PARAMETRIC GEOMETRY CREATION METHODOLOGY AND UTILITY FOR THE STARS CFD ANALYSIS PACKAGE

By

ROBERT JAMES FISCHER

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

2004

Submitted to the Faculty
of the Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2007

PARAMETRIC GEOMETRY CREATION

METHODOLOGY AND UTILITY FOR

THE STARS CFD ANALYSIS

PACKAGE

Thesis Approved:

Dr. Andrew S. Arena, Jr.
Thesis Advisor

Dr. Ronald D. Delahoussaye

Dr. Jamey D. Jacob

Dr. A. Gordon Emslie
Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to thank my parents and family for all of their support during my years of undergraduate and graduate study. They have always encouraged me to pursue my educational goals and interests and have inspired me to achieve my ambitions. Their support has been a vital component of my academic progress.

I am also very appreciative of the support and guidance provided by my advisor, Dr. Arena. He has been one of the best and most influential professors that I have encountered and has provided much inspiration and encouragement to many students, including myself. I am thankful for the opportunity that he has given me to pursue research in the CASE Lab.

I would like to thank those who have worked in the CASE Lab, both previously and presently, for providing me with the knowledge and tools to complete my research objectives. I would specifically like to thank Charles O'Neill and Nic Moffitt. They helped to train me in the use of the CASE Lab analysis tools and have provided much insight and support during my time in the CASE Lab.

Finally, I would like to thank the NASA Space Grant for providing funding in support of this research effort.

TABLE OF CONTENTS

Chapter Page

1 INTRODUCTION 1

1.1 Background 1
1.2 STARS CFD Solution Procedure Overview 2
1.3 Objective 4

2 LITERATURE REVIEW 7

2.1 Geometry Utility Approach 7
2.2 Selection of Graphics Data Exchange Format 9
2.2.1 IGES 9
2.2.2 STEP 11
2.2.3 Additional Graphics Data Exchange Formats 15
2.3 Surface Tessellation 16
2.4 Final Selection of Exchange Format 20

3 METHODOLOGY 22

3.1 Implicit and Piecewise Implicit Surface Creation 22
3.2 Parametric Curve and Surface Creation 26
3.2.1 Monomial Interpolation Method 28
3.2.2 Bezier Form for Curves and Surfaces 33
3.2.3 B-Splines (Basis-Splines) 43
3.2.4 Rational Curve Specification 53
3.2.5 Tensor Product Surfaces (Non-Uniform Rational B-Spline Surfaces) 58
3.2.6 Three and Five-Sided Surface Patches 63
3.3 Converter Program Structure 65
3.3.1 Geometry Import 66
3.3.2 Entity Processing and STARS File Creation 71
3.3.3 User Input and Graphical User Interface 73

4 RESULTS AND EXAMPLES 76

4.1 Verification Results 76
4.2 Test Cases 78
4.2.1 Test Case 1: YF-22 78

Chapter Page

4.2.2 Test Case 2: OSU Design/Build/Fly Aircraft 80
4.2.3 Test Case 3: SBC-UAV 83
4.2.4 Test Case 4: GHV 84

5 SUMMARY AND FUTURE DEVELOPMENT 85

5.1 Summary 85
5.2 Future Development 86

BIBLIOGRAPHY 87

APPENDIX A: IGES GLOBAL PARAMETERS 90

APPENDIX B: EXAMPLE IGES ENTITY DEFINITION 92

LIST OF FIGURES

Figure Page

Figure 1: Surface File (sur file) Structure 2
Figure 2: Background File (bac file) Structure 4
Figure 3: Example STEP Application Protocols (APs) (figure from STEP Application Handbook [2006]) 13
Figure 4: Example STEP file structure (figure from STEP Application Handbook [2006]) 15
Figure 5: Example faceted surface representations of 3-D objects 17
Figure 6. Example triangular elements subdividing a surface patch (figure from Hagan [1992]) 24
Figure 7: Example contour fit through linear interpolation of element vertices 25
Figure 8: Example parametric surface patch 28
Figure 9: Example of the effects of slight coefficient perturbations on monomial (light gray) and Bezier (black) formats [Farin, 1993] 31
Figure 10: Exaggerated example of the effects of round-off error on surface patch continuity (figure from Farin [1993]) 32
Figure 11: Example discontinuity due to terminal/start point adjustment 33
Figure 12: Example second degree curve and contributing basis functions 36
Figure 13: Example third degree curve (with movement of terminal control point shown on the right) 36
Figure 14: Example application of de Casteljau method for a seventh degree Bernstein curve 38
Figure 15: Example triangular array of successive control points for de Casteljau method 39
Figure 16: Example knot sequence with piecewise Bezier segments (figure from Farin [1993]) 41
Figure 17: Piecewise Bezier curve with C^1 continuity 42
Figure 18: Effect of increasing knot multiplicity on an individual basis function (left plot is with complete degree 1 multiplicity, right plot is with a multiplicity of two for t = 4) 47
Figure 19: Example multiplicity effects for second degree basis functions [Piegl, 1995] 48
Figure 20: Example B-spline curves and basis functions 50
Figure 21: Coincident internal de Boor points in B-spline 51
Figure 22: Effects of B-spline degree elevation 52
Figure 23: De Boor algorithm for B-splines (figure from Hoschek [1993]) 53
Figure 24: Projection mapping of a point in E^3 56

Figure Page

Figure 25: Example weighted representation of conic entity from homogeneous projection 56
Figure 26: Example control point weight modification (at point P_2)....57
Figure 27: Tensor product surface patch 59
Figure 28: Tensor product surface patch 61
Figure 29: Effects of surface degree elevation [Piegl, 1995] (biquadratic surface p = 3 on left; biquartic surface p = 4 on right)....62
Figure 30: Example NURBS surfaces 62
Figure 31: Triangular surface patch [Hoschek, 1993] 64
Figure 32: Five-sided surface patch 65
Figure 33: General converter structure 66
Figure 34: IGES file structure....67
Figure 35: Converter User Interface (utilizing GLUI [Rademacher, 1999]) 74
Figure 36: Line Number Display (left) and Surface Spline Display (right) 75
Figure 37: Source Display 75
Figure 38: Example Surface Used for Verification 77
Figure 39: Example Surface Error Plot (shaded areas represent error variation across the surface)....77
Figure 40: Pro/Engineer IGES YF-22....78
Figure 41: YF-22 Surface Grid 79
Figure 42: Mach Distribution at M = 0.3 79
Figure 43: 2007 Orange Team surface mesh 80
Figure 44: Example 2007 Orange Team Mach solution plot....81
Figure 45: Original 2007 Black Team surface mesh 81
Figure 46: 2007 Black Team surface mesh (low wing placement) 82
Figure 47: 2007 Black Team surface mesh (forward wing, reduced canard span)....82
Figure 48: 2007 Black Team Mach solution plot 83
Figure 49: SBC-UAV surface mesh 83
Figure 50: GHV (Model 1) surface mesh 84
Figure 51: GHV (Model 2) surface mesh 84

CHAPTER 1

INTRODUCTION

1.1 Background

Computational Fluid Dynamics (CFD) analysis tools offer an important and vital design component for many aerospace applications. As a result of the implementation of modern CFD tools and techniques, many important flight characteristics of new and experimental aircraft may be determined and examined early in the design process. The use of CFD methodology has resulted in faster, more cost efficient design evaluation and a safer, more accurate design process. Many flight characteristics that previously required expensive and time-consuming experimental testing can now be determined through the use of CFD. The use of CFD software also enables a designer to easily make and analyze changes to a particular aircraft component or configuration and thus iterate through many design ranges. Such a procedure may be prohibitive when relying solely on physical testing.

Different CFD software applications have been developed and implemented over the years, with differing design goals and solution methodologies for each. The STARS analysis package is a design and analysis tool which incorporates aeroelastic, structural, and CFD applications. It was developed at NASA's Dryden Flight Research Center. One unique element of the STARS package is the non-inertial flow solver developed by Cowan [2003]. This application, Euler3D, can solve compressible, inviscid flow problems through the use of the unsteady Euler equations.

1.2 STARS CFD Solution Procedure Overview

The use of Euler3D for test case simulation is a relatively straightforward process. Several files must first be generated by the user in order to run a steady or unsteady flow simulation. Some of these files include a surface geometry file, background file, boundary conditions file, flow properties file, and a dynamics file (if needed).

The first step in the process is the creation of a surface mesh for all model geometry entities. This mesh forms the defining basis for the volume grid generated later. The surface mesh generator requires data from a geometry file ("sur" file) and a background file ("bac" file). The surface file contains all of the information needed to describe the test case geometry (i.e. surfaces, boundary curves, and surface normal directions). An example of the basic content and structure of the sur file is given in Figure 1.

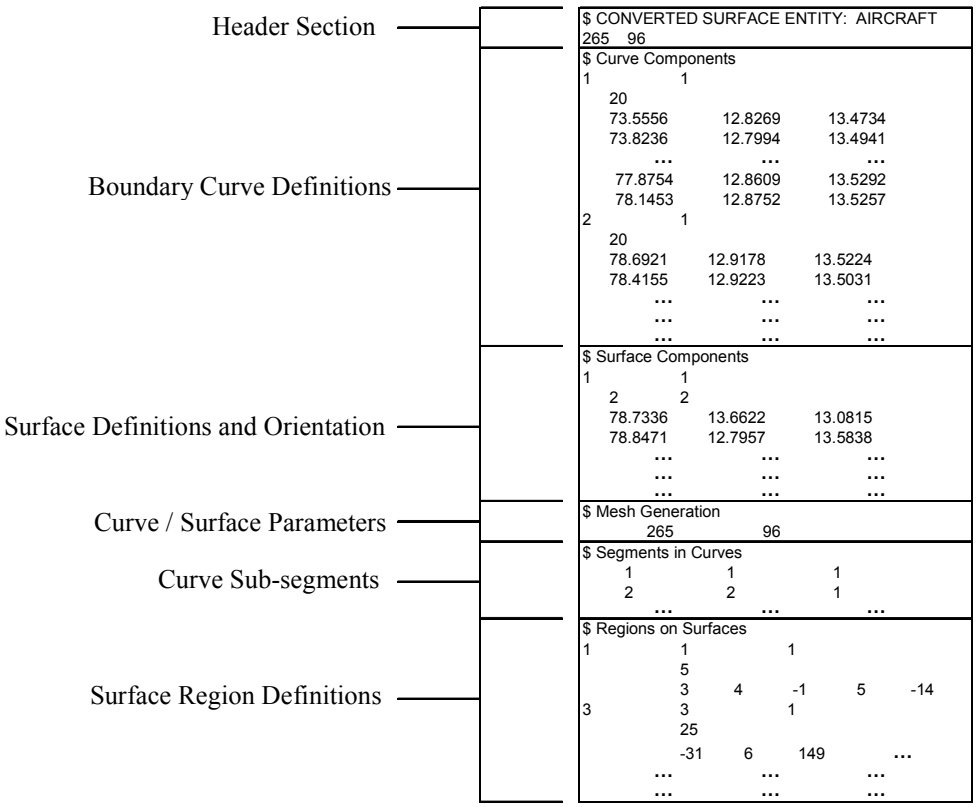

Figure 1: Surface File (sur file) Structure

The background file contains parameters which are used to define the element distribution throughout the flow domain. With respect to the initial surface meshing, this file can be thought of as specifying the "tightness" of the mesh in different surface regions. This is accomplished by allowing the user to create "sources". The grid spacing may be influenced locally by defining the spacing for each source (specified by an inner radius spacing and an outer radius spacing). The sources may be defined as point, line, or triangle entities. The placement (and size) of these sources is a very important consideration when beginning the gridding process. The number and distribution of elements can have a dramatic impact on both solution accuracy and computational time. It is desirable to include enough elements to completely capture all flow effects, however a compromise must be reached to ensure that a realistic solution runtime can be achieved. Therefore, the user must be sure to specify sufficient source definitions in all important regions on the geometry and in the flow domain. The background file is also used to specify the background grid spacing. An example of the background file structure is given in Figure 2.

The remaining two files required to complete a general test case solution are the boundary conditions file (bco file) and the flow properties file (con file). The boundary conditions file may be used to specify the surface and curve types used in the sur file. The surfaces may be defined as a solid wall, symmetry surface, or far field. The bac file defines singularity conditions for each curve (which could include all curve points or end points). The con file is used to define all necessary flow and solution properties. Mach number, alpha, beta, number of solution steps, and dissipation are a few of the possible parameters that can be specified in this file.

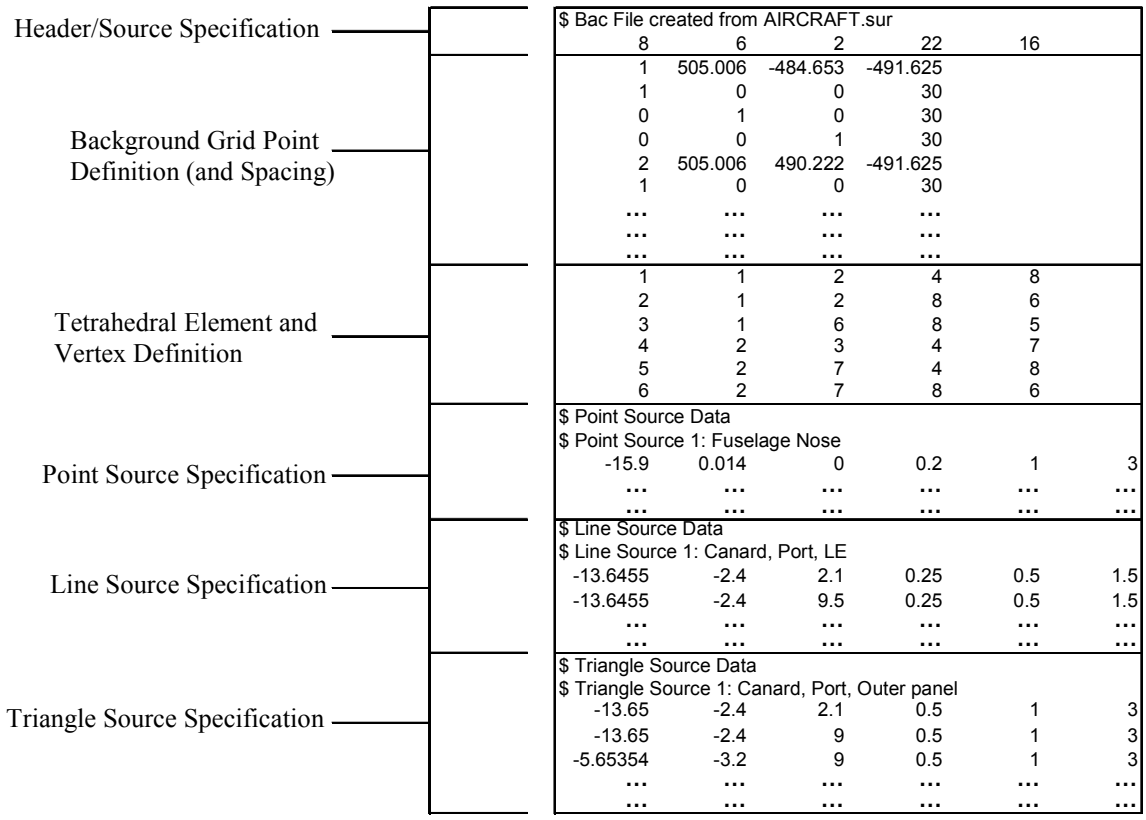

Figure 2: Background File (bac file) Structure

The surface mesh that is generated from the sur and bac files is used to create a volume grid of elements (which can contain up to several million elements). The resulting file is then reordered with an intermediate program (makeg3d) at which time the boundary conditions are applied. This file is then used with the con file as input for the Euler3D solver. Several solution files are generated by Euler3D. One of these is a loads file (lds file) which contains the calculated X, Y, and Z forces and moments at each solution step.

1.3 Objective

The creation of test case geometry files (including surface files, background files, and boundary condition files) can be quite tedious for complex geometries. One of the

most time consuming steps in the preparation of a STARS project is the generation of the primary geometry file, the "surface" file. As noted by Babcock [2004], the generation of the geometry model may require more time than the CFD analysis. An aircraft CFD model may require several hundred complex surfaces and curves to define. A model of sufficient complexity may even be prohibitive in certain cases if the designer is required to input the entire model manually. As such, the user may be required to resort to oversimplified representations of the model geometry which may not accurately capture all of the desired flow effects. A further difficulty lies in the debugging of manually created geometry files. A single complex surface may require many hundred coordinate values to define. If the model contains dozens, or hundreds of surfaces, much time can be devoted to locating errors in a model that continually fails in the mesh or grid generation process.

Also, the geometry files (including the surface file, background file, and boundary condition file) can require much time and practice for a new, inexperienced user to become proficient with. This may limit the ability of new users and restrict them to relatively simple test case models. The time that a new user spends creating these files is already significant, therefore a complex aircraft model would most likely be a very time consuming task for them.

In light of these observations, it has become desirable to develop a tool which could increase the efficiency of this step in the STARS solution process. An application which could allow the user to easily import or create geometry files from existing computer aided design (CAD) software could greatly reduce the amount of project time dedicated to this task and allow the user to focus his/her effort on the flow solution and

its evaluation. Such a tool could also allow the user to rapidly create many different geometry models representing different flight configurations or modifications to the geometry. This would make an iterative design study possible for many different design parameters even in the case of an extremely complex aircraft geometry.

The goal of the current effort is to investigate the implementation of such a tool and to develop an application to complete the desired tasks. The application should be user friendly and capable of accurately and efficiently generating all of the necessary geometry files to complete a general STARS CFD analysis. The application should be capable of generating the sur, bac, and bco files. It is also desirable to develop a user interface which will facilitate the use of the application by a wide range of users. If designed and implemented properly, the application should significantly reduce the time required to generate the supporting files for a STARS CFD project.

CHAPTER 2

LITERATURE REVIEW

2.1 Geometry Utility Approach

The approach for the geometry file generation utility (conversion program) was initially investigated as a choice between two different methods: the development of a completely independent, self-standing 3-D CAD program with an integrated user interface and conversion utility, and the development of a program which could convert existing CAD geometry generated in a commercial software package with a corresponding user interface. The decision was made early on in the process to utilize the latter method, and the justification for this decision follows.

- First, it was desired to be able to easily create STARS files for geometries generated and received from third party sources. In these cases, the party generating the geometry may not have access to a specialized STARS CAD program or may not want to recreate their geometry files.
- The creation of a program which could accept a standard commercial CAD format would allow the easy transfer of files between users and would also allow the use of the same CAD files for STARS analyses as were used for other analysis/design programs. An example of this could be a simple aircraft geometry created in Pro/Engineer. A user may desire to create a CAD model in Pro/Engineer in order to utilize the built-in structural, thermal, or machining applications. A post-CAD

converter would allow the user to utilize this existing model for a complete STARS analysis.

- A new 3-D CAD program would require a user to learn and become familiar with a new modeling program. Many designers already have experience with existing CAD software packages and are able to rapidly and easily create their desired geometry with that software. A requirement to utilize a new modeler would slow down the design process and would add unnecessary complications. This may result in fewer users utilizing the converter and in users creating representations that do not exactly match their existing geometry (due to differences in modeling tools).
- The use of a post-CAD converter program would allow the easy creation of STARS files from any existing (legacy) geometries. This would avoid the necessity of recreating several existing geometries which are desired for STARS analyses.
- Finally, the many commercial CAD programs which exist have several complex and useful design tools and features which would require extensive programming development and refinement in order to usefully employ. Most of these have required years of development and would have greatly extended the development requirements of this converter program. For example, Farin [1993] describes the development of basic CAD software as requiring multiple authors and several years of development time. It would not have been efficient to attempt to recreate the level of design quality of a commercial package and still remain focused on the overall goal of the creation of STARS geometry files in a timely manner.

2.2 Selection of Graphics Data Exchange Format

Because of the reasons listed above, the decision was made to develop a program which could accept geometry files from existing CAD software and convert them into files necessary for a STARS analysis. It was also decided that this approach, when combined with a well developed user interface (and other support programs), would allow even inexperienced users to use the STARS software with relatively little effort and would greatly reduce the time needed to develop a working knowledge of the STARS analysis process.

2.2.1 IGES

IGES Background

Since the development of early CAD software packages in the late 1960s, several different standards for the exchange of geometric data have been developed. In the late 1970s an effort was made to develop a standard, unified exchange format [Piegl, 1995]. This would allow designers to more easily create and transfer data among a wide variety of CAD and analysis programs. The resulting format was developed into the first national standard for CAD in 1979, which was collectively called the Initial Graphics Exchange Specifications, or IGES [The Initial Graphics Exchange Specification (IGES), 1999]. IGES was initially created to allow designers to transfer two-dimensional engineering drawings between non-common (dissimilar) systems. However, the specification was quickly expanded to include all entities needed for three dimensional models. IGES has also been expanded to include the translation and communication of

FEA models (to include boundary condition specification and unique entities such as loads and connectivity).

IGES has been created and maintained as an open standard, which is a key element to its broad use and success. Users create and employ new entities which are then reviewed and often incorporated into the official IGES ANSI standard. This has led to a continual improvement and modernization of the standard by the end user.

IGES Format

IGES was developed as an entity based standard. Fundamental entities necessary for CAD were initially developed, and entities of increasing complexity and versatility were gradually added to this set [The Initial Graphics Exchange Specification (IGES) 1999, Piegl 1995]. The entities include both geometric and nongeometric components. The geometric entities include all required items to fully define a physical body (such as points, arcs, splines, surfaces, etc.). In some cases there are multiple methods for defining a geometric property. These parallel the development (and implementation) of computer aided design methods. Examples include rational Bezier splines and surfaces (as created and defined by Bezier [1972], and outlined by Boehm [1987] and Joe [1989]), non-uniform rational Bezier surfaces (NURBS) (using evaluation and specification methods as outlined by Piegl [1995]), and power based surface definition [Farin 1993, Hoschek 1993]. The range of entities available for specifying a property makes IGES a versatile standard and allows users to employ a specification method which will be optimal for their particular case but will also be transferable to other systems.

Nongeometric entities are included for increased detail in shape specification as well as items necessary for the generation of engineering drawings and finite element

models. Several of these nongeometric entities allow (and are necessary for) precise surface definitions. An example would be a grouping of boundary splines, a NURBS surface, and all related references to other groups of trimming, translation, rotation, and scaling entities for the surface. This method allows for efficient data retrieval in the file and results in a decrease in redundant geometry definitions that are present in other standards. Several nongeometric entites have been incorporated to describe specific attributes and properties of both the individual and grouped geometric entities and the complete physical elements of the model as well.

The IGES data files are composed of ASCII text which can be easily created and translated by sending and receiving systems.

2.2.2 STEP

STEP Background

In 1984 the ISO (International Organization for Standardization) began to develop a new data exchange format that was meant to be the successor of the IGES standard. This standard became officially titled ISO 10303, but the acronym STEP (STandard for the Exchange of Product model data) has become the common title for the standard [STEP Application Handbook, 2006]. The goals for the development of STEP differed from those of IGES. While IGES was developed as primarily a geometric data transfer tool, STEP was developed as a means to transfer a very broad range of product data (encompassing the entire life cycle of the product / part). This data includes not only design elements, but many other elements such as machining, disposal, and maintenance data [Piegl, 1995].

STEP was developed as a modular standard. Instead of the creation of one set of standard data transfer types which each must be approved, tested, then added to the set (an example of this structure is the entity based IGES standard), STEP was created to encompass many subsets of standards for data transfer. These subsets are known as Application Protocols (or AP's) [STEP Application Handbook, 2006]. Therefore, STEP may be thought of as an umbrella for many approved and uniformly formatted standards instead of one single large standard. Each Application Protocol in STEP has been developed for a specific kind of product data either by the ISO or by third party organizations. The APs are submitted to the appropriate review committees and are evaluated and tested for completeness and usefulness (this process follows several stages in the ISO Standardization process, see STEP Application Handbook [2006]). Currently, twenty-two APs have been approved and implemented in the STEP file structure. A listing of some example APs is given in Figure 3. The current APs provide enough entities to make STEP a useful tool for engineering data transfer, however many believe that the full capabilities of STEP have not yet been realized. Many more Application Protocols are currently under review / development and it is expected that the total number of APs will eventually reach into the hundreds.

AP	Publishing date	Ballot stage	Title
AP201	1994	IS	Explicit draughting
AP202	1997	IS	Associative draughting
AP203	1994	IS	Configuration controlled 3D designs of mechanical parts and assemblies
	1998	TC	
	2000	TC	
	2004	TS	
AP204	2002	IS	Mechanical design using boundary representation
AP207	1999	IS	Sheet metal die planning and design
	2001	TC	
AP209	2001	IS	Composite and metallic structural analysis & related design
AP210	2001	IS	Electronic assembly, interconnection and exchange
AP210 2ND		DIS	
AP212	2001	IS	Electrotechnical design and installation
Ap214	2001	IS	Core data for automotive mechanical design processes
AP214 2ND	2004	IS	
AP215	2003	IS	Ship arrangement
AP216	2004	IS	Ship moulded forms
AP218	2004	IS	Ship structures
AP219	2006	DIS	Manage dimensional inspection of solid parts or assemblies
AP221	2006	DIS	Functional data and their schematic representation for process plants
AP223	2006	CD	Exchange of design and manufacturing product information for cast parts
AP224	1999	IS	Mechanical product definition for process planning using machining features
AP224 2ND	2001	IS	
AP224 3RD	2006	IS	
AP225	1999	IS	Building elements using explicit shape representation
AP227	2001	IS	Plant spatial configuration
AP227 2ND	2005	IS	
AP229	2006	NWI	Design and manufacturing product information for forged parts
AP232	2002	IS	Technical data packaging core information and exchange
AP233	2005	AWI	Systems engineering data representation
AP235	2005	CD	Materials information for the design and verification of products
AP236	2005	DIS	Furniture product data and project data

Figure 3: Example STEP Application Protocols (APs) (figure from STEP Application Handbook [2006])

The first version of STEP was implemented in 1994 (approximately ten years after the initial development began). Since that time, a second major release has occurred to update the standard (in 2002). One of the criticisms of STEP is the lengthy process required for the development and approval of APs. This has possibly led to a slower acceptance for STEP and has resulted in a slow development and implementation of the standard (other methods for exchanging some of the non-geometric product life cycle data have been developed at a much faster rate and have thus gained greater acceptance and usage).

STEP Format

The specific format and information included in a STEP data file is individually defined by each Application Protocol. Despite some variations, the overall structure for a given file follows the same general format. There are two primary sections: a header section and a data section. The header section contains optional items (which may describe the authoring system and other descriptory information) and all necessary items to read and translate the file by a receiving system (this includes entities such as the context information of the data section entries, the schema for the entries, and population groupings of the entries).

The data section is composed of "instances" which are specific for the AP currently employed (the properties of the instances are specified in the header definition section). The instances may represent either complex entity data types or single entity data types. Single entity data types are used when possible and contain a list of all necessary attributes for the instance listed in a formal predefined order (all attributes needed are listed under the name of that particular instance). Complex entity data types are necessary for some features that require multiple grouped entities (these may be either mapped locally (to internal attributes) or externally. One feature in the data section that is different from other standards (such as IGES) is that the attributes in STEP contain only non-derivable terms (for instance, derivative terms may not be given in an attribute list for an entity). A short example of a STEP file illustrating the basic format of the two sections is shown in Figure 4.

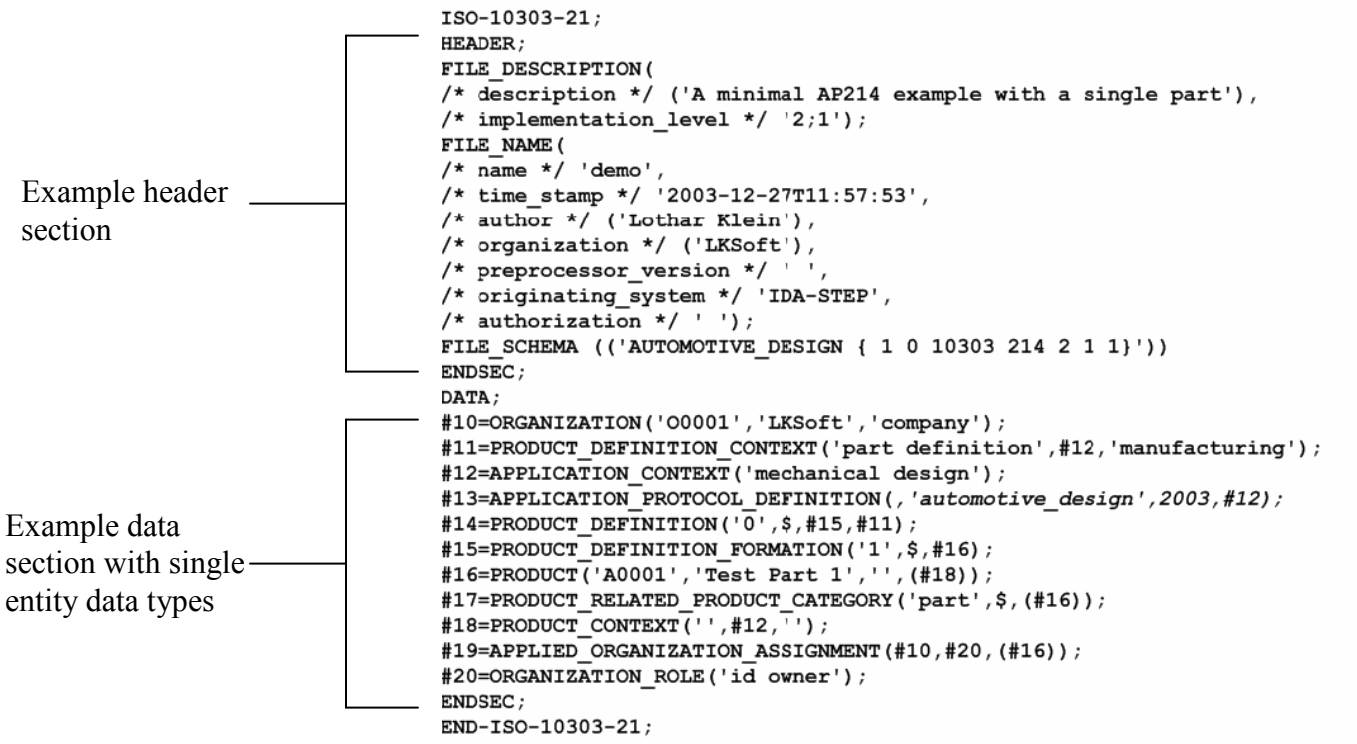

Figure 4: Example STEP file structure (figure from STEP Application Handbook [2006])

As with the IGES standard, the STEP data files are created with ACII text which helps to facilitate their readability as well as translation by receiving systems.

2.2.3 Additional Graphics Data Exchange Formats

In addition to IGES and STEP, several other data exchange standards were also examined for their potential use. The standards examined were chosen because of their current employment in various engineering and CAD fields. Some of the standards that were under consideration are listed below.

- VDA – QMS (Quality Management System): Developed primarily as a means for transferring surface model data.

- ACIS: 3-D model format developed by the Spatial Corporation and released in 1989 (latest version in 2006). Openly published until year 2000 release.
- VRML (Virtual Reality Markup Language): Developed to transfer data (points and edges) for polygon entities (specifically created for transferring models for various geometry viewer applications)
- Wavefront: Solid object file
- SET

While the data exchange formats in this list generally represent adequate methods to accurately transfer model data, they were not considered further after the initial investigation. In most cases, a primary reason for this rejection was their limited usage in current CAD software (compared to other standards). A standard which most users were familiar with and may have previous experience with would be preferable. Other reasons for their rejection at this point include inefficient geometry translation and storage (due to lack of optimal entity types) [Hagen, 1992], inability to transfer data between different CAD programs, and limited and/or poor documentation.

2.3 Surface Tessellation

In addition to utilizing standards which transfer complete geometric definitions of the model components, methods for transferring simplified tessellated versions of the model were also examined. Many CAD software packages have methods for creating and exporting faceted surface representations of solid objects. These methods create nodes on the various surface entities and use these to define the vertices of planar surface patches. Most commonly, the surface patches are created as triangular facets (see

Floriani [1987]), but it is possible to define more complex facets with an increasing number of vertices (but generating systems rarely utilize faces with more than three vertices). Examples of faceted models are shown in Figure 5.

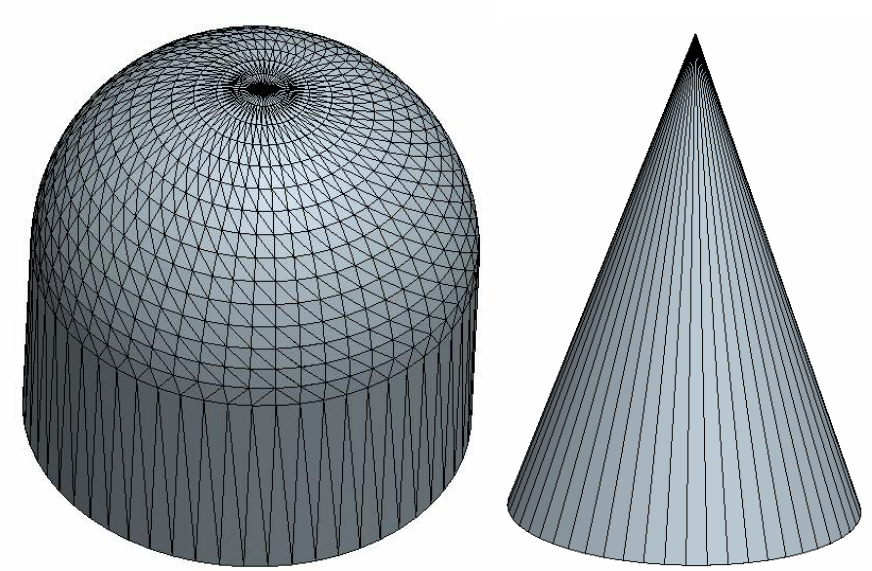

Figure 5: Example faceted surface representations of 3-D objects.

Vertex placement is dictated by the curvature of the surface being represented. Therefore, it is not uncommon to create very large, skewed facets (described in tessellation surfacing methods by Hoschek [1993]) as seen in Figure 5. The tessellation transfer format which was examined was called STL (or the Standard Tessellation Language). This is the most common format for transferring faceted surface information in CAD programs. The file generated is a simple ASCII listing of all facets for a chosen solid part. The basic STL format is given below:

solid loop *identifier*
begin global facet loop
facet normal vector *x y z*
first vertex node (*x11 y12 z13)*
second vertex node (*x21 y22 z23)*
third vertex node (*x31 y32 z33)*
facet loop
end solid definition

It is not uncommon for the ASCII STL files to become excessively large. Therefore, the facet vertex listing is not a very efficient means of describing and storing surface definitions. A binary STL format also has been developed in order to reduce memory requirements, however the files still remain very large compared to the entity based standards such as IGES and STEP.

While the generated file is easy to interpret by receiving systems, there are several disadvantages to using faceted surface definitions as a primary means to transfer solids geometry data. First, it is no longer possible to increase the surface resolution (or definition) of an object from the faceted representation. The continuous curvatures have been replaced by flat surfaces with discontinuities between them. Therefore, part of the surface definitions will always be lost and cannot be regenerated from this representation. Global surface definition items (such as boundaries) are also lost when utilizing this format; therefore it is not possible to easily group the facet elements into a single surface definition. Defining each facet as an individual surface entity was considered, but this would not be practical for the desired usage in STARS (the geometry file would become very large and virtually unreadable/editable by the user). Because the file only consists of non-grouped vertex listings, the groupings and placement of individual surfaces would be difficult for the user to interpret.

Testing was carried out to determine if the tessellation nodes (which are created on the actual surfaces and surface boundaries of the model) could be easily grouped into elements which could be used to create surface definitions. This was determined to be an unsatisfactory approach because of the non-uniformity of the point definitions (placements). Using only the STL generated tessellation nodes to define a 'U-V' grid of

surface patches can result in a very distorted surface definition with 'waves' in the resulting future surface representations/interpretations (due to the forced fitting of splines to these nodes). Although the creation of new nodes by means of interpolation/extrapolation between the STL nodes could be carried out in order to create a uniform grid of surface patches, this was not considered because of the inaccuracies and inconsistencies with the actual surface model that would result. The large spacing and very skewed elements that can exist in a faceted representation [Hoschek, 1993] made the possibility of errors in interpolated nodes very likely and the effects pronounced. A secondary, less severe problem encountered when attempting to use the STL nodes to generate surface grids was the unstructured nature of the vertex listing. Surface facet definitions are not necessarily created in an order corresponding to a single surface. Therefore, in complex models, linking the nodes to a particular desired surface entity can become complex and a source of errors.

Although the use of model representation methods such as surface tessellation may seem to be a faster, more efficient approach, the negative factors previously discussed make their use impractical for application in the creation of STARS files. In summary, a faceted representation was not chosen due to these factors:

- Use of discontinuous, flat surface 'tiles' would diminish further refinement and definition of the geometry when generating surface meshes
- Continuous, smooth surface curvature would be replaced by flat tiles with non-tangent 'corners' between them (which could, for example, result in shocks at each facet-facet interface)
- Use of STL nodes alone cannot create satisfactory 'U-V' surface grids

- Interpolated points for 'U-V' surface grids would result in inaccurate node placement
- STL geometry representations tend to be inefficient for complex CAD models

2.4 Final Selection of Exchange Format

The initial decision was whether to use one of the existing 3-D model exchange standards or utilize the data from a faceted representation to create an accurate model. As discussed in the previous section, several factors exist which make the use of a faceted model representation impractical for use in the current application for STARS. After an initial investigation, it was also determined that the creation of a new model using the data from the faceted representation would be inefficient compared to the utilization of an existing graphics standard. Therefore, the use of a faceted geometry file was not considered further.

Several standards for graphics data exchange were examined. Of the seven transfer methods previously discussed, only two warranted serious consideration. The other five standards/methods (VDA – QMS, ACIS, VRML, Wavefront, and SET) exhibited narrow usage/employment and possessed various other shortcomings which diminished their viability.

The two standards which were further considered for geometry transfer were IGES (Initial Graphics Exchange Specifications) and STEP (STandard for the Exchange of Product model data). These standards share many positive attributes. Both are widely used in many engineering applications, are efficient and effective means for translating and storing geometry data, and have good public documentation. After a careful

investigation of both standards (as discussed in previous sections), IGES was chosen as the standard to use for the current STARS application. The reasons for choosing IGES over STEP are outlined below:

- Although both are commonly used, IGES has been used more frequently and many people have more experience with this standard (and have stored many of their models with this format)
- IGES files tend to be more efficient for creating larger, more complex geometry files (which decreases the STARS conversion time)
- The format of the IGES file enables easier readability and interpretation
- Many of the additional translation features available in STEP (such as manufacturing and disposal data) are not needed for the present purposes
- IGES has been in existence and use much longer than STEP

One element of IGES which was found to be lacking with respect to STEP was the available published documentation. Documentation and guides for IGES do exist, however they were found to lack adequate information on several entities. In some cases multiple references must be consulted in order to obtain the correct user information. The documentation for STEP was found to be much more current and complete. The STEP references also contained much more detailed information about all aspects of this standard.

CHAPTER 3

METHODOLOGY

The geometry utility under consideration makes use of several key geometric entity types. Although differences in their use and employment exist, there are commonalities in the primary elements of each. Several of these entity types form the foundation of basic CAD and 3-D geometry software as well as widely used standards such as IGES and STEP.

While many simple geometry constructs are in existence (such as simple vectors, points, etc.), several more complex geometry features exist and therefore require more robust and detailed methods for their numerical representation. The deign features which are the most challenging to represent are model curves and surfaces. Several methods for accurately creating a curve or surface model have been developed and are extensively employed in geometric design. Different methods may be used to create these entities depending on algorithmic requirements, with the two primary methods being parametric and implicit definitions.

3.1 Implicit and Piecewise Implicit Surface Creation

The implicit method for creating complex curves and surfaces is the most familiar and intuitive method for many people. Surfaces and curves are defined in three

dimensions by equations which define implicit dependencies between the three coordinate components of the included points:

$$\{(x, y, z) \mid F(x, y, z) = 0\}$$

A very simple example of an implicit surface definition is that of a unit sphere (with no displacement):

$$f(x, y, z) = x^2 + y^2 + z^2 - 1 = 0$$

A primary advantage of using an implicit definition is that no control points with sometimes complex conditions need to be defined, as is the case with a parametric definition. This is true in the case that $f(x, y, z)$ and its first partial derivatives are continuous.

However, the use of implicit functions for defining complex model geometries poses several difficulties when employed in a computational manner. First, no method to define the direction of the surface or curve definition exists in the implicit format. This hinders the ordered creation of patches and grids. Also, the creation of implicit definitions for surfaces does not translate easily to a program (computational) algorithm. A more significant difficulty to overcome is that it can be problematic to create curve and surface divisions (in other words, connected bounded pieces of the curve or surface) which make up a grid of tangent surface patches.

A technique to overcome the difficulty of creating bounded surface components (patches) from implicit definitions is denoted as "free-form blending" [Hagan, 4], or piecewise implicit blending. This method involves the creation of an interpolation function (continuous to the C^k partial derivative specified by the input data) approximated from geometric data. As discussed by Hagen [1992], a body's surface geometry is first

divided into the desired number of surface regions (or patches) specified by the user. This chosen surface region is then subdivided into several polyhedral faces. An interpolating function is then defined for the set of vertices of the polyhedral faces. This can be used to define surface contours and generate a bounded grid definition for the surface patch under consideration. The polyhedral faces used are usually triangular elements as seen in Figure 6. The addition of new elements and vertices as seen in Figure 6 is analogous to adding new nodes to a purely parametric surface definition. This results in gradual modifications of the surface contour and increases the surface resolution. Adaptive surface correction algorithms can be created to modify the implicit definition in this way and therefore reduce errors between the actual geometric data and the implicit representation.

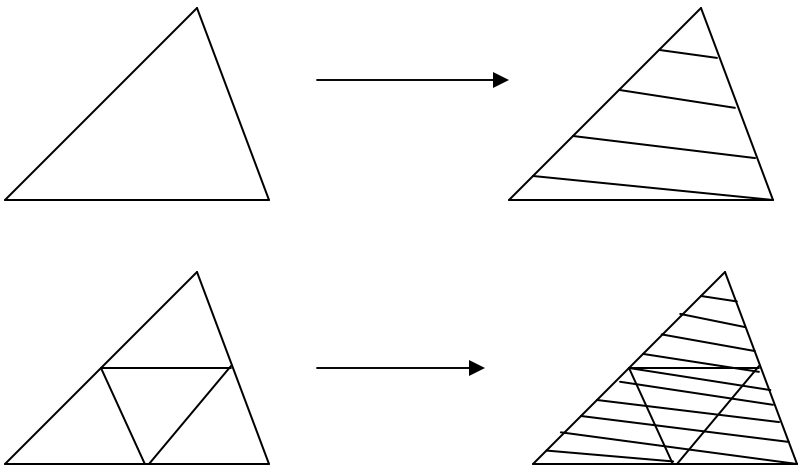

Figure 6. Example triangular elements subdividing a surface patch (figure from Hagan [1992]).

The "free-form" piecewise blending technique results in the creation of interpolants for the given set of geometry data. The interpoltant method generates

interpolating implicit functions which represent the set of vertices in the current surface patch.

In the current case we focus on the zero order contour interpolants (C^0) which represents the actual model surface contours needed for solid model geometry definitions. Hagen [1992] suggests using two classes of interpolants in this case: simplical and cubical interpolants. The simplical interpolant utilizes linear interpolation between the vertices of the elements (triangular in this case). In this case, the vertices have the effect of acting as control points as would be utilized in a B-spline definition. An example of this is given in Figure 7.

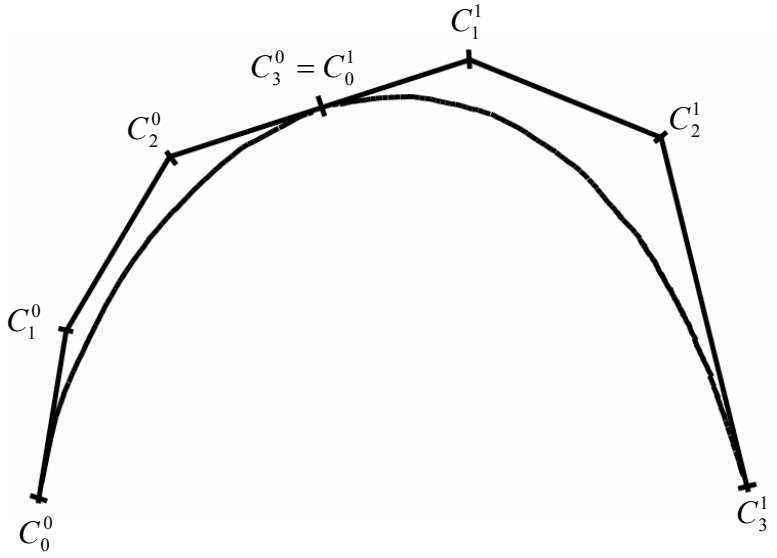

Figure 7: Example contour fit through linear interpolation of element vertices.

In the case of a cubical interpolant, Hagen [1992] suggests defining a 'hypercube' consisting of a greater number of points per element. Additional information regarding the interpolant technique can be found in Warren [1989].

Although techniques for resolving the problems associated with using implicit curve and surface definitions exist, such as "free-form blending", the implementation of

these techniques can require complex algorithms which may lead to model inaccuracies. The methods to resolve the problem of generating a properly bounded surface patch illustrate this difficulty. The "free-form blending" piecewise method basically requires the creation of contour control techniques as used in parametric surface definitions (which eliminates one of the advantages of using an implicit definition). The added difficulty of fitting implicit functions to this data and creating a new set of tangent C^k continuous patches (as seen in examples from Hagen [1992]) makes the use of techniques such as this problematic.

3.2 Parametric Curve and Surface Creation

The most predominant method for representing geometric models in a computational manner is through the use of parametric functions. The nature of parametric functions makes them far better suited for many programming algorithms. Unlike implicit geometric functions, parametric methods do not create functions which define dependencies between the coordinate components. Instead, an explicit function is defined for each separate coordinate. An independent parameter (sometimes referred to as a 'local parameter') is then used to transverse between user specified bounds for each coordinate component. This new parameter is usually normalized to transverse between 0 (start point) and 1 (terminate point), although the interval used can be arbitrarily specified by the user. Specifying this independent parameter as t, the explicit representation of a curve can be written as

$$F(t) = (x(t), y(t))$$

with

$$a \le t \le b$$

After defining the interval bounds for the parameter variable, it is a simple task to create uniform nodes along the curve from the coordinate functions.

In order to define a surface region, the same method is extended to the use of a second parameter variable.

$$F(u,v) = (x(u,v), y(u,v), z(u,v))$$

Sweeping through the range of one parameter while holding the second constant allows the user to create a uniform grid of nodes for a surface element. This makes the computational creation of a surface representation a relatively straightforward task to accomplish. Returning to the simple example of defining the surface contours of a sphere, the surface specification now occurs as a group of three coordinate functions as opposed to the single implicit function seen previously.

$$x(u,v) = \sin(u)\cos(v)$$
$$y(u,v) = \sin(u)\sin(v)$$
$$z(u,v) = \cos(u)$$

where the parameter intervals are defined as

$$0 \le u \le \pi$$
$$0 \le v \le 2\pi$$

Transitioning through the v interval at specific values of u results in the creation of a surface grid. Normal vectors may be evaluated at each node in the grid by simple cross products of the local u-v vectors. A u-v element of a surface patch is shown in Figure 8.

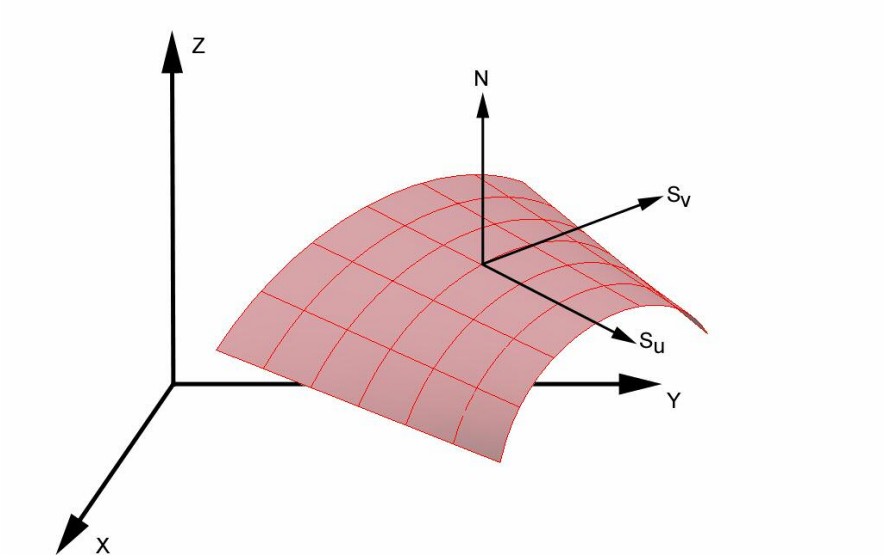

Figure 8: Example parametric surface patch.

The inherent ease of creating a bounded, ordered curve or surface through parametric definitions makes this method preferable and more appropriate for the current effort. Therefore, several methods for representing model entities through parametric means will now be discussed. All of these methods are employed by modern CAD packages and geometric data standards.

3.2.1 Monomial Interpolation Method

This method is often referred to as the 'least geometric' method used to define curves and surfaces due to its lack of control points, polygons, weights, knots, and other shape control elements. The monomials used can be viewed as replacing the basis functions used to define Bezier curves (or can be referred to as a special case of basis functions). The basis functions used to represent the polynomials in this case can be expressed as x^k, which are linearly independent functions (such as 1, x, x^2, …). Using

the monomials to create the shape polynomial, the parameterized function can be expressed in non-rational form as

$$p(t) = \sum_{j=0}^{n} A_j t^j$$

where

$$a \le t \le b$$

or in matrix format

$$\begin{pmatrix} 1 & t_o & \cdots & t_o^n \\ \vdots & & & \vdots \\ 1 & t_n & \cdots & t_n^n \end{pmatrix} \begin{pmatrix} A_o \\ \vdots \\ A_n \end{pmatrix} = \begin{pmatrix} P_0 \\ \vdots \\ P_n \end{pmatrix}$$ [Hoschek, 1993]

In this case, n represents the degree of the curve under evaluation. The individual coordinate components for a three dimensional curve can be explicitly expressed as

$$x(t) = \sum_{j=0}^{n} x_j t^j$$

$$y(t) = \sum_{j=0}^{n} y_j t^j$$

$$z(t) = \sum_{j=0}^{n} z_j t^j$$

with the x_j, y_j, and z_j representing the basis coefficients.

One advantage of using the monomial approach to curve/surface definition is the relative ease and efficiency for computing desired points compared to other methods. Less memory is typically required when using this approach and it is somewhat more computationally efficient. The method most commonly used to evaluate a monomial curve/surface is Horner's method [Piegl, 1995]. For example, assuming a curve of degree n

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_k x^k + \cdots + a_2 x^2 + a_1 x + a_0$$

and evaluating it at the point x_0, let

$$b_n = a_n$$

and

$$b_j = a_j + x_0 b_{j+1} \quad \text{for } j = n-1,\ n-2,\ n-3,\ \ldots,\ 3,\ 2,\ 1,\ 0$$

therefore

$$p(x_0) = b_0$$

Another way to express the point evaluation is with a quotient $Q_0(x)$, where

$$Q_0(x) = b_n x^{n-1} + b_{n-1} x^{n-2} + \cdots + b_2 x + b_1$$

and

$$p(x) = (x - x_0) Q_0(x) + R_0$$

where the remainder $R_0 = b_0$. Example applications of this method for curves of varying degree n follow [Piegl, 1995]:

n = 1: $p(x_0) = a_1 x_0 + x_0$

n = 2: $p(x_0) = (a_2 x_0 + a_1) x_0 + a_0 = a_0 + a_1 x_0 + a_2 x_2^2$

n = 3: $p(x_0) = a_0 + a_1 x_0 + a_2 x_0^2 + a_3 x_0^3$

n = ⋯: $p(x_0) = ((\cdots(a_n x_0 + a_{n-1}) x_0 + a_{n-2}) x_0) + \cdots + a_0$

Despite the relative ease of computing points on a monomial curve/surface, a number of significant disadvantages in their implementation exist. The first disadvantage is the lack of a geometric nature in the polynomial expression. The expressions tend to be of a more algebraic form and lack the elements that are often useful (and have become

almost commonplace) in geometric modeling (namely, knots, control points, etc.). Second, only the initial endpoint conditions are specified by the user (at u = 0). This can lead to difficulties in endpoint matching. However, a much more significant problem encountered while utilizing this design approach is the inherent instability due to numerical inaccuracies (round-off error [Piegl, 1995]). The monomial form tends to be very sensitive (and therefore greatly affected) by numerical inaccuracies and the computational precision used. This degree of sensitivity is illustrated in the figure from Farin [1993] below (Figure 9).

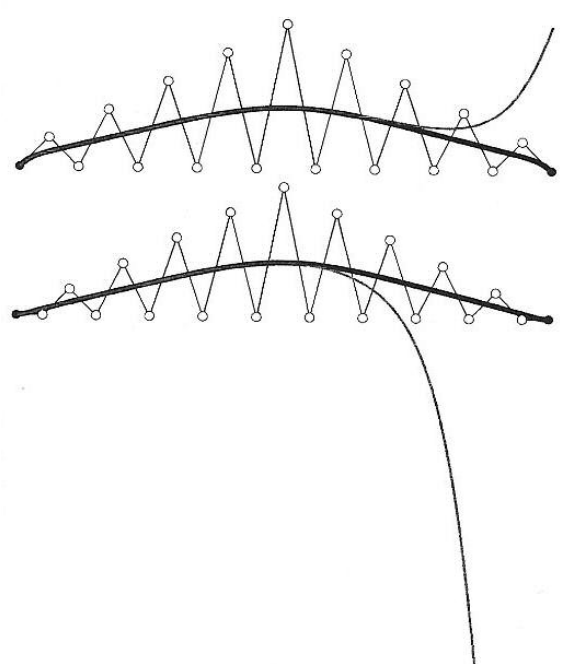

Figure 9: Example of the effects of slight coefficient perturbations on monomial (light gray) and Bezier (black) formats (figure from Farin [1993]).

The problems incurred by round-off errors are increased further by the fact that only the start point (u = 0) is specified with this method. Therefore, the terminate point for a curve (or the u, v terminal edges for a surface) must be calculated and are subject to calculation inaccuracies. This results in curve and surface discontinuities (increasingly so

for higher degree cases [Farin, 1993]) and may result in geometry failures in receiving systems. This terminal point/edge error when using monomials is shown in Figure 10.

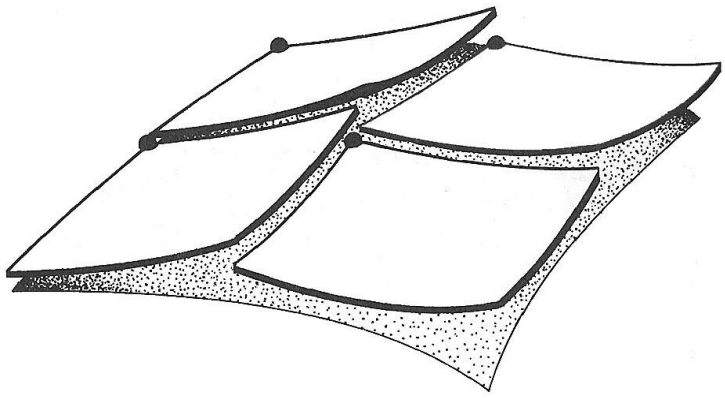

Figure 10: Exaggerated example of the effects of round-off error on surface patch continuity (figure from Farin [1993]).

Despite these potential design and implementation difficulties, this approach still remains a useful method to employ. It has historically been a much faster method for geometry evaluation and is relatively efficient for geometry data storage. Because of these two advantages, many early CAD and design software types relied heavily on monomial methods for data storage and transfer. However, advancements in computational speed and the memory available in current computers have rendered these advantages less significant. Therefore, the computational inaccuracies from round-off error (and precision) are much more of a design factor when choosing how to represent geometric entities.

The historical usage of monomial methods means that they are still commonly encountered in many current geometric standards and CAD software. So it is still necessary to include techniques to evaluate functions of this type in any receiving system software.

When using monomial functions, some generating systems typically use lower degree functions to eliminate the greater sensitivity to coefficient perturbations incurred with higher degree polynomials. Typically, the degree used is fourth order or lower. This means that the curves and surfaces must be highly segmented in order to be accurately represented. While this decreases the sensitivity to coefficient perturbations, the increased number of piecewise segments leads to greater difficulty in dealing with the terminal point round-off errors (see Figure 11). Therefore, additional algorithms must often be employed to adjust or truncate the terminal point coordinates in order to ensure proper curve/surface connectivity. While this may be appropriate in certain cases, this can lead to surface discontinuities (cusps) and other complications (Figure 11).

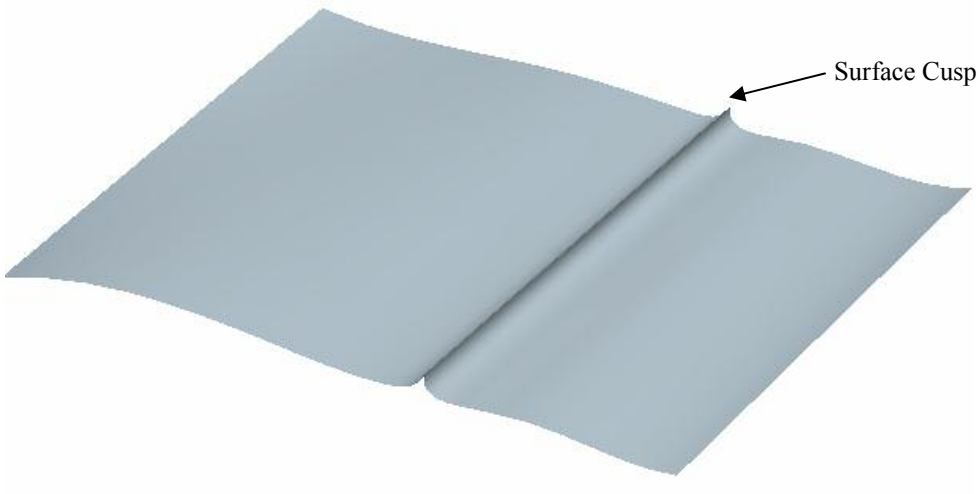

Figure 11: Example discontinuity due to terminal/start point adjustment.

3.2.2 Bezier Form for Curves and Surfaces

A much more geometrically informative method for a designer to create curves and surfaces is through the use of Bezier and B-Spline functions. In the monomial method, the coefficients that were used provided little information to the user about the

properties of the curve/surface. The Bezier method, however, places much more importance on the function coefficients and other related control entities.

The Bezier form for representing curves and surfaces (which is a more specialized form of the B-spline) has become one of the dominant and preferred methods for geometric data storage and transfer. In contrast to the monomial method, which used the linearly independent functions x^k as basis functions, this method makes use of Bernstein polynomials to define the basis functions. The binomial formula is used to derive the Bernstein polynomials [Hoschek, 1993]. The binomial formula can be defined as

$$1 = \left[(1-t)+t\right]^n = \sum_{r=0}^{n} \binom{n}{r} (1-t)^{n-r} t^r$$

with the Bernstein polynomials of degree n then given as

$$B_r^n(t) := \binom{n}{r} (1-t)^{n-r} t^t$$

The greater stability offered by the use Bezier definitions (see Figure 10) as well as the ability to fully specify both endpoint conditions makes this method attractive from a computational perspective. The additional geometric control features offered from Bezier curves make them attractive from a designer's perspective. The foremost feature that is geometrically beneficial is the control point matrix, $\{P_i\}$. A non-rational Bezier curve of degree n is specified using this set of control points as

$$C(t) = \sum_{i=0}^{n} B_{i,n}(t) P_i$$

where

$$0 \le t \le 1$$

This is the general form of a non-rational Bezier curve. The basis functions $B_{i,n}$ are Bernstein polynomials given in the form [Piegl, 1995]

$$B_{i,n}(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}$$

or recursively as

$$B_{i,n}(t) = (1-t)B_{i,n-1}(t) + tB_{i-1,n-1}(t)$$

Therefore, $C(t)$ of degree n can be written as

$$C(t) = \sum_{i=0}^{n} \left[\frac{n!}{i!(n-i)!} t^i (1-t)^{n-i} \right] P_i$$

The control points (or geometric coefficients) are the primary shape dictating elements of $C(t)$. This makes it easier for the designer to interpret and/or modify the overall shape. The placement and movement of control points pulls and stretches the curve in a very visual (straightforward) manner. An example of a second degree Bezier curve is given in Figure 12. As can be observed, there are three control points forming a parabola. The curve is parametrically defined by

$$C(t) = \sum_{i=0}^{n} B_{i,n}(t) P_i = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$$

A second example is given in Figure 13, this time for a third degree curve defined parametrically as

$$C(t) = \sum_{i=0}^{n} B_{i,n}(t) P_i = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

The effects of moving a control point are shown on the right half of Figure 13.

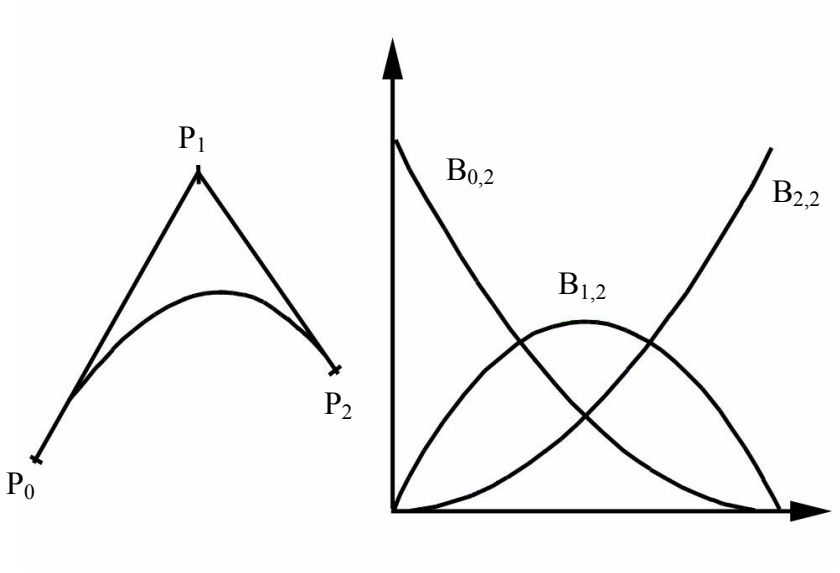

Figure 12: Example second degree curve and contributing basis functions.

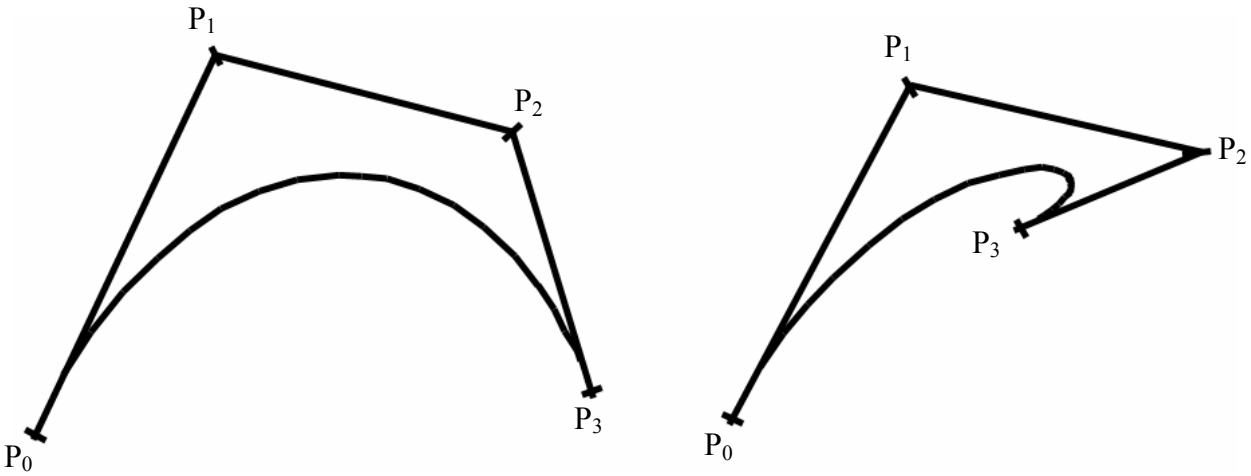

Figure 13: Example third degree curve (with movement of terminal control point shown on the right).

One seemingly intuitive design feature that is a result of using control points, but one which is very useful for curve and surface approximation, is the control polygon (sometimes referred to as a "control net" for surfaces). The control polygon is formed from a linear connection of successive control points. The "convex hull" property that is so often discussed when dealing with Bezier and B-splines is simply the property that

surfaces and curves are contained within their defining control polygon (Farin [1993]). The control polygon can be used for general approximations and for interference checking by the use of "minmax" boxes [Farin, 1993]. This allows for an additional method of fast interference checking.

Another useful quality of the control polygon is that any desired geometric transformations for the curve or surface may simply be applied to the control points/net. This allows for efficient transformations for any post-processing algorithm in the receiving system. When dealing with surfaces, it can sometimes be possible to use the control net as a general representation of the surface (for triangulation or other methods), depending on the degree of the surface. The appropriateness of using the control net to represent the surface depends on the bidirectional degree and the number of control points used.

A method commonly employed to compute non-rational Bezier points for given t values is the familiar de Casteljau method. (A more specialized algorithm for de Casteljau's method may be referred to as "blossoming" by some authors, such as Farin [Farin, 1993].) The de Casteljau method makes use of a series of repeated linear interpolations to obtain the point at the desired t. The point P(t) on the curve can be evaluated as

$$P_i^j(t) = (1-t)P_i^{j-1}(t) + tP_{i+1}^{j-1}(t)$$

where

$$i = 0, \ldots, n-j$$
$$j = 1, \ldots, n$$

The result is essentially the creation of new control points/polygons of increasing level until the point on the curve is reached. Each subsequent new control point set consists of

one point less than the previous defining set, with the final set consisting of two control points which define the desired curve point by interpolation. This method is nice in its simplicity and short algorithmic structure. It does not require the calculation of basis functions and is usually more stable than evaluation through the calculation of the Bernstein polynomials (not as susceptible to numerical errors, or round-off). An example of this evaluation method for a seventh degree Bezier curve is graphically shown in Figure 14. Figure 15 shows the triangular array used for this point evaluation.

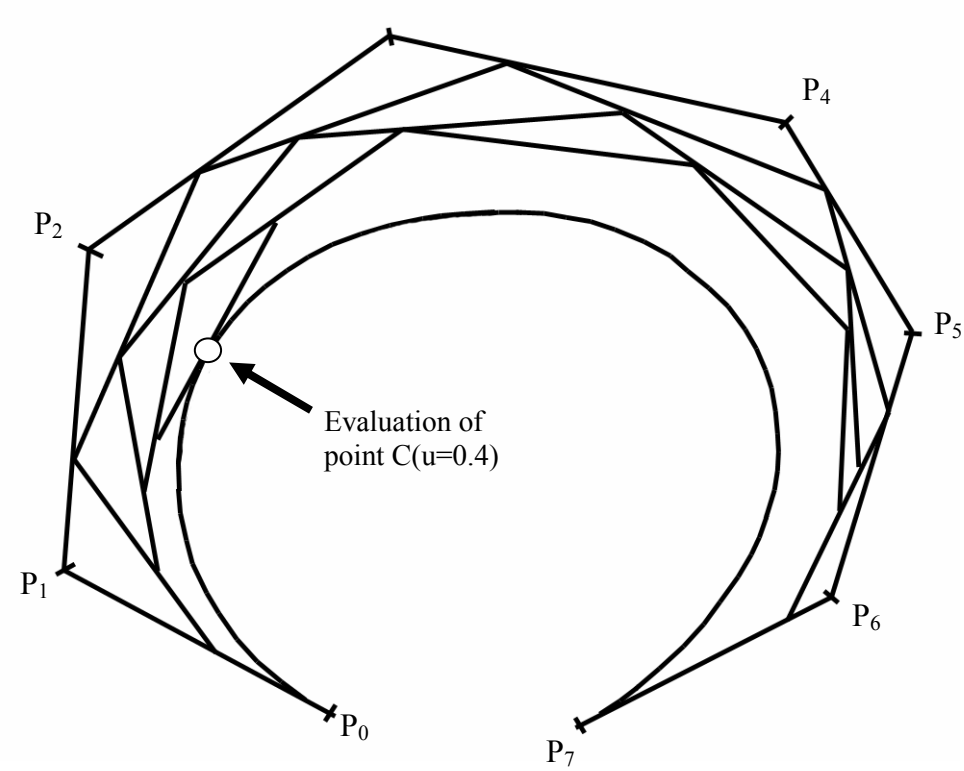

Figure 14: Example application of de Casteljau method for a seventh degree Bernstein curve.

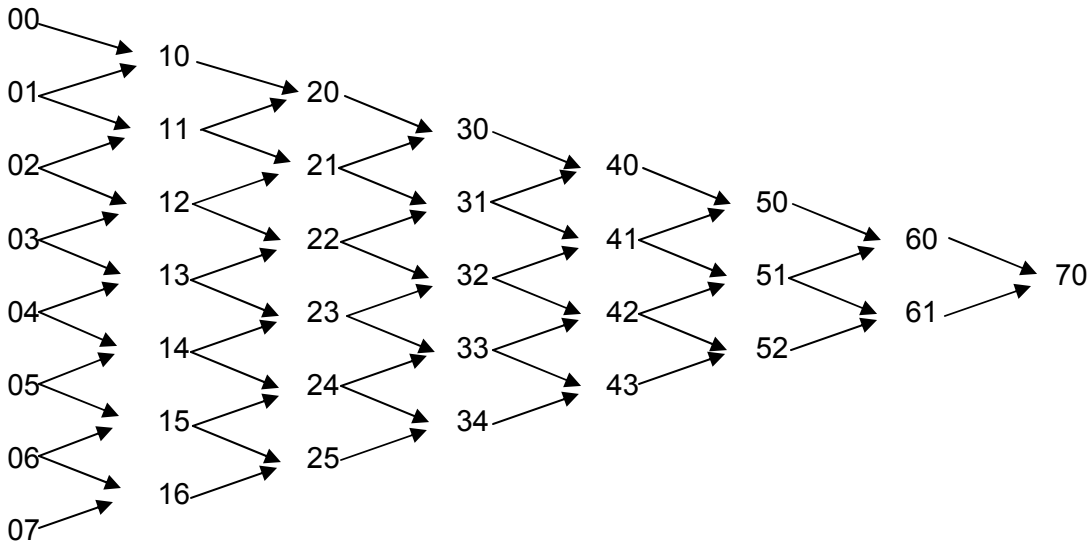

Figure 15: Example triangular array of successive control points for de Casteljau method.

The technique of blossoming mentioned previously is a more specialized extension of the de Casteljau method [Farin, 1993]. This method is analogous to knot insertion in the spline and subdivides the spline into smaller segments for evaluation. Although blossoming does offer another method for point evaluation, in most cases it does not provide a significant advantage. Unless it is already desirable or necessary to insert additional knots or segments, the blossoming technique tends to be less memory efficient and a computationally slower approach.

Despite the relative ease of representing curves and surfaces with a purely non-rational Bezier scheme, Bezier curves are not always acceptable for all geometries. If the shape is highly complex, then it will be necessary to utilize many control points in order to attempt to accurately represent the true shape. This will thus require a Bezier curve with a high degree, and a higher degree can often result in numerical errors and inaccurate (or even failed) shape representation. Commonly a degree of 10 or higher is

prohibitive [Farin, 1993]. A second difficulty when relying solely on a Bezier scheme is that of global curve distortion when redefining control points. Occasionally it may be necessary to refit a portion of the curve without modifying the entire geometry. This may be a case when the designer is attempting to maintain C^k continuity after small modifications. However, because the basis functions are global for the curve, a single point change will affect the entire curve.

One design method that addresses both of these problems is a primitive form of the B-spline. This method uses a group of piecewise Bezier curves to represent the total curve (and often uses division and degree reduction methods to subdivide a higher degree Bezier curve and reduce the local curve degree). This results in a preservation of the global curve degree and a reduction in degree for each of the "child" Bezier curves. Such curves are often referred to as composite Bezier curves, or, more informally, simply as Bezier splines (not to be confused with B-splines, which are "Basis-splines").

Piecewise Bezier curves first introduce the concepts of knots (although only through a rudimentary usage). Knots are the coincident control points of adjacent Bezier segments and are the degree defining elements of the global curve. In the case of simple composite Bezier curves, knots carry no additional methods for defining and are simply defined by current control points. This method may appear to be a combination of power-based and Bezier approaches and facilitates any desired conversion between the two. By the definition of the Bezier basis functions, the local start and terminate basis functions have the value of 1:

$$B_{i,n}(0) = B_{n,n}(1) = 1$$

This dictates that the local curve segment passes through the beginning and ending control points. As these points are knots, their sequence can be used to redefine (reinterpret) the curve as an interpolating spline if desired or required for the receiving system or any required transformations of the overall geometry.

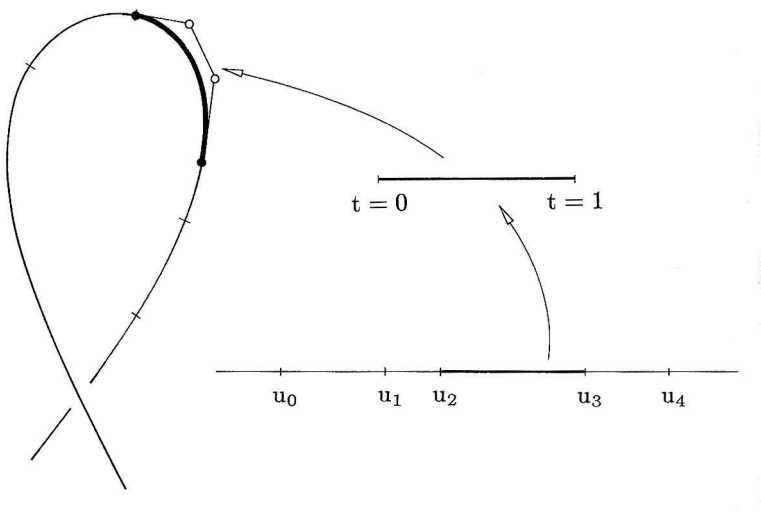

Figure 16: Example knot sequence with piecewise Bezier segments (figure from Farin [1993]).

Although the two major problems of relying completely on a polynomial or single segment Bezier scheme are nicely addressed by composite Bezier curves, a new design issue is also introduced. This is the ability to ensure C^k continuity at the curve intersection points (knots). The only method to adjust the continuity of the piecewise Bezier segments is by modifying the individual control point locations (which in turn modifies the curve geometry). The desired degree of continuity is then achieved by the fact that the current basis function definition ensures that the curve start/end vector is parallel to the vector $P_1 - P_0$. For example, to create a C^1 continuous piecewise Bezier curve, the tangent vectors to the connecting control polygon segments of the successive

curves must be identical. Expressing this condition from the general form of a Bezier curve [Piegl, 1995]:

$$\begin{aligned} C'(t) &= \sum_{i=0}^{n} B'_{i,n}(t) P_i \\ &= \sum_{i=0}^{n} n\left(B_{i-1,n-1}(t) - B_{i,n-1}(t)\right) P_i \\ &= n \sum_{i=0}^{n-1} B_{i,n-1}(t)\left(P_{i+1} - P_i\right) \end{aligned}$$

which means that for curves C_1 and C_2 of degree n = 3 with connecting knot of t_1,

$$C_1'(t_1) = C_2'(t_1)$$

$$\frac{n}{t_1 - t_0}\left(P_3^1 - P_2^1\right) = \frac{n}{t_2 - t_1}\left(P_1^2 - P_0^2\right)$$

and the constraint for the junction knot becomes [Piegl, 1995]

$$P_3^1 = \frac{(t_2 - t_1) P_2^1 + (t_1 - t_0) P_1^2}{t_2 - t_0}$$

This is illustrated in Figure 17, where the connecting control segments must be carefully constrained/positioned in order to make the global curve C^1 continuous.

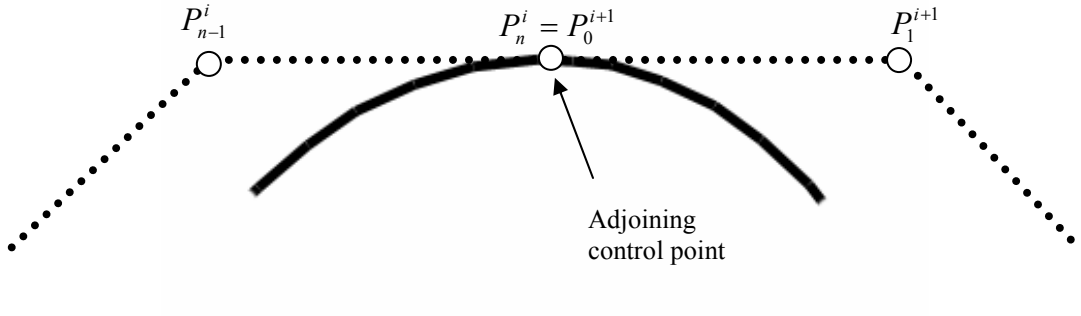

Figure 17: Piecewise Bezier curve with C^1 continuity.

This requires additional calculations and shape constraints that may require iterative shape checking in order to accurately model the geometry. In order to ensure the desired continuity, but not cause undesired modifications to the curve shape, it also may be necessary to use an increased number of control points (higher order). In some cases this may lead back to the previously discussed instability problems and can impose restrictions on the ability to accurately control the shape without additional composite segments.

3.2.3 B-Splines (Basis-Splines)

Another design method to overcome the difficulties incurred when using a high degree single Bezier or power curve to represent a complex shape (or increasing the order to ensure desired continuity) is to use B-splines, of which Bezier curves were a special case. Like composite Bezier curves, the B-spline method for shape design allows the designer to model relatively complex geometries without resorting to high degree polynomial curves. Instead, several lower order curves with fewer control points are used in a piecewise fashion (which also allows for localized shape control). The method of their definition also ensures that the global curves are C^k continuous.

At initial inspection, the B-spline method may seem to parallel that of composite Bezier splines. The global curve definition has been redefined to incorporate several lower degree curve segments, just as the case was with composite Bezier curves. However, Bezier splines, unlike B-splines, are defined by multiple independent entities which must be carefully constrained to produce the desired results. B-splines exist as single entities for the global curve definition. This results in a reduction of required

memory to store the spline data (because of redundant endpoint and continuity information required for Bezier splines). Depending on the size of the geometry file, the savings can be significant. The method of definition is also more convenient for the receiving system and can result in fewer errors.

Two fundamental differences exist to distinguish B-spline curves from Bezier splines. The first is a new set of basis functions (the application of the basis functions to the overall curve is also modified). The second difference is the inclusion of a knot vector. This vector is not only used to reduce the local degree of the spline, but also is used to control continuity within the spline and at the spline endpoints. The basis functions of 0 degree and p-degree are defined as follows [Hoscheck, 1993]

$$N_{i,0}(t) = \begin{cases} 1, & \textit{if } t_i \le t \le t_{i+1} \\ 0, & \textit{otherwise} \end{cases}$$

and

$$N_{i,p}(t) = \frac{(t - t_i)}{(t_{i+p} - t_i)} N_{i,p-1}(t) + \frac{(t_{i+p+1} - t)}{(t_{i+p+1} - t_{i+1})} N_{i+1,p-1}(t)$$

In the literature, basis functions such as those above may occasionally be defined in terms of the curve order (k = p+1) instead of degree. A number of important properties exist for the basis functions. These include [Hoscheck 1993, Farin 1993]

- $N_{i,p}(t) = 0$ for $t \notin \lfloor t_i, t_{i+p+1} \rangle$
- C^{k-2} continuity for $N_{i,p}(t)$ at all internal knots (assuming a '*simple*' knot vector)
- a quotient with zero divisor is set equal to zero
- within the defining knot interval, $N_{i,p}(t) > 0$

- $N_{i,p}(t)$ is a linear combination of two lower degree basis functions (p-1 degree) [Piegl, 1995]
- In the case that the number of knots in the knot vector is 2*(p+1), the basis functions $N_{i,p}(t)$ become the Bernstein polynomials of degree *p-1*, $B_i^{p-1}(t)$ (note that this is only true when p+1 knots are located at t = 0 and p+1 knots are located at t = 1) [Hoschek, 1993]
- $N_{i,p}(t)$ are piecewise polynomials
- For any knot span, partition of unity holds for $\sum N_{i,p}(t)$. Therefore, for the span $[t_i, t_{i+1})$ [Piegl, 1993]:

$$\sum_{j=i-p}^{i} N_{j,p}(t) = \sum_{j=i-p}^{i} \frac{t - t_j}{t_{j+p} - t_j} N_{j,p-1}(t) + \sum_{j=i-p}^{i} \frac{t_{j+p+1} - t}{t_{j+p+1} - t_{j+1}} N_{j+1,p-1}(t) = 1$$

- For the global curve span, $N_{i,p}(t)$ attains only one maximum

Although the general form of non-uniform B-spline basis functions was defined above, in certain instances uniform basis functions (equal knot spacing) may be used. This results in a simplified expression for $N_{i,p}(t)$ as

$$N_{i,p}(t) = \frac{(t-i)}{(p-1)} N_{i,p-1}(t) + \frac{(i+p-t)}{(p-1)} N_{i+1,p-1}(t)$$

In this case (uniform basis functions), the basis functions become translational equivalents and are simply shifted copies of each other along the knot vector [Piegl, 1993].

As noted in the basis function properties listed above, the basis function definitions follow a local support property. This allows them to exhibit influence over a local spline knot segment and therefore enables localized shape modification that was not possible in

a single Bezier spline scheme. This is very useful for interactive or optimization shape modification methods, as the global shape may be maintained without having to resort to a drastic increase in the spline degree.

The knot vector is a very useful element of B-spline curve design, allowing for efficient algorithms for internal as well as boundary continuity definitions. The relative spacing of the knot vector determines the overall shape of the basis functions. In some cases this spacing may be uniform, but in the more general case the spacing is non-uniform (e.g. NURBS). The knot vector for m knots is simply defined as

$$T = \{t_0, t_1, \ldots, t_{m-1}, t_m\}$$

where t_i are knots with

$$t_i \leq t_{i+1}$$

It should be noted that knots may be of zero length (repeating). Generally, the resulting knot spans (individual polynomial pieces of the curve) of degree p are joined with C^{p-2} continuity at the knots.

The spline continuity can be specified with careful selection of the knot components. This is accomplished in terms of knot multiplicity. The knot multiplicity is generally specified in two ways: through global multiplicity and local basis function multiplicity. When multiplicity is discussed in the literature, it more commonly refers to the former definition. For example, a knot vector for second degree basis functions (with p = 2, n = 5) may be specified as

$$T = \{0, 0, 0, 1, 2, 3, 3, 4, 5, 5, 5\}$$

In this specific case, the knots t = 0 and t = 5 have a global multiplicity of 3 and varying basis function multiplicity. For example the basis function multiplicity of the knot t = 0 for the first three basis functions is three, two, and one

$$N_{0,2} \rightarrow \{0, 0, 0, 1\}$$
$$N_{1,2} \rightarrow \{0, 0, 1, 2\}$$
$$N_{2,2} \rightarrow \{0, 1, 2, 3\}$$

The continuity of each basis function is affected by the multiplicity. If the multiplicity of the knot is L, then the previous basis function continuity of C^{k-2} is reduced to C^{k-1-L}. Therefore, for each multiple knot component, the basis function loses a degree of continuity. Another effect of basis function knot multiplicity L > 1 is a reduction of the influence of the current basis function. The local support interval of the basis function containing the knot of basis function multiplicity L is reduced in nonzero interval influence from k to $k-1-L$ intervals. In other words, the basis function interval length is reduced. An example of a quadratic basis function (B-spline order of three) with a knot of multiplicity 2 is shown in Figure 18.

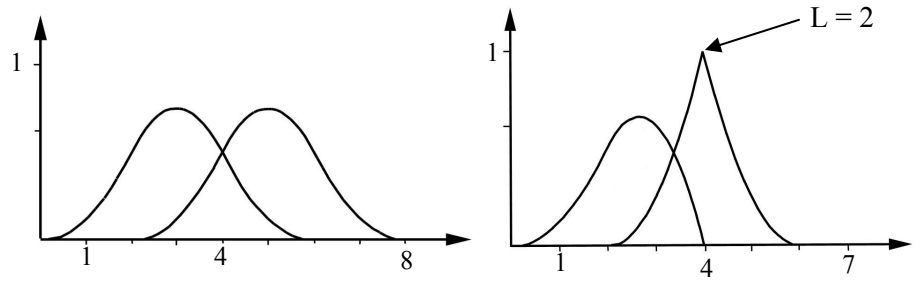

Figure 18: Effect of increasing knot multiplicity on an individual basis function (left plot is with complete degree 1 multiplicity, right plot is with a multiplicity of two for t = 4).

Another example of multiplicity effects is shown in Figure 19 [Piegl, 1995] for second degree basis functions with a knot vector of $T = \{0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5\}$. In this figure, the decrease in the local support interval can be observed for $N_{6,2}$ as a result of the knots at t = 4 and t = 5, where

$$N_{6,2} \rightarrow \{4, 4, 5, 5\}$$

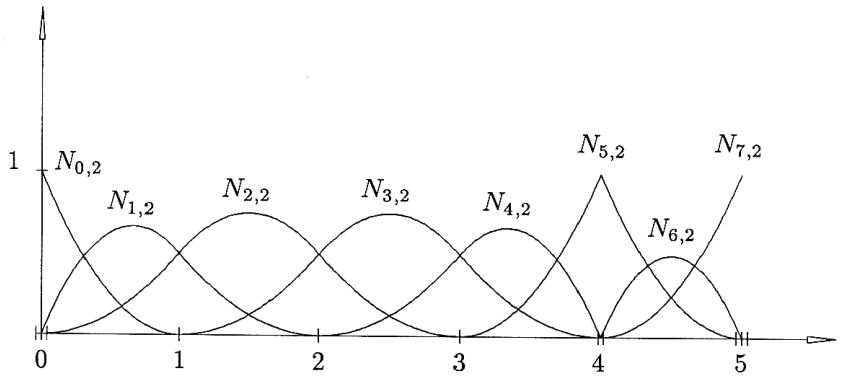

Figure 19: Example multiplicity effects for second degree basis functions [Piegl, 1995].

It should also be noted that spline end point continuity (boundary conditions) can be controlled with proper knot vector specification. In the above cases it has been assumed that the design goal of the generated splines is to have tangent, or *clamped*, ends. This is the case in most design circumstances. In order to ensure clamped boundary conditions, the knot vector must be of the form

$$T = \left\{ \underbrace{t_0, \ldots, t_0}_{p+1=k}, t_1, \ldots, t_{m-1-p}, \underbrace{t_m, \ldots, t_m}_{p+1=k} \right\}$$

where $t_0 = 0$ and $t_m = 1$. Therefore the initial and final knots must have a multiplicity equal to the order of the B-spline. The choice of this multiplicity may be adjusted in order to modify the desired behavior at the spline boundaries.

For a given knot vector and set of p-degree basis functions $N_{i,p}$, the B-spline (of p degree) can be evaluated at any parameter value t by

$$C(t) = \sum_{i=0}^{n} N_{i,p}(t) P_i$$

The set of points P_i (analogous to control points in the purely Bezier case) are termed de Boor points. Each set of de Boor points with its start and terminate knots forms a control polygon called the de Boor polygon. As with Bezier splines, each segment follows the convex hull property for its corresponding de Boor polygon. Another key property carried over from the Bezier scheme is that affine transformations can be applied to the set of de Boor points in order to effect desired geometric modifications.

For the present purposes, B-spline curves are in general desired to be closed and maintain at least C^1 boundary continuity. Therefore, the start and terminate knots are defined as 0 and 1 respectively and have a multiplicity of $p+1$. It can be noted that the corresponding basis functions reduce to Bernstein polynomials $B_i^p(t)$ due to the multiplicity of the start and terminate knots.

Examples of B-splines and their corresponding basis functions (uniform and non-uniform) are illustrated in Figure 20.

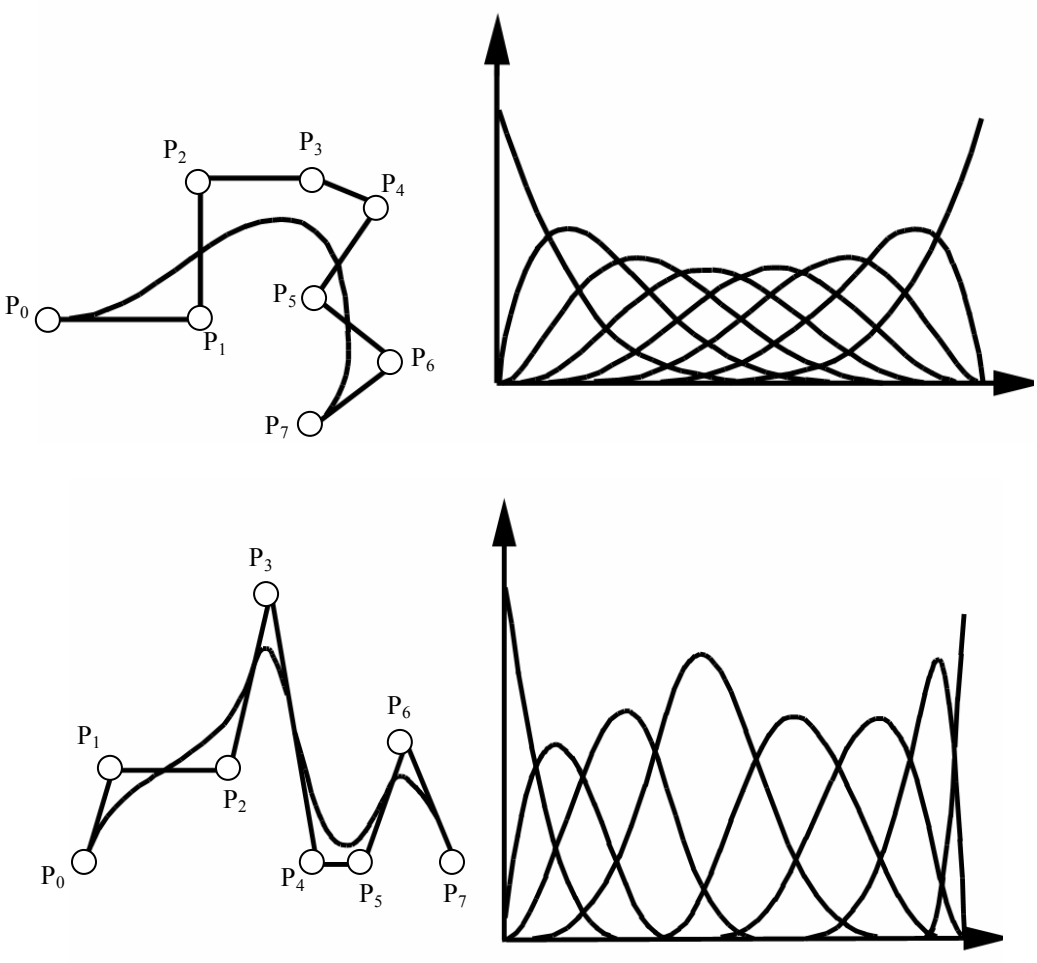

Figure 20: Example B-spline curves and basis functions.

In addition to the utilization of knot multiplicity to control internal spline continuity, the inclusion of repeating de Boor points may be used. In this case, certain de Boor points are identical (coincident) in their definition. The combination of knot and apparent de Boor point multiplicities can be used to induce certain geometric effects, such as tangency at desired angles internal to the spline or "rounded corner" effects. An example of a B-spline utilizing coincident de Boor points is given in Figure 21.

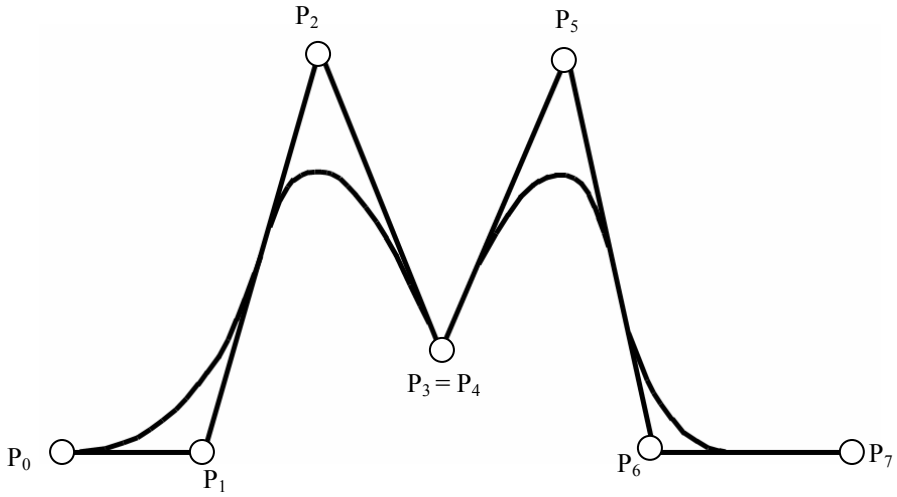

Figure 21: Coincident internal de Boor points in B-spline .

Another design element that is of significance is the choice of local spline segment degree. The choice of degree affects the "closeness" that the curve has for its de Boor polygon. Lower degree curves tend follow the de Boor polygon in a close manner, with this effect diminishing with increasing degree. This is due to the fact that fewer de Boor points contribute to the local spline segment $C(t)$ for lower degrees [Piegl, 1995]. This affects the "controllability" of the spline and may require the use of rational curves to better shape high degree splines. It also affects the degree to which the curve may be estimated through use of its de Boor polygon. For some design purposes (and for better user visualization) it may be desirable to create a curve which follows the defining control polygon in a semi-visual manner. In this case, the designer must resort to the use of rational functions, degree reduction, or knot insertion. Knot insertion and rational functions are the preferred methods for design in this case. A figure illustrating the effects of degree elevation is shown below (Figure 22).

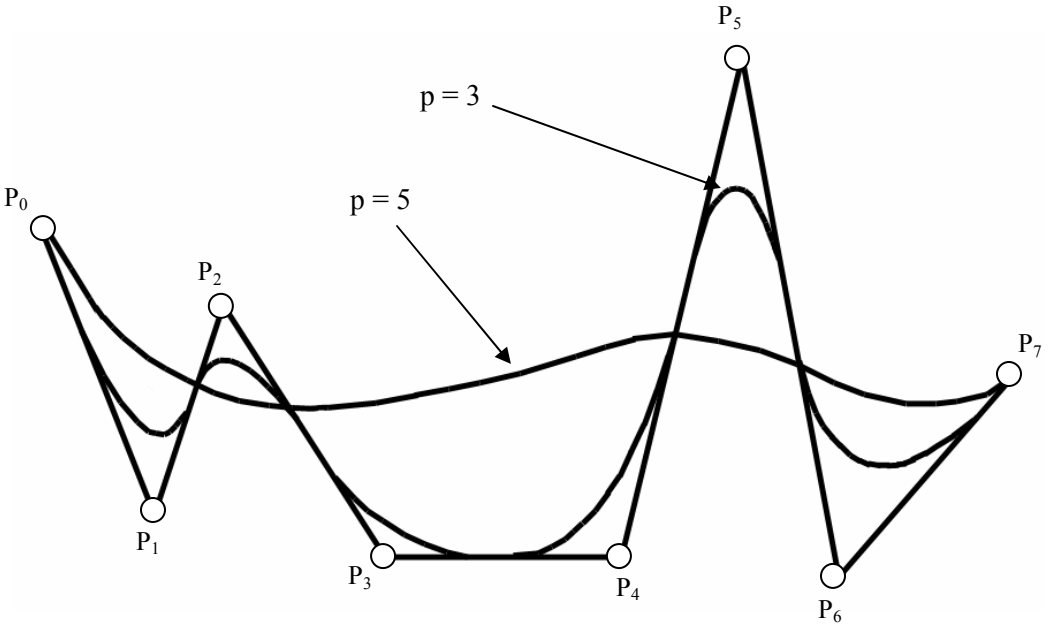

Figure 22: Effects of B-spline degree elevation.

Although basis function evaluation is preferred in most computational algorithms, it is possible to determine points on a B-spline without explicit calculation or transfer of the basis functions. The algorithm by de Boor is one such method, and this algorithm is analogous to the de Casteljau algorithm for Bezier curves. One advantage of this algorithm is that it is a very numerically stable method of curve/surface evaluation. However, it tends to be less time efficient than basis function algorithms. The de Boor evaluation method evaluates points on curves/surfaces by linear subdivision. This can be thought of as converging on a point through repeated knot insertions. As knots are inserted in the control polygon, the number of basis functions contributing to the knot will decrease (by $p-k+1$), where k is the knot multiplicity and p is the degree.

Therefore, repeated knot insertion at the desired parameter value will eventually result in convergence at the desired point. This process is illustrated in Figure 23.

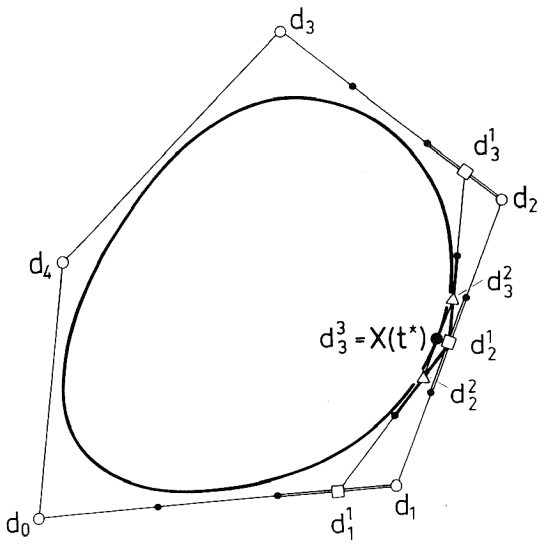

Figure 23: De Boor algorithm for B-splines (figure from Hoschek [1993]).

3.2.4 Rational Curve Specification

Although the previously defined geometric methods are of key importance to the designer and provide much latitude in accomplishing design objectives, certain geometries may be difficult to accurately represent through their use and may require a great deal of subdivision and control points to attempt to do so. This can result in a large increase in the storage space required for many designs and can result in a geometry representation that is computationally expensive for a receiving system to evaluate and modify. Therefore, another geometric modeling tool has been introduced to aid in the design of such entities. This includes the concept of rational parametric curve and surface modeling.

The concept of rationality stems from the fact that certain geometric entities cannot be accurately and/or efficiently represented through the defined basis functions (examples include conic curves). To account for this design issue in geometric modeling,

ratios of the basis function matrices are used. When a ratio is employed to define the entity, it is termed a rational entity (curve or surface):

$$x(t) = \frac{X(t)}{W(t)} \qquad y(t) = \frac{Y(t)}{W(t)} \qquad z(t) = \frac{Z(t)}{W(t)}$$

As stated previously, each non-rational B-spline segment is defined as the weighted sum of its control points, with the partition of unity property applying to the basis functions

$$C(t) = \sum_{i=0}^{n} N_{i,k}(t) P_i$$

where

$$\sum_{i=0}^{n} N_{i,k}(t) = 1$$

Therefore, in this general non-rational case, the basis functions can be thought of as acting as weights affecting the influence of the control point. It can be desirable (and necessary) in certain instances to increase or decrease the relative effect of the basis functions (and therefore control points) on the spline. This is accomplished by employing an additional weight matrix. The new rational curve may be defined as

$$C(t) = \frac{\sum_{i=0}^{n} N_{i,n}(t) w_i P_i}{\sum_{i=0}^{n} N_{i,n}(t) w_i}$$

where w_i are the basis weight modifiers (all non-negative to avoid singularities). The w_i are defined in a manner to satisfy the requirement that

$$\sum_{i=0}^{n}\left(\frac{w_i N_{i,n}(t)}{\sum_{i=0}^{n} w_i N_{i,n}(t)}\right)=1$$

For computational applications, the rational definition method can be thought of as an application of homogeneous coordinates to form a projection of a four dimensional curve, E^4, into three dimensional Euclidean space. This method results in efficient geometry data storage and receiving system interpretation. The weighted control points are now defined in four dimensional space as P'

$$P' = (wx, wy, wz, w)$$

from which the three dimensional Euclidean curve can be mapped. This mapping can be best described with a two dimensional form. P' is now defined with homogeneous coordinates (wx, wy, w) in E^3. This n+1 dimensional curve can be mapped into n-dimensional space to obtain the desired weighted 2-D curve. This is accomplished through projection onto a hyperplane of unit weight ($w = 1$). The origin of projection is considered to be the origin of the 3-D Cartesian coordinate system. Therefore, the point $P = (x, y)$ is determined by projecting the point $P' = (wx, wy, w)$ onto the hyperplane $w = 1$ by following a connecting vector to the defined origin. This is illustrated in Figure 24.

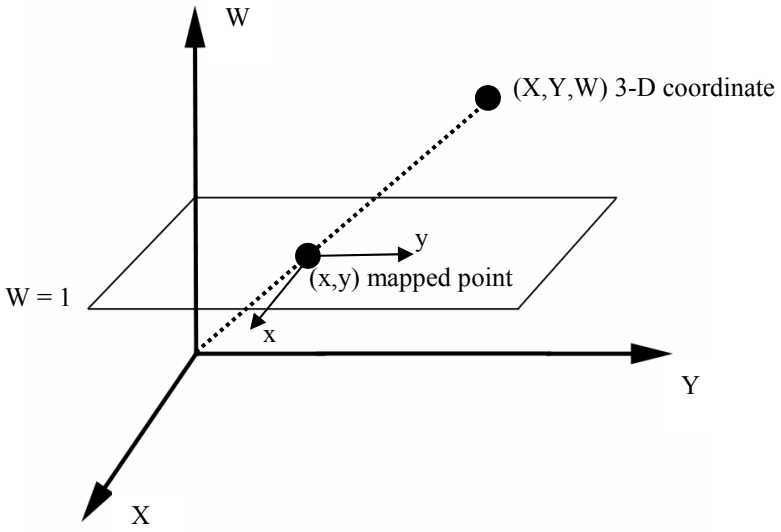

Figure 24: Projection mapping of a point in E^3.

It is in this way that conic geometric entities may be accurately created with B-spline methodology. An example of one method for the representation of a circular arc projected from E^3 homogeneous coordinates is shown in Figure 25.

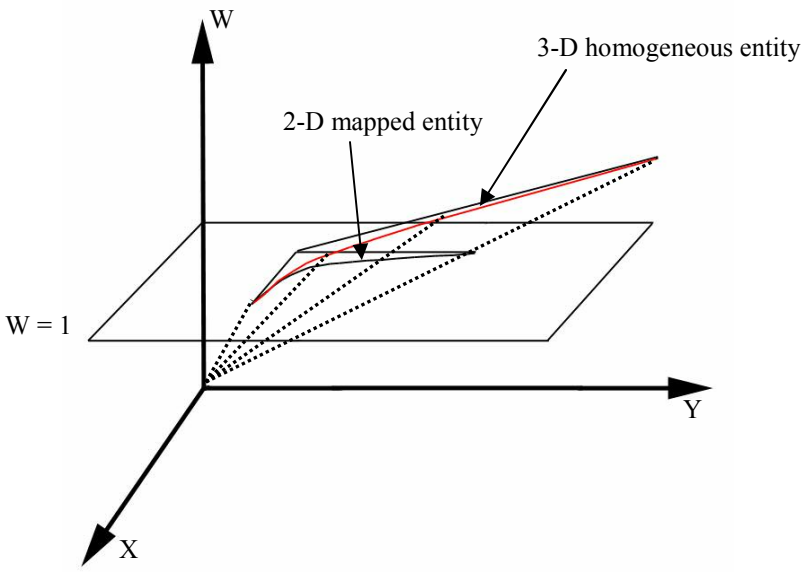

Figure 25: Example weighted representation of conic entity from homogeneous projection.

The resulting effects of weight variance can be a power design tool. It is with this design element that B-splines gained much more modeling power and near universal usage in CAD systems and design tools. Rational B-splines require comparably little data to represent complex entities and are efficiently processed by receiving systems. They also offer great control over the modeling of complex curvatures. An example of a rational B-spline with varying weights at control point P_2 is shown in Figure 26.

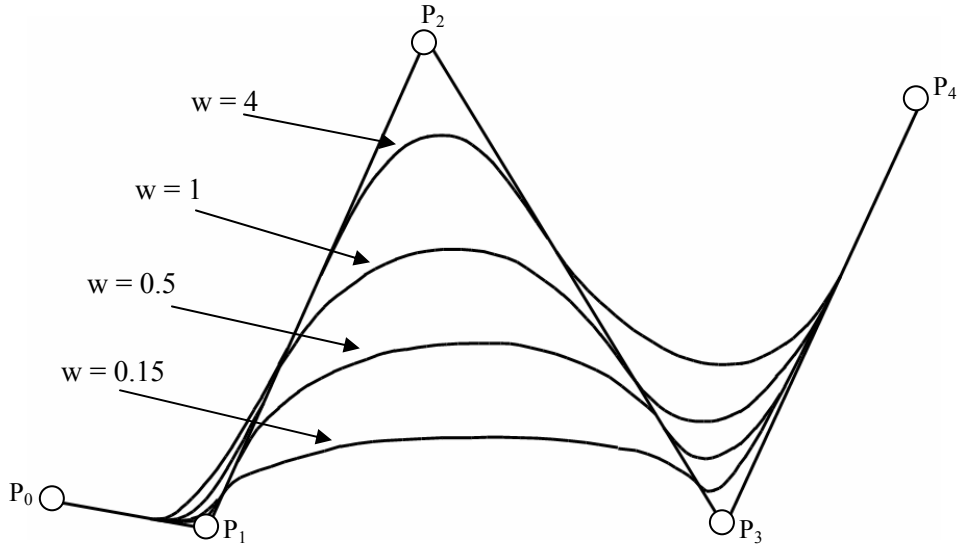

Figure 26: Example control point weight modification (at point P_2).

The added geometric features of B-splines combine to make them a powerful design tool and the preferred curve and surface modeling tool in CAD software. The use of non-uniform knot sequences allows for easy tangency control as well as efficient specification of internal continuity. The introduction of rational representations gives the designer further leverage to accomplish complicated design tasks. Together, these features represent the familiar NURBS entity (Non-Uniform Rational B-Spline). NURBS have gained wide acceptance and usage due to their efficient implementation, processing and storage.

3.2.5 Tensor Product Surfaces (Non-Uniform Rational B-Spline Surfaces)

The primary methods for surface representation in geometric modeling utilize the techniques developed for Bezier and NURBS curve design. The many advantages offered by NURBS derived design methods make them attractive to the designer and are desirable in surface design algorithms. The resulting NURBS surfaces have become one of the most prevalent design elements in CAD and geometric modeling software. The characteristics of Bezier and B-splines which made them particularly advantageous to the designer are carried over to three dimensional surfaces and make their resulting storage and processing efficient.

All parametric surface representation methods require the subdivision of the model geometry into a set of bivariate, rectangular surface patches. Each patch is created from a planar grid of isoparametric curves which are stretched and deformed (mapped to Euclidean 3-D space) due to the influence of two sets of univariate basis functions. The bivariate grid is typically defined with the parameters u and v. Although differing methods for creating parametric surfaces exist, the most common is the tensor product method [Hoschek, 1993]. The most general form of a tensor product surface is

$$X(u,v) = \sum_{i=0}^{n} \sum_{k=0}^{m} A_{ik} F_i(u) G_k(v)$$

or, in tensor matrix form, [Hoschek, 1993]

$$X(u,v) = \left(F_0(u),\ldots,F_n(u)\right) \begin{pmatrix} A_{00} & \cdots & A_{0m} \\ \vdots & & \\ A_{n0} & \cdots & A_{nm} \end{pmatrix} \begin{pmatrix} G_0(v) \\ \vdots \\ G_m(v) \end{pmatrix}$$

where $A_{ik} = (x_{i,k}, y_{i,k}, z_{i,k})$ represent the grid of control elements (points or de Boor points) and $F_i(u)$ and $G_k(v)$ are the univariate basis functions. An example tensor product surface patch is shown in Figure 27.

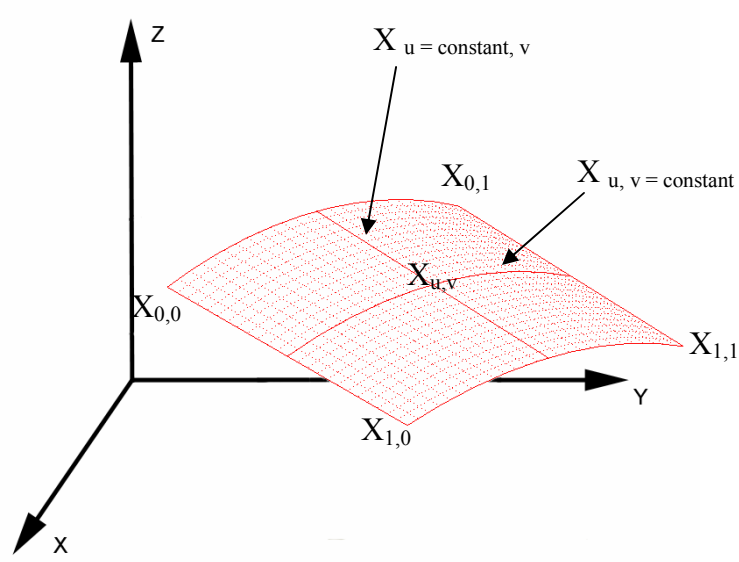

Figure 27: Tensor product surface patch

As with curve specification, the basis functions may be specified in monomial form to yield a power basis surface. In this case, the tensor product becomes [Piegl 1995, Hoschek 1993]

$$
\begin{aligned}
X(u,v) &= \sum_{i=0}^{n}\sum_{k=0}^{m} A_{ik} u^i v^k = \left[u^i\right]^T \left[A_{ik}\right]\left[v^k\right] \\
&= \underbrace{\left\{A_{0,0} + A_{0,1}v + A_{0,2}v^2 + \ldots + A_{0,m}v^m\right\}}_{b_0} + u\underbrace{\left\{A_{1,0} + A_{1,1}v + A_{1,2}v^2 + \ldots + A_{1,m}v^m\right\}}_{b_1} \\
&\quad + u^2\underbrace{\left\{A_{2,0} + A_{2,1}v + A_{2,2}v^2 + \ldots + A_{2,m}v^m\right\}}_{b_2} + \ldots + u^n\underbrace{\left\{A_{n,0} + A_{n,1}v + A_{n,2}v^2 + \ldots + A_{n,m}v^m\right\}}_{b_n} \\
&= b_0 + b_1 u + b_2 u^2 + \ldots + b_n u^n
\end{aligned}
$$

As was the case with power basis curve representations, the above monomial tensor surface representation may be evaluated with Horner's method. The use of

monomial basis functions in parametric surface design is relatively limited compared to Bezier and B-spline schemes (due to the previously discussed reasons for curve specifications). However, it is not uncommon to encounter this technique in various CAD and geometry modeling software and this method remains a viable technique.

The most prevalent forms of tensor product parametric surfaces utilize Bezier and B-spline basis function definitions. The general Bezier form is expressed as

$$X(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} A_{ij} B_{i,n}(u) B_{j,m}(v)$$

with $B_{i,n}$ and $B_{j,m}$ representing the univariate Bernstein polynomials. Similarly, the basic B-spline form utilizes basis the functions $N_{i,p}$ and $N_{j,q}$

$$X(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} A_{ij} N_{i,p}(u) N_{j,q}(v)$$

with knot vectors [Piegl, 1995]

$$U = \left\{ \underbrace{0,\ldots,0}_{p+1}, u_{p+1},\ldots,u_n, \underbrace{1,\ldots,1}_{p+1} \right\}$$

$$V = \left\{ \underbrace{0,\ldots,0}_{q+1}, v_{q+1},\ldots,v_m, \underbrace{1,\ldots,1}_{q+1} \right\}$$

Examples of B-spline tensor surfaces are shown in Figure 28. Two of the contributing basis functions are also shown. The second surface illustrates the effect of a multiple internal knot sequence.

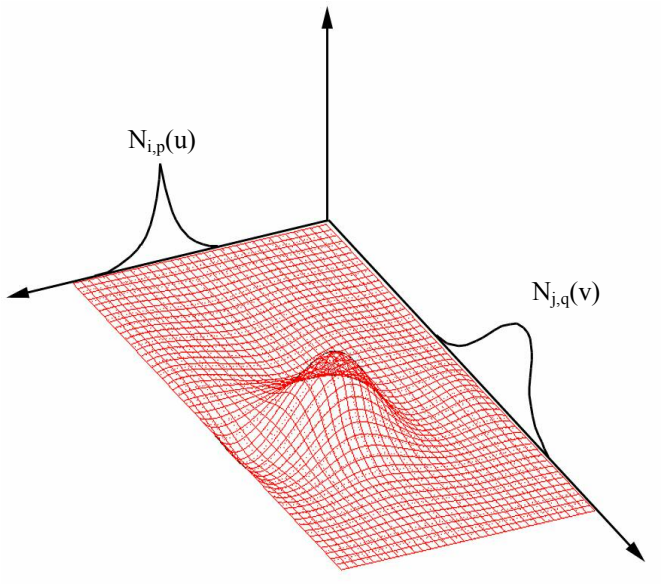

Figure 28: Tensor product surface patch

The set of control points A_{ij} are often termed the control net (or grid). This grid may be used as an approximation for the surface in the case that the surface is of sufficiently low degree in each direction (recall that a lower order has the effect of 'tightening' the surface toward the control net). Typically, tensor surfaces may be accurately created with cubic or quadratic splines (with cubic curves more often encountered in design applications). Figure 29 shows an example cubic surface patch with its control net. The effect of degree elevation is also illustrated in the figure.

A further variation of the B-spline formulation incorporates rational definitions. As with NURBS curves, rational B-spline surfaces include weight vectors to further enhance (or decrease) the effective influence of each basis function. The rational tensor surface is defined as

$$X(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j} A_{ij} N_{i,p} N_{j,q}}{\sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j} N_{i,p} N_{j,q}}$$

In the case that the included knot vectors are nonuniform (for example, to maintain boundary line inclusion) the surfaces are called NURBS surfaces. Additional examples of NURBS surfaces are given in Figure 30.

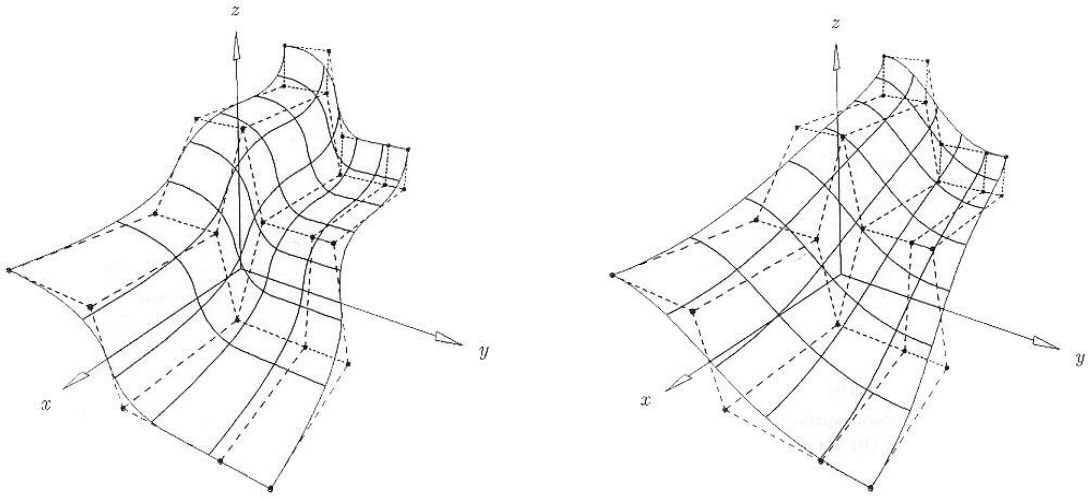

Figure 29: Effects of surface degree elevation [Piegl, 1995] (biquadratic surface, p = 3, on left; biquartic surface, p = 4, on right)

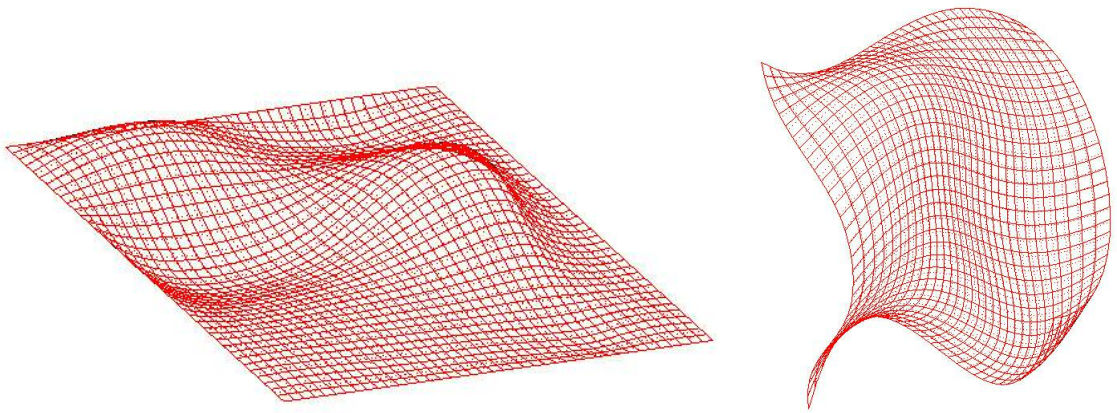

Figure 30: Example NURBS surfaces

In general, it is desirable to have at least C^1 continuity between adjacent surface patches. This boundary condition is slightly more complex than for the case of Bezier or

B-spline curves. In the simplest case, the tangent boundary curves of two adjacent surface grids are coincident; i.e. of the same length, terminate and start points, and having identical knot and control point sequences. If the connecting spline is located at $p+1$, the condition of C^1 continuity can be expressed as [Hoschek, 1993]

$$\frac{\partial X_{pq}}{\partial u}\left(u_{p+1}, v\right) = \frac{\partial X_{p+1,q}}{\partial u}\left(u_{p+1}, v\right)$$

or

$$\frac{n}{\Delta u_p}\left(A_{nk} - A_{n-1,k}\right)_{pq} = \frac{n}{\Delta u_{p+1}}\left(A_{1k} - A_{0k}\right)_{p+1,q}$$

where $\Delta u_p = u_{p+1} - u_p$, which can be used as a reference method to check for the proper evaluation (and precision) of imported models. The degree of "twist" may also be utilized in highly blended models (with large gradient areas) to evaluate the proper intersection and blending of tangent surfaces. The more difficult situations arise when the connecting surface grids are not of the same degree and vary significantly in size. Various algorithms may be employed to ensure the desired degree of continuity (including curvature continuity, C^2, and C^k in general).

3.2.6 Three and Five-Sided Surface Patches

Although parametric patches have been discussed thus far as being defined strictly by a rectangular format, it should be noted that other formats may also be encountered in standard geometry representations. The two other patch types encountered most frequently are triangular and five sided patches. The most common method of expressing a triangular patch is a simple modification of the rectangular scheme. In this case, two adjacent corner control points are made coincident, resulting in

one boundary curve of zero length (this method is commonly employed for spherical entities). Due to this simplified triangular representation, no modifications are needed in preexisting surface algorithms. A second method for triangular patches involves the use of Barycentric coordinates. In this case, a central weighted node is placed at the center of the triangular patch and a control net (with a triangular grid) is created. The specification of a control point coordinate now requires three parameter variables (typically *u*, *v*, and *w*).

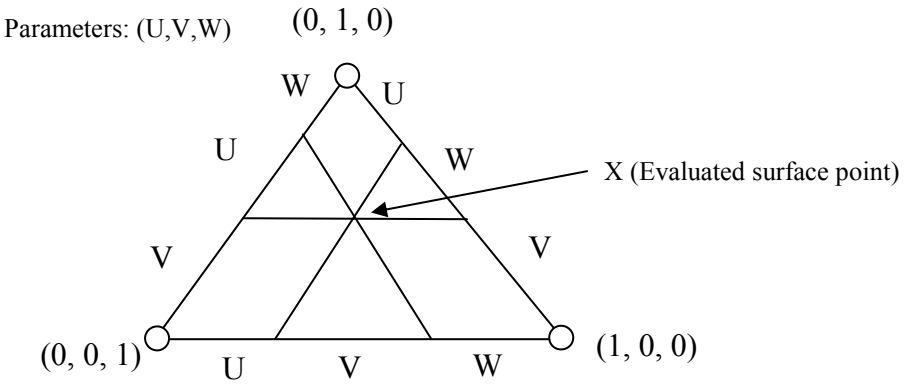

Figure 31: Triangular surface patch [Hoschek, 287]

This method requires extensive modifications to existing surface algorithms. The three parameter surface type is not commonly encountered in CAD or geometry standards. The five-sided patch type utilizes two parameters and requires little modification to rectangular algorithms. In most cases, an internal knot value (or in some cases only an internal parameter value) is specified as a special breakpoint in one of the surface boundaries. By use of a repeating knot sequence, a boundary cusp may be created which gives the appearance of a divide in the boundary. A corresponding dividing knot location is created in the opposing boundary curve. The curve with the cusp is considered as two distinct spline elements (and is defined as such), while the

opposing curve remains a single entity. The surface algorithm therefore requires little modification (still maintaining a rectangular grid). An example five-sided patch is shown in Figure 32.

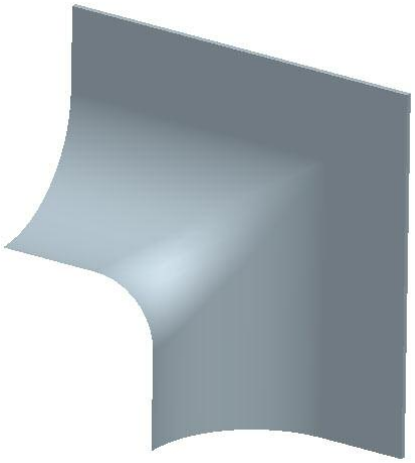

Figure 32: Five-sided surface patch

An alternative method for a five (or greater) sided patch resembles that of the triangular patch representation. In this method, a new centralized control point is created. New boundary splines are defined radially from this point, dividing the patch into several subdomains. The subdomains created are of rectangular form and can therefore be interpreted with existing surface algorithms.

3.3 Converter Program Structure

The conversion utility was created to import, error check (and correct if possible), and ultimately create STARS files from a user's CAD geometry. It also includes additional features to assist the user in the creation of the supporting STARS files.

As discussed previously, the chosen geometry transfer file type was the IGES standard. Many supporting reasons justifying this selection were described in Section

2.4. Virtually every modern CAD software package incorporates this standard as a primary file storage type. Of the major CAD programs available, Pro-Engineer was selected as the CAD program to use for the generation of all the geometry test cases.

A diagram of the basic block structure of the converter application is shown below (Figure 33).

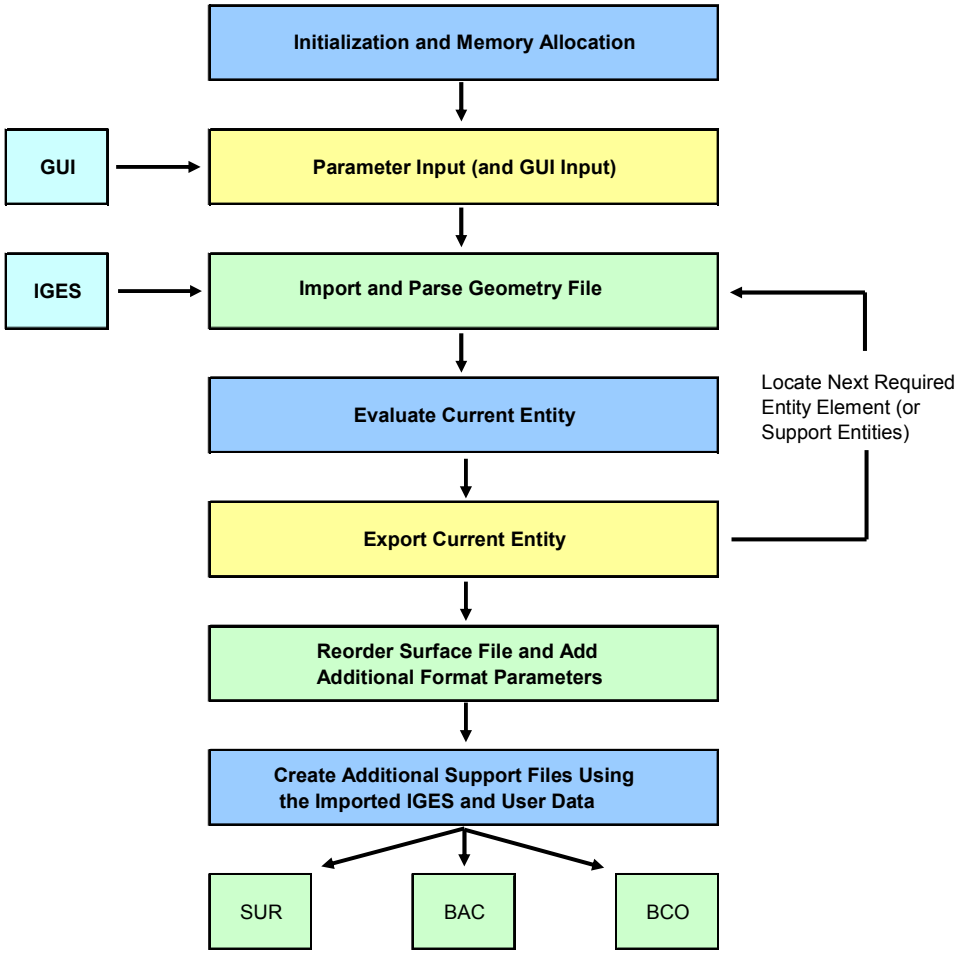

Figure 33: General converter structure

3.3.1 Geometry Import

The first design objective is to import, parse, and interpret the geometry data from the user IGES file. Many entities are available in the IGES standard, with the most significant discussed in Section 3.2. Different generating systems may create an IGES

file with variations in the specific content and file structure, but the general format is standard among all IGES geometry files. This format is outlined in Figure 34 and consists of a System Generation and Specification section ("Global" section), Directory Entry (DE) section, Parameter Data (PD) section, and Terminate section. Additional data sections may be present in some geometry files (up to six data sections, including a Flag and Start section). Each section is presented in ASCII format.

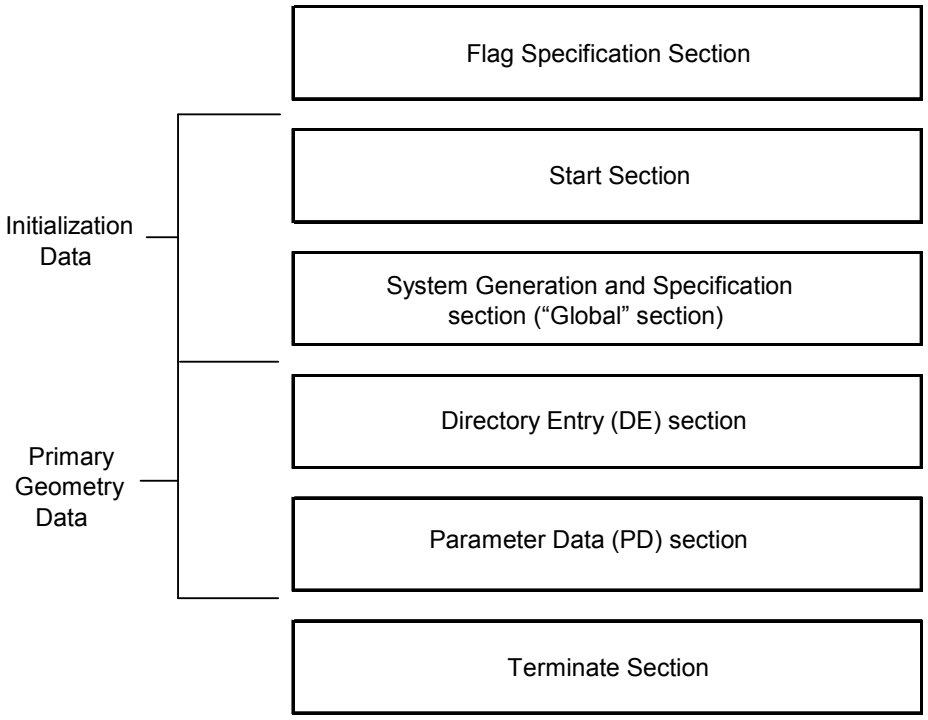

Figure 34: IGES file structure

Although the "Flag" data section may be listed as the initial component of the file, this section is not commonly encountered. Its use is limited to the binary and compressed ASCII file versions. This section contains data parameters used to specify attributes and the file type for the receiving system. However, the standard IGES files produced by the major CAD packages are in ASCII form and rarely utilize this optional data section.

The most common initial data section is the System Generation and Specification section ("Global" section). This section contains a variety of data which helps the receiving system interpret the file. Some of the parameters in this section are optional and are not needed for geometry evaluation. Commonly, at least twenty-five parameters (or "records") can be specified. The records include information about the preprocessing system and other information that can be used to identify and process the file. The other record types are of more use to any receiving systems and include such information as the precision used for different data types in the current representation, model space scale (and unit specification), user intended resolution, special system-specific delimiter characters for the various data sections, and other various information to assist in the processing of the file. Depending on the generating system, any number of these may be omitted and are then specified by the receiving program. A listing of all possible Global parameters (with corresponding descriptions) is given in Appendix A.

The next standard data section is the Directory Entry section (DE), which contains a parameter and attribute listing for every entity in the model. Each parameter entry contains up to twenty fields which are specific for each entity type. The field entries may be descriptive of the current entity or may be a pointer to another directory entry. The pointers may provide dependency relationships (child/parent), grouping, transformation, or other information types. One feature of key importance contained in the Directory Entry section is a description of entity relationships that enable the definition of higher level, grouped features. An example could be a surface element which is defined by subordinate features such as boundary curves, NURBS surfaces, a local coordinate system, trimmed elements, transformation entities, and any other necessary geometric

data. In addition to the information provided in the field sections, the Directory Entry section may also be used by the receiving system to quickly locate and reference needed entities in the Parameter Data section. The conversion program parses the DE section using the default and special-type delimiters specified in the Global section. This information is them evaluated to establish a record of grouped high-level features and also linking pointers between entities. This enables proper feature evaluation from the Parameter Data section. The directory listing also enables the program to quickly search through the file for all necessary entities to define a model feature. The directory section is much more compact than the Parameter Data section, so the time spent searching for targeted entities is significantly reduced by first searching the directory section. (to establish data locations)

The majority of the IGES file contains data in the Parameter Data (PD) section. This section contains the specific geometric data for all model entities as well as specific descriptory information for non-geometric entities. Each entity listing in this section has two parameter fields which precede any geometric data for the entity. The purpose of the first parameter field is to describe the associative relationships of the current entity to previously defined entities. The second set of parameter fields contains pointers for specific properties of the current entity type [IGES, 1999]. The remaining data in the entity listing provides any additional information needed to process the entity. The number of fields in this listing can vary significantly based on the type of geometric element being described. In general, the Parameter Data section comprises approximately 75% of an IGES file. The conversion utility processes this information in a feature based manner. Therefore, the grouping entities are used to evaluate all

necessary attributes for a given feature sequentially (such as the previously described surface feature), and then this information is evaluated and exported to the STARS geometry file. Due to the size of certain complex geometry files, storage of all entity parameters for later processing is memory and time inefficient. As noted previously, the data from the Directory Entry section is useful in locating all necessary components of a model feature. Some of the geometric entity types that may be represented in the Parameter Data section include [IGES, 1999]:

- Circular Arc -----Type Number: 100
- Composite Curve -----Type Number: 102
- Conic Arc -----Type Number: 104
- Copious Data -----Type Number: 106
    - Linear Path
    - Simple Closed Planar Curve
- Plane -----Type Number: 108
- Line -----Type Number: 110
- Parametric Spline Curve -----Type Number: 112
- Parametric Spline Surface -----Type Number: 114
- Point -----Type Number: 116
- Ruled Surface -----Type Number: 118
- Surface of Revolution -----Type Number: 120
- Tabulated Cylinder -----Type Number: 122
- Transformation Matrix -----Type Number: 124
- Flash -----Type Number: 125

- Rational B-Spline Curve -----Type Number: 126
- Rational B-Spline Surface -----Type Number: 128
- Offset Curve -----Type Number: 130
- Offset Surface -----Type Number: 140
- Boundary -----Type Number: 141
- Curve on a Parametric Surface -----Type Number: 142
- Bounded Surface -----Type Number: 143
- Trimmed Parametric Surface -----Type Number: 144

An example of a parametric field definition is given in Appendix B.

The final section in the IGES file is the Terminate section. The Terminate section includes up to ten data fields. These fields contain information which varies depending on the generating system. They usually contain data on the number of entries in each major section of the IGES file and the number of entities included. This provides a further measure to error check the interpretation of the file by the receiving system and determine if the STARS conversion was successful.

3.3.2 Entity Processing and STARS File Creation

The STARS geometry file (surface file) consists of two primary geometry sections (with other support sections included). The first contains all model curve features, and the second contains all surface features (including boundary curve specifications and curve directions to define surface normal vectors).

The conversion utility processes and exports the needed entities for each of these sections independently. So, the entities for the curve feature section are evaluated and

exported prior to evaluation of the surface feature section. The method of evaluation for each entity varies based on the entity type. For example, the parameters for surface feature components, such as Bezier and NURBS surfaces, are evaluated utilizing the methods discussed in Section 3.2.5. Other entity types require specialized evaluation methods specified in the IGES standard. An example entity definition is given in Appendix B (including the directory and parameter definitions). The processed features are exported to the STARS surface file in the required format.

Two additional support files are also created to enable STARS analyses. The first file is the boundary condition file (or BCO file). This file contains boundary definitions for all defining model curves and surfaces. By default, these are created (specified) as non-singular, solid entities. The converter GUI includes a curve and surface number viewer which enables the user to easily locate any desired curves/surfaces in order to modify its boundary definition. In the case that the flow is specified as external to the model domain, an outer boundary domain box (with user specified dimensions) is created. The boundary conditions for this domain box are appropriately specified for free flow (no symmetry planes). The second support file is the background mesh file. This file can be used to specify the mesh spacing size throughout the flow domain. The background mesh domain is created based on the user geometry, and the spacing for this (the maximum spacing in the domain) is taken as an input from the user. In order to refine the surface mesh in desired regions (and the resulting volume grid), the user may add source definitions to this file (as described in the STARS reference documents).

3.3.3 User Input and Graphical User Interface

The conversion utility was created as a module to be run with input from a GUI program. The utility was also written to allow the option of using command prompt input following the current implementation of many STARS applications.

The GUI serves two primary purposes. The first (and principal) is the specification of all parameters for the geometry converter program and supporting STARS files. The user must specify the direction of the model surface normals in order to define the domain of the flow. If the model represents a chamber or channel flow, then inward normals can be selected and the solution domain will be confined to the model body. If outward normals are selected, then the body is treated as a solid model and an outer "boundary box" will be added to the geometry file. In this case, the user may specify the boundary spacing from the edges of the model geometry. The number of divisions per surface patch may be specified for each parametric direction. This allows the user to increase the number of nodes generated for each surface representation. This may aid in adjusting the resolution of a surface and addressing any problems that occur during the meshing/gridding process. The final input for the geometry file ("surface" file) includes tolerances and precision for the geometry data. This allows the user to specify the precision for all curve and surface nodes and may help in addressing any continuity difficulties. An additional input section enables quick creation of a flow property (CON) file. All solution properties may be specified in this section. After utilizing both of these input sections, all files necessary to complete a steady state solution are complete. An example of the interface is shown in Figure 35.

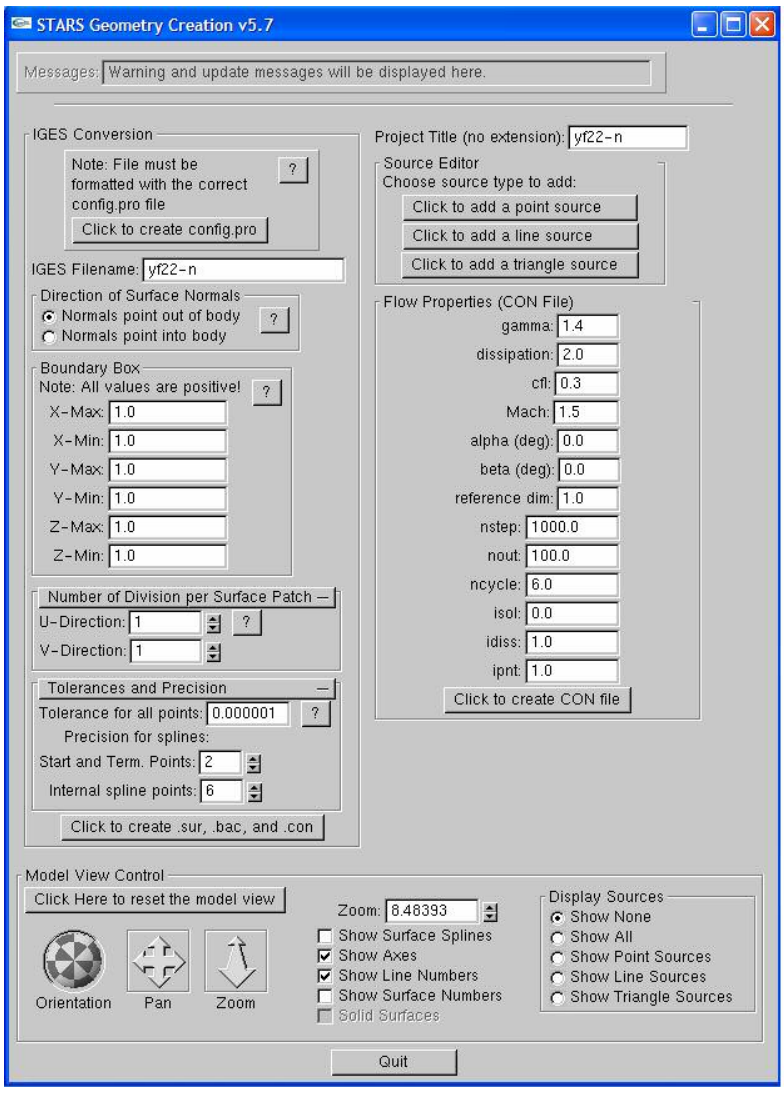

Figure 35: Converter User Interface (utilizing GLUI [Rademacher, 1999])

The second purpose of the GUI is to display the newly created model and provide features to assist in any modifications. The features include a curve and surface number display to aid in the specification of boundary conditions. Another feature is a source viewer which allows the user to selectively view point, line, or triangle sources (or all simultaneously) that have been defined in the background (BAC) file. This feature can be very helpful for test cases that require many source placements (to verify the current source definitions and to ensure that the model contains proper grid spacing at critical

locations). Another option is to display the defining surface splines for all model surfaces. This can be useful for determining the resolution of the surfaces and for locating potential errors in the geometry specification. The following figures illustrate examples of the viewing features just described.

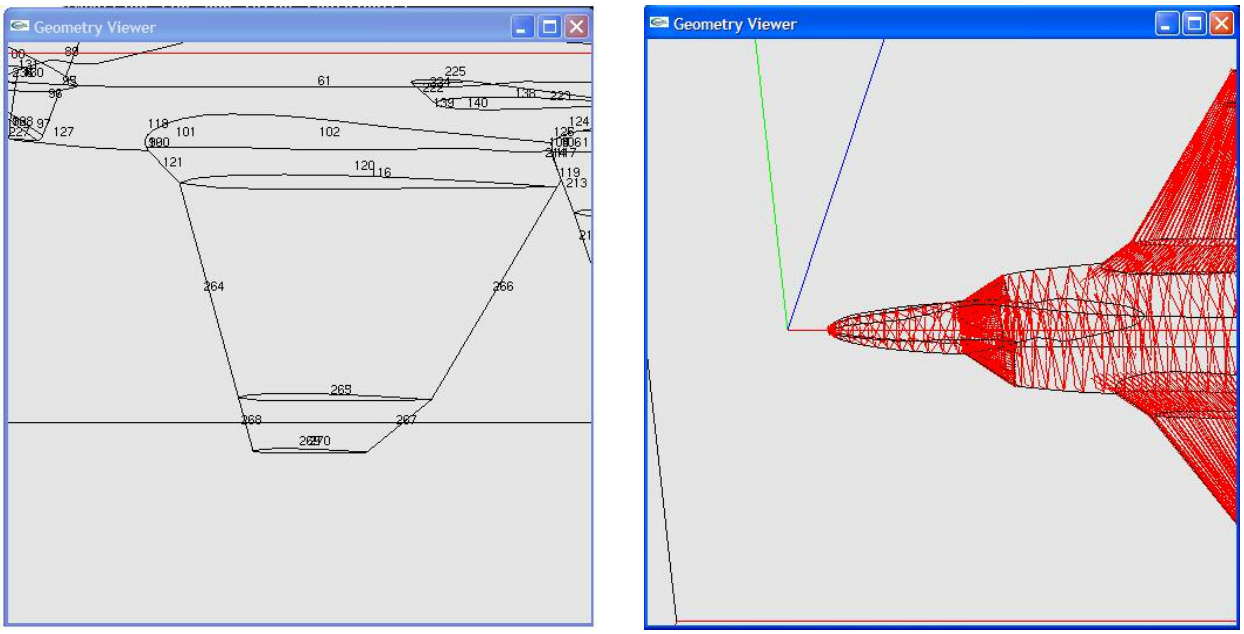

Figure 36: Line Number Display (left) and Surface Spline Display (right)

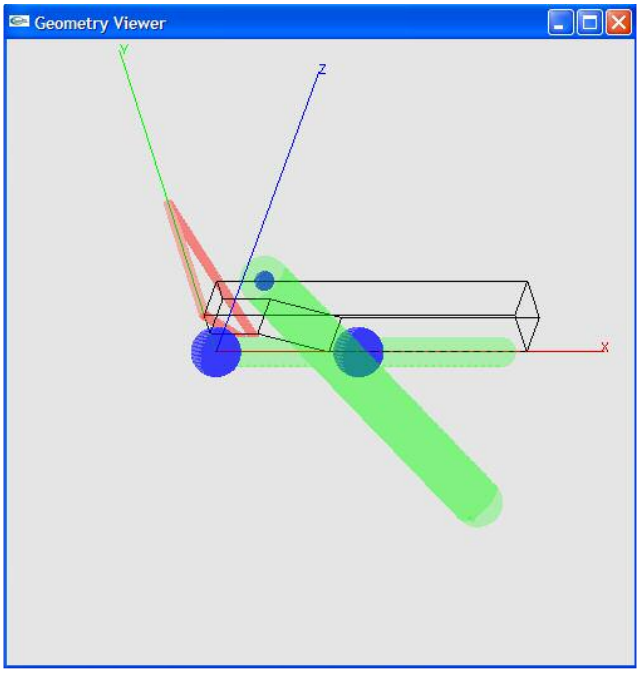

Figure 37: Source Display

CHAPTER 4

RESULTS AND EXAMPLES

4.1 Verification Results

Verification of the geometry models produced by the conversion utility was conducted in two phases. The initial phase examined the accuracy of the representation of individual surfaces and surface boundaries produced by the utility. The second phase examined the accuracy of multi-surface entities. This was conducted in order to examine surface-surface intersections as well as the ability to accurately model a complete body.

The verification for the utility required the examination of each surface export type and the included surface boundary splines. The method for completing this task involved the comparison of nodes on the original CAD surface geometry and calculated nodes on the converted STARS surface. After generating a STARS surface file from an IGES representation, the x,y,z coordinates of nodes on the surface were used to create independent datum points in the model space of the original Pro/Engineer model. The error for each STARS node could then be determined by using Pro/Engineer analysis features to determine the distance (normal to the surface) from the datum point to the original CAD geometry surface. An example of a surface used for NURBS surface verification is given in Figure 38.

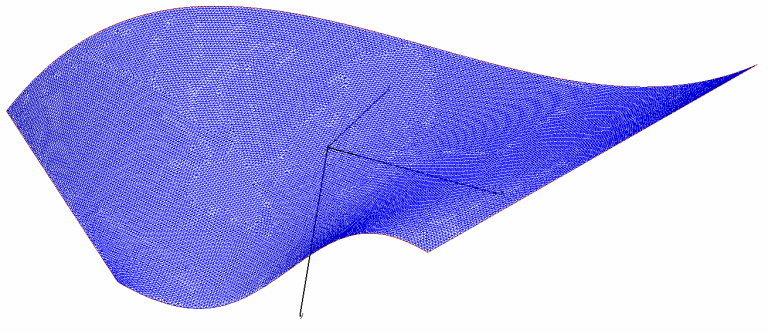

Figure 38: Example Surface Used for Verification

The datum points were then projected onto the surface to determine the coordinates of the corresponding Pro/E surface node. Using this datum point method for all points in the STARS surface model (including boundary and intersection points), a set of error values could then be generated across the original surface. A graphical representation of error values across the surface can then be created to aid in analysis, such as the Excel plot shown in Figure 39.

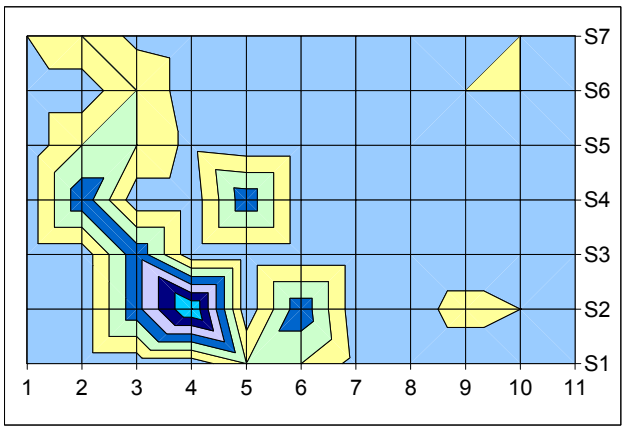

Figure 39: Example Surface Error Plot (shaded areas represent error variation across the surface)

After analyzing multiple test cases for differing surface types, average error values were determined. It should be noted that these are averages of the largest errors for each

individual test case (not over a complete surface). The boundary splines and/or intersection lines were included as part of each surface for evaluation purposes. These are reported as percent errors between the coordinates of the two surfaces. Using this method, the surface type with the least average variation error was the NURBS surface (average percent error of 0.06%) and the power based surface had the largest average error (average percent error of 0.1%).

4.2 Test Cases

The conversion utility has been used to produce files for a number of STARS test cases. Some of the test cases are briefly summarized below with various examples of intermediate meshes and solution plots.

4.2.1 Test Case 1: YF-22

The initial geometry that was received is shown below in Pro/Engineer prior to conversion.

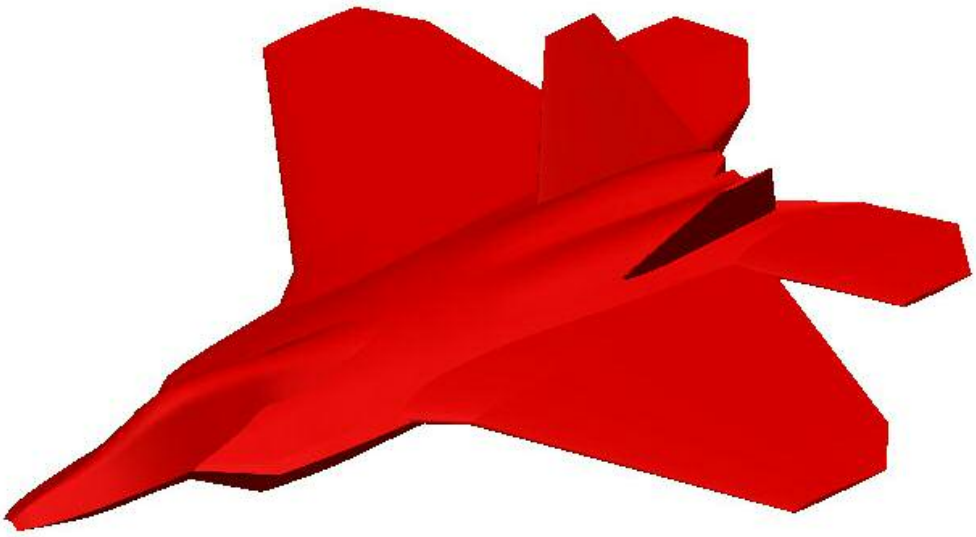

Figure 40: Pro/Engineer IGES YF-22

This was utilized to create STARS files (SUR, BAC, BCO) to be used to compute a number of steady-state solutions. The resulting STARS geometry file contained 290 curves and 104 surfaces. One example of a surface grid is shown in Figure 41, while an example Mach plot is shown in Figure 42.

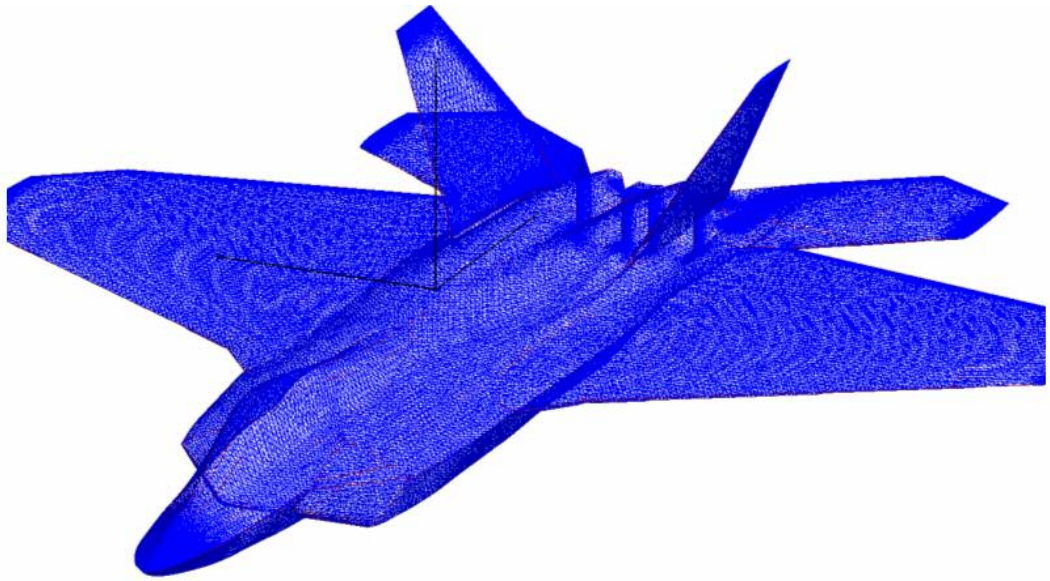

Figure 41: YF-22 Surface Grid

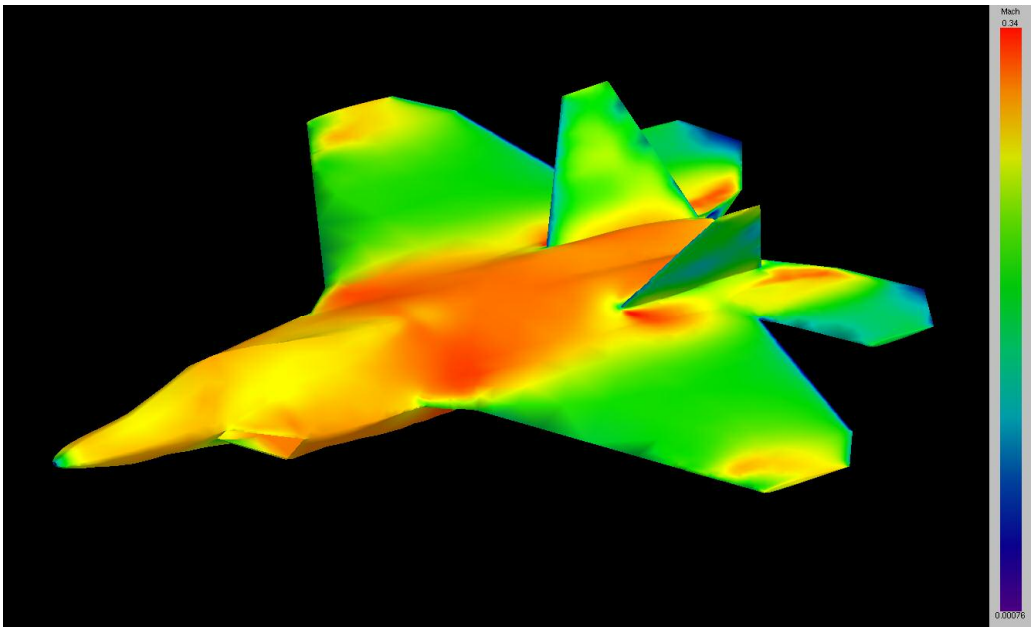

Figure 42: Mach Distribution at M = 0.3

4.2.2 Test Case 2: OSU Design/Build/Fly Aircraft

The Pro/Engineer models for the 2006/2007 Design/Build/Fly competition were converted to STARS files in order to complete a steady state alpha sweep. This analysis was carried out for both the Orange Team and the Black Team aircraft. Because of the relative speed at which the test case files could be prepared, it was possible to carry out multiple design evaluations in a short period of time. For example, the placement of the Black Team main wing and connecting plate shapes (as well as the span of the wing, canard, and horizontal tail) could be easily and quickly modified in Pro/Engineer and then exported to the conversion utility. This allowed for a fairly quick iterative analysis process that would have required much more effort and time to complete manually (by direct modification of the geometry files). These test cases provide an example of the potential usefulness of the converter application to rapidly iterate through many different design configurations. Examples of the surface grids and solution plots are shown in the following figures.

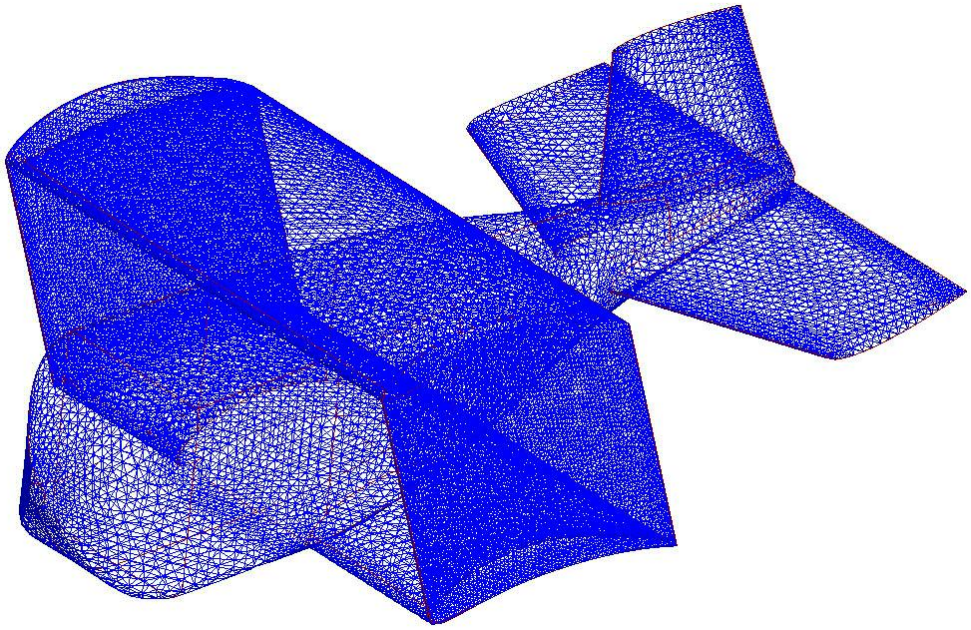

Figure 43: 2007 Orange Team surface mesh

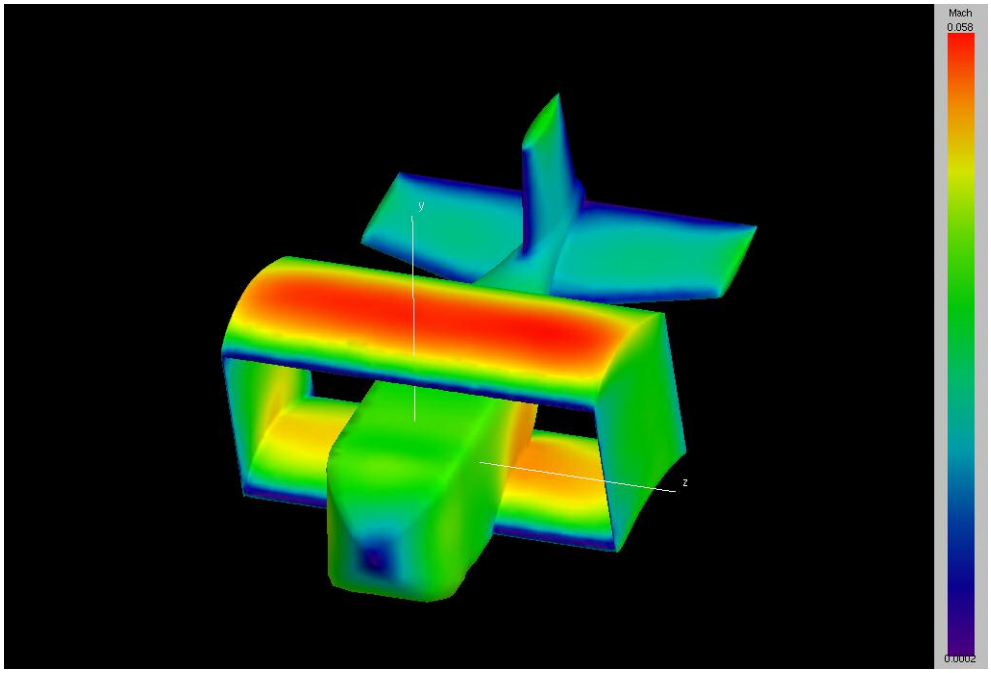

Figure 44: Example 2007 Orange Team Mach solution plot

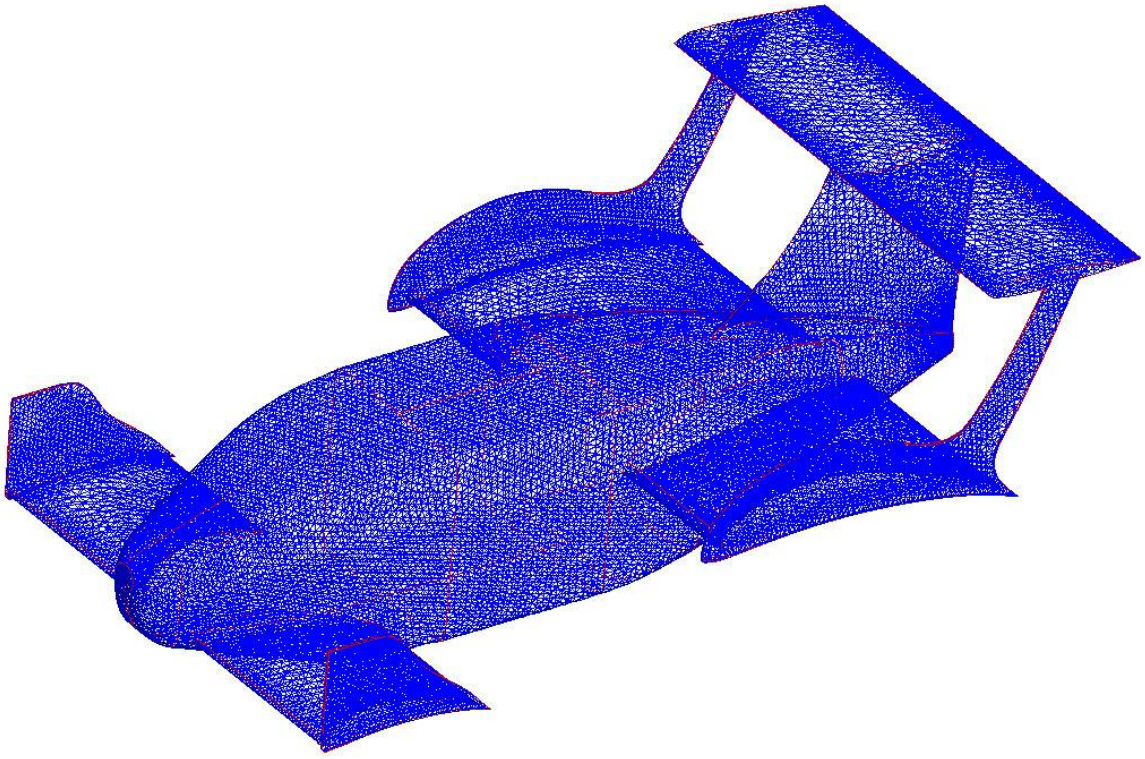

Figure 45: Original 2007 Black Team surface mesh

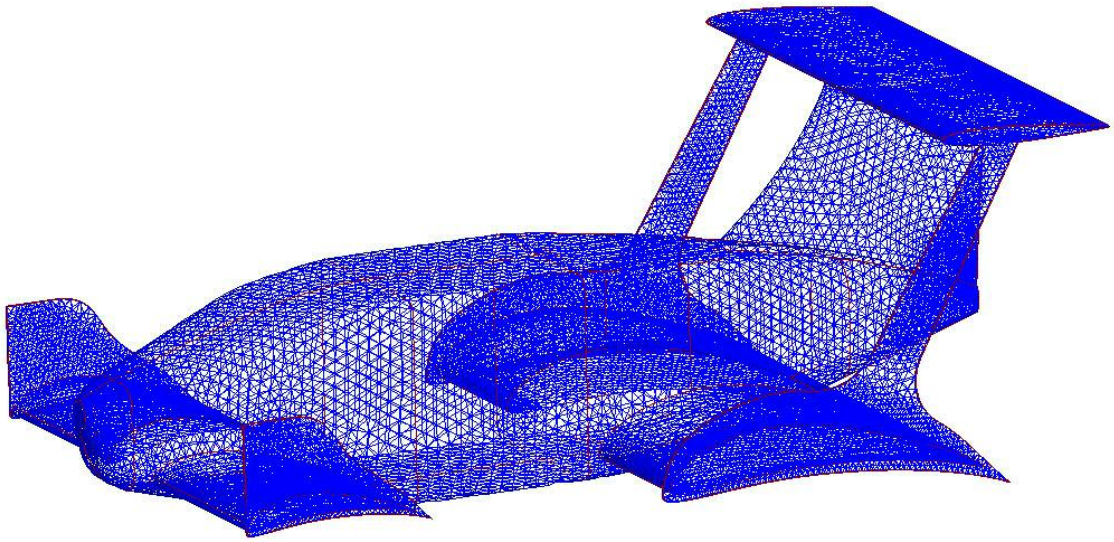

Figure 46: 2007 Black Team surface mesh (low wing placement)

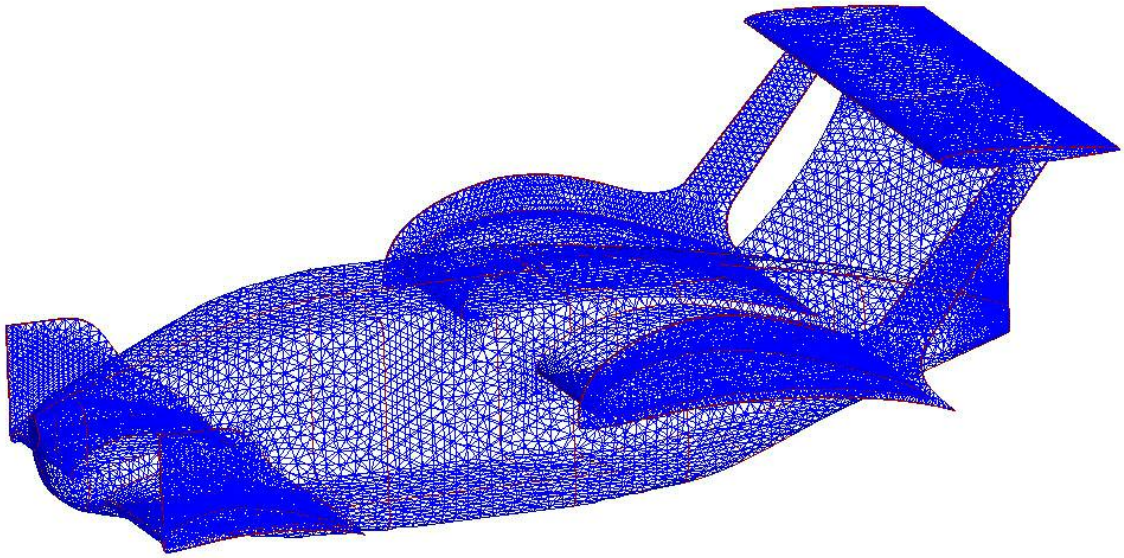

Figure 47: 2007 Black Team surface mesh (forward wing, reduced canard span)

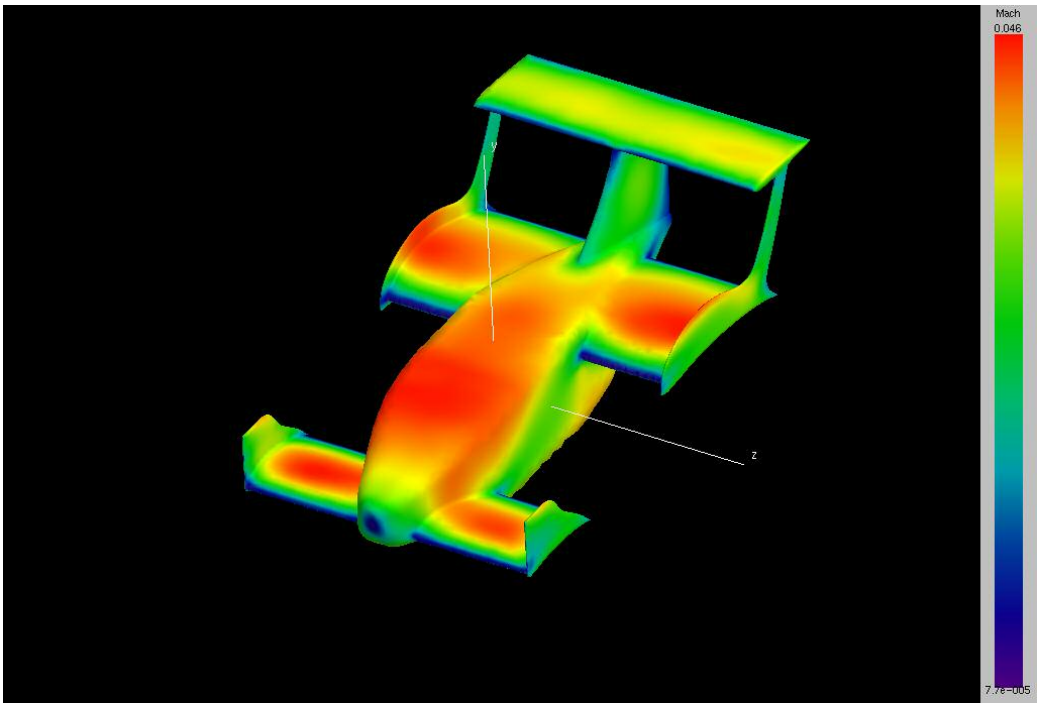

Figure 48: 2007 Black Team Mach solution plot

4.2.3 Test Case 3: SBC-UAV

The SBC UAV is a fuel cell aircraft which has been designed by California State University. A STARS model was created from an IGES file to aid in their analysis. Figure 49 illustrates an example mesh generated for this aircraft.

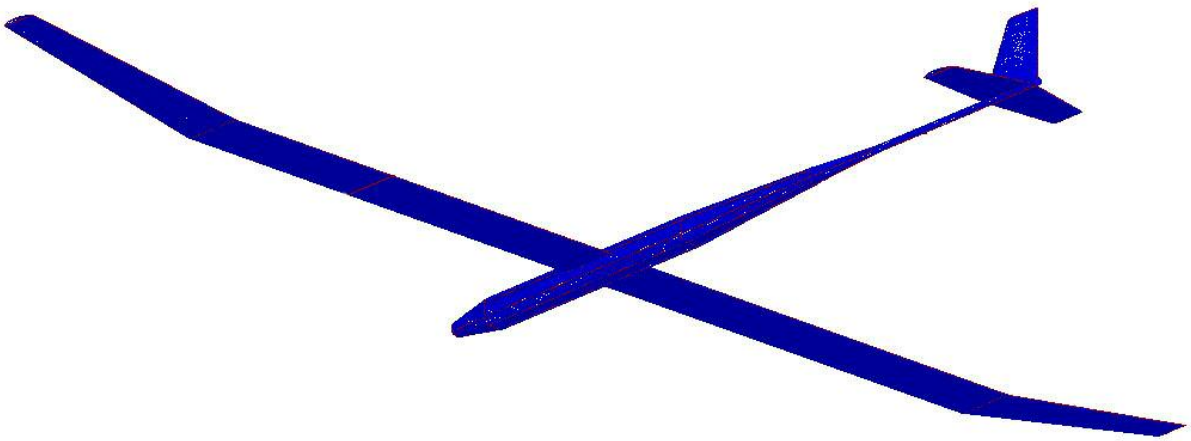

Figure 49: SBC-UAV surface mesh

4.2.4 Test Case 4: GHV

The GHV is an airbreathing hypersonic flight vehicle. Two different geometries were analyzed (as shown in Figure 50 and 51). The first model created (Model 1) possessed a simple flat inlet B.C. (far-field). The model was then modified to include a new engine inlet geometry (Model 2) which was based on 2-D compressible flow theory (ref.: California State University).

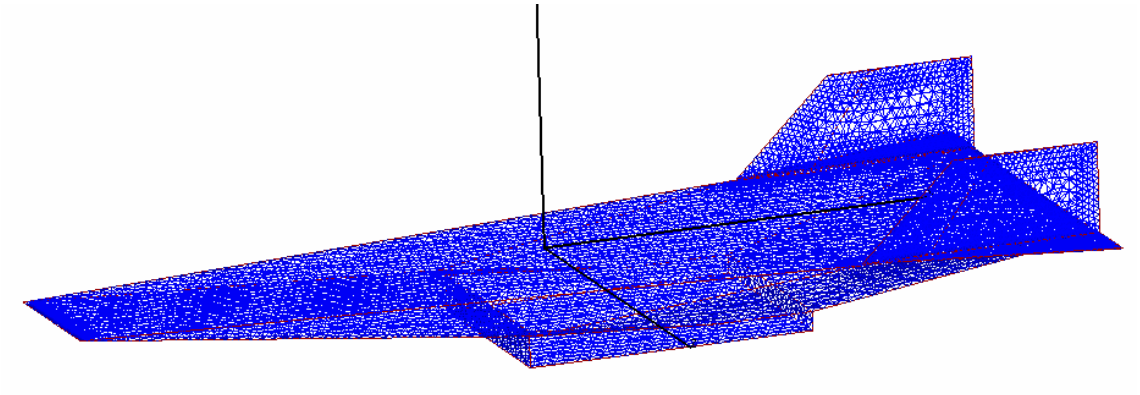

Figure 50: GHV (Model 1) surface mesh

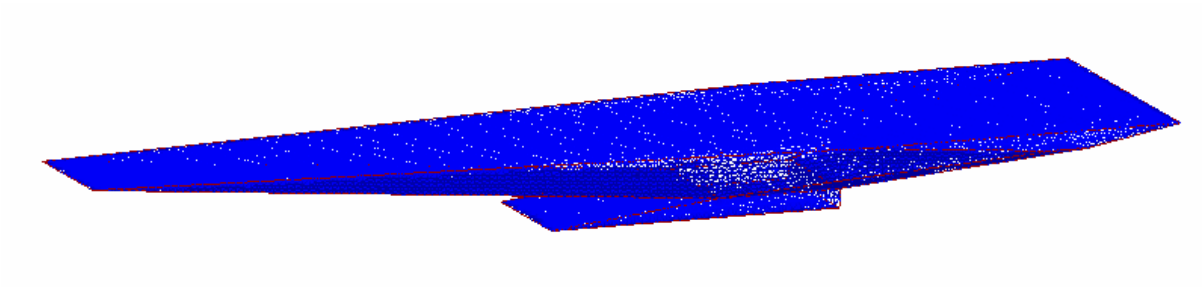

Figure 51: GHV (Model 2) surface mesh

CHAPTER 5

SUMMARY AND FUTURE DEVELOPMENT

5.1 Summary

The development of a STARS geometry creation utility proved to provide an efficient and effective means for completing an important step in the STARS analysis process. Project efficiency can be greatly increased with the use of this conversion utility. For example, a simple aircraft surface file can take approximately 2 1/2 days to complete and debug manually. Using the converter approach, the same SUR file could take less than an hour to generate, with most of this time spent creating the CAD model (the actual conversion process takes less than 1 min. on average). For much more complex geometries, such as the YF-22, the time saved by SUR file generation is much more substantial. The simple aircraft contained 88 lines and 17 surfaces, while the YF-22 contained 290 lines and 104 surfaces (which had much more complex shape and curvature definitions than the simple aircraft). It is also not uncommon to have to spend quite some time searching for errors in a complex SUR definition created manually. The converter almost eliminates this difficulty. Another advantage is that IGES files (or other CAD file types) may be obtained from any source and then converted to the proper IGES format in Pro/E, making it possible to use existing project geometries or receive geometries from third parties.

5.2 Future Development

Two primary areas are recommended for future development. The first is the integration of this utility with a gridding utility. A number of papers exist which propose methods for creating grids from common CAD entity types (NURBS surfaces, etc.). One such method involves "channeled" gridding, in which surface patches are used to define channels through the flow domain (which may intersect with each other in the case of a concave body). Techniques such as this may help to streamline the process as the initial grid front may be directly created from the parametric entity definitions (possibly resulting in a more effective initial front).

The second recommendation is for the implementation of optimization algorithms in this utility. This would be useful for basic design iterations and could be highly effective if coupled with a p-type solution convergence scheme. The method of definition/evaluation of the standard entities (Bezier curves, etc.) makes local shape modification and morphing a relatively easy process. Therefore, morphing end plates or other complex shapes as model components are transversed through a design range should be a straightforward task.

BIBLIOGRAPHY

Babcock, D. A., "Aircraft Stability Derivative Estimation from Finite Element Analysis," Master's Thesis, Department of Mechanical and Aerospace Engineering, Oklahoma State University, 2004.

Bezier, P. E., *Numerical Control: Mathematics and Applications*, John Wiley, New York, 1972.

Boehm, W., "Rational Geometric Splines," Computer Aided Geometric Design, Vol. 4, 1987, pp. 67-78.

Brown, J. L., Worsey, A. J., "Problems with Defining Barycentric Coordinates for the Sphere," Mathematical Modeling and Numerical Analysis, Vol. 26, 1992, pp. 37-49.

Casale, M.S., "Free-Form Solid Modeling with Trimmed Surface Patches," IEEE Computer Graphics and Applications, Vol. 7, 1987, pp. 30-43.

Cowan, T. J., "Finite Element CFD Analysis of Super-Maneuvering and Spinning Structures," PhD Dissertation, Department of Mechanical and Aerospace Engineering, Oklahoma State University, 2003.

De Boor, C., *A Practical Guide to Splines*, Springer, New York, 1978.

Farin, G., *Curves and Surfaces for Computer Aided Geometric Design*, 3rd Ed. Academic Press, Inc., San Diego, 1993.

Floriani, L., "Surface Representations Based on Triangular Grids," The Visual Computer, Vol. 3, 1987, pp. 27-50.

Foley, J., *Computer Graphics: Principles and Practice*, Addison-Wesley, Reading, MA, 1990.

Gordon, W., "Blending Function Methods of Bivariate and Multivariate Interpolation and Approximation," Society for Industrial and Applied Mathematics Journal of Numerical Analysis, Vol. 8, 1971, pp. 155-176.

Gupta, K. K., "STARS – An Integrated, Multidisciplinary, Finite-Element, Structural, Fluids, Aeroelastic, Aeroservoelastic Analysis Computer Program," NASA TM-4795, April 2001.

Hagen, H., *Topics in Surface Modeling*, Society for Industrial and Applied Mathematics, Philadelphia, 1992.

Hoschek, J., Lasser, D., *Computer Aided Geometric Design*, A K Peters, Wellesley, 1993.

Jie, T., "A Geometric Condition for Smoothness Between Adjacent Rational Bezier Surfaces," Computers in Industry, 1990, p. 355-360.

Joe, B., "Multiple Knot and Rational Cubic β-Splines," ACM Transactions on Graphics, Vol. 8, 1989, pp. 100-120.

Kilgard, M., "The OpenGL Utility Toolkit (GLUT) Programming Interface," API Version 3, Silicon Graphics, Inc., November, 1996.

Kjellander, J. A., "Smoothing of Bicubic Parametric Surfaces," Computer Aided Design, Vol. 15, 1983, pp. 288-293.

Koparkar, P. A., Mudur, S. P., "A New Class of Algorithms for the Processing of Parametric Curves," Computer Aided Design, Vol. 15, 1983, pp. 41-45.

Lane, J. M., Riesenfeld, R. F., "A Theoretical Development for the Computer Generation of Piecewise Polynomial Surfaces," IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI), Vol. 2, 1980, pp. 35-45.

Manning, J. R., "Continuity Conditions for Spline Curves," Computer Journal, Vol. 17, 1974, pp. 181-186.

Mastin, C. W., Parameterization in Grid Generation," Computer Aided Design, Vol. 18, 1986, pp. 22-25.

Mullenheim, G., "On Determining Start Points for a Surface / Surface Intersection Algorithm," Computer Aided Geometric Design, Vol. 8, 1991, pp. 401-407.

"The Initial Graphics Exchange Specification (IGES)," IGES / PDES Organization, Gaithersburg, MD, 1999.

Nasri, A. H., "Polyhedral Subdivision Methods for Free-Form Surfaces," ACM Transactions on Graphics, Vol. 6, 1987, pp. 29-73.

Piegl, L., Tiller, W., *The NURBS Book*, Springer, 1995.

Rademacher, P., "GLUI: A GLUT-Based User Interface Library," Version 2.0, The University of North Carolina at Chapel Hill, June, 1999.

Riesenfeld, R.F., "Applications of B-Spline Approximation to Geometric Problems of Computer-Aided Design," PhD Dissertation, Syracuse University, 1973.

"STEP Application Handbook," ISO 10303, Version 3, SCRA-Institute for Solutions Generation, North Charleston, June, 2006.

Warren, J., "Free-Form Blending: A Technique for Creating Piecewise Implicit Surfaces," Society for Industrial and Applied Mathematics, SIAM Conference on Geometric Design, November, 1989 Tempe, Arizona.

APPENDIX A:

IGES GLOBAL PAREMETERS

The following parameters are used in the Global section of the standard IGES file. This data includes all information for the initial preprocessing of the file by the receiving system. The following information is from The Initial Graphics Exchange Specification manual produced by the IGES / PDES Organization [1999].

Parameter Identifier	Item Description
1	Parameter delimeter character
2	Record delimiter character
3	Product identification from generating system
4	File Title
5	Generating System ID
6	Preprocessor version
7	Number of binary bits for integer representation
8	Maximum power of ten representable in a single precision floating point number on the sending system
9	Number of significant digits in a single precision floating point number on the sending system
10	Maximum power of ten representable in a double precision floating point number on the sending system
11	Number of significant digits in a double precision floating point number on the sending system
12	Product identification for the receiving system
13	Model space scale
14	Unit flag
15	Units

16	Maximum number of line weight gradations (1-32768). Refer to the Directory Entry Parameter 12
17	Width of maximum line weight in units. Refer to the Directory Entry Parameter
18	Date & time of exchange file generation 13HYYMMDD.HHNNSS
19	Minimum user-intended resolution or granularity of the model expressed in units defined by Parameter 15
20	Approximate maximum coordinate value occurring in the model expressed in units defined by Parameter 15
21	Name of author
22	Author's organization
23	Integer value corresponding to the version of the Specification used to create the file
24	Drafting standard in compliance to which the data encoded in this file was generated
25	Date and time the model was created or last modified, whichever occurred last HYYMMDD.HHNNSS

APPENDIX B:

EXAMPLE IGES ENTITY DEFINITION

The following is an example of the general format for an IGES entity. The specific entity defined below is the power-basis parametric spline curve entity (defined as type 112). The first section is the Directory Entry definition section and the second section is the Parameter Data definition section. While this is a general representation of the format for an IGES entity, the data and specific definition may vary significantly based on the entity type under consideration. This information is from The Initial Graphics Exchange Specification manual produced by the IGES / PDES Organization [1999]. Please refer to this reference for more detailed and additional explanations.

Directory Entry Section:

PARAMETRIC SPLINE CURVE ENTITY (TYPE 112)

Directory Entry Section

1	2	3	4	5	6	7	8	9	10
Entity Type Number 112	Parameter Data ()	Structure <n:a:>	Line Pattern #;)	Level #;)	View 0;)	Formation Matrix 0;)	Label Display 0;)	Status Number **	Sequence Number D #
11	12	13	14	15	16	17	18	19	20
Entity Type Number 112	Line Weight #	Color Number #;)	Parameter Line Count #	Form Number 0	Reserved	Reserved	Entity Label	Entity Subscript #	Sequence Number D # + 1

Parameter Data Section:

Index__ Name____ Type___ Description___

1 CTYPE Integer Spline Type:
- 1=Linear
- 2=Quadratic
- 3=Cubic
- 4=Wilson-Fowler
- 5=Modified Wilson-Fowler
- 6=B Spline

2 H Integer Degree of continuity with respect to arc length

3 NDIM Integer Number of dimensions:
- 2=planar
- 3=nonplanar

4	N	Integer	Number of segments
5	T(1)	Real	First break point of piecewise polynomial
..	.	.	
.	..	..	
5+N	T(N+1)	Real	Last break point of piecewise polynomial
6+N	AX(1)	Real	X coordinate polynomial
7+N	BX(1)	Real	
8+N	CX(1)	Real	
9+N	DX(1)	Real	
10+N	AY(1)	Real	Y coordinate polynomial
11+N	BY(1)	Real	
12+N	CY(1)	Real	
13+N	DY(1)	Real	
14+N	AZ(1)	Real	Z coordinate polynomial
15+N	BZ(1)	Real	
16+N	CZ(1)	Real	
17+N	DZ(1)	Real	
..	.	.	
.	..	..	Subsequent X, Y, Z polynomials concluding with the twelve coefficients of the Nth polynomial segment.
6+13*N	TPX0	Real	X value
7+13*N	TPX1	Real	X first derivative
8+13*N	TPX2	Real	X second derivative/2!
9+13*N	TPX3	Real	X third derivative/3!
10+13*N	TPY0	Real	Y value
11+13*N	TPY1	Real	Y first derivative
12+13*N	TPY2	Real	Y second derivative/2!
13+13*N	TPY3	Real	Y third derivative/3!
14+13*N	TPZ0	Real	Z value
15+13*N	TPZ1	Real	Z first derivative
16+13*N	TPZ2	Real	Z second derivative/2!
17+13*N	TPZ3	Real	Z third derivative/3!

VITA

Robert James Fischer

Candidate for the Degree of

Master of Science

Thesis: PARAMETRIC GEOMETRY CREATION METHODOLOGY AND UTILITY FOR THE STARS CFD ANALYSIS PACKAGE

Major Field: Aerospace Engineering

Biographical:

Personal Data: Born on October 7, 1980 in Midwest City, Oklahoma. Son of James L. Fischer and Vickie J. Fischer.

Education: Graduated from Booker T. Washington High School in May of 1999. Received Bachelor of Science Degrees in Mechanical Engineering and Aerospace Engineering from Oklahoma State University, Stillwater, Oklahoma, in December 2004. Completed the Requirements for the Master of Science degree in Mechanical Engineering at Oklahoma State University in July 2007.

Experience: NASA Undergraduate Student Research Program (USRP), 2004; Teaching Assistant, OSU Mechanical and Aerospace Engineering Department, 2004-2006; Research Assistant, OSU Mechanical and Aerospace Engineering Department, 2004-2007.

Professional Memberships: American Society of Mechanical Engineers, American Institute of Aeronautics and Astronautics, Pi Tau Sigma, Sigma Gamma Tau.

Name: Robert James Fischer Date of Degree: July, 2007

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: PARAMETRIC GEOMETRY CREATION METHODOLOGY AND UTILITY FOR THE STARS CFD ANALYSIS PACKAGE

Pages in Study: 93 Candidate for the Degree of Master of Science

Scope and Method of Study: Differing methodologies and approaches for the creation of STARS CFD test case geometry models and support files were examined. The current methods for model geometry creation have been identified by past researchers as an area needing improvement. The current methods limit the complexity of the model used, can require a significant amount of project time (in some cases, more time than the CFD analysis), and require an experienced user in order to successfully employ.

Methods of geometric data transfer, storage, and processing were examined for their applicability and usefulness in the STARS solution procedure. An approach which would utilize existing CAD geometry created with commercial software was selected in order to allow the importation of existing and third party models and to facilitate the quick creation of models from a known CAD application. The IGES (Initial Graphics Exchange Specifications) standard was chosen as the data transfer type due to its wide spread usage and efficient processing techniques. Various methods for geometric entity processing and evaluation were examined.

A CFD model creation utility was then developed for STARS. The utility converts existing IGES files into the set of files needed for a general CFD analysis. A graphical user interface was also created in order to aid the user in the specification of geometric, meshing, and solution parameters.

Findings and Conclusions: The development of a STARS geometry creation utility proved to provide an efficient and effective means for completing an important step in the STARS analysis process. Project efficiency can be greatly increased, and even inexperienced users may now quickly create the files needed for the analysis of a complex model geometry.

Several test cases were utilized in order to examine the effectiveness of the utility. Among these were the F-22, two OSU competition aircraft, and a hypersonic flight vehicle. The utility allowed for the rapid creation of STARS files for these models in what would have taken several days at a minimum to create with previous methods.

ADVISOR'S APPROVAL: Dr. Andrew S. Arena, Jr.