# Approximation with Interval B-Splines for Robust Reverse Engineering

by

Guoling Shen

B.S. in Naval Architecture and Marine Engineering (1988)
Huazhong University of Science and Technology, China

Submitted to the Department of Ocean Engineering and the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degrees of

Master of Science in Naval Architecture and Marine Engineering

and

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1997

Author ........................................................................
Department of Ocean Engineering
March 18, 1997

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

Certified by ........................................................................
David C. Gossard
Professor of Mechanical Engineering
Thesis Reader

Accepted by ........................................................................
J. Kim Vandiver
Chairman, Committee on Graduate Students
Department of Ocean Engineering

Accepted by ........................................................................
Ain A. Sonin
Chairman, Committee on Graduate Students
Department of Mechanical Engineering

Eng.

*Dedicated to the Memory of my Mother*

**Aibao Zheng**

# Approximation with Interval B-Splines for Robust Reverse Engineering

by

Guoling Shen

Submitted to the Department of Ocean Engineering
and the Department of Mechanical Engineering
on April 28, 1997, in partial fulfillment of the
requirements for the degrees of
Master of Science in Naval Architecture and Marine Engineering
and
Master of Science in Mechanical Engineering

## Abstract

The advent of accurate 3D laser ranging sensors that provide the capability of producing great amounts of precise data points in extremely short time, has created interest in automated methods for reverse engineering, i.e., the process of creating a complete CAD model from measured data. A typical application of reverse engineering, for example, is to build a CAD model from a manufactured object which has no accurate electronic model, for further design or inspection.

However, since measured data always possess uncertainty arising from sensor precision and measurement registration, most algorithms for surface reconstruction may produce an ambiguously defined object and cause a solid modeling system to fail due to possible numerical instabilities when manipulating the geometric and topological representation with complex operations. Even in the absence of measurement uncertainty, such a failure could still happen due to the limitation of computation precision.

This thesis addresses the robustness problem concerning surface reconstruction for reverse engineering and proposes a robust and reliable approach dealing with both uncertainty of measured data and computational error. In our approach, measured data are represented as interval data points involving all sources of errors, and interval surface fitting methods are developed to make it possible to construct interval solid models. These models can be processed robustly using operations based on rounded interval arithmetic within an interval solid modeling system. The creation of interval surfaces is accomplished by either interpolation or approximation. A linear programming process is used to minimize the range (thickness) of the resulting surfaces. Our interpolation technique guarantees to enclose interval data points with interval ranges almost equal to but strictly larger than those of interval data points at the interpolation locations, while our approximation technique nearly matches interval data points but still forms an enclosure around them. Examples using measured data illustrate our method.

In addition, this thesis also develops and discusses some unique geometric and numerical properties that interval B-spline curves and surfaces have as they are defined and manipulated in a new system of arithmetic (rounded interval arithmetic (RIA)), such as error propagation in the RIA evaluation of interval B-spline basis functions, coordinate translation of interval B-splines, hodograph of an interval B-spline curve, enveloping curves or surfaces of an interval B-spline curve or surface, and knot insertion for interval B-splines.

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

Thesis Reader: David C. Gossard
Title: Professor of Mechanical Engineering

## Acknowledgments

First of all, I want to thank my family for their love, understanding, and support, especially my father.

I would like to thank my advisor, Professor Nicholas M. Patrikalakis, for his expert advice on my research work and instructive guidance on my academic study. I also want to thank Professor David C. Gossard for taking time to read the thesis. Thanks also go to Dr. Takashi Maekawa for his constructive suggestions on the work and the thesis, and former Design Laboratory postdoctoral associates, Dr. Seamus T. Tuohy and Dr. Xiuzi Ye, for their assistance. I also would like to thank Professor Takis Sakkalis for his comments.

I also want to thank Professor John J. Leonard for providing information about the Robotic Measurement System and the data file of one of the examples in this thesis, Mr. Stephen L. Abrams for providing a stable hardware environment and programming assistance. Design Laboratory fellows Dr. Wonjoon Cho, Mr. Todd R. Jackson, Mr. Michael O. Jastram, and Mr. Guoxin Yu also helped me in various aspects.

Special thanks to my friends Chao Chen and Lin Zhang who gave me the first motivation to pursue further education in the United States and substantial assistance later on my way to MIT.

The work presented in the thesis was supported by the Office of Naval Research under grant numbers N00014-94-1-1001, N00014-95-1-0184, and N00014-96-1-0857.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation and Problem Statement

The current development in 3D sensing technology enables range sensors to quickly acquire data sets with high precision in large amounts. For instance, 4D Imager, a newly developed 3D range sensor, is capable of measuring 50,000 3D points on objects in under 100 milliseconds with a data acquisition time of less than 0.1 milliseconds and a standard measurement deviation of only 1/2000 of the field of view, see [11]. Such a powerful capability of measurement has increased interest in automated methods for reverse engineering, i.e. the process of creating a complete CAD model from measured data. Applications of reverse engineering, for example, are: building a CAD model from a manufactured object which has no electronic model, for interrogation or further design; inspecting a manufactured object or repairing a deformed object via comparison with its original CAD model. Typical to the reverse engineering field are developments in three areas: (1) devices for data collection; (2) registration of data from multiple views; and, (3) surface fitting and topological representations.

In this thesis, we address the problem concerning the robustness of surface reconstruction for reverse engineering.

State-of-the-art sensors are capable of efficiently providing dense sets of x, y, z data points with high position accuracy. However, as no infinite precision can be achieved, measured data are always associated with uncertainties arising from limited measurement machine precision. Moreover, additional errors are to be introduced when data points from different views are registered to create a complete, unambiguous data set. Therefore, general surface fitting algorithms only create one possible geometric shape that a real object could have, without any report about the deviation of this assumed shape from the real one. Furthermore, most algorithms for surface reconstruction require the manipulation of the geometric and topological representation with complex operations (e.g. Booleans) which may lead to numerical instabilities, because the representation is created by fitting uncertain data and therefore is not precise both geometrically and topologically.

Even in the absence of measurement uncertainty, all state-of-the-art CAD systems based on *floating point arithmetic* (*FPA*) frequently fail. The ultimate reason for this failure is that the precision of most geometrically important computations is practically limited so that conceptually continuous objects are actually represented and analyzed in a discrete domain on a real computer, see Hoffmann [14]. Thus, the floating point representation of a geometric object is only an approximation. As a solid object is specified by numerical (vertex coordinates, surface equations, etc.) and topological (adjacency, incidence, etc.) information, the impreciseness of arithmetic representations may lead to contradiction; for a solid object bordered by free-form surface patches and described with a B-rep, the boundaries of the solid created by assembling a collection of surface patches (of limited precision) according to topological information, will frequently have undesired **gaps**, see Hu et al [21, 22], which will be magnified when the geometry is based on data with measurement uncertainty. These gaps will, therefore, yield ambiguities in the definition of point sets contained in the interior of the solid. We may get vaguely defined solids which yield even more vaguely defined objects when they are used in Boolean operations, such as building an object from intersections of solid volumes.

For a possible future development of a consistent modeling system dealing with uncertainty in non-linear geometric models (and of operations on these models), we propose a geometric and topological representation paradigm for reverse engineering using methods for robust solid modeling based on *rounded interval arithmetic* (*RIA*).

In our approach, instead of using plain floating point representations of B-spline patches, we use a new representation based on *interval B-splines*. Particularly, in this thesis, we develop the methods for the creation of interval curves and surfaces from data with associated bounds on measurement error. These methods assist in automatic and robust reconstruction of an artifact. The main work of this thesis is summarized in [45].

## 1.2 Previous Work

### 1.2.1 Interval Arithmetic and Interval B-Splines

Interval arithmetic was first introduced by numerical analysts in the early sixties in their effort to seek guaranteed enclosures of numerical computations in the computer. It is a well-defined mathematical system consisting of definitions and operation rules, see [13, 27, 32] for detailed descriptions. Much research has been done on various topics of interval analysis, but only recently *interval arithmetic* has begun to be used in the fields of geometric modeling, computer graphics and CAD.

Mudur and Koparkar [28], Toth [42], Enger [9], Duff [8] and Snyder [39] applied interval algorithms to geometry processing. Interval Bézier curves were first studied and applied by Sederberg and Farouki [38] and by Sederberg and Buehler [37] in curve approximation. Interval explicit B-spline surfaces were studied by Tuohy and Patrikalakis [46]. Interval arithmetic is also used in nonlinear polynomial system solvers in Maekawa and Patrikalakis [25], in intersection problems in Hu et al [19, 20], in interval solid modeling in Hu et al [21, 22], in design automation in Bliek [4], in robust visualization in Hu [17] and Tuohy et al [47], and in robotics in Tuohy et al [44] and Hager [12].

### 1.2.2 Surface Reconstruction for Reverse Engineering

Many algorithms of surface reconstruction for reverse engineering have been proposed in the recent years. The following is a brief survey.

In Hoover *et al* [15], a polyhedral boundary representation (B-rep) model is constructed from multiple range images by Boolean intersections. In this method, each image is developed as a complete B-rep with polyhedral surfaces and each image contains a set of regions representing the union of the real object and the projection of the object onto the viewing plane. After these separate images are collected, the final B-rep is composed from the intersection of all of the different views. If the original B-reps are precise then the final B-rep should be correct. However, many sources of error were identified due to noise in the collected data which lead to instabilities in the final intersection operation and necessitate complex surface stitching algorithms to complete the B-rep model.

For nonlinear polynomial surface patches, Sapidis and Besl [35] have extended the *region-growing* technique developed earlier by Besl [3] to automate the construction of surface patches from range (or image) data. This method, however, still leaves patches that can be position discontinuous. These patches subsequently need to be joined together. For B-spline patches, Milroy *et al* [26] use a smoothing algorithm to impose $G^1$ continuity across adjacent patches. The algorithm depends on direct user manipulation to define the patch boundary curves needed for position continuity to attain a "watertight CAD model".

The representation of uncertainty for reverse engineering applications has been identified as a critical need, see, for example, Soucy and Laurendeau [40]. In their work, a *spatial neighborhood test*, which relies on the formulation of uncertainty polyhedra, is used to delineate discontinuities from common surface elements over successive range images. This is useful in edge detection and data registration (identifying the common or overlapping information content from two or more range views).

## 1.3 Organization of the Thesis

Chapter 2 is a brief review of *interval arithmetic, rounded interval arithmetic* and *interval B-splines.* It presents some definitions and operation rules in interval analysis, and also discusses some unique geometric and numerical properties of interval B-splines.

Chapter 3 develops a method of interval curve interpolation, which also provides the foundation of the methods of interval curve and surface approximation described in Chapter 4. Some differences between interval data interpolation and approximation are discussed. Chapter 4 also presents an approximation method for extremely dense data sets.

In Chapter 5, we first discuss implementation issues. Then, numerical results are presented, demonstrating the methods proposed in the previous chapters.

Chapter 6 concludes the thesis and provides some discussion about future work.

Appendix A presents a coordinate translation method for an interval B-spline curve. Appendix B proves that knot insertion in interval B-spline curves preserves a conservative enclosure. Appendix C gives the pseudocodes of some important algorithms presented in this thesis.

# Chapter 2

# Interval Geometric Representation

## 2.1 Introduction

Interval analysis first emerged as a tool "in numerical mathematics to enable digital computers to execute algorithms capturing all the roundoff errors automatically" [2]. It is a well-defined arithmetic system referred as *interval arithmetic*, consisting of some basic definitions and operation rules, as well as some interval analysis methods developed for various applications. The advantage of interval analysis is that all computations are carried on with error bounds such that a conservative enclosure is preserved. Closed ranges, in general, are returned as results instead of single values which are most likely only approximations to exact results due to computation errors, even if initial data are exact.

Although interval arithmetic was introduced many years ago, only in recent years, it found applications in the areas of *computer aided design* and *computer aided geometric design*. Researchers in these areas started to use interval analysis methods to solve robustness problems such as the numerical instability in a solid modeling system, or to process uncertain data from the real world. Interval geometries can be described by *interval B-splines* which are defined as classical B-splines but with interval coefficients. An interval B-spline curve is a slender tube consisting of a family of single-valued curves whose control points are located inside the rectangular ranges defined by interval control points. Similarly, an interval B-spline surface is a thin shell defining a volume in 3D space.

In this chapter, we first give the definition of an interval number, and the rules as well as some theorems of interval operations. Then, as the implementation on a real digital computer, we give the operation rules of *rounded interval arithmetic.* At the end, we introduce interval B-spline curves and surfaces, and discuss some of their unique properties.

## 2.2 Interval and Rounded Interval Arithmetic

### 2.2.1 Interval Arithmetic

**Definition** An interval number $X$ is defined as the set of real numbers $x \in \Re$ such that:

$$\{x | X^l \leq x \leq X^u\} \tag{2.1}$$

where real numbers $X^u$ and $X^l$ refer to the upper (or maximum) and lower (or minimum) bounds, respectively, of the interval.

An interval number is denoted as $[X]$, and the set of real intervals as $I(\Re)$. Furthermore, the *mean* or midpoint is defined as

$$\bar{X} = \frac{X^l + X^u}{2}, \tag{2.2}$$

and the *width* or range as

$$w([X]) = X^u - X^l. \tag{2.3}$$

The absolute value of an interval number is defined as

$$|[X]| = \max\{|X^l|, |X^u|\}. \tag{2.4}$$

The ordering of two interval numbers is defined by

$$[X] \leq [Y] \quad \text{if and only if } X^u \leq Y^l. \tag{2.5}$$

The distance between two interval numbers $[X], [Y]$ is defined as

$$d([X],[Y]) = \max\{|X^l - Y^l|, |X^u - Y^u|\}, \tag{2.6}$$

which gives $I(\Re)$ a metric topology.

An interval number is generally represented in the endpoint form $[X^l, X^u]$. With the above definitions, an interval number sometimes is represented in the midpoint-halfwidth form $[X] = [\,\bar{X} - w([X])/2, \bar{X} + w([X])/2\,]$.

Interval arithmetic operations are defined for the usual algebraic operators $\circ \in \{+, -, \cdot, /\}$ by

$$X \circ Y = \{x \circ y | x \in X, y \in Y\}, \tag{2.7}$$

or more explicitly,

$$\begin{aligned}
\left[X^l, X^u\right] + \left[Y^l, Y^u\right] &= \left[X^l + Y^l, X^u + Y^u\right], \qquad (2.8)\\
\left[X^l, X^u\right] - \left[Y^l, Y^u\right] &= \left[X^l - Y^u, X^u - Y^l\right], \\
\left[X^l, X^u\right] \bullet \left[Y^l, Y^u\right] &= \left[\min(X^lY^l, X^lY^u, X^uY^l, X^uY^u),\right. \\
&\quad \left.\max(X^lY^l, X^lY^u, X^uY^l, X^uY^u)\right], \\
\frac{\left[X^l, X^u\right]}{\left[Y^l, Y^u\right]} &= \left[X^l, X^u\right] \bullet \left[\frac{1}{Y^u}, \frac{1}{Y^l}\right],
\end{aligned}$$

where in the division formula, $0 \notin [Y^l, Y^u]$. Note that each operation produces a new interval, i.e. $I(\Re)$ is closed with respect to the arithmetic operations defined above.

We now present some theorems, which will be used in the following sections. For a proof, see [2].

**Theorem 1** $[A], [B]$, and $[C]$ are members of $I(\Re)$. It follows that

1. $[A] + [B] = [B] + [A]$, $[A] \cdot [B] = [B] \cdot [A]$ (commutativity),
2. $([A] + [B]) + [C] = [A] + ([B] + [C])$, $([A] \cdot [B]) \cdot [C] = [A] \cdot ([B] \cdot [C])$ (associativity),
3. $[A]([B] + [C]) \subseteq [A][B] + [A][C]$ (subdistributivity),

4. $a([B]+[C]) = a[B] + a[C], \quad a \in \Re$.

**Theorem 2** Let $[A], [B]$ be intervals, then

1. $w([A] \pm [B]) = w([A]) + w([B])$,
2. $w([A][B]) \leq w([A])|[B]| + w([B])|[A]|$,
3. $w([A][B]) \geq \max\{|[A]|w([B]), |[B]|w([A])\}$,
4. $w(a[B]) = |a|w([B]), \quad a \in \Re$.

### 2.2.2 Rounded Interval Arithmetic

The interval operations defined in Equation (2.8) are thus called *exact interval arithmetic*, and only possible on an ideal computer without precision limitation. Instead, on a real computer with *floating point arithmetic* (*FPA*), rounded interval operators, referred to as *rounded interval arithmetic* (*RIA*), are implemented, leading to conservative enclosures.

For example, the IEEE standard double-precision has 64 bits, 8 bytes wordsize, and is stored in a binary form as $(\pm)m \cdot 2^{exp}$, where $m$ is the *mantissa* $(0.5 \leq m < 1)$ and $exp$ is the *exponent*. Figure 2-1 illustrates the binary form by which data are stored on a machine. As two consecutive positive double-precision numbers differ by an amount $\epsilon$ called *ulp* (one Unit in the Last Place), and $\epsilon = 2^{exp-53}$, we can redefine (2.7) such that all machine operations satisfy the *inclusion principle*

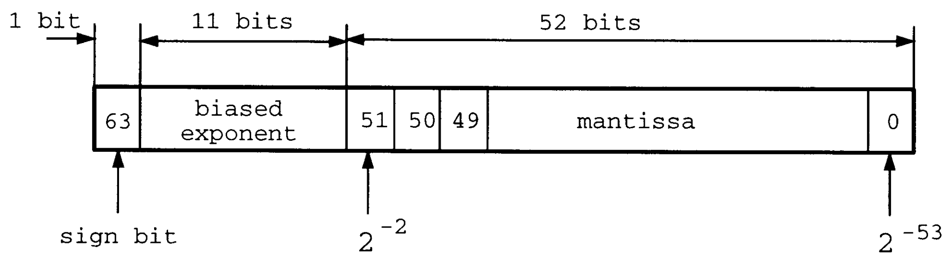


Figure 2-1: IEEE format for binary representation of double-precision floating-point number

$$\left[X^l, X^u\right] \circ \left[Y^l, Y^u\right] = \left[Z^l, Z^u\right] \in \left[Z^l - \epsilon^l, Z^u + \epsilon^u\right] \tag{2.9}$$

where $\epsilon^u$ or $\epsilon^l$ are associated with $Z^u$ or $Z^l$, respectively. Using (2.9), equation (2.8) can be rewritten as follows

$$\begin{aligned}
\left[X^l, X^u\right] + \left[Y^l, Y^u\right] &= \left[X^l + Y^l - \epsilon^l, X^u + Y^u + \epsilon^u\right] \qquad (2.10)\\
\left[X^l, X^u\right] - \left[Y^l, Y^u\right] &= \left[X^l - Y^u - \epsilon^l, X^u - Y^l + \epsilon^u\right]\\
\left[X^l, X^u\right] \bullet \left[Y^l, Y^u\right] &= \left[\min(X^lY^l, X^lY^u, X^uY^l, X^uY^u) - \epsilon^l,\right.\\
&\quad \left.\max(X^lY^l, X^lY^u, X^uY^l, X^uY^u) + \epsilon^u\right]\\
\frac{\left[X^l, X^u\right]}{\left[Y^l, Y^u\right]} &= \left[\min(X^l/Y^l, X^l/Y^u, X^u/Y^l, X^u/Y^u) - \epsilon^l,\right.\\
&\quad \left.\max(X^l/Y^l, X^l/Y^u, X^u/Y^l, X^u/Y^u) + \epsilon^u\right]
\end{aligned}$$

We continue by proving two simple theorems on numbers computed on a machine with limited precision, which will be used later.

**Theorem 3** Let $p$ be a number computed on a machine with a binary form that has $N > 1$ digits for mantissa. Let $\epsilon$ be the associated *ulp*. Then

$$|p| > \epsilon \tag{2.11}$$

**Proof:** Assume

$$m = 0.abcd\cdots, \quad \text{where } a \neq 0.$$

Then

$$\begin{aligned}
|p| &= m \cdot 2^{exp} = (0.abcd\cdots) \cdot 2^{exp}\\
&= (a.bcd\cdots) \cdot 2^{exp-1}.
\end{aligned}$$

Since $a \geq 1$ and $N > 1$, $|p| \geq 2^{exp-1} > 2^{exp-N}$. Therefore, $|p| > \epsilon$.

$\square$

**Theorem 4** Let $p_1$ and $p_2$ be two numbers computed on the same computer as in Theorem 3, and their associated *ulp*'s are $\epsilon_1$ and $\epsilon_2$, respectively. If $|p_1| \geq |p_2|$, then

$$\epsilon_1 \geq \epsilon_2 \tag{2.12}$$

**Proof:** Assume $|p_1| = m_1 \cdot 2^{exp_1}, |p_2| = m_2 \cdot 2^{exp_2}$. From the hypothesis, we have $m_1 \cdot 2^{exp_1} \geq m_2 \cdot 2^{exp_2}$. If $m_1 < m_2$, then $exp_1 > exp_2$. If $m_1 \geq m_2$, assume $exp_1 < exp_2$. Then at least $exp_2 = exp_1 + 1$ since exponents are integers. Therefore

$$|p_2| = m_2 \cdot 2^{exp_2} = m_2 \cdot 2^{exp_1+1} = 2m_2 \cdot 2^{exp_1}.$$

Since $0.5 \leq m_1, m_2 < 1$,

$$|p_2| \geq 2^{exp_1} > m_1 \cdot 2^{exp_1} = |p_1|,$$

i.e. $|p_1| < |p_2|$, which contradicts the hypothesis $|p_1| \geq |p_2|$. Therefore, $exp_1 \geq exp_2$. Since $\epsilon_1 = 2^{exp_1 - N}, \epsilon_2 = 2^{exp_2 - N}$, we conclude that $\epsilon_1 \geq \epsilon_2$.

□

## 2.3 Interval B-Spline Curves and Surfaces

This section is a review of basic B-spline properties with an introduction to interval B-spline curves and surfaces. For a more complete review of the B-spline properties, see [6, 16].

If $T$ is a set of *real numbers* such that

$$t_i \leq t_{i+1}, \quad 0 \leq i \leq n + M - 1, \tag{2.13}$$

where $n$ is the number of B-spline coefficients, a real function $f(t)$ in the domain

$[t_0, t_{n+M-1}]$ is called a spline of order $M$ or degree $M-1$ if

1. $f(t)$ is a polynomial of degree $M-1$ on each subinterval $[t_i, t_{i+1}]$;
2. $f(t)$ and its first $M-p-1$ derivatives are continuous in the entire domain, where $p$ is the knot multiplicity;
3. moreover, its higher derivatives are continuous everywhere except at $t_i, 0 \leq i \leq n+M-1$.

$T = \{t_0, t_1, \ldots, t_{n+M-1}\}$ is called the knot vector and the values $t_M, \ldots, t_{n-1}$ are called the internal knots of the B-spline function. An open knot vector is constructed by setting the first and last $M$ knots to be all equal to fixed values, usually 0 and 1, respectively.

A recursive expression for the B-spline basis functions $N_{i,M}(t)$ is [6]:

$$\begin{aligned} N_{i,1} &= \begin{cases} 1 & \text{if } t_i \leq t < t_{i+1}, \\ 0 & \text{otherwise;} \end{cases} \\ \text{or} & \quad \text{if} \quad M > 1, \\ N_{i,M}(t) &= \frac{t - t_i}{t_{i+M-1} - t_i} N_{i,M-1}(t) + \frac{t_{i+M} - t}{t_{i+M} - t_{i+1}} N_{i+1,M-1}(t). \end{aligned} \tag{2.14}$$

Figure 2-2 shows the cubic B-spline basis $N_{i,4}(t), 0 \leq i \leq 6$, defined over knot vector

$$T = \{0.0, 0.0, 0.0, 0.0, 0.3, 0.4, 0.5, 1.0, 1.0, 1.0, 1.0\}.$$

A B-spline curve is defined by a piecewise polynomial in the B-spline basis over knot vector $T$ and coefficients $\vec{P}_i, 0 \leq i \leq n-1$, which are the vertices of the control polygon $\vec{P}$ and called control points. For an interval B-spline curve, each control point is represented by an interval vector $[\vec{D}]_i = \left[\vec{D}_i^l, \vec{D}_i^u\right]$, which defines a rectangular range, such that:

$$[\vec{P}]_M(t) \quad = \quad \left[\vec{P}_M^l(t), \vec{P}_M^u(t)\right]$$

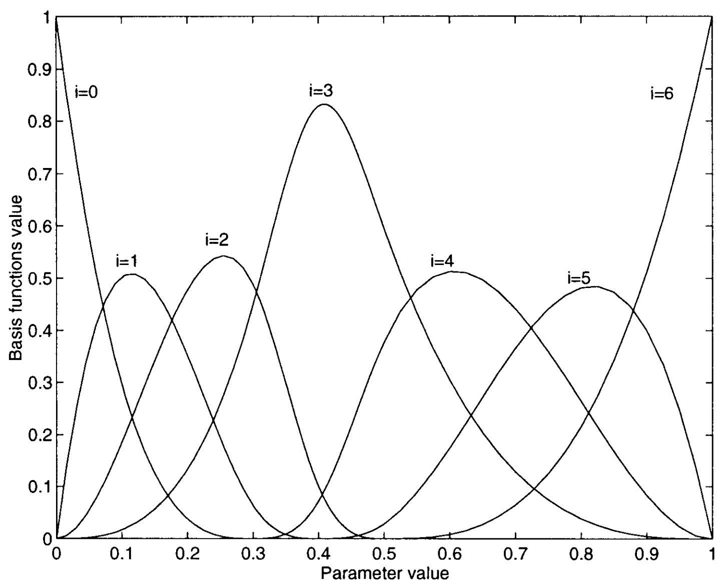


Figure 2-2: B-spline basis $N_{i,4}(t), 0 \leq i \leq 6$

$$= \sum_{i=0}^{n-1} [\vec{D}]_i N_{i,M}(t), \tag{2.15}$$

where $N_{i,M}(t)$ is the B-spline basis function assumed to be computed exactly with rational arithmetic. Equation (2.15) represents a slender tube consisting of a family of curves whose control points are located inside the rectangular ranges defined by the interval control points, see Figure 2-3 for an example. In practice, the knot vector and the interval coefficients are expressed in terms of floating point numbers.

A B-spline surface is a tensor product surface defined by two knot vectors $T$ and $S$ associated with parameters $u$ and $v$ respectively, and by a topologically rectangular set of control points $\vec{P}_{i,j}, 0 \leq i \leq m-1, 0 \leq j \leq n-1$, which are the vertices of the control polyhedron $\vec{P}$. For an interval B-spline surface patch, each control point is represented by an interval vector $[\vec{Q}]_{i,j} = \left[\vec{Q}^l_{i,j}, \vec{Q}^u_{i,j}\right]$ and the patch definition is

$$\begin{aligned}
[\vec{P}]_{M,N}(u,v) &= \left[\vec{P}^l_{M,N}(u,v), \vec{P}^u_{M,N}(u,v)\right] \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} [\vec{Q}]_{i,j} N_{i,M}(u) N_{j,N}(v). \qquad (2.16)
\end{aligned}$$

Similarly, it is a thin shell in the space, enveloping a family of B-spline surfaces whose

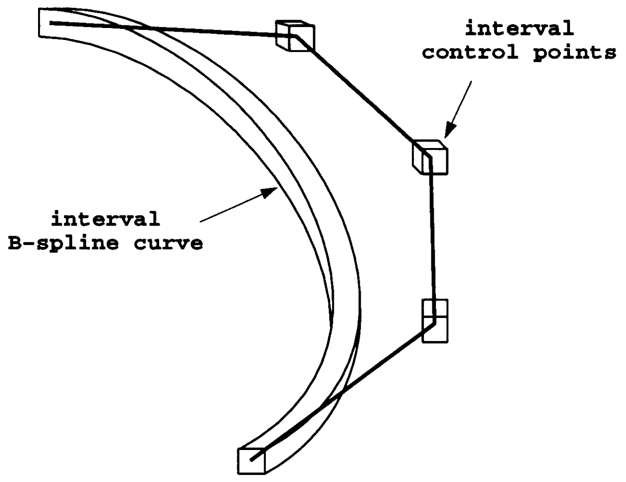


Figure 2-3: Interval B-spline curve

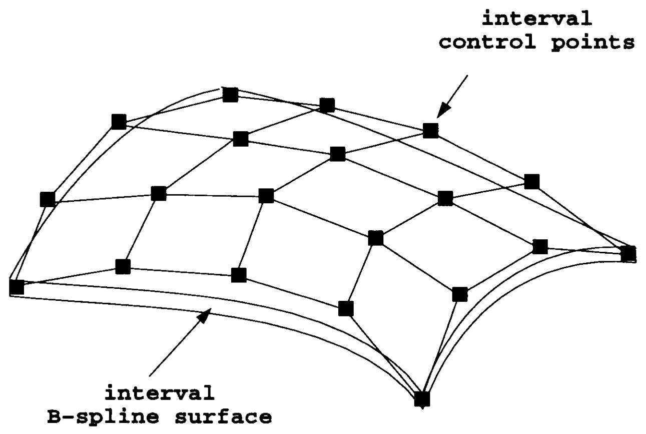


Figure 2-4: Interval B-spline surface

control points are inside the corresponding interval control points, see Figure 2-4.

## 2.4 Geometric and Numerical Properties of Interval B-Spline Curves and Surfaces

Most properties of classical B-spline curves and surfaces still apply to their interval counterparts. But as defined and manipulated in a new system of arithmetic (RIA), interval B-spline curves and surfaces have some unique properties, as described below.

### 2.4.1 Error Propagation in the RIA Evaluation of Basis Functions

The exact value of a basis function $N_{i,M}(t)$ at some floating point parameter value can be calculated exactly using rational arithmetic if the knots are given as floating point numbers. However, we opt to use RIA which guarantees $[N]_{i,M}(t) \supset N_{i,M}(t)$, because evaluation of $N_{i,M}(t)$ using rational arithmetic drastically increases computation time.

We now study the error propagation in the RIA evaluation of the B-spline basis, and will give a loose but still useful upper bound for $w([N]_{i,M}(t))$. This estimation is useful in the derivation of the subsequent sections of this chapter and in solid modeling and fitting applications. Chapter 5 will explain some implementation details about the interval evaluation of the basis functions.

As stated in Section 2.2.2, a floating point number is represented and stored as $(\pm)m \cdot 2^{exp}$ on the computer with the IEEE standard double-precision; the round-off error arising from any operation is taken into consideration by enlarging the computed interval at both directions by $\epsilon = 2^{exp-53}$, where *exp* is extracted from the computed lower and upper bounds of the interval, respectively. For instance, if a real number is in $[0,1]$, then its *exponent* is no larger than 1, and $\epsilon \leq 2^{-52}$. We will use this $\epsilon$ in the following.

From the recursive formula (2.14), the interval evaluation of the basis functions is defined as

$$\begin{aligned} [N]_{i,1} &= \begin{cases} [1,1] & \text{if } t_i \leq t < t_{i+1}, \\ [0,0] & \text{otherwise;} \end{cases} \\ \text{or} & \text{ if } \; M > 1, \\ [N]_{i,M}(t) &= [A][N]_{i,M-1}(t) + [B][N]_{i+1,M-1}(t), \end{aligned} \tag{2.17}$$

where

$$[A] = \text{RIA evaluation of } \left(\frac{t - t_i}{t_{i+M-1} - t_i}\right), \tag{2.18}$$

$$[B] \quad = \quad \text{RIA evaluation of } \left(\frac{t_{i+M}-t}{t_{i+M}-t_{i+1}}\right).$$

Since only $[N]_{i,1}, \cdots, [N]_{i+M-1,1}$ appear in the recursive formula calculating $[N]_{i,M}$, and they are all $[0,0]$ if $t \notin [t_i, t_{i+M-1}]$, here we only need to consider the situation that $t_i \le t \le t_{i+M-1}$. With the definitions of RIA operations,

$$\begin{aligned} \frac{t-t_i}{t_{i+M-1}-t_i} &\in [A] \\ &= \frac{[t-t_i-\epsilon_1, t-t_i+\epsilon_1]}{[t_{i+M-1}-t_i-\epsilon_2, t_{i+M-1}-t_i+\epsilon_2]}, \end{aligned} \tag{2.19}$$

where $\epsilon_1$ and $\epsilon_2$ are extracted from $t-t_i$ and $t_{i-M+1}-t_i$. Since all the lower and upper bounds are non-negative by Theorem 3,

$$[A] \quad = \quad \left[\frac{t-t_i-\epsilon_1}{t_{i+M-1}-t_i+\epsilon_2} - \epsilon_3, \frac{t-t_i+\epsilon_1}{t_{i+M-1}-t_i-\epsilon_2} + \epsilon_4\right], \tag{2.20}$$

where $\epsilon_3$ and $\epsilon_4$ are associated with two quotients. The first quotient is no larger than 1, therefore $\epsilon_3 \le \epsilon = 2^{-52}$. In implementation, if

$$\frac{t-t_i+\epsilon_1}{t_{i+M-1}-t_i-\epsilon_2} > 1 \tag{2.21}$$

as $t$ is extremely close to $t_{i+m-1}$, we set

$$[A] = \left[\frac{t-t_i-\epsilon_1}{t_{i+M-1}-t_i+\epsilon_2} - \epsilon_3, 1\right] \tag{2.22}$$

because $[A] \subseteq [0,1]$. Therefore

$$[A] \subseteq \left[\frac{t-t_i-\epsilon_2}{t_{i+M-1}-t_i+\epsilon_2} - \epsilon, \frac{t-t_i+\epsilon_2}{t_{i+M-1}-t_i-\epsilon_2} + \epsilon\right] \equiv [A']. \tag{2.23}$$

Here $\epsilon_1$ is replaced by $\epsilon_2$ because $\epsilon_1 \le \epsilon_2$ by Theorem 4.

The width of the RIA evaluation of $(t-t_i)/(t_{i+M-1}-t_i)$

$$w([A]) \le w([A']) \quad = \quad \frac{t-t_i+\epsilon_2}{t_{i+M-1}-t_i-\epsilon_2} + \epsilon - \frac{t-t_i-\epsilon_2}{t_{i+M-1}-t_i+\epsilon_2} + \epsilon$$

$$= \quad 2\epsilon + \frac{2\epsilon_2(t - t_i + t_{i+M-1} - t_i)}{(t_{i+M-1} - t_i)^2 - \epsilon_2^2}. \tag{2.24}$$

For $t \in [t_i, t_{i+M-1}]$,

$$w([A]) \leq w([A']) \leq 2\epsilon + 2\epsilon_2 \frac{2(t_{i+M-1} - t_i)}{(t_{i+M-1} - t_i)^2 - \epsilon_2^2}. \tag{2.25}$$

If $t_{i+M-1} - t_i > \epsilon$, we use the common $\epsilon$ so that

$$w([A]) \leq w([A']) \leq 2\epsilon + 2\epsilon \frac{2(t_{i+M-1} - t_i)}{(t_{i+M-1} - t_i)^2 - \epsilon^2}. \tag{2.26}$$

Furthermore, if $t_{i+M-1} - t_i \gg \epsilon$ as in usual case, we can simplify the above equation as

$$\begin{aligned} w([A]) &< 2\epsilon + 2\epsilon \frac{2(t_{i+M-1} - t_i + \epsilon)}{(t_{i+M-1} - t_i)^2 - \epsilon^2} \\ &= 2\epsilon + \frac{4\epsilon}{t_{i+M-1} - t_i - \epsilon}, \end{aligned} \tag{2.27}$$

without significant increase of the bound. If we define

$$\Delta t_{M-1,i} = t_{i+M-1} - t_i, \tag{2.28}$$

then

$$w([A]) < 2\epsilon \left(1 + \frac{2}{\Delta t_{M-1,i} - \epsilon}\right). \tag{2.29}$$

Similarly,

$$w([B]) < 2\epsilon \left(1 + \frac{2}{\Delta t_{M-1,i+1} - \epsilon}\right), \tag{2.30}$$

where

$$\Delta t_{M-1,i+1} = t_{i+M} - t_{i+1}. \tag{2.31}$$

Let $W_M$ be an upper bound for $w([N]_{i,M}(t)), M \geq 2$. By using Theorem 2 in

Section 2.2.1,

$$\begin{aligned} w([N]_{i,M}(t)) &= w([A][N]_{i,M-1}(t)) + w([B][N]_{i+1,M-1}(t)) \\ &\leq |[A]|w([N]_{i,M-1}(t)) + |[N]_{i,M-1}(t)|w([A]) \\ &\quad +|[B]|w([N]_{i+1,M-1}(t)) + |[N]_{i+1,M-1}(t)|w([B]). \end{aligned} \tag{2.32}$$

Since the two quotients in equation (2.18) and the basis functions are all in $[0,1]$, in implementation $[A], [B]$ and the basis function values evaluated in RIA are all forced to be in $[0,1]$ even though the computed intervals may contain values slightly outside this range in some special cases. Therefore,

$$\begin{aligned} w([N]_{i,M}(t)) &\leq w([N]_{i,M-1}(t)) + w([N]_{i+1,M-1}(t)) + w([A]) + w([B]) \\ &< 2W_{M-1} + 4\epsilon\left(1 + \frac{1}{\Delta t_{M-1,i} - \epsilon} + \frac{1}{\Delta t_{M-1,i+1} - \epsilon}\right), \end{aligned} \tag{2.33}$$

by using the upper bounds of $w([A])$ and $w([B])$. Furthermore, if we define

$$\Delta t_{M-1} = \min_{0 \leq i \leq k-M} \{\Delta t_{M-1,i} | \Delta t_{M-1,i} > 0\}, \tag{2.34}$$

where $k$ is the number of knots, then

$$w([N]_{i,M}(t)) < W_M = 2W_{M-1} + 4\epsilon\left(1 + \frac{2}{\Delta t_{M-1} - \epsilon}\right). \tag{2.35}$$

Therefore,

$$\begin{aligned} W_M &= 2\left(2W_{M-2} + 4\epsilon\left(1 + \frac{2}{\Delta t_{M-2} - \epsilon}\right)\right) + 4\epsilon\left(1 + \frac{2}{\Delta t_{M-1} - \epsilon}\right) \\ &= \cdots \\ &= 4\epsilon\left(1 + \frac{2}{\Delta t_{M-1} - \epsilon}\right) + 2 \cdot 4\epsilon\left(1 + \frac{2}{\Delta t_{M-2} - \epsilon}\right) + \cdots \\ &\quad +2^{M-2} \cdot 4\epsilon\left(1 + \frac{2}{\Delta t_1 - \epsilon}\right) + 2^{M-1} \cdot W_1 \\ &= 4\epsilon\left(\sum_{i=1}^{M-1} 2^{i-1}\left(1 + \frac{2}{\Delta t_{M-i} - \epsilon}\right)\right) + 2^{M-1} \cdot W_1. \end{aligned} \tag{2.36}$$

Furthermore, with the initial conditions in the recursive formula (2.17),

$$W_1 = w([N]_{i,1}) = 0, \tag{2.37}$$

and thus,

$$W_M = 4\epsilon \left( \sum_{i=1}^{M-1} 2^{i-1} \left( 1 + \frac{2}{\Delta t_{M-i} - \epsilon} \right) \right). \tag{2.38}$$

For a cubic B-spline, $M = 4$, and

$$W_4 \quad = \quad 28\epsilon + 8\epsilon \left( \frac{1}{\Delta t_3 - \epsilon} + \frac{2}{\Delta t_2 - \epsilon} + \frac{4}{\Delta t_1 - \epsilon} \right), \tag{2.39}$$

where $\epsilon = 2^{-52} \approx 2.220e - 16$. If the knot vector is equidistant and $\Delta t_1 = 0.1$, then

$$W_4 \approx 1.009e - 13.$$

Numerical experiments show that $w([N]_{i,4}(t))$ is in order of $10^{-16} \sim 10^{-14}$ which is consistent with the upper bound $W_4$.

### 2.4.2 Width of Interval B-Spline Curve and Surface

The width of an interval vector $[\vec{V}] = \{[V]_x, [V]_y, [V]_z\}$ is defined as a vector $w([\vec{V}])$ such that

$$\begin{aligned} w([\vec{V}]) &= \{w_x, w_y, w_z\} \\ &= \{w([V]_x), w([V]_y), w([V]_z)\}, \end{aligned} \tag{2.40}$$

i.e. the width operator on an interval vector is taken componentwise.

The width of an interval B-spline curve (as well as an interval B-spline surface) at parameter $t$ reflects the inflation of the curve and therefore is used to evaluate the quality of fitting:

$$
\begin{aligned}
w([\vec{P}](t)) &= [w_x(t), w_y(t), w_z(t)] \\
&= w\left(\sum_{i=0}^{n-1}[\vec{D}]_i N_{i,M}(t)\right) \\
&= \sum_{i=0}^{n-1} w([\vec{D}]_i) N_{i,M}(t), \qquad (2.41)
\end{aligned}
$$

which is a polynomial in B-spline basis with vector coefficients and where $N_{i,M}(t)$ is considered exact. In other words, the width of an interval B-spline curve (and also a surface) is a weighted average of the widths of its control points, with the values of the basis functions as the weights.

For an interval B-spline curve with its basis functions evaluated in RIA,

$$
\begin{aligned}
w([\vec{P}](t)) &= w\left(\sum_{i=0}^{n-1}[\vec{D}]_i [N]_{i,M}(t)\right) \qquad (2.42) \\
&= \sum_{i=0}^{n-1} w([\vec{D}]_i [N]_{i,M}(t)).
\end{aligned}
$$

With $[\vec{D}]_i = [\vec{D}_i^l, \vec{D}_i^u]$, $[N]_{i,M}(t) = [N_{i,M}^l(t), N_{i,M}^u(t)]$, and considering that basis functions are non-negative,

$$
[\vec{D}]_i [N]_{i,M}(t) = \begin{cases} \left[\vec{D}_i^l N_{i,M}^l(t), \vec{D}_i^u N_{i,M}^u(t)\right] & if\ [\vec{D}]_i \geq \vec{0}, \\ \left[\vec{D}_i^l N_{i,M}^u(t), \vec{D}_i^u N_{i,M}^u(t)\right] & if\ \vec{0} \in [\vec{D}]_i, \\ \left[\vec{D}_i^l N_{i,M}^u(t), \vec{D}_i^l N_{i,M}^l(t)\right] & if\ [\vec{D}]_i \leq \vec{0}, \end{cases} \qquad (2.43)
$$

by the definitions of interval operations. Therefore,

$$
\begin{aligned}
w([\vec{D}]_i [N]_{i,M}(t)) &= \begin{cases} \vec{D}_i^u N_{i,M}^u(t) - \vec{D}_i^l N_{i,M}^l(t) \\ \vec{D}_i^u N_{i,M}^u(t) - \vec{D}_i^l N_{i,M}^u(t) \\ \vec{D}_i^l N_{i,M}^l(t) - \vec{D}_i^l N_{i,M}^u(t) \end{cases} \\
&= \begin{cases} w([\vec{D}]_i) N_{i,M}^l(t) + w([N]_{i,M}(t)) \vec{D}_i^u & if\ [\vec{D}]_i \geq \vec{0}, \\ w([\vec{D}]_i) N_{i,M}^u(t) & if\ \vec{0} \in [\vec{D}]_i, \\ w([\vec{D}]_i) N_{i,M}^l(t) - w([N]_{i,M}(t)) \vec{D}_i^l & if\ [\vec{D}]_i \leq \vec{0}. \end{cases} \qquad (2.44)
\end{aligned}
$$

As $w([N]_{i,M}(t)) \ll 1$, we conclude that

$$w([\vec{D}]_i[N]_{i,M}(t)) \approx w([\vec{D}]_i)N_{i,M}(t), \tag{2.45}$$

where $N_{i,M}(t) \in [N]_{i,M}(t)$. Therefore,

$$w([\vec{P}](t)) \approx \sum_{i=0}^{n-1} w([\vec{D}]_i)N_{i,M}(t), \tag{2.46}$$

and it is bounded as

$$\begin{aligned}&\max\{\sum_{i=0}^{n-1} w([\vec{D}]_i)|[N]_{i,M}(t)|, \sum_{i=0}^{n-1} |[\vec{D}]_i|w([N]_{i,M}(t))\}\\ &\le w([\vec{P}](t)) \le\\ &\sum_{i=0}^{n-1} w([\vec{D}]_i)|[N]_{i,M}(t)| + \sum_{i=0}^{n-1} |[\vec{D}]_i|w([N]_{i,M}(t)),\end{aligned} \tag{2.47}$$

by using Theorem 2 in Section 2.2.1. In general,

$$\text{each component of } w([\vec{D}]_i) \gg w([N]_{i,M}(t)), \tag{2.48}$$

and therefore,

$$\begin{aligned}&\max\{\sum_{i=0}^{n-1} w([\vec{D}]_i)|[N]_{i,M}(t)|, \sum_{i=0}^{n-1} |[\vec{D}]_i|w([N]_{i,M}(t))\}\\ =\ &\sum_{i=0}^{n-1} w([\vec{D}]_i)|[N]_{i,M}(t)|.\end{aligned} \tag{2.49}$$

We thus have

$$\begin{aligned}&\sum_{i=0}^{n-1} w([\vec{D}]_i)|[N]_{i,M}(t)| \le w([\vec{P}](t)) \le\\ &\sum_{i=0}^{n-1} w([\vec{D}]_i)|[N]_{i,M}(t)| + \sum_{i=0}^{n-1} |[\vec{D}]_i|w([N]_{i,M}(t)),\end{aligned} \tag{2.50}$$

### 2.4.3 Coordinate Translation

Interval B-spline curves are not invariant under translation and rotation, if coordinate transformation is performed in RIA. However, both translation and rotation are conservative, or in other words, in the new coordinate system, an *exactly transformed* interval curve is contained in the one defined by the transformed (using RIA) control points, since the RIA operations are conservative. As in the single-valued case, the transformation of interval B-spline curves and surfaces by coordinate translation is useful in computer aided design and solid modeling.

Another observation about the interval coordinate transformation is that the interval curve transformed using RIA has different shape from the interval curve defined by the transformed control points. To illustrate this point, we discuss the translation case.

Let $[\vec{P}](t)$ be an interval B-spline curve defined by equation (2.15) but now with the basis function replaced by interval expression $[N]_{i,M}(t)$. Assume the exact translation vector is $\vec{p}$, then in the new coordinate frame, the interval curve defined by the translated interval control points is

$$[\vec{P}_1](t) = \sum_{i=0}^{n-1}([\vec{D}]_i - \vec{p})[N]_{i,M}(t), \tag{2.51}$$

and the translated interval curve is

$$[\vec{P}_2](t) = \sum_{i=0}^{n-1}[\vec{D}]_i[N]_{i,M}(t) - \vec{p}. \tag{2.52}$$

For simplicity of notation below, we use $D, p$ instead of the vectors $\vec{D}, \vec{p}$, the simplified notation $[N]_i$ instead of $[N]_{i,M}(t)$, and the summation notation without indices. From the rules of interval operations,

$$\begin{aligned} [P_1] &= \sum([D]_i - p)[N]_i \qquad (2.53) \\ &= \sum[D_i^l - p, D_i^u - p][N_i^l, N_i^u], \\ [P_2] &= \sum[D]_i[N]_i - p\sum N_i \qquad (2.54) \end{aligned}$$

$$
\begin{aligned}
&= \sum\{[D]_i[N]_i - pN_i\} \\
&= \sum\{[D_i^l, D_i^u][N_i^l, N_i^u] - pN_i\},
\end{aligned}
$$

where $N_i \in [N]_i$ is the exact value of the $i$-th basis function. We now present two contradictory cases about their inclusion relation:

**Case 1:** If $D_i^l > p > 0$ for all $i$'s, then

$$
\begin{aligned}
[P_1] &= \sum[(D_i^l - p)N_i^l, (D_i^u - p)N_i^u], && (2.55) \\
[P_2] &= \sum[D_i^l N_i^l - pN_i, D_i^u N_i^u - pN_i]. && (2.56)
\end{aligned}
$$

With $N_i^l \le N_i \le N_i^u$, we have

$$
\begin{aligned}
D_i^l N_i^l - pN_i &\le D_i^l N_i^l - pN_i^l, && (2.57) \\
D_i^u N_i^u - pN_i &\ge D_i^u N_i^l - pN_i^u. && (2.58)
\end{aligned}
$$

It follows that $[P_1] \subseteq [P_2]$.

**Case 2:** If $p > 0 > D_i^u$ for all $i$'s, then

$$
\begin{aligned}
[P_1] &= \sum[(D_i^l - p)N_i^u, (D_i^u - p)N_i^l], && (2.59) \\
[P_2] &= \sum[D_i^l N_i^u - pN_i, D_i^u N_i^l - pN_i]. && (2.60)
\end{aligned}
$$

Comparing the bounds of $[P_1]$ and $[P_2]$, we can see that

$$
\begin{aligned}
D_i^l N_i^u - pN_i &\ge D_i^l N_i^u - pN_i^u, && (2.61) \\
D_i^u N_i^l - pN_i &\le D_i^u N_i^l - pN_i^l. && (2.62)
\end{aligned}
$$

This means that $[P_1] \supseteq [P_2]$.

Therefore, we conclude that the inclusion relation between the translated curve and the curve determined by the translated control points, depends on the interval control points and the single-valued translation vector.

In Appendix A, we propose a coordinate translation method which ensures that $[P_1] \supseteq [P_2]$ with only a very small enlargement of the translated control points.

Interval surface has similar properties under coordinate transformation using RIA.

## 2.4.4 Hodograph

The first derivative (hodograph) of an interval B-spline curve is useful in intersection problems and solid modeling. It is another interval B-spline curve given by

$$\begin{aligned} [\vec{P}]'(t) &= \sum_{i=1}^{n-1} [\vec{R}]_i N_{i,M-1}(t) \qquad (2.63) \\ \text{with} \qquad & [\vec{R}]_i = (M-1)\frac{[\vec{D}]_i - [\vec{D}]_{i-1}}{t_{i+M-1} - t_i}, \end{aligned}$$

which is of order $M-1$. The value of the hodograph curve at a parameter position is an interval number, which bounds the range within which the tangent vector at that position could vary. Its width

$$w([\vec{P}]'(t)) = \sum_{i=1}^{n-1} w([\vec{R}]_i) N_{i,M-1}(t), \qquad (2.64)$$

and

$$w([\vec{P}]'(t)) > w([\vec{P}](t)), \qquad (2.65)$$

since

$$w([\vec{R}]_i) = \frac{M-1}{t_{i+M-1} - t_i}(w([\vec{D}]_i) + w([\vec{D}]_{i-1})) > w([\vec{D}]_i) + w([\vec{D}]_{i-1}), \qquad (2.66)$$

if the knot vector is normalized such that $t_{i+M-1} - t_i < 1$.

For an interval B-spline curve with control points having uniform width $W$ and a knot vector with equally distributed internal knots (assuming $t_{i+1} - t_i = \Delta t$), since

$$\begin{cases} t_{i+M-1} - t_i < (M-1)\Delta t & if \;\; i < M \;\; or \;\; i > n - M + 1, \\ t_{i+M-1} - t_i = (M-1)\Delta t & otherwise, \end{cases} \tag{2.67}$$

from equation (2.64), we have

$$\begin{aligned} w([\vec{P}]'(t)) &\approx \sum_{i=1}^{n-1} w([\vec{R}]_i) N_{i,M-1}(t) \\ &> \sum_{i=1}^{n-1} \frac{M-1}{(M-1)\Delta t} w([\vec{D}]_i - [\vec{D}]_{i-1}) N_{i,M-1}(t) \\ &= \frac{1}{\Delta t} \sum_{i=1}^{n-1} \left( w([\vec{D}]_i) + w([\vec{D}]_{i-1}) \right) N_{i,M-1}(t) \\ &= \frac{1}{\Delta t} \sum_{i=1}^{n-1} 2W N_{i,M-1}(t) \\ &= \frac{2W}{\Delta t}. \end{aligned} \tag{2.68}$$

If the knot vector is normalized thus $\Delta t < 1$, the width of the hodograph curve is of the order of ten times larger than that of the curve. Considering such an interval curve of order 4 with 9 internal knots, the width of its hodograph curve could be 20 times larger than its own width. If the interval curve itself is not thin enough, its hodograph curve could blow up and become meaningless in predicting the tangent of a single-valued curve in the curve family defined by the interval curve.

In fact, in order to evaluate the range of the tangent vector of a planar interval B-spline curve, we need to study $[Y]'(t)/[X]'(t)$, where $[X]'(t)$ and $[Y]'(t)$ are the x, y components of $[\vec{P}]'(t)$, respectively. The range of unit tangent vector is the image of a rectangular range $[X]'(t) \times [Y]'(t)$ in $R \times R$ under the map

$$f \; : \; R \times R - \{0\} \rightarrow S^1, \tag{2.69}$$

$$\text{where } f = \frac{\{x, y\}}{\sqrt{x^2 + y^2}}, \quad \text{for } x \times y \in [X]'(t) \times [Y]'(t), \tag{2.70}$$

and $S^1$ is the unit circle and $R$ is the set of real numbers. The origin is not in the domain because none of the member curves are irregular, i.e. $\mathbf{0} \notin [X]'(t) \times [Y]'(t)$.

The range of unit tangent vector is connected and closed because $f$ is continuous in its domain and $R \times R$ is a *Hausdorff* space [29], and is contained in the unit circle. Figure 2-5(a) shows that the range is determined by the size and position of the box $[X]'(t) \times [Y]'(t)$. The size of the box is only determined by the sizes of the control boxes and has nothing to do with the normalization of the parameter domain which is simply a scaling transformation, see Figure 2-5(b). For a space interval curve or surface, the range of unit tangent vector or unit normal vector is a polygonal range on the unit sphere $S^2$.

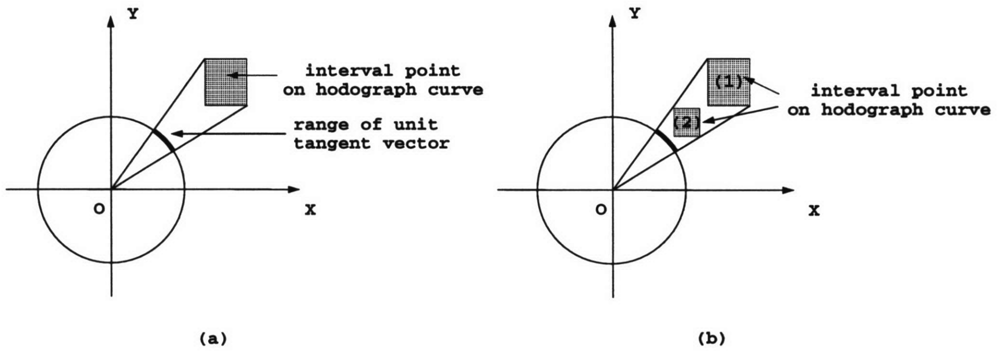

Y
interval point on hodograph curve
range of unit tangent vector
O
X
(a)
Y
(1)
(2)
interval point on hodograph curve
O
X
(b)

Figure 2-5: (a) Range of unit tangent vector of a planar interval B-spline curve at a particular parameter $t$; (b) Normalization of the parameter domain does not affect the range: (1) normalized parameter domain in $[0, 1]$; (2) non-normalized parameter domain.

### 2.4.5 Central Curve/Surface and Enveloping Curve/Surface

As mentioned before, an interval curve defines a family of curves whose control points are located inside the ranges defined by the interval control points. Namely, there are an infinite number of such curves. In this section, we are only interested in a few characteristic curves from this infinite set.

The central curve or surface of an interval B-spline curve or surface is important for NC machining and meshing applications. With the midpoint representation of an interval number, the central curve of an interval B-spline curve is defined as

$$\vec{\bar{P}}(t) = \sum_{i=0}^{n-1} \vec{\bar{D}}_i N_{i,M}(t), \tag{2.71}$$

where $N_{i,M}(t)$ is assumed to be exact. It is the single-valued curve defined by the same knot vector as the interval curve and a control polygon consisting of the centers of the interval control points. Then, the interval B-spline curve can be rewritten as

$$\begin{aligned}[\vec{P}](t) &= \sum_{i=0}^{n-1} \left[ \vec{D}_i - \frac{w([\vec{D}]_i)}{2}, \vec{D}_i + \frac{w([\vec{D}]_i)}{2} \right] N_{i,M}(t) \\ &= \sum_{i=0}^{n-1} \vec{D}_i N_{i,M}(t) + [-1,1] \sum_{i=0}^{n-1} \frac{w([\vec{D}]_i)}{2} N_{i,M}(t). \end{aligned} \tag{2.72}$$

If we consider the half-widths in the three directions of an interval point as the errors associated with its center, and define $\vec{\varepsilon}_i = w([\vec{D}]_i)/2$, the second term of equation (2.72) can be regarded as an error function accompanying the central curve. Hence, an interval curve can also be considered as a single-valued curve having errors determined by a piecewise polynomial function in the B-spline basis. The error function is

$$\vec{\varepsilon}(t) = \sum_{i=0}^{n-1} \vec{\varepsilon}_i N_{i,M}(t). \tag{2.73}$$

Similar definitions apply to an interval surface.

Another important curve, for the planar curve case, is the enveloping curve, which circumscribes an area that contains the family of curves defined above. For a 3D space interval curve or an interval surface, the enveloping entity is a closed surface. Enveloping curves and surfaces are useful in interval solid modeling, see Hu et al [21, 22].

Since an evaluated point on an interval planar curve is an affine map of the interval control points, its boundary corresponds to the boundaries of the rectangular ranges defined by the interval control points. If the interval curve is imagined as a geometric object generated by moving a rectangle in the plane, the boundary of the object is formed by sweeping the vertices and edges of the rectangle, see Figure 2-6. Therefore, the enveloping curve of an interval curve can only be composed of the curves defined by control points on the vertices or edges of the interval control points. It is no longer a single B-spline curve but a composite curve, and may lose $C^1, C^2$ continuities at some positions. For example, the enveloping curve of the interval B-spline curve in

Figure 2-6 consists of four curve segments and five straight line segments, see Figure 2-7, where *curve 1, 2, 3 and 4* are four single-valued B-spline curves corresponding to four vertices of an interval control point. At *joint point 1*, *curve 3* and *curve 4* intersect and a cusp is generated on the enveloping curve. *Curve 1* and *curve 2* are connected by a straight line segment at *joint point 2* and *3*.

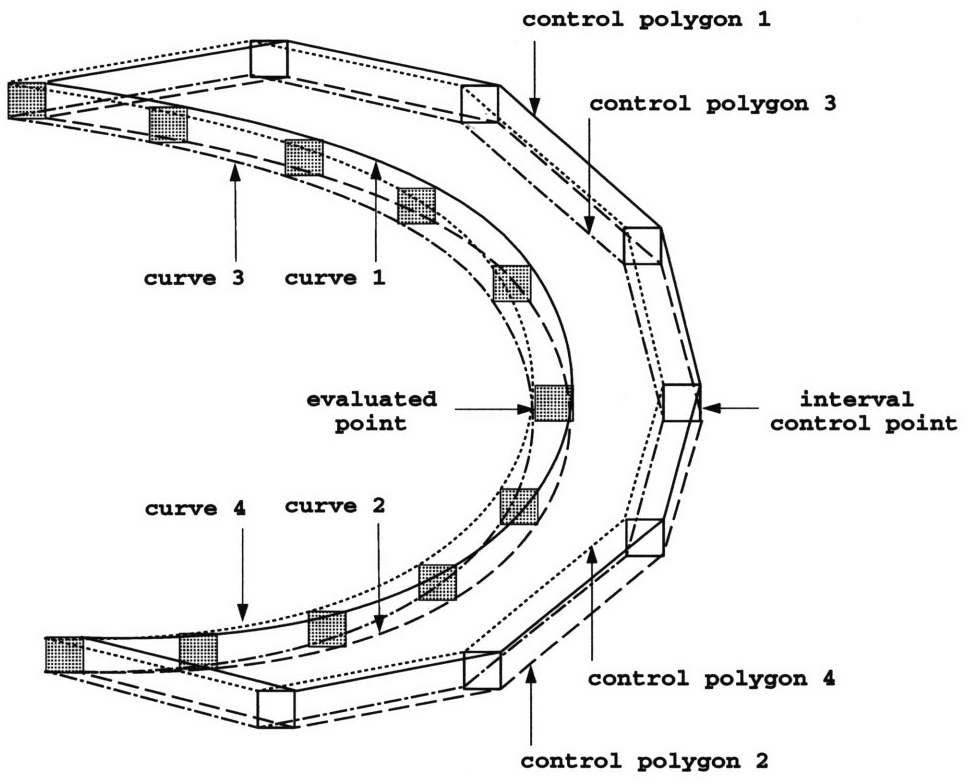


Figure 2-6: An interval planar curve

For an interval planar B-spline curve with uniform width (assuming $N_{i,M}(t)$ is exact),

$$\begin{aligned}[\vec{P}](t) &= \sum_{i=0}^{n-1}[\vec{D}]_i N_{i,M}(t) \qquad (2.74)\\ &= \sum_{i=0}^{n-1}\vec{D}_i N_{i,M}(t) + [-1,1]\sum_{i=0}^{n-1}\vec{\varepsilon}_i N_{i,M}(t)\end{aligned}$$

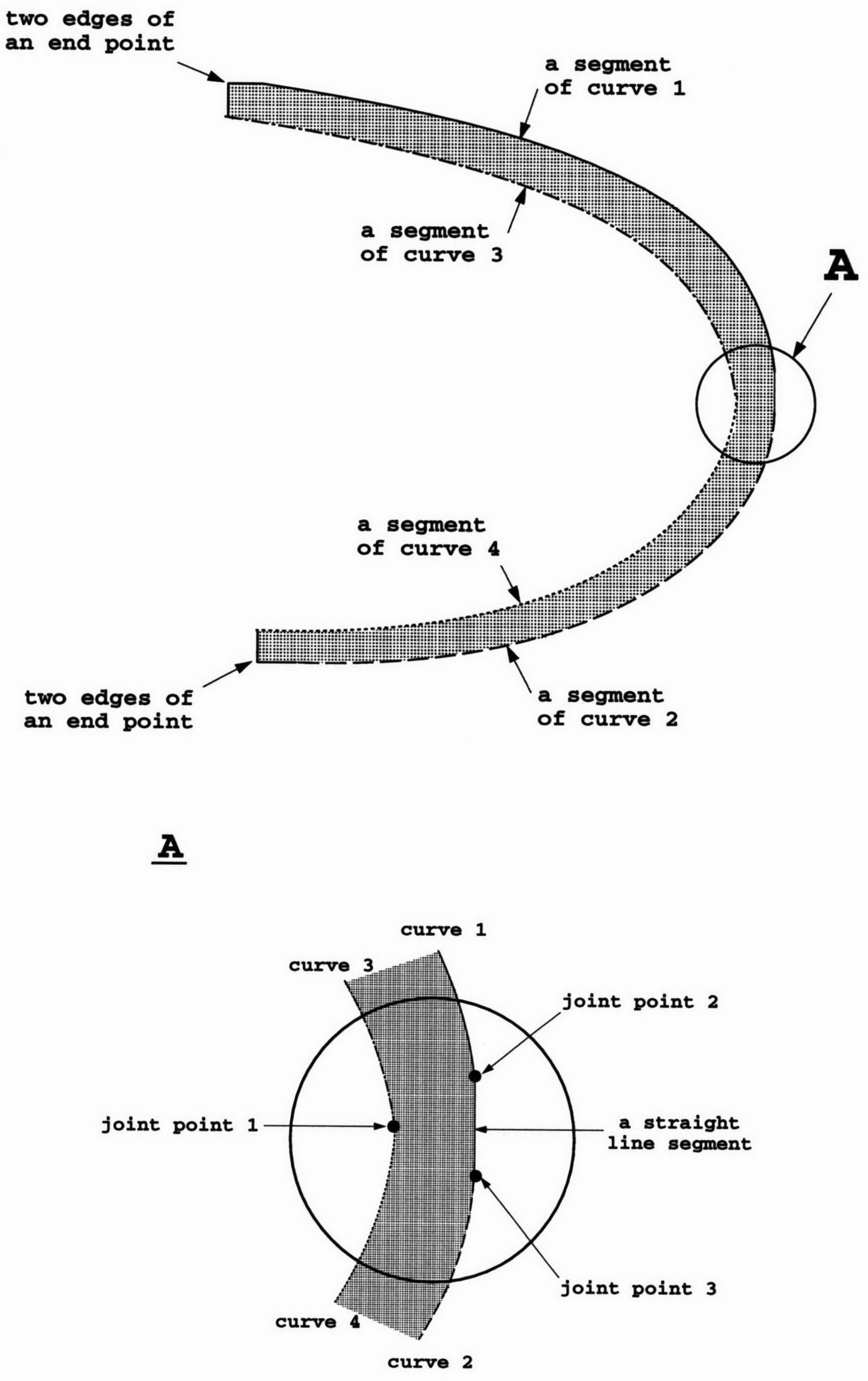


Figure 2-7: Enveloping curve of the interval planar curve in Figure 2-6

$$= \begin{pmatrix} X(t) \\ Y(t) \end{pmatrix} + [-1,1] \begin{pmatrix} \varepsilon_x(t) \\ \varepsilon_y(t) \end{pmatrix},$$

where $X(t), \varepsilon_x(t)$, and $Y(t), \varepsilon_x(t)$ refer to $x$, $y$ components, respectively, the enveloping curve can be determined by calculating the derivatives of the central curve, if the interval curve does not self-intersect.

**Regular points** $X'(t) \neq 0$ and $Y'(t) \neq 0$.

Pairs of diagonal vertices, called the *enveloping vertices* (marked with dots in Figure 2-8 and 2-9), of interval control points define the enveloping curve.

1. $X'(t)Y'(t) > 0$. See Figure 2-8.

$$\text{Curve 1} \ \ \vec{B}_1(t) = \sum_{i=0}^{n-1} \begin{pmatrix} D^l_{xi} \\ D^u_{yi} \end{pmatrix} N_{i,M}(t) \qquad (2.75)$$

$$\text{Curve 2} \ \ \vec{B}_2(t) = \sum_{i=0}^{n-1} \begin{pmatrix} D^u_{xi} \\ D^l_{yi} \end{pmatrix} N_{i,M}(t) \qquad (2.76)$$

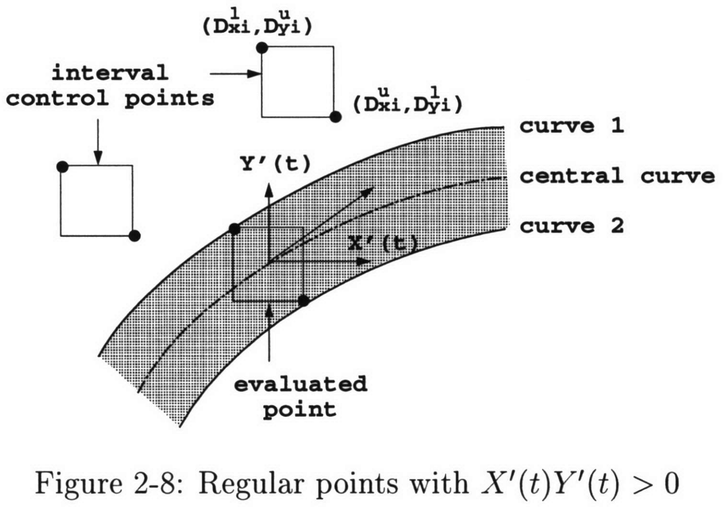


Figure 2-8: Regular points with $X'(t)Y'(t) > 0$

2. $X'(t)Y'(t) < 0$. See Figure 2-9.

$$\text{Curve 1} \quad \vec{B}_1(t) = \sum_{i=0}^{n-1} \begin{pmatrix} D^u_{xi} \\ D^u_{yi} \end{pmatrix} N_{i,M}(t) \tag{2.77}$$

$$\text{Curve 2} \quad \vec{B}_1(t) = \sum_{i=0}^{n-1} \begin{pmatrix} D^l_{xi} \\ D^l_{yi} \end{pmatrix} N_{i,M}(t) \tag{2.78}$$

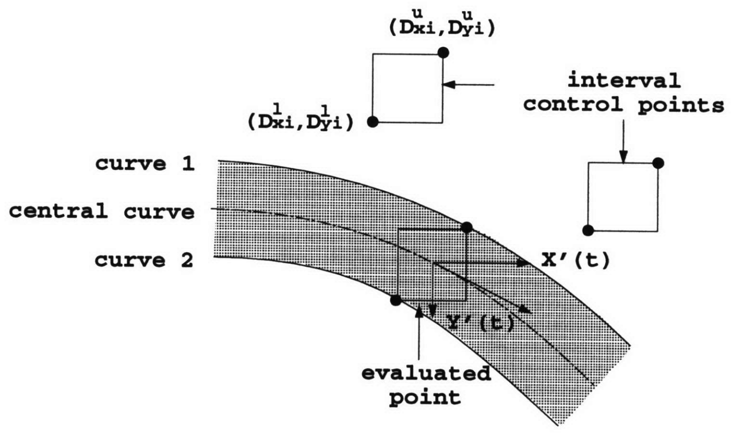


Figure 2-9: Regular points with $X'(t)Y'(t) < 0$

**Turning points** $X'(t) = 0$ or $Y'(t) = 0$.

The pair of enveloping vertices switches to another pair. One of the edges of the interval point at that position, is added to the enveloping curve on the convex side. A cusp is generated on the other side, by intersection of two single-valued curves. Figure 2-10 and 2-11 give the examples for $X'(t) \neq 0$ , $Y'(t) = 0$ and $X'(t) = 0$, $Y'(t) \neq 0$, respectively.

**Irregular points** $X'(t) = 0$ and $Y'(t) = 0$, which is a cusp on the central curve. Whether the enveloping vertices switch at an irregular point depends on whether $X'(t)Y'(t)$ changes sign across the point. One or two straight line segments may be added to one side of the enveloping curve, and a cusp is generated on the other side by intersection again. Figure 2-12 gives the examples of different cases.

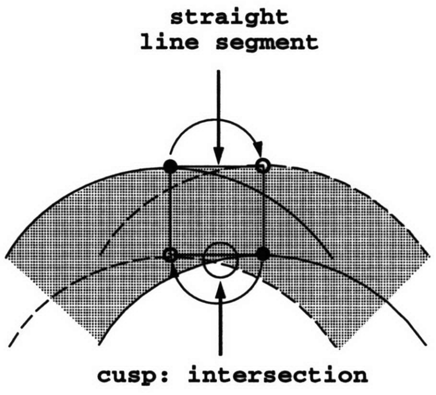


Figure 2-10: Turning point with $X'(t) \neq 0$ , $Y'(t) = 0$

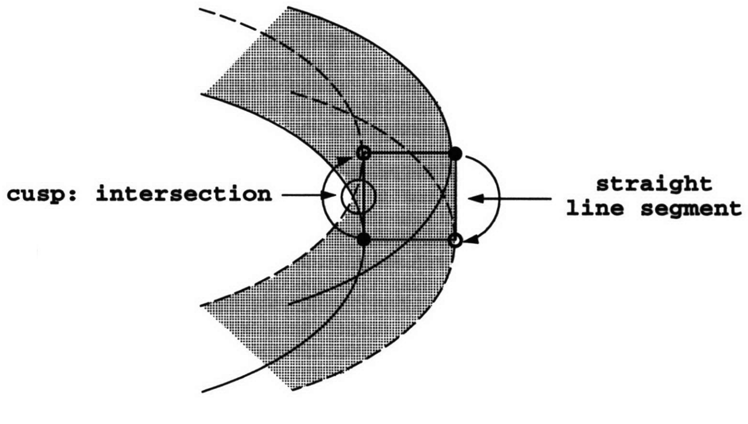
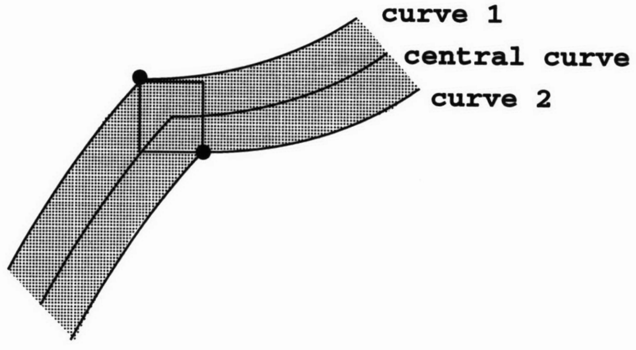
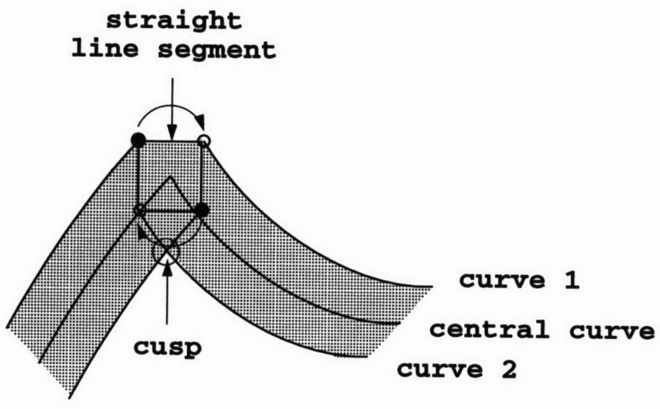


Figure 2-11: Turning point with $X'(t) = 0$, $Y'(t) \neq 0$



(a)



(b)

Figure 2-12: Irregular point: (a) $X'(t)Y'(t) > 0$ on both sides. No switch of the enveloping vertices. No straight line segment is added; (b) $X'(t)Y'(t)$ changes sign. The pair of enveloping vertices switches. A straight line segment is added.

**Endpoints** Two edges of an interval endpoint close the enveloping curve, see Figure 2-7.

The enveloping curve may be extracted by applying a subdivision algorithm to those switching points. Similar results can be obtained about interval space curve and interval surface, both with uniform width. For interval B-splines with varying width, it is more complex to extract the enveloping curves/surfaces.

### 2.4.6 Variation Diminishing Property

The variation diminishing property of a B-spline curve is important in geometric design applications. For an interval B-spline curve, neither the enveloping curve nor the central curve of an interval B-spline curve indicate how all the curves in the family potentially oscillate.

**Example 1:** Consider an interval cubic Bézier curve with 4 collinear (in interval sense) interval control points as in Figure 2-13(a). *Curve A*, which is a single-valued curve with control points inside the ranges, oscillates more than the central curve.

**Example 2:** Consider an interval cubic Bézier curve with 4 control points as in Figure 2-13(b). *Curve B* has 4 collinear control points which are still inside the ranges, and oscillates less than the central curve.

Obviously, for each curve in the family, there is one unique corresponding control polygon, for which the variation diminishing property still holds.

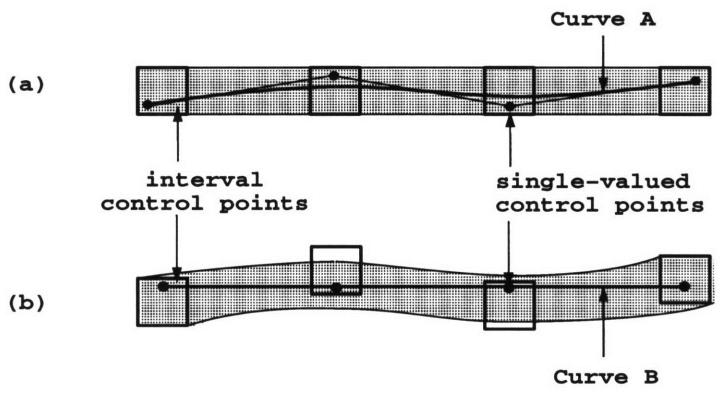


Figure 2-13: Oscillation of curves inside an interval curve

# Chapter 3

# Interval Curve Interpolation

## 3.1 Introduction

Data fitting can be approached either by interpolation or approximation. Interpolation is used when an exact fit is required. By saying interval data interpolation, we mean that an interval curve or surface precisely encloses interval data points with ranges equal to those of the interval data points at the interpolating positions, see Figure 3-1. As in the single-valued case, interval data interpolation involves the problem of solving a linear equation system. The difference is that now all the variables and constants are interval operands and all the operations are applied as in the definitions of the preceding chapter. Due to some unique properties of interval arithmetic, it is not as straightforward as in the single-valued case to reach high accuracy in interval data interpolation.

In this chapter, we will restrict our interest in the problem of interpolation with an interval cubic B-spline curve. We will propose a reliable, robust method to solve an interval linear system. The resulting interval curve nearly meets the high accuracy requirement with only a negligible enlargement introduced by RIA operations.

By the definition of an interval B-spline curve given before, the interpolation problem is stated as: given a set of interval data points $[\vec{P}]_j$, find a set of interval control points $[\vec{D}]_i$, which will make $[\vec{P}](u_j) = [\vec{P}]_j$ for $0 \leq i, j \leq n-1$, where $u_j$ is the single-valued parametric value associated with the $j$-th interval data point. This

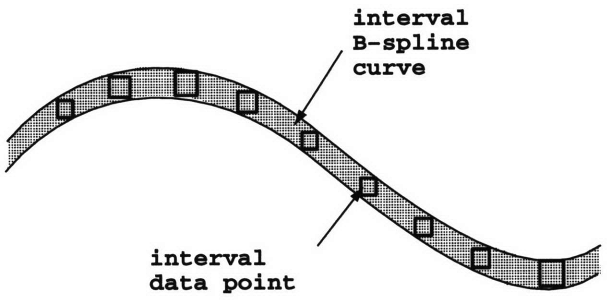


Figure 3-1: Interval data interpolation

turns out to be an $n \times n$ linear system with interval coefficients

$$[\vec{P}]_j = \sum_{i=0}^{n-1} [\vec{D}]_i [N]_{i,4}(u_j), \quad 0 \le j \le n-1 \tag{3.1}$$

or in matrix form

$$\{[\vec{P}]\} = [\mathbf{B}]\{[\vec{D}]\}, \tag{3.2}$$

where $[N]_{i,4}(u_j)$ is the RIA evaluation of a B-spline basis function in which the knot vector is made up of knots which are degenerate intervals (i.e. single floating point numbers). Evaluation of $N_{i,4}(t)$ using rational arithmetic drastically increases computation time and we therefore opt to use RIA which guarantees $[N]_{i,4}(u_j) \supset N_{i,4}(u_j)$. Therefore, $[\mathbf{B}]$ is an interval matrix, i.e. a matrix having intervals as its elements. $\{[\vec{P}]\} = \{[\vec{P}]_0, [\vec{P}]_1, \cdots, [\vec{P}]_{n-1}\}^T$ is the vector of interval data points. $\{[\vec{D}]\} = \{[\vec{D}]_0, [\vec{D}]_1, \cdots, [\vec{D}]_{n-1}\}^T$ is the vector of desired interval control points.

In the following sections, we first solve this interval linear system using the interval version of a general solver, and then explain why the solution is far away from our expectation of high accuracy; finally, we propose a method which uses *linear programming* (*LP*) technique to obtain a tighter solution.

## 3.2 Initial Interpolation

### 3.2.1 Parametrization and Knot Vector Construction

A parametric curve interpolation procedure starts with the parametrization of data points, i.e. the association of discrete data points with parametric values. Here we simply use the chord-length method, which approximates the arc length of a point on a curve with its chord length.

Given a set of discrete data points $\vec{P}_j, 0 \leq j \leq n-1$, let

$$\begin{aligned} \hat{u}_0 &= 0, \\ \hat{u}_{j+1} &= \hat{u}_j + d_{j+1},\ j = 0, 1, ..., n-2, \end{aligned} \tag{3.3}$$

where $d_{j+1} = |\vec{P}_{j+1} - \vec{P}_j|$ is the chord length between two consecutive points. The overall chord length is

$$d = \sum_{j=1}^{n-1} d_j. \tag{3.4}$$

The parametric value associated with point $\vec{P}_j$

$$u_j = \hat{u}_j / d, \tag{3.5}$$

which is normalized so that $u_j \in [0, 1]$ with $u_0 = 0$ and $u_{n-1} = 1$.

Since only approximate values are needed, interval data points $[\vec{P}]_j, 0 \leq j \leq n-1$, can be parametrized by using their midpoints $\bar{\vec{P}}_j$ in equation (3.3), and the calculations are simply performed in *floating point arithmetic* (FPA).

For the single-valued fitting problem, Schoenberg and Whitney [36] specified the condition which guarantees a non-singular coefficient matrix such that the problem has a solution. To ensure that system (3.2) has a solution, we make $0 \notin det([\mathbf{B}])$ which guarantees that the exact coefficient matrix $\mathbf{B}$ will never be singular since $\mathbf{B} \subset [\mathbf{B}]$ and $det(\mathbf{B}) \in det([\mathbf{B}])$. We realize this requirement by constructing the knot vector in a specific way as stated in, for example, [34]; the parametric values of all the data points, except the two adjacent to the ends, are used as knot values. The knot vector has a knot multiplicity of four at the ends to ensure that the resulting curve passes through the two endpoints. Thus, if the parametric values of the data points are $\{u_j\}, 0 \leq j \leq n-1$, the knot vector can be expressed as

$$\begin{aligned} T &= \underbrace{\{t_0, t_1, t_2, t_3, t_4, \cdots, t_{n-1}, t_n, t_{n+1}, t_{n+2}, t_{n+3}\}}_{n+4} \\ &= \underbrace{\{u_0, u_0, u_0, u_0, u_2, \cdots, u_{n-3}, u_{n-1}, u_{n-1}, u_{n-1}, u_{n-1}\}}_{n+4}, \end{aligned} \tag{3.6}$$

where $n$ is the number of control points, which is equal to the number of data points.

### 3.2.2 Solution of an Interval Linear System

With the parametric values and the knot vector in the preceding section, the interval coefficient matrix is obtained evaluating the B-spline basis function in RIA. The linear system, thus, can be solved using various solvers in interval arithmetic.

The fact that the knot vector is a subset of the parametric values at the interpolating positions not only ensures a non-singular coefficient matrix but also provides a highly banded one. For a cubic B-spline curve evaluated at the $i$-th knot, only the values of the $(i-1)$-th, $i$-th, and $(i+1)$-th basis are nonzero. With $u_1 \in [t_3, t_4], u_{n-2} \in [t_{n-1}, t_n]$, $[\mathbf{B}]$ takes the form as

$$\begin{pmatrix} 1 & 0 & 0 & 0 & & & & \\ b_1 & a_1 & c_1 & x & & & & \\ & b_2 & a_2 & c_2 & & & & \\ & & b_3 & a_3 & c_3 & & & \\ & & & \cdots & & & & \\ & & b_{n-4} & a_{n-4} & c_{n-4} & & & \\ & & & b_{n-3} & a_{n-3} & c_{n-3} & & \\ & & & y & b_{n-2} & a_{n-2} & c_{n-2} \\ & & & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.7}$$

where

$$\begin{aligned} a_i &= [N]_{i,4}(u_i), b_i = [N]_{i-1,4}(u_i), c_i = [N]_{i+1,4}(u_i), \\ x &= [N]_{3,4}(u_2), y = [N]_{n-4,4}(u_{n-2}), \end{aligned}$$

and they are all interval numbers though we simplified the notation for conciseness.

A specialized linear solver can be developed to minimize computation time and storage requirement.

After imposing the position boundary conditions $[\vec{D}]_0 = [\vec{P}]_0$ and $[\vec{D}]_{n-1} = [\vec{P}]_{n-1}$, the linear system is then reduced to an $(n-2) \times (n-2)$ system described by equations

$$[\mathbf{B}'] \left\{ \begin{array}{l} [\vec{D}]_1 \\ [\vec{D}]_2 \\ \ldots \\ [\vec{D}]_{n-2} \end{array} \right\} = \left\{ \begin{array}{l} [\vec{P}]_1 - [\vec{P}]_0[N]_{0,4}(u_1) \\ [\vec{P}]_2 \\ \ldots \\ [\vec{P}]_{n-2} - [\vec{P}]_{n-1}[N]_{n-1,4}(u_{n-2}) \end{array} \right\}, \tag{3.8}$$

where $[\mathbf{B}']$ is a submatrix of $[\mathbf{B}]$, consisting of the elements from the second to the ($n$-*1*)-th row and the second to the *(n-1)*-th column. The new coefficient matrix $[\mathbf{B}']$ is a tridiagonal matrix with two additional off-diagonal elements $x$ and $y$, and can be easily LU-decomposed [5] as

$$[\mathbf{L}][\mathbf{U}] = \begin{pmatrix} 1 & & & & & \\ \beta_2 & 1 & & & & \\ & \beta_3 & 1 & & & \\ & & \ldots & & & \\ & & \beta_{n-3} & 1 & & \\ & & \delta & \beta_{n-2} & 1 \end{pmatrix} \begin{pmatrix} \alpha_1 & c_1 & \gamma & & & \\ & \alpha_2 & c_2 & & & \\ & & \alpha_3 & c_3 & & \\ & & & \ldots & & \\ & & & \alpha_{n-3} & c_{n-3} \\ & & & & \alpha_{n-2} \end{pmatrix} \tag{3.9}$$

where all $\alpha's, \beta's, \gamma$, and $\delta$ are intervals, and again the notations have been simplified.

By equating (3.9) to the matrix $[\mathbf{B}']$, the interval elements in matrices $[\mathbf{L}]$ and $[\mathbf{U}]$

can be determined by the following recurrence relations

$$\begin{aligned}
\alpha_1 &= a_1, \\
\alpha_k &= a_k - \beta_k c_{k-1}, \quad 2 \le k \le n-2, \\
\beta_k &= \frac{b_k}{\alpha_{k-1}}, \quad 2 \le k \le n-3, \\
\beta_{n-2} &= \frac{b_{n-2} - c_{n-4}\delta}{\alpha_{n-3}}, \\
\gamma &= x, \\
\delta &= \frac{y}{\alpha_{n-4}}.
\end{aligned} \tag{3.10}$$

Note here that $c_2$ is actually replaced by $c_2 - \gamma\beta_2$ due to the existence of the off-diagonal element $x$.

Now the linear system is

$$[\mathbf{L}][\mathbf{U}]\{[\vec{D}']\} = \{[\vec{P}']\}, \tag{3.11}$$

where

$$\{[\vec{D}']\} = \{[\vec{D}]_1, \cdots, [\vec{D}]_{n-2}\}, \tag{3.12}$$

$$\{[\vec{P}']\} = \{[\vec{P}]_1, \cdots, [\vec{P}]_{n-2}\}. \tag{3.13}$$

If we set

$$\{[\vec{g}]\} = [\mathbf{U}]\{[\vec{D}']\} \tag{3.14}$$

with

$$\{[\vec{g}]\} = \{[\vec{g}]_1, [\vec{g}]_2, \cdots, [\vec{g}]_{n-2}\}^T, \tag{3.15}$$

then we can solve

$$[\mathbf{L}]\{[\vec{g}]\} = \{[\vec{P}']\} \tag{3.16}$$

for $\{[\vec{g}]\}$ by forward substitution and solve

$$[\mathbf{U}]\{[\vec{D}']\} = \{[\vec{g}]\} \tag{3.17}$$

for $\{[\vec{D}']\}$ by backward substitution, according to the following recursive formulae

$$\begin{aligned} [\vec{g}]_1 &= [\vec{P}]_1, \\ [\vec{g}]_k &= [\vec{P}]_k - \beta_k[\vec{g}]_{k-1}, \;\; 2 \le k \le n-3, \\ [\vec{g}]_{n-2} &= [\vec{P}]_{n-2} - \delta[\vec{g}]_{n-4} - \beta_{n-2}[\vec{g}]_{n-3}, \end{aligned} \tag{3.18}$$

and

$$\begin{aligned} [\vec{D}]_{n-2} &= \frac{[\vec{g}]_{n-2}}{\alpha_{n-2}}, \\ [\vec{D}]_k &= \frac{[\vec{g}]_k - c_k[\vec{D}]_{k+1}}{\alpha_k}, \;\; n-3 \ge k \ge 2, \\ [\vec{D}]_1 &= \frac{[\vec{g}]_1 - c_1[\vec{D}]_2 - \gamma[\vec{D}]_3}{\alpha_1}. \end{aligned} \tag{3.19}$$

## 3.3 Minimization Process

### 3.3.1 Necessity of Minimization Process

The method proposed in the last section is only a direct extension to interval analysis of a general solver of a linear system with highly banded matrix. The enclosure of data points is ensured because all the computations are performed in RIA. But the high accuracy required by the interpolation procedure is not satisfactory. We tested the accuracy simply by plugging the solution back to the equations, and we found that they did not recover the equalities. Indeed, such an interval curve could have a range several times wider than those of the interval data points. This kind of inflation is not unusual in interval analysis. We will state some of the reasons in the following paragraphs.

Consider a continuous real function $f$. An expression $f(x)$ is a procedure that determines a value of the function $f$ for every argument $x$. The expression

$$\begin{aligned}
& F(f,[X];[A]_0,\cdots,[A]_m) \\
= \ & \{f(x;a_0,\cdots,a_m) | x \in [X], a_k \in [A]_k, 0 \le k \le m\} \\
= \ & [\min_{\substack{x\in[X]\\ a_k\in[A]_k, 0\le k\le m}} f(x;a_0,\cdots,a_m), \max_{\substack{x\in[X]\\ a_k\in[A]_k, 0\le k\le m}} f(x;a_0,\cdots,a_m)] \qquad (3.20)
\end{aligned}$$

denotes the real range of the function $f$ when the argument $x \in [X]$ and the constants $a_k \in [A]_k, 0 \le k \le m$. Interval evaluation is obtained by replacing all the operands with intervals and all the operations with interval operations defined before. The relation between the real range $F(f,[X];[A]_0,\cdots,[A]_m)$ and the interval evaluation $f([X];[A]_0,\cdots,[A]_m)$ is stated in the following theorem [2].

**Theorem:** Let $f$ be a continuous function of the real variables $x_1,\cdots,x_n$, and let $f(x_1,\cdots,x_n;a_0,\cdots,a_m)$ be an expression for $f$. Furthermore, assume that the interval evaluation $f([X]_1,\cdots,[X]_n;[A]_0,\cdots,[A]_m)$ is defined for $[X]_1,\cdots,[X]_n$,$[A]_0,\cdots$, $[A]_m$. It then follows that

$$F(f,[X]_1,\cdots,[X]_n;[A]_0,\cdots,[A]_m) \subseteq f([X]_1,\cdots,[X]_n;[A]_0,\cdots,[A]_m) \qquad (3.21)$$

This is called the *inclusion property* of interval evaluation. The ultimate reason of such inflations is that each occurrence of an independent variable actually becomes an independent variable when replaced with an interval representation, i.e. the occurrences of a variable are no longer dependent as they should be. In the recursive formula (3.19), the solutions are obtained by evaluating a set of interval expressions. This is one of the reasons for the observed inflation.

We can see another reason for the observed inflation from the solution of the equation

$$[A][X] = [B], \qquad (3.22)$$

where $[A], [B]$ and $[X]$ are interval *numbers*, and $[A] \neq [0, 0]$. An auxiliary function $\chi$ is defined as

$$\chi[A] = \begin{cases} A^l/A^u & \text{if } |A^l| \leq |A^u|, \\ A^u/A^l & \text{otherwise.} \end{cases} \tag{3.23}$$

It has been proved that the equation $[A][X] = [B]$ is satisfied by an $[X] \in I(\Re)$ if and only if $\chi[A] \geq \chi[B]$, see [2]. This means there are cases where no exact interval solution exists. In general, if there does exist such a solution $[X] \in I(\Re)$, then $[X] \subseteq [B]/[A]$.

For a more complex equation system as we have here, the recursive formula (3.19) only provides a set of solutions which ensure $\{[\vec{P}]\} \subseteq [\mathbf{B}]\{[\vec{D}]\}$. The resulting interval curve thus has a width wider than those of the data points. As a tight interval interpolating curve is desired, we then need to seek a minimization process to obtain much tighter solutions, if they exist.

### 3.3.2 Bounds of an Evaluated Point on an Interval B-Spline Curve

Before we formulate the minimization process, we first study the bounds of an evaluated point on an interval cubic B-spline curve.

Let $[\vec{P}](u_j)$ be a point evaluated at $u_j$ on an interval curve $[\vec{P}](t)$, and

$$[\vec{P}](u_j) = [\vec{P}^l(u_j), \vec{P}^u(u_j)] = \sum_{i=0}^{n-1} [\vec{D}]_i [N]_{i,4}(u_j). \tag{3.24}$$

As $[N]_{i,4}(u_j) \geq 0$, the following relations hold by the operation rule of interval product

$$[\vec{D}]_i [N]_i \quad = \quad [\vec{D}_i^l, \vec{D}_i^u] \cdot [N_i^l, N_i^u]$$

$$= \begin{cases} [\vec{D}_i^l N_i^l, \vec{D}_i^u N_i^u] \supseteq [\vec{D}_i^l N_i^u, \vec{D}_i^u N_i^u] & if\ [\vec{D}]_i > \vec{0}, \\ [\vec{D}_i^l N_i^u, \vec{D}_i^u N_i^u] & if\ \vec{0} \in [\vec{D}]_i, \\ [\vec{D}_i^l N_i^u, \vec{D}_i^u N_i^l] \supseteq [\vec{D}_i^l N_i^u, \vec{D}_i^u N_i^u] & if\ [\vec{D}]_i < \vec{0}, \end{cases} \tag{3.25}$$

where simplified notations have been used for a concise description, and all the vector comparisons, here as well as elsewhere in the thesis, should be interpreted for each component. Consequently,

$$[\vec{D}]_i [N]_{i,4}(u_j) \supseteq [\vec{D}_i^l N_{i,4}^u(u_j), \vec{D}_i^u N_{i,4}^u(u_j)]. \tag{3.26}$$

Since interval addition is conservative, we have

$$\begin{aligned} \sum_{i=0}^{n-1} [\vec{D}]_i [N]_{i,4}(u_j) &\supseteq \sum_{i=0}^{n-1} [\vec{D}_i^l N_{i,4}^u(u_j), \vec{D}_i^u N_{i,4}^u(u_j)] \\ &= \left[ \sum_{i=0}^{n-1} \vec{D}_i^l N_{i,4}^u(u_j), \sum_{i=0}^{n-1} \vec{D}_i^u N_{i,4}^u(u_j) \right] \equiv [\vec{P}'](u_j), \end{aligned} \tag{3.27}$$

where the equality can be deduced using the definition of interval addition. Thus, $[\vec{P}'](u_j)$ defines such an interval point that

$$[\vec{P}](u_j) \supseteq [\vec{P}'](u_j) \tag{3.28}$$

for an arbitrary parameter value $u_j$.

From equation (3.25), we obtain that

$$w([\vec{D}]_i [N]_{i,4}(u_j)) - w([\vec{D}_i^l N_{i,4}^u(u_j), \vec{D}_i^u N_{i,4}^u(u_j)])$$

$$= \begin{cases} (\vec{D}_i^u N_i^u - \vec{D}_i^l N_i^l) - (\vec{D}_i^u N_i^u - \vec{D}_i^l N_i^u) \\ (\vec{D}_i^u N_i^u - \vec{D}_i^l N_i^u) - (\vec{D}_i^u N_i^u - \vec{D}_i^l N_i^u) \\ (\vec{D}_i^u N_i^l - \vec{D}_i^l N_i^u) - (\vec{D}_i^u N_i^u - \vec{D}_i^l N_i^u) \end{cases}$$

$$
= \begin{cases} D^l_i N^u_i(u_j) - D^l_i N^l_i(u_j) = |\vec{D}^l_i| w([N]_{i,4}(u_j)) & if\ [\vec{D}]_i > \vec{0} \\ \vec{0} & if\ \vec{0} \in [\vec{D}]_i \\ D^u_i N^l_i(u_j) - D^u_i N^u_i(u_j) = |\vec{D}^u_i| w([N]_{i,4}(u_j)) & if\ [\vec{D}]_i < \vec{0} \end{cases}
$$

$$
\leq \ |[\vec{D}]_i| w([N]_{i,4}(u_j), \tag{3.29}
$$

where the absolute value of a vector is again interpreted for each component and not interpreted as the length of the vector. It follows from Theorem 2 in Section 2.2.1 that

$$
\begin{aligned}
& w([\vec{P}](u_j)) - w([\vec{P}'](u_j)) \\
= \ & w(\sum_{i=0}^{n-1}[\vec{D}]_i[N]_{i,4}(u_j)) - w(\sum_{i=0}^{n-1}[\vec{D}^l_i N^u_{i,4}(u_j), \vec{D}^u_i N^u_{i,4}(u_j)]) \\
= \ & \sum_{i=0}^{n-1} w([\vec{D}]_i[N]_{i,4}(u_j)) - \sum_{i=0}^{n-1} w([\vec{D}^l_i N^u_{i,4}(u_j), \vec{D}^u_i N^u_{i,4}(u_j)]) \\
= \ & \sum_{i=0}^{n-1} \left( w([\vec{D}]_i[N]_{i,4}(u_j)) - w([\vec{D}^l_i N^u_{i,4}(u_j), \vec{D}^u_i N^u_{i,4}(u_j)]) \right).
\end{aligned} \tag{3.30}
$$

Using equation (3.27) and (3.29), we conclude that

$$
\vec{0} \leq w([\vec{P}](u_j)) - w([\vec{P}'](u_j)) \leq \sum_{i=0}^{n-1} |[\vec{D}]_i| w([N]_{i,4}(u_j)). \tag{3.31}
$$

Since $w([N]_{i,4}(u_j))$ is extremely small (of the order of $10^{-15} \sim 10^{-14}$), the inclusion requirement can be satisfied by setting $[\vec{P}'](u_j) \supseteq [\vec{P}]_j$ with only a very small change in width, see Figure 3-2.

### 3.3.3 Formulation of the LP Process

The objective of the minimization process is to reduce the widths of the interval curve at the interpolating locations, subject to the condition that the interval curve still forms an enclosure of all the interval data points. Mathematically, the minimization

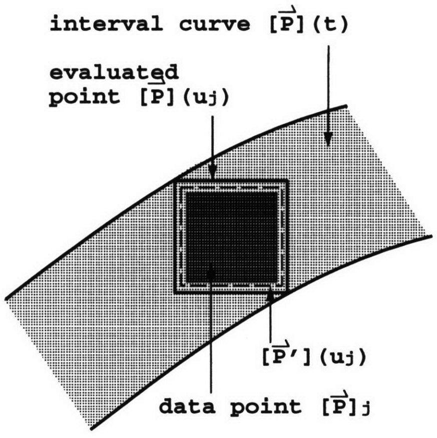

interval curve $[\vec{P}](t)$
evaluated point $[\vec{P}](u_j)$
$[\vec{P}'](u_j)$
data point $[\vec{P}]_j$

Figure 3-2: The bounds of an evaluated point on an interval B-spline curve

problem is stated as

$$\begin{aligned} \text{minimize} \quad & \vec{F}_1 = \sum_{j=0}^{n-1} w([\vec{P}](u_j)), && (3.32) \\ \text{with constraints} \quad & [\vec{P}]_j \subseteq [\vec{P}](u_j), && (3.33) \\ & w([\vec{P}](u_j)) \leq \vec{\omega}, \\ & 0 \leq j \leq n-1, \end{aligned}$$

where $\vec{\omega}$ is a vector whose components are the maximum allowable widths of the interval evaluated points in $x$, $y$, $z$ directions, and is selected by the user.

As the width of a point on an interval B-spline curve is mainly attributed to the widths of the interval control points, with only a small contribution from the numerical error induced by RIA evaluation, it will be minimum if the widths of all the interval control points reach their permissible minima. Thus, the objective of the minimization process can be equivalently set as to minimize the widths of the desired interval control points. With the inclusion relation $[\vec{P}](u_j) \supseteq [\vec{P}'](u_j)$ in Section 3.3.2, if we set

$$[\vec{P}'](u_j) \supseteq [\vec{P}]_j, \qquad (3.34)$$

then

$$[\vec{P}](u_j) \supseteq [\vec{P}'](u_j) \supseteq [\vec{P}]_j, \tag{3.35}$$

which means constraint (3.33) is automatically satisfied. We thus can reformulate the minimization problem in terms of a new objective function

$$\vec{F}_2 = \sum_{i=0}^{n-1} w([\vec{D}]_i), \tag{3.36}$$

with constraints

$$\begin{aligned} [\vec{P}'](u_j) &\supseteq [\vec{P}]_j, \\ w([\vec{D}]_i) &\leq \vec{\omega}, \\ 0 \leq i, j &\leq n-1. \end{aligned} \tag{3.37}$$

The minimization problem with objective function (3.36) and constraints (3.37), can be reformulated as a linear programming problem by introducing $6n$ auxiliary variables, which represent the lower and upper bounds of the $n$ interval *3D* control points [23]. In such a way, both the objective function and the constraints are transformed into linear equations.

For the $x$ component of the interval *3D* control points (similarly for $y$ and $z$ components), set $[D_x]_i = [x_i, x_{i+n}], 0 \leq i \leq n-1$. Now we wish to minimize

$$F_3 = \sum_{i=0}^{n-1} (x_{i+n} - x_i) \tag{3.38}$$

with constraints

$$\begin{aligned} -\infty &< \sum_{i=0}^{n-1} x_i N_{i,4}^u(u_j) \leq P_{x,j}^l, \quad 0 \leq j \leq n-1, \\ P_{x,j}^u &\leq \sum_{i=n}^{2n-1} x_i N_{i-n,4}^u(u_j) < +\infty, \quad 0 \leq j \leq n-1, \end{aligned} \tag{3.39}$$

$$0 \ \le \ x_{i+n} - x_i \le \omega_x, \ \ 0 \le i \le n-1,$$

or, in matrix form

$$\{b_l\} \le \mathbf{A}\{x\} \le \{b_u\}, \tag{3.40}$$

where $\mathbf{A}_{3n\times 2n}$ is the single-valued constraint matrix, $\{b_l\}$ and $\{b_u\}$ are the single-valued bounding vectors, and

$$\mathbf{A} = \begin{pmatrix} \mathbf{B}_1 & 0 \\ 0 & \mathbf{B}_1 \\ \mathbf{I} & -\mathbf{I} \end{pmatrix}, \tag{3.41}$$

with $\mathbf{B}_1 = (N^u_{i,4}(u_j)\,)_{n\times n}$, $0 \le i,j \le n-1$, and $\mathbf{I}$ is the $n \times n$ unit matrix, and

$$\begin{aligned} \{b_l\} &= \{\underbrace{-\infty,\cdots,-\infty}_{n},\underbrace{P^u_{x,0},\cdots,P^u_{x,n-1}}_{n},\underbrace{0,\cdots,0}_{n}\}^T, \\ \{b_u\} &= \{\underbrace{P^l_{x,0},\cdots,P^l_{x,n-1}}_{n},\underbrace{\infty,\cdots,\infty}_{n},\underbrace{\omega_x,\cdots,\omega_x}_{n}\}^T. \end{aligned} \tag{3.42}$$

The number of iterations performed in the LP process depends upon factors such as the number of control points and data points and the quality of the initial set of selected control points. The number of data points is fixed with the given problem, therefore, one way to reduce processing time is to give good initial values. We have found that the set of interval control points obtained from the recursive formulae (3.19) leads to quick convergence and therefore can be used effectively.

### 3.3.4 Summary

In summary, the procedure for interpolating interval data points with an interval cubic B-spline curve consists of the following steps:

1. Parametrize the data points by the chord-length method, using only the centers of the interval data points to approximate the chord lengths.

2. Construct the knot vector by using the parametric values of the data points (except the two just adjacent to the ends) as knots.

3. Solve the interval linear system using the recursive formulae (3.19) to get a set of loose interval control points.

4. Use the result of Step 3 as initial values to solve the LP problem for the interval control points.

# Chapter 4

# Interval Surface Approximation

## 4.1 Problem Description and the Approach

The robotic measurement system used in this project consists of a 4D Imager – a 3D laser range scanner, and a driving device which provides 5 degrees of freedom, see Figure 4-1. The entire system is controlled by a computer. Range data are collected and ordered in the form of stripes. The range points in each stripe are grouped together and separated from points in the adjacent and discontinuous stripes [11]. Figure 4-2 shows an example of the range points collected by this system from the surface of an object.

The objective of the surface reconstruction method developed in this thesis is to create an interval B-spline surface which envelopes measured points. Because each point possesses uncertainty, it is represented as an interval data point defining a rectangular range which must contain the exact position of the point.

Since interval data points are grouped in stripes, a simple and efficient way to create an interval fitting surface is to loft a family of interval B-spline curves which either interpolate or approximate the interval data points in all stripes, see [43]. Figure 4-3 gives the flow chart of the interval surface reconstruction procedure.

In the following sections, we develop a method for interval curve approximation first, then describe the details about interval surface lofting, and finally summarize the entire procedure for interval surface fitting.

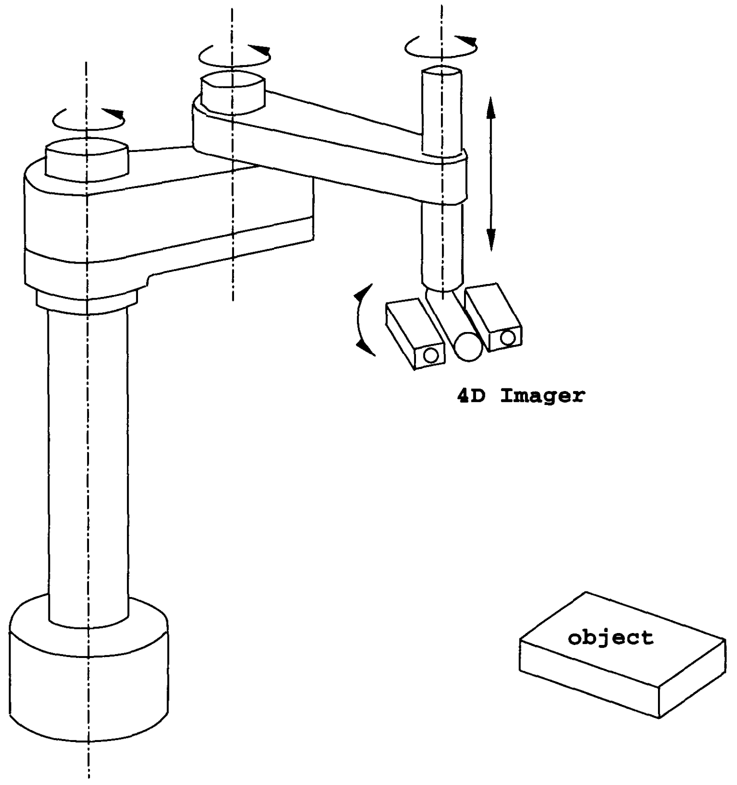


Figure 4-1: Robotic measurement system

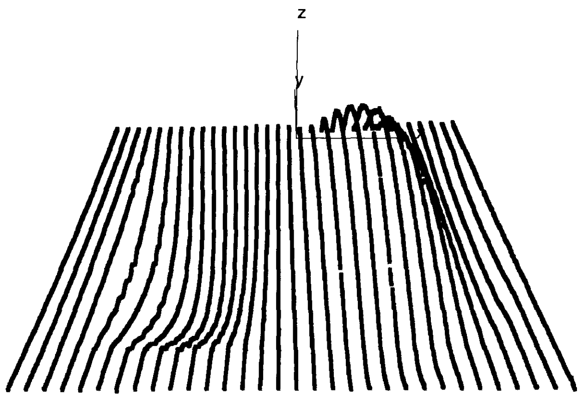


Figure 4-2: An example of measured data

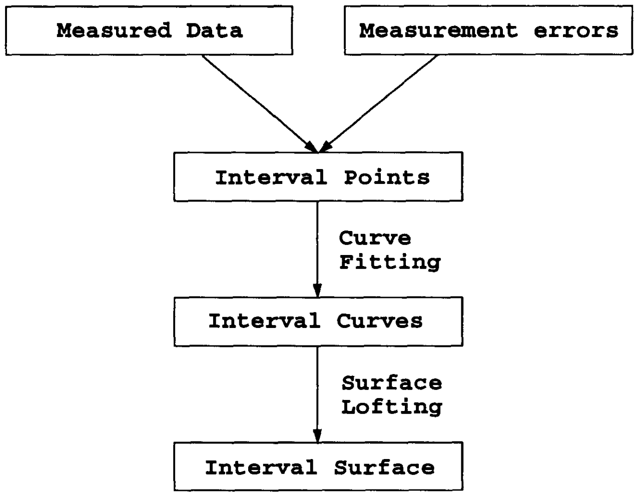


Figure 4-3: Flow chart of surface reconstruction

## 4.2 Interval Curve Approximation

In interval data approximation, both the approximation error and the data error are to be incorporated into the residual error associated with the resulting interval curve, which encloses the interval data points but with generally larger ranges than those of the data points. Directly following the interpolation problem, the approximation problem can be approached again as a minimization problem using the LP technique, but now with a reduced set of interval control points, which also means that the knot vector has to be constructed in a different way. The setup for the LP process is exactly the same as before. An initial approximation method is also needed for quick convergence of the LP process.

### 4.2.1 Construction of Knot Vector

The construction of the knot vector is most critical to the quality of interval B-spline curve approximation, because it determines the number and the distribution of interval control points. If it is ill-constructed, for instance, if the data points over a certain span defined by two knots oscillate greatly, the corresponding interval control points will inflate in order to fit the data points, resulting in a loss of accuracy. A well-balanced knot vector, on the other hand, will prevent an approximating curve from over inflating and unevenly inflating so that the width does not vary a lot along the interval curve. As stated in the *Fundamental Theorem of the Local Theory of Curves*, the curvature and the torsion of a curve completely describe the local behavior of the curve [7], therefore these two geometric features can be used to construct a well-balanced knot vector. Roughly speaking, if these two features vary greatly within a segment of the curve, then more knots are needed to subdivide this segment into several spans, within which the features only change a little. This also means that more control points will be used to control the shape of this segment by introducing more degrees of freedom.

Since initially an explicit representation of the approximating curve is not available, these geometric features can only be estimated from discrete data points which are supposed to be on the curve.

The tangent direction of a curve has a geometric interpretation as the vector pointing from one point on the curve to another point as the latter approaches the former along the curve, see Figure 4-4 and [41]. Using the fact that the resulting cubic B-spline curve is continuous in its second derivative, the vector joining two nearby consecutive data points can be roughly used as the average of the tangents within the segment bounded by these two points. The cross product of two such consecutive approximating tangent vectors can be used to approximate the binormal vector, see Figure 4-5. As the curvature $\kappa$ and the torsion $\tau$ are defined as

$$\kappa = \left|\frac{d\vec{t}}{ds}\right|, \tag{4.1}$$

$$\tau = \left|\frac{d\vec{b}}{ds}\right|, \tag{4.2}$$

where $\vec{t}$ and $\vec{b}$ are the unit tangent vector and the unit binormal vector, the angular variations of the approximating tangent vectors and the approximating binormal vectors can be used as measures of the curvature and the torsion, respectively.

The geometric information needed to construct a knot vector is extracted from the centers of the interval data points. We denote the centers as $\{\vec{p_i}\} = \{x_i, y_i, z_i\}$, $0 \leq i \leq N-1$, where $N$ is the number of the data points. The unit tangent vectors

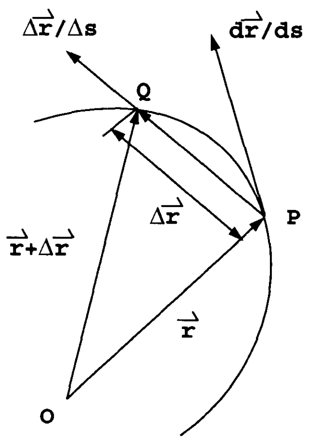


Figure 4-4: Geometric interpretation of the tangent of a curve

and the unit binormal vectors at the fitting positions are approximated as

$$\vec{t}_i = \frac{\vec{p}_{i+1} - \vec{p}_i}{|\vec{p}_{i+1} - \vec{p}_i|}, \quad 0 \le i \le N-2, \tag{4.3}$$

$$\vec{b}_i = \frac{\vec{t}_{i-1} \times \vec{t}_i}{|\vec{t}_{i-1} \times \vec{t}_i|}, \quad 1 \le i \le N-2, \tag{4.4}$$

and their angular variations $\Delta\theta_t$ and $\Delta\theta_b$ are used as measures of the curvature and the torsion, respectively, see Figure 4-5. The algorithm for constructing a well balanced knot vector for an interval approximating curve, consists of the following steps (also see Figure 4-5):

1. Construct the $N-1$ unit "tangent" vectors by equation (4.3).

2. Construct the $N-2$ unit "binormal" vectors by equation (4.4).

3. Set tolerances $\theta_\kappa$ and $\theta_\tau$ for the angular variations $\Delta\theta_t$ and $\Delta\theta_b$, respectively.

4. Set the initial knot vector as $T = \{0,0,0,0,1,1,1,1\}$ and the number of knots as 8.

5. Set the reference vector of tangent, $\vec{t}_{ref}$, and the reference vector of binormal, $\vec{b}_{ref}$, as $\vec{t}_1$ and $\vec{b}_1$ respectively.

6. From point $\vec{p}_2$ through $\vec{p}_{N-3}$, if $|1.0-\vec{t}_{ref}\cdot\vec{t}_i| < (1.0-\cos\theta_\kappa)$ and $|1.0-\vec{b}_{ref}\cdot\vec{b}_i| < (1.0-\cos\theta_\tau)$ , no knot is added, and skip to the next point; otherwise, insert the parametric value of that point into the knot vector and increase the number of knots by one, then set the $\vec{t}_i$ and $\vec{b}_i$ of the current point as the reference vectors and go to next point.

The pseudocode of this algorithm is presented in Appendix C.

The selection of the tolerances for the angular variations is determined by the user and depends upon how precisely the interval data points are expected to be fitted by the interval curve. Geometrically, the tolerances are the threshold values for the angles ($\Delta\theta_t$ and $\Delta\theta_b$ in Figure 4-5) that the two vectors ("tangent" and "binormal") sweep over as they travel from the reference point to the current point. If the angles do not exceed the tolerances, the curve is relatively flat and behaves like a straight line connecting the two points, thus no more knots are needed; otherwise, it means that the curvature or the torsion along the segment between these two points varies so much that a knot is needed to define this segment as a separate span of a B-spline curve. Larger tolerances lead to less control points and a wider interval curve, and vice versa.

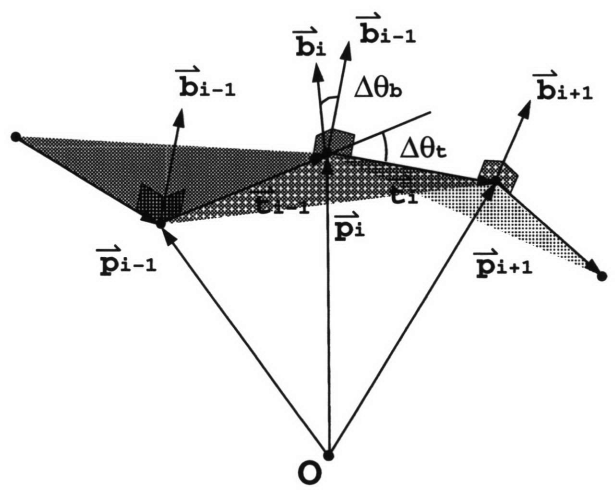


Figure 4-5: Constructing a well balanced knot vector for interval approximating curve.

### 4.2.2 Initial Approximation

Unlike the interpolation, here the initial values for the minimization process are not so straightforward since now the interval linear system is underdetermined. General least squares method can be used to obtain an interval approximation curve, but it does not ensure that the solution is feasible for the LP problem (as was the case for interpolation). Feasibility, in this case, means that the set of values satisfies the given constraints. Fortunately, most LP numerical packages accept initial values which are not feasible solutions.

As an interval B-spline curve can be represented as a single-valued B-spline curve with an associated error function which has the same form as a B-spline curve,

$$\begin{aligned} [\vec{P}](t) &= \vec{\bar{P}}(t) + [-1,1]\vec{\varepsilon}(t) \\ &= \sum_{i=0}^{n-1} \vec{\bar{D}}_i N_{i,4}(t) + [-1,1] \sum_{i=0}^{n-1} \vec{\varepsilon}_i N_{i,4}(t), \end{aligned} \tag{4.5}$$

the initial approximation procedure can thus be separated into two steps, see Figure 4-6. First, the centers of interval data points are approximated with a single-valued curve using the least squares method, which provides a set of single-valued control points and a global fitting error $\vec{\chi}$. Then, the total approximate errors $\vec{\chi}+\vec{\gamma}_i$ at all the fitting positions are fitted with a piecewise polynomial function $\vec{\varepsilon}(t)$ using the least squares method again, where $\vec{\gamma}_i$ is the vector with its components representing the errors or half-widths in three directions of the $i$-th interval data point. The results of these two steps are $\vec{\bar{D}}_i$ and $\vec{\varepsilon}_i$, respectively. The interval control points of the desired interval B-spline curve can be approximately represented as

$$[\vec{\bar{D}}_i - \vec{\varepsilon}_i, \vec{\bar{D}}_i + \vec{\varepsilon}_i], \tag{4.6}$$

which does not ensure that the resulting interval curve envelops the interval data points but has been found experimentally to be accurate enough for use as an initial approximation for the LP process.

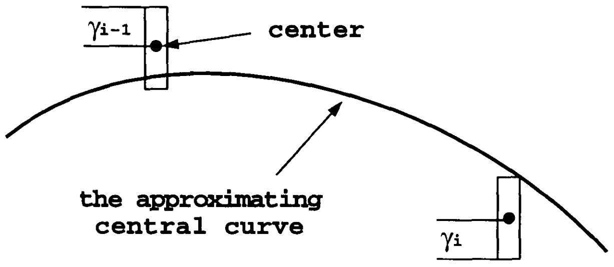


Figure 4-6: Initial curve approximation

## 4.3 Dense Data Approximation

As the number of data points increases enormously and data points get extremely close to each other, we found that an LP method will cycle indefinitely or may be unable to find a solution satisfying the constraints imposed by the inclusion principle, due to nearly dependent constraints.

In this section, we propose an interval curve approximation method using only the least squares method instead of the LP process. The basic idea is to construct a single-valued approximating curve first, and then enlarge its single-valued control points to interval control points which define an interval curve enclosing interval data points though not optimally, see Figure 4-7. Here we assume that all interval control point coordinates are positive. If not, coordinate translation and iteration of the approximation procedure are necessary. This assumption reduces various possible cases in the interval product involving control points to only one, and makes the method mathematically valid. The benefit can be seen in the proof at the end of this section.

Let

$$[\vec{P}]_i = \{[P_x]_i, [P_y]_i, [P_z]_i\}^T, \ \ 0 \leq i \leq N-1 \tag{4.7}$$

be a set of interval data points, and their centers are denoted as $\vec{p}_i$, correspondingly. After parametrization and knot construction, a single-valued curve approximating $\vec{p}_i$

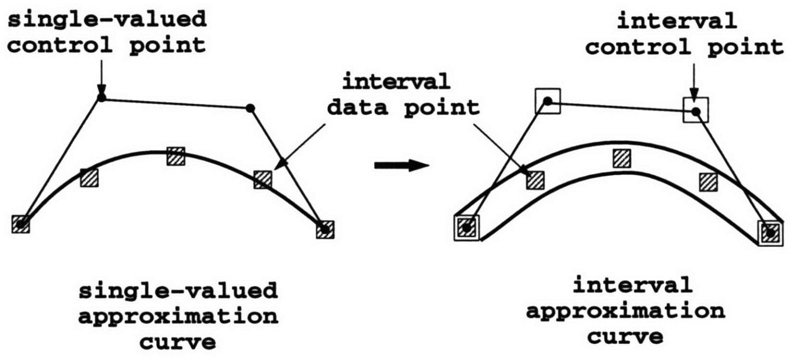


Figure 4-7: Schematic illustration of interval curve approximation method for dense data points

can be constructed as

$$\vec{p}(t) = \{p_x(t), p_y(t), p_z(t)\}^T = \sum_{k=0}^{n-1} \vec{d}_k N_{k,4}(t), \tag{4.8}$$

where $n$ is the number of control points, and $N_{k,4}(t), 0 \leq k \leq n-1$, are the exact cubic B-spline basis functions. Its control points $\vec{d}_k, 0 \leq k \leq n-1$, are obtained by applying the least squares method to the linear system

$$\vec{p}_i = \sum_{i=0}^{n-1} \vec{d}_k \bar{N}_{k,4}(u_i), \quad 0 \leq i \leq N-1, \tag{4.9}$$

where $\bar{N}_{k,4}(t)$ is the midpoint of $[N]_{k,4}(t)$, and $u_i$ is the parametric value of the $i$-th data point. The RIA evaluation of this curve is

$$[\vec{p}](t) = \{[p_x], [p_y], [p_z]\}^T = \sum_{k=0}^{n-1} \vec{d}_k [N]_{k,4}(t). \tag{4.10}$$

If we denote the componentwise differences between the lower and upper bounds of $i$-th interval data point and its corresponding evaluated point on this single-valued approximating curve as

$$\vec{\delta}_i^l = \{\delta_{x,i}^l, \delta_{y,i}^l, \delta_{z,i}^l\}^T = \begin{pmatrix} |p_x(u_i) - P_{x,i}^l| \\ |p_y(u_i) - P_{y,i}^l| \\ |p_z(u_i) - P_{z,i}^l| \end{pmatrix}, \tag{4.11}$$

$$\vec{\delta}_i^u = \{\delta_{x,i}^u, \delta_{y,i}^u, \delta_{z,i}^u\}^T = \begin{pmatrix} |p_x(u_i) - P_{x,i}^u| \\ |p_y(u_i) - P_{y,i}^u| \\ |p_z(u_i) - P_{z,i}^u| \end{pmatrix}, \tag{4.12}$$

$$0 \le i \le N-1,$$

then, $\vec{\delta}_i^l$ and $\vec{\delta}_i^u$ are no larger in components than

$$\vec{\varepsilon}_i^l = \{\varepsilon_{x,i}^l, \varepsilon_{y,i}^l, \varepsilon_{z,i}^l\}^T = \begin{pmatrix} |p_x^u(u_i) - P_{x,i}^l| \\ |p_y^u(u_i) - P_{y,i}^l| \\ |p_z^u(u_i) - P_{z,i}^l| \end{pmatrix}, \tag{4.13}$$

$$\vec{\varepsilon}_i^u = \{\varepsilon_{x,i}^u, \varepsilon_{y,i}^u, \varepsilon_{z,i}^u\}^T = \begin{pmatrix} |p_x^l(u_i) - P_{x,i}^u| \\ |p_y^l(u_i) - P_{y,i}^u| \\ |p_z^l(u_i) - P_{z,i}^u| \end{pmatrix}, \tag{4.14}$$

$$0 \le i \le N-1,$$

respectively, see Figure 4-8. If any of the components of $\vec{\varepsilon}_i^l$ and $\vec{\varepsilon}_i^u$ is beyond an expected magnitude, the parametric value $u_i$ of $[\vec{P}]_i$ where violation happens will be inserted as a new knot and the approximation will proceed with the augmented knot vector.

Assume the interval approximating curve is

$$[\vec{P}](t) = \sum_{k=0}^{n-1} [\vec{D}]_k [N]_{k,4}(t). \tag{4.15}$$

An arbitrary interval data point $[\vec{P}]_i$ with parametric value $u_i$ is to be enclosed by

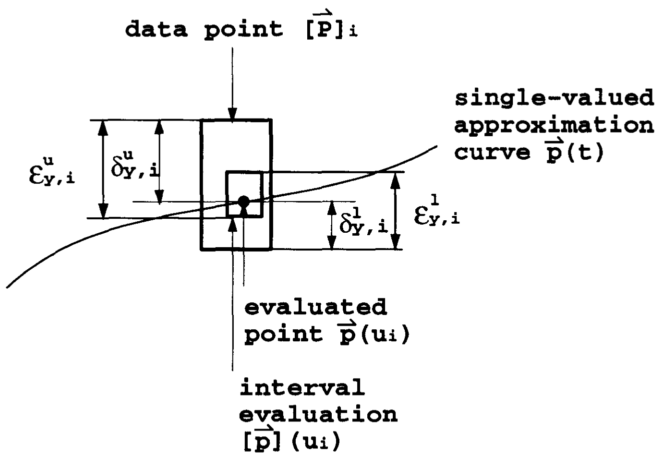

data point $[\vec{P}]_i$
single-valued approximation curve $\vec{p}(t)$
$\varepsilon^u_{y,i}$
$\delta^u_{y,i}$
$\delta^l_{y,i}$
$\varepsilon^l_{y,i}$
evaluated point $\vec{p}(u_i)$
interval evaluation $[\vec{p}](u_i)$

Figure 4-8: Componentwise differences between a single-valued approximating curve and the lower and upper bounds of interval data points (only $y$ component is shown)

the evaluated point $[\vec{P}](u_i)$, i.e.

$$[\vec{P}]_i \subseteq [\vec{P}](u_i). \tag{4.16}$$

Because $[\vec{P}](u_i)$ must be located in a certain span of the curve, equation (4.16) is determined by at most four control points which affect the span. We can use this property of local support and the fact that all the interval control point coordinates are supposed to be positive, to get a set of interval control points by enlarging $\vec{d}_k$ to an interval $[\vec{D}]_k$.

If the $i$-th data point $[\vec{P}]_i$ is located in the $j$-th span, it is affected by control points $[\vec{D}]_k, j-1 \leq k \leq j+2$, only. We thus define these four interval control points as

$$[\vec{d}_k - \vec{\varepsilon}^{\,l}_i, \vec{d}_k + \vec{\varepsilon}^{\,u}_i], \quad j-1 \leq k \leq j+2. \tag{4.17}$$

We iterate through all the $N$ points. If a control point has already been defined and the previous enlargement exceeds the current, no change is needed, i.e. the enlargements of the $k$-th control point in $x$, $y$, $z$ directions are the maximum of the corresponding components of $\vec{\varepsilon}^{\,l}_i$ and $\vec{\varepsilon}^{\,u}_i$ of all the data points within the 4 spans affected by this control point. The precise enlargement of $k$-th control point can therefore be explicitly calculated by

$$\begin{aligned} \vec{\varepsilon}_k^{\,l} &= \max_{u_i \in [t_k, t_{k+4}], 0 \le i \le N-1} \{\vec{\varepsilon}_i^{\,l}\} && (4.18) \\ \vec{\varepsilon}_k^{\,u} &= \max_{u_i \in [t_k, t_{k+4}], 0 \le i \le N-1} \{\vec{\varepsilon}_i^{\,u}\}, && (4.19) \end{aligned}$$

and the interval control points are

$$[\vec{D}]_k = \left[\vec{d}_k - \vec{\varepsilon}_k^{\,l}, \vec{d}_k + \vec{\varepsilon}_k^{\,u}\right], \quad 0 \le k \le n-1, \tag{4.20}$$

where the maximum operation is applied on each component. Remember that all the operations, even those involving only exact values, should be performed in RIA, and appropriate bounds should be taken to preserve enclosures.

We prove the enclosure of the resulting interval curve as follows

$$\begin{aligned} [\vec{P}](u_i) &= \sum_{k=0}^{n-1} [\vec{D}]_k [N]_{k,4}(u_i) \\ &= \sum_{k=j-1}^{j+2} [\vec{D}]_k [N]_{k,4}(u_i) \quad \text{(assuming } [\vec{P}](u_i) \text{ is in the } j\text{-th span).} && (4.21) \end{aligned}$$

Upon using (4.20) and considering that the interval control points are positive, we have

$$\begin{aligned} [\vec{P}](u_i) &= \sum_{k=j-1}^{j+2} \left[\vec{d}_k - \vec{\varepsilon}_k^{\,l}, \vec{d}_k + \vec{\varepsilon}_k^{\,u}\right] [N]_{k,4}(u_i) \\ &= \sum_{i=j-1}^{j+2} \left[(\vec{d}_k - \vec{\varepsilon}_k^{\,l}) N_{k,4}^l(u_i), (\vec{d}_k + \vec{\varepsilon}_k^{\,u}) N_{k,4}^u(u_i)\right] \\ &\quad \text{(by the definition of interval product)} \\ &= \left[\sum_{k=j-1}^{j+2} (\vec{d}_k - \vec{\varepsilon}_k^{\,l}) N_{k,4}^l(u_i), \sum_{k=j-1}^{j+2} (\vec{d}_k + \vec{\varepsilon}_k^{\,u}) N_{k,4}^u(u_i)\right] \\ &\quad \text{(by the definition of interval addition)} \end{aligned}$$

$$\supseteq \left[\sum_{k=j-1}^{j+2} (\vec{d}_k - \vec{\varepsilon}_k^{\,l}) N_{k,4}(u_i), \sum_{k=j-1}^{j+2} (\vec{d}_k + \vec{\varepsilon}_k^{\,u}) N_{k,4}(u_i)\right]$$
$$(\text{where} N_{k,4}(u_i) \in [N]_{k,4}(u_i))$$

$$= \left[\left(\sum_{k=j-1}^{j+2} \vec{d}_k N_{k,4}(u_i)\right) - \vec{\varepsilon}_k^{\,l}, \left(\sum_{k=j-1}^{j+2} \vec{d}_k N_{k,4}(u_i)\right) + \vec{\varepsilon}_k^{\,u}\right]. \qquad (4.22)$$

Therefore,

$$[\vec{P}](u_i) \supseteq \left[\left(\sum_{k=j-1}^{j+2} \vec{d}_k N_{k,4}(u_i)\right) - |\vec{p}^{\,u}(u_i) - \vec{P}_i^l|, \left(\sum_{k=j-1}^{j+2} \vec{d}_k N_{k,4}(u_i)\right) + |\vec{p}^{\,l}(u_i) - \vec{P}_i^u|\right]. \qquad (4.23)$$

Note that all the vector operations here are defined componentwise.

**Case 1:** If $\vec{p}^{\,u}(u_i) \geq \vec{P}_i^l$, then the lower bound of the right hand side of equation (4.23)

$$\begin{aligned}
& \sum_{k=j-1}^{j+2} \vec{d}_k N_{k,4}(u_i) - |\vec{p}^{\,u}(u_i) - \vec{P}_i^l| \\
= & \sum_{k=j-1}^{j+2} \vec{d}_k N_{k,4}(u_i) - \vec{p}^{\,u}(u_i) + \vec{P}_i^l \\
\leq & \vec{P}_i^l
\end{aligned} \qquad (4.24)$$

since

$$\sum_{k=j-1}^{j+2} \vec{d}_k N_{k,4}(u_i) \in [\vec{p}](u_i) = [\vec{p}^{\,l}(u_i), \vec{p}^{\,u}(u_i)]; \qquad (4.25)$$

**Case 2:** If $\vec{p}^{\,u}(u_i) < \vec{P}_i^l$, then the lower bound of the right hand side of equation (4.23)

$$\begin{aligned}
& \sum_{k=j-1}^{j+2} \vec{d}_k N_{k,4}(u_i) - |\vec{p}^{\,u}(u_i) - \vec{P}_i^l| \\
< & \sum_{k=j-1}^{j+2} \vec{d}_k N_{k,4}(u_i)
\end{aligned}$$

$$\leq \ \vec{p}^{u}(u_i) < \vec{P}_i^l. \tag{4.26}$$

From both cases, the lower bound of $[\vec{P}](u_i)$ is no larger than $\vec{P}_i^l$. Similarly, the upper bound of $[\vec{P}](u_i)$ is no less than $\vec{P}_i^u$. This proves that

$$[\vec{P}](u_i) \supseteq [\vec{P}]_i \tag{4.27}$$

This method works very well for dense data fitting. Another advantage over the method in relation to an LP process is the lower requirement of CPU time and memory. The algorithm is given in Appendix C.

## 4.4 Surface Fitting

### 4.4.1 Knot Vector Construction

Since measured data can be fitted first with a set of interval B-spline curves by either interpolation or approximation, surface fitting can be efficiently accomplished using a surface lofting technique, which creates an interval surface having those interval curves as its isoparametric curves.

As all the isoparametric curves are in the same parameter direction, they must have a unique, common knot vector which is also the knot vector in that direction of the resulting surface. However, if the curves are produced by the algorithms developed before, their knot vectors are most likely different. Therefore, the first step of surface creation is to construct the knot vector in the curve fitting direction for the lofting surface.

When the interpolation technique is used for curve fitting, the knot vector is selected in such a way that the number of control points does not exceed the number of data points in any individual stripe. The knot vector can be the knot vector constructed for the stripe with the maximum number of data points, or the average of all the knot vectors if the number of data points in all the stripes is identical.

If there is an enormous number of data points in each stripe, the approximation procedure is preferable to reduce storage requirements. In this case, we develop an algorithm to merge different knot vectors. Since the knot vector of each approximating curve is well constructed based on the geometrical information extracted from the discrete points in the corresponding stripe, we expect that the merged knots stay as close as possible to the original ones such that the merged knot vector is still well balanced and a high-quality approximation is achievable. In the algorithm presented in Appendix C, we essentially compare two knot vectors. If two knots from different knot vectors are within a certain small tolerance, we take an average of these knots and replace them with the resulting average value; otherwise, we just merge them in a non-decreasing sequence. The number of the knots in the resulting knot vector is bounded as

$$\max_{0 \le i \le M-1} \{K_i\} \le K \le \sum_{i=0}^{M-1} K_i \tag{4.28}$$

where $K$ is the number of knots in the merged knot vector, $M$ is the number of curves, $K_i$ is the number of knots in the knot vector of the $i$-th curve.

It should be noted that the number of control points, which is determined by $K$ and knot multiplicities, may be larger than the number of data points in a certain stripe. This will probably cause the LP problem to have an unbounded solution when fitting the data points in the stripe. Even if the number of control points does not exceed the number of data points, sometimes in a certain stripe there are more control points than data points locally, i.e. there is no data point in several adjacent spans such that there is no constraint on some control points because of the local support property. For example, in the situation of Figure 4-9 the $j$-th control point only affects the following four spans

$$[t_j, t_{j+1}], \quad [t_{j+1}, t_{j+2}], \quad [t_{j+2}, t_{j+3}], \quad [t_{j+3}, t_{j+4}].$$

Since no data point locates inside these four spans, the $j$-th control point is completely free.

To exclude the possibility of unbounded solution, the knot vector for individual curve approximation is created by the following two steps:

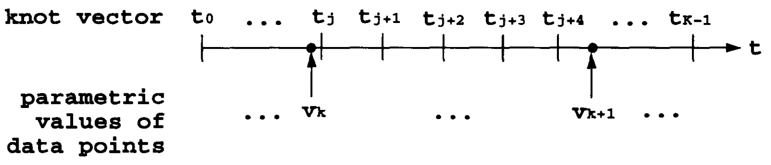

knot vector
$t_0$ ... $t_j$ $t_{j+1}$ $t_{j+2}$ $t_{j+3}$ $t_{j+4}$ ... $t_{K-1}$
t
parametric values of data points
... $v_k$ ... $v_{k+1}$ ...

Figure 4-9: A situation with unbounded solution

- *Adjustment of parametric values*

  Assume that the parametric values of the data points in the $i$-th data stripe are

$$V = \{v_0, v_1, v_2, \cdots, v_{N_i}\}, \tag{4.29}$$

where $N_i$ is the number of the data points in the stripe, and that the merged knot vector is

$$T = \{t_0, t_1, t_2, \cdots, t_{K-1}\}. \tag{4.30}$$

For each of the $N_i$ data points, we compare its parametric value with the knot vector and find the closest knot. If the difference between the parametric value and the knot value is within a given tolerance, the parametric value is set to be equal to the knot value. For example, in Figure 4-9, we set $v_k = t_j$ if

$$|v_k - t_j| \leq \text{a given tolerance.} \tag{4.31}$$

- *Intersection of the parametric values and the merged knot vector*

  Assume the adjusted parametric values are

$$V' = \{v'_0, v'_1, v'_2, \cdots, v'_{N_i}\}. \tag{4.32}$$

We take the intersection set $T_i = V' \cap T$ as the knot vector for curve approximation of the $i$-th data stripe.

Since $T_i \subseteq V'$, every span has at least one data point, which ensures that no unbounded solution will occur. And also, since $T_i \subseteq T$, a knot insertion process is needed after fitting the $i$-th stripe using $T_i$. In Appendix B, we prove that knot insertion in interval B-spline curves also preserves a conservative enclosure.

The curve fitting procedure thus produces a rectangular set of interval control points $[\vec{D}]_{i,j}, 0 \leq i \leq M-1, 0 \leq j \leq m_2 - 1$, where $m_2$ is the number of control points in the curve fitting direction.

## 4.4.2 Surface Lofting in Interval Arithmetic

As each of the interval curves becomes an isoparametric curve of the resulting surface, it must be assigned a parametric value $u_k, 0 \leq k \leq M-1$, if we assume that the lofting direction is the $u$ parameter direction. Since the control polygon of a B-spline curve has the variation diminishing property, it is reasonable to consider that it approximates the B-spline curve. Hence, we parametrize the curves using their control polygons as follows:

1. Construct $m_2$ sequences of data points. The $j$-th sequence consists of all the $j$-th control points of the curves, where $0 \leq j \leq m_2 - 1$. In Figure 4-10, such a sequence is marked by a dotted line.

2. Apply the chord length method to each of the $m_2$ sequences to obtain $m_2$ sets of parametric values, each of which has $M$ numbers.

3. Take the average of the $m_2$ sets of parametric values as the parametric values of the curves.

Since the interval curves which fit the data stripes now become the isoparametric curves with fixed parametric values $u_k, 0 \leq k \leq M-1$, of the resulting interval surface, the following conditions must hold for each of these $M$ curves

$$\sum_{i=0}^{m_1-1} \sum_{j=0}^{m_2-1} [\vec{Q}]_{i,j}[N]_{i,4}(u_k)[N]_{j,4}(v) = \sum_{j=0}^{m_2-1} [\vec{D}]_{k,j}[N]_{j,4}(v) \tag{4.33}$$

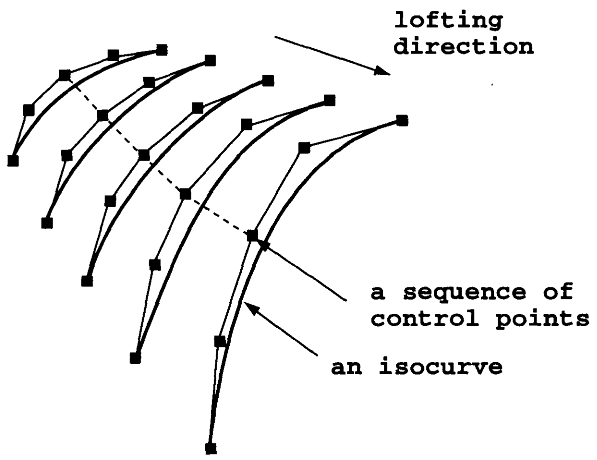


Figure 4-10: Surface lofting

where $[\vec{D}]_{k,j}, 0 \leq k \leq M-1$, are the control points of the $k$-th interval curve, $m_1$ is the number of control points in the $u$ direction of the surface.

Equation (4.33) is satisfied if and only if

$$\sum_{i=0}^{m_1-1} [\vec{Q}]_{i,j}[N]_{i,4}(u_k) = [\vec{D}]_{k,j} \tag{4.34}$$

where $0 \leq j \leq m_2 - 1$, since $[N]_{j,4}(v)$ is the RIA evaluation of a basis for splines of order 4 over a fixed knot vector $S$.

Equation (4.34) is a set of $m_2$ linear systems of $M$ equations and $m_1$ unknowns, all with the same coefficient matrix. They can be rewritten in matrix form as

$$[\mathbf{B}][\vec{\mathbf{Q}}] = [\vec{\mathbf{D}}] \tag{4.35}$$

where

$$[\mathbf{B}] = ([N]_{i,4}(u_k))_{0\leq k\leq M-1, 0\leq i\leq m_1-1} \tag{4.36}$$

$$[\vec{\mathbf{Q}}] = ([\vec{Q}]_{i,j})_{0\leq i\leq m_1-1, 0\leq j\leq m_2-1} \tag{4.37}$$

$$[\vec{\mathbf{D}}] = ([\vec{D}]_{k,j})_{0\leq k\leq M-1, 0\leq j\leq m_2-1}. \tag{4.38}$$

Each of the $m_2$ linear system is equivalent to the problem of fitting an interval B-spline curve to a corresponding sequence of data points constructed as in Figure 4-10, and therefore can be solved by either curve interpolation or approximation method. The knot vector in the $u$ direction can be constructed in the same way as before. But it is easier now to merge the knot vectors because all the knot vectors are subsets of $\{u_k\}, 0 \leq k \leq M-1$, which are the parametric values of the isoparametric curves.

### 4.4.3 Summary

In closing, the entire procedure for the approximation of a set of interval data points with an interval bicubic B-spline surface consists of the following steps

1. Parametrize the data points on each stripe, and construct the knot vector for each approximating curve.
2. Merge the knot vectors of all the approximating curves to one knot vector.
3. Use this knot vector to approximate the data points on each stripe by an interval cubic B-spline curve.
4. Use the interval control points obtained in the last step to assign each of the interval approximating curves a parametric value, and construct the knot vector in the lofting direction.
5. Loft the interval cubic B-spline curves obtained in Step 3 to form an interval bicubic B-spline surface.

For the approximation of a data set of size $M \times N$ with an interval B-spline surface having $m \times n$ control points, the running time is $T(M,N) = O(MT_1(N,n)) + O(nT_1(M,m))$, where $T_1$ is the running time of the LP routine used and depends on the number of iterations. The number of operations performed per iteration is roughly proportional to $n_f^2$, where $n_f$ is the number of variables fixed on their upper or lower bounds, see [30].

# Chapter 5

# Implementation and Numerical Examples

## 5.1 Implementation

### 5.1.1 Overview

The complete procedure of approximating measured data with an interval bicubic B-spline surface, is summarized as follows:

- **Input**

  1. Interval data points converted from measured data by incorporating uncertainties from all sources.
  2. Some control variables, such as the tolerances for knot construction and the maximum permissible width for interval control points.

- **Curve Approximation**

  1. Parametrize the data points in each data stripe.
  2. Construct a knot vector for curve approximation of each data stripe.
  3. Merge the knot vectors to create a common knot vector.

4. Approximate all the stripes of interval data points with interval cubic B-spline curves defined over the same knot vector.

- **Surface Lofting**

  1. Assign a parametric value to each curve.
  2. Loft the interval curves into an interval bicubic B-spline surface.

- **Output**

  1. Display of the interval data points and the interval evaluated points at the fitting positions on the interval surface.
  2. Data files storing numerical results.

This interval data fitting method is implemented in the *C++ language* on a graphics workstation, and has been tested with various examples. Besides functional modules, a new data type *Interval*, which is defined using the *Object Oriented Programming* technique [33] and which allows RIA operations on computer, is used. Moreover, the linear programming routine in *NAG* software package is used [30].

The input of the software system is a data file containing measured data which is grouped in stripes, and some control variables, such as the maximum permissible width of interval control points and the tolerances for those angular variations defined for knot construction (see Section 4.2.1). The output is both numerical and graphical. We use *Examiner Viewer*, a visualization tool in *OpenInventor*, to display the interval evaluated points at the fitting positions on the resulting interval surface, as well as the data points for comparison. The output data files are the file storing numerical results, and the file in the format required by *Praxiteles*, a surface interrogation system developed at the Design Laboratory of the Department of Ocean Engineering at MIT [1].

### 5.1.2 Some Issues about the Implementation

**1. Evaluation of the B-spline basis in RIA**

To preserve a *conservative enclosure*, each B-spline basis function is evaluated in RIA, that is, all the operations involved in the recursive formula (2.14) are performed as in the definitions in Section 2.2.2. But this evaluation certainly leads to negative values in some cases. For example, if both $[N]_{i,M-1}(t)$ and $[N]_{i+1,M-1}(t)$ are exactly zero in the RIA evaluation (2.17), $[N]_{i,M}(t)$ will be a symmetric interval with its mid-point at zero. This is not desirable according to the property of the B-spline basis functions being non-negative. Hence, in every step of recurrence, we intentionally discard the negative part in an interval return value, or in the above case force $[N]_{i,M}(t)$ to be zero.

**2. Oscillation of raw data**

The data collected from the surface of an object may oscillate, if there are some small ridges and the sampling density is very high. If we want to neglect those ridges, because, for example, they are far below the dimension we are concerned with, we can use a certain filter to eliminate the oscillation such that only major features are recovered, since the oscillation will drastically increase the number of control points needed to accurately fit the data. We construct a knot vector based on the filtered data. However, data fitting is still performed on the raw data, because it is the data we want to fit.

**3. Linear constraints in LP**

The LP routine used here (based on [30]) is the only weak point in the present implementation which could potentially damage the robustness of the method, because it is performed in FPA. If necessary, an LP routine which computes an optimal solution strictly satisfying the imposed constraints needs to be developed.

The input of NAG routine has a set of positive tolerances needed to be specified by the user, defining the maximum permissible absolute violation in each constraint in order for a point to be considered feasible, see [30]. It is said that large values of these tolerances mean that the corresponding constraints could be violated by a significant amount. As a remedial measure, we widen interval data points by these tolerances to eliminate possible violation. This helped in all examples tested.

**4. Dense data approximation**

If measured points are extremely dense, the linear constraints imposed by some of the points may be roughly identical and regarded as nearly dependent by an LP routine. This will cause the LP routine to cycle indefinitely, if it does not have any device to avoid cycling. Since *NAG* software package only has an LP routine without cycling detection, we use the method proposed in Section 4.3 for dense data approximation. In the next section, we will show two examples of same measured object with different sampling densities.

**5. Display of the resulting surface**

We did not develop a sophisticated 3D graphics system which is able to help us visualize an interval curve or surface in detail. As we discussed in Chapter 2, it is also very difficult to extract those enveloping curves or surfaces. For some of the examples, we display an interval curve or surface in the form of a dense set of interval points evaluated on the curve or surface. If it is an interval planar curve without self-intersection and has a visible width, we use two single-valued B-spline curves to approximate the enveloping curve which actually is a composite curve. Similarly, we use two single-valued B-spline surfaces to approximate the enveloping surface of an interval surface. However, we do not prove the enclosure graphically but numerically. These approximating enveloping curves and surfaces are just for display. As an example, Figure 5-1 shows how we generally approximate the enveloping curve of an interval curve using two single-valued B-spline curves. The approximation is based on the discussion on enveloping curve in Section 2.4.5 and the local support property of B-spline curves and surfaces.

## 5.2 Numerical Examples

The sources of the data in the following examples fall into three categories:

1. Data sampled from geometric objects which can be described by analytic equations, with truncating errors;

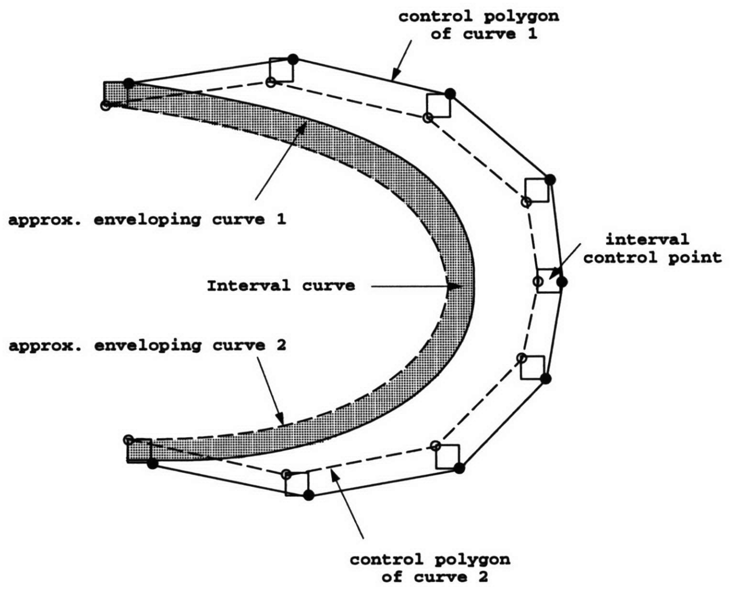

control polygon of curve 1
approx. enveloping curve 1
interval control point
Interval curve
approx. enveloping curve 2
control polygon of curve 2

Figure 5-1: Approximation of the enveloping curve of an interval B-spline curve with two single-valued B-spline curves. The control points of these two curves are marked with dots and circles, respectively.

2. Grid data sampled uniformly from objects which have idealized B-spline representations, with truncating errors; and,

3. Data collected from the surfaces of real objects by the measurement system described in Chapter 4, with measurement errors.

All the examples are run on a 33MHz graphics workstation. We will give the real running time of some of the examples as rough indications of CPU time required.

### 5.2.1 Curve Interpolation

As we will see from the following examples, our interpolation method provides a set of interval control points which define an interval interpolating curve tightly enveloping data points with extremely small inflation in width.

**Example 1:** Interpolation of data points sampled from an ellipse

- Data source: an ellipse described by

$$\begin{cases} x = 10cos(t) \\ y = 5sin(t) \end{cases} \tag{5.1}$$

- Sampling method: uniform evaluation over $t \in [-\pi/2, \pi/2]$
- Error type: truncating error equal to $\pm 0.05$
- Number of data points: 11
- Width of data points: uniformly equal to 0.1
- Fitting method: interpolation
- Number of control points: 11
- Average width of fitting points: 0.100000001
- Running time: $<$ 2sec.
- Observations:
  1. Table 5.1 lists the interval control points, including those of the initial interpolation and those of the final minimized result. The minimization process reduced the average width from 0.3621 to 0.100000001.

  2. Table 5.2 compares the data points with the evaluated points at the interpolating positions on the resulting curve. The width increases are actually introduced by RIA operations and are of the order of $10^{-10}$.
  3. Figure 5-2 shows the enveloping curves of the interval curve.

**Example 2:** Interpolation of data points sampled from a self-intersecting curve

- Data source: A self-intersecting curve described by

$$r = -a \cos 2\theta \sec \theta \tag{5.2}$$

- Sampling method: uniform evaluation over $\theta \in [-\pi/3, \pi/3]$

|  | Initial Interpolation | Final Result |
|---|---|---|
|  | [ -0.1 , 0 ] | [ -0.1000000005 , 4.9999606685e-10 ] |
|  | [ 1.5795830388,2.2049633139 ] | [ 1.8422731758,1.9422731768 ] |
|  | [ 4.4784311036,5.0064718147 ] | [ 4.6924514587,4.7924514597 ] |
|  | [ 7.7245404559,8.0314655907 ] | [ 7.8280030228,7.9280030238 ] |
|  | [ 9.3351828719,9.6284112916 ] | [ 9.4317970813,9.5317970822 ] |
| x | [ 10.018172821,10.289369419 ] | [ 10.103771120,10.203771121 ] |
|  | [ 9.3350545375,9.6285396260 ] | [ 9.4317970813,9.5317970823 ] |
|  | [ 7.7237850789,8.0322209677 ] | [ 7.8280030228,7.9280030238 ] |
|  | [ 4.4729281145,5.0119748039 ] | [ 4.6924514587,4.7924514597 ] |
|  | [ 1.5837837561,2.2007625966 ] | [ 1.8422731758,1.9422731768 ] |
|  | [ -0.1 , 0 ] | [ -0.1000000005,4.9999622507e-10 ] |
|  | [ -5.1,-5 ] | [ -5.1000000005,-4.9999999995 ] |
|  | [-5.2460724422,-4.6206921671] | [-4.9833823052,-4.8833823042] |
|  | [-4.9043701167,-4.3763294056] | [-4.6903497617,-4.5903497607] |
|  | [-3.2545416069,-2.9476164721] | [-3.1510790400,-3.0510790390] |
|  | [-1.9345568672,-1.6413284475] | [-1.8379426579,-1.7379426569] |
| y | [-0.1855982988,0.0855982988 ] | [-0.1000000005, 4.9999670783e-10 ] |
|  | [ 1.5412001131,1.8346852016 ] | [ 1.6379426569,1.7379426579 ] |
|  | [ 2.8468610951,3.1552969839 ] | [ 2.9510790390,3.0510790400 ] |
|  | [ 4.2708264165,4.8098731059 ] | [ 4.4903497607,4.5903497617 ] |
|  | [ 4.5248928844,5.1418717249 ] | [ 4.7833823042,4.8833823052 ] |
|  | [ 4.9 , 5 ] | [ 4.8999999995 , 5.0000000005 ] |

Table 5.1: The control points of the interval curve interpolating the data points sampled from an ellipse with artificial errors.

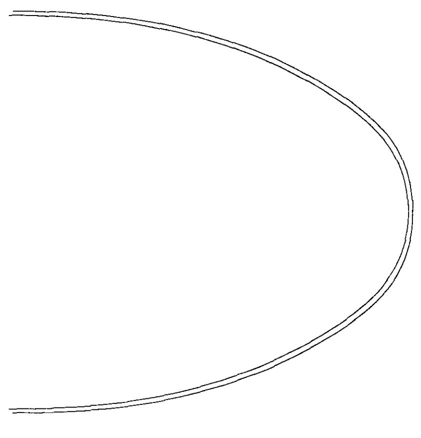


Figure 5-2: The lower and upper enveloping curves of the interval curve interpolating the data points sampled from an ellipse.

| | Given Pts. | Evaluated Pts. |
|---|---|---|
| x | [-0.1, 0.0] | [-0.1000000005, 5.000011e-10] |
| | [2.9, 3.0] | [2.8999999995, 3.0000000005] |
| | [5.7, 5.8] | [5.6999999995, 5.8000000005] |
| | [7.9, 8.0] | [7.8999999995, 8.0000000005] |
| | [9.4, 9.5] | [9.3999999995, 9.5000000005] |
| | [9.9, 10.0] | [9.8999999995, 10.000000001] |
| | [9.4, 9.5] | [9.3999999995, 9.5000000005] |
| | [7.9, 8.0] | [7.8999999995, 8.0000000005] |
| | [5.7, 5.8] | [5.6999999995, 5.8000000005] |
| | [2.9, 3.0] | [2.8999999995, 3.0000000005] |
| | [-0.1, 0.0] | [-0.1000000005, 5.000013e-10] |
| y | [-5.1, -5.0] | [-5.1000000005, -4.9999999995] |
| | [-4.8, -4.7] | [-4.8000000005, -4.6999999995] |
| | [-4.1, -4.0] | [-4.1000000005, -3.9999999995] |
| | [-3.0, -2.9] | [-3.0000000005, -2.8999999995] |
| | [-1.6, -1.5] | [-1.6000000005, -1.4999999995] |
| | [-0.1, 0.0] | [-0.1000000005, 5.000022e-10] |
| | [1.4, 1.5] | [1.3999999995, 1.5000000005] |
| | [2.8, 2.9] | [2.7999999995, 2.9000000005] |
| | [3.9, 4.0] | [3.8999999995, 4.0000000005] |
| | [4.6, 4.7] | [4.5999999995, 4.7000000005] |
| | [4.9, 5.0] | [4.8999999995, 5.0000000005] |

Table 5.2: The numerical result of the interpolation of a set of interval points originally sampled from an ellipse.

- Error type: truncating error equal to $\pm 0.05$
- Number of data points: 41
- Width of data points: uniformly equal to 0.1
- Fitting method: interpolation
- Number of control points: 41
- Average width of fitting points: 0.100000001
- Observations:
  1. Figure 5-3 displays the resulting interval curve by showing a dense set of discrete interval points on the curve.

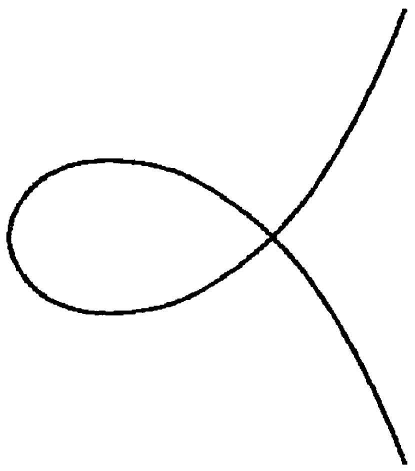


Figure 5-3: Interval curve interpolating a set of points sampled from the curve described by equation (5.2).

### 5.2.2 Curve Approximation

An important aspect concerning the quality of interval curve approximation is the width inflation. In our method, it is reduced by using a well-balanced knot vector as well as a LP process. We will see from the following examples that the width of an approximating curve is directly related to the knot vector that determines the number and the distribution of control points.

**Example 3:** Approximation of data points sampled from a helix

- Data source: A helix described by

$$\begin{cases} x = 10cos(t), \\ y = 10sin(t), \\ z = \frac{t}{2\pi} \end{cases} \tag{5.3}$$

- Sampling method: uniform evaluation over $t \in [0, 6\pi]$
- Error type: artificial error in *Gaussian* distribution ranging from 0.0 to 0.001
- Number of data points: 101
- Width of data points: $\leq 0.002$
- Fitting method: approximation
- Observations:
  1. As discussed before, the width inflation of an approximating curve is directly related to the selection of the knot vector. Table 5.3 gives the approximation results with different knot vectors. In this table, $\theta_\kappa$ and $\theta_\tau$ are the angular tolerances defined for knot construction, see Section 4.2.1. Larger tolerances result in a smaller number of control points and a wider interval curve. Here $\omega$ is the maximum permissible ratio of the widths of the desired control points to the average width of the data points.
  2. Figure 5-4 shows the interval curve of Result 3 represented by a dense set of interval points on the curve.

| | Given Data | Result 1 | Result 2 | Result 3 |
|---|---|---|---|---|
| $\theta_\kappa$ and $\theta_\tau$ (in degrees) | | 15.0 | 30.0 | 45.0 |
| $\omega$ | | 7.0 | 7.0 | 18.0 |
| Control points | | 52 | 36 | 23 |
| Avg. width in $X$ | 0.001690 | 0.011749 | 0.011829 | 0.030696 |
| Avg. width in $Y$ | 0.001837 | 0.012661 | 0.012861 | 0.033072 |
| Avg. width in $Z$ | 0.001518 | 0.010612 | 0.010625 | 0.027321 |
| Running time (in sec.) | | 40 | 22 | 12 |

Table 5.3: Comparison of some approximation results with different knot vectors, of the helix example.

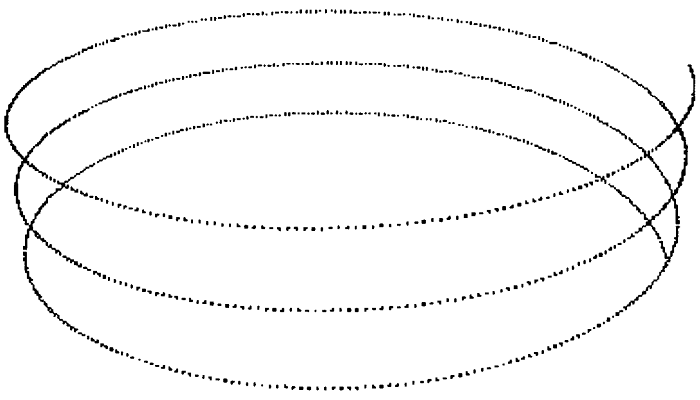


Figure 5-4: Interval curve approximating a set of points sampled from the helix described by equation (5.3).

**Example 4:** Approximation of data points sampled from a skewed propeller blade

- Data source: A skewed propeller blade described by a bicubic B-spline surface with $9 \times 88$ control points (Figure 5-17)
- Sampling method: uniform evaluation of an isoparametric curve
- Error type: truncating error equal to $\pm 0.0005$
- Number of data points: 100
- Width of data points: uniformly equal to 0.001
- Fitting method: approximation
- Number of control points: 16
- Average width of fitting points: 0.007
- Observations:
    1. The knot vector

$$\begin{aligned} T = \quad & \{0, 0, 0, 0, 0.1967, 0.3529, 0.4476, 0.4831, 0.4948, 0.5008, \\ & 0.5068, 0.5173, 0.5492, 0.7037, 0.8584, 0.9722, 1, 1, 1, 1\} \end{aligned}$$

shows that more knots are used around the leading edge (corresponding to a parameter value $\approx 0.5$). If the knot vector has uniform internal knots, i.e. if

$$T = \{0, 0, 0, 0, \frac{1}{13}, \frac{2}{13}, \frac{3}{13}, \frac{4}{13}, \frac{5}{13}, \frac{6}{13}, \frac{7}{13}, \frac{8}{13}, \frac{9}{13}, \frac{10}{13}, \frac{11}{13}, \frac{12}{13}, 1, 1, 1, 1\},$$

though the same number of control points is used, the average width increases to 0.042 from 0.007. Figure 5-5 shows both curves.

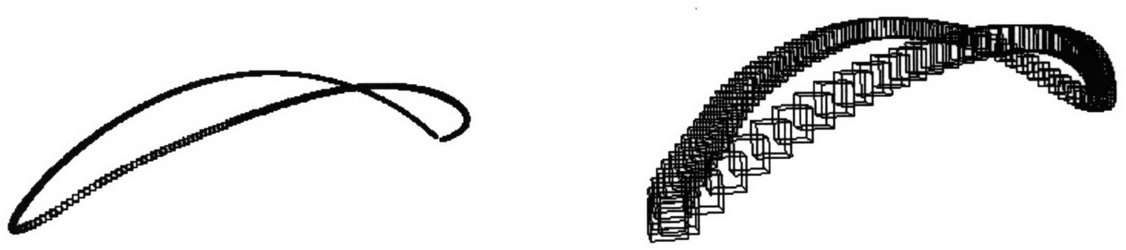


Figure 5-5: The interval curves approximating a set of points on the blade shown in Figure 5-17, with different knot vectors.

**Example 5:** Approximation of data points sampled from a sinusoid curve with high-frequency noise

- Data source: a curve defined by

$$y = 10 \sin x/20 + \sin x/2 \tag{5.4}$$

- Sampling method: uniform evaluation over $x \in [0, 20\pi]$
- Error type: truncating error equal to $\pm 0.05$
- Number of data points: 101
- Width of data points: uniformly equal to 0.1
- Fitting method: approximation
- Observations:
  1. Table 5.4 compares two fitting results. Figure 5-6 shows some of the data points and the interval approximating curve with a smaller number of control points by displaying two enveloping curves (in the same region as the data points). Figure 5-7 shows a part of the approximating curve with more control points by displaying a dense set of interval points on the curve. With the different selections of the knot vector, different geometric features are recovered.

|  | No. of control points | Average width |
|---|---|---|
| Result 1 | 10 | 3.3897 in X, 11.9847 in Y |
| Result 2 | 33 | 0.2675 in X, 0.4074 in Y |

Table 5.4: The approximation of a set of data points sampled from a sinusoid curve with different numbers of control points.

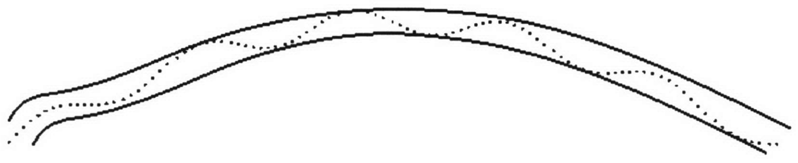


Figure 5-6: Interval B-spline curve which approximates a set of data points sampled from a sinusoid curve with 10 control points. The interval curve is shown by two enveloping curves, between which are the data points.

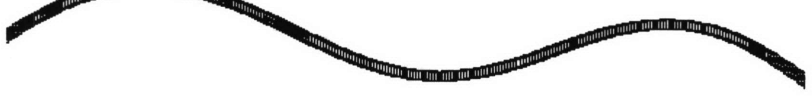


Figure 5-7: A part of the interval B-spline curve which approximates the same data points as Figure 5-6 with 33 control points. No data points are shown here.

### 5.2.3 Surface Fitting

This section provides some examples demonstrating our surface approximation method. In Examples 6 to 9, the data are obtained by evaluating surfaces which already have idealized B-spline representations. We truncate the values to a certain number of digits and represent them as interval data points. Examples 10 to 12 use data measured by the robotic measurement system described in Chapter 4.

**Example 6:** Interpolation of data points sampled from the suction surface of an airfoil

- Data source: the suction surface of an airfoil, described by a bicubic B-spline surface with $37 \times 31$ control points (see Figure 5-8)
- Sampling method: evaluation over a parametrically uniform grid of $20 \times 20$
- Error type: truncating error equal to $\pm 0.005$
- Number of data points: 400
- Width of data points: uniformly equal to 0.01
- Fitting method: interpolation
- Number of control points: $20 \times 20$
- Average width of fitting points: 0.010000002
- Observations:
    1. Figure 5-9 shows the upper (wireframe) and the lower (shaded) enveloping surfaces.

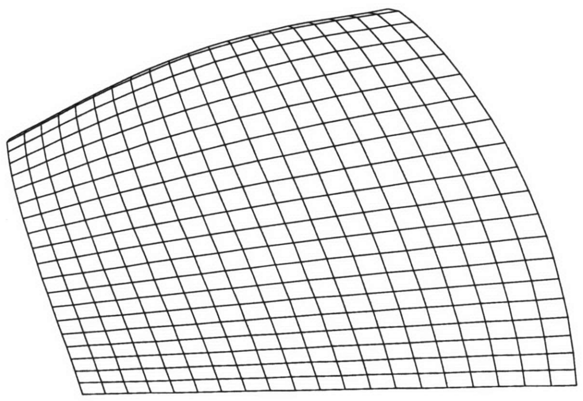


Figure 5-8: The suction surface of an airfoil.

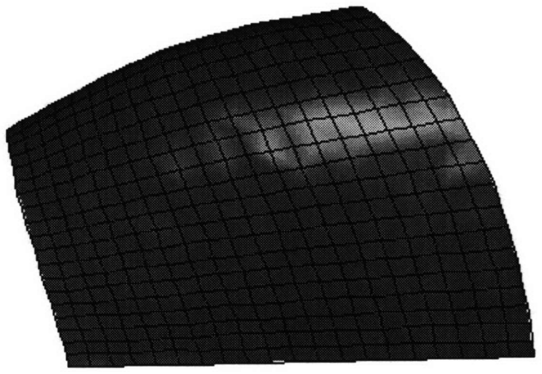


Figure 5-9: The lower and upper enveloping surfaces of the interval surface which interpolates a set of data points sampled from the airfoil in Figure 5-8.

**Example 7:** Approximation of data points sampled from a turbine blade

- Data source: a turbine blade described by a bicubic B-spline surface with $53 \times 20$ control points (see Figure 5-10)
- Sampling method: evaluation over a parametrically uniform grid of $50 \times 10$ (see Figure 5-11)
- Error type: truncating error equal to $\pm 0.0005$
- Number of data points: 500
- Width of data points: uniformly equal to 0.001
- Fitting method: approximation in curve fitting direction (with higher density of data points) and interpolation in the other direction
- Number of control points: $13 \times 10$
- Average width of fitting points: 0.16
- Observations:
  1. The average width of the evaluated points at the fitting positions on the surface is 0.16, which is much larger than 0.001, due to the small number of control points of the approximating surface.
  2. Figure 5-12 shows two sides of the interval surface, where the shaded is the interior enveloping surface while the wireframe is the exterior enveloping surface.

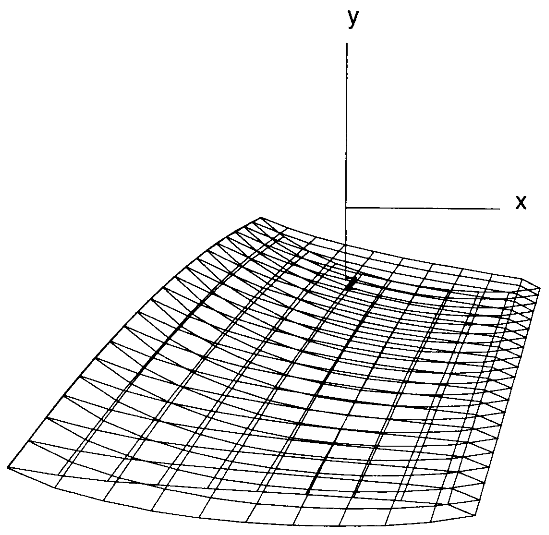


Figure 5-10: A turbine blade

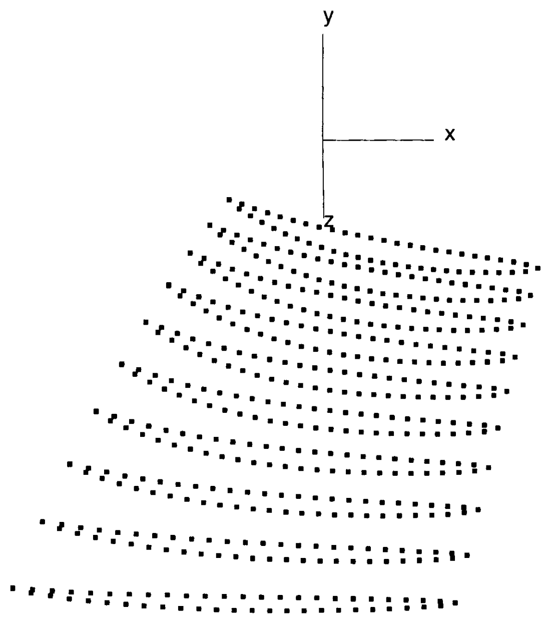


Figure 5-11: Data points sampled from the turbine blade in Figure 5-10.

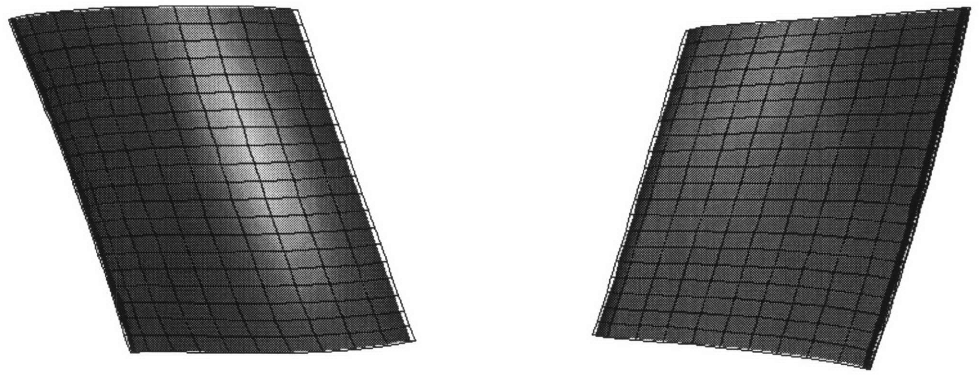


Figure 5-12: Two sides of the interval surface approximating the data points in Figure 5-11.

**Example 8:** Approximation of data points sampled from a propeller blade

- Data source: A propeller blade described by a bicubic B-spline surface with $11 \times 87$ control points (see Figure 5-13)
- Sampling method: evaluation over a parametrically uniform grid of $50 \times 40$ (see Figure 5-14)
- Error type: truncating error equal to $\pm 0.0005$
- Number of data points: 2000
- Width of data points: uniformly equal to 0.001
- Fitting method: approximation in both directions
- Number of control points: $24 \times 27$
- Average width of fitting points: 0.226
- Observations:
  1. In the knot construction for the curve approximation of 40 transverse sections, we set the tolerances of angular variations for knot construction $\theta_\kappa = \theta_\tau = 10°$. The knot vectors of different sections have different sizes, and are merged to a common knot vector with 28 knots. Therefore, in the curve fitting direction, 24 control points are used.

2. In the lofting direction, 27 knots are used, as we try to reduce the size of the control net. The average width of the fitting points reaches a large value equal to 0.226, which is an inaccurate approximation. The failure results from both the curve fitting and the surface lofting. In the curve fitting, the number of control points used is much smaller than that of the original surface; in the surface lofting, we can see from Figure 5-15 that some of the longitudinal stripes, which consist of control points from different curves and are lined up from the root to the tip, have great oscillation. Therefore, an interpolation procedure is more preferable in the lofting direction, if the number of transverse data stripes is not enormous, to avoid further enlargement of interval width.

3. A more accurate approximation with a control net of $40 \times 34$, has the average widths in $x$, $y$, and $z$ equal to 0.0163, 0.0158 and 0.0218, respectively. Figure 5-16 shows the interval approximating surface where again the wireframe surface is the exterior enveloping surface and the shaded one is the interior.

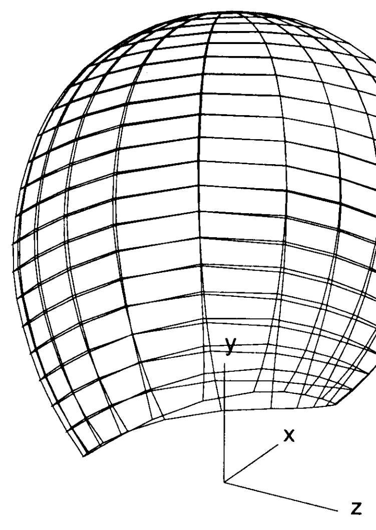
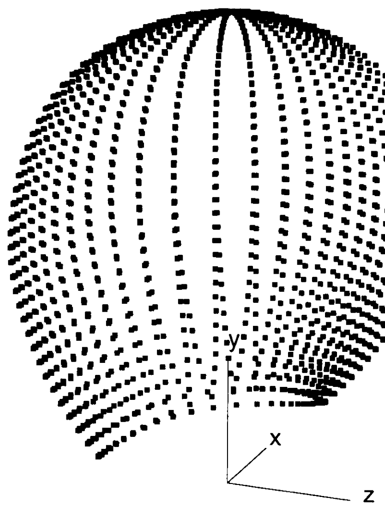


Figure 5-13: A propeller blade



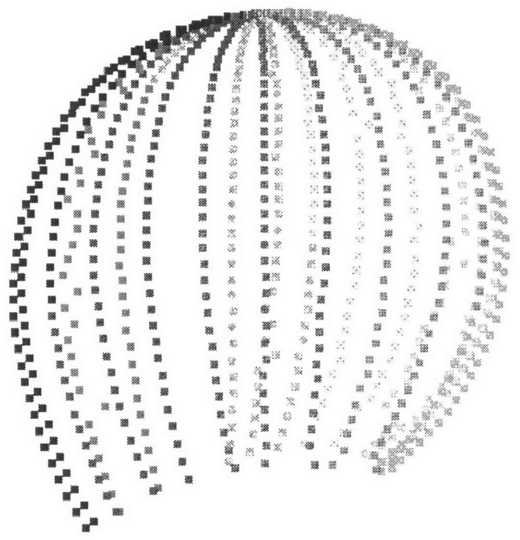


Figure 5-15: The control points of the interval curves fitting the transverse sections of the propeller blade.

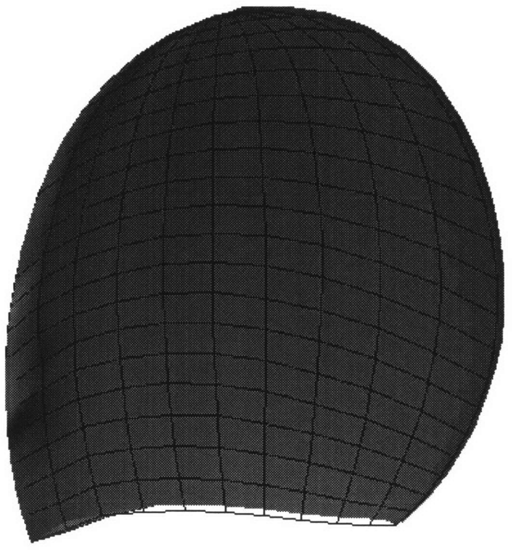


Figure 5-16: The interval surface which approximates the data points in Figure 5-14.

**Example 9:** Approximation of data points sampled from a skewed propeller blade

- Data source: A skewed propeller blade described by a bicubic B-spline surface with $9 \times 88$ control points (see Figure 5-17)
- Sampling method: evaluation over a parametrically uniform grid of $20 \times 100$ (see Figure 5-18)
- Error type: truncating error equal to $\pm 0.0005$
- Number of data points: 2000
- Width of data points: uniformly equal to 0.001
- Fitting method: approximation in the direction with high density of data points and interpolation in the other direction
- Number of control points: $20 \times 30$
- Average width of fitting points: 0.046
- Running time: 265 sec.
- Observations:
  1. In the curve fitting in this example, due to high sampling density around the leading edge, the merged knot vector may have more knots than data points in the neighborhood of the leading edge; that is, there may be no point thus no constraint in some spans of a B-spline approximating curve. A subset of the complete merged knot vector is then used and knot insertion is applied following the curve approximation.
  2. Figure 5-19 shows a set of interval points evaluated on the interval approximating B-spline surface.

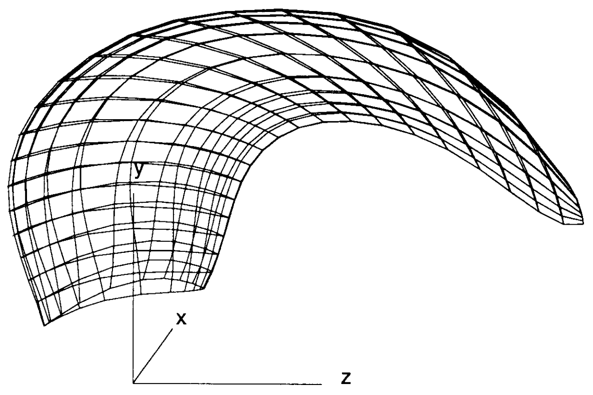


Figure 5-17: A skewed propeller blade represented by a bicubic B-spline surface.

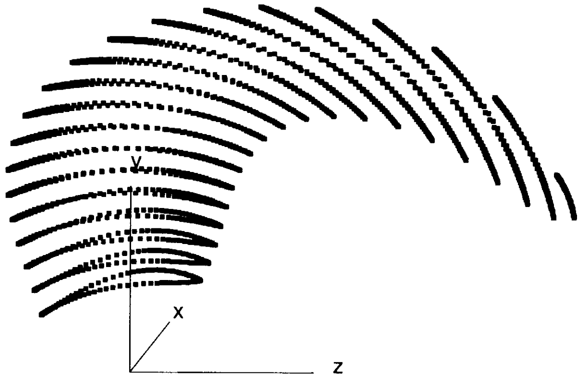


Figure 5-18: Data points sampled the skewed propeller blade in Figure 5-17.

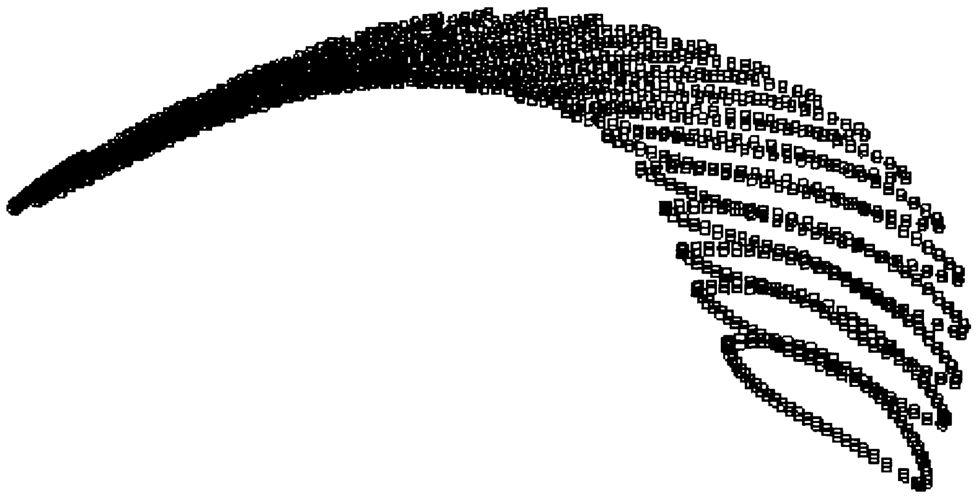


Figure 5-19: Interval B-spline surface which approximates a set of data points sampled from the skewed propeller blade in Figure 5-17.

**Example 10:** Approximation of data points measured from a manufactured object by 4D Imager [11]

- Data source: a manufactured wooden block (see Figure 5-20)
- Sampling method: measurement by the 4D Imager described in Chapter 4 (see Figure 5-21)
- Error type: the precision of the instrument is four effective decimal digits, therefore, the error is no more than 0.00005.
- Number of data points: 2698 (32 stripes)
- Width of data points: uniformly equal to 0.0001
- Fitting method: approximation in the direction with higher density of data points and interpolation in the other direction (see Figure 5-21)
- Number of control points: $32 \times 33$
- Average width of fitting points: 0.0240
- Running time: 583 sec.
- Observations:
  1. Since the data contains high-frequency noise (due in most part to the method used in manufacturing the wooden model, see one of the data stripes in Figure 5-22 for example,) a Gaussian filter [31] is passed over the data before the knot vectors are constructed, since the oscillation of the data due to this noise causes an overestimate of the knots that are needed for fitting. Thus the filter is used so that only the major features of the curves are considered. The curves, however, are fit to the original data, including noise.
  2. The noise in the real data also causes an expected increase in the width of the interval approximating curves. If the approximation procedure is again used for surface lofting, the widths of the interval control points will be further enlarged. Considering that the data lines in

the lofting direction are not so dense (only 32 lines), the interpolation procedure is the better approach.

3. It can be seen that the average width of the evaluated points on the fitting surface is much larger than that of the data points. The reason is that the magnitude of the oscillation of the data points due to the noise is much larger than the widths of the data points and therefore the interval surface has to use much larger widths to fit those oscillatory data points.
4. The interval surface is shown by the lower and upper enveloping surfaces in Figure 5-23.

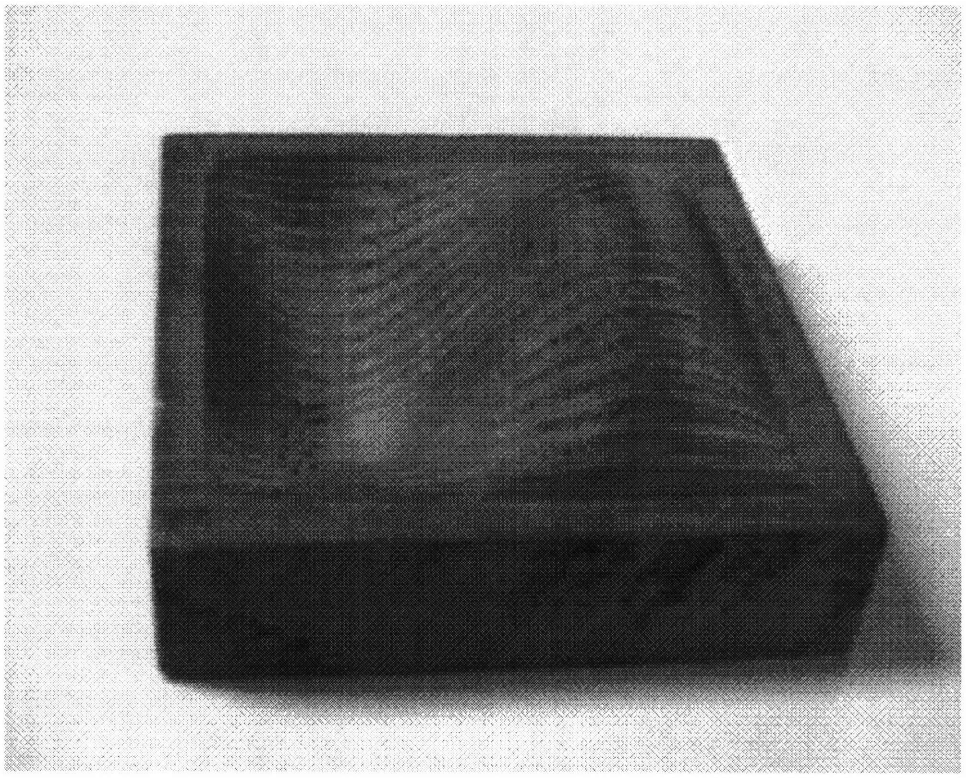


Figure 5-20: The picture of the wooden model of Example 10.

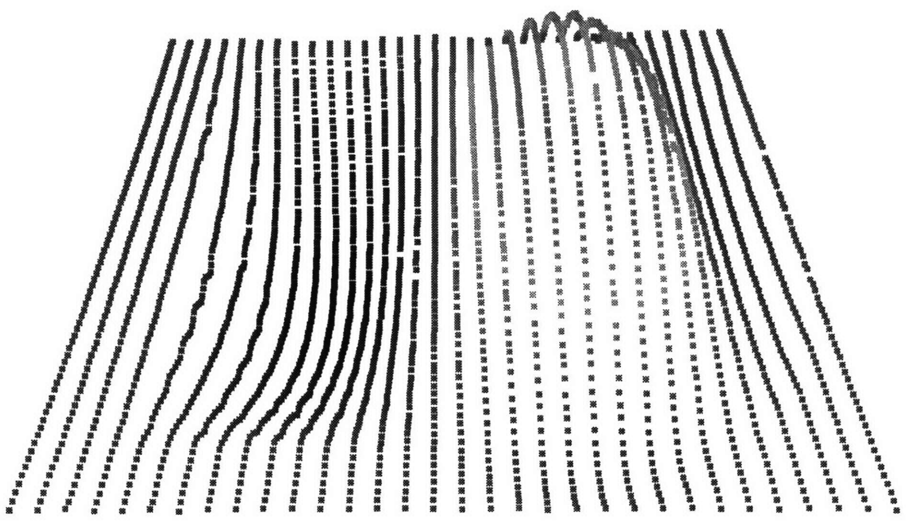


Figure 5-21: Data points collected from the model shown in Figure 5-20.

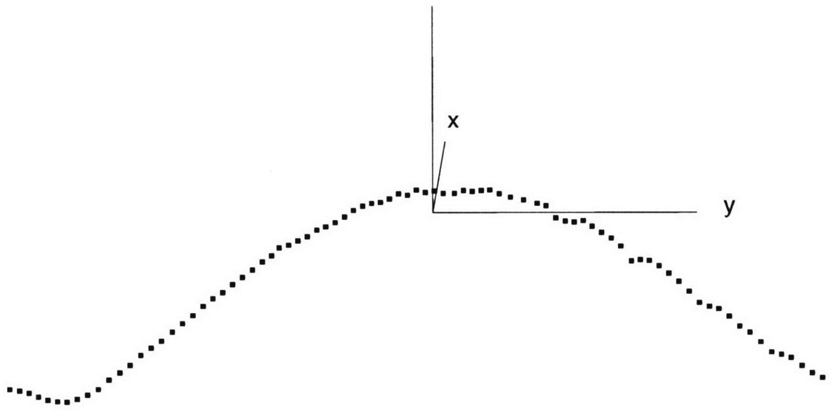


Figure 5-22: One of the data stripes in Figure 5-21.

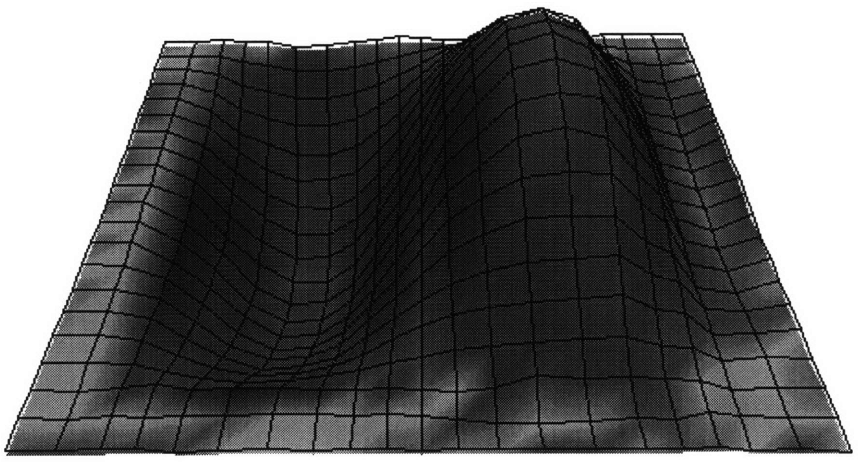


Figure 5-23: The interval surface fitting the data of Figure 5-21; the upper enveloping surface shown in wireframe, the lower enveloping surface shown shaded.

### 5.2.4 Dense Data Approximation

In this section, we approximate a dense data set with interval B-splines, using the method proposed in Section 4.3. The data were collected from the same object of Example 10, but with higher density, see Figure 4-2. We first approximate one stripe with an interval curve, and then fit the entire set with an interval B-spline surface.

**Example 11:**

The data stripe to be approximated has 336 data points, see Figure 5-24. The maximum allowable deviation from the single-valued approximating curve is set as large as 200 times of the width of the data points, due to the oscillation of the data points. In the initial approximation which uses a single-valued curve to fit the interval data points, there are 14 points where the approximation errors are over the maximum allowable deviation. The knot vector is then refined by inserting the parametric values of those 14 points into the knot vector. A new single-valued approximating curve is generated based on the new knot vector and no more violation happens. Table 5.5 is the approximation result. Figure 5-25 displays the interval approximating curve. A numerical test shows that all the data points in the stripe are enclosed by this interval B-spline curve.

| | |
|---|---|
| Initial knot construction | 54 knots |
| Deviation violation | 14 points |
| Adjusted knot vector | 68 knots |
| Avg. width in $X$ | 0.00316 |
| Avg. width in $Y$ | 0.00548 |
| Avg. width in $Z$ | 0.01508 |

Table 5.5: Curve approximation of the data stripe in Figure 5-24.

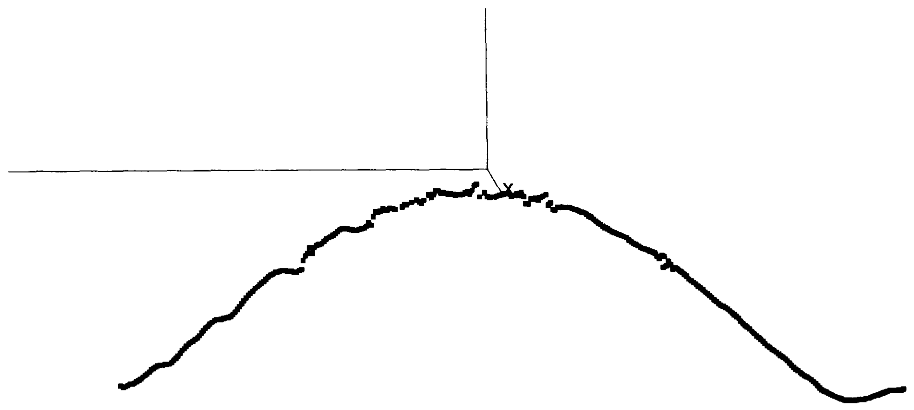


Figure 5-24: One stripe of the data points of Figure 4-2.

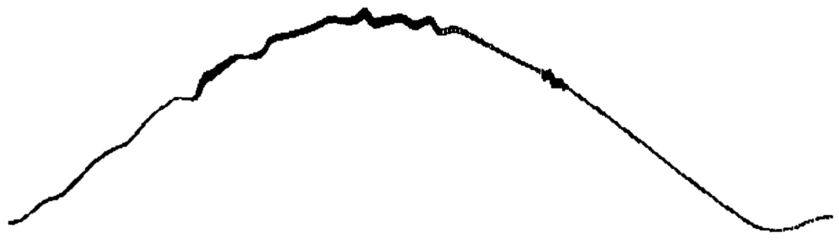


Figure 5-25: Interval curve which approximates the data points in Figure 5-24

**Example 12:**

In this surface approximation example, we actually do not adjust the knot vector according to the violation of the maximum allowable deviation. Such a method may be implemented but complicates the knot merging process. We simply construct a common knot vector, and then approximate each stripe and loft the approximating curves. The lower and upper enveloping surfaces are similar to Figure 5-23 in display. Table 5.6 is the numerical result.

| | |
|---|---|
| Number of data points | 10,826 |
| Size of control net | $32 \times 42$ |
| Avg. width in $X$ | 0.00217 |
| Avg. width in $Y$ | 0.00906 |
| Avg. width in $Z$ | 0.01376 |

Table 5.6: Surface approximation of the data set in Figure 4-2

# Chapter 6

# Conclusion and Future Work

## 6.1 Summary

In this thesis, interval surface reconstruction methods for robust reverse engineering are developed. As all sources of errors are accounted for, we ensure that the resulting interval B-spline surface envelopes the real positions of measured points on the surface of an object. This will allow us to construct an interval solid model which will not have contradictory topological and geometrical information, and finally to develop data for a robust solid modeling system that will never fail due to uncertain data and limited computation precision.

In addition, we discuss some unique numerical and geometric properties of interval B-splines, such as error propagation in the RIA evaluation of interval B-spline basis functions, coordinate translation of interval B-splines, hodograph of an interval B-spline curve, enveloping curves or surfaces of an interval B-spline curve or surface, and knot insertion for interval B-splines.

Surface fitting can be done by either interpolation or approximation, depending on the requirement of fitting precision. Interval data points can be interpolated with an interval B-spline curve by solving an interval linear system using a general solver in interval arithmetic. But this proves to be unsatisfactory with respect to the accuracy. The interpolation problem is then formulated as a minimization problem with the constraints imposed by the inclusion principle, and approached using a linear programming process with the loose solution obtained by solving the interval linear system as an initial approximation. This interval interpolation technique creates an interval curve which tightly bounds interval data points with its ranges at the interpolating positions almost the same as those of the data points.

The approximation method also applies a linear programming technique to make the interval curve as tight as possible. Since the knot vector determines the number and the distribution of the control points and therefore determines the approximation quality, a well-balanced knot vector is constructed based on the geometrical information extracted from discrete data points such that a high-quality approximation will be possible. Also, an initial approximation method using the least squares method is used to provide a linear programming process a close but not necessarily feasible initial guess to speed up convergence. Additionally, a reliable approximation method without the LP process is developed for dense data fitting, if no LP routine that has the capability of preventing cycling is available.

Interval surface is created by lofting the interval curves constructed by either interpolation or approximation. Here, an algorithm is developed to merge a set of different knot vectors to a unique, common knot vector.

## 6.2 Future Work

The linear programming routine used for minimization is the only step which uses *floating-point arithmetic* (FPA) in the entire procedure. So far, we have not yet investigated the robustness problem that may arise due to FPA computation. A linear programming routine which strictly abides by the constraints taking into account numerical computation errors should be developed.

Another problem in interval solid modeling is interval surface intersection and the representation of the resulting intersection curves. Though the problem has been studied thoroughly in [17, 18, 19], the integration method used there will cause intersection curve (represented by a set of discrete points) to increase in width due to error propagation. A better solver of the interval initial value problem should be used to get a tighter intersection curve evaluation. In the literature, Lohner [24] proposed an integration method that uses interval versions of one-step methods as well as some other techniques to reduce error accumulation. Numerical examples of interval surface intersection are needed to test both robustness and efficiency of this integration method.

Research on the properties of interval B-splines, such as the enveloping curves or surfaces of an space interval B-spline curve or an interval surface, and the range of the tangent vector of an interval B-spline curve or surface, needs to be done. Algorithms, such as for the extraction of a single-valued and well defined CAD model from an interval solid model, need to be developed.

# Appendix A

# Coordinate Translation of an Interval B-Spline Curve

Coordinate translation changes the shape of an interval B-spline curve if performed in *RIA*. The inclusion relation between the translated curve and the curve defined by the translated control points depends on the control points and the translation vector. However, an interval curve that envelopes the translated curve can be constructed by slightly enlarging the translated control points.

Assume that interval curve $[\vec{P}_2](t)$ is obtained translating interval curve $[\vec{P}](t)$ by $\vec{p}$, that is

$$[\vec{P}_2](t) = [\vec{P}](t) - \vec{p}, \tag{A.1}$$

where

$$[\vec{P}](t) = \sum_{i=0}^{n-1} [\vec{D}]_i [N]_{i,M}(t). \tag{A.2}$$

Then, since

$$1 \in \sum_{i=0}^{n-1} [N]_{i,M}(t), \tag{A.3}$$

and $\vec{p}$ is a real vector so that Theorem 1(4) can be applied,

$$\begin{aligned}
[\vec{P}_2](t) &\subseteq [\vec{P}](t) - \vec{p}\sum_{i=0}^{n-1}[N]_{i,M}(t) \\
&= \sum_{i=0}^{n-1}[\vec{D}]_i[N]_{i,M}(t) - \sum_{i=0}^{n-1}\vec{p}\,[N]_{i,M}(t) \\
&= \sum_{i=0}^{n-1}\Big([\vec{D}]_i[N]_{i,M}(t) - \vec{p}\,[N]_{i,M}(t)\Big). \qquad \text{(A.4)}
\end{aligned}$$

As we know from the property of subdistributivity (Theorem 1(4)),

$$\begin{aligned}
[X] &= [A]\cdot([B]+[C]) \qquad \text{(A.5)} \\
&= \{x = a(b+c) | a \in [A], b \in [B], c \in [C]\} \\
&\subseteq [Y] = [A][B] + [A][C] \qquad \text{(A.6)} \\
&= \{y = ab + \tilde{a}c | a, \tilde{a} \in [A], b \in [B], c \in [C]\}.
\end{aligned}$$

Assuming $[A] \geq 0$, we want to find a real number $W$ such that

$$[Y] \subseteq [X] + W[-1,1]. \qquad \text{(A.7)}$$

One observation about the product of two interval numbers $[M]$ and $[N]$, where $[M]$ is non-negative and $[N]$ is arbitrary, is that

$$\begin{aligned}
[P] &= [M]\cdot[N] = [M^l, M^u]\cdot[N^l, N^u] \\
&= [(M^l \textit{ or } M^u)\cdot N^l, (M^l \textit{ or } M^u)\cdot N^u], \qquad \text{(A.8)}
\end{aligned}$$

by the definition of interval product, see equation (2.8). From equation (A.5) and (A.6), we have

$$\begin{aligned}
[X] &= [X^l, X^u] \\
&= [A^l, A^u]\cdot[B^l + C^l, B^u + C^u] \qquad \text{(A.9)} \\
[Y] &= [Y^l, Y^u]
\end{aligned}$$

$$= [A^l, A^u] \cdot [B^l, B^u] + [A^l, A^u] \cdot [C^l, C^u]. \tag{A.10}$$

In the computation of $X^l$ and $Y^l$,

- If $B^l + C^l \leq 0$, then

$$X^l = A^u \cdot (B^l + C^l) \tag{A.11}$$

$$Y^l = \begin{cases} A^u \cdot B^l + A^u \cdot C^l, & if \;\; B^l \leq 0 \;\; and \;\; B^u \leq 0; \\ A^l \cdot B^l + A^u \cdot C^l, & if \;\; B^l \geq 0; \\ A^u \cdot B^l + A^l \cdot C^l, & if \;\; C^l \geq 0. \end{cases} \tag{A.12}$$

Hence,

$$|Y^l - X^l| = \begin{cases} 0 \\ |B^l| \cdot w([A]) \\ |C^l| \cdot w([A]). \end{cases} \tag{A.13}$$

Because $B^l + C^l \leq 0$, if one of $B^l, C^l$ is greater than 0, it must be the one with the smaller absolute value. Therefore,

$$|Y^l - X^l| \leq \min\{|B^l|, |C^l|\} \cdot w([A]). \tag{A.14}$$

- If $B^l + C^l \geq 0$, then

$$X^l = A^l \cdot (B^l + C^l) \tag{A.15}$$

$$Y^l = \begin{cases} A^l \cdot B^l + A^l \cdot C^l & if \;\; B^l \geq 0 \;\; and \;\; B^u \geq 0; \\ A^u \cdot B^l + A^l \cdot C^l & if \;\; B^l \leq 0; \\ A^l \cdot B^l + A^u \cdot C^l & if \;\; C^l \leq 0. \end{cases} \tag{A.16}$$

Similarly,

$$|Y^l - X^l| \leq \min\{|B^l|, |C^l|\} \cdot w([A]). \tag{A.17}$$

For $X^u$ and $Y^u$, similarly, we have

$$|Y^u - X^u| \leq \min\{|B^u|, |C^u|\} \cdot w([A]). \tag{A.18}$$

With the definition of the absolute value of an interval number,

$$\min\{|B^l|, |C^l|\} \leq \max\{|[B]|, |[C]|\}, \tag{A.19}$$

$$\min\{|B^u|, |C^u|\} \leq \max\{|[B]|, |[C]|\}, \tag{A.20}$$

which leads to the following relation

$$|Y^l - X^l| \leq \max\{|[B]|, |[C]|\} \tag{A.21}$$

$$|Y^u - X^u| \leq \max\{|[B]|, |[C]|\}. \tag{A.22}$$

Since

$$X^l - |Y^l - X^l| \leq Y^l \leq Y^u \leq X^u + |Y^u - X^u|, \tag{A.23}$$

we have

$$X^l - \max\{|[B]|, |[C]|\} \leq Y^l \leq Y^u \leq X^u + \max\{|[B]|, |[C]|\}, \tag{A.24}$$

and thus,

$$[Y] \subseteq [X] + \max\{|[B]|, |[C]|\} w([A]) \cdot [-1, 1]. \tag{A.25}$$

Applying this relation to equation (A.4), we obtain a new interval curve $[\vec{P}_1](t)$ in the new frame, where

$$[\vec{P}_2](t) \subseteq [\vec{P}_1](t) = \sum_{i=0}^{n-1} [\vec{D}_1]_i [N]_{i,M}(t), \tag{A.26}$$

with its control points

$$[\vec{D}_1]_i = [\vec{D}]_i - \vec{p} + \max\{|[\vec{D}]_i|, |\vec{p}\,|\} w([N]_{i,M}(t)) \cdot [-1,1], \quad 0 \le i \le n-1, \quad \text{(A.27)}$$

where $[\vec{D}]_i - \vec{p}$ are the translated control points. Because $w([N]_{i,M}(t))$ is extremely small, the width increase due to the adjustment is small.

**Example:**

The original interval B-spline curve in this example is the same as *Example 1* in Chapter 5, except that the $x$ components of the interval control points have been increased by 20.0 so that they are all positive. The translation vector is $\vec{p} = (5,0,0)$.

We demonstrate the translation method only using the $x$ component of the interval curve. The interval point evaluated at $t = 0.5$ on the translated curve is

$$[P_2]_x(0.5) = [24.89999999949983, 25.000000000500084].$$

The interval point with the same parametric value on the interval curve defined by the translated control points is

$$[P_3]_x(0.5) = [24.899999999499844, 25.000000000500062].$$

Thus, $[P_3]_x(0.5) \subset [P_2]_x(0.5)$, which confirms Case 1 in Section 2.4.3 because the $x$ components of all the original control points are greater than $p_x$ and $p_x > 0$. If we use the translation method proposed in this appendix, the same evaluated point is

$$\begin{aligned} [P_1]_x(0.5) &= [24.899999999499752, 25.000000000500158] \\ &\supset [P_2]_x(0.5). \end{aligned}$$

# Appendix B

# Knot Insertion for Interval Cubic B-Spline Curve

Knot insertion for interval B-spline curve follows the same procedure as in the single-valued case except that the calculation of control points is performed in RIA. In this appendix, we first present the interval knot insertion, then prove that it preserves an enclosure.

The interval knot insertion scheme presented here is a direct extension from the method for a single-valued B-spline curve presented in Farin et al [10].

Let an interval cubic B-spline curve $[\vec{P}](t)$ be defined over the knot vector

$$\{t_0, \cdots, t_{j-1}, t_{j+1}, \cdots, t_K\} \tag{B.1}$$

with $t_0$ and $t_K$ having multiplicity 4, where $K-1$ is the number of spans created by the knot vector. The interval control points are $[\vec{D}]_i, 0 \leq i \leq K+1$. The same B-spline curve can also be defined over a refined knot vector

$$\{t_0, \cdots, t_{j-1}, t_j, t_{j+1}, \cdots, t_K\} \tag{B.2}$$

and the new control points are related to the original $[\vec{D}]_i$ via the equations:

$$
\begin{aligned}
[\vec{Q}]_i &= [\vec{D}]_i, \quad 0 \le i \le j-3, \\
[\vec{Q}]_i &= \left(\frac{t_{i+3}-t_j}{t_{i+3}-t_{i-1}}\right)[\vec{D}]_{i-1} + \left(\frac{t_j-t_{i-1}}{t_{i+3}-t_{i-1}}\right)[\vec{D}]_i, \\
& \quad i = j-2, j-1, j, \\
[\vec{Q}]_{i+1} &= [\vec{D}]_i, \quad j \le i \le K+1.
\end{aligned}
\tag{B.3}
$$

For an arbitrary curve in the family of curves defined by the interval B-spline curve $[\vec{P}](t)$, its control points $\vec{d_i} \in [\vec{D}]_i, 0 \le i \le K+1$. Because the new control points $[\vec{Q}]_i, 0 \le i \le K+2$ are affine maps of $[\vec{D}]_i$ (see Figure B-1), there must exist a set of points $\vec{q_i} \in [\vec{Q}]_i, 0 \le i \le K+2$, which are the images of $\vec{d_i} \in [\vec{D}]_i, 0 \le i \le K+1$, under the same mapping relation as equation B.3 and defines the same curve as the latter. This means that each curve in $[\vec{P}](t)$ must also be in the new interval curve. Indeed, because RIA is used in interval evaluation, the new curve envelopes the old one with a conservative enclosure.

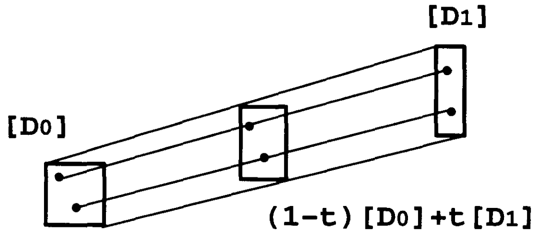


Figure B-1: Affine map of a new control point

# Appendix C

# Algorithms

In this appendix, we present the algorithms of the following procedures in pseudocode:

**ESTIMATE-KNOT()** constructs a knot vector for curve approximation of a set of data points. The following subroutine is called but not included here:

- NORMALIZE(), which normalizes a vector.

**MERGE()** merges the internal knot vectors of different curves to a common knot vector for surface approximation. The following subroutines are called and included here:

- MERGE-TWO(), which merges two internal knot vectors;
- ADJUST(), which adjusts the parametric values of data points in a data stripe and also creates a knot vector for curve approximation of the data stripe.

The following subroutine is called by MERGE-TWO() but not included here:

- COMMBINE(), which simply combines two one-dimensional arrays in ascending sequence and without multiplicity.

**APPROX-DENSE()** approximates a dense set of data points with an interval cubic B-spline curve. The following subroutine is called and included here:

- SET-CTRL-PTS(), which calculates the interval control points of the resulting interval curve.
- ESTIMATE-KNOT().

The following subroutines are called but not included here:

- READ-FILE(), which reads a data file;
- TRANSLATE(), which performs coordinate translation;
- PARAMETRIZE(), which parametrizes data points;
- LEAST(), the least squares method for single-valued data approximation;
- CAL-ERROR(), which calculates the deviations of a single-valued curve approximating interval data points, see equation (4.13) and (4.14).
- CHECK-ERROR(), which checks if the deviations exceed a prespecified magnitude;
- ADJUST-KNOT(), which refines the knot vector to improve the quality of approximation;
- FIND-MAX-ERROR(), which finds the maximum errors.

The algorithms for other procedures like curve interpolation and approximation and surface approximation can be easily constructed with the algorithms provided here and thus are not given.

**Algorithm 1** ESTIMATE-KNOT: construct a knot vector for the curve approximation of data points $\vec{P}_i, 1 \leq i \leq n$

```
Require: $n$, the number of data points; $\vec{P}[1..n]$, the data points; $t[1..n]$, the parametric values; $\theta_\kappa$ and $\theta_\tau$, the tolerances for the angular variations defined for knot estimation.
Ensure: $m$, the number of control points; $k[0..m+3]$, the knot vector.
    {$\triangleright$ get the vectors connecting two consecutive points}
 1: for $i = 1$ to $n - 1$ do
 2:     $\vec{T}[i] \leftarrow \vec{P}[i+1] - \vec{P}[i]$
    {$\triangleright$ calculate the cross products of two consecutive $\vec{T}_i$}
 3: for $i = 1$ to $n - 2$ do
 4:     $\vec{B}[i] \leftarrow \vec{T}[i] \times \vec{T}[i+1]$
    {$\triangleright$ initialize}
 5: $m \leftarrow 4$
 6: for $i = 1$ to $n$ do
 7:     $index[i] \leftarrow 0$
 8: $\vec{T}_{old} \leftarrow \vec{T}[2]$
 9: $\vec{B}_{old} \leftarrow \vec{B}[1]$
10: $\vec{T}_{new} \leftarrow \vec{T}[3]$
11: $\vec{B}_{new} \leftarrow \vec{B}[2]$
    {$\triangleright$ estimate the number of knots}
12: for $i = 3$ to $n - 2$ do
13:     $\Delta\vec{T} \leftarrow$ NORMALIZE($\vec{T}_{old}$)$\cdot$ NORMALIZE($\vec{T}_{new}$)
14:     $\Delta\vec{B} \leftarrow$ NORMALIZE($\vec{B}_{old}$)$\cdot$ NORMALIZE($\vec{B}_{new}$)
15:     if $|1 - \Delta\vec{T}| > \varepsilon_\kappa$ or $|1 - \Delta\vec{B}| > \varepsilon_\tau$ then
16:         $index[i] \leftarrow 1$
17:         $m \leftarrow m + 1$
18:         $\vec{T}_{old} \leftarrow \vec{T}_{new}$
19:         $\vec{B}_{old} \leftarrow \vec{B}_{new}$
20:     $\vec{T}_{new} \leftarrow \vec{T}[i+1]$
21:     $\vec{B}_{new} \leftarrow \vec{B}[i]$
    {$\triangleright$ construct the knot vector}
22: for $i = 0$ to $3$ do
23:     $k[i] \leftarrow t[1]$
24: $j \leftarrow 4$
25: for $i = 3$ to $n - 2$ do
26:     if $index[i] = 1$ then
27:         $k[j] \leftarrow t[i]$
28:         $j \leftarrow j + 1$
29: for $i = m$ to $m + 3$ do
30:     $k[i] \leftarrow t[n]$
```

**Algorithm 2** MERGE: merge internal knot vectors of different B-spline curves

**Require:** $nl$, the number of curves; $nk1[1..nl]$, the numbers of internal knots in the knot vectors; $k1[1..nl][1..nk1[i]], 1 \leq i \leq nl$, the internal knot vectors to be merged; $ng[1..nl]$, the number of data points in the data stripes; $v[1..nl][1..ng[i]], 1 \leq i \leq nl$, the parametric values;

**Ensure:** $n$, the number of internal knots in the merged knot vector; $k[1..n]$, the merged internal knot vector; $nk2[1..n]$, the numbers of internal knots in the knot vectors constructed for curve approximation of all the data stripes; $k2[1..nl][1..nk2[i]], 1 \leq i \leq nl$, the internal knot vectors constructed for curve approximation of all the data stripes.

```
   {▷ initialize}
1: $n \leftarrow nk1[1]$
2: **for** $i = 1$ to $nk1[1]$ **do**
3:     $k[i] \leftarrow k1[i]$
   {▷ merge the knot vectors}
4: **for** $i = 2$ to $nl$ **do**
5:     MERGE-TWO($n, k, nk1[i], k1[i]$)
   {▷ adjust the parametric values}
6: **for** $i = 1$ to $nl$ **do**
7:     ADJUST($n, k, ng[i], v[i], nk2[i], k2[i]$)
```

**Algorithm 3** MERGE-TWO: merge two internal knot vectors

**Require:** $n1$, the number of internal knots in the first knot vector; $k1[1..n1]$, the first internal knot vector; $n2$, the number of internal knots in the second knot vector; $k2[1..n2]$, the second internal knot vector; $tol$, the tolerance for two knots from different knot vectors to be considered as the same;

**Ensure:** $n$, the number of internal knots in the merged knot vector; $k[1..n]$, the merged internal knot vector;

```
 1: $n \leftarrow 0$
    {$\triangleright$ combine the two knot vectors. $index$ indicates which knot vector a knot in the combined knot vector originated from}
 2: $ttl \leftarrow$ COMBINE$(n1, k1, n2, k2, index[1..ttl])$
 3: $knot1 \leftarrow k1[1]$; $knot2 \leftarrow k1[2]$; $i \leftarrow 3$
 4: while $i \leq ttl$ do
 5:   $knot3 \leftarrow k1[i]$; $fix \leftarrow i$; $i \leftarrow i + 1$
 6:   if $index[fix - 2] = index[fix - 1]$ then
 7:     $n \leftarrow n + 1$; $last \leftarrow fix$
 8:     $k[n] \leftarrow knot1$; $knot1 \leftarrow knot2$; $knot2 \leftarrow knot3$
 9:   else
10:     if $index[fix - 1] = index[fix]$ then
11:       if $|knot2 - knot1| < tol$ then
12:         $n \leftarrow n + 1$; $k[n] \leftarrow \frac{knot1+knot2}{2}$; $knot1 \leftarrow knot3$; $knot2 \leftarrow k1[i]$
13:       else
14:         $n \leftarrow n + 1$; $k[n] \leftarrow knot1$; $n \leftarrow n + 1$; $k[n] \leftarrow knot2$
15:         $knot1 \leftarrow knot3$; $knot2 \leftarrow k1[i]$;
16:       $last \leftarrow i$; $i \leftarrow i + 1$
17:     else
18:       if $|knot2 - knot1| \leq |knot3 - knot2|$ then
19:         if $|knot2 - knot1| \leq tol$ then
20:           $n \leftarrow n + 1$; $k[n] \leftarrow \frac{knot1+knot2}{2}$; $knot1 \leftarrow knot3$; $knot2 \leftarrow k1[i]$
21:           $last \leftarrow i$; $i \leftarrow i + 1$
22:         else
23:           $n \leftarrow n+1$; $k[n] \leftarrow knot1$; $knot1 \leftarrow knot2$; $knot2 \leftarrow knot3$; $last \leftarrow fix$
24:       else
25:         $n \leftarrow n + 1$; $k[n] \leftarrow knot1$ $knot1 \leftarrow knot2$; $knot2 \leftarrow knot3$; $last \leftarrow fix$
    {$\triangleright$ deal with the end of the combined knot vector}
26: if $last = ttl$ then
27:   if $index[last] = index[last - 1]$ then
28:     $n \leftarrow n + 2$; $k[n - 1] \leftarrow knot1$; $k[n] \leftarrow knot2$
29:   else
30:     if $|knot2 - knot1| \leq tol$ then
31:       $n \leftarrow n + 1$; $k[n] \leftarrow \frac{knot1+knot2}{2}$
32:     else
33:       $n \leftarrow n + 2$; $k[n - 1] \leftarrow knot1$; $k[n] \leftarrow knot2$
34: if $last = ttl + 1$ then
35:   $n \leftarrow n + 1$; $k[n] \leftarrow knot1$
```

**Algorithm 4** ADJUST: adjust the parametric values of data points in a stripe, and construct the knot vector of the approximating curve of the stripe

```
Require: $nk$, the number of internal knots in the merged knot vector; $k[1..nk]$, the
    internal knots in the merged knot vector; $ng$, the number of data points in the
    stripe; $v[1..ng]$, the parametric values of the data points in the stripe; $tol$, the
    tolerance for two numbers to be considered close enough.
Ensure: $nk1$, the number of knots in the knot vector of the approximating curve;
    $k1[1..nk1]$, the knot vector of the approximating curve; $v[1..ng]$, the adjusted
    parametric values.
    {$\triangleright$ remember if a parametric value has been set equal to a knot}
 1: for $i = 1$ to $ng$ do
 2:   $index[i] \leftarrow 0$
    {$\triangleright$ construct the knot vector of the approximating curve, and adjust the para-
    metric values}
 3: for $i = 0$ to $3$ do
 4:   $k1[i] \leftarrow 0$
 5: $nk1 \leftarrow 4$
 6: for $i = 1$ to $nk$ do
 7:   $min \leftarrow$ index j such that $v[j] < k[i] < v[j+1]$
 8:   if $(|v[min] - k[i]| \leq |v[min+1] - k[i]|)$ and $(|v[min] - k[i]| \leq tol)$ and
      $(index[min] = 0)$ then
 9:     $v[min] \leftarrow k[i]$
10:     $index[min] \leftarrow 1$
11:     $k1[nk1] \leftarrow k[i]$
12:     $nk1 \leftarrow nk1 + 1$
13:   if $(|v[min] - k[i]| > |v[min+1] - k[i]|)$ and $(|v[min] - k[i]| \leq tol)$ and
      $(index[min+1] = 0)$ then
14:     $v[min+1] \leftarrow k[i]$
15:     $index[min+1] \leftarrow 1$
16:     $k1[nk1] \leftarrow k[i]$
17:     $nk1 \leftarrow nk1 + 1$
18: $nk1 \leftarrow nk1 + 4$
19: for $i = 0$ to $3$ do
20:   $k1[nk1 + i] \leftarrow 1$
```

**Algorithm 5** APPROX-DENSE: approximate a dense set of data points with an interval curve

```
Require: $filename$, the file name of data file; $tol1$, the tolerances for knot construction; $tol2$, the tolerance for single-valued curve fitting.
Ensure: the resulting interval cubic B-spline curve.
    {$\triangleright$ input and make data positive}
 1: $(ng, [\vec{P}][1..ng]) \leftarrow$ READ-FILE($filename$)
 2: $(ng, [\vec{P}][1..ng]) \leftarrow$ TRANSLATE($ng, [\vec{P}][1..ng]$)
    {$\triangleright$ parametrize}
 3: $t[1..ng] \leftarrow$ PARAMETRIZE($ng, [\vec{P}][1..ng]$)
    {$\triangleright$ extract the centers of intervals}
 4: $\vec{p}[1..ng] \leftarrow center([\vec{P}][1..ng])$
    {$\triangleright$ construct the knot vector}
 5: $k[0..nc+3] \leftarrow$ ESTIMATE-KNOT($ng, \vec{p}[1..ng], t[1..ng], tol1$)
    {$\triangleright$ approximation by the least squares method}
 6: $flag1 \leftarrow 0$
 7: while 1 do
 8:   for $i = 1$ to $ng$ do
 9:     $index[i] \leftarrow 0$
      {$\triangleright$ single-valued approximation}
10:   $\vec{c}[1..nc] \leftarrow$ LEAST($ng, t[1..ng], \vec{p}[1..ng], nc, k[0..nc+3]$)
      {$\triangleright$ error calculation}
11:   $\vec{e}^l[1..ng], \vec{e}^u[1..ng] \leftarrow$ CAL-ERROR($ng, t[1..ng], [\vec{P}][1..ng], nc, \vec{c}[1..ng], k[0..nc+3]$)
12:   $flag2, index[1..ng] \leftarrow$ CHECK-ERROR($ng, [\vec{P}][1..ng], \vec{e}^l[1..ng], \vec{e}^u[1..ng], tol2$)
13:   if $(flag2 = 1)$ or $(flag1 = -1)$ then
14:     $break$
15:   else
16:     $flag1, k[0..nc+3] \leftarrow$ ADJUST-KNOT($ng, index[1..ng], t[1..ng]nc, k[0..nc+3]$)
17: if $flag1 = -1$ then
18:   $\vec{e}^l_{max}, \vec{e}^u_{max} \leftarrow$ FIND-MAX-ERROR($ng, \vec{e}^l[1..ng], \vec{e}^u[1..ng]$)
19:   message: tolerance can not be satisfied.
20:   message: current maximum errors are $\vec{e}^l_{max}, \vec{e}^u_{max}$
    {$\triangleright$ calculate the interval control points}
21: $[\vec{C}][1..nc] \leftarrow$ SET-CTRL-PTS($nc, k[0..nc+3], \vec{c}[1..nc], ng, t[1..ng], \vec{e}^l[1..ng], \vec{e}^u[1..ng]$)
```

**Algorithm 6** SET-CTRL-PTS: calculate the interval control points of an interval curve approximating a dense set of data points

**Require:** $nc$, the number of control points; $k[0..nc+3]$, the knot vector; $\vec{c}[1..nc]$, the control points of the singled-valued approximating curve; $ng$, the number of data points; $t[1..ng]$, the parametric values of data points; $\vec{e}^l[1..ng], \vec{e}^u[1..ng]$, the approximation errors of the single-valued approximating curve.

**Require:** $[\vec{C}][1..ng]$, the resulting interval control points.

```
    {▷ find the index of the span where a data point is located}
 1: $index[1..ng] \leftarrow$ FIND-SPAN($nc, k[0..nc+3], ng, t[1..ng]$)
    {▷ initialize the control points as intervals}
 2: for $i = 1$ to $nc$ do
 3:     $[\vec{C}][i] \leftarrow [\vec{c}[i], \vec{c}[i]]$
    {▷ initialize the lower and the upper enlargements}
 4: $\vec{d}^l[1] \leftarrow \vec{e}^l[1]$
 5: $\vec{d}^u[1] \leftarrow \vec{e}^u[1]$
 6: $\vec{d}^l[nc] \leftarrow \vec{e}^l[ng]$
 7: $\vec{d}^u[nc] \leftarrow \vec{e}^u[ng]$
 8: for $i = 2$ to $nc - 1$ do
 9:     $\vec{d}^l[i] \leftarrow 0$
10:     $\vec{d}^u[i] \leftarrow 0$
    {▷ set the lower and the upper enlargements}
11: for $i = 2$ to $ng - 1$ do
12:     for $j = 0$ to $3$ do
13:         $ctrl \leftarrow index[i] + j$
14:         if $component(\vec{d}^l[ctrl]) < component(\vec{e}^l[i])$ then
15:             $component(\vec{d}^l[ctrl]) \leftarrow component(\vec{e}^l[i])$
16:         if $component(\vec{d}^u[ctrl]) < component(\vec{e}^u[i])$ then
17:             $component(\vec{d}^u[ctrl]) \leftarrow component(\vec{e}^u[i])$
    {▷ set the interval control points}
18: for $i = 1$ to $nc$ do
19:     $[\vec{T}]_1 \leftarrow [\vec{C}][i] - \vec{d}^l[i]$
20:     $[\vec{T}]_2 \leftarrow [\vec{C}][i] + \vec{d}^u[i]$
21:     $[\vec{C}][i] \leftarrow [lower([\vec{T}]_1), upper([\vec{T}]_2)]$
```

# Bibliography

[1] S. L. Abrams, L. Bardis, C. Chryssostomidis, N. M. Patrikalakis, S. T. Tuohy, F.-E. Wolter, and J. Zhou. The geometric modeling and interrogation system Praxiteles. *Journal of Ship Production*, 11(2):116–131, May 1995.

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

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

[4] C. Bliek. *Computer Methods for Design Automation.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, July 1992.

[5] G. Dahlquist and A. Björck. *Numerical Methods.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1974.

[6] C. De Boor. On calculating with B-splines. *Journal of Approximation Theory*, 6:50–62, 1972.

[7] P. M. do Carmo. *Differential Geometry of Curves and Surfaces.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1976.

[8] T. Duff. Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry. *ACM Computer Graphics*, 26(2):131–138, July 1992.

[9] W. Enger. Interval Ray Tracing - A divide and conquer strategy for realistic computer graphics. *The Visual Computer*, 9(2):91–104, November 1992.

[10] G. Farin, G. Rein, N. Sapidis, and A. J. Worsey. Fairing cubic B-spline curves. *Computer Aided Geometric Design*, 4(1-2):91–103, July 1987.

[11] S. J. Gordon. 4DI: A real-time three-dimensional imager. In *Proceedings of the SPIE Conference on Industrial Inspection*, volume 2348, pages 221–226, Boston, MA, November 1994.

[12] G. D. Hager. Constraint solving methods and sensor-based decision making. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pages 1662–1667. IEEE, 1992.

[13] E. Hansen, editor. *Topics in Interval Analysis.* Oxford University Press, Ely House, London, UK, 1969.

[14] C. M. Hoffmann. The problems of accuracy and robustness in geometric computation. *Computer*, 22(3):31–41, March 1989.

[15] A. Hoover, D. Goldgof, and K. W. Bowyer. Extracting a valid boundary representation from a segmented range image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(9):920–924, September 1995.

[16] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design.* A. K. Peters, Wellesley, MA, 1993. Translated by L. L. Schumaker.

[17] C.-Y. Hu. Robust Algorithms for Sculptured Shape Visualization. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, July 1993.

[18] C.-Y. Hu. *Towards Robust Interval Solid Modeling for Curved Objects.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, May 1995.

[19] C.-Y. Hu, T. Maekawa, N. M. Patrikalakis, and X. Ye. Robust interval algorithm for surface intersections. *Computer Aided Design*, 1997. To appear.

[20] C.-Y. Hu, T. Maekawa, E. C. Sherbrooke, and N. M. Patrikalakis. Robust interval algorithm for curve intersections. *Computer Aided Design*, 28(6/7):495–506, June/July 1996.

[21] C.-Y. Hu, N. M. Patrikalakis, and X. Ye. Robust interval solid modeling: Part I, Representations. *Computer Aided Design*, 28(10):807–817, October 1996.

[22] C.-Y. Hu, N. M. Patrikalakis, and X. Ye. Robust interval solid modeling: Part II, Boundary evaluation. *Computer Aided Design*, 28(10):819–830, October 1996.

[23] R. B. Kearfott. Preconditioners for the interval Gauss-Seidel method. *SIAM Journal of Numerical Analysis*, 27(3):804–822, June 1990.

[24] R. Lohner. Enclosing the solutions of ordinary initial and boundary value problems. In Kaucher *et al.*, editor, *Computer Arithmetic, Scientific Computation and Program Languages*, pages 255–286. B. G. Teubner, Stuttgart, 1987.

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

[26] M. J. Milroy, C. Bradley, G. W. Vickers, and D. J. Weir. $G^1$ continuity of B-spline surface patches in reverse engineering. *Computer Aided Design*, 27(6):471–478, June 1995.

[27] R. E. Moore. *Interval Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1966.

[28] S. P. Mudur and P. A. Koparkar. Interval methods for processing geometric objects. *IEEE Computer Graphics and Applications*, 4(2):7–17, February 1984.

[29] J. R. Munkres. *Topology: a First Course.* Prentice-Hall, Englewood Cliffs, NJ, 1975.

[30] Numerical Algorithms Group, Oxford, England. *NAG Fortran Library Introductory Guide*, Mark 14 edition, 1990.

[31] D. J. Orser and M. Roche. The extraction of topographic features in support of autonomous underwater vehicle navigation. In *Proc. Int. Symp. on Unmanned Untethered Submersible Technology*, pages 502–514, Durham, NH, June 1987. Marine Systems Engineering Laboratory, University of New Hampshire. Vol. 2.

[32] H. Ratschek and J. Rokne. *Computer Methods for the Range of Functions.* Ellis Horwood, Chichester, England, 1984.

[33] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design.* Prentice–Hall, Englewood Cliffs, NJ, 1991.

[34] N. Sapidis. Algorithms for locally fairing B-spline curves. Master's thesis, University of Utah, June 1987.

[35] N. S. Sapidis and P. J. Besl. Direct construction of polynomial surface from dense range images through region growing. *ACM Transactions on Graphics*, 14(2):171–200, April 1995.

[36] I. J. Schoenberg and A. Whitney. On Polya frequency functions III. *Transactions of the American Mathematical Society*, 74:246–259, 1953.

[37] T. W. Sederberg and D. B. Buehler. Offsets of polynomial Bézier curves: Hermite approximation with error bounds. In T. Lyche and L. L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design*, volume II, pages 549–558. Academic Press, 1992.

[38] T. W. Sederberg and R. T. Farouki. Approximation by interval Bézier curves. *IEEE Computer Graphics and Applications*, 15(2):87–95, September 1992.

[39] J. M. Snyder. Interval analysis for computer graphics. *ACM Computer Graphics*, 26(2):121–130, July 1992.

[40] M. Soucy and D. Laurendeau. A general surface approach to the integration of a set of range views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(4):344–358, April 1995.

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

[42] D. Toth. On ray tracing parametric surfaces. *ACM Computer Graphics*, 19(3):171–179, July 1985.

[43] S. T. Tuohy and L. Bardis. Low degree approximation of high degree B-spline surfaces. *Engineering with Computers*, 9(4):198–209, Winter 1993.

[44] S. T. Tuohy, T. Maekawa, and N. M. Patrikalakis. Interrogation of geophysical maps with uncertainty for AUV micro-navigation. In *Engineering in Harmony with the Ocean, Proceedings of Oceans '93, Victoria, Canada.* IEEE Oceanic Engineering Society, October 1993.

[45] S. T. Tuohy, T. Maekawa, G. Shen, and N. M. Patrikalakis. Approximation of measured data with interval splines. Design Laboratory Memorandum 95-18, MIT, Department of Ocean Engineering, Cambridge, MA, 1995. Revised, October 15, 1996, and April 11, 1997. Submitted for publication.

[46] S. T. Tuohy and N. M. Patrikalakis. Nonlinear data representation for ocean exploration and visualization. *Journal of Visualization and Computer Animation*, 7(3):125–139, July–September 1996.

[47] S. T. Tuohy, J. W. Yoon, and N. M. Patrikalakis. Reliable interrogation of 3-D non-linear geophysical databases. In J. A. Vince and R. A. Earnshaw, editors, *Computer Graphics: Developments in Virtual Environments, Proceedings of CG International '95*, pages 327–341, Leeds, UK, June 1995. London, Academic Press.