Robust Evaluation of Differential Geometry Properties using Interval Arithmetic Techniques

by

Chih-kuo Lee

B.A. in Mathematics

The Citadel, SC, USA 2001

Submitted to the Department of Ocean Engineering

in partial fulfillment of the requirements for the degree of

Master of Science in Naval Architecture and Marine Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2005 [June 2005]

Author..... .....

Department of Ocean Engineering

May 6, 2005

Certified by.....

Nicholas M. Patrikalakis, Kawasaki Professor of Engineering,

Professor of Mechanical and Ocean Engineering

Thesis Supervisor

Accepted by. .....

Michael S. Triantafyllou, Professor of Mechanical and Ocean Engineering

Chairman, Department Committee on Graduate Students

Robust Evaluation of Differential Geometry Properties using Interval Arithmetic Techniques

by

Chih-kuo Lee

Submitted to the Department of Ocean Engineering
on May 6, 2005 in partial fulfillment of the
requirements for the degree of Master of Science in
Naval Architecture and Marine Engineering

Abstract

This thesis presents a robust method for evaluating differential geometry properties of sculptured surfaces by using a validated ordinary differential equation (ODE) system solver based on interval arithmetic. Iso-contouring of curvature of a Bezier surface patch, computation of curvature lines of a Bezier surface patch and computation of geodesics of a Bezier surface patch are computed by the Validated Numerical Ordinary Differential Equations (VNODE) solver which employs rounded interval arithmetic methods. Then, the results generated from the VNODE program are compared with the results from Praxiteles code which uses non-validated ODE solvers operating in double precision floating point arithmetic for the solution of the same problems. From the results of these experiments, we find that the VNODE program performs these computations reliably, but at increased computational cost.

Thesis Supervisor: Nicholas M. Patrikalakis, Kawasaki Professor of Engineering
Professor of Mechanical and Ocean Engineering

Acknowledgments

First, I would like to thank Professor Nicholas M. Patrikalakis, my advisor, who guided and supported me throughout the entire thesis process with his great patience and knowledge. Without his expert advice, a careless person like me would never be able to finish his thesis.

I would also like to thank Harish Mukundan for the great amount of knowledge he has taught me about the VNODE program, and the help from him all the time. I also want to thank Design Laboratory staff Dr. W. Cho who helped me to solve various problems in the Praxiteles program and Mr. F. Baker who installed the entire VNODE program for me.

Many thanks to Taiwan's Navy for supporting my tuition and living cost in the last two years while I was studying at MIT. And thank all the friends in Boston for their help either in living or in academic issues during the two years.

Finally, I would like to thank my family in Taiwan for their encouragement all the time, especially thank to my beloved wife, Chieh-Yin, who is always being supportive and loving.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Validated methods for initial value problems for ordinary differential equations produce bounds that are guaranteed to enclose the true solution of the problem. The validated numerical ordinary differential equations (VNODE) package [25], an experimental implementation, was developed using standard libraries and component packages. The VNODE program is implemented in the C++ language for computing rigorous bounds on the solution of an initial value problem for a system of ordinary differential equations. Several examples of differential geometry properties of a Bezier surface patch are evaluated by the VNODE program and the results are compared with the results generated from the Praxiteles program version 10.4 which is developed and maintained by members of the Design Laboratory of the Department of Ocean Engineering at the Massachusetts Institute of Technology. Moreover, the data in the u-v parametric space generated from the VNODE program are also mapped into the 3-D space which can show the relations between the evaluation and the surface. All calculations from the VNODE program are performed on a PC running at 1.4 GHz with 896MB of physical memory under the Linux operating system installed in the Design Laboratory of the Department of Ocean Engineering.

1.2 Thesis Overview

The thesis is structured as follows: Chapter 2 briefly introduces the interval arithmetic technique on which the VNODE program is based and which is used in this thesis. In Chapter 3, the definition and basic properties of a Bezier surface patch are reviewed because all the examples presented in this thesis involve a Bezier surface patch. Chapter 4 describes the iso-contouring of curvature and the performance of algorithm, including Gaussian curvature, mean curvature and principal curvatures. In Chapter 5, curvature lines are introduced and two executions of algorithm are described. Chapter 6 describes an initial value problem for geodesic line computation and two executions of the algorithm. The last chapter, Chapter 7, concludes this thesis and recommends directions for future research.

Chapter 2

Interval Arithmetic Techniques: VNODE

In this chapter, the interval arithmetic technique used in the VNODE program (Validated Numerical Ordinary Differential Equations) developed by Nedialkov [23] is described. VNODE is a C++ package for computing rigorous bounds on the solution of an initial value problem for the system of ordinary differential equations [25].

2.1 Interval Arithmetic Operations and Properties

An interval is a set of real numbers defined as:

$$[\,a\,,\,b\,] = \{x \mid a \leq x \leq b\}$$

If [a , b] and [c , d] are two intervals, the interval arithmetic operations are:

[a , b] + [c , d] = [a+c , b+d],

[a , b] - [c , d] = [a-d , b-c],

[a , b] [c , d] = [min(ac,ad,bc,bd) , max(ac,ad,bc,bd)],

[a , b] / [c , d] = [min(a/c,a/d,b/c,b/d) , max(a/c,a/d,b/c,b/d)], where the closed interval [c , d] does not contain zero.

The algebraic properties of interval arithmetic are:

Commutative:

$$[a,b]+[c,d]=[c,d]+[a,b]$$

$$[a,b]\cdot[c,d]=[c,d]\cdot[a,b]$$

Associative:

$$[a,b]+([c,d]+[e,f])=([a,b]+[c,d])+[e,f]$$

$$[a,b]\cdot([c,d]\cdot[e,f])=([a,b]\cdot[c,d])\cdot[e,f]$$

Although interval arithmetic is not distributive [1], the subdistributive law still holds:

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

However, the distributive law will be true for interval arithmetic when [a , b] is a degenerate interval (a = b).

There is also another important property for interval arithmetic operations:

If $[a,b] \subseteq [a1,b1]$ and $[c,d] \subseteq [c1,d1]$,

Then $[a,b]$ function $[c,d] \subseteq [a1,b1]$ function $[c1,d1]$.

function $\subseteq \{ +, -, *, / \}$

The power of interval arithmetic when used on a computer is in computing rigorous enclosures of real operations by including rounding errors in the computed bounds. In order to obtain machine interval arithmetic also known as rounded interval arithmetic, the real intervals are rounded outwards including those errors. For instance, if [a , b] and [c , d] are added, the b + d is rounded up and a + c is rounded down. The intervals computed in machine interval arithmetic always contain the corresponding real intervals because of the outward roundings. Besides the scalar case for these intervals, the arithmetic operations also work for interval vectors and matrices.

2.2 Validated Methods for IVPs for ODEs

Most validated methods for initial value problems (IVPs) for ordinary differential equations (ODEs) are based on Taylor series [25]. There are several reasons as to why the Taylor series approach has been used most: First, the coefficients of Taylor series can be efficiently generated by automatic differentiation. Second, Taylor series have the simple form of error term and such term can be readily bounded using interval arithmetic. Third, the order of the method can be changed easily by adding or deleting Taylor series terms. Finally, the step size can be changed without doing extra work to recompute Taylor series coefficients.

There are two phases in most Taylor series methods [26]. Algorithm one involves validating the existence and uniqueness of the solution, and algorithm two involves computing a tight enclosure. In algorithm one, the limit of the stepsize is restricted to Euler steps. However, one can obtain methods that enable larger stepsizes by using polynomial enclosures [19] or more Taylor series terms to obtain a Taylor series enclosure method [8,22,23]. In algorithm two, when the enclosures are computed, the widths of the computed intervals may decrease. However, this algorithm may work poorly in some cases, when the so-called wrapping effect happens. The wrapping effect is clearly described by Moore [22]. In order to reduce the effects caused by wrapping, Lohner's QR-factorization method [18] which is one of the most successful, general-purpose methods can be used. Recently, besides the Taylor series method, a new scheme which is an interval Hermite-Obreschkoff method [23, 24] has been developed.

Base on this method, the results have smaller local errors, better stability, and require fewer Jacobian evaluations than interval Taylor series methods. However, the extra cost is one matrix inversion and a few matrix multiplications.

2.3 The VNODE Package

Currently, there are three available software packages for computing guaranteed bounds on the solution of an IVP for ODEs [25]. The three packages are:

AWA[17]:

This package is an implementation of Lohner's method [18]. This is also the constant enclosure method for validating the existence and uniqueness of the solution. The stepsize in this package is often restricted to Euler steps by algorithm one. It is written in an extension of Pascal for scientific computing which is called Pascal- XSC [14].

ADIODES[31]:

This package is a C++ implementation of a solver using the constant enclosure method in algorithm one and Lohner's method in algorithm two. The stepsize is also often restricted to Euler steps by algorithm one.

COSY INFINITY[5]:

This package is a Fortran-based code to study and design beam physics systems. High-order Taylor polynomials with respect to time and the initial conditions are used to verify integration of ODEs. Computations are carried out with Taylor polynomials with real floating-point coefficients and a guaranteed error bound for the remainder term. Then, the arithmetic operations and standard functions are executed with such Taylor polynomials as operands. Therefore, by establishing functional dependency between initial and final conditions, the wrapping effect can be reduced [6]. However, working with polynomials is more expensive than working with intervals.

The VNODE package is designed using an object-oriented approach and implemented in C++ [10]. Before this design, the Godess [27] and TIDE [12] packages which are also object-oriented designs offer generic ODE solvers that implement standard methods for IVPs for ODEs. Moreover, Diffpack [16] is another successful object-oriented package for solving partial differential equations. Since VNODE is an object-oriented design, the following are three important object-oriented concepts supported in C++ and used in VNODE [7]:

Data Abstraction:

The software system in the object-oriented model can be viewed as a collection of objects that interact with each other to achieve a desired functionality. The objects are instances of a class which defines the structure and behavior of its objects. By grouping data and methods inside the class and specifying its interface, separating the interface from the implementation can be achieved. Therefore, the data representation and the implementation of the methods of a class can be changed without modifying the software. By doing so, function calls with long parameter lists can be avoided.

Inheritance and Polymorphism:

Inheritance in object-oriented models can allow a derived class to reuse data and functions of its base class. Polymorphism can let a given function apply to different types of objects. They both often are used with abstract classes which define abstract functions implemented in the subclasses.

Operator Overloading:

Operator overloading allows the operators of the language to be overloaded. In order to program interval operations without explicit function calls, a language that supports operator overloading has to be used. Due to the fact that interval-arithmetic operations

have to be coded by using function calls, programming interval-arithmetic expressions will be cumbersome without operator overloading.

The properties of the VNODE package include:

Modularity:

The VNODE package is organized as a set of modules with well-defined interfaces. The implementation of each module is hidden, however, the user is able to modify the implementation if necessary.

Flexibility:

There is a flexibility property in VNODE that allows users to replace a method inside the solver without affecting the rest of the solver. Moreover, methods following the established structure can be added without modifying the existing code.

Efficiency:

The efficiency of a validated solver is determined mainly by the efficiency of the underlying automatic differentiation package, however, there are also some other contributing factors including the efficiency of the interval arithmetic package, the programming language, and the actual implementation of the methods. Although the methods incorporated in VNODE require the computation of high-order Taylor coefficients and Jacobians of Taylor coefficients, they do not have theoretical limits on the size of the ODE system or order of the method. Therefore, VNODE is structured so as to achieve this objective.

There are two packages for automatic differentiation of interval Taylor coefficients for the solution of an ODE and the Jacobians of these coefficients in the VNODE package. These

two packages are FADBAD/TADIFF [3,4] and IADOL-C [13]. In FADBAD/TADIFF package, TADIFF generates Taylor coefficients with respect to time, and then FADBAD computes Jacobians of Taylor coefficients by applying the forward mode of automatic differentiation [29] to these coefficients. The IADOL-C package is an extension of ADOL-C package [11] which computes Taylor coefficients by using the forward mode and their Jacobians by applying the reverse mode [30] to their coefficients. It can exploit the sparsity structure of the Jacobian of the function for computing the right side when generating Jacobians of Taylor coefficients. The IADOL-C package applies well on large and complex problems. On the other hand, the FADBAD/TADIFF package applies well on small to medium size problems.

The interval arithmetic package in VNODE is PROFIL/BAIS which is among the fastest interval arithmetic packages. It uses directional rounding facilities of the processor on the machines on which it is installed. Isolating the machine dependent code in small assembler files allows portability of the code. The number of rounding mode switches and sign tests are minimized in vector and matrix operations. Moreover, it provides matrix and vector operations and essential routines including guaranteed linear equation solvers and optimization methods.

In summary, the VNODE package is developed to help users compute validated solutions of IVPs for ODEs in an effective and user-friendly manner. The design and implementation of the VNODE package follows well-defined patterns. Therefore, the methods which are implemented, modified or used are very systematic. The VNODE package also allows users to construct solvers by choosing an appropriate method from a set of methods. Moreover, it enables users to isolate and compare methods implementing the same part of a solver.

Chapter 3

Bezier Surface

In this chapter, the Bezier surface patch is introduced as the surfaces which are going to be used in this thesis are of this form. Such a patch is represented using Bernstein basis polynomials which have better numerical stability under perturbation of their coefficients than in the power basis.

3.1 Bernstein Polynomials

The Bernstein polynomials are defined as

$$B_{i,n}(t) = \frac{n!}{i!\,(n-i)!}(1-t)^{n-i}\cdot t^{i} \quad , i = 0,\ldots,n.$$

The properties of Bernstein polynomials are:

Non-negativity:

$B_{i,n}(t)$ is greater than or equal to zero. $0 \le t \le 1$, $i = 0,\ldots,n.$

Partition of unity:

$$\sum_{i=0}^{n} B_{i,n}(t) = (1-t+t)^{n} = 1$$

Symmetry:

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

Recursion:

$$B_{i,n}(t) = (1-t)B_{i,n-1}(t) + t\,B_{i-1,n-1}(t) \text{ with } B_{i,n}(t)=0 \text{ for } i<0,\ i>n \text{ and } B_{0,0}(t) = 1$$

Linear precision:

The monomial t can be expressed as the weighted sum of Bernstein polynomials of degree n with coefficients evenly space in the interval [0,1].

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

Degree elevation:

The basis functions of degree n can be expressed in terms of those of degree n+1 as

$$B_{i,n}(t) = \left(1 - \frac{i}{n+1}\right) B_{i,n+1}(t) + \left(\frac{i+1}{n+1}\right) B_{i+1,n+1}(t)$$

where i=0,1,…,n. Or more generally in terms of basis functions of degree n+r as

$$B_{i,n}(t) = \sum_{j=i}^{i+r} \frac{\binom{n}{j}\binom{r}{j-i}}{\binom{n+r}{j}} B_{j,n+r}(t) \quad , \ i = 0,\ldots,n.$$

3.2 Arithmetic Operations of Polynomials in Bernstein Form

Arithmetic operations between polynomials are often required for shape interrogation. The arithmetic operations include addition, subtraction and multiplication. Let the two polynomials f(t) and g(t) of degree m and n with Bernstein coefficients f(m,i) and g(n,i) be as follows:

$$f(t) = \sum_{i=0}^{m} f(m,i) B_{i,m}(t)$$

$$g(t) = \sum_{i=0}^{n} g(n,i)\, B_{i,n}(t) \qquad 0 \le t \le 1$$

Addition and subtraction:

If the degrees of the two polynomials are the same, i.e. m=n, we simply add or subtract the coefficients:

$$f(t)+g(t) = \sum_{i=0}^{m} (f(m,i) + g(m,i))\ B_{i,m}(t)$$

$$f(t)-g(t) = \sum_{i=0}^{m} (f(m,i) - g(m,i))\ B_{i,m}(t)$$

In terms of basis functions of degree n+r as:

$$B_{i,n}(t) = \sum_{j=i}^{i+r} \frac{\binom{n}{j}\cdot\binom{r}{j-i}}{\binom{n+r}{j}} B_{j,n+r}(t) \quad , \; i = 0,\ldots,n.$$

If m>n, degree elevation of g(t) m-n times is needed by using the above formula, and then add and subtract the coefficients:

$$f(t)+g(t) = \sum_{i=0}^{m} \left[f(m,i) + \sum_{j=\max(0,\,i-m+n)}^{\min(n,\,i)} \frac{\binom{n}{j}\cdot\binom{m-n}{i-j}}{\binom{m}{i}} \cdot g(n,j) \right] B_{i,m}(t)$$

Multiplication:

Multiplication of two polynomials of degree m and n yields a degree m+n polynomial.

$$f(t)g(t) = \sum_{i=0}^{m+n} \left[\sum_{j=\max(0,\,i-n)}^{\min(m,\,i)} \frac{\binom{m}{j}\cdot\binom{n}{i-j}}{\binom{m+n}{i}} \cdot f(m,j)\cdot g(n,i-j) \right] B_{i,m+n}(t)$$

3.3 Definition and Properties of a Bezier Curve

A Bezier curve is a parametric curve that uses the Bernstein polynomials as a basis. A Bezier curve of degree n (order n+1) is represented as:

$$\mathbf{r}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_{i,n(t)} \qquad 0 \le t \le 1$$

The shape of the curve is determined by the coefficients, $\mathbf{b}_i$, which are control points or Bezier points and the basic function $B_{i,n}(t)$. Lines drawn between consecutive control points of the curve form the control polygon. Figure 3-1 shows a cubic Bezier curve together with its control polygon.

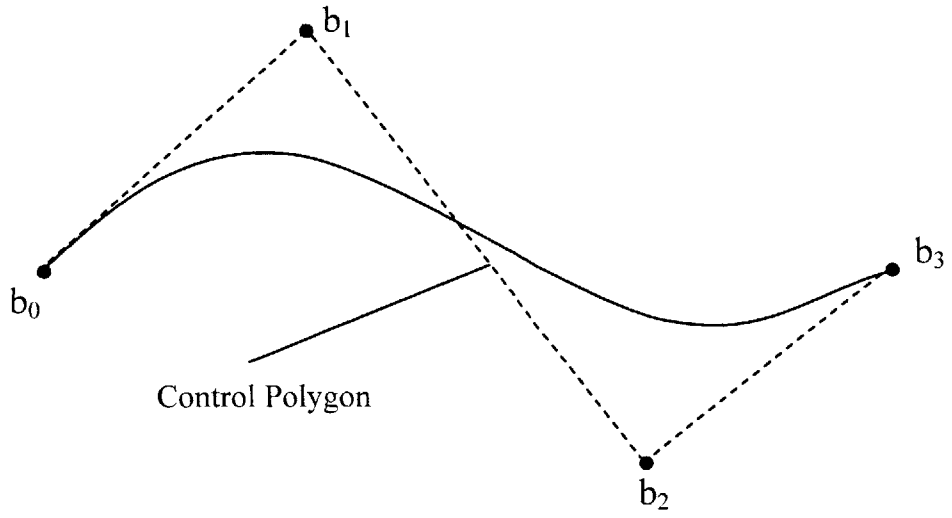


Figure 3-1: A cubic Bezier curve with control polygon (adapted from [28])

The properties of Bezier curves are:

Geometry invariance property:

Partition of unity property of the Bernstein polynomial assures the invariance of the shape

of the Bezier curve under translation and rotation of its control points.

End points geometric property:

The first and last control points are the endpoints of the curve, and the curve is tangent to the control polygon at the endpoints.

Convex hull property:

A domain D is convex if for any two points P_1 and P_2 in the same domain, the segment P_1P_2 is entirely contained in the domain D. The intersection of convex domains is also a convex domain. The convex hull of a set of points P is the boundary of the smallest convex domain containing P. Therefore, the convex hull of a Bezier curve is the boundary of the intersection of all the convex sets containing all vertices or the intersection of the half spaces generated by taking three vertices at a time to construct a plane and having all other vertices on one side. The convex hull can also be conceptualized at the shape of a rubber band in 2-D or a sheet in 3-D stretched taut over the polygon vertices [9]. Figure 3-2 shows a cubic Bezier curve with convex hull.

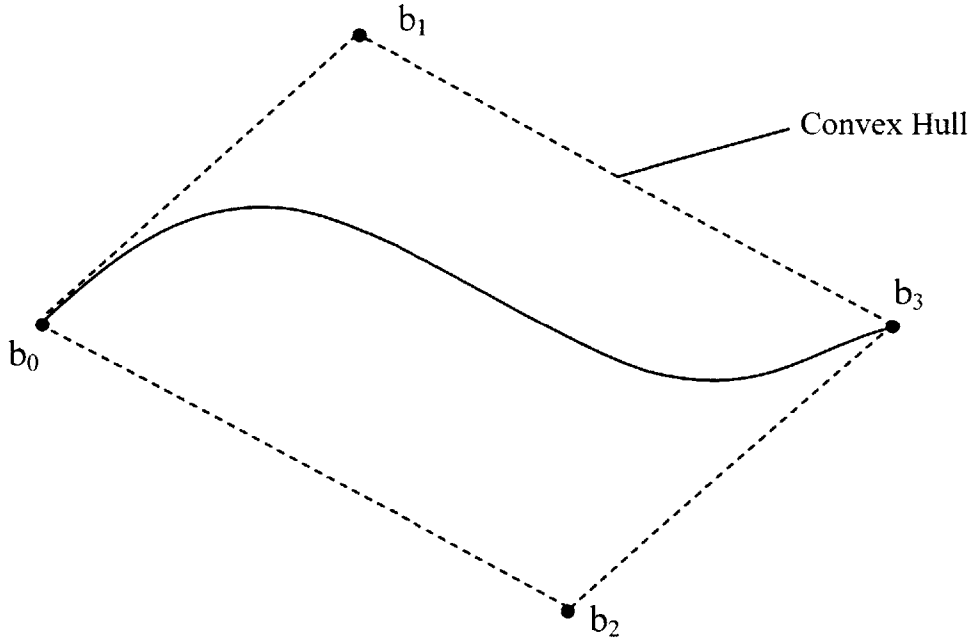
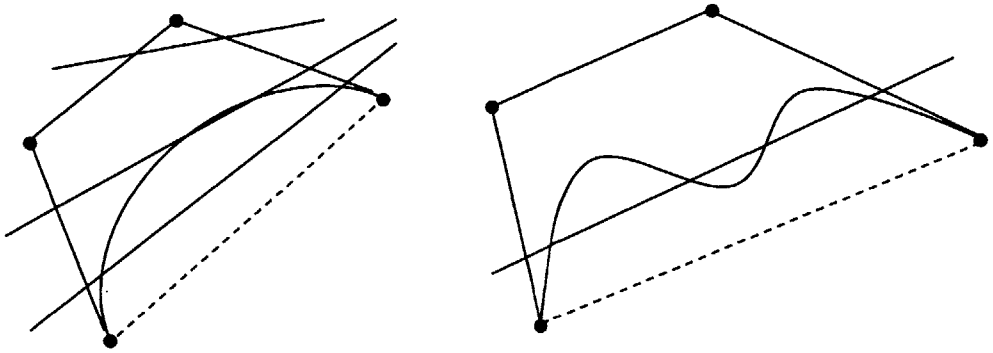


Figure 3-2: A cubic Bezier curve with convex hull (adapted from [28])

Variation diminishing property:

The number of intersections of a straight line with a planar Bezier curve is no greater than the number of intersections of the line with the control polygon. A line intersecting the

convex hull of a planar Bezier curve may intersect the curve transversally, be tangent to the curve, or not intersect the curve at all. However, it may not intersect the curve more times than it intersects the control polygon. Figure 3-3 shows a cubic Bezier curve with the variation diminishing property.



Possible Impossible

Figure 3-3: A cubic Bezier curve with the variation diminishing property (adapted from [28])

The same relation is also true for a plane with a space Bezier curve. Therefore, a Bezier curve oscillates less than its control polygon which also means the control polygon's segments exaggerate the oscillation of the curve. This property is important in intersection algorithms and in detecting the fairness of Bezier curves [28].

Symmetry property:

If we renumber the control points from $\mathbf{b}_0,\mathbf{b}_1,\ldots,\mathbf{b}_n$ to $\mathbf{b}_n,\mathbf{b}_{n-1},\ldots,\mathbf{b}_0$, by using the symmetry property of the Bernstein polynomial, the following equation is also true:

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

3.4 Definition and Properties of a Bezier Surface

A tensor product surface patch represented using Bernstein polynomials, known as a Bezier surface patch, is given by the following equation:

$$\mathbf{r}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{b}_{ij} B_{i,m}(u)\ B_{j,n}(v) \qquad 0 \le u, v \le 1$$

The set of straight lines drawn between consecutive control points $\mathbf{b}_{ij}$ is referred to as the control net. The Bezier surface is a direct extension of the univariate Bezier curve to its bivariate form, therefore, it also has the following properties of the Bezier curve:

Geometry invariance property

End points geometric property

Convex hull property

However, the Bezier surface patch does not have the variation diminishing property. Figure 3-4 shows a bi-quadratic Bezier surface with its control net.

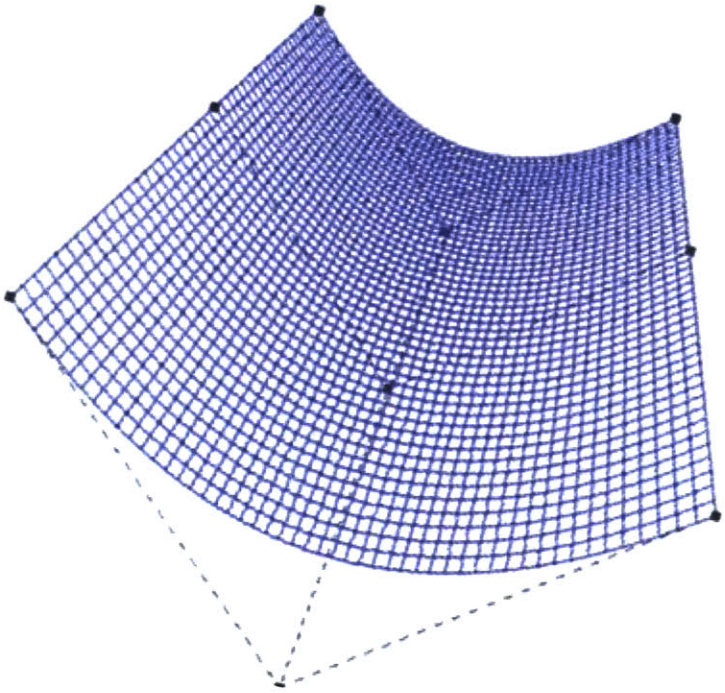


Figure 3-4: A bi-quadratic Bezier surface with its control net

Chapter 4

Iso-Contouring of Curvature of a Bezier Surface Patch

4.1 Stationary Points of Curvature of Free-Form Parametric Surfaces

Stationary points of surface curvature are important in methods for the correct topological decomposition of the surface on the basis of curvature [20]. A curvature C(u , v) of a parametric surface patch **r** = **r** (u , v), is a scalar function of u, v, where u and v are between 0 and 1. In order to locate all the stationary points of curvature and to find the global maximum and minimum values of C (u , v) to provide a correct topological decomposition of the surface, the following values have to be evaluated:

First, the four values of curvature at the parameter domain corners.

$$C(0,0), C(0,1), C(1,0), C(1,1).$$

Second, the roots of the following four equations (stationary points along parameter domain boundaries).

$$C_u(u,0) = 0, C_u(u,1) = 0, \qquad 0 \leq u \leq 1$$

$$C_v(0,v) = 0, C_v(1,v) = 0, \qquad 0 \leq v \leq 1$$

where C_u and C_v are the first partial derivatives of C with respect to u and v, respectively.

Third, the roots of the following two equations (stationary points within the parameter domain).

$$C_u(u,v) = 0,\ C_v(u,v) = 0, \qquad 0 \le u, v \le 1$$

Those equations are applied to the Gaussian curvature $K(u,v)$, the mean curvature $H(u,v)$ and the two principal curvatures $\kappa(u,v)$ in order to compute the stationary points for each case [28].

4.1.1 Gaussian Curvature: $K(u,v)$

In the case of finding the stationary points of Gaussian curvature within the domain, the following equations are true when $C(u,v)$ is replaced by $K(u,v)$ [28]:

$$K_u(u,v) = \frac{\Lambda(u,v)}{S^6(u,v)} = 0$$

$$K_v(u,v) = \frac{I(u,v)}{S^6(u,v)} = 0 \qquad 0 \le u, v \le 1$$

where

$$S(u,v) = |\mathbf{S}| = |\mathbf{r}_u \times \mathbf{r}_v|$$

$$L(u,v) = A_u S^2 - 4(\mathbf{S} \cdot \mathbf{S}_u) A$$

$$I(u,v) = A_v S^2 - 4(\mathbf{S} \cdot \mathbf{S}_v) A$$

$$A = (\mathbf{S} \cdot \mathbf{r}_{uu})(\mathbf{S} \cdot \mathbf{r}_{vv}) - (\mathbf{S} \cdot \mathbf{r}_{uv})(\mathbf{S} \cdot \mathbf{r}_{uv})$$

$\mathbf{r}_u$ and $\mathbf{r}_v$ are the first partial derivatives of $\mathbf{r}(u,v)$ with respect to u and v, respectively.

$\mathbf{S}_u$ and $\mathbf{S}_v$ are the first partial derivatives of $\mathbf{S}(u,v)$ with respect to u and v, respectively.

A_u and A_v are the first partial derivatives of A with respect to u and v, respectively.

$\mathbf{r}_{uu}$ and $\mathbf{r}_{uv}$ are the first partial derivatives of $\mathbf{r}_u$ with respect to u and v, respectively.

$\mathbf{r}_{vv}$ is the first partial derivative of $\mathbf{r}_v$ with respect to v.

When S is not zero,

$$\Lambda\ (u,v)=0,\quad I\ (u,v)=0,\qquad 0 \le u,v \le 1$$

The stationary points along the four boundary edges can be found by finding the solutions of the following equations:

$$\Lambda\ (u,0)=0,\quad \Lambda\ (u,1)=0,\qquad 0 \le u \le 1$$

$$I\ (0,v)=0,\quad I\ (1,v)=0,\qquad 0 \le v \le 1$$

4.1.2 Mean Curvature: H (u , v)

Similarly, in order to find the stationary points of mean curvature within the domain, the following equations are true when C (u , v) is replaced by H (u , v) [28]:

$$H_u(u,v) = \frac{\Phi(u,v)}{2S^5(u,v)} = 0$$

$$H_v(u,v) = \frac{\Psi(u,v)}{2S^5(u,v)} = 0 \qquad 0 \le u,v \le 1$$

where

$$S(u,v) = |\mathbf{S}| = |\mathbf{r}_u \times \mathbf{r}_v|$$

$$F(u,v) = B_u S^2 - 3(\mathbf{S}\cdot\mathbf{S}_u)B$$

$$Y(u,v) = B_v S^2 - 3(\mathbf{S}\cdot\mathbf{S}_v)B$$

$$B = 2(\mathbf{r}_u\cdot\mathbf{r}_v)(\mathbf{S}\cdot\mathbf{r}_{uv}) - (\mathbf{r}_u\cdot\mathbf{r}_u)(\mathbf{S}\cdot\mathbf{r}_{vv}) - (\mathbf{r}_v\cdot\mathbf{r}_v)(\mathbf{S}\cdot\mathbf{r}_{uu})$$

$\mathbf{S}_u$, $\mathbf{S}_v$,$\mathbf{r}_u$, $\mathbf{r}_v$, $\mathbf{r}_{uu}$,, $\mathbf{r}_{uv}$ and $\mathbf{r}_{vv}$ are the same as those described in section 4.1.1.

B_u and B_v are the first partial derivatives of B with respect to u and v, respectively.

When S is not zero,

$$\Phi\ (u, v) = 0, \psi(u, v) = 0, \qquad 0 \le u, v \le 1$$

The stationary points along the four boundary edges can be found by finding the solutions of the following equations:

$$\Phi(u, 0) = 0, \Phi(u, 1) = 0, \qquad 0 \le u \le 1$$

$$\psi(0, v) = 0, \psi(1, v) = 0, \qquad 0 \le v \le 1$$

4.1.3 Principal Curvature: $\kappa\ (u, v)$

For principal curvature, the stationary points within the domain can be found by solving the following equations when $C(u, v)$ is replaced by $\kappa\ (u, v)$ [28]:

Maximum principal curvature:

$$k_u(u, v) = \frac{p(u,v) + q(u,v)\cdot\sqrt{g(u,v)}}{2S^5(u,v)\cdot\sqrt{g(u,v)}} = 0$$

$$k_v(u, v) = \frac{m(u,v) + n(u,v)\cdot\sqrt{g(u,v)}}{2S^5(u,v)\cdot\sqrt{g(u,v)}} = 0 \qquad 0 \le u, v \le 1$$

Minimum principal curvature:

$$k_u(u, v) = \frac{p(u,v) - q(u,v)\cdot\sqrt{g(u,v)}}{2S^5(u,v)\cdot\sqrt{g(u,v)}} = 0$$

$$k_v(u, v) = \frac{m(u,v) - n(u,v)\cdot\sqrt{g(u,v)}}{2S^5(u,v)\cdot\sqrt{g(u,v)}} = 0 \qquad 0 \le u, v \le 1$$

where

$$S(u,v) = |\mathbf{S}| = |\mathbf{r}_u \times \mathbf{r}_v|$$

$$p(u,v) = (BB_u - 2 \cdot A_u \cdot S^2) \cdot S^2 + (8 \cdot A \cdot S^2 - 3 \cdot B^2) \cdot (\mathbf{S} \cdot \mathbf{S}_u)$$

$$q(u,v) = B_u S^2 - 3 (\mathbf{S} \cdot \mathbf{S}_u) B$$

$$m(u,v) = (BB_v - 2 \cdot A_v \cdot S^2) \cdot S^2 + (8 \cdot A \cdot S^2 - 3 \cdot B^2) \cdot (\mathbf{S} \cdot \mathbf{S}_v)$$

$$n(u,v) = B_v S^2 - 3 (\mathbf{S} \cdot \mathbf{S}_v) B$$

$$g(u,v) = B^2 - 4 \cdot A \cdot S^2$$

$\mathbf{S}_u$, $\mathbf{S}_v$,$\mathbf{r}_u$, $\mathbf{r}_v$, A_u, A_v, B_u, B_v , A and B are the same as those described in sections 4.1.1 and 4.1.2.

When S and g (u , v) are not zero, for maximum principal curvature,

$$p(u,v) + q(u,v) \cdot \sqrt{g(u,v)} = 0$$

$$m(u,v) + n(u,v) \cdot \sqrt{g(u,v)} = 0 \qquad 0 \le u,v \le 1$$

The stationary points along the four boundary edges can be found by finding the solutions of the following equations:

$$p(u,0) + q(u,0) \cdot \sqrt{g(u,0)} = 0$$

$$p(u,1) + q(u,1) \cdot \sqrt{g(u,1)} = 0$$

$$m(0,v) + n(0,v) \cdot \sqrt{g(0,v)} = 0$$

$$m(1,v) + n(1,v) \cdot \sqrt{g(1,v)} = 0 \qquad 0 \le u,v \le 1$$

The plus signs in the above six equations can be changed to minus signs in order to find the solutions for minimum principal curvature as well.

When g (u , v) is zero which means the two principal curvatures are identical for that point, i.e. an umbilical point, the following equations are true [21]:

$$(B \cdot B_u - 2 \cdot A_u \cdot S^2) - 4 \cdot A \cdot (\mathbf{S} \cdot \mathbf{S}_u) = 0$$

$$(B \cdot B_v - 2 \cdot A_v \cdot S^2) - 4 \cdot A \cdot (\mathbf{S} \cdot \mathbf{S}_v) = 0$$

$$B^2 - 4 \cdot A \cdot S^2 = 0 \qquad 0 \leq u, v \leq 1$$

Therefore, the umbilical point can be checked if it corresponds to a local maximum or minimum of principal curvature.

4.2 Contouring Constant Curvature Lines

Contour curves for constant curvature can be described as in the following equation:

$$C (u , v) = \text{constant},$$

Here, C (u , v) is a curvature at the given point (u , v). As the contour curve lies on the surface r (u , v), then the curve in parametric form can be written as r (t) = r [u (t), v (t)]. The differentiation of C (u , v) = constant with respect to t results in:

$$C_u \dot{u} + C_v \dot{v} = 0$$

$\dot{u}$ and $\dot{v}$ are the first derivatives with respect to t, and ($\dot{u}$, $\dot{v}$) shows the direction of the contour line in parameter space. Then, the solution can be found as:

$$\dot{u} = \varepsilon C_v, \quad \dot{v} = - \varepsilon C_u$$

When the curvature map is constructed in the u-v parameter space, ε can be chosen as follows in order to provide arc-length parametrization in the parameter domain [28].

$$\varepsilon = \frac{\pm 1}{\sqrt{C_u^{\,2} + C_v^{\,2}}}$$

The contour lines in the parameter space of a bivariate function can be separated into three parts: First, local maxima and minima of the function are encircled by closed contour curves [15]. Second, at the precise level of a saddle point, the contour curves cross or exhibit more complex behavior. Third, contour curves start from a domain boundary point and end at a domain boundary point. In the following section 4.3, C_u and C_v are replaced by derivatives of the Gaussian, mean and principal curvatures separately to allow comparison with the results of the Praxiteles program [28].

4.3 Performance of Algorithm

The example here is a wave-like bicubic integral Bezier surface patch $[\mathbf{P}_1](u,v)$ (Figure 4-1). The control points are:

$$[\mathbf{P}_1](u,v) = \begin{bmatrix} (0,0,0) & \left(0,\frac{1}{3},0\right) & \left(0,\frac{2}{3},0\right) & (0,1,0) \\ \left(\frac{1}{3},0,0\right) & \left(\frac{1}{3},\frac{1}{3},1\right) & \left(\frac{1}{3},\frac{2}{3},\frac{3}{5}\right) & \left(\frac{1}{3},1,0\right) \\ \left(\frac{2}{3},0,0\right) & \left(\frac{2}{3},\frac{1}{3},-1\right) & \left(\frac{2}{3},\frac{2}{3},\frac{-3}{5}\right) & \left(\frac{2}{3},1,0\right) \\ (1,0,0) & \left(1,\frac{1}{3},0\right) & \left(1,\frac{2}{3},0\right) & (1,1,0) \end{bmatrix}$$

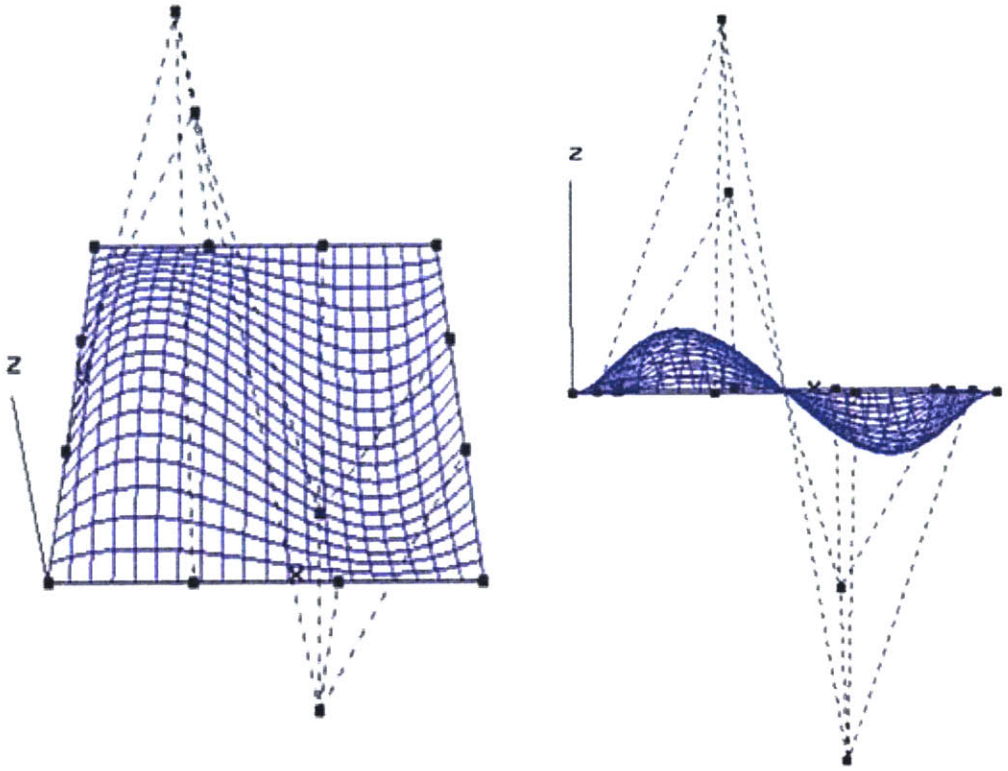


Figure 4-1: Wave-like bicubic integral Bezier surface $[\mathbf{P}_1](u,v)$ with control points

Gaussian Curvature:

Figure 4-2 shows the color map of the contour lines of Gaussian curvature for surface $[\mathbf{P}_1](u,v)$ in parametric space from the Praxiteles program.

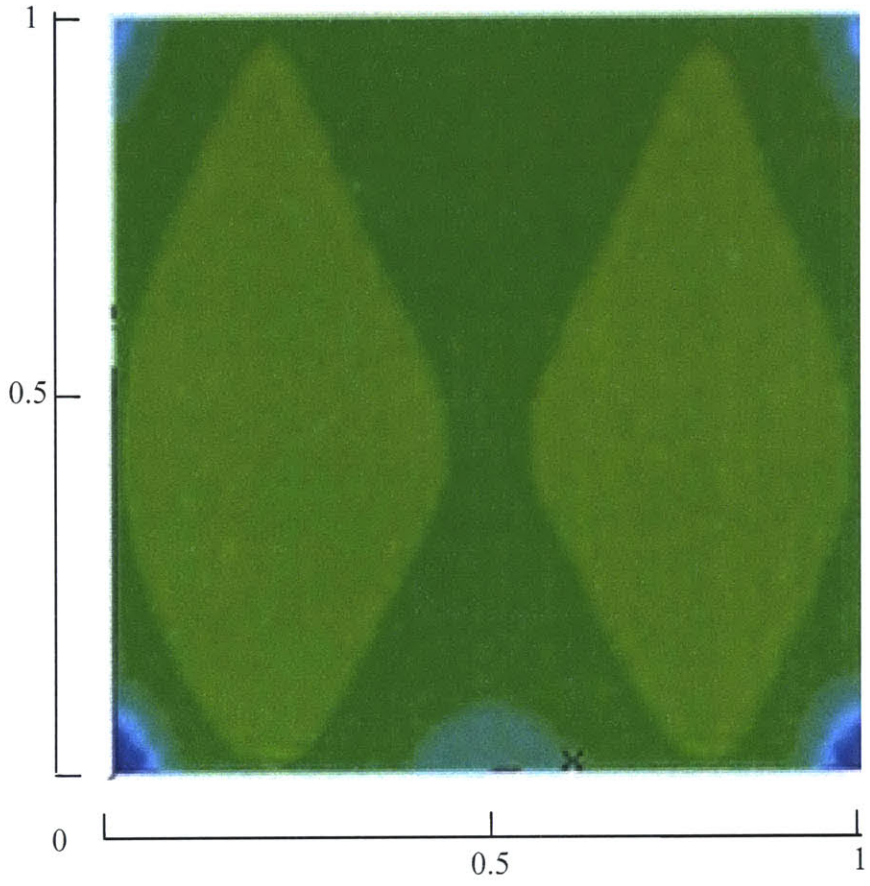


Figure 4-2: Color map of contour lines of Gaussian curvature from the Praxiteles program for surface $[\mathbf{P}_1](u,v)$

Since surface $[\mathbf{P}_1](u,v)$ is anti-symmetric with respect u = 0.5, the contour lines of Gaussian curvature are also symmetric with respect to u = 0.5 due to the fact that the Gaussian curvature is the product of maximum and minimum principal curvatures. The global maximum is located at (0.2 , 0.37) and (0.8 , 0.37) and the global minimum is

located at (0 , 0) and (1 , 0). By inputting the equations from section 4.1.1 into the VNODE program, the contour lines of Gaussian curvature lines can be found. Figure 4-3 shows the contour lines of Gaussian curvature lines generated from the VNODE program.

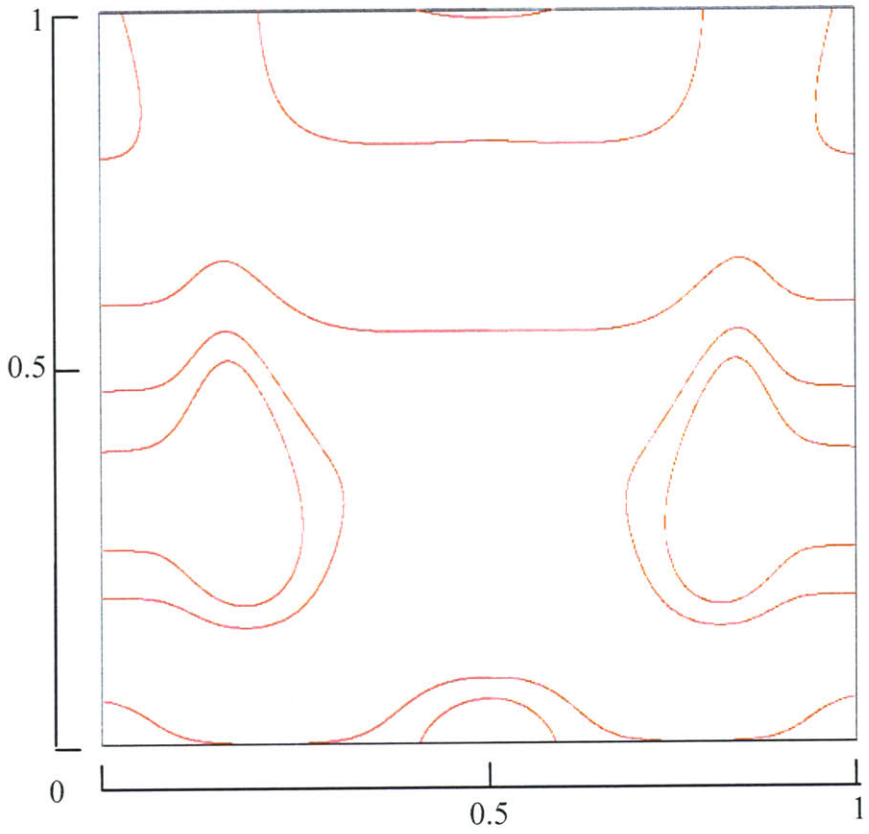


Figure 4-3: Contour lines of Gaussian curvature from the VNODE program for surface $[\mathbf{P}_1](u,v)$

Similarly, the contour lines of Gaussian curvature generated from the VNODE program are also symmetric with respect to u = 0.5. The starting points are chosen on each border from 0.2 to 0.8 by the increasing step size by 0.2. The length of integration is determined by not exceeding the u-v parametric space. The average time for integrating the length of

1 is 1.82 minutes. Table 4-1 shows the input in the VNODE program for contour lines of Gaussian curvature for surface $[\mathbf{P}_1](u,v)$.

Number	**Starting Point: (u , v)**	**Length of Integration**	**Tolerance**	**Direction along X-axis**
1	(0 , 0.2)	0.98	10^{-15}	+
2	(0 , 0.4)	0.91	10^{-15}	+
3	(0 , 0.6)	1.33	10^{-15}	+
4	(0 , 0.8)	0.24	10^{-15}	+
5	(1 , 0.2)	0.98	10^{-15}	-
6	(1 , 0.4)	0.91	10^{-15}	-
7	(1 , 0.8)	0.24	10^{-15}	-
8	(0.2 , 0)	0.66	10^{-15}	+
9	(0.2 , 0)	0.22	10^{-15}	-
10	(0.4 , 0)	0.26	10^{-15}	+
11	(0.8 , 0)	0.22	10^{-15}	+
12	(0.2 , 1)	0.92	10^{-15}	+
13	(0.4 , 1)	0.20	10^{-15}	+

Table 4-1: Input in the VNODE program for contour lines of Gaussian curvature for surface $[\mathbf{P}_1](u,v)$

Mean Curvature:

Figure 4-4 shows the color map of the contour lines of mean curvature for surface $[\mathbf{P}_1](u,v)$ in parametric space from the Praxiteles program.

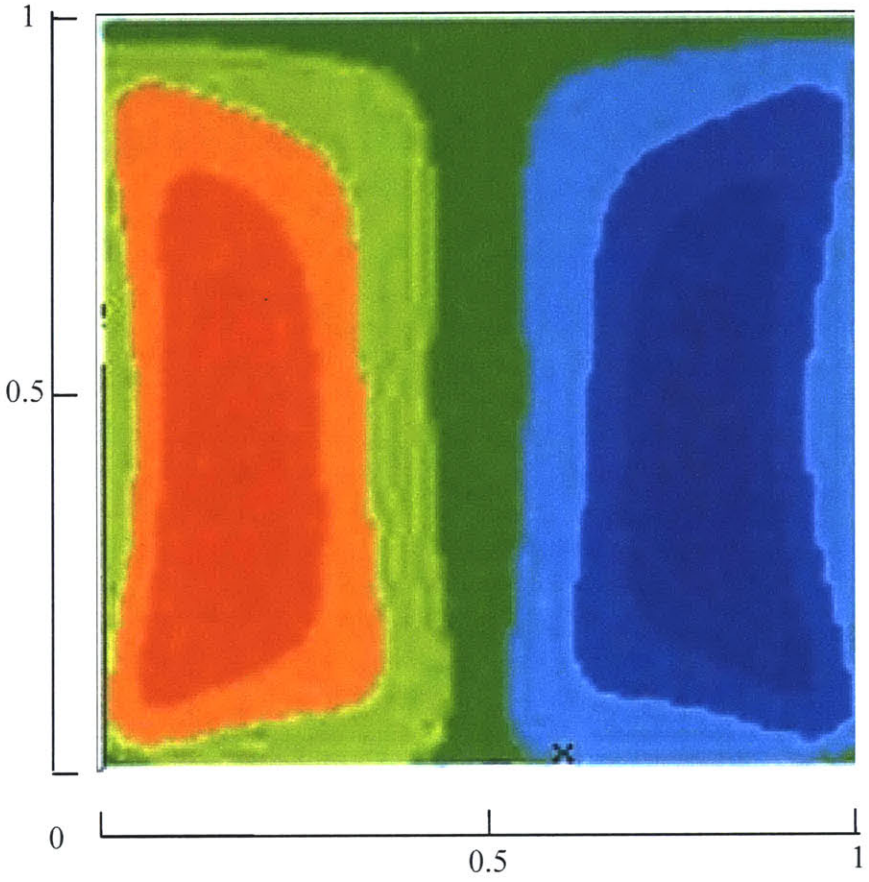


Figure 4-4: Color map of contour lines of mean curvature from the Praxiteles program for surface $[\mathbf{P}_1](u,v)$

Due to the fact that surface $[\mathbf{P}_1](u,v)$ is anti-symmetric with respect u = 0.5, the contour lines of mean curvature have the contour line H = 0 when u = 0.5. The global maximum is located at (0.19 , 0.41) and the global minimum is located at (0.81 , 0.41). By inputting the equations from section 4.1.2 into the VNODE program, the contour lines of

mean curvature lines can be found. Figure 4-5 shows the contour lines of mean curvature lines generated from the VNODE program.

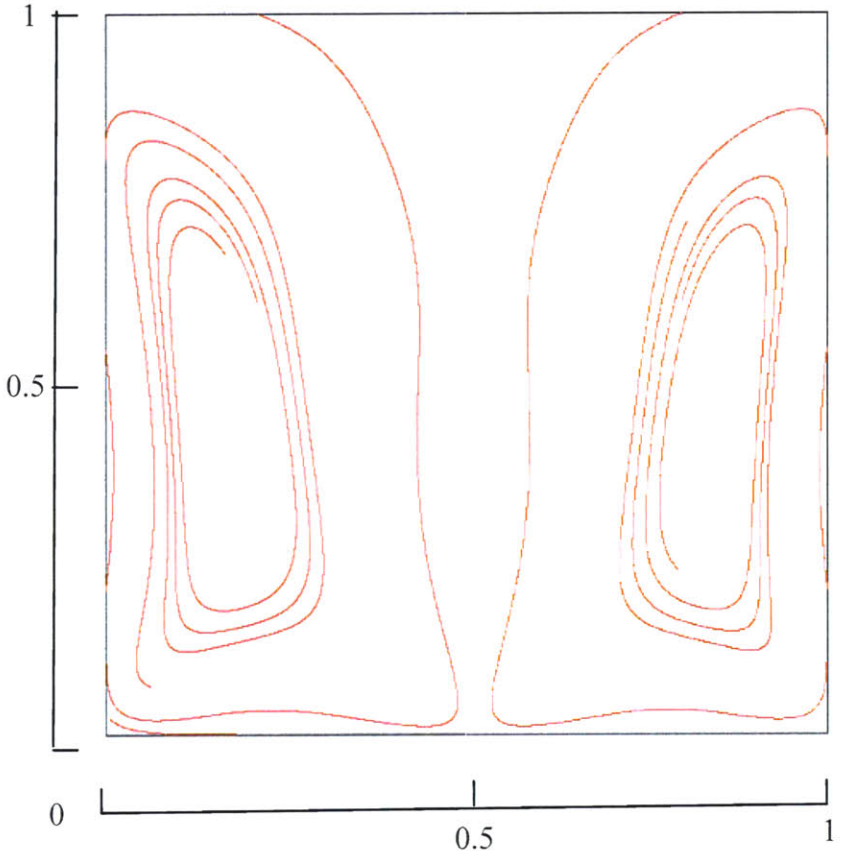


Figure 4-5: Contour lines of mean curvature from the VNODE program for surface $[\mathbf{P}_1](u,v)$

The contour lines of mean curvature generated from the VNODE program are similar to those in Figure 4-4. The starting points are chosen on each border from 0.2 to 0.8 by the increasing step size by 0.2. However, the curvature lines from some starting points do not show well in this example because the denominator from the formula becomes very small and this leads to error. Therefore, those points are omitted. The length of integration is

determined by not exceeding the u-v parametric space as well. The average time for integrating the length of 1 is 7.72 minutes. Moreover, some contour lines of mean curvature do not continue to extend after some point. The reason for this is the integration becomes more and more complicated and the intervals from the ODE integration become very small such as 10^{-15}, and therefore, it takes very long time to show the full trace of the curvature lines. Table 4-2 shows the input in the VNODE program for contour lines of mean curvature for surface $[\mathbf{P}_1](u,v)$.

Number	**Starting Point: (u , v)**	**Length of Integration**	**Tolerance**	**Direction along X-axis**
1	(0 , 0.2)	0.34	10^{-25}	+
2	(0 , 0.8)	3.61	10^{-25}	+
3	(0.2 , 0)	0.20	10^{-25}	-
4	(0.2 , 0.6)	2.50	10^{-25}	-
5	(0.2 , 1)	1.60	10^{-25}	+
6	(0.8 , 0.6)	1.45	10^{-25}	+
7	(0.8 , 1)	1.60	10^{-25}	-
8	(1 , 0.2)	0.34	10^{-25}	-
9	(1 , 0.8)	2.79	10^{-25}	-

Table 4-2: Input in the VNODE program for contour lines of mean curvature for surface $[\mathbf{P}_1](u,v)$

Maximum Principal Curvature:

Figure 4-6 shows the color map of the contour lines of maximum principal curvature for surface $[\mathbf{P}_1](u,v)$ in parametric space from the Praxiteles program.

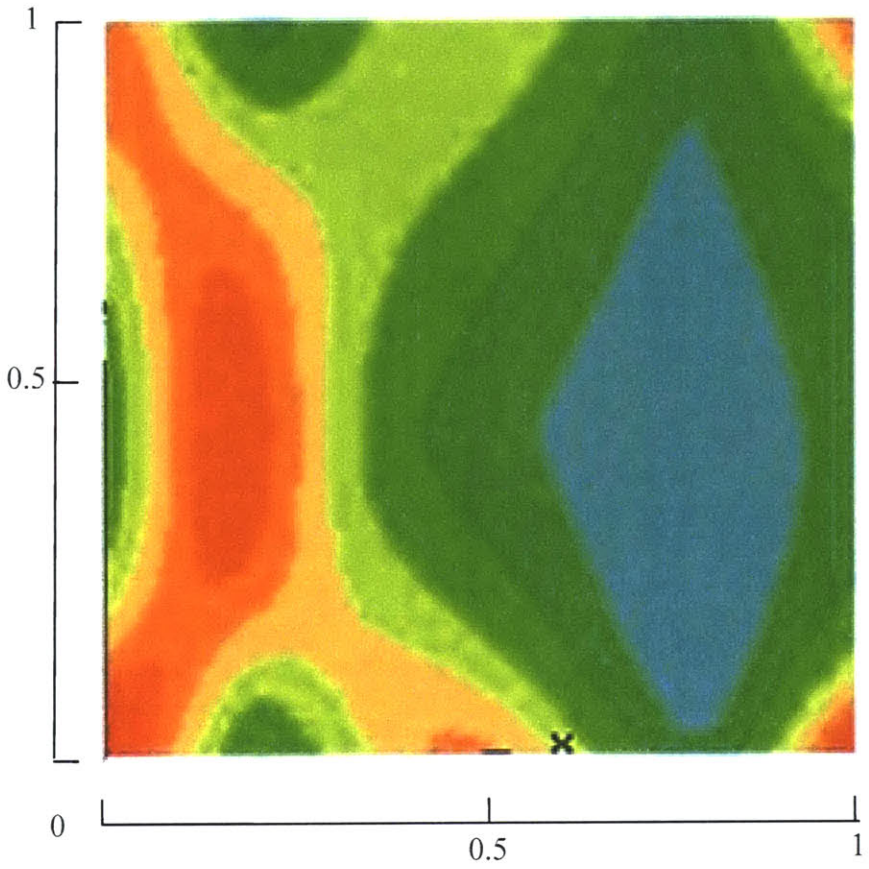


Figure 4-6: Color map of contour lines of maximum principal curvature from the Praxiteles program for surface $[\mathbf{P}_1](u,v)$

As for contour lines of maximum principal curvature for surface $[\mathbf{P}_1](u,v)$, the global maximum is located at (0 , 0) and (1 , 0) and the global minimum is located at (0.79 , 0.3). By inputting the equations from section 4.1.3 into the VNODE program, the contour lines of maximum principal curvature can be found. Figure 4-7 shows the contour lines of maximum principal curvature generated from the VNODE program.

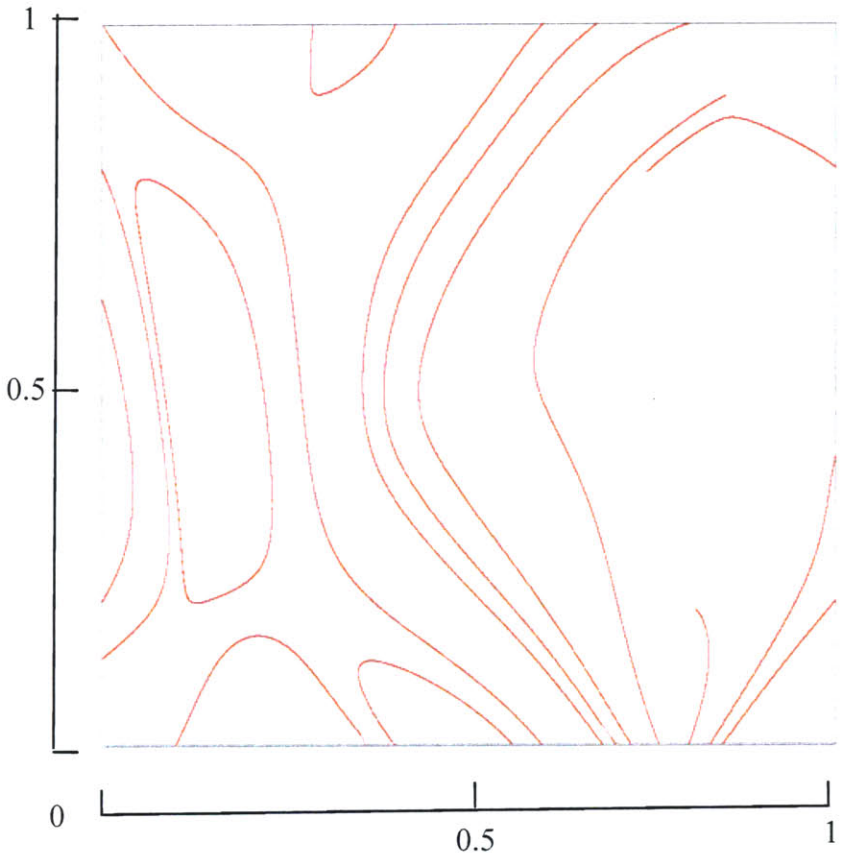


Figure 4-7: Contour lines of maximum principal curvature from the VNODE program for surface $[\mathbf{P}_1](u,v)$

It can be seen easily that the contour lines of maximum principal curvature generated from the VNODE program are similar to Figure 4-6. The starting points are chosen randomly on each border and two from the inside which are (0.2 , 0.6) and (0.6 , 0.6) in order to compare the curvature lines generated from the VNODE program to those from the Praxiteles program. The length of integration is also determined by not exceeding the u-v parametric space. The average time for integrating the length of 1 is 87.42 minutes. Besides the long time for integrating, some contour lines of maximum principal curvature do not continue to extend after some point like the discontinued lines shown in Figure 4-7.

The reason for this is the same with the situation happening in mean curvature. Table 4-3 shows the input in the VNODE program for contour lines of maximum principal curvature for surface $[\mathbf{P}_1](u,v)$.

Number	Starting Point: (u , v)	Length of Integration	Tolerance	Direction along X-axis
1	(0 , 0.2)	0.45	10^{-30}	+
2	(0 , 0.8)	0.76	10^{-30}	+
3	(0.1 , 0)	0.40	10^{-30}	+
4	(0.4 , 0)	0.39	10^{-30}	-
5	(0.6 , 0)	1.26	10^{-30}	-
6	(0.7 , 0)	1.29	10^{-30}	-
7	(0.8 , 0)	0.26	10^{-30}	+
8	(0.4 , 1)	0.26	10^{-30}	-
9	(0.6 , 1)	1.27	10^{-30}	-
10	(0.8 , 1)	1.37	10^{-30}	+
11	(1 , 0.2)	0.27	10^{-30}	-
12	(1 , 0.4)	0.52	10^{-30}	-
13	(1 , 0.8)	0.61	10^{-30}	-
14	(0.2 , 0.6)	0.86	10^{-30}	+
15	(0.2 , 0.6)	1.08	10^{-30}	-
16	(0.6 , 0.6)	0.44	10^{-30}	+
17	(0.6 , 0.6)	0.72	10^{-30}	-

Table 4-3: Input in the VNODE program for contour lines of maximum principal curvature for surface $[\mathbf{P}_1](u,v)$

Minimum Principal Curvature:

Figure 4-7 shows the color map of the contour lines of minimum principal curvature for surface $[\mathbf{P}_1](u,v)$ in parametric space from the Praxiteles program.

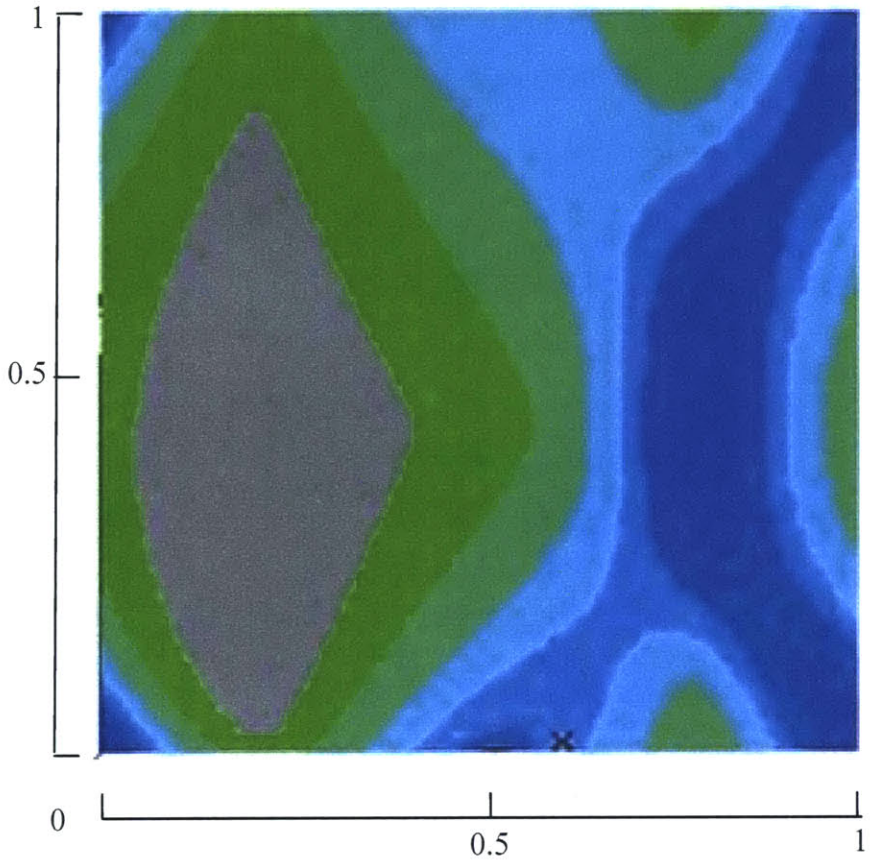


Figure 4-8: Color map of the contour lines of minimum principal curvature from the Praxiteles program for surface $[\mathbf{P}_1](u,v)$

As for contour lines of minimum principal curvature for surface $[\mathbf{P}_1](u,v)$, the global maximum is located at (0.21 , 0.3) and the global minimum is located at (0 , 0) and (1 , 0). By inputting the equations from section 4.1.3 into the VNODE program, the contour lines of minimum principal curvature can be found. Figure 4-9 shows the contour lines of minimum principal curvature generated from the VNODE program.

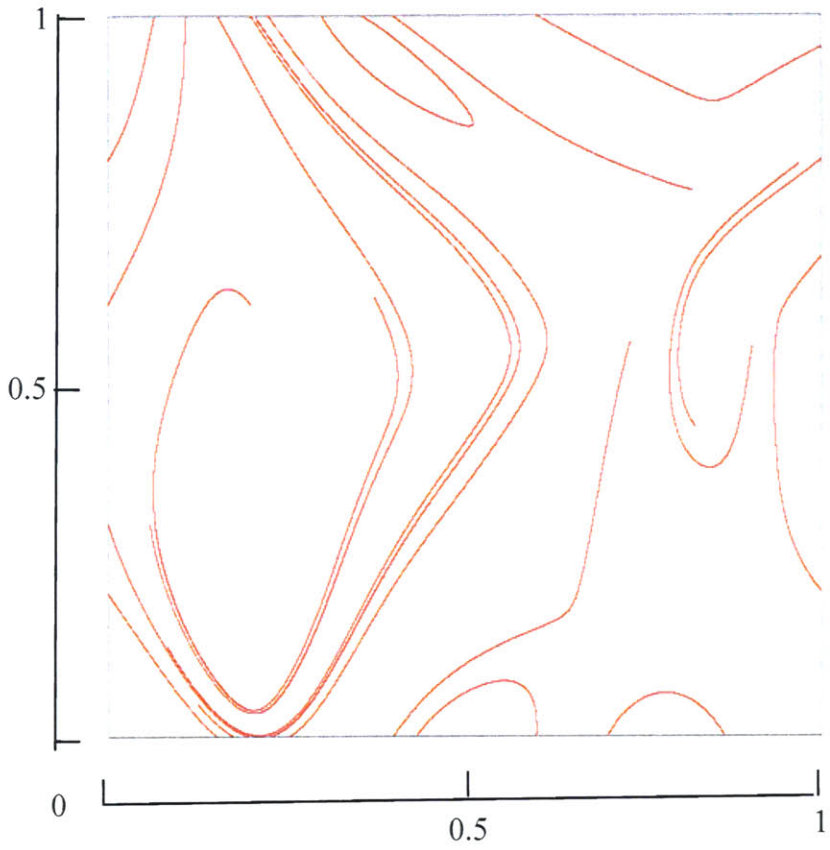


Figure 4-9: Contour lines of minimum principal curvature from the VNODE program for surface $[\mathbf{P}_1](u,v)$

It can also be seen easily that the contour lines of minimum principal curvature generated from the VNODE program are similar to Figure 4-8 although some curvature lines do not completely extend to the border. The starting points are also chosen randomly on each border and few from the inside in order to compare the curvature lines generated from the VNODE program to those from the Praxiteles program. The length of integration is determined by not exceeding the u-v parametric space as well. The average time for integrating the length of 1 is 88.25 minutes. Table 4-4 shows the input in the VNODE program for contour lines of minimum principal curvature for surface $[\mathbf{P}_1](u,v)$.

Number	Starting Point: (u , v)	Length of Integration	Tolerance	Direction along X-axis
1	(0 , 0.2)	0.27	10^{-30}	+
2	(0 , 0.6)	0.42	10^{-30}	+
3	(0 , 0.8)	0.21	10^{-30}	+
4	(0.2 , 0)	0.38	10^{-30}	-
5	(0.4 , 0)	0.82	10^{-30}	+
6	(0.6 , 0)	0.26	10^{-30}	-
7	(0.7 , 0)	0.21	10^{-30}	+
8	(0.2 , 1)	1.60	10^{-30}	+
9	(0.3 , 1)	0.49	10^{-30}	+
10	(0.4 , 1)	0.61	10^{-30}	+
11	(0.6 , 1)	0.61	10^{-30}	+
12	(1 , 0.2)	0.67	10^{-30}	-
13	(1 , 0.8)	0.52	10^{-30}	-
14	(0.2 , 0.6)	1.44	10^{-30}	+
15	(0.2 , 0.6)	1.38	10^{-30}	-
16	(0.4 , 0.6)	1.00	10^{-30}	+
17	(0.4 , 0.6)	0.48	10^{-30}	-
18	(0.6 , 0.6)	1.00	10^{-30}	+
19	(0.6 , 0.6)	0.96	10^{-30}	-
20	(0.8 , 0.6)	0.30	10^{-30}	+
21	(0.8 , 0.6)	0.49	10^{-30}	-

Table 4-4: Input in the VNODE program for contour lines of minimum principal curvature for surface $[\mathbf{P}_1](u,v)$

Chapter 5

Lines of Curvature of a Bezier Surface Patch

5.1 Lines of Curvature

The description of a line of curvature is a curve on a surface that has tangents which are principal directions at all of its points. There are two principal directions at a given non-umbilical point and they are orthogonal. The umbilic is a point on a surface where all of the normal curvatures are equal in all directions. Therefore, the principal directions are undefined at umbilics. More discussion on umbilics can be found in [28]. A line of curvature indicates a directional flow for the maximum or the minimum curvature across the surface [28]. Lines of curvatures can be found by integrating an initial value problem for a system of few nonlinear ordinary differential equations. The number of integrated points used to represent the contour line by straight line segments also affects the accuracy of the lines of curvature. In the parametric surface $\mathbf{r} = \mathbf{r}(u, v)$ where u and v are between 0 and 1, the equations for lines of curvature can be found to be as follows:

$$u' = \frac{du}{ds} = \eta (M + \kappa F)$$

$$v' = \frac{dv}{ds} = - \eta (L + \kappa E)$$

where

$$M = \mathbf{r}_{uv} \cdot \mathbf{N}, \; L = \mathbf{r}_{uu} \cdot \mathbf{N}, \; E = \mathbf{r}_{u} \cdot \mathbf{r}_{u}, \; F = \mathbf{r}_{u} \cdot \mathbf{r}_{v}$$

$$\eta = \frac{1 \text{ or } (-1)}{\sqrt{E\cdot(M+\kappa F)^2 - 2F(M+\kappa F)(L+\kappa E) + G\cdot(L+\kappa E)^2}}$$

$\mathbf{r}_u$ and $\mathbf{r}_v$ are the first partial derivatives of $\mathbf{r}$ (u , v) with respect to u and v, respectively.

$\mathbf{r}_{uu}$ and $\mathbf{r}_{uv}$ are the first partial derivatives of $\mathbf{r}_u$ with respect to u and v, respectively.

$\mathbf{N}$ is the unit normal vector which also equals ($\mathbf{r}_u \times \mathbf{r}_v$) / | $\mathbf{r}_u \times \mathbf{r}_v$ |.

κ is either κ_{max} or κ_{min} which represent maximum principal curvature and minimum principal curvature respectively.

$$\kappa_{max} = H + \sqrt{H^2 - K}$$

$$\kappa_{min} = H - \sqrt{H^2 - K}$$

$$K = \frac{L\cdot N - M^2}{E\cdot G - F^2}$$

$$H = \frac{2\cdot F\cdot M - E\cdot N - G\cdot L}{2\cdot\left(E\cdot G - F^2\right)}$$

$$N = \mathbf{r}_{vv} \cdot \mathbf{N},\ G = \mathbf{r}_v \cdot \mathbf{r}_v$$

$\mathbf{r}_{vv}$ is the first partial derivative of $\mathbf{r}_v$ with respect to v.

The sign of η determines which direction the solution should proceed.

However, when computing the lines of curvature by the above equations, two cases may happen which affect the results: First, that the coefficients in one of the equations can both be zero while they are not both zero in the other equation. Second, that both coefficients in one equation are small in absolute value while the other equation contains one coefficient which is large in absolute value [28]. In the first case, the results will not

be right because the near zero coefficients do not let the equations have enough information to find out the principal curvature direction. As for the second case, the small coefficients may lead to the numerical inaccuracies. Alourdas [2] found that the size of the coefficients can help to decide to use the equations above or not. In the situation when $|L + \kappa E| \geq |N + \kappa G|$, the equations above can be used. Otherwise, the following equations can be used for computing lines of curvature:

$$u' = \frac{du}{ds} = \mu (N + \kappa G)$$

$$v' = \frac{dv}{ds} = -\mu (M + \kappa F)$$

where

$$\mu = \frac{1 \text{ or } (-1)}{\sqrt{E \cdot (N + \kappa G)^2 - 2F(K + \kappa G)(M + \kappa F) + G \cdot (M + \kappa F)^2}}$$

M, N ,G, F and κ are the same as those described above.

The sign of μ also determines which direction the solution should proceed.

Since the unit normal vector **N** is used here for principal curvature lines, one important factor has to be taken into consideration which is the definition of the positive normal curvature. There are two different situations. The first one is when the center of curvature is on the same side of the normal vector and the second one, when the center of curvature is on the opposite side of the normal vector. Under these two different situations, a sign convention needs to be used for the positive normal curvature. In this thesis, the positive normal curvature is defined as the case in which the center of curvature is on the opposite

side of the normal vector and this is the same convention that the Praxiteles program uses [28].

5.2 Performance of Algorithm (Example: 1)

The first example is a symmetric bi-quadratic Bezier surface $[\mathbf{P}_2](u,v)$ (Figure 5-1). The control points are given by:

$$[\mathbf{P}_2](u,v) = \begin{bmatrix} (0,0,0) & \left(\frac{1}{2},0,2\right) & (1,0,0) \\ (0,1,0) & \left(\frac{1}{2},1,1\right) & (1,1,0) \\ (0,2,0) & \left(\frac{1}{2},2,2\right) & (1,2,0) \end{bmatrix}$$

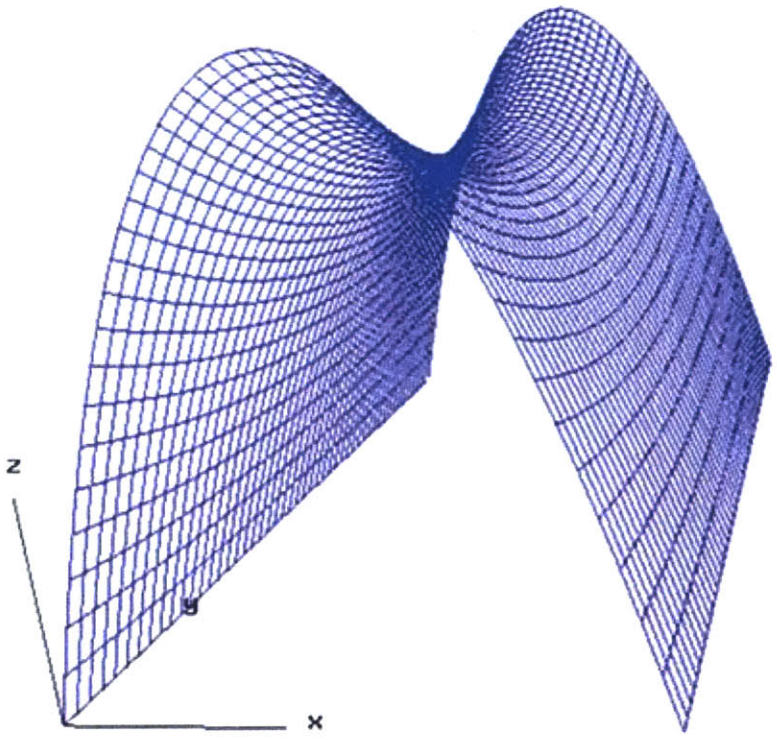


Figure 5-1: A symmetric bi-quadratic Bezier surface $[\mathbf{P}_2](u,v)$

Figure 5-2 shows the lines of curvature for surface $[\mathbf{P}_2](u,v)$ in 3-D space from the Praxiteles program.

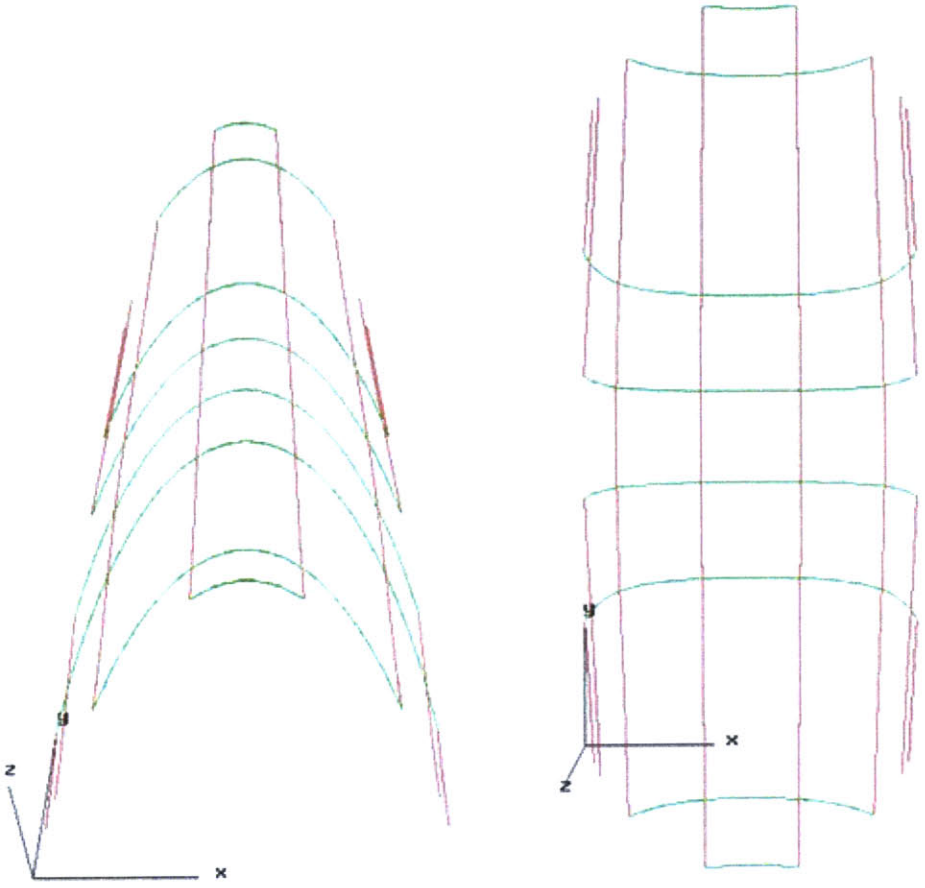


Figure 5-2: Lines of curvature from the Praxiteles program for surface $[\mathbf{P}_2](u,v)$

The lines of curvature for surface $[\mathbf{P}_2](u,v)$ are plotted by two files generated from the Praxiteles program. The vertical lines represent the lines of maximum principal curvature and the horizontal lines represent the lines of minimum principal curvature. In the Praxiteles program, a few options for lines of curvature can be specified including the

step size, how many points in the u-v parametric space of the surface can be calculated in u direction and v direction respectively. By inputing the equations from section 5.1 into the VNODE program, the lines of curvature can be found. Figure 5-3 shows the lines of curvature for surface $[\mathbf{P}_2](u,v)$ in 3-D space from the VNODE program.

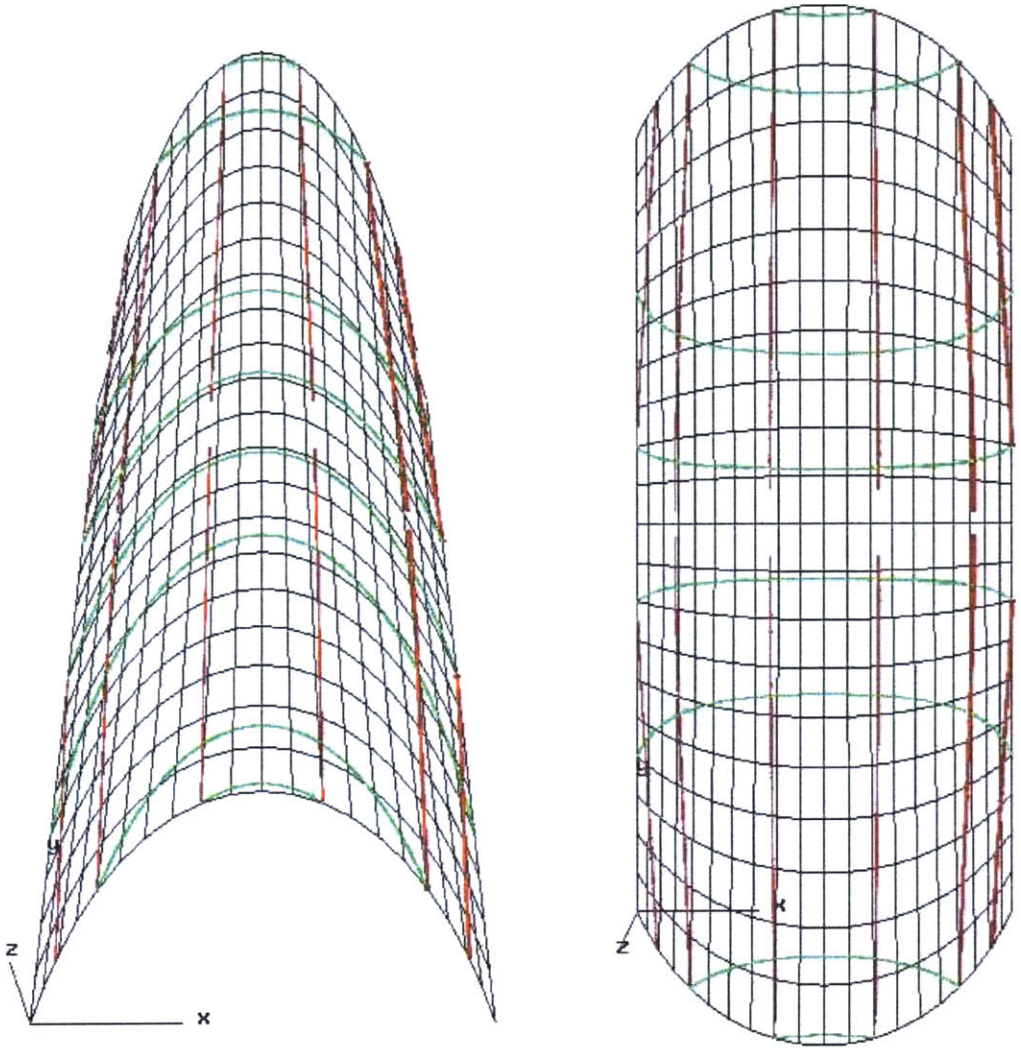


Figure 5-3: Lines of curvature from the VNODE program for surface $[\mathbf{P}_2](u,v)$

It can be seen easily that the lines of curvature generated from the VNODE program are the same as those generated from the Praxiteles program. However, the integration via VNODE becomes very difficult or even cannot continue when it proceeds to the middle of this surface. The starting points are chosen on each border from 0.2 to 0.8 by increasing the step size by 0.2. The length of integration is determined by not exceeding the u-v parametric space. The average time for integrating the length of 1 is 2.15 minutes. Tables 5-1 and 5-2 show the input in the VNODE program for lines of maximum principal curvature and lines of minimum principal curvature for surface $[\mathbf{P}_2](u,v)$ respectively.

Number	**Starting Point: (u , v)**	**Length of Integration**	**Tolerance**	**Direction along X-axis**
1	(0.2 , 0)	1.02	10^{-30}	+
2	(0.2 , 1)	1.02	10^{-30}	-
3	(0.4 , 0)	0.97	10^{-40}	+
4	(0.4 , 1)	0.97	10^{-40}	-
5	(0.6 , 0)	0.97	10^{-40}	+
6	(0.6 , 1)	0.97	10^{-40}	-
7	(0.8 , 0)	1.02	10^{-30}	+
8	(0.8 , 1)	1.02	10^{-30}	-
9	(0 , 0.4)	0.87	10^{-30}	-
10	(1 , 0.4)	0.87	10^{-30}	+
11	(0 , 0.6)	0.87	10^{-30}	-
12	(1 , 0.6)	0.87	10^{-30}	+

Table 5-1: Input in the VNODE program for lines of maximum principal curvature for surface $[\mathbf{P}_2](u,v)$

Number	Starting Point: (u , v)	Length of Integration	Tolerance	Direction along X-axis
1	(0 , 0.2)	1.01	10^{-30}	+
2	(1 , 0.2)	1.01	10^{-30}	-
3	(0 , 0.4)	0.95	10^{-30}	+
4	(1 , 0.4)	0.95	10^{-30}	-
5	(0 , 0.6)	0.95	10^{-30}	+
6	(1 , 0.6)	0.95	10^{-30}	-
7	(0 , 0.8)	1.01	10^{-30}	+
8	(1 , 0.8)	1.01	10^{-30}	-
9	(0.2 , 0)	0.47	10^{-30}	+
10	(0.8 , 0)	0.47	10^{-30}	-
11	(0.4 , 0)	0.11	10^{-30}	+
12	(0.6 , 0)	0.11	10^{-30}	-
13	(0.2 , 1)	0.47	10^{-30}	+
14	(0.8 , 1)	0.47	10^{-30}	-
15	(0.4 , 1)	0.11	10^{-30}	+
16	(0.6 , 1)	0.11	10^{-30}	-

Table 5-2: Input in the VNODE program for lines of minimum principal curvature for surface $[\mathbf{P}_2](u,v)$

5.3 Performance of Algorithm (Example: 2)

The second example is a bi-quadratic Bezier surface $[\mathbf{P}_3](u,v)$ (Figure 5-4). The control points are given by:

$$[\mathbf{P}_3](u,v) = \begin{bmatrix} (0,0,0) & \left(\frac{1}{2},0,0\right) & (1,0,0) \\ (0,1,0) & \left(\frac{1}{2},1,\frac{1}{2}\right) & (1,1,0) \\ (0,2,2) & \left(\frac{1}{2},2,\frac{3}{2}\right) & (1,2,3) \end{bmatrix}$$

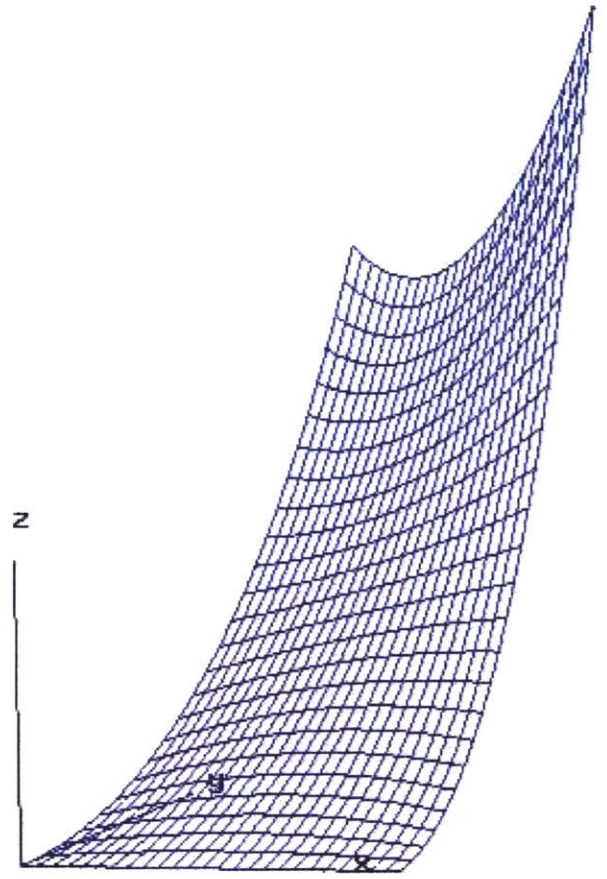


Figure 5-4: A bi-quadratic Bezier surface $[\mathbf{P}_3](u,v)$

Figure 5-5 shows the lines of curvature for surface $[\mathbf{P}_3](u,v)$ in 3-D space from the Praxiteles program.

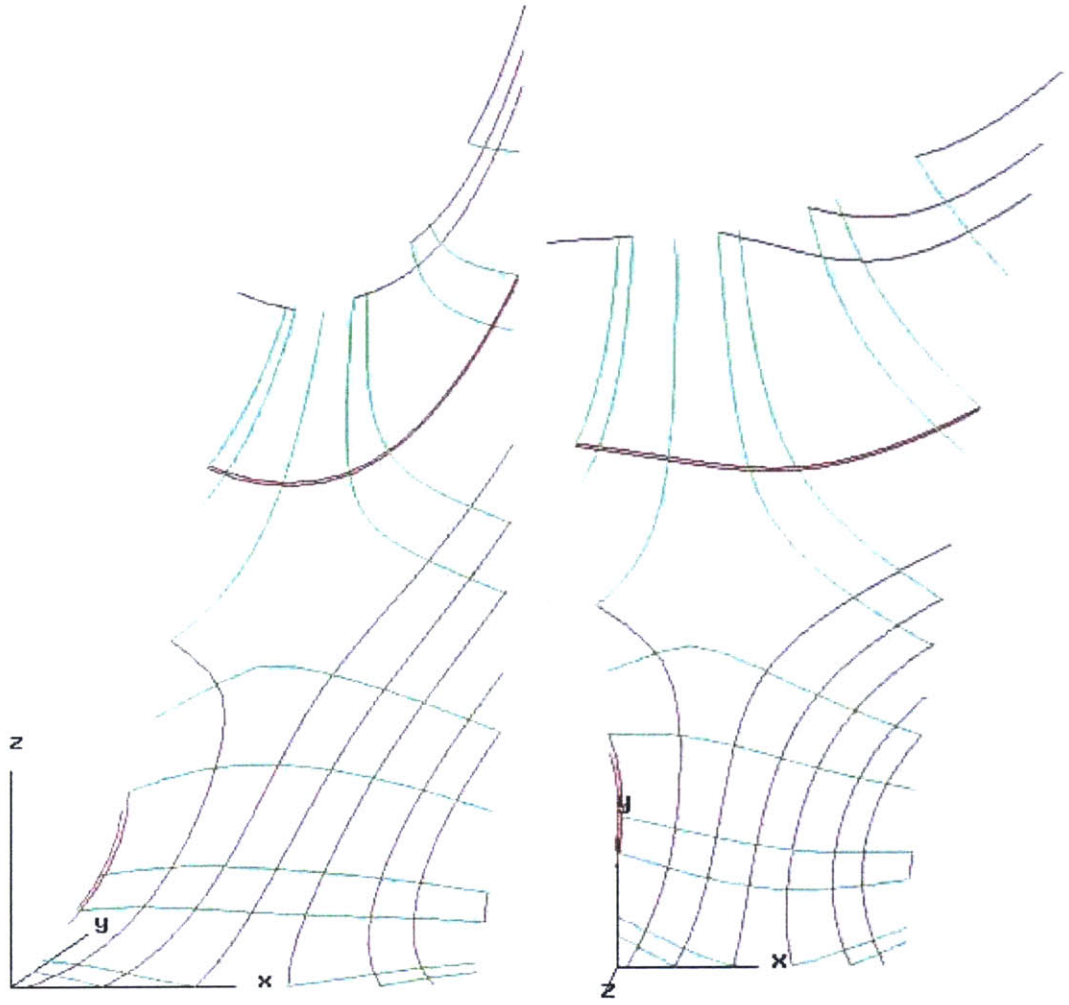
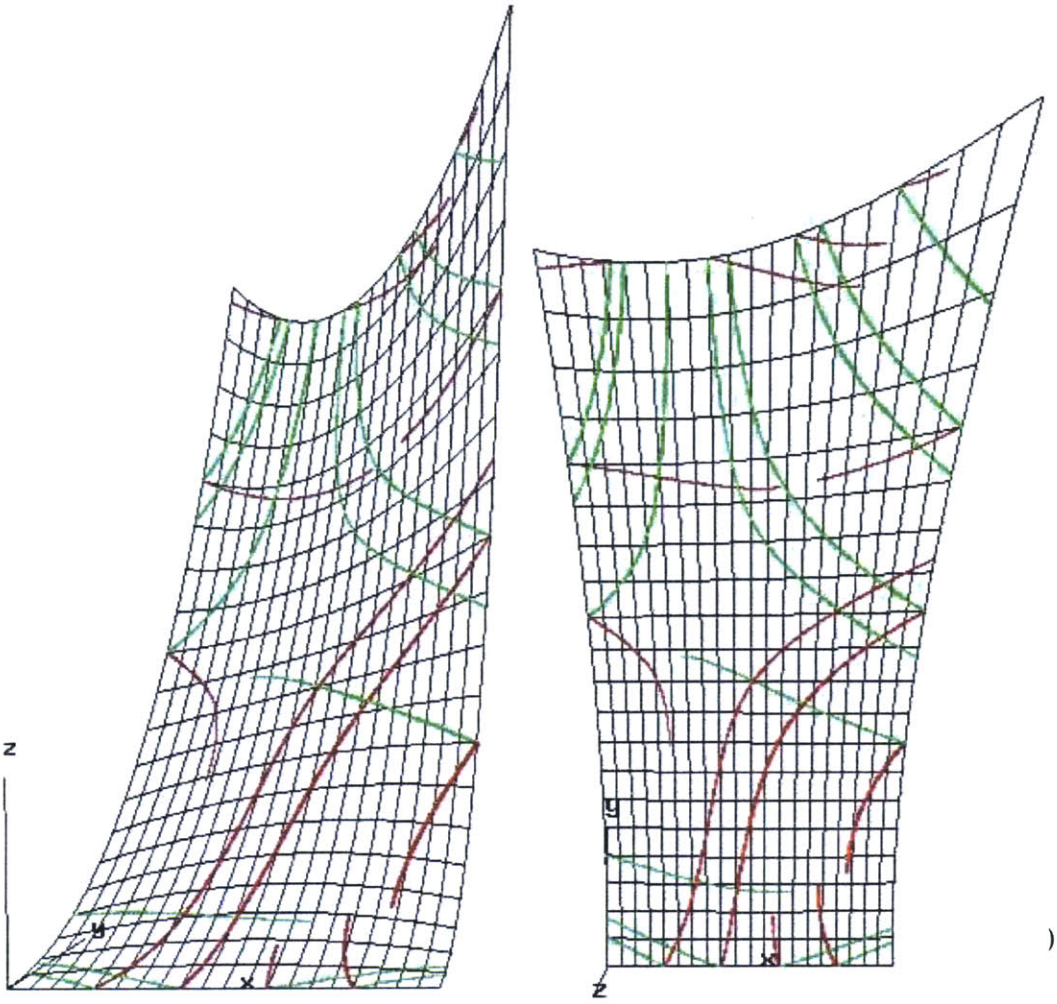


Figure 5-5: Lines of curvature from the Praxiteles program for surface $[\mathbf{P}_3](u,v)$

The lines of curvature for surface $[\mathbf{P}_3](u,v)$ are also plotted by two files generated from the Praxiteles program. By using the same equations from last example into the VNODE program, the lines of curvature can also be found. Figure 5-6 shows the lines of curvature for surface $[\mathbf{P}_3](u,v)$ in 3-D space from the VNODE program.



The lines of curvature generated from the VNODE program are the same as those generated from the Praxiteles program. However, the integration stops after some point for few lines. The starting points are chosen on each border from 0.2 to 0.8 by increasing the step size by 0.2. The length of integration is determined by not exceeding the u-v parametric space as well. The average time for integrating the length of 1 is 1.63 minutes. Tables 5-3 and 5-4 show the input in the VNODE program for lines of maximum principal curvature and lines of minimum principal curvature for surface $[\mathbf{P}_3](u,v)$

respectively.

Number	Starting Point: (u , v)	Length of Integration	Tolerance	Direction along X-axis
1	(0.2 , 0)	2.17	10^{-30}	+
2	(0.4 , 0)	1.81	10^{-30}	+
3	(0.6 , 0)	0.19	10^{-40}	-
4	(0.8 , 0)	0.32	10^{-40}	-
5	(0 , 0.6)	0.61	10^{-30}	+
6	(0 , 0.8)	0.61	10^{-30}	+
7	(0.2 , 1)	0.21	10^{-30}	-
8	(0.4 , 1)	0.42	10^{-30}	+
9	(0.6 , 1)	0.27	10^{-30}	+
10	(0.8 , 1)	0.18	10^{-30}	+
11	(1 , 0.4)	0.62	10^{-30}	-
12	(1 , 0.6)	1.81	10^{-30}	-
13	(1 , 0.8)	0.62	10^{-30}	-

Table 5-3: Input in the VNODE program for lines of maximum principal curvature for surface $[\mathbf{P}_3](u,v)$

Number	Starting Point: (u , v)	Length of Integration	Tolerance	Direction along X-axis
1	(0.2 , 0)	0.22	10^{-30}	-
2	(0.4 , 0)	0.43	10^{-30}	-
3	(0.6 , 0)	0.42	10^{-30}	+
4	(0.8 , 0)	0.22	10^{-30}	+
5	(0 , 0.2)	0.66	10^{-30}	+
6	(0 , 0.6)	1.47	10^{-30}	+
7	(0 , 0.8)	0.75	10^{-30}	+
8	(0.2 , 1)	0.91	10^{-30}	-
9	(0.4 , 1)	1.65	10^{-40}	+
10	(0.6 , 1)	0.77	10^{-30}	+
11	(0.8 , 1)	0.31	10^{-30}	+
12	(1 , 0.4)	0.80	10^{-30}	-
13	(1 , 0.6)	1.39	10^{-30}	-
14	(1 , 0.8)	0.61	10^{-30}	-

Table 5-4: Input in the VNODE program for lines of minimum principal curvature for surface $[\mathbf{P}_3](u,v)$

Chapter 6

Initial Value Problem for Geodesics on a Bezier Surface Patch

6.1 Geodesics

The geodesic path can be described as the shortest path between two points on a surface. According to Struik [32] , geodesics are curves of zero geodesic curvature. There may be more than one geodesic between two points on the surface according to this definition. An example can involve two points on a sphere. In the parametric surface $\mathbf{r} = \mathbf{r}(u, v)$ where u and v are between 0 and 1, the equations for geodesics are [28]:

$$\frac{du}{ds} = p$$

$$\frac{dv}{ds} = q$$

$$\frac{dp}{ds} = -\Gamma_1 . p^2 - 2\Gamma_2 . p . q - \Gamma_3 . q^2$$

$$\frac{dq}{ds} = -Z_1 . p^2 - 2Z_2 . p . q - Z_3 . q^2$$

where

$$\Gamma_1 = \frac{G \cdot E_u - 2 \cdot F \cdot F_u + F \cdot E_v}{2 \cdot (E \cdot G - F^2)} \qquad Z_1 = \frac{2 \cdot E \cdot F_u - E \cdot E_v + F \cdot E_u}{2 \cdot (E \cdot G - F^2)}$$

$$\Gamma_2 = \frac{G \cdot E_v - F \cdot G_u}{2 \cdot (E \cdot G - F^2)} \qquad Z_2 = \frac{E \cdot G_u - F \cdot E_v}{2 \cdot (E \cdot G - F^2)}$$

$$\Gamma_3 = \frac{2 \cdot G \cdot F_v - G \cdot G_u + F \cdot G_v}{2 \cdot (E \cdot G - F^2)} \qquad Z_3 = \frac{E \cdot G_v - 2 \cdot F \cdot F_v + F \cdot G_u}{2 \cdot (E \cdot G - F^2)}$$

E, F, G are defined in section 5.1.

E_u and E_v are the first partial derivatives of E with respect to u and v, respectively.

F_u and F_v are the first partial derivatives of F with respect to u and v, respectively.

G_u and G_v are the first partial derivatives of G with respect to u and v, respectively.

When all four boundary conditions are given at one point, the four first order ordinary differential equations above can be solved as an initial value problem. The solution for the initial value problem for geodesics is unique.

6.2 Performance of Algorithm (Example: 1)

The first example is the symmetric bi-quadratic Bezier surface $[\mathbf{P}_2](u,v)$ (Figure 5-1). Figure 6-1 shows the geodesics for surface $[\mathbf{P}_2](u,v)$ in 3-D space from the Praxiteles program.

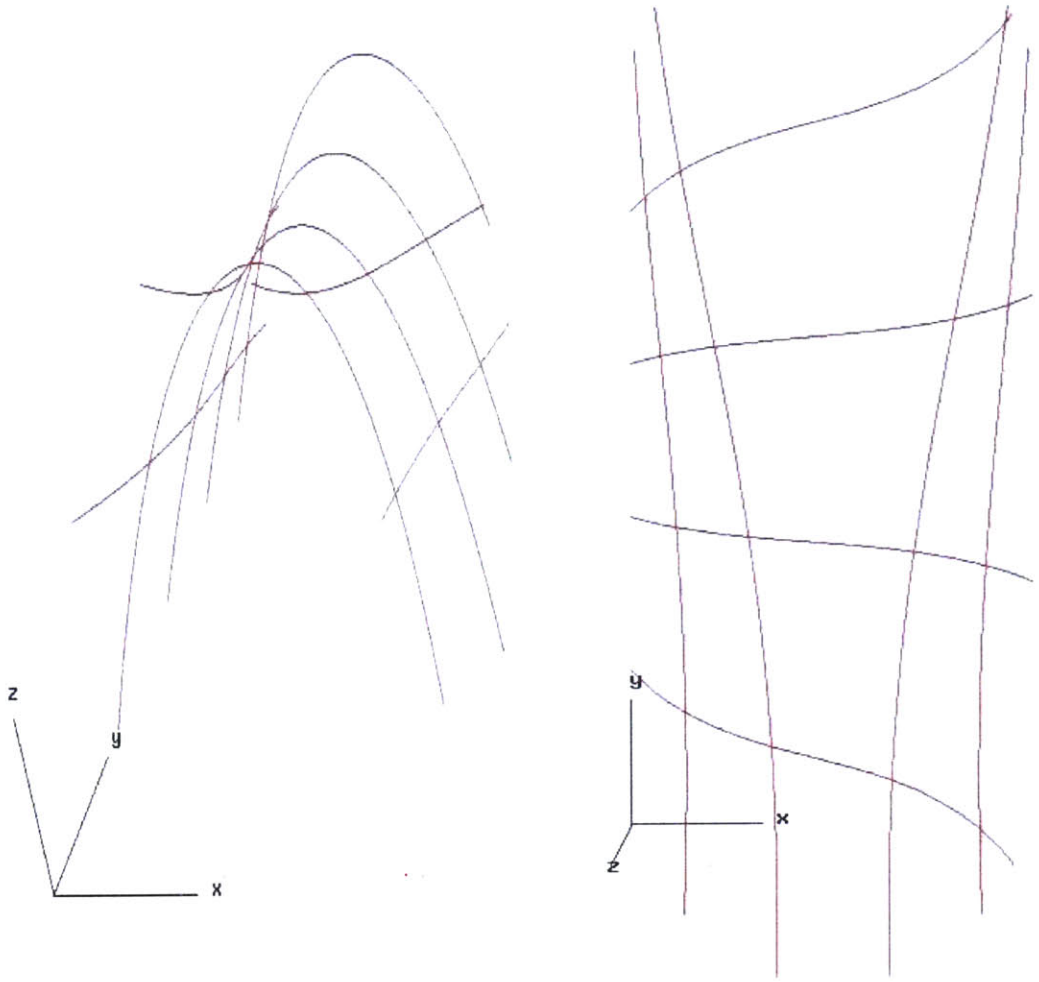


Figure 6-1: Geodesics from the Praxiteles program for surface $[\mathbf{P}_2](u,v)$

The geodesics for surface $[\mathbf{P}_2](u,v)$ are plotted by one file generated from the Praxiteles program. A few options for geodesics can be specified including the step size, how many points in the u-v parametric space of the surface can be calculated in u direction and v direction respectively in the Praxiteles program. By inputing the equations from section 6.1 into the VNODE program, the geodesics can be found. Figure 6-2 shows the geodesics for surface $[\mathbf{P}_2](u,v)$ in 3-D space from the VNODE program.

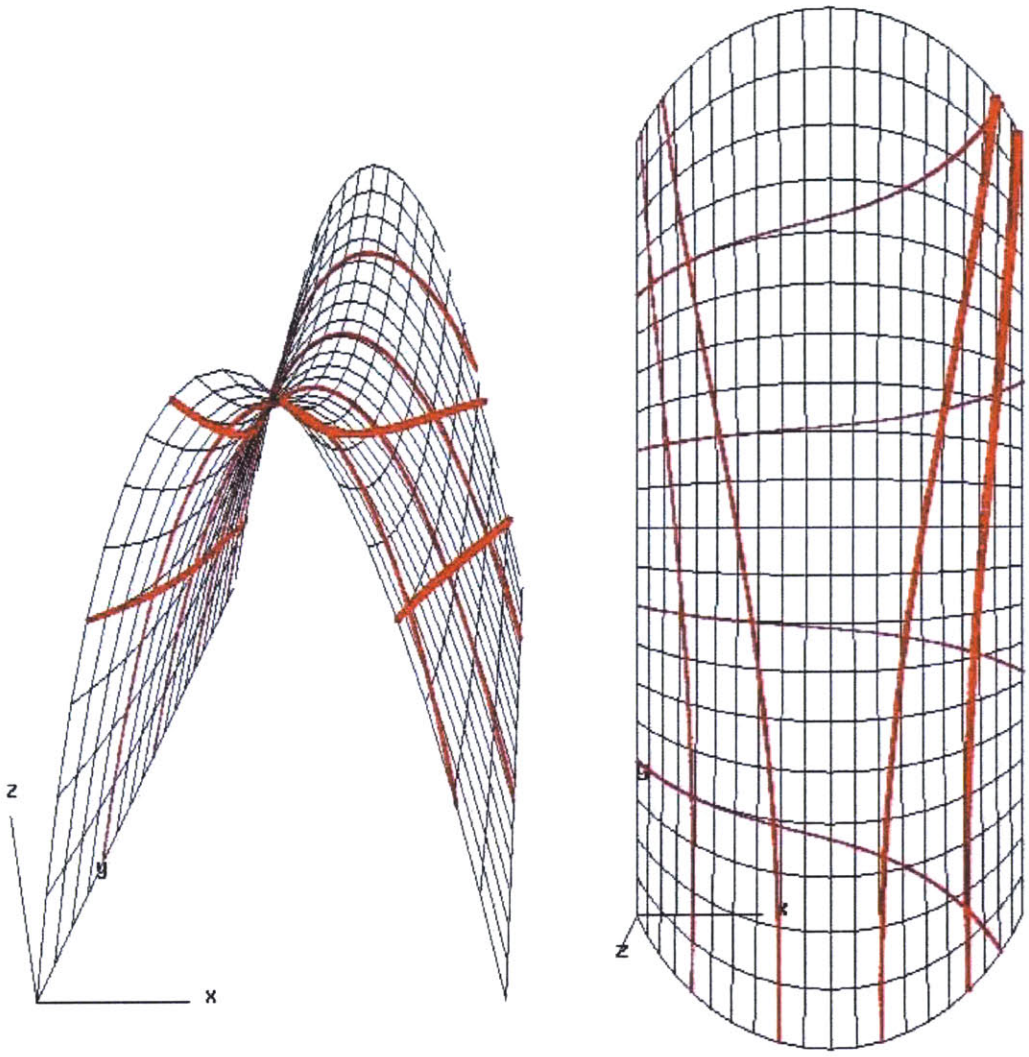


Figure 6-2: Geodesics from the VNODE program for surface $[\mathbf{P}_2](u,v)$

The geodesics generated from the VNODE program are the same as those generated from the Praxiteles program. The starting points are chosen on each border from 0.2 to 0.8 by increasing the step size by 0.2. The length of integration is determined by not exceeding the u-v parametric space as well. The average time for integrating the length of 1 is 0.42 minutes. Table 6-1 shows the input in the VNODE program for geodesics for surface $[\mathbf{P}_2](u,v)$.

Number	**Starting Point: (u , v)**	**Length of Integration**	**Tolerance**	**Direction along X-axis**
1	(0 , 0.2)	0.51	10^{-40}	-
2	(0 , 0.4)	0.60	10^{-40}	-
3	(0 , 0.6)	0.60	10^{-40}	+
4	(0 , 0.8)	0.52	10^{-40}	+
5	(0.2 , 0)	0.99	10^{-40}	+
6	(0.4 , 0)	0.95	10^{-40}	+
7	(0.6 , 0)	0.95	10^{-40}	+
8	(0.8 , 0)	0.99	10^{-40}	+

Table 6-1: Input in the VNODE program for geodesics for surface $[\mathbf{P}_2](u,v)$

6.3 Performance of Algorithm (Example: 2)

The second example is the wave-like bicubic integral Bezier surface $[\mathbf{P}_1](u,v)$ (Figure 4-1). Figure 6-3 shows the geodesics for surface $[\mathbf{P}_1](u,v)$ in 3-D space from the Praxiteles program.

Figure 6-3: Geodesics from the Praxiteles program for surface $[\mathbf{P}_1](u,v)$

By using the same equations from the last example into the VNODE program, the geodesics can be found. Figure 6-4 shows the geodesics for surface $[\mathbf{P}_1](u,v)$ in 3-D space from the VNODE program.

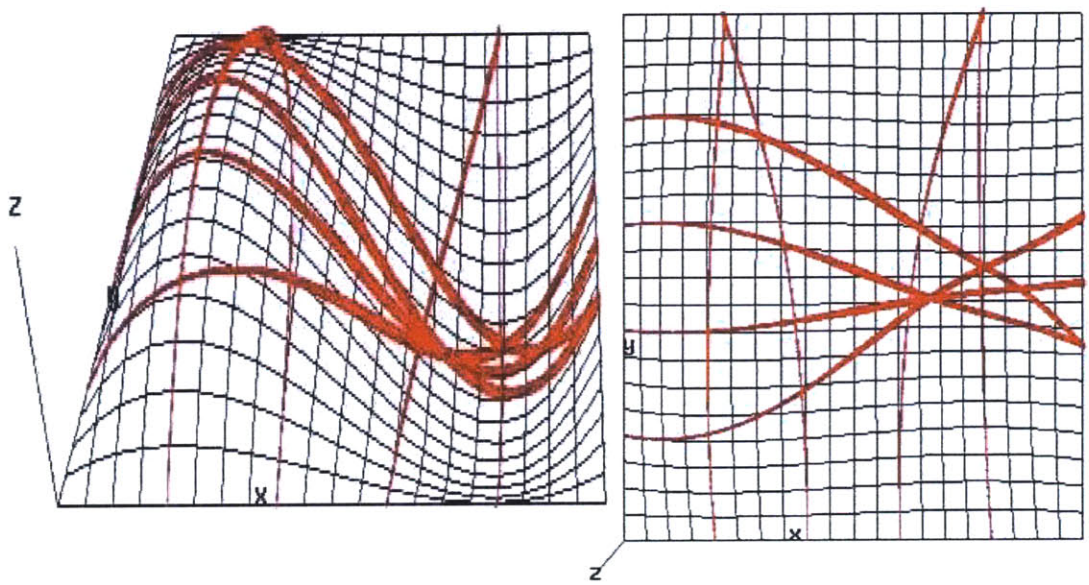


Figure 6-4: Geodesics from the VNODE program for surface $[\mathbf{P}_1](u,v)$

The geodesics generated from the VNODE program are similar as those generated from the Praxiteles program. The starting points are chosen on each border from 0.2 to 0.8 by increasing the step size by 0.2. The length of integration is determined by not exceeding the u-v parametric space as well. The average time for integrating the length of 1 is 2.14 minutes. Table 6-2 shows the input in the VNODE program for geodesics for surface $[\mathbf{P}_1](u,v)$. However, there are differences between these two programs for some geodesics. Figure 6-5 shows both the geodesics from the Praxiteles program and the VNODE program in this example. The thin lines are geodesics from the Praxiteles program and the bold lines are geodesics from the VNODE program. The reason for the difference is that the Praxiteles program uses a fixed tolerance but the VNODE program allows users to input different tolerances. Therefore, the width of the intervals from the VNODE program differ when the tolerance changes.

Number	Starting Point: (u , v)	Length of Integration	Tolerance	Direction along X-axis
1	(0 , 0.2)	0.78	10^{-40}	-
2	(0 , 0.4)	0.61	10^{-40}	-
3	(0 , 0.6)	0.65	10^{-40}	+
4	(0 , 0.8)	0.88	10^{-40}	-
5	(0.2 , 0)	0.82	10^{-40}	+
6	(0.4 , 0)	0.97	10^{-40}	+
7	(0.6 , 0)	0.97	10^{-40}	+
8	(0.8 , 0)	0.82	10^{-40}	+

Table 6-2: Input in the VNODE program for geodesics for surface $[\mathbf{P}_1](u,v)$

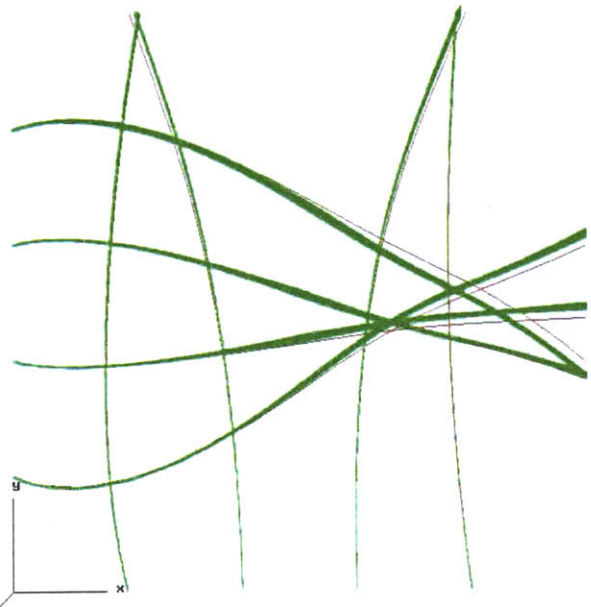


Figure 6-5: Geodesics from the Praxiteles program and the VNODE program for surface $[\mathbf{P}_1](u,v)$

Chapter 7

Conclusions and Recommendations

7.1 Conclusions

From the numerical experiments in Chapters 4, 5 and 6, robust evaluation using interval arithmetic techniques based on the VNODE software package compute the results reliably. However, there are two limitations including large computation time cost and discontinuing of the integration for some cases. Table 7-1 shows the average time for integrating the length of 1 for each experiment using VNODE.

Number	Experiment	Average time (Minute)
1	Iso-Contouring of Gaussian Curvature	1.82
2	Iso-Contouring of Mean Curvature	7.72
3	Iso-Contouring of Maximum Principal Curvature	87.42
4	Iso-Contouring of Minimum Principal Curvature	88.25
5	Lines of Curvature: Example 1	2.15
6	Lines of Curvature: Example 2	1.63
7	Geodesics: Example 1	0.42
8	Geodesics: Example 2	2.14

Table 7-1: Average time for integrating the length of 1 for each experiment

For the case of iso-contouring of curvature of a Bezier surface patch, the average time for integrating the length of 1 varies widely. The iso-contouring of principal curvatures takes almost one and half hours to complete the length of 1 in the u-v parametric space. For the cases of lines of curvature and geodesics, the average time varies under different surfaces even for the same case which means using the same formula. Therefore, the integrating time for the length of 1 in u-v parametric space will take longer when the surface becomes more complicated.

The discontinuing of the integration problem happens in sections 4.3, 5.2 and 5.3. The intervals in these examples become smaller and smaller when the integration proceeds. The intervals in the end can be very small such as 10^{-15} and this slows down the process. Therefore, those computations have to be stopped when they are under this situation. The reason for this may be that the denominator approaches 0 when the integration proceeds. Therefore, it cannot continue but proceeds slowly with a small step size.

Since the integration results from the VNODE program are mapped into 3-D space in order to show the relations between the results and the surface, the input tolerance used in the VNODE program plays an important role because it will decide the width of the mapping intervals in the 3-D space. Figure 7-1 shows the different mapping results from three different input tolerances. The top and bottom lines represent the upper and lower bounds of the mapping results from the input tolerance 10^{-15}, the bold line represent the mapping results from the input tolerance 10^{-30} and the thin line represent the mapping results from the input tolerance 10^{-40}. The width of the intervals for the lines with its input tolerance is 10^{-15} is large in comparison to others and it can not help users to determine the proper answer although the exact answer is in this interval. Therefore, the smaller input tolerance is given, the smaller width of the intervals is generated which gives the

nearest exact answer for the evaluation.

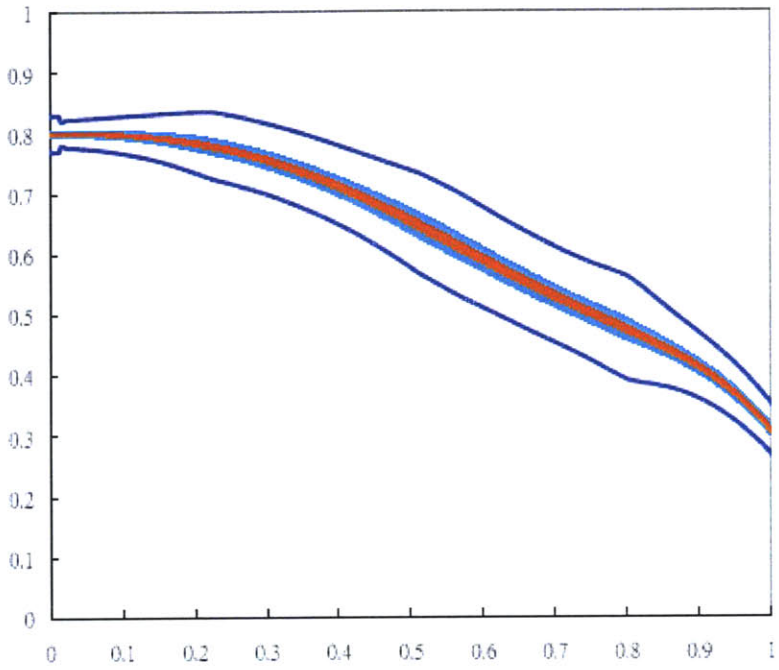


Figure 7-1: Three different mapping results from three different input tolerances

In summary, robust evaluation of differential geometry properties using interval arithmetic techniques with the VNODE program is possible and compares well with the existing program, Praxiteles, which operates in double precision floating point arithmetic and uses standard ODE solvers.

7.2 Recommendations for Future Research

Future work on the topic of robust evaluation of differential geometry properties of surfaces using interval arithmetic techniques should focus on reducing the computation time cost and solving the discontinuing integration problem.

Another direction for future research is to obtain robust evaluation of differential geometry properties on more complicated surfaces such as offsets, generalized cylinders, blending surfaces and medial surfaces.

References

[1] G. Alefeld and J. Herzberger. *Introduction to Interval Computations*. Academic Press, New York, 1983.

[2] P. G. Alourdas, *Shape Creation, Interrogation and Fairing Using B-Splines*. Engineer's thesis, Massachusetts Institute of Technology, Department of Ocean Engineering, Cambridge, Massachusetts, 1989.

[3] C. Bendsten and O. Stauning. FADBAD, a flexible C++ package for automatic differentiation using the forward and backward methods. Technical Report 1996-x5-94, Department of Mathematical Modelling, Technical University of Denmark, DK-2800, Lyngby, Denmark, August 1996.

[4] C. Bendsten and O. Stauning. TADIFF, a flexible C++ package for automatic differentiation using Taylor series. Technical Report 1997-x5-94, Department of Mathematical Modelling, Technical University of Denmark, DK-2800, Lyngby, Denmark, April 1997.

[5] M. Berz. COSY INFINITY version 8 reference manual. Technical Report MSUCL-1088, National Superconducting Cyclotron Lab., Michigan State University, East Lansing, Michigan, 1997.

[6] M. Berz and K. Makino. Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models. *Reliable Computing*, 4:361-369,1998.

[7] G. Booch. *Object-Oriented Analysis and Design*. The Benjamin/Cummings Publishing Company Inc., Rational, Santa Clara, California, 2nd edition, 1994.

[8] G. F. Corliss and R. Rihm. Validating an a priori enclosure using high-order Taylor series. In G. Alefeld and A. Frommer, editors, *Scientific Computing, Computer Arithmetic, and Validated Numerics*, pages 228-238. Akademie Verlag, Berlin, 1996.

[9] Q. Ding and B. J. Davies. *Surface Engineering Geometry for Computer-Aided Design and Manufacture.* Ellis Horwood, Chichester, UK, 1987.

[10] M. A. Ellis and B. Stroustrup. *The Annotated C++ Reference Manual.* Addison-Wesley, 1990.

[11] A. Griewank, D. Juedes, and J. Utke. ADOL-C, a package for the automatic differentiation of algorithms written in C/C++. *ACM Transactions on Mathematical Software*, 22(2):131-167, June 1996.

[12] A. Hohmann. An implementation of extrapolation codes in C++. Technical Report, Berlin, Germany, 1993.

[13] R. V. Iwaarden. IADOL-C, personal communications, 1997. IADOL-C is available through the author. E-mail: vaniwaar@metsci.com.

[14] R. Klatte, U. Kulisch, M. Neaga, D. Ratz, and C. Ullrich. *Pascal- XSC: Language Reference with examples.* Springer-Verlag, Berlin, 1992.

[15] G. A. Kriezis, N. M. Patrikalakis, and F.-E. Wolter. Topological and differential-equation methods for surface intersections. *Computer-Aided Design*, 24(1):41-55, January 1992.

[16] H. P. Langtangen. *Computational Partial Differential Equations – Numerical Methods and Diffpack Programming.* Springer-Verlag, 1999.

[17] R. J Lohner. PhD thesis, University Karlsruhe, 1988.

[18] R. J. Lohner. Enclosing the solutions of ordinary initial and boundary value problems. In E. W. Kaucher, U. W Kulisch, and C. Ullrich, editors, *Computer Arithmetic: Scientific Computation and Programming Languages*, pages 255-286. Wiley-Teubner Series in Computer Science, Stuttgart, 1987.

[19] R. J. Lohner. Step size and order control in the verified solution of IVP with ODE's,1995. SciCADE'95 International Conference on Scientific Computation and Differential Equations, Standford, California, March 28 – April 1,1995.

[20] T. Maekawa and N. M. Patrikalakis. Interrogation of differential geometry properties for design and manufacture. *The Visual Computer*, 10(4):216-237, March 1994.

[21] T. Maekawa, F.-E. Wolter, and N. M. Patrikalakis. Umbilics and lines of curvature for shape interrogation. *Computer Aided Geometric Design*, 13(2):133-161, March 1996.

[22] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, N.J., 1966.

[23] N. S. Nedialkov. *Computing Rigorous Bounds on the Solution of an Initial Value Problem for an Ordinary Differential Equation*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Canada, M5S 3G4, February 1999.

[24] N. S. Nedialkov and K. R. Jackson. An interval Hermite-Obreschkoff method for computing rigorous bounds on the solution of an initial value problem for an ordinary differential equation. *Reliable Computing*, Vol. 5, pp. 289-310,1999.

[25] N. S. Nedialkov and K. R. Jackson. *ODE Software that Computes Guaranteed Bounds on the Solution*. March 1999.

[26] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Journal of Applied Mathematics and Computation*, Vol. 105, pp. 21-68, 1999.

[27] H. Olsson. Object-oriented solvers for initial value problems. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools for Scientific Computing*. Birkhauser, 1997.

[28] N. M. Patrikalakis and T. Maekawa. Shape Interrogation for Computer Aided Design and Manufacturing. Springer-Verlag, Heidelberg, 2002.

[29] L. B. Rall. *Automatic Differentiation: Techniques and Applications*, volume 120 of Lecture Notes in Computer Science. Springer Verlag, Berlin,1981.

[30] B. Speelpenning. *Compiling Fast Partial Derivatives of Functions Given by Algorithms*. PhD thesis, Department of Computer Science, University of Illinois at

Urbana-Champaign, Urbana-Champaign, Ill., January 1980.

[31] O. Stauning. Automatic Validation of Numerical solutions. PhD thesis, Technical University of Denmark, Lyngby, Denmark, October 1997.

[32] D. J. Struik. *Lectures on Classical Differential Geometry*. Addison-Wesley, Cambridge, MA, 1950.