection but are not orientation-reversing.

Inversion in the Origin

We can consider inversions as a special form of reflection in that they take place through a point. In the x, y plane, inversion through the origin is given

Figure C.6 Reflecting a curve through the three principal planes.

Figure C.7 Reflecting a curve through the origin and a principal axis.

by

$$x' = -x$$
$$y' = -y \tag{C.50}$$

or, in matrix form

$$\mathbf{p}' = \mathbf{R}_f \mathbf{p} \tag{C.51}$$

where

$$\mathbf{R}_f = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

We notice that $|\mathbf{R}_f| = +1$, and this transformation preserves orientation.

Inversion in the origin in three-dimensional space is given by

$$x' = -x$$
$$y' = -y$$
$$z' = -z \tag{C.52}$$

and

$$\mathbf{R}_f = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{C.53}$$

so that in three dimensions

$$|\mathbf{R}_f| = -1 \tag{C.54}$$

A study of inversion in higher dimensions demonstrates that inversion in a point is a true reflection only in odd-dimension spaces. In even-dimension spaces, the transformation produces a *half-turn*, which is a rotation through 180°. Inversion in a point is an *isometry* because distances are preserved.

Reflections in the Plane

Reflection in the x axis is given by

$$\begin{aligned} x' &= x \\ y' &= -y \end{aligned} \quad \text{and} \quad \mathbf{R}_f = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{C.55}$$

Reflection in the y axis is given by

$$\begin{aligned} x' &= -x \\ y' &= y \end{aligned} \quad \text{and} \quad \mathbf{R}_f = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

Reflection in an arbitrary line m through the origin, at an angle ϕ_m with the x axis, requires that we first execute a rotation that makes the line coincident with the x axis, $\phi = -\phi_m$; then we execute a reflection about the x axis and, last, reverse the initial rotation. Thus

$$\begin{aligned} \mathbf{R}_{f^m} &= \begin{bmatrix} \cos\phi_m & -\sin\phi_m \\ \sin\phi_m & \cos\phi_m \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \cos\phi_m & \sin\phi_m \\ -\sin\phi_m & \cos\phi_m \end{bmatrix} \\ &= \begin{bmatrix} \cos 2\phi_m & \sin 2\phi_m \\ \sin 2\phi_m & -\cos 2\phi_m \end{bmatrix} \end{aligned} \tag{C.56}$$

Reflection in Space

Earlier we found that a combination of two or more reflections can produce all of the rigid-body transformations. It is easy to show that two reflections in parallel planes are equivalent to a translation. It is also true that two reflections in intersecting planes are equivalent to a rotation. We demonstrate both of these in a later section when we discuss multiple sequential transformations. For now we merely list some very simple reflections.

Reflection in the $x = 0$ plane is given by

$$\begin{aligned} x' &= -x \\ y' &= y \\ z' &= z \end{aligned} \quad \text{and} \quad \mathbf{R}_f = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{C.57}$$

Reflection in the $y = 0$ plane is given by

$$\begin{aligned} x' &= x \\ y' &= -y \\ z' &= z \end{aligned} \quad \text{and} \quad \mathbf{R}_f = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{C.58}$$

Reflection in the $z = 0$ plane is given by

$$\begin{aligned} x' &= x \\ y' &= y \\ z' &= -z \end{aligned} \quad \text{and} \quad \mathbf{R}_f = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{C.59}$$

A transformation given by

$$\begin{aligned} x' &= x \\ y' &= -y \\ z' &= -z \end{aligned} \quad \text{and} \quad \mathbf{R}_f = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{C.60}$$

is the same as a half-turn about the x axis. It is really a pseudoreflection, because here $|\mathbf{R}_f| = 1$. Similar half-turns may occur about the y and z axes.

C.6 DILATION AND SHEAR

Dilation (often called *scaling*) and *shear* are more general affine transformations, where at least parallel lines are preserved. Isotropic dilation, that is uniform expansion or contraction in all directions, preserves shape and angles but not distance. Shear preserves distance along parallel lines, but not shape and angle.

Isotropic Dilation

We define a *uniform dilation* as follows: If the distance between any pair of points is d, then after a uniform dilation the distance between them is kd. The equations describing a uniform dilation fixing the origin are

$$\begin{aligned} x' &= kx \\ y' &= ky \\ z' &= kz \end{aligned} \tag{C.61}$$

In matrix form, these equations become

$$\begin{aligned} \mathbf{p}' &= D\mathbf{p} \\ &= \begin{bmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & k \end{bmatrix} \mathbf{p} \end{aligned} \tag{C.62}$$

where k is the scale factor. If $k > 1$, then the dilation is an expansion. If $0 < k < 1$, then the dilation is a contraction. If $k = 1$, it is the identity transformation. If $k < 0$, the transformation is a reflection or inversion coupled with an expansion or contraction.

We execute a dilation of a Bézier or B-Spline curve or surface by simply applying the dilation transformation to the control points. We scale a cubic Hermite curve via its geometric coefficients. Thus, $\mathbf{B}' = k\mathbf{B} = [k\mathbf{p}_0 \quad k\mathbf{p}_1 \quad k\mathbf{p}_0'' \quad k\mathbf{p}_1'']^T$ (see Figure C.8). We see the effect of scaling on the tangent vectors in Figure C.9.

The inverse of any dilation is

$$\mathbf{D}^{-1} = \begin{bmatrix} \frac{1}{k} & 0 & 0 \\ 0 & \frac{1}{k} & 0 \\ 0 & 0 & \frac{1}{k} \end{bmatrix} \tag{C.63}$$

We will consider dilation fixing an arbitrary point in the next section.

Figure C.8 Scaling a curve with respect to the origin.

Figure C.9 The effect of scaling on tangent vectors.

The product of two dilations $\mathbf{D}_1$ and $\mathbf{D}_2$ fixing the origin is

$$\begin{aligned}\mathbf{p}' &= \mathbf{D}_2\mathbf{D}_1\mathbf{p} \\ &= \begin{bmatrix} k_1k_2 & 0 & 0 \\ 0 & k_1k_2 & 0 \\ 0 & 0 & k_1k_2 \end{bmatrix}\mathbf{p}\end{aligned} \tag{C.64}$$

By inspection, we conclude that $\mathbf{D}_2\mathbf{D}_1 = \mathbf{D}_1\mathbf{D}_2$ and that the order of the dilations is not important.

Anisotropic Dilation

Anisotropic dilation is a nonuniform expansion or contraction. In general, there may be three independent scale factors fixing an arbitrary point, referred to as the three principal directions in space. For the simplest case, we specify a different scale factor for each of these directions, fixing the origin, and thus producing

$$\begin{aligned} x' &= k_x x \\ y' &= k_y y \\ z' &= k_z z \end{aligned} \tag{C.65}$$

In matrix form, we write this as

$$\mathbf{p}' = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix} \mathbf{p} \tag{C.66}$$

Clearly, in this situation the scale factor must vary with direction. To demonstrate this, we look at the simpler case of anisotropic dilation in the plane, where $x' = k_x x$ and $y' = k_y y$ and where the dilation axes correspond to the x and y axes. We will let k_ϕ denote the ratio in the direction ϕ (before dilation) with respect to the x axis (see Figure C.10) and compare the distance between a pair of points $\mathbf{p}_1$ and $\mathbf{p}_2$ on a line through the origin before d and after d', an anisotropic dilation. We compute k_ϕ as d'/d. The distance between $\mathbf{p}_1$ and $\mathbf{p}_2$ before dilation is

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Following an anisotropic dilation with ratios k_x and k_y, the distance between $\mathbf{p}'_1$ and $\mathbf{p}'_2$ is

$$d' = \sqrt{(x'_2 - x'_1)^2 + (y'_2 - y'_1)^2}$$

Substituting for x' and y', we find

$$d' = \sqrt{k_x^2(x_2 - x_1)^2 + k_y^2(y_2 - y_1)^2} \tag{C.67}$$

and

$$k_\phi = \frac{d'}{d} = \sqrt{\frac{k_x^2(x_2 - x_1)^2 + k_y^2(y_2 - y_1)^2}{(x_2 - x_1)^2 + (y_2 - y_1)^2}}$$

Figure C.10 Anisotropic dilation.

This equation becomes more meaningful if we use $y = x \tan \phi$, so that

$$k_\phi = \sqrt{k_x^2 \cos^2 \phi + k_y^2 \sin^2 \phi} \tag{C.68}$$

Because in this simple case k_x and k_y are orthogonal, it is relatively easy to interpret the maximum and minimum dilation and the effect of direction. A graph of k_ϕ versus ϕ generates an ellipse with k_x and k_y the major and minor axes, depending on their relative magnitudes.

The product of two sets of anisotropic dilations along the principal axes and fixing the origin is given by

$$\mathbf{p}' = \begin{bmatrix} k_{x_1}k_{x_2} & 0 & 0 \\ 0 & k_{y_1}k_{y_2} & 0 \\ 0 & 0 & k_{z_1}k_{z_2} \end{bmatrix} \mathbf{p} \tag{C.69}$$

Shear in the Plane Fixing a Principal Axis or Plane

A *shear transformation* induces a deformation in a geometric model like that shown in Figure C.11. Here, lines parallel to the x axis slip in the x direction an amount proportional to their y coordinate. In three-dimensional space, parallel planes do the slipping. The equations describing this transformation in the plane fixing the x axis are

$$\begin{aligned} x' &= x + k_x y \\ y' &= y \end{aligned} \tag{C.70}$$

or

Figure C.11 Shear fixing the x axis.

$$\mathbf{p}' = \begin{bmatrix} 1 & k_y \\ 0 & 1 \end{bmatrix} \mathbf{p} \tag{C.71}$$

Similar equations define the shear transformation that fixes the y axis:

$$\begin{aligned} x' &= x \\ y' &= k_x x + y \end{aligned} \tag{C.72}$$

or

$$\mathbf{p}' = \begin{bmatrix} 1 & 0 \\ k_x & 1 \end{bmatrix} \mathbf{p} \tag{C.73}$$

A shear transformation in three-dimensional space fixing the $z = 0$ plane is given by

$$\begin{aligned} x' &= x + k_1 z \\ y' &= y + k_2 z \\ z' &= z \end{aligned} \qquad \text{or} \qquad \mathbf{p}' = \begin{bmatrix} 1 & 0 & k_1 \\ 0 & 1 & k_2 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p} \tag{C.74}$$

Similar equations fix the $x = 0$ and $y = 0$ planes.

C.7 MULTIPLE SEQUENTIAL TRANSFORMATIONS

Rotation about an arbitrary point in the plane or an arbitrary axis in space requires that we first translate the model so that the arbitrary point coincides with the origin or so that the arbitrary axis passes through the origin. We then perform the indicated rotations and translate the results so that the point or axis returns to its initial position. We handle reflections, dilations, and shears fixing arbitrary points, lines, or planes in a similar way: by executing appropriate translations and rotations before and after the primary transformation. Whenever translations are part of the sequence of transformations, matrix addition is required if we write the equations for rotation, reflection, dilation, and shear in their Cartesian (nonhomogeneous) form. This approach rapidly becomes a nested mess of matrix algebra.

It is not possible to express a general linear transformation in two or three dimensions as a square matrix of order two or three, respectively, if it must include translations or projections. An extra dimension is necessary in both cases. This requires the use of homogeneous coordinates, where $\mathbf{p} = [hx \quad hy \quad hz \quad h]^T$. If $h = 1$, then $\mathbf{p} = [x \quad y \quad z \quad 1]^T$ and $\mathbf{p}' = [x' \quad y' \quad z' \quad 1]^T$.

Translation

We may now write the Cartesian translation $\mathbf{p}' = \mathbf{p} + \mathbf{t}$ in matrix form as

$$\mathbf{p}' = \mathbf{T}\mathbf{p} \tag{C.75}$$

where $\mathbf{p}$ and $\mathbf{p}'$ are given in homogeneous coordinates and

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.76}$$

Because $h = t_{44} = 1$, when we perform the matrix multiplication of Equation (C.75), we obtain

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{bmatrix} \tag{C.77}$$

A sequence of translations is given by

$$\mathbf{T}_\Sigma = \mathbf{T}_1\,\mathbf{T}_2\,\mathbf{T}_3 \cdots \mathbf{T}_n \tag{C.78}$$

or

$$\mathbf{T}_\Sigma = \begin{bmatrix} 1 & 0 & 0 & t_{x_1} + t_{x_2} + \cdots + t_{x_n} \\ 0 & 1 & 0 & t_{y_1} + t_{y_2} + \cdots + t_{y_n} \\ 0 & 0 & 1 & t_{z_1} + t_{z_2} + \cdots + t_{z_n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.79}$$

Order is not important, so that, for example, $\mathbf{T}_2\,\mathbf{T}_1 = \mathbf{T}_1\,\mathbf{T}_2$.

Rotation in the Plane

Rotation in the plane about the origin using a homogeneous transformation matrix is expressed as

$$\mathbf{p}' = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p} \tag{C.80}$$

Rotation about an arbitrary point $\mathbf{p}_c$ requires that we first translate the model so that $\mathbf{p}_c$ coincides with the origin. Then we perform the rotation and translate the results so that $\mathbf{p}_c$ returns to its initial position. This sequence of transformations is represented by the matrix product

$$\mathbf{p}' = \begin{bmatrix} 1 & 0 & x_c \\ 0 & 1 & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_c \\ 0 & 1 & -y_c \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p} \tag{C.81}$$

Carrying out this matrix multiplication, we obtain

$$\mathbf{p}' = \begin{bmatrix} \cos\phi & -\sin\phi & -x_c\cos\phi + y_c\sin\phi + x_c \\ \sin\phi & \cos\phi & -x_c\sin\phi - y_c\cos\phi + y_c \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p} \tag{C.82}$$

In terms of Cartesian coordinates, this is

$$\begin{aligned} x' &= (x - x_c)\cos\phi - (y - y_c)\sin\phi + x_c \\ y' &= (y - y_c)\sin\phi + (y - y_c)\cos\phi + y_c \end{aligned} \tag{C.83}$$

Rotation in Space

Rotation about the z axis is expressed as

$$\mathbf{p}' = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{p} \tag{C.84}$$

Similar equations, easy to derive, apply to rotation about the x and y axes.

Again, rotation about an arbitrary axis not through the origin requires that we first translate the model so that some point on this axis coincides with the origin and, second, rotate the model so that the arbitrary axis aligns with one of the principal axes. Next, we execute the primary rotation and finish the sequence of transformations by reversing the secondary rotation and translation so that the arbitrary axis once again coincides pointwise with the initial position, carrying the model along with it, of course.

The rotation of a point about the z axis, followed by a translation expressed as the product of homogeneous matrices, is

$$\begin{aligned} \mathbf{p}' &= \mathbf{TRp} \\ &= \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \end{aligned} \tag{C.85}$$

$$= \begin{bmatrix} \cos\phi & -\sin\phi & 0 & t_x \\ \sin\phi & \cos\phi & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Reversing the order of these transformations, we obtain

$$\mathbf{p}' = \mathbf{RTp}$$

$$= \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{C.86}$$

$$= \begin{bmatrix} \cos\phi & -\sin\phi & 0 & t_x\cos\phi - t_y\sin\phi \\ \sin\phi & \cos\phi & 0 & t_x\sin\phi + t_y\cos\phi \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

From these results it is clear that order is important, because we see that $\mathbf{TRp} \neq \mathbf{RTp}$.

Glide Rotation

The *glide rotation* (otherwise known as the *twist* or *screw rotation*) consists of a translation and rotation about an axis parallel to the direction of translation, where the translation is proportional to the rotation angle. The following equation describes a glide rotation about the z axis:

$$\mathbf{p}' = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & k\phi \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{p} \tag{C.87}$$

Reflection in Space

The homogeneous transformation matrix for inversion in the origin is

$$\mathbf{R}_f = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.88}$$

Because $r_{f44} = +1$, the parity computation is unaffected, and $|\mathbf{R}_f| = -1$.

Inversion of a geometric model relative to an arbitrary point $\mathbf{p}_1$ requires first a translation that moves $\mathbf{p}_1$ to the origin, then the primary inversion, followed by a translation that moves $\mathbf{p}_1$ back to its original position. Thus,

$$\mathbf{p}' = \mathbf{T}_1\mathbf{R}_f\mathbf{T}_1^{-1}\,\mathbf{p} \tag{C.89}$$

where

$$\mathbf{T}_1\mathbf{R}_f\mathbf{T}_1^{-1} = \begin{bmatrix} 1 & 0 & 0 & x_1 \\ 0 & 1 & 0 & y_1 \\ 0 & 0 & 1 & z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -1 & 0 & 0 & 2x_1 \\ 0 & -1 & 0 & 2y_1 \\ 0 & 0 & -1 & 2z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.90}$$

Dilation in Space

The *anisotropic dilation* about an arbitrary point $\mathbf{p}_c$ of a set of points defining a geometric object is expressed concisely using homogeneous transformation matrices. Here we assume that the scale factors k_x, k_y, and k_z are relative to the principal axes. We have

$$\mathbf{p}' = \mathbf{T}_c\mathbf{D}\mathbf{T}_c^{-1}\mathbf{p} \tag{C.91}$$

where

$$\mathbf{T}_c\mathbf{D}\mathbf{T}_c^{-1} = \begin{bmatrix} 1 & 0 & 0 & x_c \\ 0 & 1 & 0 & y_c \\ 0 & 0 & 1 & z_c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_c \\ 0 & 1 & 0 & -y_c \\ 0 & 0 & 1 & -z_c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} k_x & 0 & 0 & (1-k_x)x_c \\ 0 & k_y & 0 & (1-k_y)y_c \\ 0 & 0 & k_z & (1-k_z)z_c \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.92}$$

This page intentionally left blank

BIBLIOGRAPHY

This bibliography provides points of entry into a much more extensive literature, too large for a comprehensive listing here. In some cases, only the most accessible works are cited, not necessarily the earliest, on a particular subject. Because a significant part of geometric modeling comes out of the research and development efforts of private industry (and this is still true today), access to these results is often difficult, in part due to proprietary considerations as well as varying standards of documentation and reporting.

Many new entries are included in this bibliography of the second edition, of course, and not all entries of the first edition are repeated here. The many reasons for this include limitations of space and changes of emphasis. Omission of important works is not intended but is almost predictably inevitable in a work of this size. I apologize for any omission and welcome comments on additions and corrections pending subsequent printings of this work.

Abelson, H., and A. Disessa. *Turtle Geometry: The Computer as a Medium for Exploring Mathematics.* Cambridge, Mass: MIT Press, 1981.

Abhyankar, S. "Algebraic Geometry for Scientists and Engineers." *Mathematical Surveys and Monographs* 35. American Mathematical Society, 1990.

Abhyankar, S. S., and C. L. Bajaj. "Automatic Parameterization of Rational Curves and Surfaces IV: Algebraic Space Curves." *ACM Transactions on Graphics* 8(4): 325 (1989).

Acton, F. S. *Numerical Methods that Work.* New York: Harper and Row, 1970.

Adams, J. A. "Geometric Concepts for Computer Graphics." Engineering and Weapons Report no. EW-72-4. Annapolis, Md.: U. S. Naval Academy, Sept. 1972.

Affentranger, F., and J. A. Wiseacker. "On the Convex Hull of Uniform Random Points in a Simple d-Polytope." *Discrete and Computational Geometry* 6(4): 291 (1991).

Agarwal, S. C., and W. N. Waggenspak, Jr. "Decomposition Method for Extracting Face Topologies from Wireframe Models." *Computer Aided Design* 24(3): 123 (1992).

Agin, G. J., and T. O. Binford. "Computer Descriptions of Curved Objects." *IEEE Transactions on Computers* C-25: 439 (1976).

Agoston, M. K. *Algebraic Topology.* New York: Marcel Dekker, 1976.

Ahlberg, J., E. Nilson, and J. Walsh. *The Theory of Splines and Their Applications.* New York: Academic Press, 1967.

Ahuja, D. V., and S. A. Coons. "Interactive Graphics in Data Processing: Geometry for Construction and Display." *IBM Systems Journal* 3(4): 188 (1968).

Aldefeld, B. "Variation of Geometries Based on a Geometric-Reasoning Method." *Computer Aided Design* 20(3): 117–126 (1988).

Alexandroff, P. *Elementary Concepts of Topology.* New York: Dover, 1961.

Alt, L. "Rational Linear Reparametrization of NURBS and the Blossoming Principle." *Computer Aided Geometric Design* 10(5): 465 (1993).

Ansaldi, S., L. De Floriani, and B. Facidieno. "Geometric Modeling of Solid Objects by Using a Face Adjacency Graph Representation." *Computer Graphics* 19(3): 131 (1985).

Aomura, S., and T. Uehara. "Self intersection of an Offset Surface." *Computer Aided Design* 22(7): 417 (1990).

Aono, M., D. E. Breen, and M. J. Wozny. "Fitting a Woven Cloth Model to a Curved Surface: Mapping Algorithms." *Computer Aided Design* 26(4): 278 (1994).

Apostol, T. M. *Calculus, Vol. 2: Calculus of Several Variables with Applications to Probability and Vector Analysis.* New York: Blaisdell, 1978.

Arbab, F. "Set Models and Boolean Operations for Solids and Assemblies." *IEEE Computer Graphics and Applications* 10(6): 76 (1990).

Armit, A. "TAG—A high performance interactive 3D graphics systems." *Computers in Industry* 3: 117–123 (1982).

Armit, A. P. "Curve and Surface Design Using Multipatch and Multiobject Design Systems." *Computer Aided Design* 25(4): 251 (1993).

Armstrong, C. G. "Modeling Requirements for Finite-Element Analysis." *Computer Aided Design* 26(7): 573 (1994).

Arner, P. R. "Another Look at Surface/Surface Intersection." Ph.D. thesis, University of Utah, 1987.

Aziz, N. M., R. Bata, and S. Bhat. "Bézier Surface/Surface Intersection." *IEEE Computer Graphics and Applications* 10(1): 50–58 (1990).

Baer, A., C. M. Eastman, and M. Henrion. "Geometric Modeling: A Survey." *Computer Aided Design* 11(5): 253–272 (1979).

Bajaj, C. L., J. E. Hoffman, and R. E. Lynch. "Tracking Surface Intersections." *Computer Aided Geometric Design* 5: 285–307 (1988).

Bajaj, C., and I. Ihm. "Hermite Interpolation of Rational Space Curves Using Real Algebraic Surfaces." *Proceedings of the 5th Annual ACM Symposium on Computational Geometry*. 1989.

———. "Algebraic Surface Design with Hermite Interpolation." *ACM Transactions on Graphics* 11(1): 61 (1992).

Ball, A. A. "A Simple Specification of the Parametric Cubic Segment." *Computer-Aided Design* 10(3): 181–82 (1978).

———. "Reparametrization and its Application in Computer-Aided Geometric Design." *International Journal for Numerical Methods in Engineering* 20: 197–216 (1984).

———. "An Investigation of Curvature Variations over Recursively Generated B-Spline Surfaces." *ACM Transactions on Graphics* 9(4): 424 (1990).

———. "CONSURF Part 1: Introduction of the Conic Lofting Tile." *Computer Aided Design* 25(8): 513 (1993).

Barbic, J., and M. Spegel. "Language for Description of Geometric Configurations." *Computer Graphics Forum* 9(1): 9 (1990).

Bardis, L., and N. M. Patrikalakis. "Blending Rational B-Spline Surfaces." *Proceedings of Eurographics 1989*. Amsterdam: North-Holland, 1989.

Bardis, L., and M. Vafiadou. "Ship-Hull Geometry Representation with B-Spline Surface Patches." *Computer Aided Design* 24(4): 217 (1992).

Barnhill, R. E. "Smooth Interpolation over Triangles." In *Computer Aided Geometric Design*, eds. R. E. Barnhill and R. F. Riesenfeld. New York: Academic Press, 1974.

———. "Blending-Function Interpolation: A Survey and Some New Results." Numerical Analysis Report No. 9. Department of Mathematics, University of Dundee, Dundee, Scotland, July 1975.

———. Coons Patches. *Computers in Industry* 3: 37–43 (1982).

———. "A Survey of the Representation and Design of Surfaces." *IEEE Computer Graphics and Applications* 3(7): 9–16 (1983).

———. "Surfaces in Computer Aided Geometric Design: A Survey with New Results." *Computer Aided Geometric Design* 2: 1–17 (1985).

Barnhill, R. E., G. Birkhoff, and W. J. Gordon. "Smooth Interpolation in Triangles." *Journal of Approximation Theory* 8: 114–28 (1973).

Barnhill, R. E., and J. Gregory. "Blending Function Interpolation to Boundary Data on Triangles." *Mathematics of Computation* 29(131): 726–735 (1975).

Barnhill, R. E., G. Farin, M. Jordan, and B. R. Piper. "Surface/Surface Intersection." *Computer Aided Geometric Design* 4: 3–16 (1987).

Barnhill, R. E., and R. F. Riesenfeld, eds. *Computer-Aided Geometric Design.* New York: Academic Press, 1974.

Barr, A. H. "Superquadrics and Angle-Preserving Transformations." *IEEE Computer Graphics and Applications* 1(1) (1981).

———. "Global and Local Deformations of Solid Primitives." *Computer Graphics* 18(3): 21 (1984).

Barry, P. J., J. C. Beatty, and R. N. Goldman. "Unimodal properties of B-Splines and Bernstein-Basis Functions." *Computer Aided Design* 24(12): 627 (1992).

Barry, P. J., and R. N. Goldman. "A Recursive Proof of a B-Spline Identity for Degree Elevation." *Computer Aided Geometric Design* 5(2): 173–175 (1988).

Barsky, B. A. "The Beta-Spline: A Local Representation Based on Shape Parameters and Fundamental Geometric Measures." Ph.D. dissertation, Department of Computer Science, University of Utah, 1981.

———. "Local Control of Bias and Tension in Beta-Splines." *ACM Transactions in Graphics* 8(2): 109–34 (1983).

———. "A Study of the Parametric Uniform B-Spline Curve and Surface Representation." Technical Report CSD 83/118. Computer Science Division, University of California, Berkeley, May 1983.

———. "A Description and Evaluation of Various 3-D Models." *IEEE Computer Graphics and Applications* 4(1): 38–52 (1984).

———. *Computer Graphics and Geometric Modeling Using Beta-Splines.* New York: Springer-Verlag, 1988.

———. "Rational Beta-Splines for Representing Curves and Surfaces." *IEEE Computer Graphics and Applications* 13(6): 24 (1993).

Barsky, B. A., R. H. Bartels, and J. C. Beatty. "An Introduction to the Use of Splines in Computer Graphics." Technical Report CS-83-9. Department of Computer Science, University of Waterloo, Waterloo, Ontario, 1983.

Barsky, B. A., and J. C. Beatty. "Varying the Betas in Beta-Splines. Technical Report CSD 82/112. Computer Science Division, University of California, Berkeley, Dec. 1982.

Barsky, B. A., and T. D. DeRose. "The Beta 2-Spline: A Special Case of the Beta Spline Curve and Surface Representation." *IEEE Computer Graphics and Applications* 7(3): 15 (1985).

———. "Geometric Continuity of Parametric Curves: Three Equivalent Characterizations." Part 1. *IEEE Computer Graphics and Applications* 9(6): 60–67 (1989).

———. "Geometric Continuity of Parametric Curves: Construction of Geometrically Continuous Splines." Part 2. *IEEE Computer Graphics and Applications* 10(1): 60–68, (1990).

Bartels, R. H., and J. C. Beatty. "A Technique for the Direct Manipulation of Spline Curves." In *Graphics Interface 89*. Palo Alto, Calif.: Morgan Kaufmann, 1989.

Bartels, R. H., J. C. Beatty, and B. A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Palo Alto, Calif.: Morgan Kaufmann, 1987.

Bartels, R. H., and R. T. Hardock. "Curve-to-Curve Associations in Spline-Based Inbetweening and Sweeping." *Computer Graphics* 23(3): 167 (1989).

Beacon, G. R., J. R. Dodsworth, and S. E. Howe. "Boundary Evaluation Using Inner and Outer Sets: The ISOS Method." *IEEE Computer Graphics and Applications* 9(2): 39 (1989).

Beatty, P. L., P. A. Fitzhorn, and G. J. Herron. "Extensions in Variational Geometry that Generate and Modify Object Edges Composed of Rational Bézier Curves." *Computer Aided Design* 26(2): 85 (1994).

Beck, J., R. Farouki, and J. Hinds. "Surface Analysis Methods." *IEEE Computer Graphics and Applications* 6(12): 18–36 (1986).

Bedi, S. "Surface Design Using Functional Blending." *Computer Aided Design* 24(9): 505 (1992).

Bercovier, M., and E. Shilat. "Enhancement of Gordon-Coons Interpolations by 'bubble functions.' " *Computer Aided Geometric Design* 10(3–4): 253 (1993).

Berger, D. N., and M. Kallay. "Computing the Unit Normal on a Degenerate Edge." *Computer Aided Design* 24(7): 395 (1992).

Bez, H. "On Invariant Curve Forms." *Computer Aided Geometric Design* 3(3): 193–204 (1986).

Bézier, P. "Définition numérique des courbes et surfaces I." *Automatisme* 11: 625–632 (1966).

———. "Définition numérique des courbes et surfaces II." *Automatisme* 12: 17–21 (1967).

———. "Procédé de définition numérique des courbes et surfaces non mathématique." *Automatisme* 13(5) (1968).

———. *Numerical Control: Mathematics and Applications*, trans. A. R. Forrest and A. F. Pankhurst. London: Wiley, 1972.

———. "Mathematical and Practical Possibilities of UNISURF." In *Computer Aided Geometric Design*, eds. R. E. Barnhill and R. F. Riesenfeld. New York: Academic Press, 1974.

———. "Essay de définition numérique des courbes et des surfaces expérimentales." Ph.D. thesis, University of Paris IV, 1977.

———. "General Distortion of an Ensemble of Biparametric Patches." *Computer Aided Design* 19(2): 116–120 (1978).

———. *The Mathematical Basis of the UNISURF CAD System*. London: Butterworths, 1986.

———. "First Steps of CAD." *Computer Aided Design* 21(5): 259 (1989).

———. "Style, Mathematics and NC." *Computer Aided Design* 22(9): 524 (1990).

Binford, T. O., and G. J. Agin. "Computer Description of Curved Objects." *Proceedings of the Third International Conference on Artificial Intelligence*, 1973.

Bishop, R. L. "There Is More Than One Way to Frame a Curve." *American Mathematical Monthly* 82: 246 (1975).

Blackmore, D., M., C. Leu, and F. Shih. "Analysis and Modeling of Deformed Swept Volumes." *Computer Aided Design* 26(4): 315 (1994).

Bloomenthal, J. "Polygonization of Implicit Surfaces." *Computer Aided Geometric Design* 5: 341–355 (1988).

Bloomenthal, J., and K. Shoemake. "Convolution Surfaces." *Computer Graphics* 25(4): 251 (1991).

Bloor, M. I. G., and M. J. Wilson. "Generating Blend Surfaces Using Partial Differential Equations." *Computer Aided Design* 21(3): 165 (1989).

Bodduluri, R. M. C., and B. Ravani. "Design of Developable Surfaces Using Duality between Plane and Point Geometries." *Computer Aided Design* 25(10): 621 (1993).

Boehm, W. "Cubic B-Spline Curves and Surfaces in Computer Aided Geometric Design." *Computing* 19: 29–34 (1977).

———. "Inserting New Knots into B-Spline Curves." *Computer Aided Design* 12(4): 199–201 (1980).

———. "Generating the Bézier Points of B-Splines." *Computer Aided Design* 13(6) (1981).

———. "On Cubics: A Survey." *Computer Graphics and Image Processing* 19: 201–226 (1982).

———. "Curvature Continuous Curves and Surfaces." *Computer Aided Geometric Design* 2(2): 313–323 (1985).

———. "Rational Geometric Splines." *Computer Aided Geometric Design* 4: 67–77 (1987).

———. "On the Definition of Geometric Continuity." Letter to the editor. *Computer Aided Design* 20(7): 370–372 (1988).

Boehm, W., G. Farin, and J. Kahmann. "A Survey of Curve and Surface Methods in CAGD." *Computer Aided Geometric Design* 1(1): 1–60 (1984).

Bonner, D. L., M. J. Jakiela, and M. Watanabe. "PSEUDOEDGE: Nonintersected Parametric Quilt Modeling of Multiply Connected Objects." *Computer Aided Design* 25(7): 438 (1993).

Borrel, P., and D. Bechmann. "Deformation of *n*-Dimensional Objects." *Proceedings of the Symposium on Solid Modeling Foundations and CAD/CAM Applications.* Washington, D.C.: ACM Press, 1991.

Borrel, P., and A. Rappoport. "Simple Constrained Deformations for Geometric Modeling and Interactive Design." *ACM Transactions on Graphics* 13(2): 137 (1994).

Boyse, J. W. "Data Structure for a Solid Modeler." Research publication GMR-2933. Computer Science Department, General Motors Research Laboratories. Warren, Mich., 1979.

———. "Interference Detection among Solids and Surfaces." *Communications of the ACM* 21: 3–9 (1979).

Boyse, J. W., and J. E. Gilchrist. "GMSolid: Interactive Modeling for Design and Analysis of Solids." *IEEE Computer Graphics and Applications* 2(2): 27–40 (1982).

Braid, I. C. "Designing with Volumes." CAD Group report. Cambridge University, 1973.

———. "The Synthesis of Solids Bounded by Many Faces." *Communications of the ACM* 18(4): 209–16 (1975).

———. "New Directions in Geometric Modeling." *Proceedings of the Workshop on Geometric Modeling, CAM-I.* 1978.

Braid, I. C., R. C. Hillyard, and I. A. Stroud. "Stepwise Construction of Polyhedra in Geometric Modeling." CAD Group report no. 100. Cambridge University, 1978.

Brewer III, J. A., D. Vicknair, and M. Courter. "Converting Wireframes to Solids." *Computer Aided Engineering* 16(3): 51 (1989).

Brisson, E. "Representing Geometric Structures in *d* Dimensions: Topology and Order." *Discrete and Computational Geometry* 9(4): 387 (1993).

Brown, C. M. "Some Mathematical and Representational Aspects of Solid Modeling." *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-3(4): 444–53 (1981).

Brown, C. M. "PADL-2: A technical summary." *IEEE CG&A* 2(2): 69–84 (1982).

Brown, C. M., A. A. G. Requicha, and H. B. Voelker. "Geometric-Modeling Systems for Mechanical Design and Manufacturing." *Proceedings of the 1978 Annual Conference of ACM.* Washington, D.C.: ACM Press, 1978.

Brown, C. M., and H. B. Voelker. "The PADL-2 Project." *Proceedings of the Seventh NSF Conference on Production Research Technology.* Ithaca, N.Y.: National Science Foundation, 1979.

Brown, K. Q. "Comments on Algorithms for Reporting and Counting Geometric Intersections." *IEEE Transactions on Computing* C-30(2): 147–48 (1981).

Brunet, P., and I. Navazo. "Solid Representation and Operation Using Extended Octrees." *ACM Transactions on Graphics* 9(2): 170 (1990).

Bu-Qing, S., and L. Ding-Yuan. *Computational Geometry.* New York: Academic Press, 1989.

Burger, H., and R. Schaback. "A Parallel Multistage Method of Surface-Surface Intersection." *Computer Aided Geometric Design* 10(3–4): 277 (1993).

Butterfield, K. R. "The Computation of All the Derivatives of a B-Spline Basis." Oxford, England: British Leyland, May 1974.

Cameron, S., and C. K. Yap. "Refinement Methods for Geometric Bounds in Constructive Solid Geometry." *ACM Transactions on Graphics* 11(1): 12 (1992).

Carlbom, I., I. Chakravarty, and D. Vanderschel. "A Hierarchical Data Structure for Representing the Spatial Decomposition of 3D Objects." *IEEE Computer Graphics and Applications* 5(4): 24–31 (1985).

Casale, M. S. "Free-Form Solid Modeling with Trimmed Surface Patches." *IEEE Computer Graphics and Applications* 7(1): 33–43 (1987).

Casale, M. S., and J. E. Bobrow. "A Set Operation Algorithm for Sculptured Solids Modeled with Trimmed Patches." *Computer Aided Geometric Design* 6(3): 235–248 (1989).

Casale, M. S., J. E. Bobrow, and R. Underwood. "Trimmed-Patch Boundary Elements: Bridging the Gap between Solid Modeling and Engineering Analysis." *Computer Aided Design* 24(4): 193 (1992).

Casale, M. S., and E. L. Stanton. "An Overview of Analytic Solid Modeling." *IEEE Computer Graphics and Applications* 5: 45–56 (1992).

Catmull, E. E. "Computer Display of Curved Surfaces." *Proceedings of the IEEE Conference on Computer Graphics, Pattern Recognition, and Data Structure.* 1975.

Catmull, E. E. and J. Clark. "Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes." *Computer Aided Design* 10(6) (1978).

Catmull, E. E., and R. Rom. "A Class of Local Interpolating Spline." In *Computer-Aided Geometric Design*, eds. R. E. Barnhill and R. F. Riesenfeld. New York: Academic Press, 1974.

Celniker, G., and W. Welch. "Linear Constraints for Deformable B-Spline surfaces." In *Computer Graphics Special Issue: 1992 Symposium on Interactive 3D Graphics* 165–170 (1992).

Chang, G. "Matrix Formulation of Bézier Technique." *Computer Aided Design* 14(6): 345 (1982).

Chang, S.-L., M. Shantz, and R. Rochetti. "Rendering Cubic Curves and Surfaces with Integer Adaptive Forward Differencing." Proceedings of SIGGRAPH '89. *Computer Graphics* 23(3): 157–166 (1989).

Charrot, P. and J. A. Gregory. "A Pentagonal Surface Patch for Computer Aided Geometric Design." *Computer Aided Geometric Design* 1(1): 87–94 (1984).

Chazelle, B., and L. Guibas. "Visibility and Intersection Problems in Plane Geometry." *Computer Graphics Forum* 4(6): 51 (1989).

Chen, B. "On the Euler Characteristic of Finite Unions of Convex Sets." *Discrete and Computational Geometry* 10(1): 79 (1993).

Cheng, F. "Estimating Subdivision Depths for Rational Curves and Surfaces." *ACM Transactions on Graphics* 11(2): 140 (1992).

Chirehdast, M., and P. Y. Papalambros. "Conversion of Spatial-Enumeration Scheme into Constructive Solid Geometry." *Computer Aided Design* 26(4): 302 (1994).

Chiyokura, H. "An Extended Rounding Operation for Modeling Solids with Free-Form Surfaces." *IEEE Computer Graphics and Applications* 7(12): 27–36 (1987).

———. *Solid Modeling with DesignBase,* Reading: Addison Wesley 1988.

Chiyokura, H., and F. Kimura. "Design of Solids with Free-Form Surfaces." *Computer Graphics* 17(3): 289–298 (1983).

Choi, B. K., and S. Y. Ju. "Constant Radius Blending in Surface Modeling." *Computer Aided Design* 21(4): 213 (1989).

Choi, B. K., and C. S. Lee. "Sweep Surfaces Modeling via Coordinate Transformations and Blending." *Computer Aided Design* 22(2): 87 (1990).

Choi, B. K., W. S. Yoo, and C. S. Lee. "Matrix Representation for NURB Curves and Surfaces." *Computer Aided Design* 22(4): 235 (1990).

Chou, J. J., and M. W. Blake. "Planar Cubics through a Point in a Direction." *Computer Aided Design* 25(6): 348 (1993).

Clark, J. H. "Designing Surfaces in 3D." *Communications of the ACM* 19(8): 454 (1976).

Cohen, E. "Algorithms for Degree-Raising of Splines." *ACM Transactions on Graphics* 4(3): 171 (1985).

Cohen, E., T. Lyche, and R. Riesenfeld. "Discrete B-Splines and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics." *Computer Graphics and Image Processing* 14(2): 87–111 (1980).

Cohen, E., and R. F. Riesenfeld. "General Matrix Representations for Bézier and B-Spline Curves." *Computer Industry* 3: 9–15 (1982).

Coons, S. A. "An Outline of the Requirements for a Computer-Aided Design System." MIT Memo ESL-TM-169. Cambridge, Mass.: Massachusetts Institute of Technology, Mar. 1963.

———. "Surfaces for Computer-Aided Design of Space Figures." MIT Memo ESL-9442-M-139. Cambridge, Mass.: Massachusetts Institute of Technology, July 1965.

———. "Surfaces for Computer-Aided Design of Space Forms." MIT Memo MAC-TR-41. Cambridge, Mass.: Massachusetts Institute of Technology, June 1967.

———. "Rational Bicubic Surface Patches." Project MAC technical report. Cambridge, Mass.: Massachusetts Institute of Technology, 1968.

———. "Surface Patches and B-Spline Curves." *Computer Aided Geometric Design*, eds. R. E. Barnhill and R. F. Riesenfeld. New York: Academic Press, 1974.

———. *Méthode Matricielle*. Trans. from English by P. Bézier and M. Moronval. Paris: Hermes, 1987.

Coquillart, S. "Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling." Proceedings of the SIGGRAPH 90 Conference. *Computer Graphics* 24(4): 187–196 (1990).

Courter, S. M., and J. A. Brewer. "Automated Conversion of Curvilinear Wire-Frame Models to Surface Boundary Models: A Topological Approach." *Computer Graphics* 20(4): 171 (1986).

Cox, M. G. "The Numerical Evaluation of B-Splines." *Journal of the Institute of Mathematics and Applications* 10: 134–149 (1972).

Coxeter, H. *Introduction to Geometry.* New York: Wiley, 1961.

Crocker, G. A., and W. F. Reinke. "Boundary Evaluation of Non-Convex Primitives to Produce Parametric Trimmed Surfaces." *Computer Graphics* 21(4): 129 (1987).

de Boor, C. "On Calculating with B-Splines." *Journal of Approximation Theory* 6: 50–62 (1972).

———. Good Approximation by Splines with Variable Knots." *ISNM* 21: 57–72 (1973).

———. *A Practical Guide to Splines.* New York: Springer-Verlag, 1978.

de Boor, C., K. Hollig, and M. Sabin. "High Accuracy Geometric Hermite Interpolation." *Computer Aided Geometric Design* 4(4): 269–278 (1988).

de Casteljau, P. "Outillages méthods calcul." Technical Report. Paris: Citroen, 1959.

———. "Courbes et surfaces a poles." Technical Report. Paris: Citroen, 1963.

———. *Shape, Mathematics and CAD.* London: Kegan Page, 1986.

de Floriani, L., and B. Falcipieno. "A Hierarchical Boundary Model for Solid Object Representation." *ACM Transactions on Graphics* 7(1): 42 (1988).

Degen, W. L. F. "High Accurate Rational Approximation of Parametric Curves." *Computer Aided Geometric Design* 10(3–4): 293 (1993).

de Montaudouin, Y. "Cross Product Cones of Revolution." *Computer Aided Design* 21(6): 404 (1989).

DeRose, A. "Geometric Continuity: A Parametrization Independent Measure of Continuity for Computer Aided Geometric Design." Ph.D. thesis, University of California, Berkeley, 1985. Also published as Technical Report UCB/CSD 86/255.

DeRose, T. "Composing Bézier Simplices." *ACM Transactions on Graphics* 7(3): 198–221 (1988).

DeRose, T. D., and B. A. Barsky. "Geometric Continuity, Shape Parameters, and Geometric Constructions for Catmul-Rom Splines." *ACM Transactions on Graphics* 7(1): 1–41 (1988).

Dimsdale, B. "Bicubic Patch Bounds." *Computing and Mathematics with Applications* 3(2): 95–104 (1977).

Dokken, T., V. Shytt and A. M. Ytrehus. "The role of NURBS in Geometric Modeling and CAD/CAM." In *Advanced Geometric Modeling for Engineering Applications.* Netherlands: Elsevier, 1990.

Du, W. H., and F. J. M. Schmitt. "On the G^1 Continuity of Piecewise Bézier Surfaces: A Review with New Results." *Computer Aided Design* 22(9): 556–573 (1990).

Eastman, C. M. "Vector versus Raster: A Functional Comparison of Drawing Technologies." *IEEE Computer Graphics and Applications* 10(5): 68–80 (1990).

Eastman, C. M., and K. Weiler. "Geometric Modeling Using the Euler Operators." In *Proceedings of the First Annual Conference on Computer Graphics in CAD/CAM Systems.* Cambridge, Mass.: Massachusetts Institute of Technology. 1979.

Eck, M. "Degree Reduction of Bézier Curves." *Computer Aided Geometric Design* 10(3–4): 237 (1993).

Elber, G., and E. Cohen. "Tool Path Generation for Free-Form Surface Models." *Computer Aided Design,* 26(6): 490 (1994).

Elmaraghy, W. H., S. R. Valzuri, and B. M. Skubnik. "Intersection Volumes and Surface Areas of Cylinders for Geometrical Modeling and Tolerancing." *Computer Aided Design* 26(1): 29 (1994).

Epstein, M. "On the Influence of Parametrization in Parametric Interpolation." *SIAM Journal of Numerical Analysis* 13: 261–268 (1976).

Eshleman, A. L. Informal notes on parametric cubic curves. Undated.

Eshleman, A. L., and H. D. Meriwether. "Animated Display of Dynamic Characteristics of Complex Structures." Presented to the 1966 UAIDE Annual Meeting, San Diego, Calif., Oct.–Nov. 1966.

———. "Graphic Applications to Aerospace Structural Design Problems." Presented to SHARE, Fourth Annual Design Automation Workshop, Los Angeles, June 1967.

Falcidieno, B. "Special Issue: Modeling in Computer Graphics." Editorial. *Computer Aided Design* 26(8): 595 (1994).

Farin, G. "Bézier Polynomials over Triangles and the Construction of Piecewise C^r Polynomials." Report TR/91. Department of Mathematics, Brunel University, Uxbridge, England, Jan. 1980.

———. "Designing C^1 Surfaces Consisting of Triangular Cubic Patches." *Computer Aided Design* 14(5): 253–256 (1982).

———. "Piecewise Triangular C^1 Surface Strips." *Computer Aided Design* 18(1): 45–47 (1986).

———. "Triangular Bernstein-Bézier Patches." *Computer Aided Geometric Design* 3(2): 83–128 (1986).

———. "Curvature Continuity and Offsets for Piecewise Conics." *ACM Transactions on Graphics* 8(2): 89 (1989a).

———. "Rational Curves and Surfaces." In *Mathematical Methods in Computer Aided Geometric Design.* New York: Academic Press, 1989b.

———. "Trends in Curve and Surface Design." *Computer Aided Design* 21(5): 293 (1989).

———. "From Conics to NURBS: A Tutorial and Survey." *IEEE Computer Graphics and Applications* 12(5): 78 (1992).

———. *Curves and Surfaces for Computer-Aided Geometric Design.* New York: Academic Press, 1993a.

———. "Tighter Convex Hulls for Rational Bézier Curves." *Computer Aided Geometric Design* 10(2): 123 (1993b).

Farin, G., ed. *Geometric Modeling: Algorithms and New Trends.* New York: SIAM, 1987.

———. ed. *NURBS for Curve and Surface Design,* Philadelphia, SIAM, 1991.

Farin, G., and P. Barry. "A Link between Lagrange and Bézier Curve and Surface Schemes." *Computer Aided Design* 18: 525–528 (1986).

Farin, G., and N. Sapidis. "Curvature and the Fairness of Curves and Surfaces." *IEEE Computer Graphics and Applications* 9(2): 52–57 (1989).

Farouki, R. T., and J. K. Hinds. "A Hierarchy of Geometric Forms." *IEEE Computer Graphics and Applications* 5(5): 51–78 (1985).

Farouki, R., and V. Rajan. "On the Numerical Condition of Polynomials in Bernstein Form." *Computer Aided Geometric Design* 4(3): 191–216 (1987).

Faux, I. D., and M. J. Pratt. *Computational Geometry for Design and Manufacture.* Chichester, England: Halsted Press, 1980.

Ferguson, D. "Construction of Curves and Surfaces Using Numerical Optimization Techniques." *Computer Aided Design* 18(1): 15–21 (1986).

Ferguson, D. R., and T. A. Grandine. "On the Construction of Surface Interpolating Curves: I. A Method for Handling Nonconstant Parameter Curves." *ACM Transactions on Graphics* 9(2): 212 (1990).

Ferguson, E. N. "Computer Graphics Education Directory." *Computer Graphics* 23(4): 243 (1989).

Ferguson, J. C. "Multivariable Curve Interpolation." *Journal of the ACM* 11: 221–28 (1964).

Ferrucci, V., and A. Paoluzzi. "Extrusion and Boundary Evaluation for Multidimensional Polyhedra." *Computer Aided Design* 23(1): 40 (1991).

Filip, D. J. "Adaptive Subdivision Algorithms for a Set of Bézier Triangles." *Computer Aided Design* 18(2): 74–78 (1986).

———. "Blending Parametric Surfaces." *ACM Transactions on Graphics* 8(3): 164 (1989).

———. "Determining the Orientation of Closed Planar Curves." *Computer Aided Design* 22(7): 401 (1990).

Filip, D. J., and T. W. Ball. "Surfaces: Procedurally Representing Lofted Surfaces." *IEEE Computer Graphics and Applications* 9(6): 27–33 (1989).

Fjallstrom, P.-O. "Evaluation of a Delaunay-Based Method for Surface Approximation." *Computer Aided Design* 25(11): 711 (1993).

Fong, P., and H. P. Seidel. "Control Points for Multivariate B-Spline Surfaces over Arbitrary Triangulations." *Computer Graphics Forum* 10(4): 309 (1991).

———. "An Implementation of Triangular B-Spline Surfaces over Arbitrary Triangulations." *Computer Aided Geometric Design* 10(3–4): 267 (1993).

Forrest, A. R. "Curves and Surfaces for Computer-Aided Design." Ph.D. diss., Cambridge University, 1968.

———. "Coons Surfaces and Multivariate Functional Interpolation." CAD Group document submitted to the *Journal of the ACM*. Cambridge University, Feb. 1969.

———. "Computational Geometry." *Proceedings of the Royal Society of London* A(321): 187–195 (1971).

———. "Computer-Aided Design of Three-Dimensional Objects: A Survey." *Proceedings of the ACM/AICA International Computing Symposium*. Venice, Italy: ACM, 1972.

———. "Interactive Interpolation and Approximation by Bézier Polynomials." *Computing Journal* 15: 71–79 (1972).

———. "A New Curve Form for Computer-Aided Design." CAD Group document no. 66, Cambridge University, June 1972.

———. "On Coons and Other Methods for Representation of Curved Surfaces." *Computer Graphics and Image Processing* 1(4): 341–389 (1972).

———. "Computational Geometry—Achievements and Problems." In *Computer Aided Geometric Design*, eds. R. E. Barnhill and R. F. Riesenfeld. New York: Academic Press, 1974.

———. "On the Rendering of Surfaces." *Computer Graphics* 13(3): 253 (1979).

———. "Recent Work on Geometric Algorithms." In *Mathematical Methods in Computer Graphics*, ed. K. Brodlie. New York: Academic Press, 1980.

———. "The Twisted Cubic Curve: A Computer-Aided Geometric Design Approach." *Computer Aided Design* 12(4): 165–172 (1980).

———. "Free-Form Surfaces and Solid Modeling." In *SIGGRAPH '84 Course Notes on Free-Form Curves and Surfaces.* ACM, 1984.

Fowler, B. "Geometric Manipulation of Tensor Product Surfaces." *Computer Graphics (Special Issue: 1992 Symposium on Interactive 3D Graphics)*: 101–108 (1992).

Fowler, B., and R. Bartels. "Constraint-Based Curve Manipulation." *IEEE Computer Graphics and Applications* 13(5): 43 (1993).

Gans, D. *Transformations and Geometries.* New York: Appleton-Century-Crofts, 1969.

Georgiades, P. N., and D. P. Greenberg. "Locally Manipulating the Geometry of Curved Surfaces." *IEEE Computer Graphics and Applications* 12(1): 54–64 (1992).

Ghosh, P., and P. K. Jain. "An Algebra of Geometric Shapes." *IEEE Computer Graphics and Applications* 13(5): 50 (1993).

Goldfeather, J. "Solid Modeling." *IEEE Computer Graphics and Applications* 9(3): 20–28 (1989).

Goldman, R. "Subdivision Algorithms for Bézier Triangles." *Computer Aided Geometric Design* 159–166 (1983).

———. "Illicit Expressions in Vector Algebra." *ACM Transactions on Graphics* 4(3): 223–243 (1985).

———. "The Method of Resolvents: A Technique for the Implicitization, Inversion, and Intersection of Non-Planar, Parametric, Rational Cubic Curves." *Computer Aided Geometric Design* 2(4): 237–255 (1985).

Goldman, R., and T. DeRose. "Recursive Subdivision without the Convex Hull Property." *Computer Aided Geometric Design* 3(4): 247–266 (1986).

Goldman, R., and C. Micchelli. "Algebraic Aspects of Geometric Continuity." In *Mathematical Methods in Computer Aided Geometric Design,* eds. T. Lyche and L. Schumaker. New York: Academic Press, 1989.

Goodman, T. "Properties of Beta-Splines." *Journal of Approximation Theory* 44(2): 132–153 (1985).

Gordon, W. J. "Free-Form Surface Interpolation through Curve Networks." Technical Report GMR-921. General Motors Research Laboratories, 1969.

———. "Blending-Function Methods of Bivariate and Multivariate Interpolation and Approximation." *SIAM Journal of Numerical Analysis* 8(1): 158–177 (1971).

———. "An Operator Calculus for Surface and Volume Modeling." *IEEE Computer Graphics and Applications* 3(7): 18–22 (1983).

Gordon, W. J., and R. F. Riesenfeld. "Bernstein-Bézier Methods for the Computer-Aided Design of Freeform Curves and Surfaces." *Journal of the ACM* 21(2): 293–310 (1974a).

———. "B-Spline Curves and Surfaces." *Computer Aided Geometric Design*, eds. R. E. Barnhill and R. F. Riesenfeld. New York: Academic Press, 1974b.

Gosh, P., and P. K. Jain. "An Algebra of Geometric Shapes." *IEEE Computer Graphics and Applications* 13(5): 50 (1993).

Gossard, D. C., R. P. Zuffante, and H. Sakurai. "Representing Dimensions, Tolerances, and Features in MCAE Systems." *IEEE Computer Graphics and Applications*, 8(2): 69–84 (1988).

Gossing, T. "Bulge, Shear and Squash: A Representation for the General Conic Arc." *Computer Aided Design* 13(2): 81–84 (1981).

Grabowski, H., and X. Li. "Coefficient Formula and Matrix of Nonuniform B-Spline Functions." *Computer Aided Design* 24(12): 637 (1992).

Gregory, J. A. "Smooth Interpolation without Twist Constraints." In *Computer Aided Geometric Design*, eds. R. E. Barnhill and R. F. Riesenfeld. New York: Academic Press, 1974.

———. "C^1 Rectangular and Non-Rectangular Surface Patches." In *Surfaces in Computer Aided Geometric Design*, eds. R. E. Barnhill and W. Boehm. Amsterdam: North-Holland, 1983a.

———. "*N*-Sided Surface Patches." In *Mathematics of Surfaces*, ed. J. A. Gregory. Oxford: Clarendon Press, 1983b.

———, ed. *The Mathematics of Surfaces*. Oxford: Claredon Press, 1986.

Greiner, G., and H. Seidel. "Modeling with Triangular B-Splines." *IEEE Computer Graphics and Applications* 14(2): 56–59 (1994).

Griessmair, J., and W. Purgathofer. "Deformation of Solids with Trivariate B-Splines." *Proceedings of Eurographics '89*. Amsterdam: North-Holland, 1989.

Guenter, B., and R. Parent. "Computing the Arc Length of Parametric Curves." *IEEE Computer Graphics and Applications* 10(3): 72–78 (1990).

Gunther, O., and S. Dominguez. "The Performance of Curve Representation Schemes." *IEEE Computer Graphics and Applications* 13(3): 55 (1993).

Gursoz, E. L., Y. Choi, and F. B. Prinz. "Boolean Set Operations on Non Manifold Boundary Representation Objects." *Computer Aided Design* 23(1): 33 (1991).

Hagen, H. "Geometric Spline Curves." *Computer Aided Geometric Design* 2(1–3): 223–228 (1985).

———. "Bézier Curves with Curvature and Torsion Continuity." *Rocky Mountain Journal of Mathematics* 16(3): 629–638 (1986).

———. "Geometric Surface Patches without Twist Constraints." *Computer Aided Geometric Design* 3: 179–184 (1986).

Hahn, J. "Geometric Continuous Patch Complexes." *Computer Aided Geometric Design* 6(1): 11–22 (1989).

Hall, M., and J. Warren. "Adaptive Polygonalization of Implicitly Defined Surfaces." *IEEE Computer Graphics and Applications* 10(6): 33–42 (1990).

Hanna, S. L., J. F. Abel, and D. P. Greenberg. "Intersection of Parametric Surfaces by Means of Look-up Tables." *IEEE Computer Graphics and Applications* 3(7): 39–48 (1983).

Hands, J. "Reparametrization of Rational Surfaces." In *The Mathematics of Surfaces II,* ed. R. Martin. Oxford University, 1987.

Hartley, P., and C. Judd. "Parametrization and Shape of B-Spline Curves." *Computer Aided Design* 12(5): 235–238 (1980).

———. "Parametrization of Bézier-Type B-Spline Curves." *Computer Aided Design* 10(2): 130–135 (1978).

Hatvany, J., W., M. Newman, and M. A. Sabin. "World Survey of Computer-Aided Design." *Computer Aided Design* 25(12): 776 (1993).

Heisserman, J. "Generative Geometric Design." *IEEE Computer Graphics and Applications* 14(2): 37–45 (1994).

Herman, I. "On the Projective Invariant Representation of Conics in Computer Graphics." *Computer Graphics Forum* 8(4): 301 (1989).

Higashi, M. "High-Quality Solid-Modeling System with Free-Form Surfaces." *Computer Aided Design* 25(3): 172 (1993).

Hilbert, D. and S. Cohn-Vossen. *Geometry and the Imagination.* New York: Chelsea, 1952.

Hinds, J. K. "Methods of Representing Conic Segments by Means of Rational Parametric Cubic Equations." Northern Illinois University, De Kalb, Ill. Informal notes, undated.

Hoffmann, C. M. *Geometric and Solid Modeling: An Introduction.* San Mateo, Calif.: Morgan Kaufman, 1989.

———. "Implicit Curves and Surfaces in CAGD." *IEEE Computer Graphics and Applications* 13(1): 79 (1993).

Hoffmann, C. M. and J. E. Hopcroft. "The Potential Method for Blending Surfaces." In *Geometric Modeling: Algorithms and New Trends*, ed. G. Farin. Philadelphia: SIAM, 1987.

Hoffmann, C. M., J. E. Hopcroft, and M. S. Karasick. "Robust Set Operations on Polyhedral Solids." *IEEE Computer Graphics and Applications* 9(6): 50–59 (1989).

Hoffmann, C. M., and J. Zhou. "Some Techniques for Visualizing Surfaces in Four-Dimensional Space." *Computer Aided Design* 23(1): 83 (1991).

Hohmeyer, M. E., and B. A. Barsky. "Rational Continuity: Parametric, Geometric, and Frenet Frame Continuity of Rational Curves." *ACM Transactions on Graphics* 8(4): 325 (1989).

Hojnicki, J. S. and P. R. White. "Converting CAD Wireframe Data to Surfaced Representations." *Computers in Mechanical Engineering* Mar./Apr.: 19–25 (1988).

Holmström, L. "Piecewise Quadric Blending of Implicitly Defined Surfaces." *Computer Aided Geometric Design* 4: 171 (1987).

Hosaka, M., and F. Kimura. "Non Four-Sided Patch Expressions with Control Points." *Computer Aided Geometric Design* 1(1): 75–88 (1984).

Hoschek, J. "Offset Curves in the Plane." *Computer Aided Design* 17(2): 77 (1985).

———. "Circular Splines." *Computer Aided Design* 24(11): 611 (1992).

———. "Free-Form Curves and Free-Form Surfaces." *Computer Aided Geometric Design* 10(3–4): 173 (1993).

Hoschek, J., and F. Schneider. "Spline Conversion for Trimmed Rational Bézier and B-Spline Surfaces." *Computer Aided Design* 22(9): 580 (1993).

Houghton, E., E. Emmett, R. Factor, and L. Sabharwal. "Implementation of a Divide-and-Conquer Method for the Intersection of Parametric Surfaces." *Computer Aided Geometric Design* 2(1–3): 173–184 (1985).

Joe, B. "Discrete Beta-Splines." *Computer Graphics* 21(4): 137 (1987).

———. "Multiple-Knot and Rational Cubic Beta Splines." *ACM Transactions on Graphics* 8(2): 100 (1989).

———. "Rational Beta-Spline Curves and Surfaces and Discrete Beta-Splines." Technical Report no. TR 87-04. Department of Computing Science, University of Alberta, Edmonton, April 1987.

———. "Knot Insertion for Beta-Spline Curves and Surfaces." *ACM Transactions on Graphics* 9(1): 41 (1990).

———. "Quartic Beta-Splines." *ACM Transactions on Graphics* 9(3): 301 (1990).

Joe, B., W. Wang, and F. Cheng. "Reduced-Knot NURBS Representations of Rational G^1 Composite Bézier Curves." *Computer Aided Design* 26(5): 393 (1994).

Julien, C., and M. Laborie. "Bias and Tension, Two Shape Parameters for Convenient Specification of the Parametric Cubic Segment." *Computer Aided Design* 21(8): 485 (1989).

Kalay, Y. E. "The Hybrid Edge: A Topological Data Structure for Vertically Integrated Geometric Modeling." *Computer Aided Design* 21(3): 130 (1989).

Kato, K. "Generation of *n*-Sided Surfaces Patches with Holes." *Computer Aided Design* 23(10): 676 (1991).

Kela, A. "Hierarchical Octree Approximations for Boundary Representation-Based Geometrie Models." *Computer Aided Design* 21(6): 355 (1989).

Kela, A., J. E. Davis, and P. M. Finnigan. "Wireframe to Solid Models: A Survey." In *Proceedings of the ASME Computers in Engineering Conference.* ASME, 1989.

Kent, J. R., W. E. Carlson, and R. E. Parent. "Shape Transformation for Polyhedral Objects. *Computer Graphics* 26(2): 47 (1992).

Klok, F. "Two Moving Coordinate Frames for Sweeping along a 3D Trajectory." *Computer Aided Geometric Design* 3: 217 (1986).

Kocic, L. M. "Modification of Bézier Curves and Surfaces by Degree-Elevation Technique." *Computer Aided Design* 23(10): 692 (1991).

Koenderink, J. J. *Solid Shape.* Cambridge, MIT Press, 1990.

Koparkar, P. "Designing Parametric Blends: Surface Models and Geometric Correspondence." *The Visual Computer* 7(1): 39 (1991).

Kouh, J. S., and S. W. Chau. "Computer-Aided Geometric Design and Panel Generation for Hull Forms Based on Rational Cubic Bézier Curves." *Computer Aided Geometric Design* 10(6): 537 (1993).

Kramer, G. A. *Solving a Geometric Constraint System: A Case Study in Kinematics.* Cambridge, Mass.: Massachusetts Institute of Technology Press, 1991.

Kriezis, G. A., N. M. Patrikalakis, and F.-E. Wolter. "Topological and Differential-Equation Methods for Surface Intersections." *Computer Aided Design* 24(1): 41 (1992).

Kuriyama, S. "Surface Modeling with an Irregular Network of Curves Sweeping and Blending." *Computer Aided Design* 26(8): 597 (1994).

Laidlaw, D. H., W. H. Trumbore, and J. F. Hughes. "Constructive Solid Geometry for Polyhedral Objects." *Computer Graphics* 20(4): 161 (1986).

Lancaster, P., and K. Salkauskas. *Curve and Surface Fitting.* New York: Academic Press, 1986.

Lane, E. P. *Metric Differential Geometry of Curves and Surfaces,* University of Chicago Press, Chicago, 1940.

Lane, J. M., and R. F. Riesenfeld. "A Theoretical Development for Computer Generation and Display of Piecewise Polynomial Surfaces." *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-2: 35–46 (1980).

Lang-Lendorff, G., and J. Unterburg. "Changes in Understanding of CAD/CAM: A Database-Oriented Approach." *Computer Aided Design* 21(5): 309 (1989).

Lasser, D. "Bernstein-Bézier Representation of Volumes." *Computer Aided Geometric Design* 2(1–3): 145–150 (1985).

———. "Intersection of Parametric Surfaces in the Bernstein-Bézier Representation." *Computer Aided Design* 18(4): 186–192 (1986).

Laurent-Gengoux, P., and M. Mekhilef. "Optimization of NURBS Representation." *Computer Aided Design* 25(11): 699 (1993).

Lazarus, F., S. Coquillart, and P. Jancene. "Axial Deformations: An Intuitive Deformation Technique." *Computer Aided Design* 26(8): 607 (1994).

Lee, B., and D. Fredericks. "Intersection of Parametric Surface Patches and a Plane." *IEEE Computer Graphics and Applications* 4(8): 48–51 (1984).

Lee, D. T., and F. P. Preparata. "Computational Geometry—A Survey." *IEEE Transactions on Computers* C-13(12): 1072–1101 (1984).

Lee, E. "The Rational Bézier Representation for Conics." In *Geometric Modeling: Algorithms and New Trends,* ed. G. Farin. Philadelphia: SIAM, 1987.

Lee, Y. T., and A. S. Koh. Performing Euler angle rotations in CAD systems. *Computer-Aided Design* 27(1): 75–78, Jan 95.

Lee, Y. T., and A. A. G. Requicha. "Algorithms for Computing the Volume and Other Integral Properties of Solid Objects." Production Automation Project Technical Memo no. 35. Production Automation Project, University of Rochester, Rochester, N.Y., 1980.

———. "Algorithms for Computing the Volume and Other Integral Properties of Solids. Part 1: Known Methods and Open Issues." *Communications of the ACM* 25(9): 635–641 (1982a).

———. "Algorithms for Computing the Volume and Other Integral Properties of Solids. Part 2: A Family of Algorithms Based on Representation

Conversion and Cellular Approximation." *Communications of the ACM* 25(9): 642–650 (1982b).

Leesley, M. "PC CAD is Closing in on Mainframe CAD." Comment. *Computer Aided Design* 25(12): 739 (1993).

Lequelte, R. "Automatic Construction of Curvilinear Solids from Wireframe Views." *Computer Aided Design* 29(4): 171 (1988).

Letcher, J. S. Jr. "Lofting and Fabrication of Compound-Curved Plates." *J. Ship Res.* 37(2): 166–175 (1993).

Letcher, J. S. Private communication. Aug. 28, 1995.

Letcher, J. S. Jr., and D. M. Shook. "Applications of Relational Geometric Synthesis in Sailing Yacht Design." In *Proceedings, 11th Chesapeake Sailing Yacht Symposium.* Annapolis, Md.: SNAME, 1993.

Letcher, Jr., J. S. and D. M. Shook. "Parametric Design and Gridding through Relational Geometry." In *Proceedings of the Surface Modeling, Grid Generation and Related Issues in Computational Flow Dynamics (CFD) Solutions.* NASA Lewis Research Center, May 1995a.

———. "NURBS Considered Harmful for Gridding (Alternative Offered)." Paper presented at the Fourth Annual International Meshing Roundtable, sponsored by Sandia National Laboratories, Albuquerque, N.M., Oct. 1995b.

Letcher, Jr., J. S., D. M. Shook, and S. G. Shepherd. "Relational Geometric Synthesis. Part 1: Framework." *Computer Aided Design* 27(11) (1995).

Levin, J. "Mathematical Models for Determining the Intersection of Quadric Surfaces." *Computer Graphics and Image Processing* 11: 73–87 (1979).

Li, H., and S.-Q. Liu. "Local Interpolation of Curvature-Continuous Surfaces." *Computer Aided Design* 24(9): 491 (1992).

Lien, S., M. Shantz, and V. Pratt. "Adaptive Forward Differencing for Rendering Curves and Surfaces." SIGGRAPH '87 Proceedings. *Computer Graphics* 21(4): 111–118 (1987).

Lienhardt, P. "Topological Models for Boundary Representation: A Comparison with *n*-Dimensional Generalized Maps." *Computer Aided Design* 23(1): 59 (1991).

Light, R., and D. Gossard. "Modification of Geometric Models through Variational Geometry." *Computer Aided Design* 14(4): 209–214 (1982).

Lin, F., and W. T. Hewitt. "Expressing Coons-Gordon Surfaces as NURBS." *Computer Aided Design* 26(2): 145 (1994).

Liu, D. "A Geometric Condition for Smoothness between Adjacent Bézier Surface Patches." *Acta Mathematica Applications Sinica* 9: 432 (1986).

Liu, D. and J. Hoschek. "G^1 and C^1 Continuity Conditions between Adjacent Rectangular and Triangular Bézier Surface Patches." *Computer Aided Design* 21(4): 194 (1989).

Liu, Q. and T. C. Sun. "G^1 Interpolation of Mesh Curves." *Computer Aided Design* 26(4): 259 (1994).

Llewelyn, A. I. "Review of CAD/CAM." *Computer Aided Design* 21(5): 297 (1989).

Lodha, S., and J. Warren. "Bézier Representation for Quadric Surface Patches." *Computer Aided Design* 22(9): 574 (1990).

Loop, C. T., and T. D. DeRose. "A Multisided Generalization of Bézier Surfaces." *ACM Transactions on Graphics* 8(3): 204 (1989).

Lorentz, G. G. *Bernstein Polynomials.* Toronto Press, 1953.

Lossing, D. L., and A. L. Eshleman. "Planning a Common Data Base for Engineering and Manufacturing." Presented to SHARE *XLIII,* Chicago, Aug. 1974.

Lounsbery, M., S. Mann, and T. De Rose. "Parametric Surface Interpolation." *IEEE Computer Graphics and Applications* 12(5): 45–52 (1992).

Lyche, T., and V. Morken. "Knot Removal for Parametric B-Spline Curves and Surfaces." *Computer Aided Geometric Design* 4(3): 217–230 (1987).

Lyche, T., and L. L. Schumaker. *Mathematical Methods in Computer Aided Geometric Design.* New York: Academic Press, 1989.

Lyche, T., and L. L. Schumaker, eds. *Curves and Surfaces.* New York: Academic Press, 1992.

Maekawa, T., and N. M. Patrikalakis. "Computation of Singularities and Intersections of Offsets of Planar Curves." *Computer Aided Geometric Design* 10(5): 407 (1993).

Mallet, J.-L. "Discrete Smooth Interpretation." *ACM Transactions on Graphics* 8(2): 121 (1989).

———. "Discrete Smooth Interpolation in Geometric Modeling." *Computer Aided Design* 24(4): 178 (1992).

Manning, J. R. "Continuity Conditions for Spline Curves." *The Computer Journal* 17: 181–186 (1974).

Manocha, D. "Solving Systems of Polynomial Equations." *IEEE Computer Graphics and Applications* 14(2): 46–55 (1994).

Manocha, D., and J. Canny. "A New Approach for Surface Intersection." *International Journal of Computational Geometry and Applications* 1(4): 491 (1991).

———. "Implicit Representations of Rational Parametric Surfaces." *Journal of Symbolic Computation* 13: 485 (1992).

Mäntylä, M. "An Inversion Algorithm for Geometric Models." Proceedings of SIGGRAPH '82. *Computer Graphics* 16(3): 51–59 (1982).

———. "A Note on the Modeling Space of Euler Operators." *Computer Vision, Graphics, and Image Processing* 26(1): 45–60 (1984).

———. "Boolean Operations of 2-Manifolds through Vertex Neighborhood Classification." *ACM Transactions on Graphics* 5(1): 1 (1986).

———. *An Introduction to Solid Modeling.* Rockville, Md.: Computer Science, 1988.

Mäntylä, M., and M. Tamminen. "Localized Set Operations for Solid Modeling." Proceedings of SIGGRAPH '83. *Computer Graphics* 17(3): 279–288 (1983).

Markowsky, G., and M. A. Wesley. "Fleshing Out Wireframes." *IBM Journal of Research and Development* 24(5): 582 (1980).

Martin, R. R., ed. *The Mathematics of Surfaces II.* Oxford: Claredon, 1987.

Martin, R. R., and P. C. Stephenson. "Sweeping of Three-Dimensional Objects." *Computer Aided Design* 22(4): 223 (1990).

Masuda, H. "Topological Operators and Boolean Operations for Complex-Based Nonmanifold Geometric Models." *Computer Aided Design* 25(2): 119 (1993).

Mazzetti, M., and L. Ciminiera. "Computing CSG-Free Boundaries as Algebraic Expressions." *Computer Aided Design* 26(6): 417 (1994).

McConnell, J. J. "Dissertation Abstracts in Computer Graphics." *Computer Graphics* 23(2): 191 (1989).

McLaughlin, H. "Shape Preserving Planar Interpolation: An Algorithm." *IEEE Computer Graphics and Applications* 3(3): 58–67 (1985).

Meagher, D. J. "Octree Encoding: A New Technique for the Representation, Manipulation, and Display of Arbitrary Three-Dimensional Objects by Computer." Technical Report IPL-TR-80. Image Processing Laboratory, Rensselaer Polytechnic Institute, Troy, N.Y., Oct. 1980.

———. "Efficient Synthetic Image Generation of Arbitrary 3-D Objects." *Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing.* IEEE, 1982a.

———. "Geometric Modeling Using Octree Encoding." *Computer Graphics and Image Processing* 19: 129–147 (1982b).

———. "The Octree Encoding Method for Efficient Solid Modeling." Technical Report IPL-TR-032. Image Processing Laboratory, Rensselaer Polytechnic Institute, Troy, N.Y., Aug. 1982c.

———. "Octree Generation, Analysis, and Manipulation." Technical Report IPL-TR-027. Image Processing Laboratory, Rensselaer Polytechnic Institute, Troy, N.Y., Apr. 1982d.

Meek, D. S., and D. J. Walton. "Shape Determination of Planar Uniform Cubic B-Spline Segments." *Computer Aided Design* 22(7): 434 (1990).

Menon, J. P. "Constructive Shell Representations for Free-Form Surfaces and Solids." *IEEE Computer Graphics and Applications* 14(2): 24–36 (1994).

Menon, J., R. J. Marisa, and J. Zagajac. "More Powerful Solid Modeling through Ray Representations." *IEEE Computer Graphics and Applications* 14(3): 22–35 (1994).

Meriwether, H. D. Informal notes. Undated.

———. Personal communication, Aug. 1995.

Micchelli, C., and H. Prautzsch. "Computing Surfaces Invariant under Subdivision." *Computer Aided Geometric Design* 4(4): 321–328 (1988).

Middleditch, A. E., and K. H. Sears. "Blend Surfaces for Set Theoretic Volume Modeling Systems." *Computer Graphics* 19(3): 161 (1985).

Milenkovic, V. "Robust Polygon Modeling." *Computer Aided Design* 25(9): 546 (1993).

Miller, J. R. "Sculptured Surfaces in Solid Models: Issues and Alternative Approaches." *IEEE Computer Graphics and Applications* 6(12): 37–48 (1986).

———. "Geometric Approaches to Nonplanar Quadric Surface Intersection Curves." *ACM Transactions on Graphics* 6(4): 274–307 (1987).

———. "Incremental Boundary Evaluation Using Inference of Edge Classifications." *IEEE Computer Graphics and Applications* 13(1): 71–78 (1993).

Miller, J. V., D. E. Breen, and R. M. O'Bara. "Geometrically Deformed Models: A Method for Extracting Closed Geometric Models from Volume Data." *Computer Graphics* 25(4): 217 (1991).

Mortenson, M. E. *Geometric Modeling.* 1st ed. New York: Wiley, 1985.

———. *Computer Graphics: An Introduction to the Mathematics and Geometry.* New York: Industrial Press, 1989.

———. *Geometric Transformations.* New York: Industrial Press, 1995.

Nasri, A. H. "Polyhedral Subdivision Methods for Free-Form Surfaces." *ACM Transactions on Graphics* 6(1): 29 (1987).

Nicholl, R. A., and T. M. Nicholl. "Performing Geometric Transformations by Program Transformation." *ACM Transactions on Graphics* 9(1): 28 (1990).

Nielson, G. M. "Computer-Aided Geometric Design." *IEEE Computer Graphics and Applications* 12(5): 29–30 (1992).

Ning, P., and J. Bloomenthal. "An Evaluation of Implicit Surface Tilers." *IEEE Computer Graphics and Applications* 13(6): 33–40 (1993).

Ohkura, K., and Y. Kakazu. "Generalization of the Potential Method for Blending Three Surfaces." *Computer Aided Design* 24(11): 599 (1992).

Ohsuga, S. "Toward Intelligent CAD Systems." *Computer Aided Design* 21(5): 315 (1989).

Oliver, J. C., J. S. Letcher, Jr., and N. Salvesen. "Performance Predictions for *Stars & Stripes*." *Trans. S.N.A.M.E.* 95: 239–261 (1987).

O'Neill, B. *Elementary Differential Geometry*, Academic Press, New York, 1966.

Palacol, E. L. and E. L. Stanton. "Anisotropic Parametric Plate Discrete Elements." *International Journal of Numerical Methods in Engineering* 4: 413–25 (1973).

Patrikalakis, N. M. "Surface-to-Surface intersections." *IEEE Computer Graphics and Applications* 13(1): 89–95 (1993).

Patterson, R. R. "Projective Transformations of the Parameter of a Bernstein-Bézier Curve." *ACM Transactions on Graphics* 4(4): 296 (1985).

———. "Projective Transformations of the Parameter of a Rational Bernstein-Bézier Curve." *ACM Transactions on Graphics* 4: 276–290 (1986).

Pavlidis, T. "Curve Fitting with Conic Splines." *ACM Transactions on Graphics* 2: 1–31 (1983).

Pegna, J. "Interactive Design of Curvature Continuous Fairing Surfaces." *Proceedings of the 8th International Conference on Offshore Mechanics and Arctic Engineering of the ASME*. New York: ASME, 1989.

Pegna, J. and F. E. Wolter. "Geometrical Criteria to Guarantee Curvature Continuity of Blend Surfaces." *Journal of Mechanical Design, Transaction of the ASME* 114(1): 201 (1992).

Peng, Q. S. "An Algorithm for Finding Intersection Lines between Two B-Spline Surfaces." *Computer Aided Design* 16(4): 191–196 (1984).

Penna, M., and R. Patterson. *Projective Geometry and its Applications to Computer Graphics*. New York: Prentice Hall, 1986.

Peters, G. J. "Interactive Computer Graphies Application of the Parametric Bicubic Surface." In *Computer Aided Geometric Design*, eds. R. E. Barnhill and R. F. Riesenfeld. New York: Academic Press, 1974.

Peters, J. "Loeal Generalized Hermite Interpolation by Quartic C^2 Space Curves." *ACM Transactions on Graphics* 8(3): 235 (1989).

Pham, B. "Offset Curves and Surfaces: A Brief Survey." *Computer Aided Design* 24(4): 223 (1992).

Phillips, M. B., and G. M. Odell. "An Algorithm for Locating and Displaying the Intersection of Two Arbitrary Surfaces." *IEEE Computer Graphics and Applications* 4(9): 48 (1984).

Piegl, L. "A Geometric Investigation of the Rational Bézier Scheme in Computer Aided Geometric Design." *Computers in Industry* 7: 401–410 (1987).

———. "Hermite- and Coons-like Interpolants Using Rational Bézier Approximation Form with Infinite Control Points." *Computer Aided Design* 20(1): 2–10 (1988).

———. "Geometric Method of Intersecting Natural Quadrics Represented in Trimmed Surface Form." *Computer Aided Design* 21(4): 201 (1989a).

———. "Key Developments in Computer-Aided Geometric Design." *Computer Aided Design* 21(5): 262 (1989b).

———. "Modifying the Shape of Rational B-Splines. Part I: Curves." *Computer Aided Design* 21(8): 509–518 (1989c).

———. "Modifying the Shape of Rational B-Splines. Part 2: Surfaces." *Computer Aided Design* 21(9): 538 (1989d).

———. "On NURBS: A Survey." *IEEE Computer Graphics and Applications* 11(1): 55 (1991).

———. *Fundamental Developments of Computer-Aided Geometric Modeling.* New York: Academic Press, 1993.

Piegl, L. and W. Tiller. "Curve and Surface Constructions Using Rational B-Splines." *Computer Aided Design* 19(9): 485–498 (1987).

———. *The NURBS Book.* Berlin: Springer Verlag, 1995.

———. "Circles: A Menagerie of Rational B-Spline Circles." *IEEE Computer Graphics and Applications* 9(5): 48 (1989).

———. "Software-Engineering Approach to Degree Elevation of B-Spline Curves." *Computer Aided Design* 26(1): 17 (1994).

Piper, B. "Visually Smooth Interpolation with Triangular Bézier Patches." In *Geometric Modeling: Algorithms and New Trends*, ed. G. Farin. SIAM, 1987.

Pottman, H. "Projectively Invariant Classes of Geometric Continuity for CAGD." *Computer Aided Geometric Design* 6(4): 307–322 (1989).

———. "Locally Controllable Conic Splines with Curvature Continuity." *ACM Transactions on Graphics* 10(4): 366 (1991).

Pratt, M. J., R. J. Goult, and L. Lee. "On Rational Parametric Curve Approximation." *Computer Aided Geometric Design* 10(3–4): 363 (1993).

Pratt, V. "Techniques for Conic Splines." *Computer Graphics* 19(3): 151 (1985).

———. "Direct Least-Squares Fitting of Algebraic Surfaces." *Computer Graphics* 21(4): 145 (1987).

Prautzsch, H. "Degree Elevation of B-Spline Curves." *Computer Aided Geometric Design* 1(2): 193–198 (1984).

———. "A Round Trip to B-Splines via de Casteljau." *ACM Transactions on Graphics* 9(3): 243 (1989).

Prautzsch, H., and C. Micchelli. "Computing Curves Invariant under Halving." *Computer Aided Geometric Design* 4(1–2): 133–140 (1987).

Preparata, F. P., and M. I. Shamos. *Computational Geometry.* New York: Springer-Verlag, 1985.

Putnam, L. K., and P. A. Subrahmanyam. "Boolean Operations on *n*-Dimensional Objects." *IEEE Computer Graphics and Applications* 6(6): 43–51 (1986).

Qin, H. "Dynamic NURBS with Geometric Constraints to Interactive Sculpting." *ACM Transactions on Graphics* 13(2): 103 (1994).

Qin, K., J. Sun, and X. Wang. "Representing Conics Using NURBS of Degree Two." *Computer Graphics Forum* 11(5): 285 (1992).

Ramshaw, L. "Béziers and B-Splines as Multiaffine Maps." In *Theoretical Foundations of Computer Graphics and CAD,* ed. R. Earnshaw. New York: Springer-Verlag, 1988.

Rappoport, A. "Rendering Curves and Surfaces with Hybrid Subdivision and Forward Differencing." *ACM Transactions on Graphics* 10(4): 323 (1991).

Ratshek, H., and J. Rokne. "Test for Intersection between Plane and Box." *Computer Aided Design* 25(4): 249 (1993).

———. "Box-Sphere Intersection Tests." *Computer Aided Design* 26(7): 579 (1994).

Redont, P. "Representation and Deformation of Developable Surfaces." *Computer Aided Design* 21(1): 13 (1989).

Requicha, A. A. G. "Mathematical Models of Rigid Solid Objects." Technical Memo no. 28. Production Automation Project, University of Rochester, Rochester, N.Y., 1977.

———. "Representations for Rigid Solids: Theory, Methods, and Systems." *Computing Surveys* 12(4): 437–464 (1980).

———. "Solid Modeling: A 1988 Update." *CAD Based Programming Sensory Robots,* ed. B. Ravani. New York: Springer-Verlag, 1988.

Requicha, A. A. G., and J. R. Rossignac. "Solid Modeling and Beyond." *IEEE Computer Graphics and Applications* 12(5): 31–44 (1992).

Requicha, A. A. G., and R. B. Tilove. "Mathematical Foundations of Constructive Solid Geometry: General Topology of Regular Closed Sets." Technical Memo no. 27. Production Automation Project, University of Rochester, Rochester, N.Y., 1978.

Requicha, A. A. G., and H. B. Voelker. "Constructive Solid Geometry." Technical Memo no. 25, Production Automation Project, University of Rochester, Rochester, N.Y., 1977.

———. "Geometric Modeling of Mechanical Parts and Machining Processes." *COMPCONTROL*. Sopron, Hungary: 1979.

———. "An Introduction to Geometric Modeling and its Applications." *Advances in Information Systems Sciences*, ed. J. T. Tou. Vol. 8. New York: Plenum, 1981.

———. "Solid Modeling: A Historical Summary and Contemporary Assessment." *IEEE Computer Graphics and Applications* 2(2): 9–24 (1982).

———. "Solid Modeling: Current Status and Research Directions." *IEEE Computer Graphics and Applications* 3(27): 25–37 (1983).

———. "Boolean Operations in Solid Modeling: Boundary Evaluation and Merging Algorithms." *Proceedings of IEEE* 73(1): 30 (1985).

Riesenfeld, R. F. "Applications of B-Spline Approximation to Geometric Problems of Computer-Aided Design." Ph.D. diss., Syracuse University, Syracuse, N.Y., 1972. Also published as University of Utah report UTEC-CSc-73-126.

———. "Aspects of Modeling in Computer-Aided Geometric Design." *Proceedings of the National Computer Conference*. AFIPS Press, 1975.

———. "Homogeneous Coordinates and Projective Planes in Computer Graphics." *IEEE Computer Graphics and Applications* 1: 50–55 (1981).

Roberts, L. G. "Homogeneous Matrix Representation and Manipulation of n-Dimensional Constructs." Report MS-1405, Lincoln Laboratory, Massachusetts Institute of Technology, May 1965.

Rockwood, A. P. "The Displacement Method for Implicit Blending Surfaces in Solid Models." *ACM Transactions on Graphics* 8(4): 279 (1989).

Rockwood, A. P., and J. C. Owen. "Blending Surfaces in Solid Modeling." In *Geometric Modeling: Algorithms and New Trends*, ed. G. Farin. New York: SIAM, 1987.

Rosenblum, L. J. "Research Issues in Scientific Visualization." *IEEE Computer Graphics and Applications* 14(2): 61–63 (1994).

Ross, D. T. "The Design and Use of the APT Language for Automatic Programming of Numerically Controlled Machine Tools." *Proceedings of the Computer Applications Symposium*. Chicago: 1959.

———. "The AED Approach to Generalized Computer-Aided Design." *Proceedings of the ACM National Conference*. ACM 1967.

Ross, R. "Transformations." *IEEE Computer Graphics and Applications* 9(3): 70 (1989).

Rossignac, J. R. "Constraints in Constructive Solid Geometry." In *Proceedings of the 1986 Workshop on Interactive 3D graphics*, eds. F. Crow and S. M. Pizer. Chapel Hill, N.C.: Association for Computing Machinery Press, 1986.

Rossignac, J. R., and A. A. G. Requicha. "Offsetting Operations in Solid Modeling." *Computer Aided Geometric Design* 3(2): 129–148 (1986).

———. "Constant Radius Blending in Solid Modeling." *ASME Computers in Mechanical Engineering (CIME)* 3(1): 65–73 (1984).

———. "Constructive Non-regularized Geometry." *Computer Aided Design* 23(1): 21 (1991).

Roulier, J. A. "Specifying the Arc Length of Bézier Curves." *Computer Aided Geometric Design* 10(1): 25 (1993).

Sabin, M. "A Method for Displaying the Intersection Curve of Two Quadric Surfaces." *The Computer Journal* 19: 336–338 (1976).

———. "Non-rectangular Surface Patches Suitable for Inclusion in a B-Spline Surface." In *Proceedings of Eurographics 1983*, ed. P. ten Hagen. Amsterdam: North-Holland, 1983.

———. "Geometric Modeling—Fundamentals." In *Eurographics Tutorials '83*, ed. P. J. W. ten Hagen. Berlin: Springer-Verlag, 1984.

———. "Recursive Subdivision." In *The Mathematics of Surfaces*, ed. J. Gregory. Oxford: Clarendon, 1986.

———. "Some Negative Results in *N*-Sided Patches." *Computer Aided Design* 18(1): 38–44 (1986).

Sakkalis, T., and R. T. Farouki. "Algebraically Rectifiable Parametric Curves." *Computer Aided Geometric Design* 10(6): 551 (1993).

Samet, H. "The Quadtree and Related Hierarchical Data Structures." *ACM Computing Surveys* 16: 187–260 (1984).

Samet, H., and M. Tamminen. "Bintrees, CSG Trees, and Time." *Computer Graphics* 19(3): 121 (1985).

Sapidis, N., and R. Perucchio. "Solid/Solid Classification Operations for Recursive Spatial Decomposition and Domain Triangulation of Solid Models." *Computer Aided Design* 24(10): 517 (1992).

Sarraga, R. F. "Computer Modeling of Surfaces with Arbitrary Shapes." *IEEE Computer Graphics and Applications* 10(2): 67 (1990).

Schene, C. K. "Test for Intersection between a Plane and a Connected Compact Polyhedron." *Computer Aided Design* 26(7): 585 (1994).

Schichtel, M. "G^2 Blend Surfaces and Filling of n-Sided Holes." *IEEE Computer Graphics and Applications* 13(5): 68 (1993).

Schmitt, F. J. M., B. A. Barsky, and W-H Du. "An Adaptive Subdivision Method for Surface Fitting from Sampled Data." *Computer Graphics* 20(4): 179 (1986).

Schoenberg, I. "Contribution to the Problem of Approximation of Equidistant Data by Analytic Functions." *Quarterly of Applied Mathematics* 4: 45–99, 121–141 (1946).

———. "On Variation Diminishing Approximation Methods." In *On Numerical Approximation*, ed. R. E. Langer. Madison, Wis.: University of Wisconsin Press, 1953.

Schumaker, L. L. *Spline Functions: Basic Theory*. New York: Wiley, 1981.

———. "Triangulations in CAGD." *IEEE Computer Graphics and Applications* 13(1): 47 (1993).

Schwartz, A. "Subdividing Bézier Curves and Surfaces." In *Geometric Modeling: Algorithms and New Trends*, ed. G. Farin. Philadelphia: SIAM, 1987.

Sederberg, T. W. "Implicit and Parametric Curves and Surfaces for Computer Aided Geometric Design." Ph.D. thesis, Purdue University, 1983.

———. "Steiner Surface Patches." *IEEE Computer Graphics and Applications* 5(5): 23–36 (1985).

———. "Improperly Parametrized Rational Curves." *Computer Aided Geometric Design* 1(3): 67–75 (1986).

———. "Algorithm for Algebraic Curve Intersection." *Computer Aided Design* 21(9): 547 (1989).

———. "Techniques for Cubic Algebraic Surfaces." Part 1. *IEEE Computer Graphics and Applications* 10(4): 14–25 (1990).

———. "Techniques for Cubic Algebraic Surfaces." Part 2. *IEEE Computer Graphics and Applications* 10(5): 12 (1990).

Sederberg, T., W., and D. C. Anderson. "Steiner Surface Patches." *IEEE Computer Graphics and Applications* 5(5): 23 (1985).

Sederberg, T. W., H. Christiansen, and S. Katz. "Improved Test for Closed Loops in Surface Intersections." *Computer Aided Design* 21(8): 505–508 (1989).

Sederberg, T. W., and R. J. Meyers. "Loop Detection and Surface Patch Intersections." *Computer Aided Geometric Design* 5(2): 161–171 (1988).

Sederberg, T. W., H. N. Christiansen, and S. Katz. "Two Theorems Concerned with Surface Intersection Are Proven and Their Usefulness Outlined." *Computer Aided Design* 21(8): 505 (1989).

Sederberg, T. W., and R. T. Farouki. "Approximation by Interval Bézier Curves." *IEEE Computer Graphics and Applications* 12(5): 87 (1992).

Sederberg, T. W., and R. N. Goldman. "Algebraic Geometry for Computer-Aided Geometric Design." *IEEE Computer Graphics and Applications* 6(6): 52–59 (1986).

Sederberg, T. W., and R. J. Meyers. "Loop Detection and Surface Patch Intersections." *Computer Aided Geometric Design* 5(2): 161–171 (1988).

Sederberg, T. W., and S. Parry. "Comparison of Three Curve Intersection Algorithms." *Computer Aided Design* 18(1): 58–64 (1986).

———. "Free-Form Deformation of Solid Geometric Models." *Computer Graphics* 20(4): 151 (1986).

Seidel, H.-P. "Computing B-Spline Control Points Using Polar Forms." *Computer Aided Design* 23(9): 634 (1991).

Serano, D., and D. C. Gossard. "Constraint Management: A Prerequisite to Conceptual Design." In *Knowledge Based Expert Systems in Engineering: Planning and Design,* eds. D. Sriram and R. A. Adey. United Kingdom: Computational Mechanics Publications, 1987.

Shaffer, C. A. "Dissertation Abstracts in Computer Graphics." *Computer Graphics* 27(2): 86 (1993).

Shah, J. J., and M. Mäntylä, eds. *Advances in Feature-Based Manufacturing.* New York: Elsevier, 1994.

Shamos, M. I. *Computational Geometry.* New York: Springer-Verlag, 1975.

———. "Geometric Complexity." *Proceedings of the 7th Annual ACM Symposium on Theory of Computing.* Washington, D.C.: ACM Press, 1975.

Shantz, M., and S. Chang. "Rendering Trimmed NURBS with Adaptive Forward Differencing." *Computer Graphics* 22(4): 189 (1988).

Shapiro, V., and D. L. Vossler. "Construction and Optimization of CSG Representations." *Computer Aided Design* 23(1): 4 (1991).

———. "Separation for Boundary to CSG Conversion." *ACM Transactions on Graphics* 12(1): 35 (1993).

Sheng, X., and B. E. Hirsch. "Triangulation of Trimmed Surfaces in Parametric Space." *Computer Aided Design* 24(8): 437 (1992).

Shephard, D. "A Two-Dimensional Interpolation Function for Irregularly Spaced Data." *Proceedings of the ACM National Conference.* Washington, D.C.: ACM Press, 1965.

Shoemake. K. "Animating Rotation with Quaternion Curves." *Computer Graphics* 19(3): 245 (1985).

Shpitalni, M., Y. Koren, and C. C. Lo. "Realtime Curve Interpolators." *Computer Aided Design* 26(11): 832 (1994).

Snepp, D. K., and R. C. Pomeroy, "A geometry system for aerodynamic design" (AIAA-87-2902), 1987.

Snyder, J. M. Generative Modeling for Computer Graphics and CAD, Academic Press, New York, 1992.

Sodhi, R., and J. U. Turner. "Towards Modeling of Assemblies for Product Design." *Computer Aided Design* 26(2): 85 (1994).

Solano, L., and P. Brunet. "Constructive Constraint-Based Model for Parametric CAD Systems." *Computer Aided Design* 26(8): 614 (1994).

Sridhar, N., R. Agrawal, and G. L. Kinzel. "Active Occurrence-Matrix-Based Approach to Design Decomposition." *Computer Aided Design* 25(8): 500 (1993).

Stanton, E. L. Untitled, informal notes on tricubic interpolation and modeling. 1976.

Stanton, E. L., and L. M. Crain. "PATCHES-III User's Manual." McDonnell Douglas Astronautics Company Report no. MDC G5538, Nov. 1974.

Stanton, E. L., L. M. Crain, and T. F. New. "A Parametric Cubic Modeling System for General Solids of Composite Material." *International Journal of Numerical Methods in Engineering* 11: 653–670 (1977).

Stone, M. C., and T. D. DeRose. "A Geometric Characterization of Parametric Cubic Curves." *ACM Transactions on Graphics* 8(3): 147 (1989).

Strasser, W., ed. *Theory and Practice of Geometric Modeling.* New York: Springer-Verlag, 1989.

Stroud, I. "Boundary Modeling with Special Representations." *Computer Aided Design* 26(7): 543 (1994).

Su, G., and D. Liu. "An Affine Invariant and Its Application in Computational Geometry." *Scientia Sinica,* ser. A, 24(3): 259 (1983).

———. *Computational Geometry—Curve and Surface Modeling.* New York: Academic Press, 1989.

Sugihara, K. "An Intersection Algorithm Based on Delaunay Triangulation." *IEEE Computer Graphics and Applications* 12(2): 59–67 (1992).

Supta, S., and J. U. Turner. "Variational Solid Modeling for Tolerance Analysis." *IEEE Computer Graphics and Applications* 13(3): 64 (1993).

Sutherland, I. E. "Sketchpad: A Man-Machine Graphical Communication System." *Proceedings of the SJCC* 23: 329–349 (1963).

Tamminen, M. "Efficient Octree Conversion by Connectivity Labeling." *Computer Graphics* 18(3): 43 (1984).

Terzopoulos, D., J. Platt, A. Barr, and K. Fleishcher. "Elastically Deformable Models." *Computer Graphics* 21(4): 205 (1987).

Tiller, W. "Rational B-Splines for Curves and Surface Representation." *IEEE Computer Graphics and Applications* 3(6): 61–69 (1983).

———. "Geometric Modeling Using Nonuniform Rational B-Splines: Mathematical Techniques." *SIGGRAPH Tutorial Notes, ACM.* Washington, D.C.: ACM Press, 1986.

———. "Knot-Removal Algorithms for NURBS Curves and Surfaces." *Computer Aided Design* 24(8): 445 (1992).

Tilove, R. B. "A Study of Geometric Set-Membership Classification." Technical Memo no. 30. Production Automation Project, University of Rochester, N.Y., Nov. 1977.

———. "Set-Membership Classification: A Unified Approach to Geometric Intersection Problems." *IEEE Transactions on Computers* C-29 (1980).

———. "Line/Polygon Classification: A Study of the Complexity of Geometric Computation." *IEEE Computer Graphics and Applications* 1(2): 75–78, 80–82, 84 (1981).

———. "Exploiting Spatial and Structural Locality in Geometric Modeling." Technical Memo no. 38. Production Automation Project, University of Rochester, Rochester, N.Y., 1981.

Tilove, R. B., and A. A. G. Requicha. "Closure of Boolean Operations on Geometric Entities." *Computer Aided Design* 12(5): 219–220 (1980).

Timmer, H. G. "Spline-Blended Interpolation of a Network of Curves." Douglas Aircraft Company Technical Memorandum C1-250-CAT-76-044. Aug. 1976.

———. "Analytical Background for Computation of Surface Intersections." Douglas Aircraft Company Technical Memorandum C1-250-CAT-77-036. Apr. 1977.

———. Informal notes. Undated.

Timmer, H. G., C. L. Arne, T. R. Stokes Jr., and H. H. Tang. "Ablation Aerodynamics for Slender Reentry Bodies." AFFDL-TR-70-27, 1, March 1970.

Timmer, H. G., and J. M. Stern. "Computation of Global Geometric Properties of Solid Objects." *Computer Aided Design* 12(6) (1980).

Toennies, K. D. "3D Modeling Using an Extended Cell Enumeration Representation." *Computer Graphics* 24(5): 13 (1990).

Toennies, K. D., J. K. Udupa, and G. T. Herman. "Registration of 3D Objects and Surfaces." *IEEE Computer Graphics and Applications* 10(3): 52–62 (1990).

Tokomasu, S., I. Harashima, and N. Nakajima. "Spanning a C^1 Surface over a Given Wireframe." *Computer Graphics Forum* 12(4): 211 (1993).

Toth, D. L. "On Ray Tracing Parametric Surfaces." *Computer Graphics* 19(3): 171 (1985).

Turk, G. "Re-tiling Polygonal Surfaces." *Computer Graphics* 26(2): 55 (1992).

Turner, J., and J. Rossignac. "Special Issue: Solid Modeling '93." Editorial. *Computer Aided Design* 26(6): 402 (1994).

van Overveld, C. W. A. M. "Family of Recursively Defined Curves, Related to the Cubic Bézier Curve." *Computer Aided Design* 22(9): 591 (1990).

Varady, T. "Survey and New Results in *n*-Sided Patch Generation." In *The Mathematics of Surfaces II*, ed. R. Martin. Oxford University Press, 1987.

Veltkamp, R. C. "Survey of Continuities of Curves and Surfaces." *Computer Graphics Forum* 11(2): 93 (1992).

Verroust, A., F. Schonek, and D. Roller. "Rule-Oriented Method for Parameterized Computer-Aided Design." *Computer Aided Design* 24(10): 531 (1992).

Vesprille, K. J. "Computer Aided Design Applications of the Rational B-Spline Approximation Form." PhD. thesis, Syracuse University, Syracuse, N.Y., 1975.

Vida, J., R. R. Martin, and T. Varady. "A Survey of Blending Methods That Use Parametric Surfaces." *Computer Aided Design* 26(5): 341 (1994).

Voelcker, H. B., and A. A. G. Requicha. "Geometric Modeling Research at the University of Rochester." Presented at the First Annual Conference of Computer Graphics in CAD/CAM Systems, MIT, Cambridge, Mass., Apr. 1979.

———. "Boundary-Evaluation Procedures for Objects Defined via Constructive Solid Geometry." Technical Memo no. 26. Production Automation Project, University of Rochester, Rochester, N.Y., 1980.

von Herzen, B. "Accurate Triangulations of Deformed, Intersecting Surfaces." *Computer Graphics* 21(4): 103 (1987).

Wang, C. "Shape Classification of the Parametric Cubic Curve and Parametric B-Spline Cubic Curve." *Computer Aided Design* 13(4): 199–296 (1981).

Warren, J. "Blending Algebraic Surfaces." *ACM Transactions on Graphics* 8(4): 263 (1989).

———. "Creating Multisided Rational Bézier Surfaces Using Base Points." *ACM Transactions on Graphics* 11(2): 127 (1992).

Watkins, M., and A. Worsey. "Degree Reduction for Bézier Curves." *Computer Aided Design* 20(7): 398–405 (1988).

Welch, W., A. Gleicher, and A. Wilkin. "Manipulating Surfaces Differentially." *Proceedings of Computer Graphics '91*. ACM 1991.

Welch, W., and A. Witkin. "Variational Surface Modeling." Proceedings of SIGGRAPH '92. *Computer Graphics* 26(2): 157 (1992).

Wesley, M. A., and G. Markowsky. "Generation of Solid Models from Two-Dimensional and Three-Dimensional Data." In *Solid Modeling by Computers: From Theory to Applications*. eds. M. S. Pickett and J. W. Boyse. New York: Plenum Press, 1984.

Wilf, I., and Y. Manor. "Quadric-Surface Intersection Curves: Shape and Structure." *Computer Aided Design* 25(10): 633 (1993).

Witkin, A., K. Fleischer, and A. Barr. "Energy Constraints on Parameterized Models." *Proceedings of SIGGRAPH '87*. Palo Alto, Calif.: Morgan Kaufmann, 1987.

Woodward, C. D. "Cross-sectional Design of B-Spline Surfaces." *Computer Graphics* 11(2): 193 (1987).

Woodwark, J. R. "Blends in Geometric Modeling." In *Mathematics of Surfaces II*, ed. R. R. Martin. Oxford University Press, 1989.

Yamaguchi, F. *Curves and Surfaces in Computer Aided Geometric Design*. New York: Springer-Verlag, 1988.

This page intentionally left blank

INDEX

A

Ab initio design, 3
Addition, vector, 421
Adjacency matrix, 313
Affine transformation(s), Bézier curve invariance under, 87
Algebraic coefficients, 40
Algebraic geometry, 2
Ancestor, 315, 382
Angle excess, 287
Animation, 408
Approximation, 11
APT language, 6
Arc length, 20
Area, triangle, 424
Atlas, 296, 297
Auxiliary curve, 176

B

Barsky, B. A., 10
Basis, 40, 421
Basis function(s):
  Bernstein polynomials, 83, 88
  Bézier, 83
  B-Spline, 113, 126
  characteristics of, 82
  conversion between, 85, 137
  even-degree polynomials, 45
  Hermite, 41, 42–47
  Hermite basis transformation matrix, 48
  odd-degree polynomials, 45
  orthogonal, 45
  symmetry between pairs of, 45
  transformation matrix, 48
Basis vectors, 417, 421
Beads, 376, 390
Bernstein polynomials:
  Bézier curves, 83, 88
  partition-of-unity property, 90
Beta spline curves:
  Barsky, B. A., 10
  bias, 34, 142
  cubic, 142
  tension, 34, 142
Bézier curve(s):
  affine transformations, invariance under, 87

Bézier curve(s) *(Continued)*:
application to geometric modeling, 33, 81
basis functions, 83, 93
basis transformation matrix, 84
Bernstein polynomials, 83, 88
Bézier-Hermite conversion, 85
binomial coefficient, 83
characteristic polygon, 82
circle, 112
closed, 93
composite, 105
conic representations, 111
construction of, 88
continuity, 105
control points, 82, 89
control polygon, 82, 90
convex hull, 90
cubic, 84
curvature at endpoints, 107
degree elevation, 95
degree reduction, 97
ellipse, 112
G^2 continuity, 107
higher-order, 107
hyperbola, 112
matrix form, 84
modifying, 93
parabola, 112
points on, 81
properties of, 82
rational, 108
recursive subdivision, 102
subdividing, 97, 102
truncating, 97
UNISURF system, 81
weights, 110
Bézier, P., 7, 10, 81
Bézier surface:
basis function transformations, 222
bicubic patch, 218
characteristic polyhedron, 218
composite, 224
continuity, 224
control points, 217, 220
converting between Bézier and Hermite, 222
degree elevation, 223
isoparametric curve on, 219
matrix form, 217, 222
points on, 217
rational Bézier patch, 225
tensor product, 217
Binary tree, 316, 342, 346, 350
Binomial coefficient, 83
Blend surfaces, 213, 215
Boolean algebra, 2, 321
Boolean model(s):
Boolean algebra, 321
boundary of closed set, 321
boundary, valid, 328
closed set, 319
constructive solid geometry, 348
difference, 320
dimensional homogeneity, 323
directed surface, 342
empty set, 319
graph-based model, 310
half-space, 342–343
half-spaces, intersection of, 337
interior of a closed set, 321
intersection, 320
null set, 319
open set, 319
operators, 319
primitives, 319
procedural model, 341
regularized set operators, 323
set-membership classification, 332
sets, 319
sets, operations on, 319
sets, properties of, 319
subset, 319
unevaluated model, 341

union, 319
universal set, 314
Venn diagram, 320
Boolean operators:
difference, 320, 321
intersection, 320, 321
union, 319, 321
de Boor, C., 10
Boundary evaluator, 364
Boundary model(s):
boundary representations (b-reps), 356, 360
data structure, 362
generalized concept of a boundary, 358
region, 358
topology, 363
validity, 358, 363
Boundary, valid, 358, 363
B-Spline curve(s):
application to geometric modeling, 10, 33, 113
basis functions, 113, 126
basis transformations, 124
circle, 141
closed, 130
conics, 139
conic shape factor, 141
continuity, 134
continuity conditions, summary, 136
control points, 114
control points, multiple, 115
conversion between basis functions, 137
cubic basis functions, 128
degree of basis function polynomial, 114
knot spacing, nonuniform, 115, 117
knot spacing, uniform, 115
knot values, 114
knot vector, 114
matrix form, 124
nonperiodic, 113
nonuniform basis functions, 113, 139
nonuniform rational, 139
NURBS, 139
periodic, 127
points on, 114
polygon, 121
quadratic basis functions, 128
recursive definition of basis, 114
uniform basis functions, 126
weights, 139
B-Spline surface:
characteristic polyhedron, 227
closed, 228
control points, 227
matrix form, 228
nonperiodic, 227
NURBS, 235
open, 228
points on, 227
quadric, 232
tensor product, 227
Building-block geometry, 348

C

Cardinal splines, 196, 199
de Casteljau, P., 7, 88
Cayley, A., 431
Cell decomposition, 367
Cells, parametric network of, 254
Characteristic equation, 435
Characteristic polygon, 82
Circle, 70, 141
Circuit, graph, 313
Circular arc, 70
Class evaluators, 384
Classification of points, 332
Closed curved surfaces, 305
Closed paths, 282
Closed path theorem, 286

CNC machining, 411
Cofactors, of determinants, 437
Combinatorial structure, 363
Completeness, logical, 411
Composite:
  curves, 74, 105
  solids, 252
  surfaces, 192, 224
Compound plate layout, 408
Computational geometry, 2
Computer-aided design (CAD), 1, 6
Computer-aided geometric design (CAGD), 2
Computer-aided manufacturing (CAM), 1, 5, 6
Computer graphics, 1
Cone, 207
Conic curve(s):
  characteristics, 32
  circle, 70
  conic shape factor, 141
  construction of, traditional, 64
  Hermite approximations of, 64
  hyperbola, 69
  implicit equation of, 31
  NURBS curve, 139
  parabola, 68
  surface, 64
Conjugate net, 159
Connectivity, 312
Connectivity matrix, 312
Connectivity number, 296
Constructive dependency, 375
Constructive solid geometry (CSG):
  binary tree, ordered, 346, 350
  Boolean models, 320
  Boolean operators, 319
  boundary evaluator, 364
  boundary representation (b-reps), 352
  branches, binary tree, 348
  cell decomposition, 349
  definition, 348
  gluing operation, 349
  history, 2, 348
  leaf nodes, 350
  neighborhood model, 355
  primitives, 348, 350
  procedural representation, 351
  Requicha, A. A. G., 10
  root node, 350
  Voelker, H. B., 10
Continuity:
  Bézier curves, 105
  B-Spline curves, 134
  composite curves, 74, 105
  first order, 77
  geometric, 77
  Hermite curves, 74, 77
  parametric, 77
  second order, 77, 107
Contours, 399
Controlled deformation, 275
Control points:
  adding, 89
  Bézier curve, 89
  Bézier surface, 217, 220
  B-Spline curve, 114
  B-Spline surface, 114
  characteristic polygon, 82, 89
Conventions and notation, 16
Convex hull, 90
Coons, S. A., 7, 10, 81
Coordinates, homogeneous, 108, 439, 441
Cox, M. G., 10
Crain, L. M., 11
Cross plots, 28
Cubic curves, 32, 39
Curvature, 20, 301
Curvature density, 288
Curvature, total, 288
Curve(s):
  auxiliary, 176
  Beta-Spline, 142
  Bézier, 32, 81

B-Spline, 32
circular arc, 70
composite, 13, 74, 105
conic, 31, 64
conic Hermite, 64
continuity, 74, 77, 105
conversion between implicit and parametric form, 22
cubic, parametric, 25, 32, 39
curvature, 20, 107
cusps, 53
decomposition of, 28
direct point solution, 34
explicit equation of, 22
fitting, 12
forward difference method for points on, 36
geometric continuity, 77
Hermite, 32, 39
implicit equation of, 22
interpolation, 33
intrinsic equations of, 19
intrinsic properties, 19
isoparametric on a surface, 160
loops, 53
monomial basis, 48
natural equation of, 20
nets on surfaces, 159, 181
NURBS, 9, 139
on surfaces, 159, 161
orthogonal net, 159
parabola, 68
parametric continuity, 77
parametric equations of, 23
parameterization, reversing, 37
points on, computing, 34
position-direction, 268
power basis, 40
rational Bézier, 108
segment, 26
subdividing, 54, 97
subdivision, quick, 59
symmetries of, 22
torsion, 20
truncating, 54, 97
Cusps, 53
Cylindrical surface, 206

D

Decomposition, cell, 367
Deformation(s):
axial, 276
bivariate, 277
controlled, 275
shapes, 284
surfaces, deformable, 279
topology of, 284
trivariate, 277
Degree elevation, 95, 223
Degree reduction, 97
Dependency, constructive, 375
Dependency, interobject, 379
Descendant, 315, 382
Design optimization, 407
Determinants, 436
Difference operator, 320
Digraph, 382
Dilation (scaling), 440, 462, 464, 471
Directed graph, 313, 382
Directed surface, 343
Direction cosines, 419
Direction numbers, 419
Direct-point solution, 34
Discretization, surface and volume, 405
Discretization, topological invariant, 405
Disk, topological, 286
Durable properties, 375

E

Eigenvalues, 436
Eigenvectors, 436
Einstein summation convention, 40
Ellipse, 32

Ellipsoid, parametric, 152
Embedded geometry, 161, 376
Entity, 376, 378
Entity classes and subclasses, 379
Eshleman, A. E., xi, 7
Euler angles, 450
Euler characteristic, 305
Euler operators, 306
Euler-Poincaré formula, 295
Euler's formula, 294
Extrinsic properties, 19
Extruded sweep, 265

F

Ferguson, J. C., 7, 10, 81
Finite element model and analysis, 4, 405
Forrest, A. R., 7
Forward difference method, 36
Four-point interpolation, 61
Frames, 401
Free vectors, 417

G

Gauss-Bonnet theorem, 306
Gaussian quadrature, 16
General sweep surface, 265, 267
Geometric coefficients, 41
Geometric continuity, 77
Geometric modeling:
  ab initio design, 3
  geometric metaphor, 2
  history, 6
  polygonal schemes, 8, 9
  rendering, 3
  representation, 3
  visualization, 3
  What is it?, 1
  wireframe, 8
Gibbs, J. W., 419
Glide rotation, 470
Gluing operation, 349
Gordon, W. J., 10
Graph-based models:
  adjacency matrix, 313
  ancestor, 315
  ancestor, proper, 315
  binary tree, 316
  branches, 315
  circuit, 313
  connected, 313
  connectivity matrix, 312
  descendant, 315
  directed graph, 313
  in degree, 313
  inorder traversal, 315
  leaf node, 315
  node, 315
  out degree, 313
  postorder traversal, 316
  preorder traversal, 316
  relational geometric synthesis, 373
  root, 315
  spanning tree, 313
  subgraph, 313
  traversal, recursive, 316
  traversal, tree, 316
  tree, 313
  wireframe model, 318
Graph, directed, 313
Graphs, 313, 382, 400
Group technology, 263

H

Half-spaces, 166
Hamilton, W. R., 419, 456
Handles, spheres with, 289
Hermite, C., 33
Hermite basis transformation matrix, 48
Hermite curves:
  algebraic coefficients, 40, 49
  algebraic form, 40

application to geometric modeling, 33, 39
basis functions, 39, 41
basis transformation matrix, 48
Bézier-Hermite conversion, 85
circular arc, approximation of, 70
composite, 74
conic, 64
continuity, 74, 77
cubic, 39, 61
cusps, 53
directionality, parametric, 41
endpoints, 41
four-point interpolation, 61
geometric coefficients, 41
geometric continuity, 77
geometric form, 41, 49
hyperbola, approximation, 69
matrix form, 47
parabola, 68
parametric continuity, 77
quintic, 46
relationship between algebraic and geometric forms, 49
reparameterizing, 54
subdividing, 54
tangent vectors, 50–53
three-point interpolation, 60
truncating, 54
unit tangent vector, 53
Hermite solid:
algebraic form, 241
basis transformation matrix, 241
cells, 254
composite solids, 252
continuity, 252
curves and surfaces in, 253
generalized notation scheme, 257
geometric form, 241
higher-dimensional elements, 257
isoparametric curves in, 253
nonisoparametric curves in, 253
parameter space of, 252
parametric cells, 254
tangent vectors, 250
trimmed boundaries, 256
twist vectors, 250
Hermite surface, bicubic:
algebraic coefficients, 169
algebraic form, 169
auxiliary curve, 176
basis functions, 178, 179
basis function transformation matrix, 170
blend surfaces, 213
boundaries, 172, 180
composite, 192
continuity, 192
converting algebraic and geometric forms, 179
curve net, 180
cylindrical surface, 206
degenerate patches, 210
geometric form, 169, 179
matrix notation for, 170
normals, 182
orthogonal net, 174
plane patch, 203
reparameterization of a patch, 186
ruled surface, 207
sixteen-point form, 184
spline interpolation, 196
subdividing, 191
tangent vectors, 180
tensor product, 169, 179
truncating, 191
twist vectors, 173, 180
unit normal, 182
History, 6
Homogeneous coordinates, 108, 439, 441
Homotopy, regular, 285
Horner's rule, 34
Hyperbola, 32, 69
Hyperpatch, 237

I

Identity matrix, 432
Implicitization of parametric form, 22, 144
Inner product, 422
Inorder traversal, 315
Instances, 260–261
Interobject dependencies, 379
Interpolation, 7, 12
Intersection, 320
Intrinsic equation of curves, 19
Invariant property, 439
Inverse matrix, 435
Inverse-point solution, 37
Inversion, 459
Isometries, 440
Isoparametric curves, 160

J

Joins between surfaces, 376
Jordan, C., 291
Jordan curve theorem, 286

K

Kinematic analysis, 4
Klein bottle, 300
Knot, 291
Knot group, 293
Knot spacing, 115, 117
Knot values, 114, 291
Knot vector, 114
Kronecker delta, 422, 432

L

Leaf node, 315
Length, vector, 419
Letcher, J. S., v, xi, 10, 373
Line, vector equation of, 425
Linear transformations, 439
Logical completeness, 411
Logical model, 382
Loops and cusps in curves, 53

M

Magnets, 376, 391
Magnitude, vector, 419
Matrices:
  addition, 433
  adjacency, 313
  antisymmetric matrix, 432
  application to geometric modeling, 12
  Bézier basis transformation, 84
  Bézier curve, 84
  B-Spline basis transformation, 124
  B-Spline curve, 124
  characteristic equation, 435
  cofactor, 437
  column matrix, 431
  conformable, 433
  connectivity, 312
  determinants, 436
  diagonal matrix, 432
  eigenvalues, 436
  eigenvectors, 436
  Hermite basis transformation, 48
  identity matrix, 432
  inversion of, 435
  Kronecker delta, 432
  matrix algebra, 48
  multiplication of, 433
  null matrix, 432, 434
  order of, 431
  partitioned, 434
  postmultiply, 433
  premultiply, 433
  properties, summary of, 436
  rank of, 421
  row matrix, 431

scalar matrix, 432
scalar multiplication, 433
scalar product as a matrix product, 435
skew symmetric matrix, 432
square matrix, 431
symmetric matrix, 432
transpose of, 432
unit matrix, 432
vector product as a matrix product, 435
Matrix form:
Bézier curves, 84
Bézier surface, 217
B-Spline curve, 124
B-Spline surface, 228
Hermite curve, 47
Hermite surface, 170
Tricubic solid, 241
Merging algorithm, 364
Meriwether, H. D., xi, 7, 176
Mobius strip, 299
Model evaluation, recursive, 383, 386
Model(s) and modeling:
Boolean, 318, 341
boundary representation (b-rep), 281, 356
CSG, 3, 281, 348
evaluation, 383, 386
finite element, 4, 405
graph-based, 310
logical, 411
neighborhood, 355
parametric, 377, 404
procedural, 341, 351
relational geometry, 373
solid, 4
space-partitioning, 367
subtractive, 6
unevaluated, 341
wireframe, 8, 318
Model space, 28
Model structure, relational, 378, 382
Modularity in relational models, 412
MultiSurf, v, 378

N

Natural equations of a curve, 20
Neighborhood model, 355
Nets, curve:
conjugate, 159
covariant, 160
isoparametric, 160
orthogonal, 160
parametric, 159
Newton-Cotes quadrature formulas, 14
Newton-Raphson method for finding roots, 13
Node, 315
Node, leaf, 315
Nonlinear transformations, 440
Nonorientable surface, 298
Norm, 419
Normals, surface, 182,
Notation and conventions, 16
Numerical analysis, 13
NURBS curve:
circle, 141
conic representation, 139
conic shape factor, 139
equation of, 139
shape modification, 140
weights, 139
NURBS form, deficiencies of, 157
NURBS surface(s), 235

O

Object, 376, 378
Object evaluation, 384
Octrees, 369
Optimization, design, 407

Orientable, 298
Orientation, 298
Orientation-preserving, 298
Orientation-reversing, 298
Orthogonal net of curves, 159
Outline surface, 268

P

Parabola, 68
Parameterization, default, 392
Parameterization, effects of, 37
Parameterization, natural, 392
Parameterized shapes, 260, 377, 404
Parameter space, 28, 153
Parameter transformations, 54
Parametric continuity, 77
Parametric cubic curve, 39
Parametric derivatives, computation of, 27
Parametric equations, advantages of in modeling, 23, 29
Parametric modeling, 377, 404
Parametric variable, normalizing domain of, 25
Parity, transition, 301
Partition of unity, 90
Patch, 149
Path, self-intersecting, 282
Path, simple, 282
PD curve, 268
Perspective transformation, 111
Piecewise flat surface, 293
Piecewise polynomials, 13
Plane, 203, 427
Point(s):
  classification, 165, 332
  forward difference, 36
  importance of in modeling, 24
  inverse-point solution, 34
  on curves, 34
  on surfaces, 158
Polygons:
  control, 82, 90
  star, 282
Polyhedron:
  convex, 293
  nonsimple, 293, 295
  regular, 293
  simple, 293
Polynomial interpolation, 13
Position-direction curve, 268
Position vector, 417
Postorder traversal, 316
Power basis, for curves, 40
Preorder traversal, 316
Primitives, 261, 319
Procedural model, 341, 351
Profile curve, 271
Projective plane, 108, 300

Q

Quadrature:
  Gaussian quadrature, 16
  Newton-Cotes quadrature formulas, 14
Quadric surface(s):
  classification of, 148
  equation of, 144–145
  surface of revolution, 149
Quadtrees, 369
Quaternions, 456

R

Radius vector, 417
Rational forms:
  Bézier curves, 108
  Bézier surface, 225
  homogeneous coordinates, 108
  invariance under perspective transformation, 111
  polynomial, 108

projective space, 105
weights, 110, 139
Recursive model evaluation, 386
Recursive subdivision, 102
Reflection, 457, 470
Regular polyhedra, 293
Regularized set operators, 323
Relational entities, 376, 378
Relational geometric synthesis (RGS), 318, 373
Relational geometry (RG):
ancestors, 382
animation, 408
beads, 376, 390
boundary representation, 414
class evaluators, 384
CNC machining, 411
compound plate layout, 408
constructive dependency, 375
contours, 399
curves, 392
dependency, constructive, 375
descendants, 382
design optimization, 407
developable surface, 408
digraph, 382
directed graph, 375, 382
discretization, surface and volume, 405
discretization, topological invariant, 405
durable properties, 375
embedded geometry, 376
entities, 376, 378
entity classes and subclasses, 379
finite element model and analysis, 405
frames, 401
graphs, 400
history, 10
joins between surfaces, 376
interobject dependencies, 379
logical completeness, 411
magnets, 376, 391
model evaluation, 383, 386
model structure, relational, 378, 382
modularity, 412
MultiSurf, v, 378
object evaluation, 384
objects, 376, 378
parameterization, default, 392
parameterization, natural, 392
parametric modeling, 377, 404
planes, 400
points, three-dimensional, 389
program structure, 384
relational entities, 379, 388
rings, 390
snakes, 376, 394
solids, 398
surfaces, 395
synthesis, model, 374
trimmed surface, 414
Relational model structure, 378, 382
Rendering, 4
Reparameterization, 29, 54, 100, 186
Representation, 3
Requicha, A. A. G., 10
Revolution, surface of, 152
Riesenfeld, R. F., 10
Rigid-body transformations, 439
Rings, 390
Root node, 315
Roots, finding polynomial:
Newton-Raphson method, 13
Newton's method, 13
sliding tangent method, 13
Ross, D. T., 6
Rotation number, 282
Rotational sweep, 265, 271
Rotations:
about an arbitrary axis, 453
about an arbitrary point, 446
about the origin, 444

Rotations *(Continued)*:
coordinate system, 447, 449
curves and surfaces, 446
equivalent about principal axes, 451
Euler angle(s), 450
in the plane, 444, 468
in space about principal axes, 447
quaternions, 456
successive, 445, 448
Rubber sheet deformation, 284
Ruled surface, 207

S

SAGE system, 6
Scalar product, 422
Scaling (dilation), 440, 462
Schoenberg, I., 10
Set, closed, 321
Set, open, 321
Set-membership classification, 332
Set theory, 3, 319
Shear, 440, 462, 466
Signature of quadratic form, 147
Simple-closed-path theorem, 282
Simple polyhedra, 293
Sketchpad system, 6
Snakes, 376, 394
Solid modeling, 2, 4, 10, 238
Solids:
algebraic form, 241
boundary elements, 238, 242
bounding faces, 239
composite solids, 252
continuity, 252
controlled deformation, 275
curves in, 253
embedded geometry, 253
extrusion, 265
generalized notation, 257
geometric form, 241
group technology, 263
Hermite basis transformation matrix, 244
hyperpatch,
instances, 260–261
isoparametric surfaces, 238
matrix form, 241, 244
parameterized, 260
parameter space of, 252
parametric cell, 254
parametric solid, 237
sweeps, 264
tangent vectors, 250
tricubic Hermite solid, 239, 240
trimmed boundaries, 256
twist vectors, 250
Space-partitioning model, 367
Spanning tree, 313
Spatial occupancy enumeration, 367
Sphere, 151
Sphere with handles, 289
Spline interpolation, 196, 199
Stanton, E. L., 11
Star polygons, 282
Subdividing:
Bézier curves, 97
equal segments, 58
geometric construction, 104
Hermite bicubic patch, 191
Hermite curves, 54
quick subdivision, 59
recursive, 102
reparameterization, 54, 100
Subgraph, 313
Surface(s):
B-Spline, 156
Bézier, 156
blend, 213, 215
composite, 192
conjugate nets on, 159
curve nets on, 159, 181
curves embedded on, 161
cylindrical, 146, 206

degenerate patches, 154
directed, 343
ellipsoid, 152
embedded curves on, 161
explicit equation of, 144
fitting, 12
Hermite, 156
hyperbolic paraboloid, 208
implicit equation of, 144
intuitive notion of, 143
inverse-point solution, 158
irregular boundaries, 161
isoparametric curves on, 160
nonorientable, 298
normal vector, 149, 182
NURBS, 235
orthogonal nets on, 159
parametric equations of, 149
parametric nets on, 159, 174
patch, 149, 155
piecewise flat, 293
plane, 203, 427
point classification, 165
point evaluation, 158
points on a surface, 158
quadric, 145
rational Bézier, 225
reparameterization of a patch, 186
ruled, 207
sphere, 151
surface of revolution, parametric, 152
tangent vectors, 149
tensor product, 143, 157
trimmed patch, 164
twist vectors, 149, 157
Sutherland, I., 6
Sweep shapes:
extruded, 265
general, 265, 267
position-direction (PD) curve, 268
profile curve, 271
rotational, 265, 271
translational, 265, 267
Synthesis, model, 374

T

Tangent vector(s):
affect on curve shape, 50–53
Bézier curves, 89
cusps, 53
Hermite curves, 50
length, 50
loops, 53
magnitude, 50
multiple of chord length, 53
parameterization, dependence on, 50
Tensor(s):
application of, to geometric modeling, 3
Einstein summation convention, 40
scalar product in tensor notation, 422
tensor product surface, 143, 157, 227
Tensor product surface, 169, 217, 227
Three-point interpolation, 60
Topology:
angle excess, 287
atlas, topological, 296, 297
closed path, 282
connectivity, 312
connectivity number, 296
curvature, 288, 301
curvature density, 288
deformations, 284
discretizations, invariant, 405
disk, topological, 286
Euler characteristic, 305
Euler operators, 306
Euler-Poincaré formula, 295

Topology *(Continued)*:
Euler's formula, 294
Gauss-Bonnet theorem, 306
handles, 289
homotopy, regular, 285
in modeling, 3
invariant properties, 284
isotropy, ambient, 285
Jordan curve theorem, 286
Klein bottle, 300
knots, 291
Mobius strip, 299
orientable, 298
orientation, 298
orientation-preserving, 298
orientation-reversing, 298
parity, transition, 301
piecewise flat surfaces, 293
Poincaré's generalization of Euler's formula, 295
polyhedron, 293
polyhedron, nonsimple, 293, 295
projective plane, 300
regular polyhedra, 293
rotation number, 282
rubber sheet deformation of the plane, 284
self-intersecting paths, 282
simple paths, 282
simple polyhedra, 293
sphere, 289
structure, topological, 363
surfaces, closed curved, 305
surfaces, nonorientable, 298
torus, 289
Torsion, 20
Torus, 289
Transformations, geometric:
affine transformations, 440
Cartesian and homogeneous coordinates, 441
coordinate system rotation, 447, 449
dilation (scaling), 440, 462
dilation, anisotropic, 440, 464, 471
dilation, isotropic, 440, 462
Euler angles, 450
geometric, 439
glide rotation, 470
homogeneous coordinates, 439, 441
invariant property, 439
inversion, 459
isometries, 440
linear, 439
multiple sequential, 467
nonlinear, 440
projections, 440
reflection, 457, 470
rotation (see also Rotations), 444, 468
scaling (dilation), 440, 462
shear, 440, 462, 466
sweeps, 265
topological, 440
translation, 441, 468
Transition parity, 301
Translation, 441, 468
Translational sweep, 265, 267
Transpose of a matrix, 432
Traversals, tree, 316
Tree, 313
Tree, binary, 316
Tree, spanning, 313
Tricubic Hermite solid, 240
Trimmed patch, 164
Triple scalar product, 424
Triple vector product, 425
Truncating:
Bézier curves, 97
control point transformations, 97
cubic Bézier curves, 97
Hermite bicubic patch, 191
Hermite curve, 54
reparameterization, 54
shape invariance, 54
Twist vector, 173, 180

U

Union, 319, 321
UNISURF system, 81
Unit vector, 420

V

Vector product, 423
Vector(s):
  addition, 421
  application to geometric modeling, 3, 11
  area of a triangle, 424
  basis, 421
  basis vectors, 417, 421
  direction cosines, 419
  direction numbers, 419
  Einstein summation convention, 419
  equations, solution of, 429
  free vectors, 417
  inner product, 422
  Kronecker delta, 422
  length, 419
  line, equation of, 425
  magnitude, 419
  norm, 419
  plane, equation of, 427
  plane, normal form, 428
  position vector, 417
  projection, 425
  properties of, summary, 429
  radius vector, 417
  right-hand rule, 423
  scalar product, 422
  tensor form, 418, 422, 424
  triple scalar product, 424
  triple vector product, 425
  unit vector, 420
  vector product, 423
Venn diagram, 320
Voelker, H. B., 10

W

Weights, 110, 139
Wireframe model, 8, 318