# DEFINING AND FAIRING CURVES AND SURFACES USING AN ITERATIVE B-SPLINE ERROR TECHNIQUE

by

Hui Ping Wang

B.Eng., Beijing Institute of Telecommunications, Beijing, 1984

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE
in the Department
of
Electrical Engineering



We accept this thesis as conforming
to the required standard

[redacted]

Supervisor Dr. G. W. Vickers

[redacted]

Dr. A. Zielinski and Dr. A. K. S. Bhat

[redacted]

Dr. V. K. Bhargava and Dr. D. E. Hewgill

[redacted]

Dr. H. A. Muller

## Abstract

Supervisor: Professor G. W. Vickers

A novel approach for defining and fairing B-spline curves and surfaces was developed in this work. The approach uses the error between the B-spline approximation solution and the data vertices to iteratively adjust the control vertices and hence the new curve or surface.

In the first part of the work, the errors are added iteratively to the original data vertices to generate a new set of B-spline interpolation control vertices. Consequently, the B-spline interpolation problems can be solved by the B-spline approximation method. In addition, the errors can be easily controlled to modify the control vertices at any region of the specified area. This ensures that the resulting curves or surfaces meet design requirements, especially the criterion for smoothness.

In the second part of the work, an iterative fairing algorithm, based on checking the nature of the control polygon or polyhedron, is developed. For curves, the sign of the cross-product at the control polygon vertices is used as a tool to detect the fairness of the original data. For surfaces, the unfair regions can be identified by irregular distributions of the B-spline error sign matrix.

This work was done as a fourth order (degree three) B-spline, because generally, in practical applications, only continuities up to a second degree (curvature) are required. This approach has found a wide application in representing curves and surfaces, which range from analytical functions to human face to kayak hulls.

Examiners:

[redacted]

Supervisor Dr. G. W. Vickers

[redacted]

Dr. A. Zielinski

[redacted]

Dr. A. K. S. Bhat

[redacted]

Dr. V. K. Bhargava

[redacted]

Dr. D. E. Hewgill

[redacted]

Dr. H. A. Muller

# Contents


**i Abstract** ii

**ii Contents** iv

**iii List of Figures** vii

**iv List of Tables** x

**v Acknowledgements** xi

**vi Dedications** xii

**1 Introduction** 1

1.1 Introduction . . . . . 1

1.2 Literature Survey in Curve and Surface Definition . . . . . 1

1.3 Literature Survey in Curve and Surface Fairing . . . . . 3

1.4 Problem Definition and Current Work . . . . . 5

1.5 Thesis Plan . . . . . 6

**2 B-spline Approximation** 7

2.1 Introduction . . . . . 7

2.2 Definition of the B-spline Basis Functions . . . . . 7

2.3 Properties of the B-spline Basis Functions . . . . . 8

2.3.1 Local Support . . . . . 8

2.3.2 "Sum" Equal to 1 . . . . . 9

2.3.3 Positive . . . . . 9

2.3.4 Recursive Computation . . . . . 10

2.4 B-spline Curve Approximation . . . . . 12
2.5 Knot Vector and Order . . . . . 13
2.6 Recursive Algorithm . . . . . 15
2.7 Examples . . . . . 16
2.8 B-spline Surface Approximation . . . . . 18

**3 Translation from B-spline Approximation to B-spline Interpolation 25**
3.1 Introduction . . . . . 25
3.2 Error Definition . . . . . 25
3.3 Updating Control Polygon for Curve Fitting . . . . . 27
3.4 Iterative Algorithm . . . . . 27
3.5 Curve Fitting Illustration . . . . . 28
3.6 Updating Control Polyhedron for Surface Fitting . . . . . 28
3.7 Surface Fitting Illustration . . . . . 29
3.8 Summary . . . . . 30

**4 Fairing of Compound Curvature Curves 40**
4.1 Introduction . . . . . 40
4.2 Development of Smoothing Technique . . . . . 41
4.2.1 Detecting Unfair Vertices . . . . . 41
4.2.2 Determining the Range of Adjustment . . . . . 42
4.2.3 Adjusting Unfair Vertices . . . . . 43
4.2.4 Other Remarks . . . . . 44
4.2.5 B-spline Approximation Smoothing Procedure . . . 45
4.2.6 B-spline Interpolation Smoothing Procedure . . . . . 45
4.3 Iterative Smoothing Algorithm . . . . . 46
4.4 Curve Smoothing Illustration . . . . . 46

**5 Fairing of Compound Curvature Surfaces 67**
5.1 Background Information . . . . . 67
5.1.1 The Definition of the Error Sign Matrix . . . . . 67

5.1.2 The Definition of the Sub-Block . . . . . 69
5.1.3 The Definition of Non-Convex Sub-Block . . . . . 70
5.2 The Classification of Matrix Sign Patterns . . . . . 70
5.3 Recursive Sub-Block Definition . . . . . 71
5.4 Local Modifications . . . . . 75
5.4.1 Range Location . . . . . 75
5.4.2 Adjusting Unfair Vertices . . . . . 76
5.4.3 Relevant Algorithms . . . . . 77
5.5 B-spline Approximation Smoothing Procedure . . . . . 80
5.6 B-spline Interpolation Smoothing Procedure . . . . . 81
5.7 Other Remarks . . . . . 81

**6 Conclusions and Further Work** **92**
6.1 Conclusions . . . . . 92
6.2 Further Work . . . . . 93

**7 Bibliography** **94**

**8 Appendices** **98**
A Hermite Interpolation and Coons Patch . . . . . 98
A.1 Hermite Interpolation . . . . . 98
A.2 Coons Patches . . . . . 98
B Bezier Curves and Surfaces . . . . . 101
B.1 Bezier Curves . . . . . 101
B.2 Bezier Surfaces . . . . . 101
C Divided Difference . . . . . 102
D List of Symbols . . . . . 103

# List of Figures


2.1 Fourth order B-spline basis functions. . . . . 20
2.2 B-spline curves with different orders. . . . . 21
2.3 Multiple vertex effect: B-spline curves with different orders, defined on different number of the control vertices. . . . . 22
2.4 Multiple vertex effect: B-spline curves with different orders, defined on the same number of control vertices. . . . . 23
2.5 B-spline curves with local modification. . . . . 24

3.1 B-spline approximation and interpolation to four points. . . 31
3.2 An illustration of the iterative algorithm for equally spaced $x$ values. . . . . 32
3.3 An illustration of the iterative algorithm for unequally spaced $x$ values. . . . . 33
3.4 B-spline curves defined by the data vertices from a *sine* function. . . . . 34
3.5 B-spline interpolation to five arbitrary data vertices. . . . . 35
3.6 B-spline surface interpolation (five iterations) to $11 \times 9$ data vertices from a known analytic equation. . . . . 36
3.7 B-spline surface approximation defined on $6 \times 6$ arbitrary control polyhedron vertices. . . . . 37
3.8 B-spline surface interpolation (ten iterations) to the same control polyhedron as given in Figure 3.7. . . . . 38
3.9 An intermediate B-spline surface fitted to $80 \times 80$ data points obtained by stereo photogrammetry. . . . . 39

4.1 A typical concave down polygon with its B-spline approximation. . . . . 48
4.2 A concave down polygon with inflection point caused by moving down vertex $P_2$. . . . . 49
4.3 A concave down polygon with inflection point caused by moving up vertices $P_1$ and $P_3$. . . . . 50
4.4 The notation illustration. . . . . 51
4.5 A faired curve to the control polygon vertices given in Figure 4.3 with $rate1 = 0.6$, and $rate2 = 0.3$. . . . . 52
4.6 A faired curve to the control polygon vertices given in Figure 4.3 with $rate2 \gg rate1$. . . . . 53
4.7 A faired curve to the control polygon vertices given in Figure 4.3 with $rate2 \ll rate1$. . . . . 54
4.8 A parabola with perturbation and its corrected curve. . . . 55
4.9 Original and improved sheer line of a single kayak hull ($x-y$ view). . . . . 56
4.10 Original and improved sheer line of a double kayak hull ($x-y$ view). . . . . 57
4.11 Flowchart of the curve fairing program. . . . . 58
4.12 Flowchart detailing subroutine (smooth_procedure1). . . . . 59
4.13 Flowchart detailing subroutine (smooth_procedure2). . . . . 60
4.14 Flowchart detailing subroutine (sign_cross_product). . . . . 61
4.15 Flowchart detailing subroutine (extreme_position). . . . . 62
4.16 Flowchart detailing subroutine (adjusting_range). . . . . 63
4.17 Flowchart detailing subroutine (adjust). . . . . 64
4.18 Flowchart detailing subroutine (add_error). . . . . 65
4.19 Flowchart detailing subroutine (swap_position). . . . . 66

5.1 Perspective views of the control polyhedrons before and after the corrections. . . . . 86

5.2 Perspective views of the control polyhedrons before and after the corrections. . . . . 87
5.3 $(y-z)$ views of the control polyhedrons before and after the corrections. . . . . 88
5.4 $(y-z)$ views of the kayak hull before and after the corrections. . . . . 89
5.5 Perspective view of a single kayak hull. . . . . 90
5.6 Perspective view of a double kayak hull. . . . . 91

# List of Tables


5.1 Error sign matrix with $S_{min} = 0$, $S_{mid} = -1$ and $S_{max} = 1$. . 68

5.2 Error sign matrix with $S_{min} = -1$, $S_{mid} = -3$ and $S_{max} = 1$. 69

5.3 Error sign matrix with $S_{min} = -2$, $S_{mid} = -3$ and $S_{max} = 1$. 69

5.4 Error sign matrix with non-convex subblock. . . . . 71

5.5 Error sign matrix with non-convex sub-block. . . . . 71

5.6 Error sign matrix before separating different sub-blocks. . . 72

5.7 Error sign matrix after separating different sub-blocks. . . . 72

5.8 Error sign matrix with irregular distribution. . . . . 82

5.9 Error sign matrix with irregular distribution. . . . . 83

5.10 Error sign matrix after correction is made to Table 5.9. . . 83

5.11 Error sign matrix with irregular distribution. . . . . 83

5.12 Error sign matrix after corrections are made to Table 5.11. 84

5.13 Error sign matrix with irregular distribution. . . . . 85

5.14 Error sign matrix after corrections are made to Table 5.13. 85

## ACKNOWLEDGEMENTS

The author would like to thank her supervisor, Dr. Geoffrey W. Vickers for his guidance during the course of this work and for his helpful advice in the preparation of this manuscript.

The author would also like to express her appreciation to others who helped:

To Dr. Denton E. Hewgill for his interest and helpful comments in the early stages of the thesis.

To Dr. Sanjeev Bedi, Ken K. Law, Minh H. Ly and Mile Erlic for their support and for their many helpful suggestions.

The assistance from faculty and staff in the Department of Electrical Engineering is also greatly appreciated.

Finally, the author would like to thank the Natural Sciences and Engineering Research Council of Canada and the University of Victoria for providing financial support in the form of a grant and a fellowship.

To
my dear mother, father
and sister.

# Chapter 1

# Introduction

## 1.1 Introduction

Mathematical models of curves or surfaces are frequently required in computer-aided design and manufacture applications. The modeling requires the construction of compact representations for given sets of data points. The representations need to be flexible and be easily manipulated. If the data points are accurate, the model can be constructed to pass through the points. This process is called *interpolation.* If the data points are in error, the model can be constructed to pass among the points. This process is called *approximation.* The choice of a particular model is dictated by the requirements of the application. The selected model may need to be modified until it meets some design constraint on smoothness of form. This technique is called *fairing.*

The next two sections give a brief review of the most commonly used curve and surface generation methods. A number of existing fairing techniques are also discussed.

## 1.2 Literature Survey in Curve and Surface Definition

There are several methods of defining curves or surfaces that have found wide application in computer-aided design and manufacture. Early ap-

proaches were the Hermite curve interpolation and the Coons patch methods. Cubic Hermite interpolation uses a set of basis functions, which is often referred to as the Hermite basis functions, to interpolate given position and slope conditions. The approach gives a weighting to the position and slope constraints that vary across the range. A difficulty with the method is that it sometimes generates unexpected oscillations, particularly with large number of points.

If the Hermite basis functions are used to blend four boundary curves, the resulting patch is called a bicubic Coons patch. One of the great merits of the Coons patch is that it does not require a particular form of boundary curve. In practice, however, the most commonly used boundary curve is the piecewise parametric cubic polynomial, which has sufficient flexibility to meet both position and slope constraints at its ends.

Although the bicubic Coons patch has appeared in a number of practical implementations such as the POLYSURF [15] and NMG (Numerical Master Geometry) [29] systems, there are major disadvantages. One difficulty is that the position, tangent and twist vectors, all generally of different orders of magnitude, must be specified to describe the patch. The user does not ordinarily have an intuitive feel for the behavior of twist vectors and the effect they have on the surface. The mathematical expression of Hermite basis functions and Coons patches are given in Appendix A.

Bezier curves and surfaces overcome these difficulties by using sets of data points [14] to form a control polygon or polyhedron. The shape of the curve or surface is controlled by the polygon or polyhedron but does not, however, bear any close resemblance to the points. This is a useful tool for interactive design of curves or surfaces by adjustment of the control points. Bernstein polynomials form the basis functions for Bezier curves and surfaces.

There are other shortcomings, however, with Bezier curves or surfaces. One is that moving a single point of the control vertices will affect the

entire Bezier curve or surface because of the global nature of the Bernstein polynomials. Another is that there are only sixteen conditions to specify the bicubic Bezier patch and hence the flexibility of the shape description is limited. If multiple patches are used, the degree of freedom per patch is further reduced as position and slope matching at the patch boundaries is required. Furthermore, the order increases proportionally to the number of vertices used. The formulations for Bezier curves and surfaces are given in Appendix B.

Recently, B-spline curves or surfaces have been used to overcome some of these difficulties. The B-spline method [8,27] is based on the same kind of control points as the Bezier technique; but instead of using Bernstein polynomials as the basis functions, it employs B-spline basis functions (defined and discussed in Chapter 2). The effect of each control point is limited to a proportion of the span. This characteristic gives the better facility to make alterations. In addition, the degree of the basis function is independent of the number of the control points, so that a low degree B-spline curve or surface can represent a complex shape. An efficient algorithm for computing B-spline curve and surface approximations is given by de Boor [8].

## 1.3 Literature Survey in Curve and Surface Fairing

To smooth or fair curves and surfaces, it is first necessary to detect unsuitable regions and then to develop methods to remove them while at the same time not unduly changing the shape. Recently, a number of authors have investigated this problem [26,10,28,5].

Renz [26] uses the first and the second divided differences to detect the data smoothness. Local corrections are performed on the first and the second difference curves. The improved data points are then computed by integration.

Poeschl [24] uses isophotes (lines of equal light intensity) to display irregularities of a surface's first and second derivatives as well as irregularities of its Gaussian curvature. This method reveals discontinuities of the surface's derivatives at the common boundaries of surface patches. It can give good results for curved surfaces but the method does not work well for flat parts of a surface.

Hoschek [17] has shown that a polarity with respect to the complex unit circle or the complex unit sphere can be used for detecting the undesirable points on Bezier curves or surfaces. A point on a surface with vanishing Gaussian curvature leads to singularities of the polar image (cusps, edges of regression). These singularities can be easily seen on a graphic display. The deviation can be removed interactively using the Bezier approach.

Recently, more attention has focused on the use of Gaussian curvature as a mean of detecting unfairness in surface [10,28,5]. The Gaussian or total curvature at a point on a surface is the product of the maximum and minimum curvatures, and indicates whether the surface is locally elliptic, hyperbolic, or parabolic (Gaussian curvature positive, negative, or zero).

Dill [10] used contours of Gaussian and average curvature combined with color raster graphics to examine the fairness of a surface. He applied this technique to toroidal and catenoidal test surfaces and to automobile hood and fender surfaces. The results clearly show surface curvature discontinuities and hence provide an indication of surface fairness.

For many applications, it is the principal curvature that is of primary interest. When a surface is to be machined by a spherical cutter, for example, it is important to establish that the cutter radius is smaller than the smallest concave radius of curvature of the surface if gouging is to be avoided. For this reason, Beck [5] uses color-coded maps of the principal curvatures as a tool to indicate unfair spots.

## 1.4 Problem Definition and Current Work

B-spline curves and surfaces are suitable for computer-aided design and manufacture applications. However, a difficulty arises if the curve or surface is required to pass precisely through a series of predefined data points. Usually, for B-spline curve interpolation, a new set of control polygon vertices can be obtained by solving a number of linear equations through a band matrix solver [8]. For B-spline surface interpolation, it often requires matrix inversion [28], which is computationally inefficient. In addition, if the data base is large, the inversion may give erroneous results caused by truncation and round-off errors. In order to apply more effectively the B-spline method to the curve and surface interpolation problems, that is, not only allow it to be suitable for shape design but also suitable for precise shape representation, a new effective B-spline interpolation approach needs to be developed.

Although several detecting methods have been developed, their correcting methods have almost always been based on manual input. This work is time consuming and requires a high level of skill. In addition, there is no efficient method available that is particularly developed to fair or smooth the B-spline curves or surfaces.

In the first part of this work (Chapter 3), a novel and efficient iterative algorithm for defining B-spline interpolation curves and surfaces is developed. The difference or error between a B-spline approximation solution and the actual data vertices is used to cumulatively modify the control vertices in order to generate a B-spline interpolation solution. In the second part of the work (Chapter 4 and 5), an iterative fairing algorithm for detecting and smoothing B-spline curves and surfaces has been developed. In the case of B-spline curves, the lack of smoothness is determined from the cross products of the position vectors of the control polygon for a convex curve. The unfair region is automatically detected and appropriately ad-

justed. In the case of B-spline surfaces, the unfair spots are identified by the irregularities of the B-spline error sign matrix. The decision can then be made on whether or not to correct these unfair regions. If the decision is made to make a correction, then the corresponding modifications are iteratively made. The unique feature of this approach is that detection and smoothing are done simply on the control polygons or polyhedrons and not on the curves or surfaces themselves.

## 1.5 Thesis Plan

A definition and listing of the main properties of B-splines together with the recursive algorithm are given in Chapter 2. In Chapter 3, an efficient algorithm for B-spline interpolation is presented. The algorithm uses the error between the original data vertices and the corresponding B-spline approximation curve values to accumulatively adjust the control polygon vertices. In Chapter 4, an iterative method to identify and modify unfair spots on a B-spline control polygon is developed. The unfairness is detected by using the cross products of the position vectors of the control polygon. In Chapter 5, a similar smoothing procedure is developed for B-spline surfaces. The irregularities of the parametric B-spline error sign matrices are used as the indicators to the unfair spots on the control polyhedron vertices. By iteratively determining these unfair regions and making corresponding modifications, the faired control polyhedron vertices can be obtained. In this way, the resulting B-spline surface satisfies the desired criterion of smoothness. In Chapter 6, the conclusions and suggestions for further research work are outlined.

# Chapter 2

# B-spline Approximation

## 2.1 Introduction

The B-spline method is a generalization of the Bezier technique. There are a number of unique features (such as local support) that make it very attractive to the designer. In this chapter, a brief review is given on the definition of the B-spline basis functions, its main characteristics, recursive computational algorithm, and B-spline curve and surface approximation problems [8].

## 2.2 Definition of the B-spline Basis Functions

The $k^{th}$ order (degree $k-1$) B-spline can be defined as an appropriately scaled $k^{th}$ divided differences of the truncated power function [8] (see Appendix C for the definition of divided differences). A useful notation, defining the truncated power function $t_+^k$ for any variable $t$ and positive integer $k$, is given by

$$t_+^k = \begin{cases} t^k & \text{if } t \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

A knot vector, $\mathbf{x}$, is defined as a vector of real numbers, called *knots*, in nondecreasing order, that is, $\mathbf{x} = [x_0, x_1, \cdots, x_q]$, such that $x_{i-1} \leq x_i$, $i = 1, \cdots, q$.

The $i^{th}$ normalized B-spline basis function of order $k$ for the knot vector $\mathbf{x}$ is denoted by $B_{i,k}$ and is defined by the rule [8]

$$B_{i,k}(t) = (x_{i+k} - x_i)[x_i, \cdots, x_{i+k}](\cdot - t)_+^{k-1}. \tag{2.2}$$

The "placeholder" notation employed here is to indicate that the $k^{th}$ divided difference of the function $(x-t)_+^{k-1}$ of the two variables $x$ and $t$ is to be taken by fixing $t$ and considering $(x-t)_+^{k-1}$ as a function of $x$ alone. The resulting number depends, of course, on the particular value of $t$ chosen, i.e., the resulting number varies as $t$ is varied, and so the function $B_{i,k}$ of $t$ is eventually obtained.

## 2.3 Properties of the B-spline Basis Functions

### 2.3.1 Local Support

From the definition (2.2) above, it can be noted that $B_{i,k}(t)$ has small support, i.e.,

$$B_{i,k}(t) = 0 \quad \text{for } t \notin [x_i, x_{i+k}]. \tag{2.3}$$

For, if $t \notin [x_i, x_{i+k}]$, then $g(x) = (x-t)_+^{k-1}$ is a polynomial of degree $< k$ on $[x_i, x_{i+k}]$ and therefore, $[x_i, \cdots, x_{i+k}]g = 0$ (see Appendex C, Theorem C.1).

It follows that only $k$ B-splines have any particular interval $[x_j, x_{j+1}]$ in their support, i.e., of all the B-splines of order $k$ for the knot vector $\mathbf{x}$, only the $k$ B-splines $B_{j-k+1,k}$, $B_{j-k+2,k}$, $\cdots$, $B_{j,k}$ might be nonzero on the interval $[x_j, x_{j+1}]$.

### 2.3.2 "Sum" Equal to 1

Definition (2.2) can be rewritten as (see Appendex C, Theorem C.1):

$$B_{i,k}(t) = [x_{i+1}, \cdots, x_{i+k}](\cdot - t)_+^{k-1} - [x_i, \cdots, x_{i+k-1}](\cdot - t)_+^{k-1}. \qquad (2.4)$$

Now take $t$ in the open interval $(x_j, x_{j+1})$, by the small support of the B-spline basis function, the results are:

$$\begin{aligned}
\sum_i B_{i,k}(t) &= \sum_{i=j+1-k}^{j} B_{i,k}(t) \\
&= \sum_{i=j+1-k}^{j} [x_{i+1}, \cdots, x_{i+k}](\cdot - t)_+^{k-1} - \sum_{i=j+1-k}^{j} [x_i, \cdots, x_{i+k-1}](\cdot - t)_+^{k-1} \\
&= [x_{j+1}, \cdots, x_{j+k}](\cdot - t)_+^{k-1} - [x_{j+1-k}, \cdots, x_j](\cdot - t)_+^{k-1} \\
&= 1.
\end{aligned}$$

The last equality comes about as follows. For $t$ in the open interval $(x_j, x_{j+1})$, the function $g(x) = (x - t)_+^{k-1}$ is a polynomial of degree $k-1$ on $[x_{j+1}, x_{j+k}]$, so that, by Theorem C.1, $[x_{j+1}, \cdots, x_{j+k}]g = 1$, while $g = 0$ on $[x_{j+1-k}, x_j]$, hence $[x_{j+1-k}, \cdots, x_j]g = 0$. Thus it can be concluded that

$$\sum_i B_{i,k}(t) = \sum_{i=r+1-k}^{s-1} B_{i,k}(t) = 1, \qquad (2.5)$$

for all $x_r < t < x_s$.

### 2.3.3 Positive

Another property of the B-spline basis functions is that $B_{i,k}(t)$ is positive on its support, i.e., $B_{i,k}(t) > 0$, for $x_i < t < x_{i+k}$. This is proved in the next section.

### 2.3.4 Recursive Computation

The direct evaluation of the B-spline $B_{i,k}(t)$ from its definition (2.2) as a divided difference has to be carried out with caution because of the possible loss of significance during the computation of the various difference quotients. Also, special provisions have to be made in the case of repeated or multiple knots. To avoid these difficulties, de Boor [8] developed an algorithm that uses a recurrence relation of the B-spline. The recurrence relation can be obtained by applying Leibniz' formula (see Appendix C) for the $k^{th}$ divided difference of a product as follows

$$(x-t)_+^{k-1} = (x-t)(x-t)_+^{k-2}.$$

This gives

$$\begin{aligned}[x_i,\cdots,x_{i+k}](\cdot-t)_+^{k-1} &= (x_i-t)[x_i,\cdots,x_{i+k}](\cdot-t)_+^{k-2} + \\ &\quad 1[x_{i+1},\cdots,x_{i+k}](\cdot-t)_+^{k-2}, \qquad (2.6)\end{aligned}$$

since

$$[x_i](\cdot-t) = (x_i-t),$$

$$[x_i,x_{i+k}](\cdot-t) = 1,$$

and

$$[x_i,\cdots,x_j](\cdot-t) = 0 \qquad \text{for } j > i+1.$$

As a result

$$(x_i-t)[x_i,\cdots,x_{i+k}] = \frac{x_i-t}{x_{i+k}-x_i}\{[x_{i+1},\cdots,x_{i+k}] - [x_i,\cdots,x_{i+k-1}]\},$$

equation (2.6) can also be stated as

$$[x_i, \cdots, x_{i+k}](\cdot - t)_+^{k-1} = \frac{t - x_i}{x_{i+k} - x_i}[x_i, \cdots, x_{i+k-1}](\cdot - t)_+^{k-2} + \frac{x_{i+k} - t}{x_{i+k} - x_i}[x_{i+1}, \cdots, x_{i+k}](\cdot - t)_+^{k-2}. \quad (2.7)$$

In terms of the (normalized) B-spline, (2.7) reads

$$\frac{B_{i,k}(t)}{x_{i+k} - x_i} = \frac{t - x_i}{x_{i+k} - x_i}\frac{B_{i,k-1}(t)}{x_{i+k-1} - x_i} + \frac{x_{i+k} - t}{x_{i+k} - x_i}\frac{B_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}}. \quad (2.8)$$

Note that the weight factors in this formula (2.8) always adds up to 1

$$\frac{t - x_i}{x_{i+k} - x_i} + \frac{x_{i+k} - t}{x_{i+k} - x_i} = 1.$$

Also, both factors are positive for $x_i < t < x_{i+k}$. Therefore, if it is known that $B_{j,k-1}(t) > 0$ for $x_j < t < x_{j+k-1}$, all $j$, then, it can be concluded from (2.8) that $B_{i,k}(t) > 0$ for $x_i < t < x_{i+k}$. This proves that the B-splines are positive on their support.

By induction on $k$, starting with the evident fact

$$B_{i,1}(t) = \begin{cases} 1 & \text{if } x_i \leq t < x_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \quad (2.9)$$

The recurrence relation (2.8) can also be written as

$$B_{i,k}(t) = \frac{t - x_i}{x_{i+k-1} - x_i}B_{i,k-1}(t) + \frac{x_{i+k} - t}{x_{i+k} - x_{i+1}}B_{i+1,k-1}(t), \quad (2.10)$$

where $x_i$ is the element of the knot vector and $t$ is the local parameter.

In words, (2.10) means that the B-spline of order $k$ in the $i^{th}$ span is the weighted average of the B-splines of order $k-1$ on the $i^{th}$ and $(i+1)$st span, each weight being the ratio of the distance between the parameter and the

end knot to the length of the $k-1$ spans. If the knot vector is chosen from a set of integers, then the so-called uniform B-spline basis will be generated using (2.9) and (2.10). Choosing the elements (knots) of the knot vector from real numbers, still subject to $x_i \leq x_{i+1}$, will generate the non-uniform B-spline basis. Figure 2.1 shows the fourth order uniform B-spline basis functions. More discussion about the knot vector is given in Section 2.5.

## 2.4 B-spline Curve Approximation

To apply the B-spline basis function to the problem of curve approximation, let $P_i$ $(i = 0, 1, \cdots, n)$ be $n+1$ ordered points in two-dimensional (or three-dimensional) space. The open polygon formed by joining successive points is referred to as the control polygon.

The parametric B-spline curve approximation of order $k$ to the control polygon is defined [8] as

$$R(t) = \sum_{i=0}^{n} P_i B_{i,k}(t), \tag{2.11}$$

where $0 \leq t \leq t_{max}$, and

$R(t)$ is the parametric B-spline curve approximation, which has two or three components for two or three dimensional curves respectively,

$P_i$ is the $i^{th}$ polygon vertex,

$B_{i,k}(t)$ is the normalized B-spline basis function,

$t$ is the local parameter,

$t_{max}$ is the maximum range of $t$, which is determined by the order $k$ and the total number of the control polygon vertices.

From the property (Section 2.2.1) of local support, it is noted that the computation of $B_{i,k}(t)$ involves all the knots from $x_i$ to $x_{i+k}$ but no others, that is

$$B_{i,k}(t) > 0 \qquad \text{for } x_i \leq t < x_{i+k},$$
$$B_{i,k}(t) = 0 \qquad \text{otherwise.}$$

These relations show that to evaluate the function $R(t)$ at $t \in [x_j, x_{j+1}]$, it is necessary to calculate only $k$ numbers of $B_{i,k}(t)$, $i = j-k+1, \cdots, j$. Then $R(t)$ is given by

$$R(t) = \sum_{i=j-k+1}^{j} P_i B_{i,k}(t). \tag{2.12}$$

Equation (2.12) shows that, as a consequence of the local support of a B-spline basis function, B-spline approximation is a local approximation scheme. The summation in (2.12) involves, at most, $k$ successive nonzero terms. Therefore, a local perturbation in the control polygon produces only a local perturbation in the B-spline approximation. This stands in contrast to the Bernstein approximation, which is a global approximation scheme.

Since the B-spline weighting functions (basis functions) $B_{i,k}(t)$ in (2.11) are non-negative and sum to 1, each point on a B-spline curve is a convex combination of the control vertices. Specifically, for a B-spline curve of order $k$ (degree $k-1$), a given point lies within the convex hull of the neighboring $k$ vertices.

## 2.5 Knot Vector and Order

It is known that the parameter range for any order of a Bezier curve [14] is arbitrarily chosen to be $0 \leq t \leq 1$. For a B-spline curve, it is necessary to deviate from this convention and use a knot vector to specify the parameter variation for the curve. The number of intermediate knots depends on the number of spans in the defining polygon. A duplicate intermediate knots value indicates that a multiple vertex (span of zero length) occurs at a point, and an intermediate knot value in triplicate indicates three concurrent vertices (two zero-length spans). The restriction on the

specification of the knot vector other than in nondecreasing order is that the same value cannot appear more than $k$ (the order) times. Specifically, $x_i$ is a knot of multiplicity $M$, if

$$x_i = x_{i+1} = \cdots = x_{i+M-1}, \tag{2.13}$$

where $M \leq k$. The continuity of $R(t)$ at this knot is reduced by $M - 1$. Since the continuity at a knot would otherwise be $C^{k-2}$, this means that in general, the continuity at a knot is $C^{k-M-1}$.

It is convenient to use an evenly spaced knot vector with unit separation between noncoincident knots. This gives integer values for the components of the knot vector. Since there are $n + 1$ control vertices in the control polygon, and each control vertex has a corresponding basis function, there are $n + 1$ basis functions. Moving through the knot vector, each basis function is nonzero over a successive set of $k+1$ knots. Thus, $k+n+1$ knots define $n + 1$ basis functions that correspond to the $n + 1$ control vertices. From this, it can be seen that the uniform knot vector with multiple end knots is

$$[0 \cdots 0\ 0\ 1\ 2 \cdots q\ q \cdots q]$$

where $q = n - k + 2$, that is

$$x_i = \begin{cases} 0 & \text{if } i = 0, \cdots, k-2, \\ i-k+1 & \text{if } i = k-1, \cdots, n+1, \\ n-k+2 & \text{if } i = n+2, \cdots, n+k. \end{cases} \tag{2.14}$$

In addition to the knot vector values, the order of the curve must be specified. The order of a curve is reflected in the knot vector that is used to generate the curve. If the order $k$ equals the number of polygon vertices and there are no multiple vertices, then a Bezier curve will be generated. As the order decreases, the curve produced lies closer to the defining control polygon. When $k = 2$, the generated curve is a series of straight lines that are identical to the defining polygon. For a third-order curve defined by

five vertices, the values of $t_{max} = 4 - 3 + 2 = 3$. The complete knot vector, using the multiplicity of three at each end, is then given by [ 0 0 0 1 2 3 3 3 ]. For a second-order curve with four vertices, if the two center vertices are made to coincide, the knot vector is [ 0 0 1 1 2 2 ].

## 2.6 Recursive Algorithm

The computational aspects of computing with B-splines have been considered by de Boor [8], who developed an algorithm that overcomes the problems of numerical instability inherent in previous algorithms based upon divided difference definitions of the B-splines. This is the essential algorithm that is adopted for the computation of the parametric B-spline curve approximations in this work. For the sake of completeness, the procedure is stated here for evaluating B-spline functions of the form in (2.11).

To be clear, the parameters used in the procedure are declared first as:

$P_i$ is the $i^{th}$ vertex on the control polygon,

$R_j$ is the $j^{th}$ point on the resulting B-spline approximation curve,

$k$ is the order of the B-spline basis function,

$n + 1$ is the total number of the control vertices,

$x_i$ is the $i^{th}$ element of the knot vector,

$M$ is the multiplicity of the knot,

$\triangle t$ is the increment of the local parameter $t$,

$num$ is the total number of points defined on the curve in the current knot interval,

$t_S$ and $t_E$ are the first and last parameter values of the current knot interval,

$NINT(x)$ is an internal function calculating the nearest integer of $x$.

The recursive procedure is as follows:

- Step 1. Calculate knot vector **x** according to the formula given in (2.14). If there is an intermediate vertex that has multiplicity $M$

($M \leq k$), the corresponding knot will also have multiplicity $M$ (see expression (2.13)).

- Step 2. Initialize variables, $j = 0$ and $g = 0$.
- Step 3. For $L = k - 1$ to $n$, do
  - step 3.1. For $i = 0$ to $n + k - 1$,
    find $i$, such that $i = L$ and $x_i \neq x_{i+1}$, then set $B_{i,1} = 1.0$;
    else, set $B_{i,1} = 0.0$.
  - step 3.2. Compute $t_S$, $t_E$ and $num$ as
    $t_S = x_L$, if $L = n$, then set $t_E = x_{L+1}$;
    else, set $t_E = x_{L+1} - \triangle t$.
    Set $num = NINT(\frac{t_E - t_S}{\triangle t})$.
  - step 3.3. For $N = 0$ to $num$,
    set $t = t_S + N \times \triangle t$ and
    $j = j + 1$.
    - step 3.3.1. For $order = 2$ to $k$, do
      - step 3.3.1a. For $i = 0$ to $n$,
        calculate $B_{i,order}$ using (2.10). If $order = k$, compute parametrically the summation as
        $g = P_i B_{i,k} + g$.
    - step 3.3.2. Store $g$ in an array $R_j$ and reset $g = 0$.

## 2.7 Examples

Due to the flexibility of B-spline curves, different types of control can be used to change the shape of a curve. Control can be achieved by changing the integer order $k$ for $2 \leq k \leq n + 1$, by use of repeating vertices or by

changing the number and/or position of nonrepeating vertices in the control polygon. These effects are shown in the following figures.

Figure 2.2 shows three B-spline curves of different order, each defined by the same four polygon vertices given by

$$\begin{bmatrix} 0 & 10 \\ 3 & 3 \\ 6 & 9 \\ 9 & 0 \end{bmatrix}$$

The second-order curve creates three straight lines between the four vertices, the fourth-order curve corresponds to Bezier curve for the polygon set, and the third-order curve produces a looser curve between the two end points. As the order of a curve increases, the resulting shape looks less like the defining polygon shape.

Figure 2.3 shows the effects of multiple vertices in the control polygon. For each of the four curves shown, the order of the curve is equal to the number of vertices in the defining polygon. The lower curve in Figure 2.3 is identical to the lower curve in Figure 2.2. The second lower curve in Figure 2.3 is a fifth-order curve with a double vertex at (6,9). The third curve is a sixth-order curve with a triple vertex at (6,9). The final seventh-order curve has a defining polygon given by

$$\begin{bmatrix} 0 & 10 \\ 3 & 3 \\ 6 & 9 \\ 6 & 9 \\ 6 & 9 \\ 6 & 9 \\ 9 & 0 \end{bmatrix}$$

e.g., four multiple vertices at [6,9]. This figure clearly shows how a curve can be pulled closer to a specific vertex position by use of multiple vertices.

In Figure 2.4 the control polygon vertices are

$$\begin{bmatrix} 0 & 6 \\ 2 & 4 \\ 4 & 2 \\ 6 & 8 \\ 6 & 8 \\ 8 & 5 \\ 10 & 0 \end{bmatrix}$$

for each curve. Here the curve is altered by changing the order of each curve, keeping the defining polygon constant. The first curve is of order seven. The second curve is of order five. This curve shape is closer to the polygon shape, especially near the double vertex. The third curve is of order three; notice that a "knuckle" occurs at the double vertex, since the slope and curvature are discontinuous.

Figure 2.5 demonstrates the fact that local changes can be made without affecting the entire shape of a curve. One can see that the last control vertex has been moved to show how it only affects a parametrically limited portion of the curve.

## 2.8 B-spline Surface Approximation

The parametric tensor product B-spline surface approximation to a control polyhedron can be defined [16] as

$$Q(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} V_{i,j} B_{i,k}(u) B_{j,l}(v), \tag{2.15}$$

where

$Q(u,v)$ is the B-spline surface approximation,

$V_{i,j}$ is the point forming a control polyhedron,

$B_{i,k}(u)$ and $B_{j,l}(v)$ are the normalized B-spline basis functions of order $k$ and $l$ along $u$ and $v$ directions respectively,

$u$ and $v$ are the local surface parameters.

The shape of a B-spline surface is controlled by the polyhedron vertices. All characteristics that B-spline curve approximation possesses can be extended similarly to the B-spline surface approximation, e.g., since B-spline basis function provides only local support, the extent of the surface influenced by a given polyhedron vertex is limited.

Figure 2.1: Fourth order B-spline basis functions.

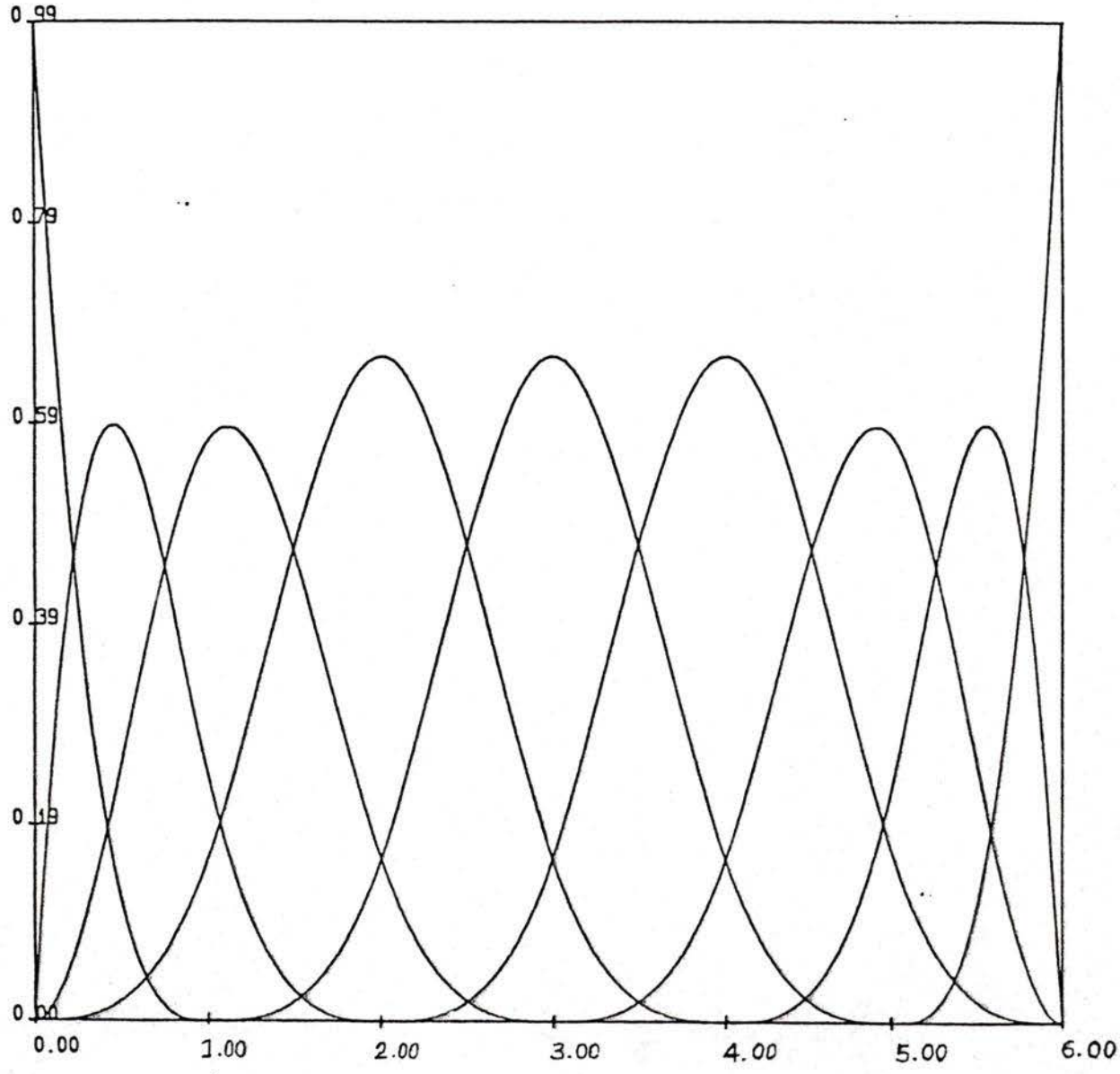


Figure 2.1: Fourth order B-spline basis functions.

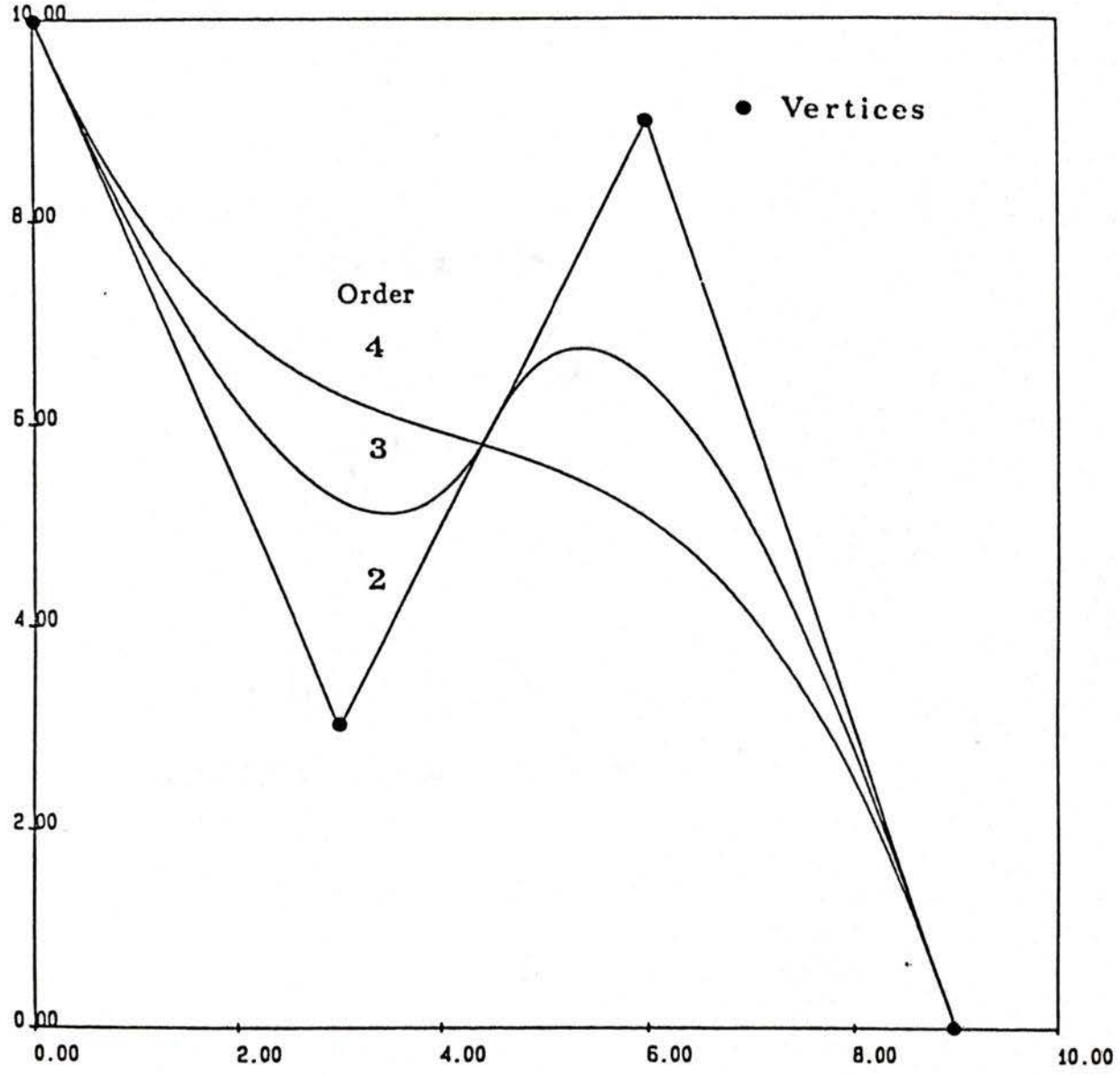


Figure 2.2: B-spline curves with different orders.

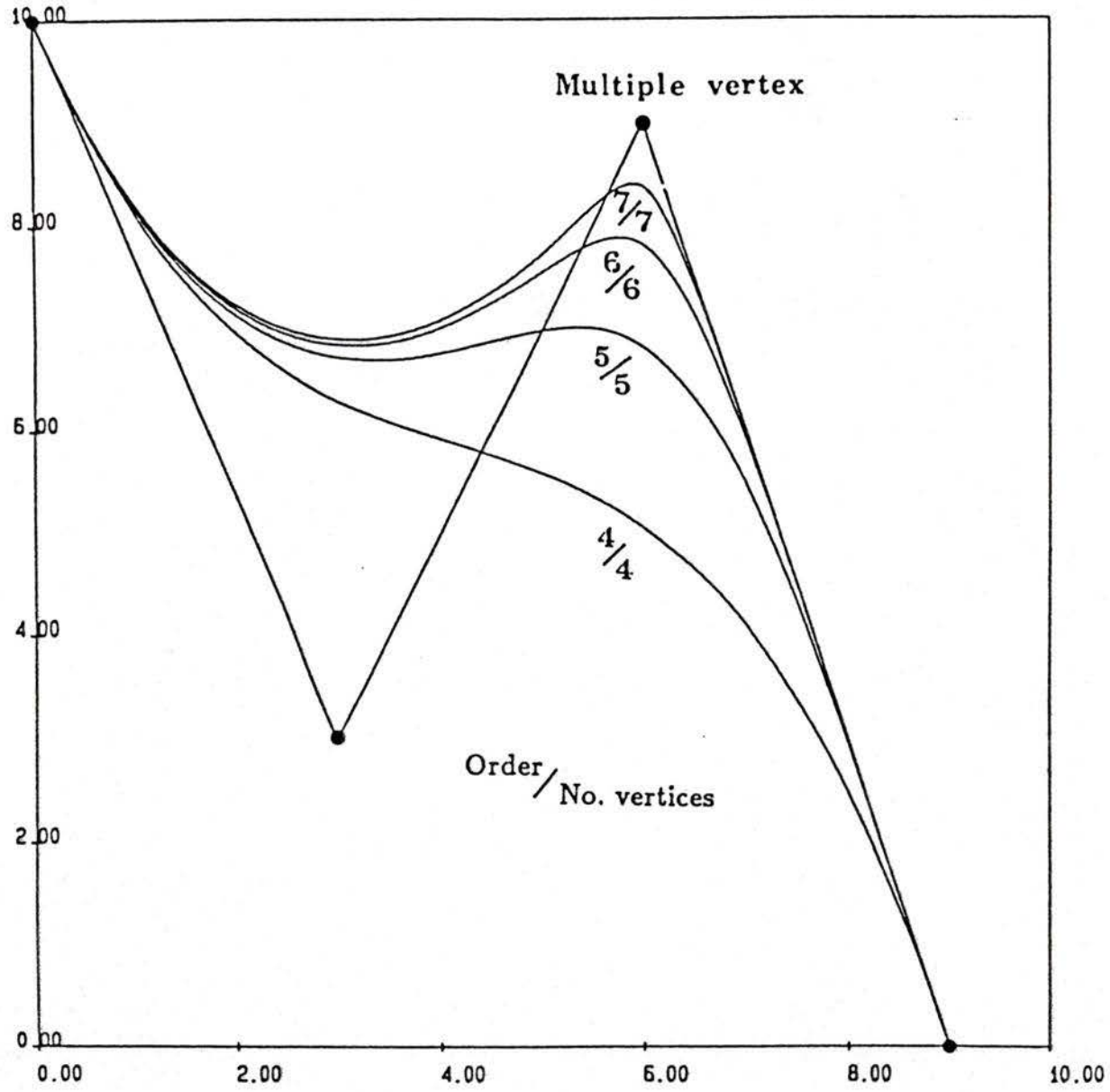


Figure 2.3: Multiple vertex effect: B-spline curves with different orders, defined on different number of the control vertices.

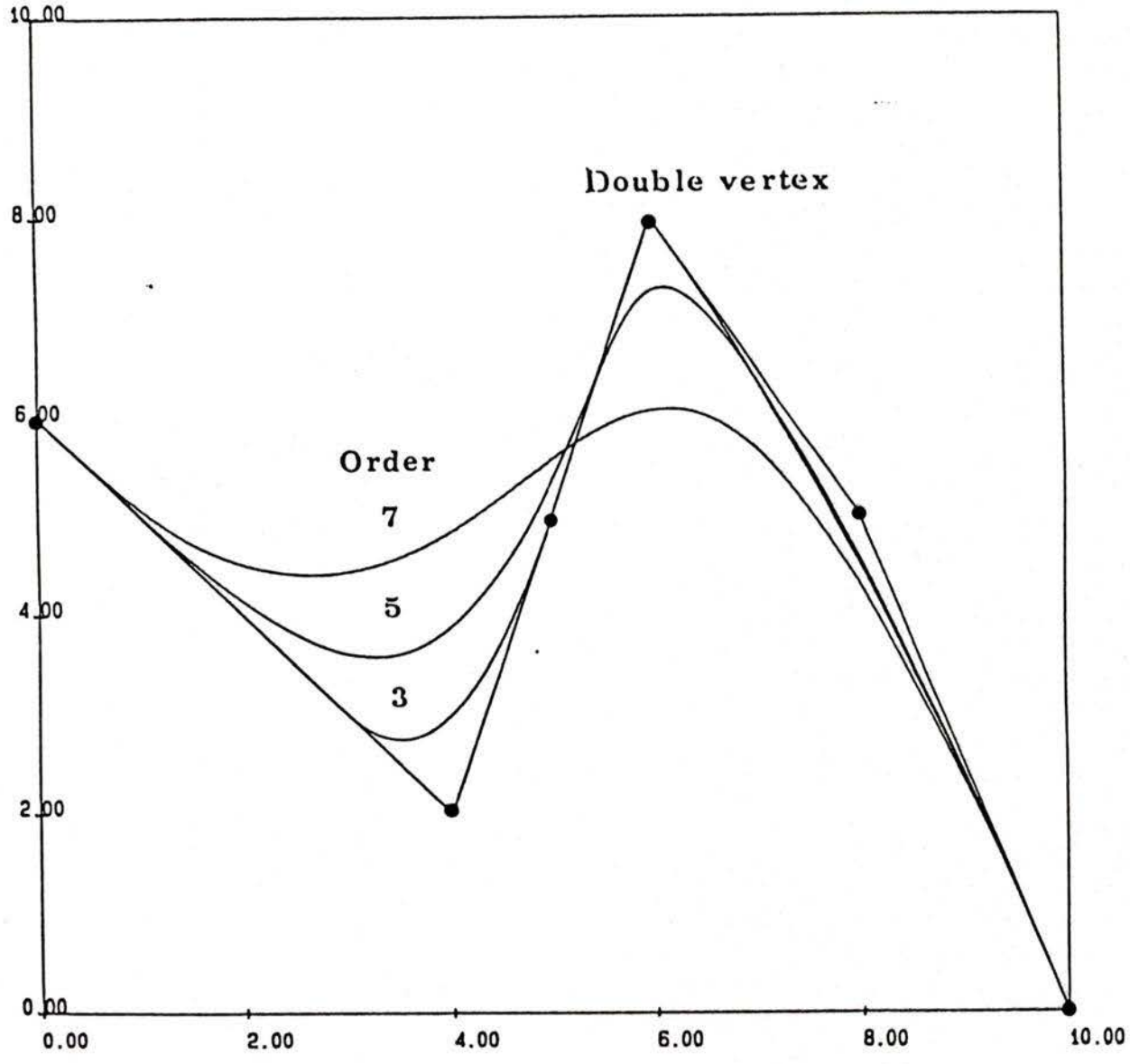


Figure 2.4: Multiple vertex effect: B-spline curves with different orders, defined on the same number of control vertices.

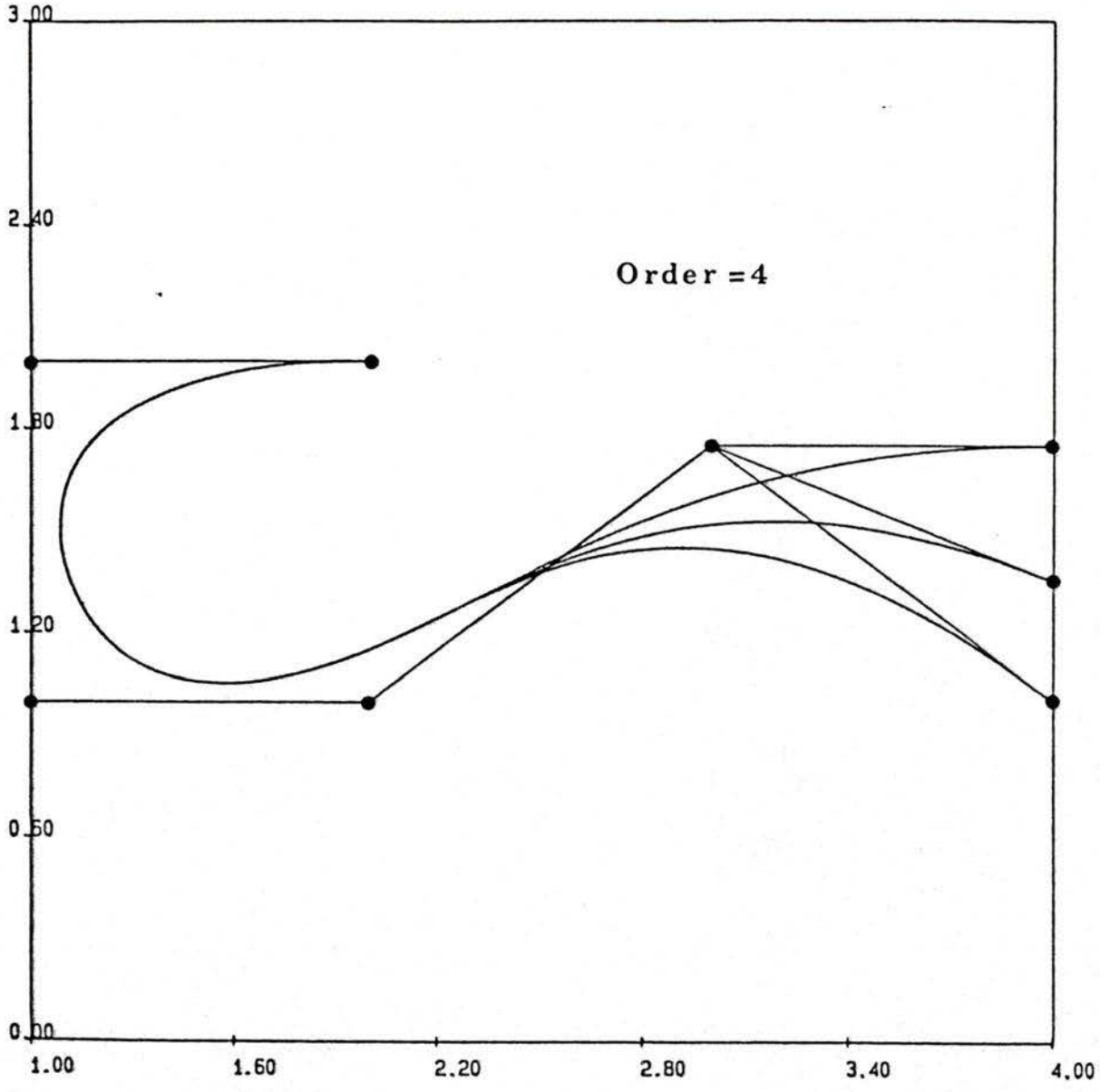


Figure 2.5: B-spline curves with local modification.

# Chapter 3

# Translation from B-spline Approximation to B-spline Interpolation

## 3.1 Introduction

In this chapter, a new efficient iterative algorithm for defining B-spline interpolation curves and surfaces is developed. The difference between a B-spline approximation solution and the data vertices is used to cumulatively modify the control vertices in order to generate a B-spline interpolation solution. The method converges rapidly to the interpolation solution and is more computationally efficient than the current matrix inversion interpolation solution. In addition, intermediate B-spline solutions are generated between the extremes of approximation and interpolation solutions.

## 3.2 Error Definition

B-spline approximation has a variation diminishing property [8], which means that the curve always lies within the convex hull of the enclosing polygon, as shown in Figure 3.1. In general, there is always a difference between each data vertex and the corresponding curve value. The error is defined parametrically as the difference between the original data vertices

$P_i$'s, and the current parametric B-spline approximation curve values $R_i$'s at the relevant knot points as

$$E_i = P_i - R_i, \tag{3.1}$$

for $0 \leq i \leq n$.

The values of $R_i$ at the end points of the curve are given by

$$R_0 = P_0, \tag{3.2}$$

$$R_n = P_n. \tag{3.3}$$

Since normalized B-spline basis functions have local support, the intermediate values of $R_i$ can be simply computed as [2]

$$R_i = B_i P_i + B_{i-1} P_{i-1} + B_{i+1} P_{i+1}, \tag{3.4}$$

where $B_i = \frac{2}{3}$, $B_{i-1} = \frac{1}{3}$, $B_{i+1} = \frac{1}{3}$, and $0 < i < n$.

For two-dimensional data points, (2.11) can be written as

$$R_x(t) = \sum_{i=0}^{n} P_{xi} B_{i,k}(t), \tag{3.5}$$

$$R_y(t) = \sum_{i=0}^{n} P_{yi} B_{i,k}(t). \tag{3.6}$$

For equally spaced $x$ values, $P_{xi}$'s, there is a linear relationship between $R_x$ and $t$, hence the errors in the $x$ component are zero. The total error $E_i$ (3.1) at each control vertex can then be interpreted as $E_{yi}$ ($y$ component) as depicted in Figure 3.2. If the data points are unequally spaced in $x$ values, the errors occur both in $x$ and $y$ directions and given as $E_{xi}$ and $E_{yi}$, as shown in Figure 3.3.

## 3.3 Updating Control Polygon for Curve Fitting

With data vertices equally spaced in the $x$ direction, the control vertices remain unchanged in the $x$ values, as shown in Figure 3.2. The error $E_{yi}$ is added to the original data value $P_{yi}$ ($y$ component) to give a temporary control value $PT_{yi}$. The points $(P_{xi},PT_{yi})$'s are then used as new control vertices, thereby obtaining the temporary curve value $RT_{yi}$ corresponding to the control vertex through (3.4). The error $E_{yi} = P_{yi} - RT_{yi}$ is again calculated. If $|E_i|$ $(0 \leq i \leq n)$ is less than a given tolerance, the iteration will terminate, otherwise $PT_{yi}$ is updated to $(PT_{yi} + E_{yi})$ and the process is repeated until the desired accuracy is reached. With data vertices unequally spaced in the $x$ direction, the control vertices in $x$ and $y$ are processed together, as shown in Figure 3.3.

Once the final control polygon is determined, the B-spline approximation to these points, which is a B-spline interpolation to the original data vertices, can be obtained using de Boor's algorithm. This algorithm requires two additional vertices that are linearly interpolated in the first and last spans of the final control polygon at one-third distance from each end. The method can be readily extended to three-dimensional curves.

## 3.4 Iterative Algorithm

The procedure required to implement the algorithm is as follows:

- Step 1. Input original control data vertices $P$ and tolerance $\varepsilon$.
- Step 2. Store $P$ in a temporary array $PT$, $PT = P$.
- Step 3. Calculate the current curve value $R$ corresponding to each new control vertex $PT$ using (3.2), (3.3) and (3.4).

- Step 4. Compute the error between the original data vertex $P$ and current curve value $R$ using (3.1).

- Step 5. Check if $|E| < \varepsilon$, if yes go to step 6;
  else, set $PT = PT + E$, go to step 3 (next iteration).

- Step 6. Linearly interpolate two points in the first and last interval of the new control polygon formed by $PT$ at one-third distance from each end point.

- Step 7. Calculate the B-spline approximation solution to the final control vertices.

## 3.5 Curve Fitting Illustration

To demonstrate the effectiveness of the technique and the very rapid convergence from the approximation to the interpolation solutions, two examples are given in Figures 3.4 and 3.5.

In Figure 3.4 the data vertices are derived from an analytical function $(y = sin(\frac{\pi x}{2}))$. The B-spline interpolation solution after four iterations is within a tolerance of 0.0001.

In Figure 3.5 a B-spline curve is fitted to the five data vertices given in Figure 3.2 and has a tolerance equal to 0.001 after ten iterations.

## 3.6 Updating Control Polyhedron for Surface Fitting

The same concept and similar algorithm can be applied to B-spline surface interpolation. For the parametric B-spline surfaces given by (2.15) with $k = l = 4$ and uniform knot sequences, the surface points corresponding to the control polyhedron vertices can be easily calculated. The four sets of boundary points are determined using (3.2), (3.3) and (3.4). The

interior surface points may be determined in an equivalent way to (3.4) and given as

$$\begin{aligned} Q_{i,j} &= (\frac{2}{3})^2 V_{i,j} + \frac{2}{3} \times \frac{1}{3}(V_{i-1,j} + V_{i+1,j} + \\ &\quad V_{i,j-1} + V_{i,j+1}) + (\frac{1}{3})^2 (V_{i-1,j-1} + V_{i-1,j+1} + \\ &\quad V_{i+1,j-1} + V_{i+1,j+1}), \end{aligned} \tag{3.7}$$

for $0 < i < n$ and $0 < j < m$.

According to the values of $Q_{i,j}$'s computed using (3.2), (3.3), (3.4) and (3.7), the errors are determined and control polyhedron vertices updated in a similar way to the procedure given previously. The B-spline surface approximation to the final set of control vertices may again be calculated using de Boor's algorithm. The additional vertices are calculated along the boundary spans.

## 3.7 Surface Fitting Illustration

B-spline interpolation solutions for two different surfaces are given in Figures 3.6 and 3.8.

In Figure 3.6 the original data vertices all lie on an analytical surface given by

$$z = \frac{1.25 + cos(5.4y)}{6 + 6(3x - 1)^2}. \tag{3.8}$$

The original control polyhedron formed by $(11 \times 9)$ vertices and the B-spline interpolation solution after five iterations are shown. The maximum error between the interpolating surface data points and the given data vertices is 0.0001 and the average error is 0.00002.

In Figure 3.7 an arbitrary surface is defined by $(6 \times 6)$ data vertices. The B-spline approximation and the control polyhedron are shown. The B-spline interpolation solution after ten iterations is given in Figure 3.8. The maximum and average error for B-spline interpolation solution after ten iterations is 0.01 and 0.001 respectively.

An intermediate B-spline fit to a set of data, obtained by stereo photogrammetry of a human face, is given in Figure 3.9. The face surface is defined by $(80 \times 80)$ points and the fitted surface by $(160 \times 160)$ points. The B-spline fit in this case was done so that the error above a certain level was not added and hence the surface not adjusted at these points. As the error at the periphery of the face was the largest, this had the effect of adjusting only the interior surface. This selective smoothing gives a close interior fit to the face and yet avoids the edge oscillations that would normally occur with a regularly interpolated surfaces.

## 3.8 Summary

In this chapter, an iterative algorithm for B-spline interpolation is developed. The number of iterations necessary to achieve the final interpolating curves or surfaces that satisfy a given tolerance is finite. The algorithm is numerically stable and computationally efficient; it can handle a large amount of sampled data well. Furthermore, the algorithm generates the intermediate B-spline solutions between the two extremes of approximation and interpolation, which may be used as preferred curve or surface definition.

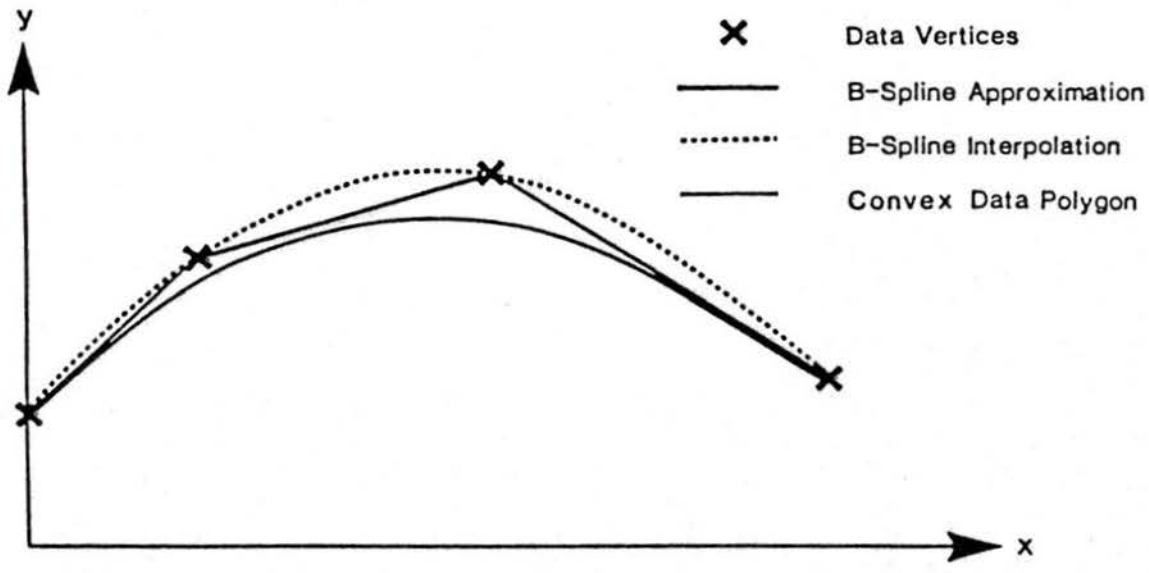


Figure 3.1: B-spline approximation and interpolation to four points.

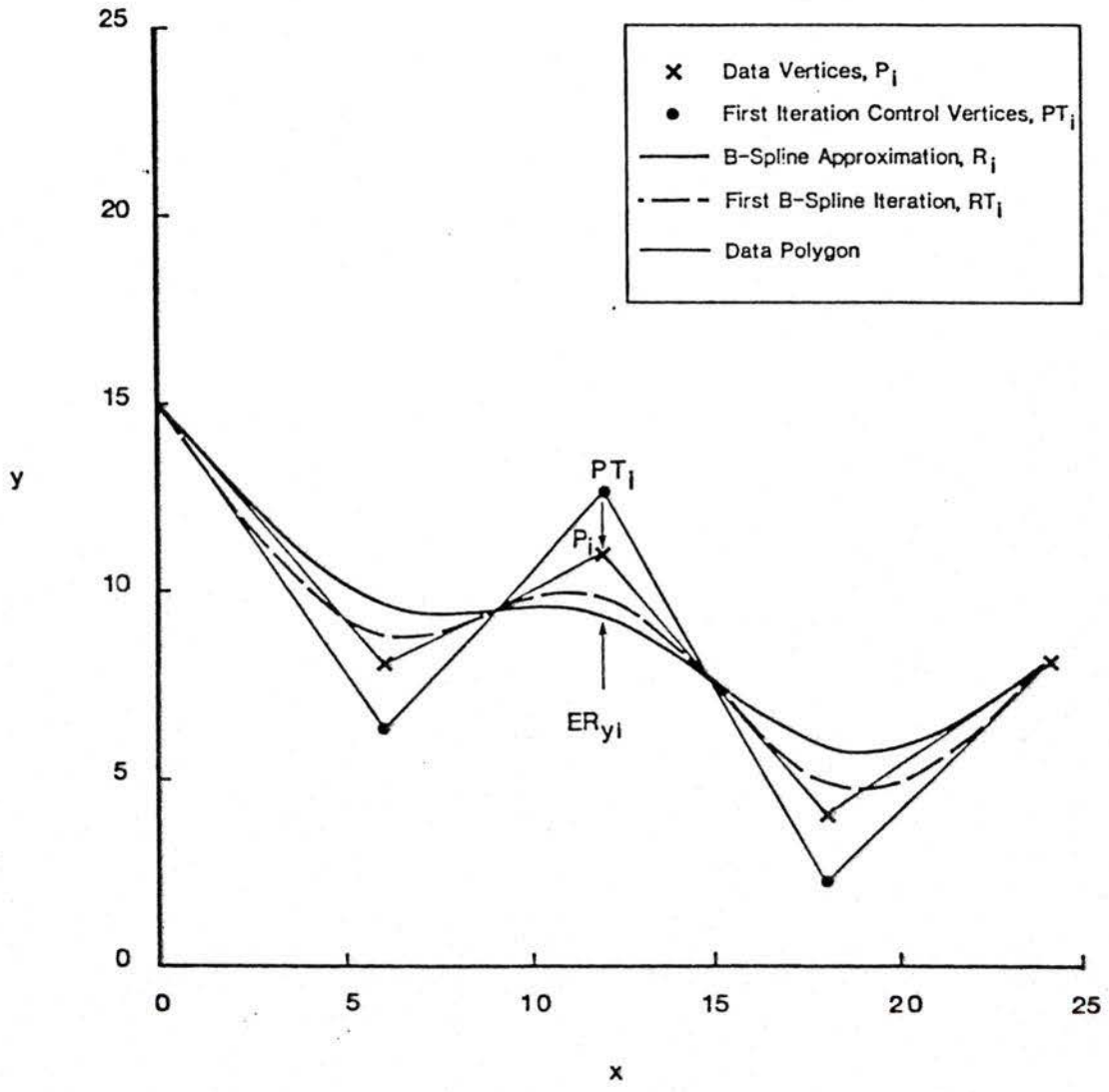


Figure 3.2: An illustration of the iterative algorithm for equally spaced $x$ values.

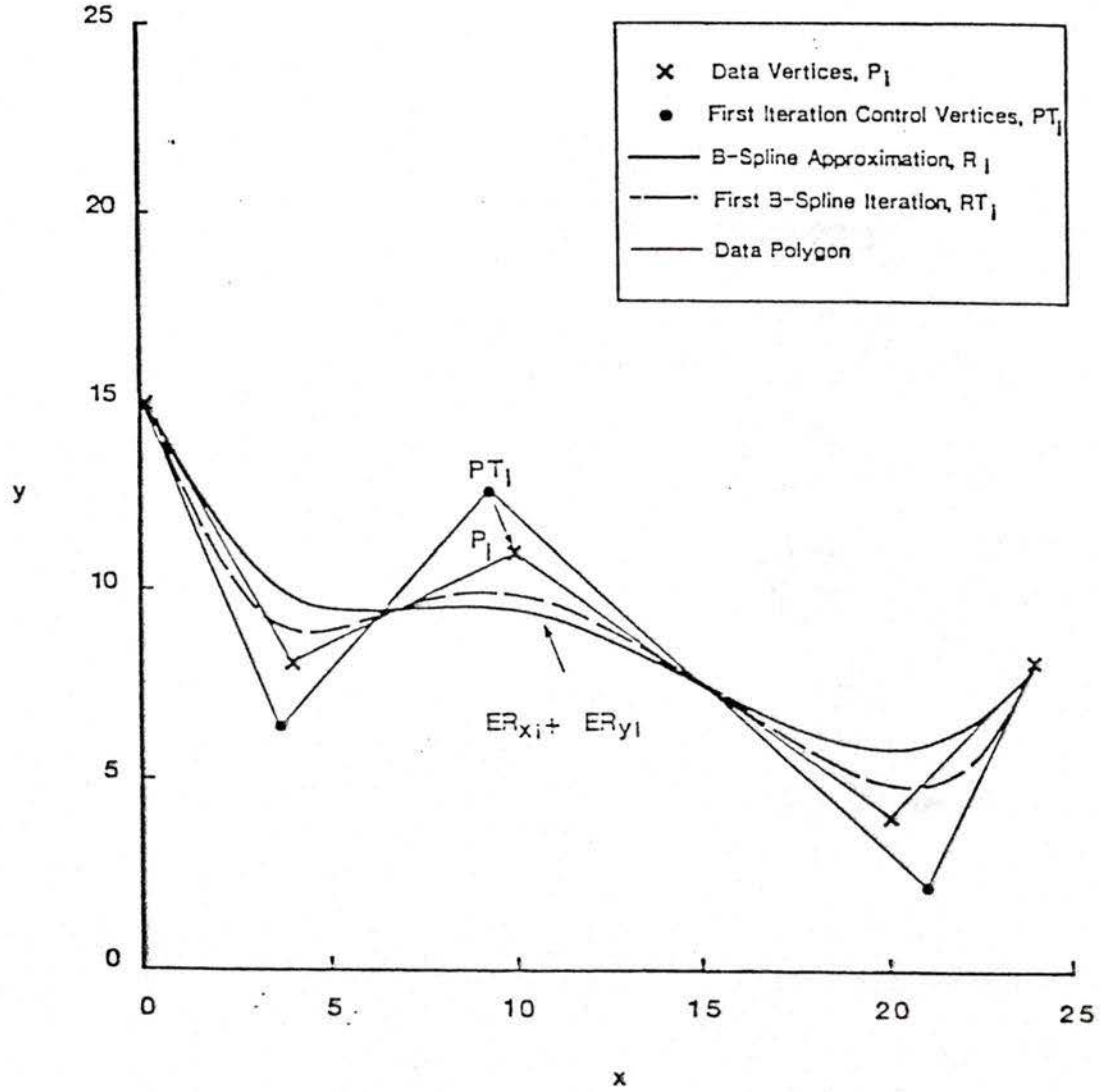


Figure 3.3: An illustration of the iterative algorithm for unequally spaced $x$ values.

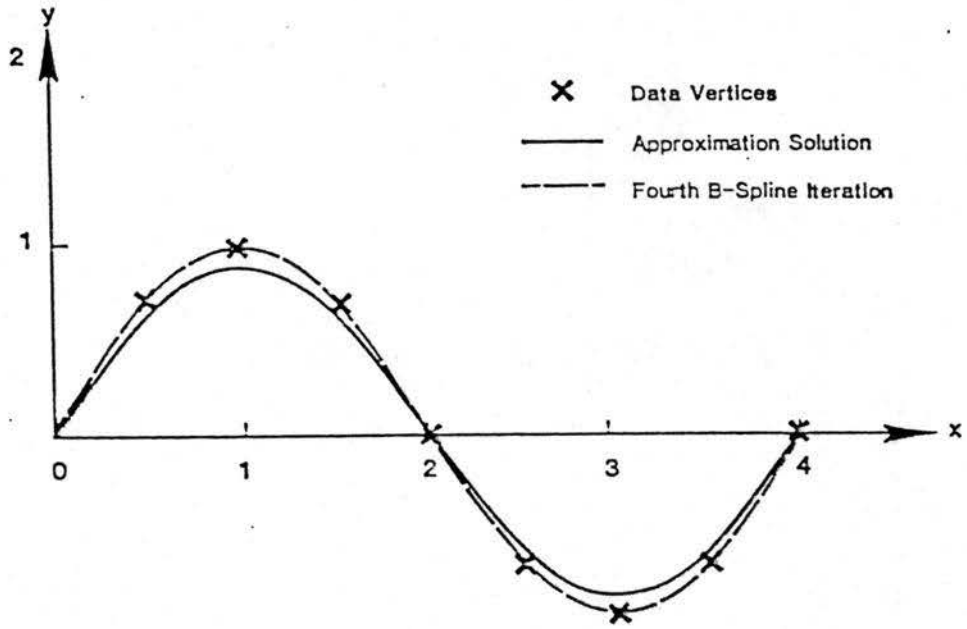


Figure 3.4: B-spline curves defined by the data vertices from a *sine* function.

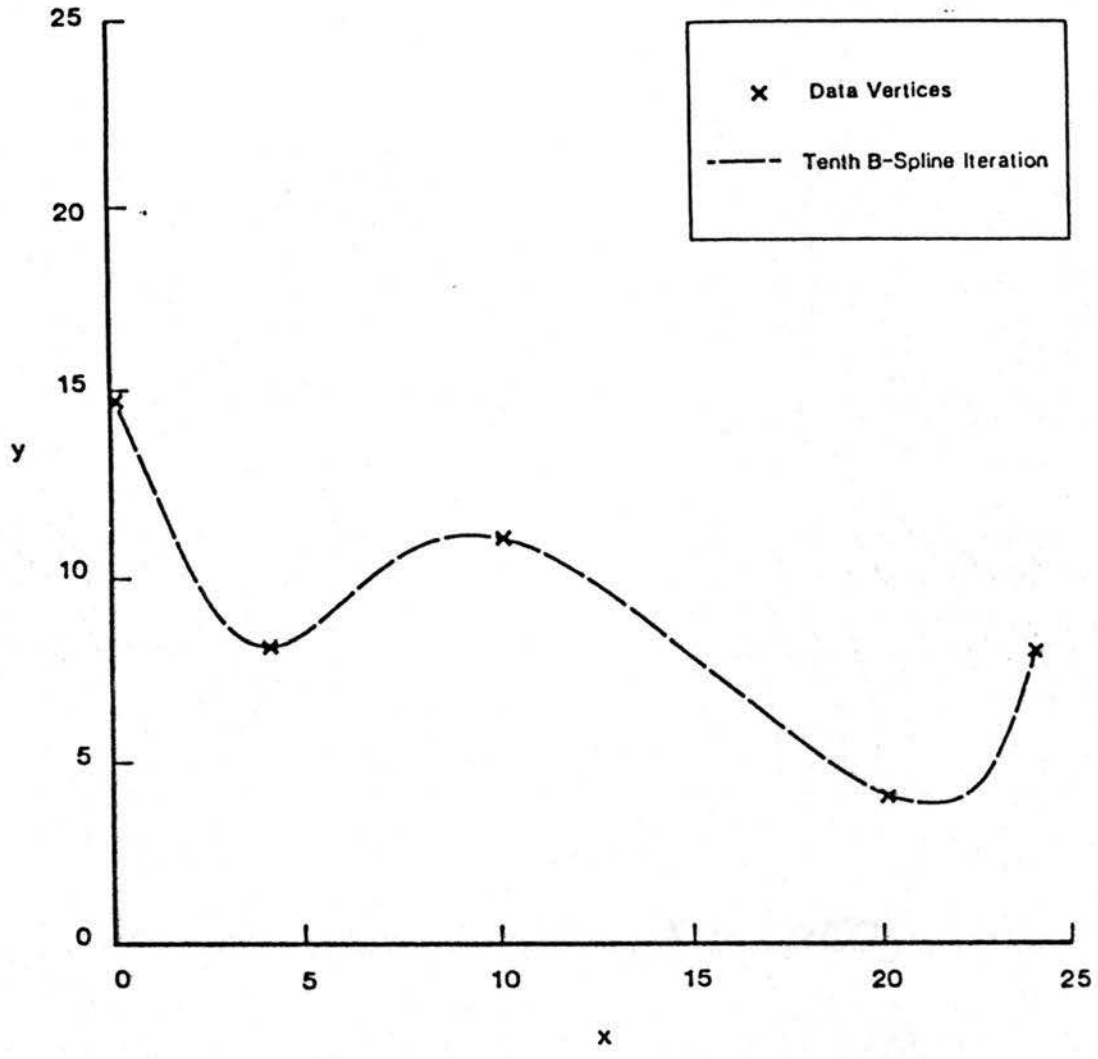


Figure 3.5: B-spline interpolation to five arbitrary data vertices.

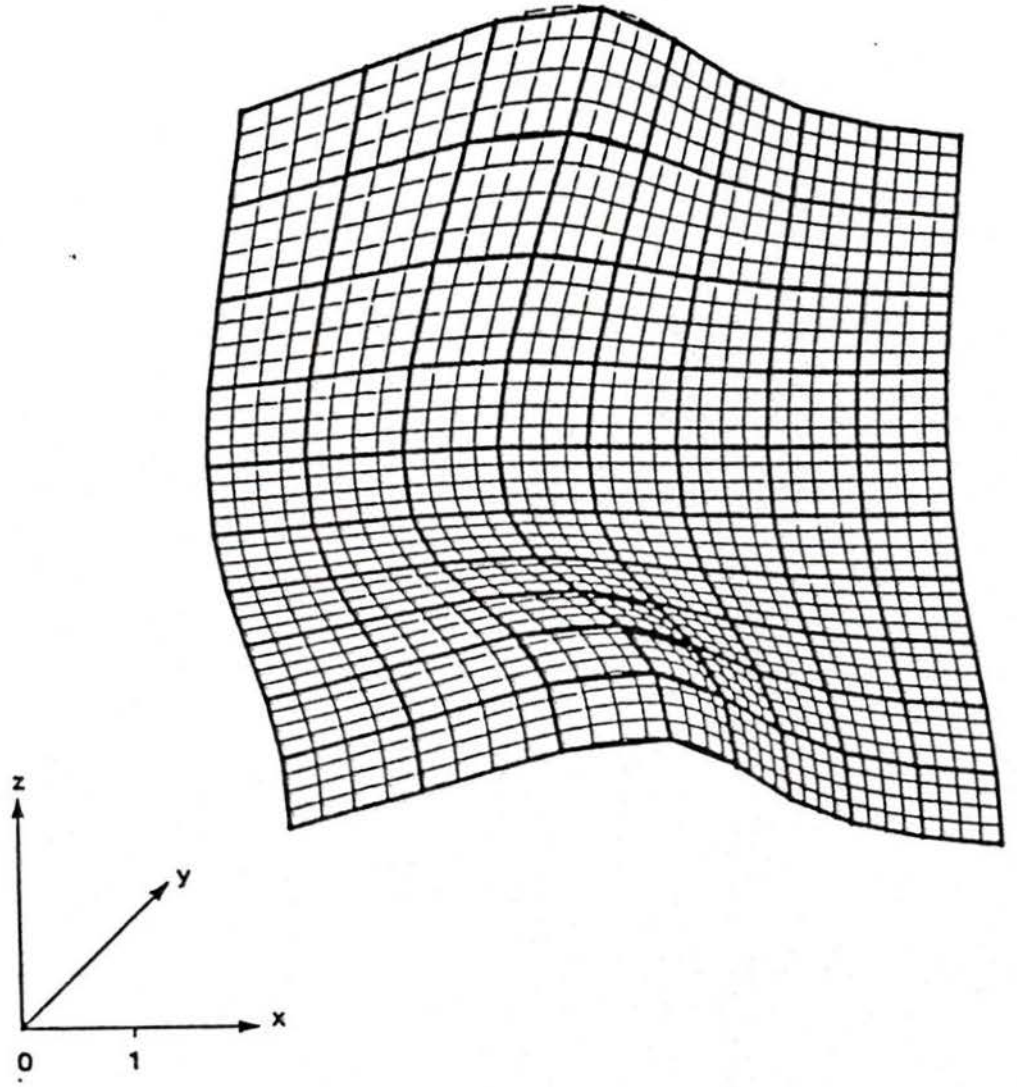


Figure 3.6: B-spline surface interpolation (five iterations) to 11 × 9 data vertices from a known analytic equation.

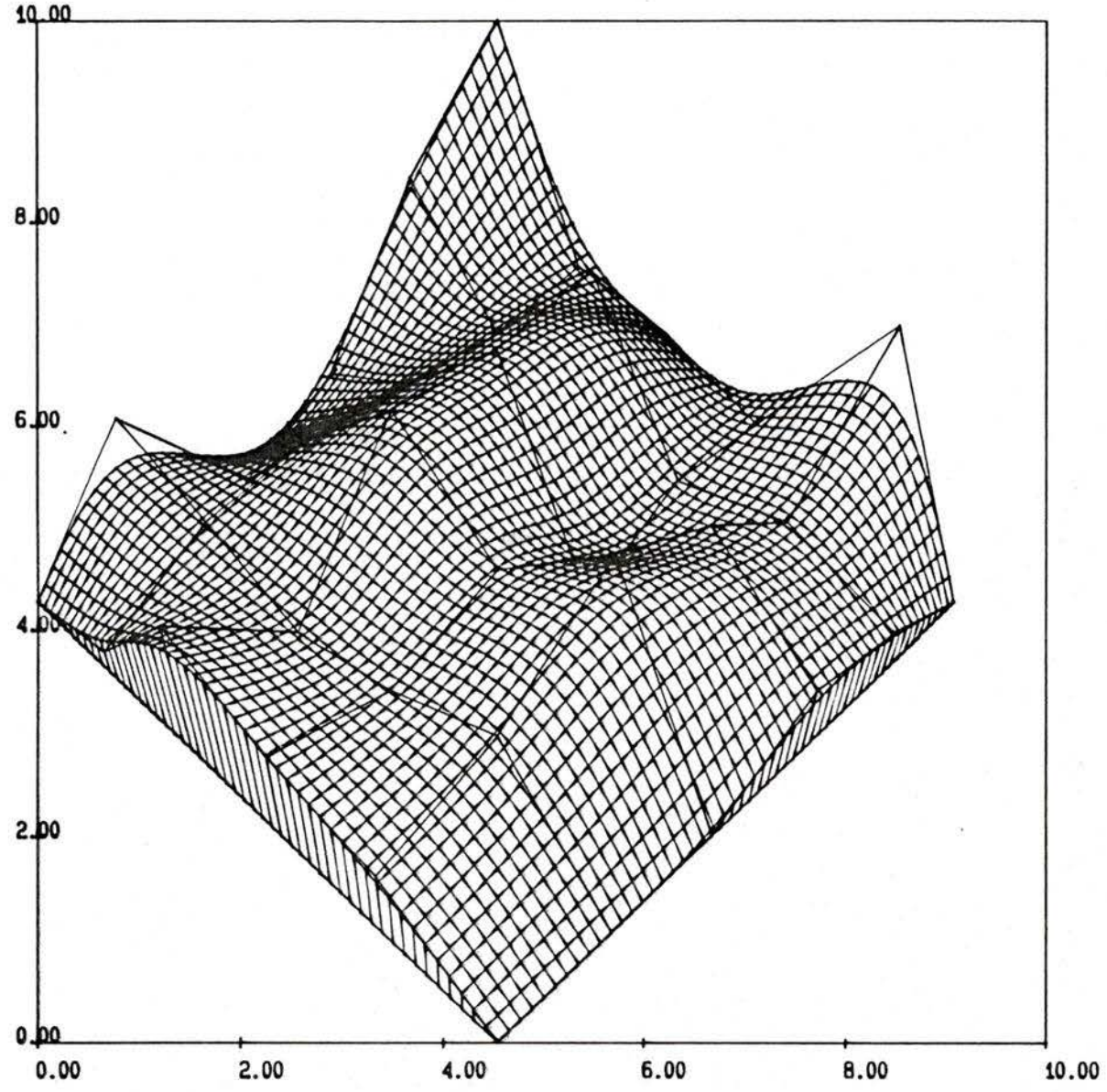


Figure 3.7: B-spline surface approximation defined on 6×6 arbitrary control polyhedron vertices.

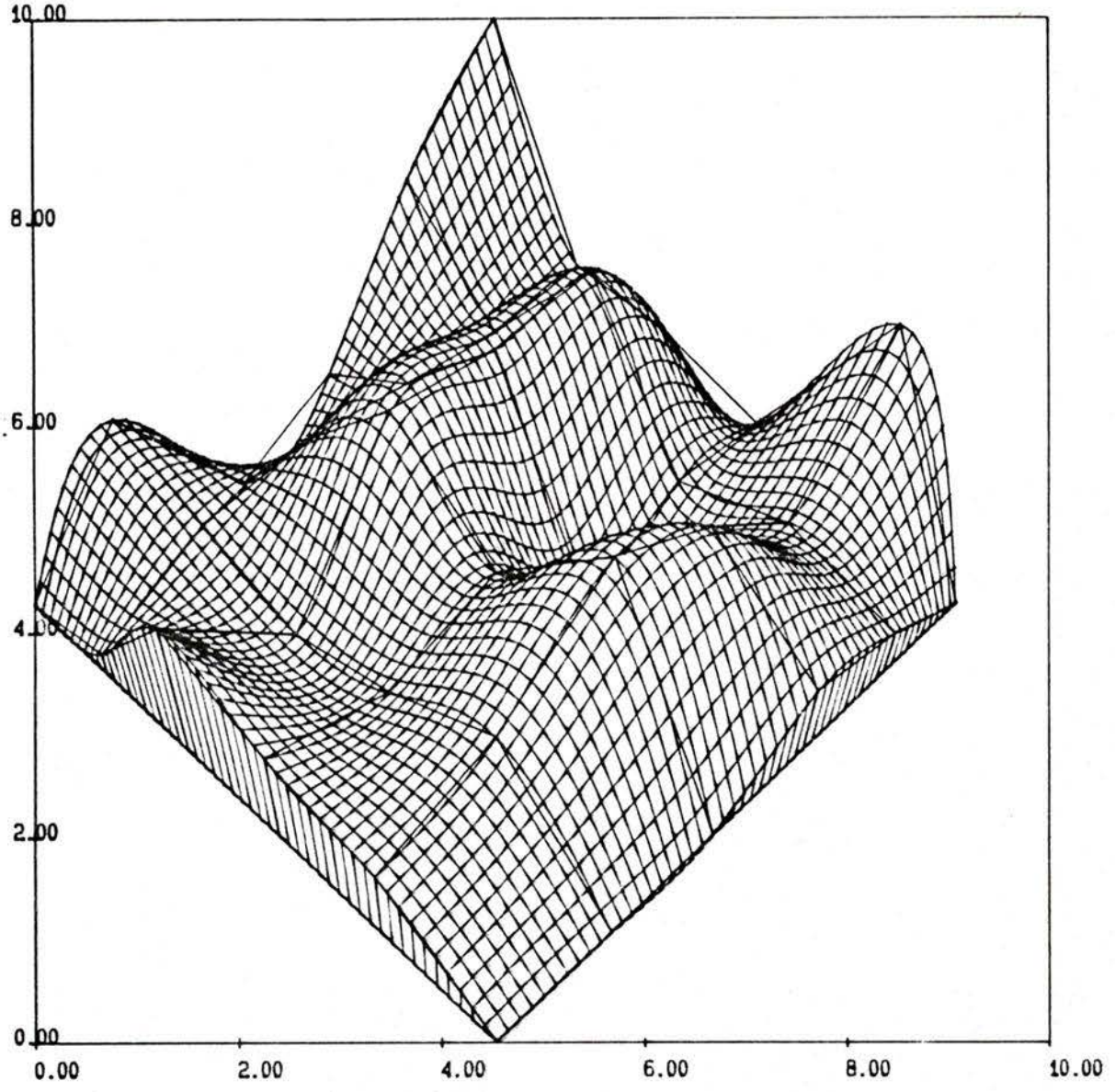


Figure 3.8: B-spline surface interpolation (ten iterations) to the same control polyhedron as given in Figure 3.7.

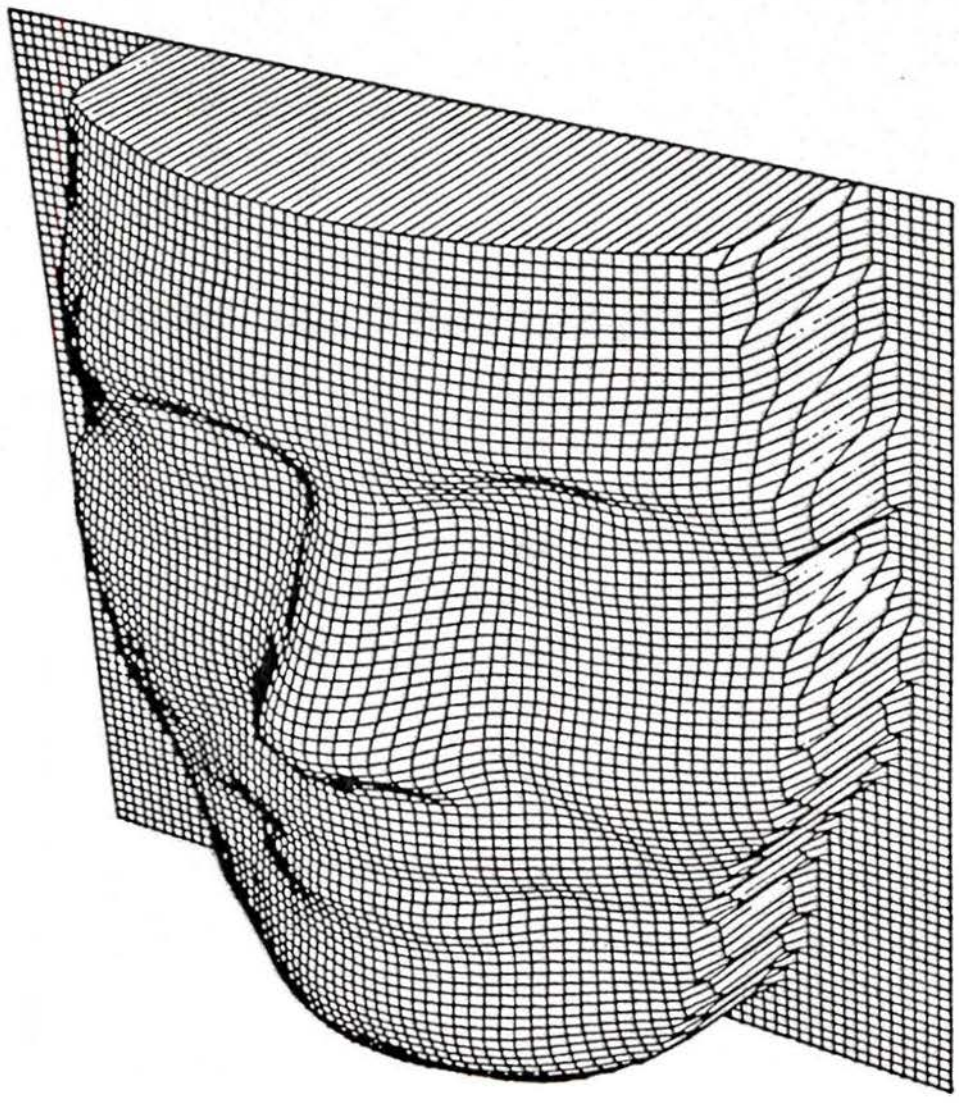

Figure 3.9: An intermediate B-spline surface fitted to 80 × 80 data points obtained by stereo photogrammetry.

# Chapter 4

# Fairing of Compound Curvature Curves

## 4.1 Introduction

With approximation (or interpolation) of curves using the B-spline method, the approximating curves may sometimes have undesirable inflection points. In some engineering applications, for example, fairing ship hull lines, this is unacceptable. Although several methods of detecting this lack of fairness have been developed, interpretation and correction of the data are based on manual input. This work is time consuming and requires high skill.

In this chapter, a semi-automatic, iterative detection and fairing technique based on the *B-spline error methods* given in Chapter 3 is presented. The detection of lack of smoothness is based on the sign of the cross-product of consecutive line segments of the control polygon. In this approach, therefore, all detection and correction procedures are made on the control polygon and not on the curve itself.

## 4.2 Development of Smoothing Technique

### 4.2.1 Detecting Unfair Vertices

It is well known that the B-spline approximation has a variation diminishing property, that is [16],

1. it approximates linear functions exactly, and
2. the number of intersections of the graph of the B-spline approximation with any straight line ($y = a + bx$) does not exceed the number of crossings of that straight line with control polygon.

In brief, the variation diminishing property means that the approximation is always "smoother" in the sense of undulations than the control polygon, that is, if the control polygon has no change of sign in curvature, then neither does the B-spline approximating curve. Using these characteristics, the smoothness of the approximating curve can be predicted by checking the nature of its control polygon.

In order to simplify the problem, consider the two-dimensional concave down polygon vertices, as shown in Figure 4.1. The control polygon vertex vector is given by $\vec{P_i}$ $(0 \leq i \leq n)$, and the vector *cross-product* on a vertex $P_i$ is defined as

$$\vec{CP_i} = (\vec{P_i} - \vec{P_{i-1}}) \times (\vec{P_{i+1}} - \vec{P_i}) \tag{4.1}$$

for $0 < i < n$.

For the control polygon lying on the $x - y$ plane, the vector $\vec{CP_i}$ can only have two directions, namely $+\vec{z}$ or $-\vec{z}$. The *concave down* polygon is defined as one whose vector cross-product $(\vec{CP_i})$ $(0 < i < n)$ can only have one direction, that is, the direction of $-\vec{z}$.

For the sake of simplicity, denote that

- if $\vec{CP_i}$ is in the $-\vec{z}$ direction, then $CP_i < 0$,

- if $\vec{CP_i}$ is in the $+\vec{z}$ direction, then $CP_i > 0$,
- if $\vec{CP_i}$ equals to zero, then $CP_i = 0$,

for all $0 < i < n$.

To illustrate the detection procedure, based on the sign of the vector cross-products ($CP_i$'s) of the control polygon, the vertex $P_2$ of Figure 4.1 is moved downwards, as shown in Figure 4.2. If the vector cross-products are calculated along the control polygon, the resultant $CP_i$'s will be negative at all vertices, except at one vertex, ($P_2$), which is positive, $CP_2 > 0$. This indicates that if a B-spline curve were formed from this control polygon, an inflection point might occur around the vertex $P_2$.

The same result is obtained if the vertices $P_1$ and $P_3$ of Figure 4.1 are moved upwards while vertex $P_2$ is not moved, as shown in Figure 4.3. After calculating the vector cross-product at vertices along the control polygon, it is found that only $CP_2 > 0$. Therefore, if the vector cross-product at one point is non-negative, its two neighboring points may also need to be adjusted.

### 4.2.2 Determining the Range of Adjustment

In many practical applications, the two end points on the curve are fixed. Thus the maximum range of adjustment is between the second and second to the last vertex of the control polygon. For convenience, the sequence marks of these two vertices are denoted *ifirst* and *ilast*.

The actual range is determined in two steps. In the first step, the first and last sequence marks of the vertices whose vector cross-product are non-negative are determined. For convenience, these two extreme position marks are denoted as *low* and *upper* respectively. In the second step, the sequence position marks immediately before *low* and after *upper* are taken as the limits of the adjustment range. The possibilities for the limit position

of the adjusting range with starting sequence mark, *ibegin*, and ending sequence mark, *iend*, are given below.

At first, if $low = upper$, there are three possibilities, which are

- if $low = ifirst$, set $ibegin = low$ and $iend = low + 1$,
- if $low = ilast$, set $ibegin = low - 1$ and $iend = low$,
- if $low \neq ifirst$ and $low \neq ilast$, set $ibegin = low - 1$ and $iend = low + 1$.

Next, when $low \neq upper$, there are four cases, which are

- if $low \neq ifirst$ and $upper \neq ilast$, set $ibegin = low - 1$ and $iend = upper + 1$,
- if $low \neq ifirst$ and $upper = ilast$, set $ibegin = low - 1$ and $iend = upper$,
- if $low = ifirst$ and $upper \neq ilast$, set $ibegin = low$ and $iend = upper + 1$,
- if $low = ifirst$ and $upper = ilast$, set $ibegin = low$ and $iend = upper$,

In the following section, the problem of how to adjust the points whose sequence position mark is within *ibegin* and *iend* will be discussed.

### 4.2.3 Adjusting Unfair Vertices

In order to adjust unfair vertices so that the resulting curve does not deviate too far from the original curve, corrections made to a vertex are restricted to a triangular area formed by the vertex and the two surrounding vertices.

For convenience, denote that as shown in Figure 4.4,

- the line $(L_i)$ connecting $P_{i-1}$ and $P_{i+1}$ is associated with vertex $P_i$,

- the unit vector $\vec{N_c}$ which is perpendicular to line $L_i$ and satisfies the condition,

$$(\vec{P_{i+1}} - \vec{P_{i-1}}) \times \vec{N_c} > 0,$$

  is the direction of adjustment for $P_i$,

- the distance between $P_i$ and $L_i$ is $d_i$,

- the triangle consisting of $P_i$, $P_{i-1}$ and $P_{i+1}$ is $P_i$'s associated triangle $\triangle_i$,

The adjustment to a vertex $P_i$ is given by

$$P\vec{N}_i = \vec{P}_i + rate1 \times d_i \times \vec{N}_c \quad \text{for } CP_i \geq 0, \tag{4.2}$$

and

$$P\vec{N}_i = \vec{P}_i - rate2 \times d_i \times \vec{N}_c \quad \text{for } CP_i < 0, \tag{4.3}$$

where $PN_i$ is the new control polygon vertex after a correction is made to $P_i$. The amount of correction can be controlled by adjusting rates ($rate1$, and $rate2$). If $rate1$ and $rate2$ are given about the same values, the resulting curve will go through the middle between the control vertices whilst if $rate1 \gg rate2$, the finishing curve will go closer to those vertices whose vector cross-products are negative, and vice versa. The curves corresponding to different adjusting rates are shown in Figures 4.5, 4.6 and 4.7.

### 4.2.4 Other Remarks

For curves with two or more concave sections, it is necessary to divide the curve in segments in which only a single sign of curvature is expected. The dividing sequence marks ($ifirst$ and $ilast$) are required to be input manually. If the control polygon is concave up, it can be first swapped into concave down form and then swapped back after the fairing procedure

is finished. If the curve is given in three dimensions, the curve can be smoothed in two dimensions by considering the cases in the $x-y$ and $x-z$ planes respectively.

### 4.2.5 B-spline Approximation Smoothing Procedure

The steps in the detection and adjustment of the control polygon for B-spline approximation are given below. First, the vector cross-product values, $CP_i$'s, for each segment range $(ifirst, ilast)$ are calculated. If one or more results of the cross-products are found non-negative, it implies that unfair spots exist within the segment. The corresponding correction is made according to equations (4.2) and (4.3). In this way, a set of temporary vertices, $\vec{PN_i}$'s, $(ifirst \leq i \leq ilast)$ are obtained. The values for $CP_i$'s are calculated for the temporary control polygon vertices ($PN_i$'s). If non-negative results are still found, the above procedure is repeated. The iteration terminates when the cross-product at all vertices in the segment becomes negative. A B-spline approximation curve is then fitted to the final control polygon as shown in Chapter 2.

### 4.2.6 B-spline Interpolation Smoothing Procedure

With B-spline interpolation, the procedure described above is initially undertaken. Once the final set of control polygon vertices is obtained, the B-spline interpolation control polygon vertices, $PT_i$'s, are calculated using the B-spline error adding technique described in Chapter 3. The cross-product check is then applied to the vertices, $PT_i$'s. If unfair spots are detected, the corrections are made to the corresponding first smoothed vertices, $PN_i$'s. To make sure that the result from the second correction does not disturb the fairness obtained through the first smoothing procedure, the newly adjusted vertices are passed through the first smoothing process again. The whole procedure is repeated until the criterion of the

cross-product check to the vertices, $PT_i$'s, is satisfied. A B-spline approximation curve to this control polygon actually gives a fully faired B-spline interpolation to the original data.

## 4.3 Iterative Smoothing Algorithm

The steps required to implement the algorithm are as follows:

- Step 1. Input data file name.
- Step 2. Set the adjusting rates (*rate1* and *rate2*).
- Step 3. Identify curve segments and enter the dividing sequence marks, *ifirst* and *ilast*, in which data points are to be examined.
- Step 4. Determine if any segment is concave down or concave up within the above region. If it is concave up, swap the position of the data points and set $swap\_flag = 1$.
- Step 5. Go through the approximation smoothing procedure.
- Step 6. Decide if to smooth the data points further. If *yes*, go through the interpolation smoothing procedure.
- Step 7. If $swap\_flag = 1$, swap the final data points back.
- Step 8. Output the faired data file.

The flowcharts of the resulting computer program with relevant subroutines are given in Figures 4.11 to 4.19.

## 4.4 Curve Smoothing Illustration

To illustrate the effectiveness of the technique, three examples are given in Figures 4.8, 4.9 and 4.10.

In Figure 4.8, two curves are shown. One is a parabolic curve with perturbations and the other is the corrected curve. It is clearly seen that the corrections are only made around the region of disturbances.

The curves shown in Figures 4.9 and 4.10 represent the $x - y$ view of two sheer lines of a single and double kayak hull. In Figure 4.9, the original "concave down" data points show oscillations. The faired curve eliminates these oscillations while preserving the original shape.

In Figure 4.10, the original curve is divided into three segments according to convexity. The first and the last segments have concave up characteristics and no inflection points, while the curve in the middle region should be concave down. However, the cross-product check indicates that unfair spots exist. Corrections are made accordingly. The improved data points (Figures 4.9 and 4.10) have been used for the single and double kayak hull generation, respectively.

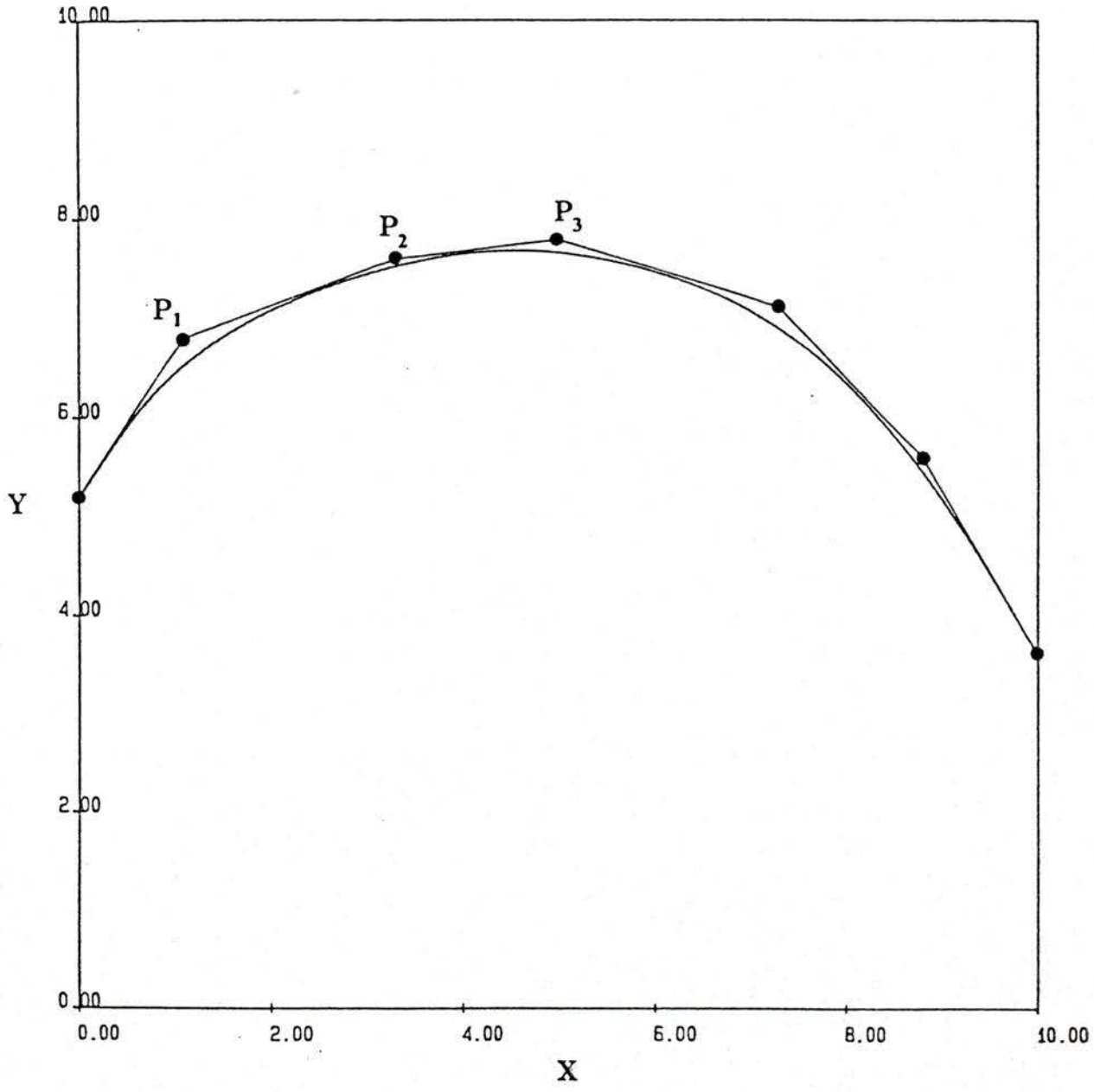


Figure 4.1: A typical concave down polygon with its B-spline approximation.

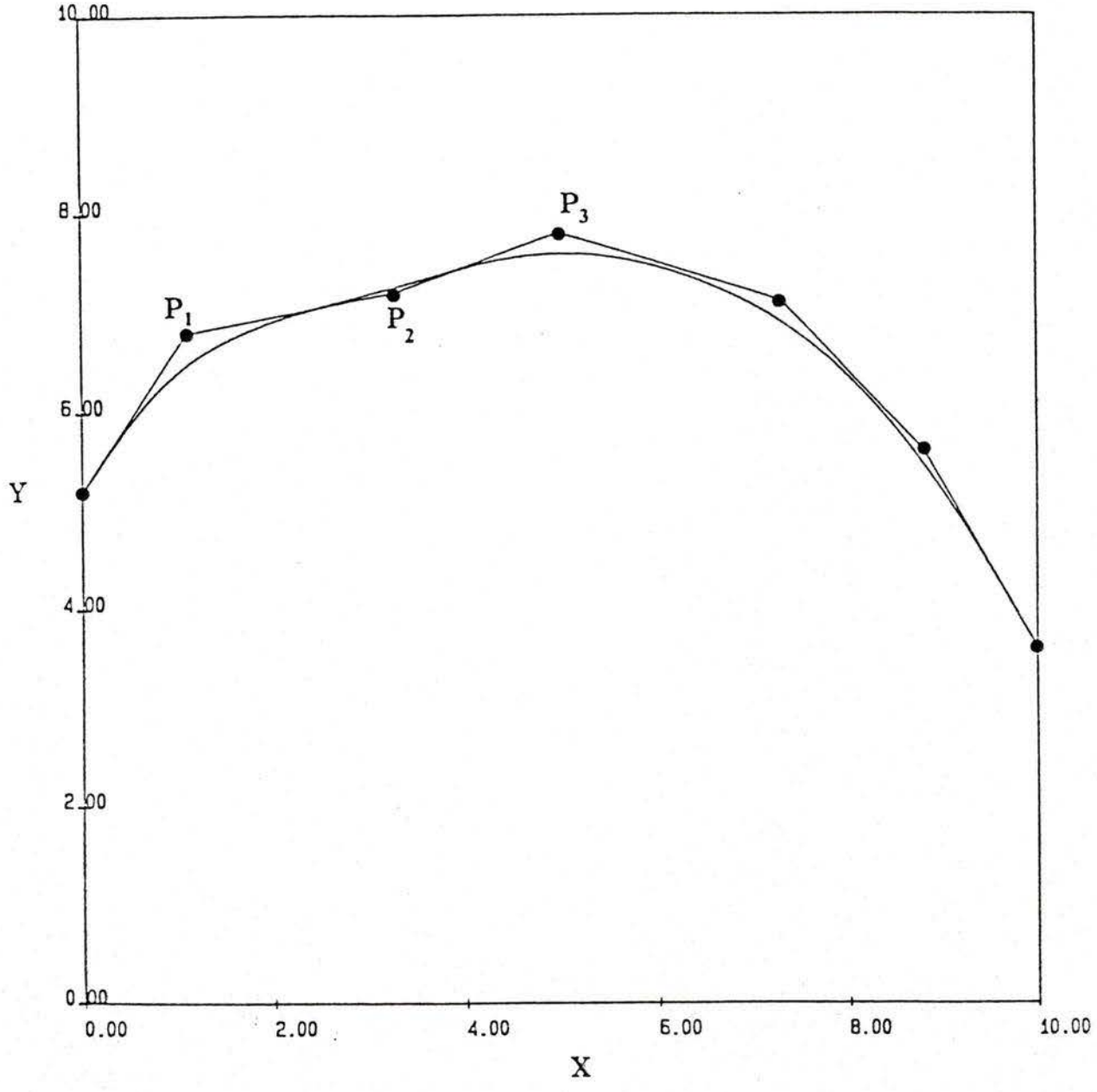


Figure 4.2: A concave down polygon with inflection point caused by moving down vertex $P_2$.

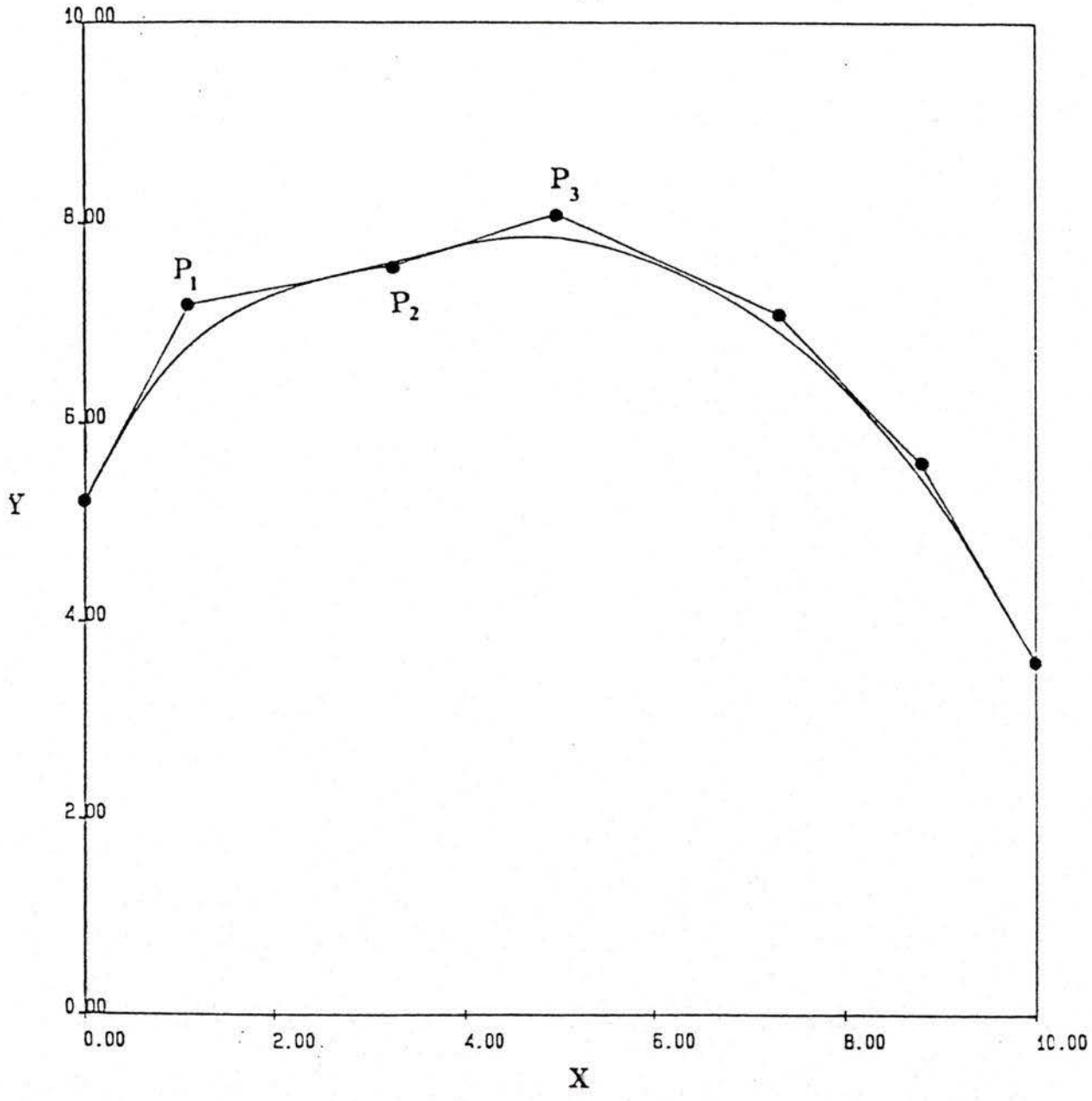


Figure 4.3: A concave down polygon with inflection point caused by moving up vertices $P_1$ and $P_3$.

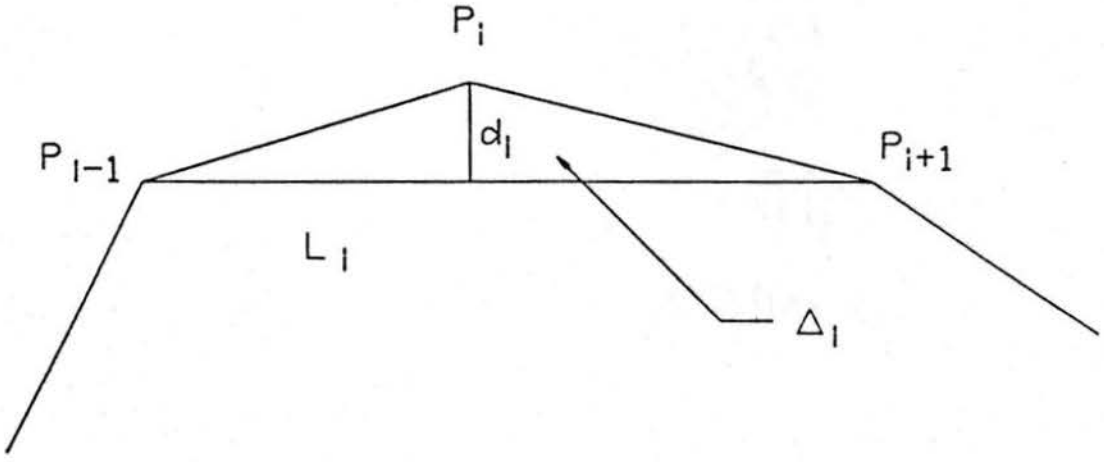


Figure 4.4: The notation illustration.

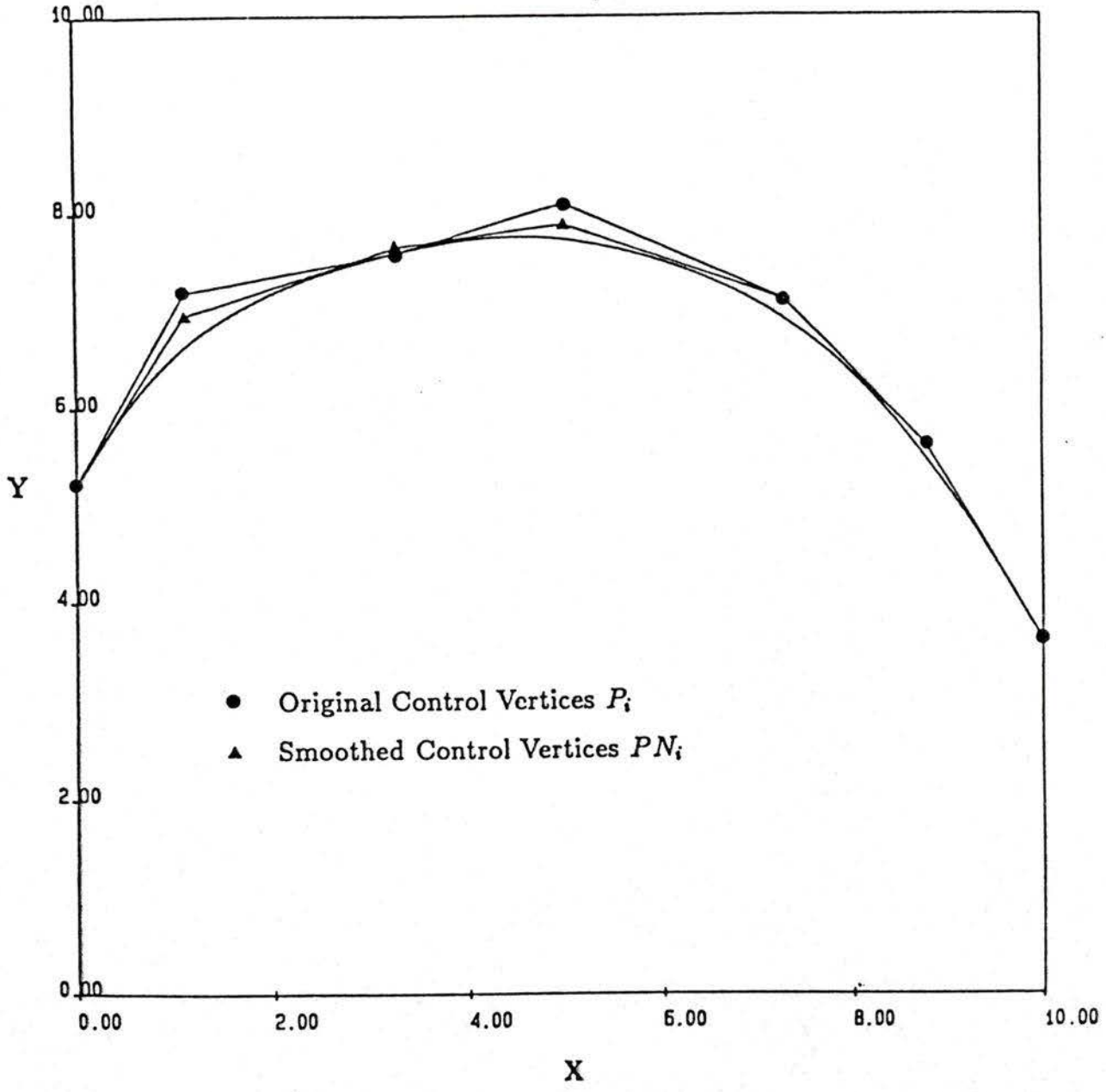


Figure 4.5: A faired curve to the control polygon vertices given in Figure 4.3 with $rate1 = 0.6$, and $rate2 = 0.3$.

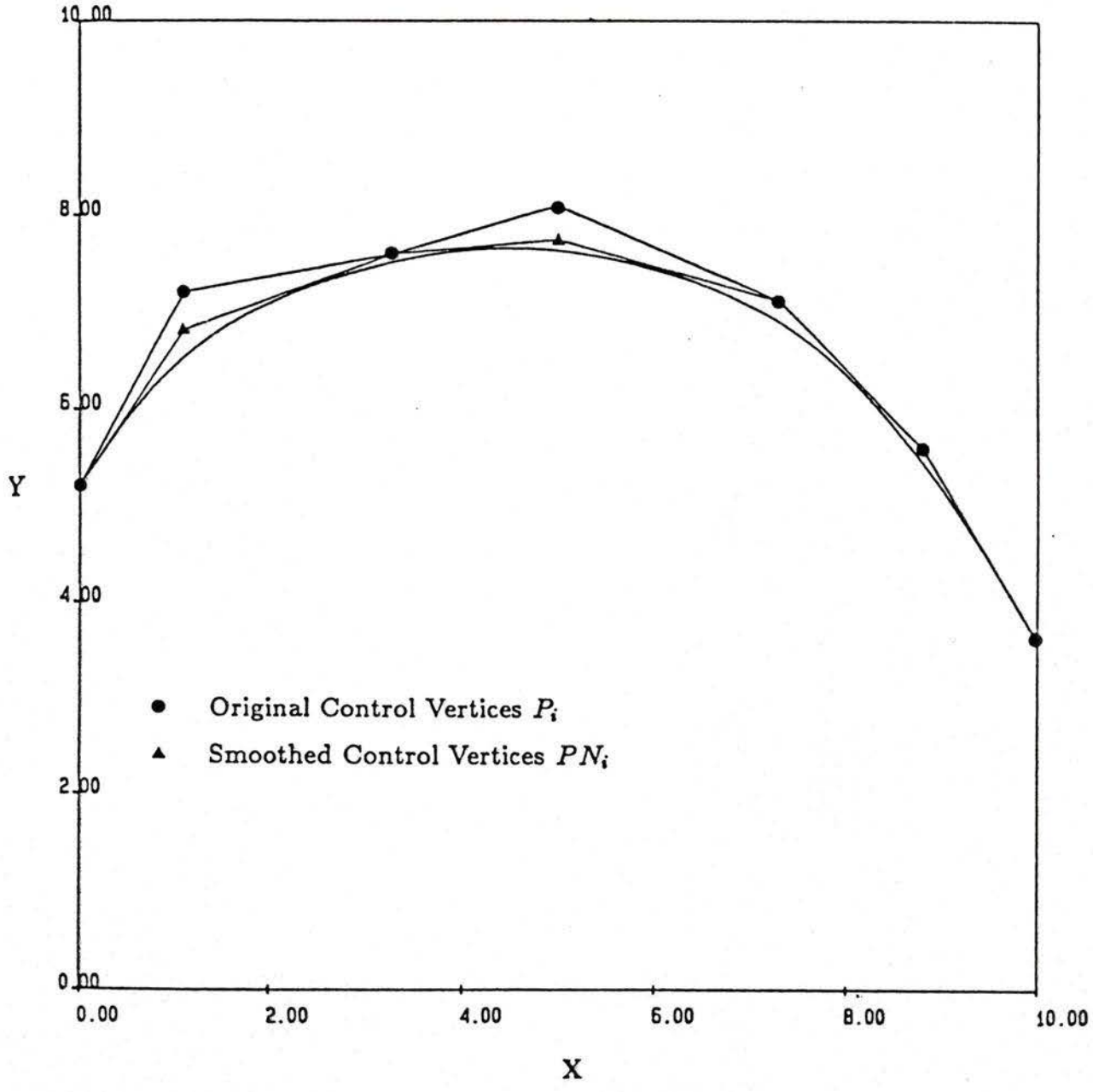


Figure 4.6: A faired curve to the control polygon vertices given in Figure 4.3 with $rate2 \gg rate1$.

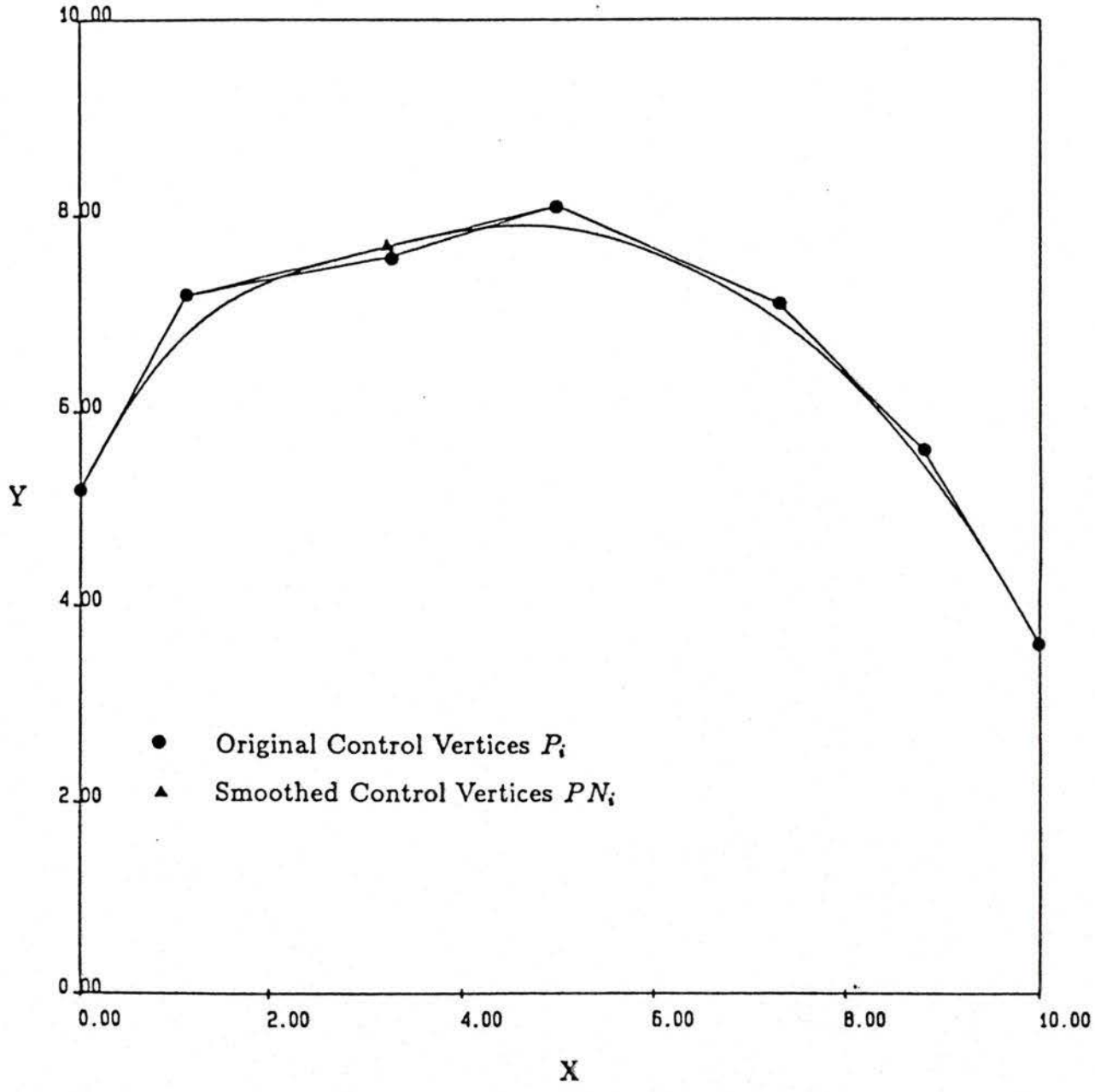


Figure 4.7: A faired curve to the control polygon vertices given in Figure 4.3 with $rate2 \ll rate1$.

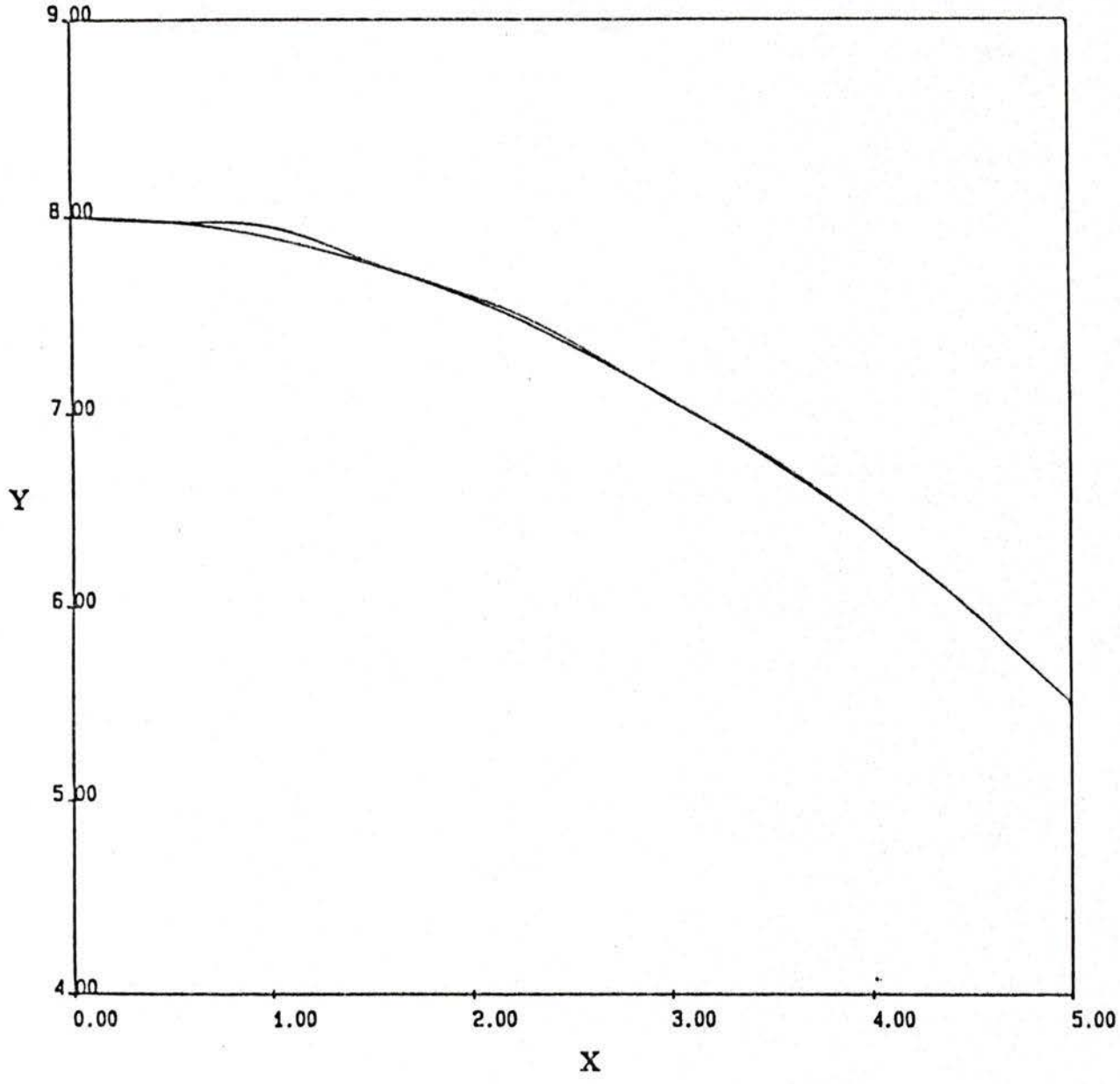


Figure 4.8: A parabola with perturbations and its corrected curve.

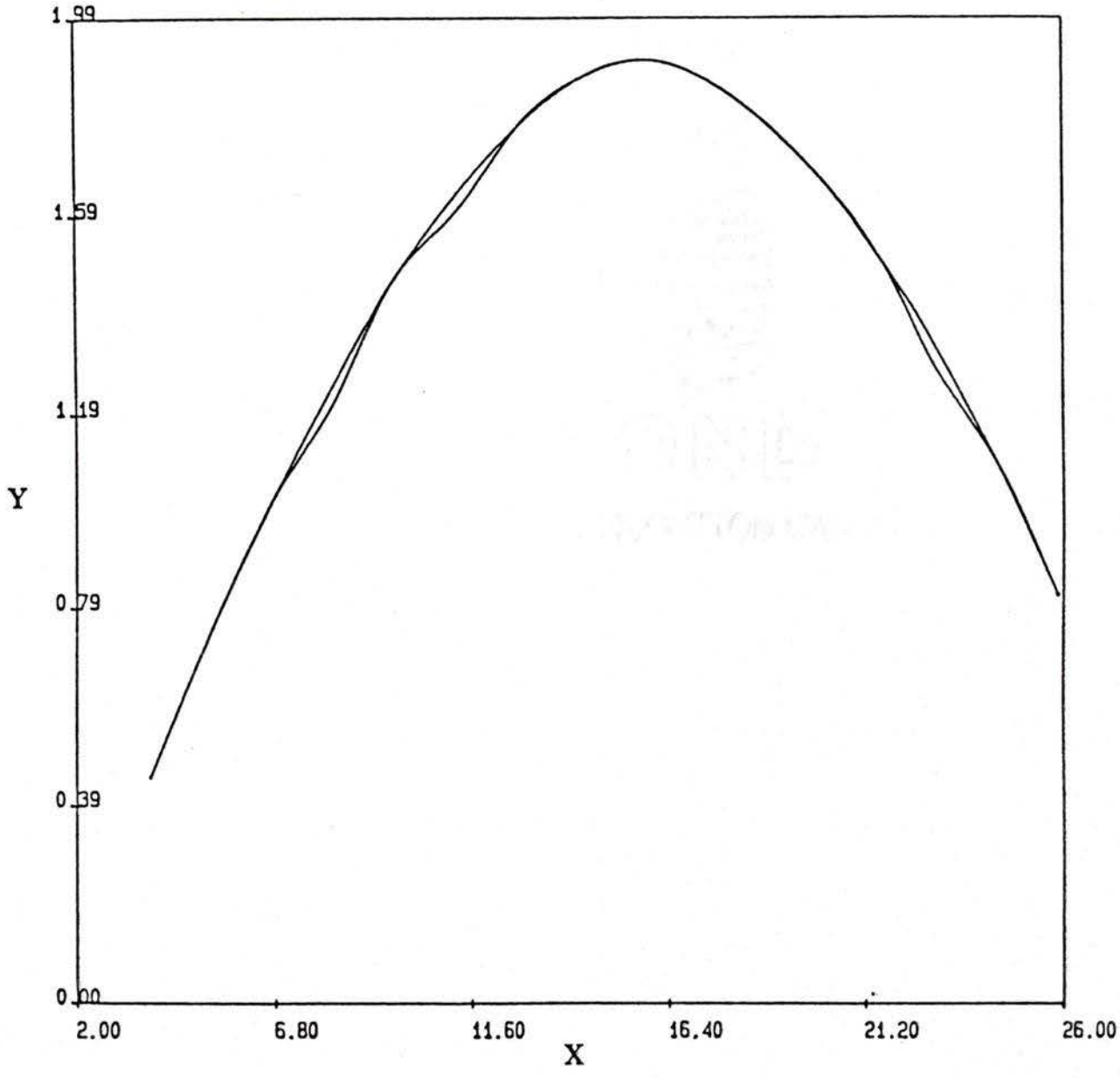


Figure 4.9: Original and improved sheer line of a single kayak hull ($x - y$ view).

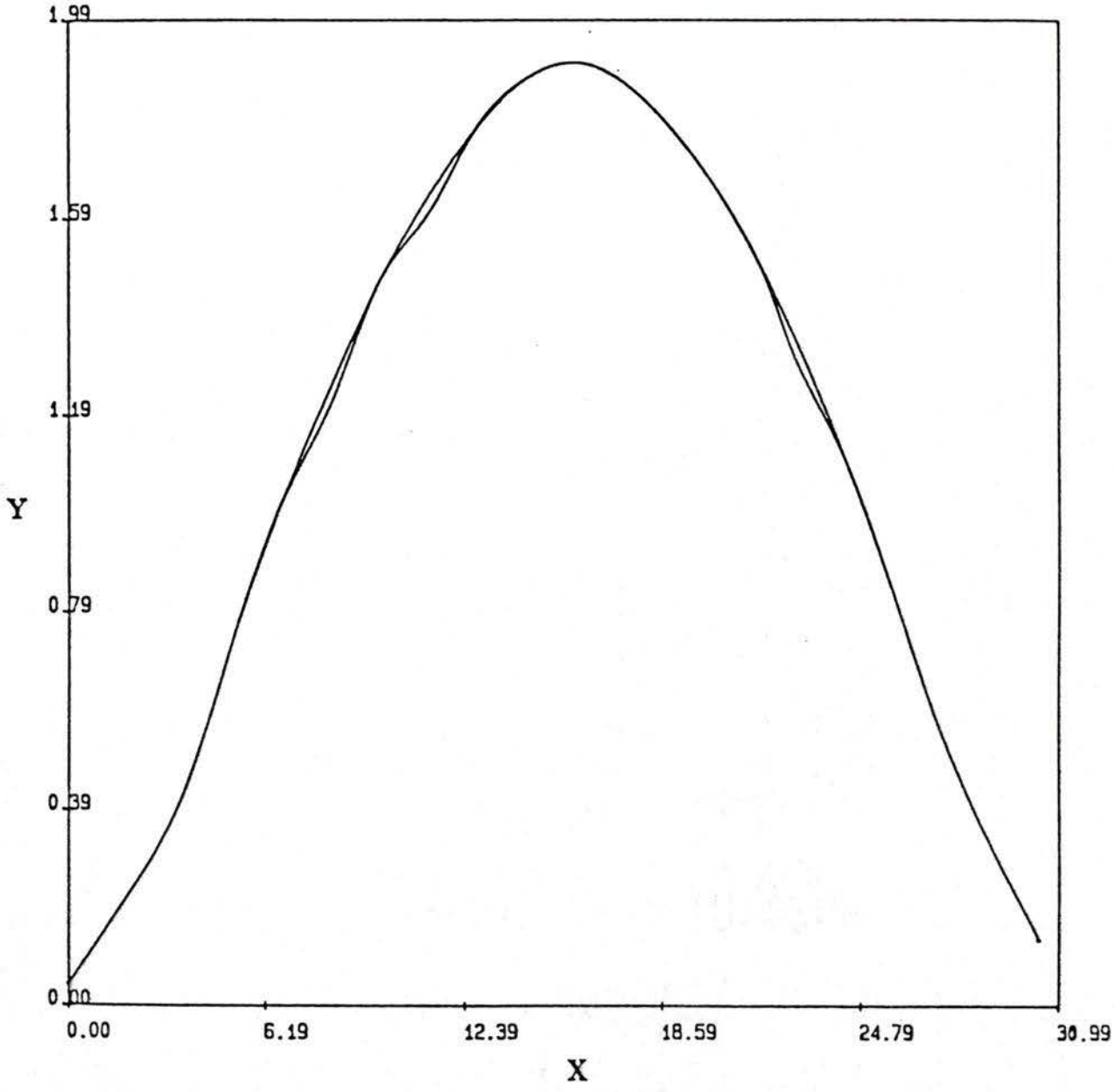


Figure 4.10: Original and improved sheer line of a double kayak hull ($x - y$ view).

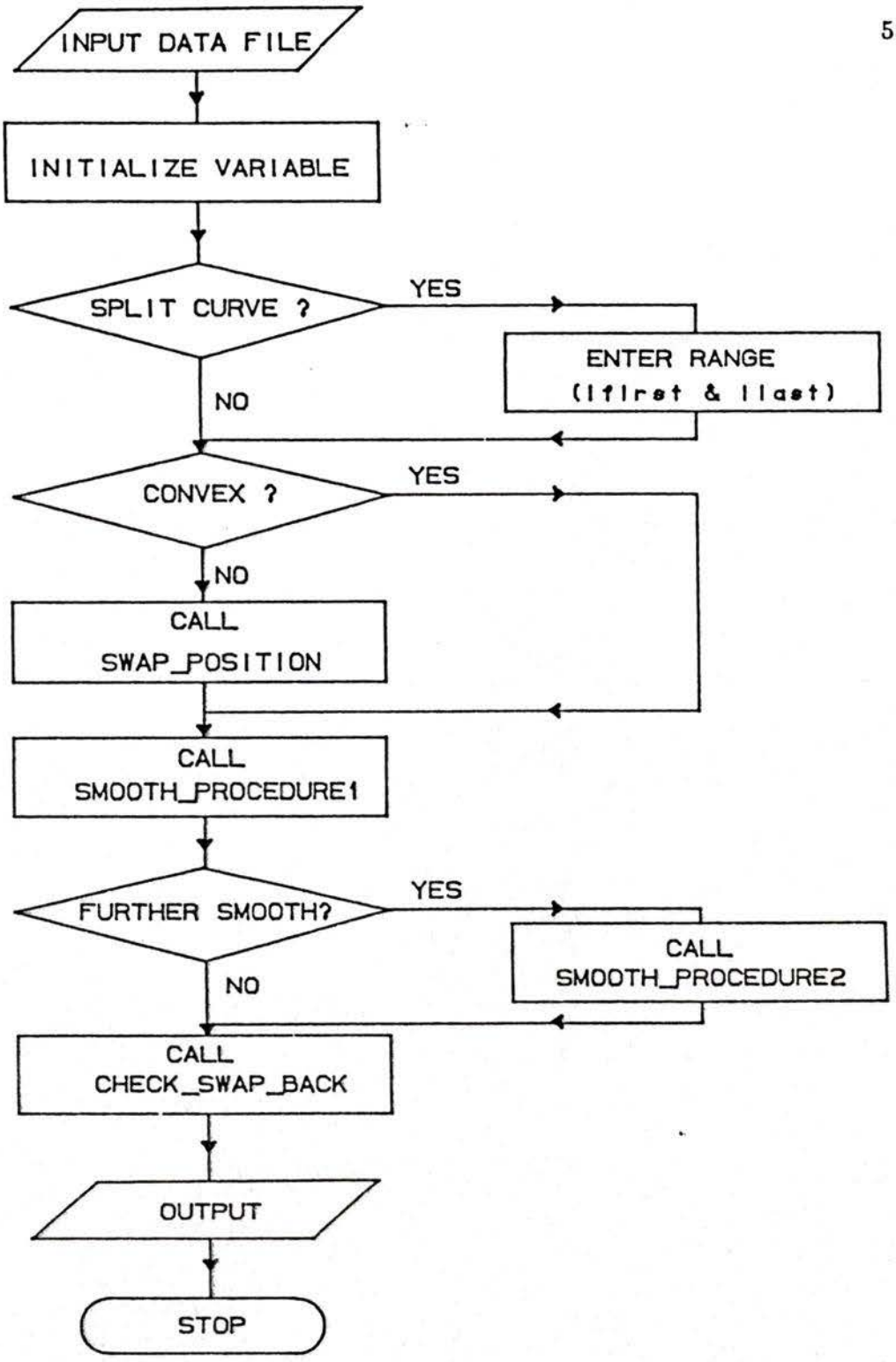


Figure 4.11: Flowchart of the curve fairing program.

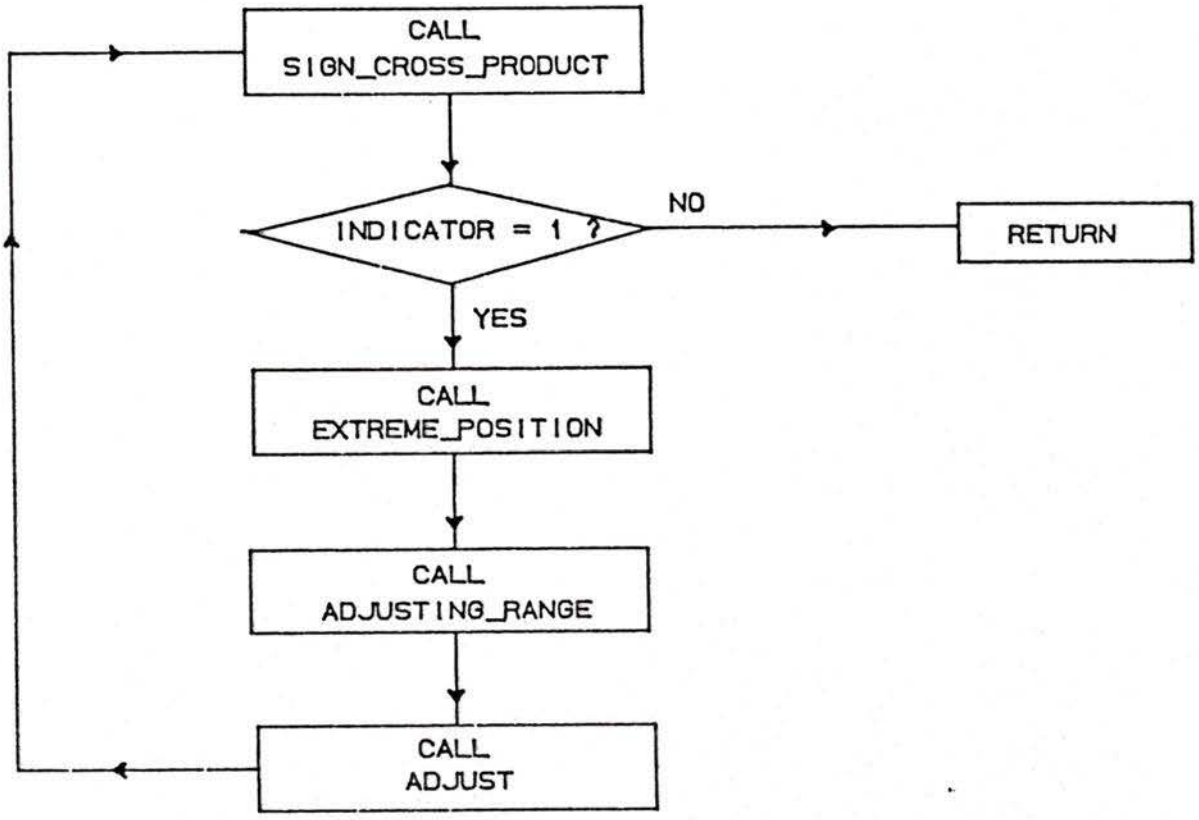


Figure 4.12: Flowchart detailing subroutine (smooth_procedure1).

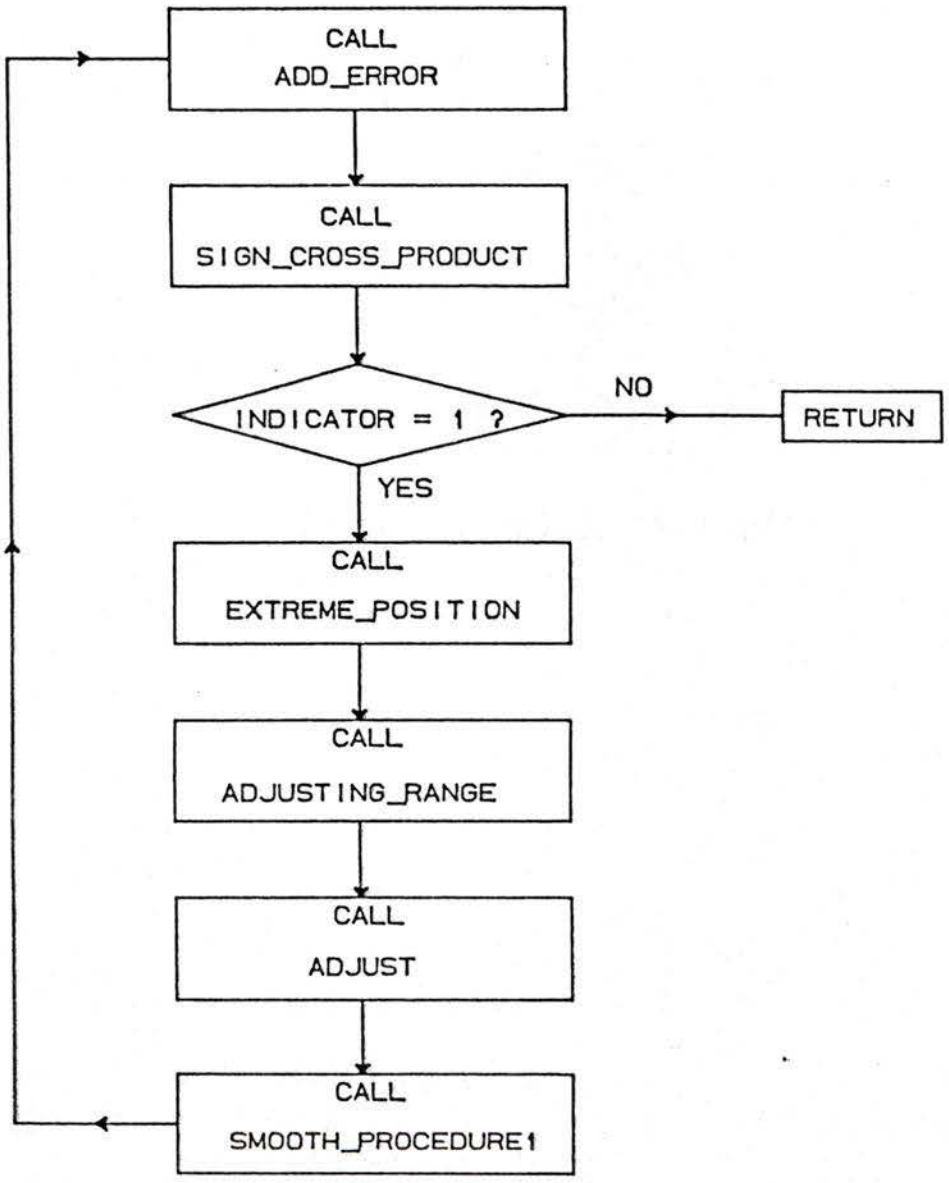


Figure 4.13: Flowchart detailing subroutine (smooth_procedure2).

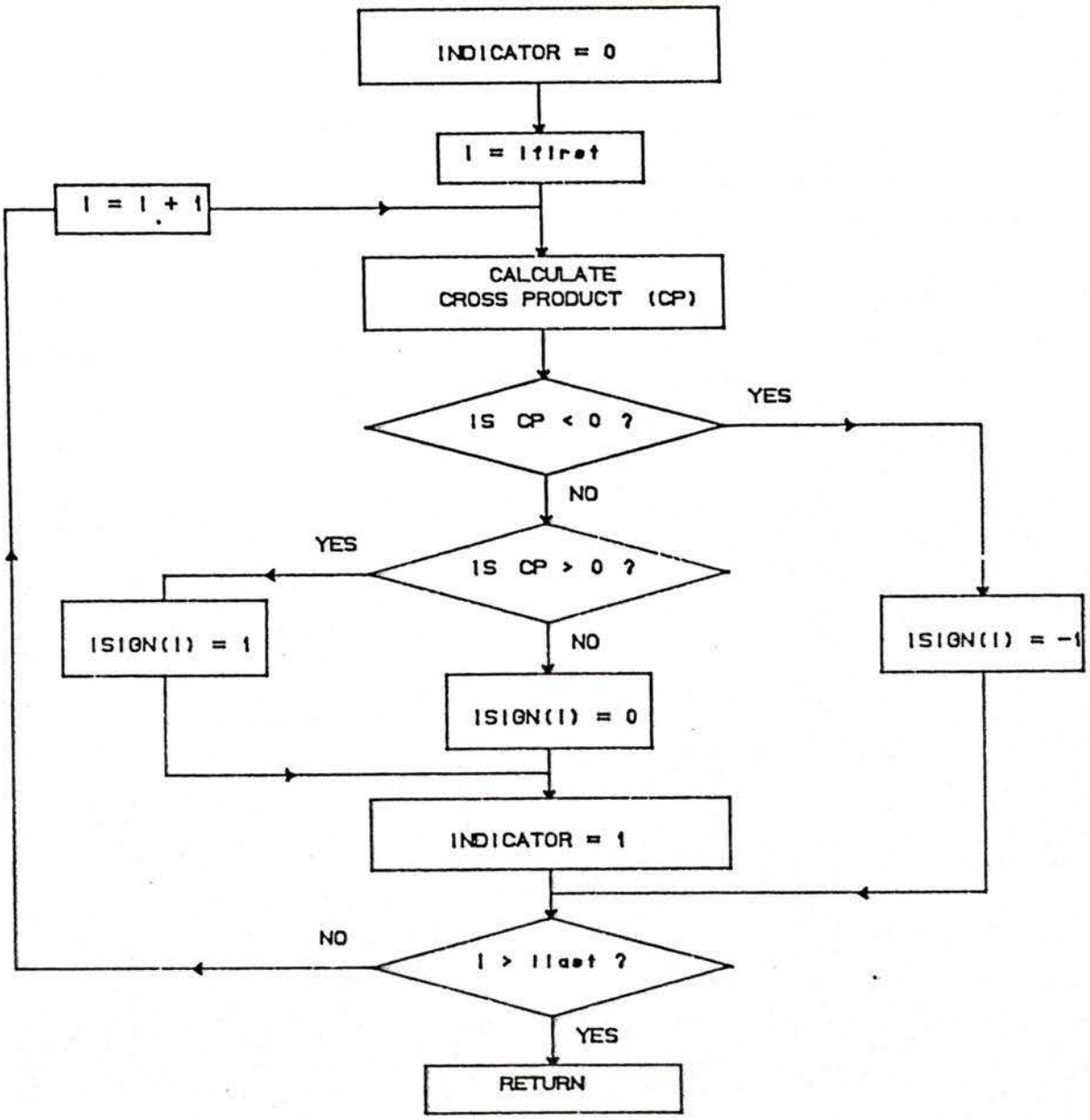


Figure 4.14: Flowchart detailing subroutine (sign_cross_product).

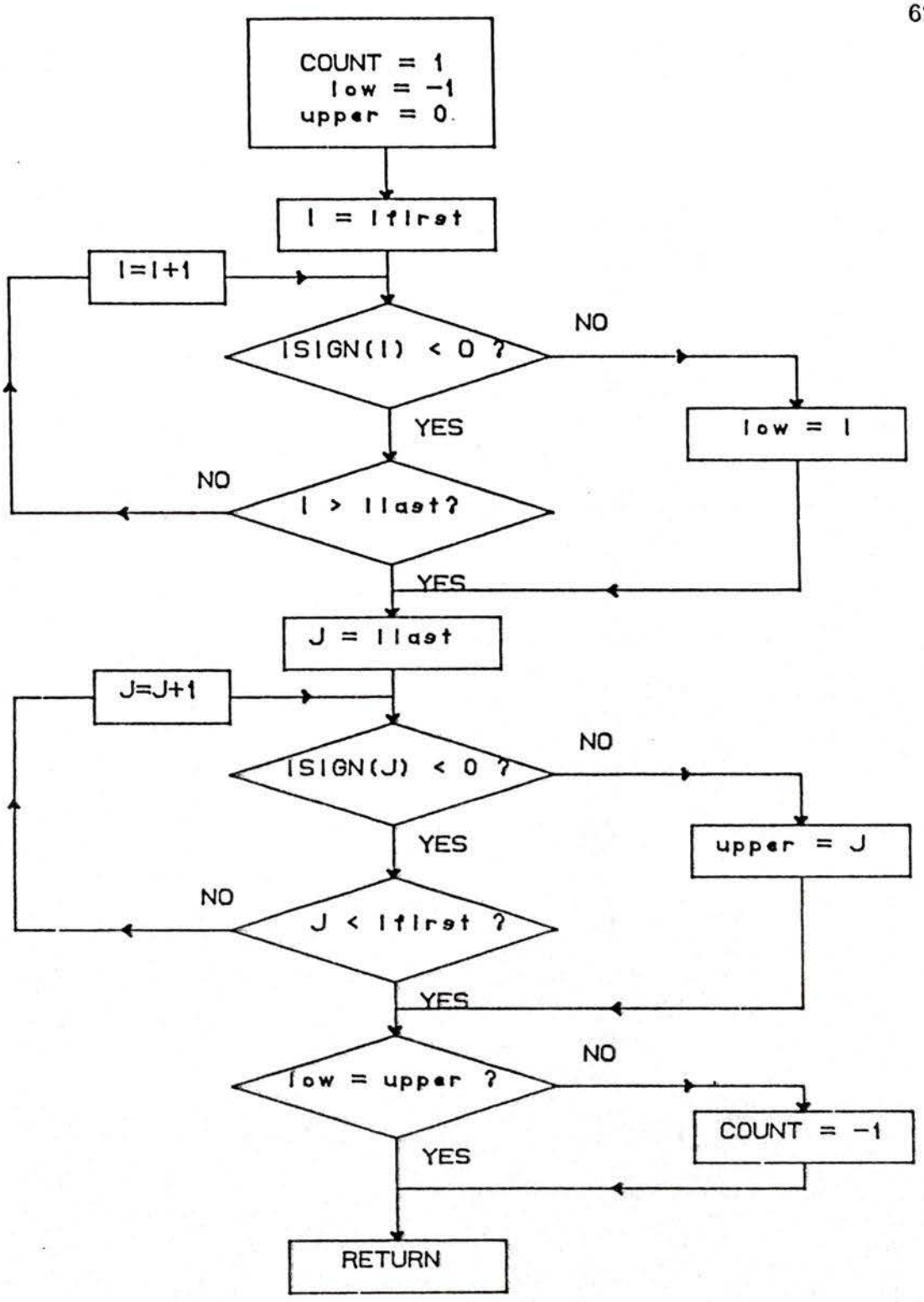


Figure 4.15: Flowchart detailing subroutine (extreme_position).

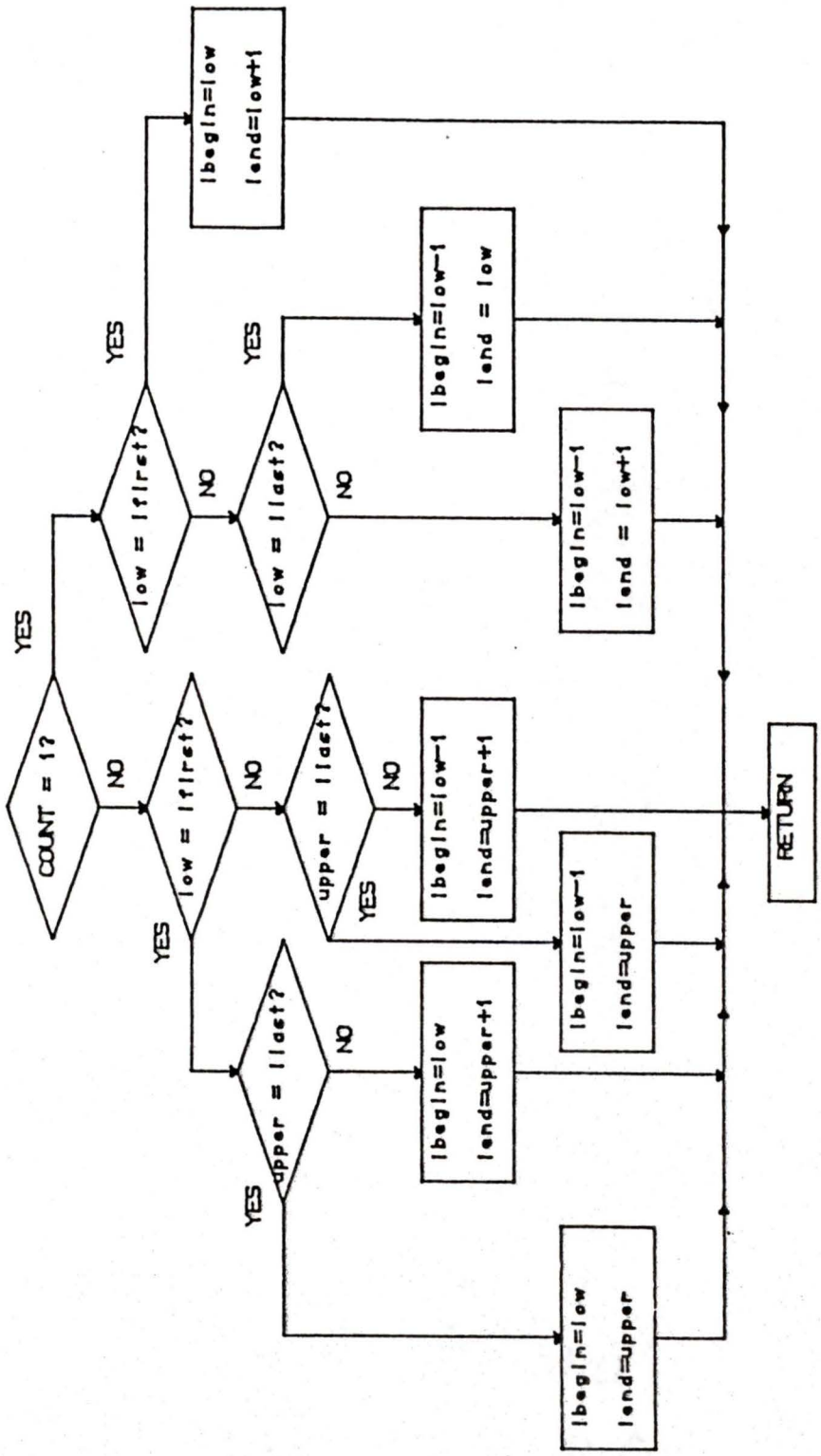


Figure 4.16: Flowchart detailing subroutine (adjusting_range).

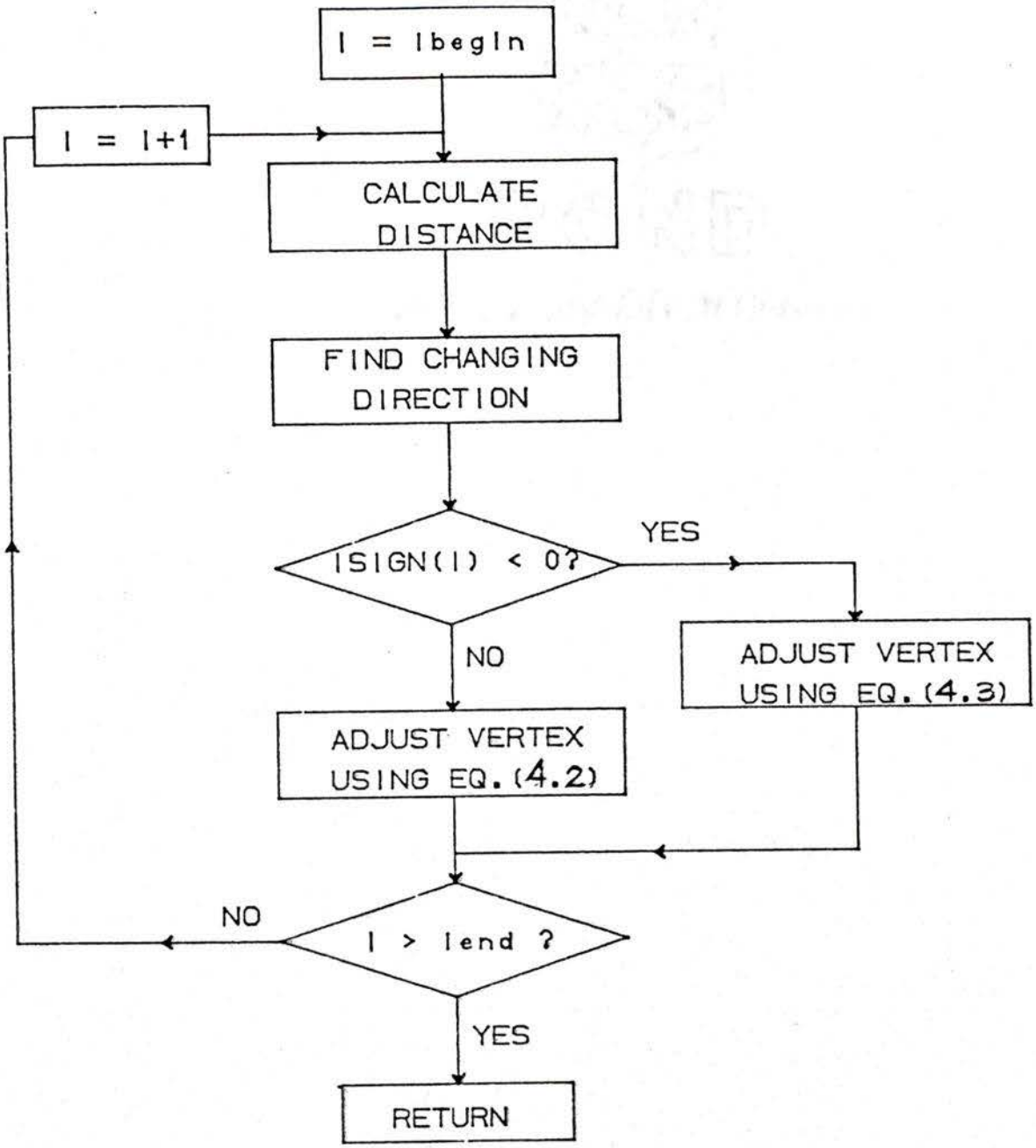


Figure 4.17: Flowchart detailing subroutine (adjust).

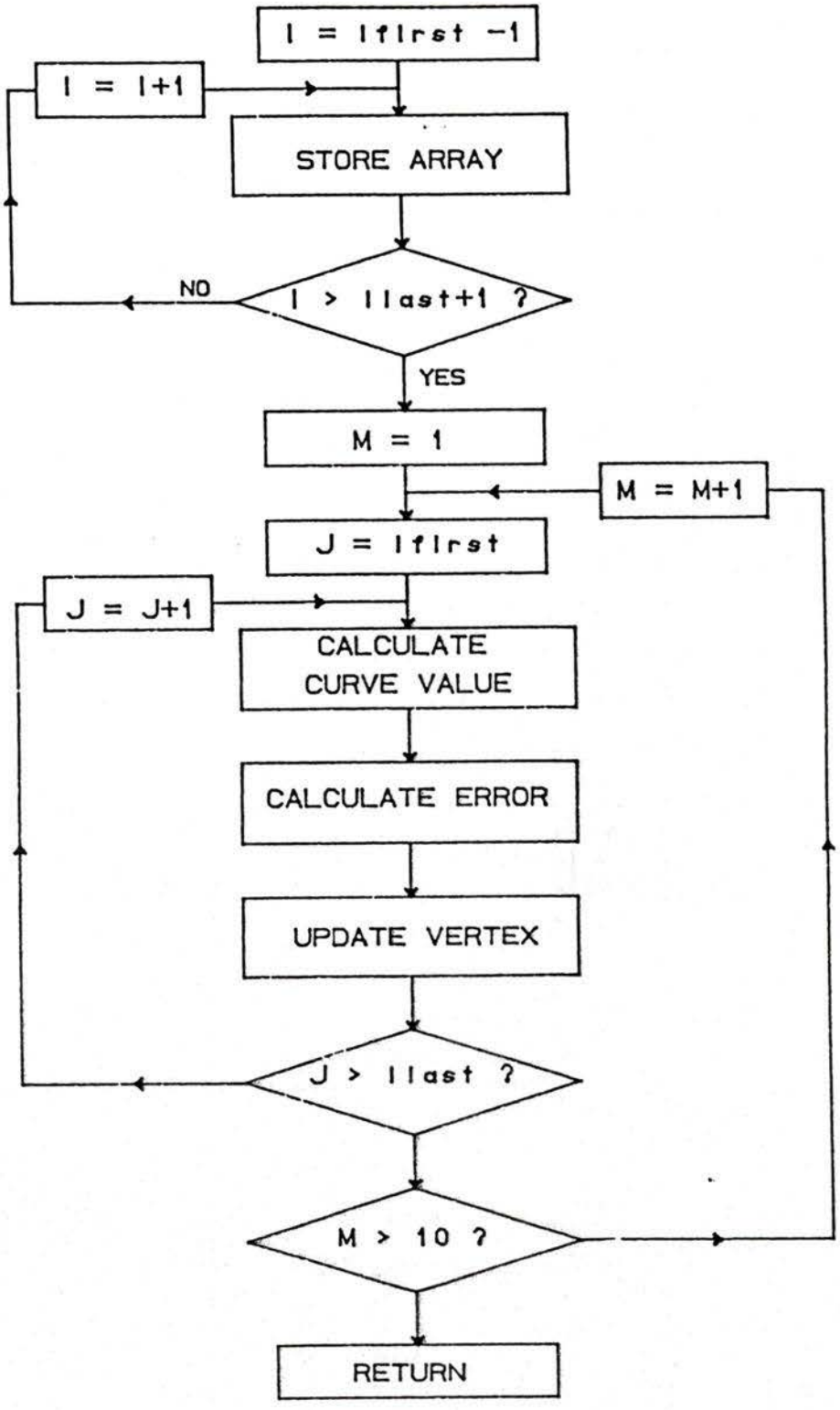


Figure 4.18: Flowchart detailing subroutine (add_error).

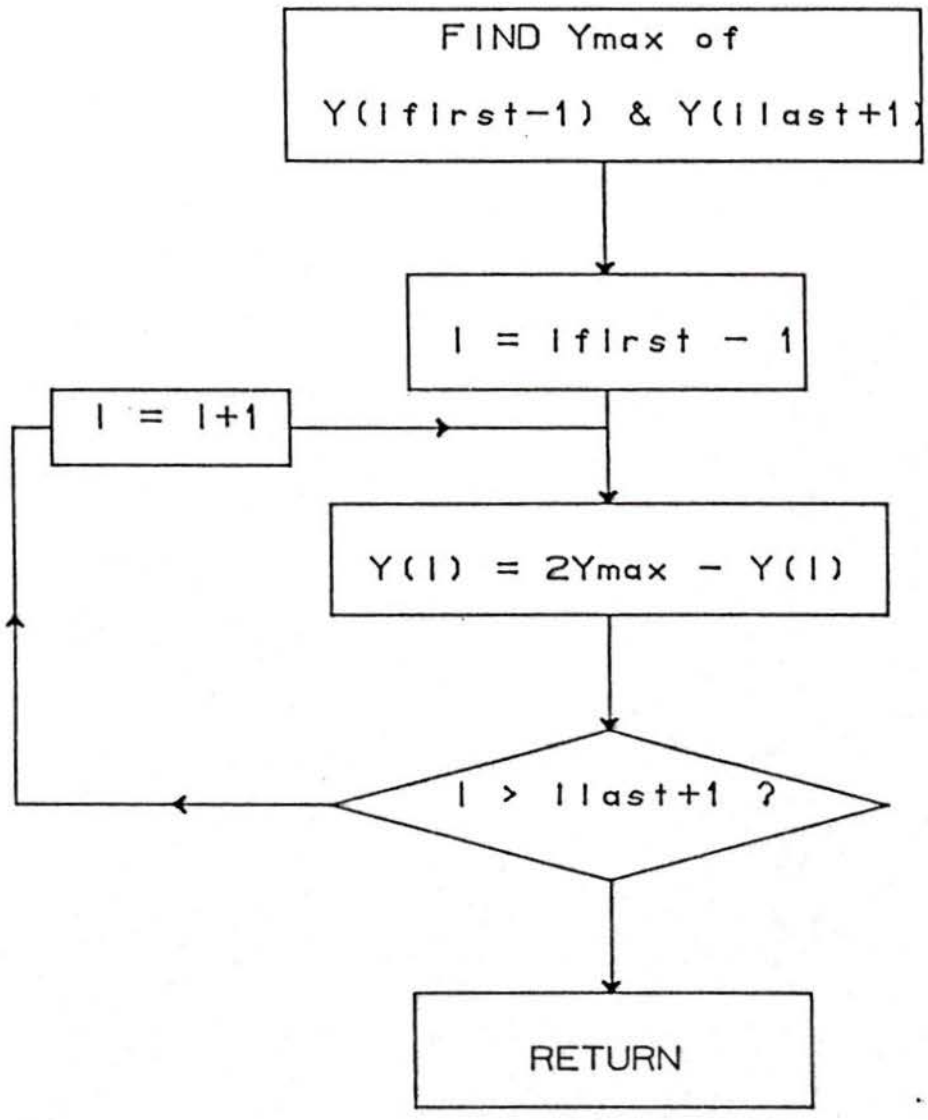


Figure 4.19: Flowchart detailing subroutine (swap_position).

# Chapter 5

# Fairing of Compound Curvature Surfaces

## 5.1 Background Information

The method of detecting and fairing compound curvature surfaces is analogous to the procedures for compound curvature curves given in Chapter 4. The detection of unfairness using the vector cross-product method is replaced by sign of the error between the data vertices and the B-spline approximation at the control vertices. The resulting error signs are represented in a matrix format for further interpretation. This new approach, called the *error sign matrix method*, can also be used for curves.

### 5.1.1 The Definition of the Error Sign Matrix

The parametric B-spline errors, $E_{i,j}$ $(0 \leq i \leq n,\ 0 \leq j \leq m)$, corresponding to the control polyhedron vertices $V_{i,j}$, can be calculated as described in Chapter 3. The elements of the error sign matrix $(S_{i,j})$ are defined as

$$S_{i,j} = \begin{cases} 0 & \text{if } E_{i,j} = 0, \\ 1 & \text{if } E_{i,j} > 0, \\ -1 & \text{if } E_{i,j} < 0. \end{cases} \tag{5.1}$$

Vertices at the four outermost corners of the surface are fixed and the error at these four corners is 0. In order to distinguish them from non-corner '0' elements, the value '$-2$' is assigned to the four corner elements.

Error sign matrix identifies unfair regions of the control polyhedron, which are then modified to obtain a fair surface. A surface has three corresponding error sign matrices, one for each of the three coordinates. If the control polyhedron data vertices are based on linearly spaced $x$, $y$ coordinates, the $z$ error sign matrix is the basis for surface fairing. However, for irregularly spaced $x, y$ data vertices, $x$, $y$ and $z$ error sign matrices form the basis of the surface fairing. For the sake of simplicity, the control polyhedrons with linearly spaced $x$ and $y$ data vertices are discussed first.

In the error sign matrix, the most frequently occurring element value is called $S_{max}$, the second most frequently occurring element value is called $S_{mid}$, and the fewest occurring element value is called $S_{min}$. In Table 5.1 there are 32 elements whose value is 1, three elements whose value is $-1$, and one element whose value is 0. Therefore, $S_{max} = 1$, $S_{mid} = -1$ and $S_{min} = 0$. In the whole matrix, if all the elements except that at the outermost corners take only two values, $S_{mid}$ can be assigned arbitrarily a value of $-3$, as shown in Table 5.2. Furthermore, if the elements have only one value, i.e., a typical even distribution of the error sign matrix as is the case in Table 5.3, then $S_{min}$ can be set arbitrarily to a value of $-2$.

| | | | | | | |
|---|---|---|---|---|---|---|
| -2 | 1 | 1 | 1 | 1 | 1 | -2 |
| 1 | -1 | 1 | 1 | 0 | 1 | 1 |
| 1 | -1 | -1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| -2 | 1 | 1 | 1 | 1 | 1 | -2 |

Table 5.1: Error sign matrix with $S_{min} = 0$, $S_{mid} = -1$ and $S_{max} = 1$.

| | | | | | | |
|---|---|---|---|---|---|---|
| -2 | 1 | 1 | 1 | 1 | 1 | -2 |
| 1 | -1 | 1 | 1 | 1 | 1 | 1 |
| 1 | -1 | -1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| -2 | 1 | 1 | 1 | 1 | 1 | -2 |

Table 5.2: Error sign matrix with $S_{min} = -1$, $S_{mid} = -3$ and $S_{max} = 1$.

| | | | | | | |
|---|---|---|---|---|---|---|
| -2 | 1 | 1 | 1 | 1 | 1 | -2 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| -2 | 1 | 1 | 1 | 1 | 1 | -2 |

Table 5.3: Error sign matrix with $S_{min} = -2$, $S_{mid} = -3$ and $S_{max} = 1$.

### 5.1.2 The Definition of the Sub-Block

A sub-block is defined as a rectangular set of elements that enclose all horizontally and vertically contiguous elements with value $S_{min}$ or $S_{mid}$. The particular value on which the sub-block is based (formed) is called the "*dominant*" element value and is denoted $S_d$. The notion of horizontally and vertically contiguous elements is best explained by considering any two sets of elements $\{ S_{i,j'}, \cdots, S_{i,j}, \cdots, S_{i,j''} \}$ and $\{ S_{i',j}, \cdots, S_{i,j}, \cdots, S_{i'',j} \}$. In the first set, the elements are horizontal neighbors as they have the same value. Similarly, in the second set, the elements are vertical neighbors; but as there exists at least one common element, $S_{i,j}$, between the two sets, they belong to the same sub-block.

Let $w$ be the integer label for the sub-block. The extent of the sub-block $(w)$ is denoted as $Ind_{w,q}$, where $1 \leq q \leq 4$. Every element position, $(i,j)$, in the sub-block satisfies

$$Ind_{w,1} \leq i \leq Ind_{w,2},$$

and

$$Ind_{w,3} \leq j \leq Ind_{w,4}.$$

Furthermore, the number of dominant elements in the sub-block ($w$) is denoted as $Jn_w$.

### 5.1.3 The Definition of Non-Convex Sub-Block

A non-convex sub-block is defined as one in which there are at least two dominant elements surrounding at least one dissimilar element in either horizontal or vertical direction. More precisely, assume that an element in the sub-block is denoted as $S_{i,j}$ ($Ind_{w,1} \leq i \leq Ind_{w,2}$ and $Ind_{w,3} \leq j \leq Ind_{w,4}$). A sub-block is defined as a non-convex form if there exists at least one element, called the non-convex spot, $S_{i,j}$, satisfying the condition $S_{i,j} \neq S_d$; while at least one of the following two conditions hold:

1. There exists $i'$ and $i''$, which are in the range of $Ind_{w,1} \leq i' < i$ and $i < i'' \leq Ind_{w,2}$, and satisfy the equation

$$S_{i',j} = S_{i'',j} = S_d.$$

2. There exists $j'$ and $j''$, which are in the range of $Ind_{w,3} \leq j' < j$ and $j < j'' \leq Ind_{w,4}$, and satisfy the equation

$$S_{i,j'} = S_{i,j''} = S_d.$$

Examples of non-convex sub-blocks are given in Tables 5.4 and 5.5.

## 5.2 The Classification of Matrix Sign Patterns

There is a large but finite number of sign distribution possibilities for any size of matrix. They can be interpretated in many ways, and as with surface fairness itself, there is not a single correct answer. In this work, two main types of matrix sign sub-block were selected as requiring modifications to ensure surface fairness.

*Case 1.* The number of dominant elements in a sub-block are less than or equal to four ( $Jn_w \leq 4$ ), which means that the elements with value $S_d$, are surrounded by the elements with value $S_{max}$. A typical example is shown in Table 5.1.

*Case 2.* The sub-block is non-convex as shown in Table 5.4 and 5.5.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| -2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -2 |
| -1 | -1 | 1 | 1 | -1 | 1 | 1 | 1 | -1 | -1 |
| -1 | -1 | -1 | 1 | 1 | 1 | 1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -2 |

Table 5.4: Error sign matrix with non-convex subblock.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| -2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -2 |
| -1 | -1 | 1 | 1 | 0 | 1 | 1 | 1 | -1 | -1 |
| -1 | -1 | -1 | 1 | 1 | 1 | 1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -2 |

Table 5.5: Error sign matrix with non-convex sub-block.

In order to automatically detect the irregular distribution states, it is necessary to first separate the different sub-blocks, then to decide whether or not to make modifications.

## 5.3 Recursive Sub-Block Definition

A recursive method is used to define sub-blocks. To explain the method clearly, Table 5.6 is used as an example. Here $S_d$ is equal to $S_{min}$, which is equal to -1. The matrix elements, $S_{i,j}$'s, which satisfy the condition, $S_{i,j} = S_d$, distribute into four sub-blocks. In order to separate these four

sub-blocks, the simplest way is to assign different values to the dominant elements in the four sub-blocks. A convenient way to assign the values is as follows.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| -2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -2 |
| -1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 |
| -1 | 1 | 1 | 1 | -1 | -1 | 1 | 1 | 1 | -1 |
| -1 | 1 | 1 | 1 | -1 | 1 | 1 | 1 | 1 | -1 |
| -1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 |
| 1 | 1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 | 1 |
| -2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -2 |

Table 5.6: Error sign matrix before separating different sub-blocks.

Assume in a sub-block, $w$ (in the particular case given in Table 5.6 $1 \leq w \leq 4$), the dominant elements, $S_{i,j}$'s, have the value of $S_d$, then $S_{i,j}$'s can be reassigned with values of $(w+1)$, that is,

$$S_{i,j} = (w+1). \tag{5.2}$$

The matrix, after separating different sub-blocks, is given in Table 5.7.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| -2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -2 |
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 |
| 2 | 1 | 1 | 1 | 4 | 4 | 1 | 1 | 1 | 3 |
| 2 | 1 | 1 | 1 | 4 | 1 | 1 | 1 | 1 | 3 |
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 5 | 5 | 5 | 5 | 5 | 1 | 1 |
| 1 | 1 | 5 | 5 | 5 | 1 | 5 | 5 | 5 | 1 |
| -2 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | -2 |

Table 5.7: Error sign matrix after separating different sub-blocks.

The first step in the recursive search procedure is to check whether or not the current position mark, $(i,j)$, is in the range of $0 \leq i \leq n$ and $0 \leq j \leq m$, and if $S_{i,j}$ is equal to $S_d$. If either condition is violated, the current search procedure terminates. Otherwise, $S_{i,j}$ is given a new value

using the formula (5.2). Taking $S_{i,j}$ as a base, expansions are made in four directions (lower, left, upper, right) to determine whether or not the values of $S_{i-1,j}$, $S_{i,j-1}$, $S_{i+1,j}$ and $S_{i,j+1}$ are equal to $S_d$. If it is found that one of them is equal to $S_d$, then the particular element value is changed to $(w+1)$. Following, this new position is taken as a base, and the same expansions are made again. The above procedure is repeated, until no more expansions can be made. In this way, a new value $(w+1)$ is assigned to every dominant element in the sub-block $(w)$.

The automatic separating sub-block method is based on linear scanning the whole matrix, finding a element, $S_{i,j}$, that satisfies the condition, $S_{i,j} = S_d$, setting the sub-block mark, $w$, and calling the search subroutine. Thus the different values can be assigned to the dominant elements in the different sub-blocks. In addition, the extreme position marks, $Ind_{w,q}$, of each different sub-block are determined.

The defining sub-block and search procedures are given respectively in Algorithms 5.1 and 5.2.

*Algorithm 5.1* ( subroutine *defining sub-block* )

- Step 1. Initialize, set $w = 1$.
- Step 2. For $i = 0$ to $n$, do
  - Step 2.1 For $j = 0$ to $m$, do
    - Step 2.1.1. Determine if $S_{i,j} = S_d$. If *yes*, set $w = w + 1$; else, go back to Step 2.1.
    - Step 2.1.2. Call subroutine *search* $(i, j, w)$ (as follow).
- Step 3. For $i = 1$ to $w - 1$,
  initialize extreme marks, $Ind_{i,q}$ $(1 \leq q \leq 4)$ and the dominant joining numbers, $Jn_i$, as

$$Ind_{i,1} = 9999,$$

$$Ind_{i,2} = 0,$$

$$Ind_{i,3} = 9999,$$

$$Ind_{i,4} = 0,$$

$$Jn_i = 0,$$

- Step 4. For $i = 0$ to $n$, do
  - Step 4.1. For $j = 0$ to $m$, do
    * Step 4.1.1. Determine if $S_{i,j} \geq 2$. If false, go back to Step 4.1;
      else, set $w = S_{i,j} - 1$ and $Jn_w = Jn_w + 1$.
    * Step 4.1.2. Determine if $i < Ind_{w,1}$. If true, set $Ind_{w,1} = i$.
    * Step 4.1.3. Determine if $i > Ind_{w,2}$. If true, set $Ind_{w,2} = i$.
    * Step 4.1.4. Determine if $j < Ind_{w,3}$. If true, set $Ind_{w,3} = j$.
    * Step 4.1.5. Determine if $j > Ind_{w,4}$. If true, set $Ind_{w,4} = j$.

*Algorithm 5.2* ( subroutine *search* $(i, j, w)$)

- Step 1. Determine if $0 \leq i \leq n$. If false, go to Step 9.
- Step 2. Determine if $0 \leq j \leq m$. If false, go to Step 9.
- Step 3. Determine if $S_{i,j} = S_d$. If false, go to Step 9.
- Step 4. Set $S_{i,j} = w$.
- Step 5. Based on the position mark of $(i-1, j)$, call *search* $(i-1, j, w)$.
- Step 6. Based on the position mark of $(i, j-1)$, call *search* $(i, j-1, w)$.
- Step 7. Based on the position mark of $(i+1, j)$, call *search* $(i+1, j, w)$.
- Step 8. Based on the position mark of $(i, j+1)$, call *search* $(i, j+1, w)$.
- Step 9. Return.

## 5.4 Local Modifications

Once the sub-blocks are identified, polyhedron vertices may need to be changed. With *Case 1* sub-block, it is likely that all the dominant elements of the sub-block require adjustment so that a matrix of single sign elements exists. With *Case 2*, it is likely that any minority elements of the sub-block surrounded by the dominant elements of the sub-block should be changed, thus giving an even variation in the sub-block.

The user is given the option of adjusting each sub-block previously identified. This is a manual override to the program that enables certain irregular feature of the surface to be retained if required.

### 5.4.1 Range Location

In either of the above cases, if a vertex $V_{i,j}$ needs adjustment, the neighboring points are also modified to account for the local support of the B-spline basis functions. The range of the neighborhood that needs adjustment are denoted by $istart$, $iend$, $jstart$ and $jend$, that is, any position mark, $(i', j')$, in the range, satisfies the following conditions

$$istart \leq i' \leq iend,$$

and

$$jstart \leq j' \leq jend.$$

According to the position of $(i, j)$, there are five situations to determine the range, they are,

- if $i = 0$, set $istart = i$, $iend = i + 1$, $jstart = j - 1$ and $jend = j + 1$.
- if $j = 0$, set $istart = i - 1$, $iend = i + 1$, $jstart = j$ and $jend = j + 1$.
- if $i = n$, set $istart = i - 1$, $iend = i$, $jstart = j - 1$ and $jend = j + 1$.
- if $j = m$, set $istart = i - 1$, $iend = i + 1$, $jstart = j - 1$ and $jend = j$.

- if $i \neq 0$, $i \neq n$, $j \neq 0$ and $j \neq m$, set $istart = i - 1$, $iend = i + 1$, $jstart = j - 1$ and $jend = j - 1$.

### 5.4.2 Adjusting Unfair Vertices

The amount of adjustment to vertices within the above range is discussed below. For convenience, the following terms are denoted first:

- $SUM_e$ is the sum of the absolute errors in the range,
- $SUM_v$ is the sum of the absolute parametric vertex values in the range,
- $C_r$ is the average relative change in the vertices set by the user,
- $R_{ev}$ is the ratio between $SUM_e$ and $SUM_v$,
- $Coeff$ is the ratio between $C_r$ and $R_{ev}$, and
- $D$ is the distance between an arbitrary position mark, $(i', j')$, in the range and the center, $(i, j)$, of the range, that is

$$D = Abs(i' - i) + Abs(j' - j). \tag{5.3}$$

The first step in the modification is to set the value of $C_r$, e.g., assume that the surface errors are in the range between 1% and 0.1%, then $C_r$ can be set reasonably to 0.1%. Next, the values of $SUM_e$ and $SUM_v$ are calculated, and $R_{ev}$ and $Coeff$ are obtained using the following formulas, (5.4) and (5.5) as

$$R_{ev} = \frac{SUM_e}{SUM_v}, \tag{5.4}$$

$$Coeff = \frac{C_r}{R_{ev}}. \tag{5.5}$$

The modification made to a arbitrary vertex point, $V_{i',j'}$, can be computed as follows,

$$VN_{i',j'} = V_{i',j'} - Weight \times Coeff \times E_{i',j'} \tag{5.6}$$

where $VN_{i',j'}$ is the new control polyhedron vertex after a correction is made to $V_{i',j'}$, and $Weight$ is determined by the value of $D$. A set of commonly used value setting is given as

$$Weight = \begin{cases} 1 & \text{if } D = 0, \\ 0.6 & \text{if } D = 1, \\ 0.1 & \text{if } D = 2. \end{cases} \tag{5.7}$$

From formulas (5.6) and (5.7), it is found that for the fixed values of $Weight$ and $Coeff$, the amount of correction made to a vertex is proportional to the error at that position. This enables the elimination of undesirable "peaks" and "valleys" on the resulting surface. As the modifications are focused on the center of the range, the corrections made to the neighbouring points are appropriately relaxed. The parameter, $Weight$, determines the relaxation at various positions in the range.

### 5.4.3 Relevant Algorithms

For the sake of completeness, the algorithms to implement the above procedures are given below.

*Algorithm 5.3* (subroutine *classify*)

- For $ll = 2$ to $w$,
  - Determine if $Jn_{ll-1} > 4$. If true, call subroutine *case 1*; else, call subroutine *case 2*.

*Algorithm 5.4* (subroutine *case 1*)

- Step 1. Determine if the current sub-block needs modification; if *yes*, then

- Step 1.1. For $i = Ind_{ll-1,1}$ to $Ind_{ll-1,2}$, do
  - Step 1.1.1. For $j = Ind_{ll-1,3}$ to $Ind_{ll-1,4}$, do
    - Step 1.1.1a. Determine if $S_{i,j} < 2$. If true, go back to Step 1.1.1;
      else, call subroutine *range* and set *repeat* = 1.

*Algorithm 5.5* (subroutine *case 2*)

- Step 1. For $i = Ind_{ll-1,1}$ to $Ind_{ll-1,2}$, do
  - Step 1.1 For $j = Ind_{ll-1,3}$ to $Ind_{ll-1,4}$, do
    - Step 1.1.1. Determine if $S_{i,j} < 2$. If false, go back to Step 1.1.
    - Step 1.1.2 For $jj = j$ to $Ind_{ll-1,3}$ increment $-1$, do
      - Step 1.1.2.A. Determine if $S_{i,jj} = ll$. If false, go back to Step 1.1.2.
      - Step 1.1.2.B. For $jj = j$ to $Ind_{ll-1,4}$, do
      - Step 1.1.2.B1. Determine if $S_{i,jj} = ll$. If false, go back to Step 1.1.2.B;
        else, call subroutine *range*, set *repeat* = 1 and go to Step 1.1.
    - Step 1.1.3 For $ii = i$ to $Ind_{ll-1,1}$ increment $-1$, do
      - Step 1.1.3.A. Determine if $S_{ii,j} = ll$. If false, go back to Step 1.1.3.
      - Step 1.1.3.B. For $ii = i$ to $Ind_{ll-1,2}$, do
      - Step 1.1.3.B1. Determine if $S_{ii,j} = ll$. If false, go back to Step 1.1.3.B;
        else, call subroutine *range*, set *repeat* = 1 and go to Step 1.1.

*Algorithm 5.6* (subroutine *range*)

- Step 1. Determine if $E_{i,j} = 0$. If true, calculate the average error in the range, update the vertex by adding this average error, and go to Step 8.
- Step 2. Determine if $i = 0$. If true, set $istart = i$, $iend = i + 1$, $jstart = j - 1$, $jend = j + 1$ and go to Step 7.
- Step 3. Determine if $j = 0$. If true, set $istart = i - 1$, $iend = i + 1$, $jstart = j$, $jend = j + 1$ and go to Step 7.
- Step 4. Determine if $i = n$. If true, set $istart = i - 1$, $iend = i$, $jstart = j - 1$, $jend = j + 1$ and go to Step 7.
- Step 5. Determine if $j = m$. If true, set $istart = i - 1$, $iend = i + 1$, $jstart = j - 1$, $jend = j$ and go to Step 7.
- Step 6. If $i \neq 0$, $j \neq 0$, $i \neq n$ and $j \neq m$, set $istart = i - 1$, $iend = i + 1$, $jstart = j - 1$, $jend = j + 1$.
- Step 7. Call subroutine *modify*.
- Step 8. Return.

*Algorithm 5.7* (subroutine *modify*)

- Step 1. Initialize variables and set $C_r$.
- Step 2. Calculate sum ($SUM_e$) of the absolute errors, $E_{i,j}$'s, and sum ($SUM_v$) of the absolute vertices, $V_{i,j}$'s, where ($istart \leq i \leq iend$ and $jstart \leq j \leq jend$).
- Step 3. Compute values of $R_{ev}$ and $Coeff$ using the formulas (5.4) and (5.5).

- Step 4. For $ii = istart$ to $iend$, do
  - Step 4.1. For $jj = jstart$ to $jend$, do
    * Step 4.1.1. modify vertex using equation (5.6), (5.7) and (5.3).

## 5.5 B-spline Approximation Smoothing Procedure

The first step in the fairing procedure is to calculate the error at the vertex, $V_{i,j}$ $(0 \leq i \leq n,\ 0 \leq j \leq m)$, using the method described in Chapter 3, and obtain the corresponding error sign matrix. Then the values of $S_{min}$, $S_{mid}$ and $S_{max}$ are computed. If $S_{min}$ is equal to $-2$, then by definition, it means that the matrix has a regular distribution, as shown in Table 5.3, and the fairing procedure is terminated. If $S_{mid}$ is equal to $-3$, that is, two values occur in the matrix as shown in Table 5.2 (the elements at the most outside four corners are not considered here), then $S_d$ can be set equal to $S_{min}$. The sub-block identification procedures are then performed based on the dominant elements, and the type of the irregularities of the different sub-blocks are classified, and the corresponding modifications are made. If $S_{min} \neq -2$ and $S_{mid} \neq -3$, that is, three values occur in the matrix, as the example given in Table 5.1, then $S_d$ is set at first equal to $S_{mid}$ and the procedures just described are followed. Next, $S_d$ is set equal to $S_{min}$, and the same procedures are executed. The total procedures described above are repeated until all the irregular distributions of the vertex error sign matrix disappear.

## 5.6 B-spline Interpolation Smoothing Procedure

In order to do B-spline interpolation, a given set of data points is smoothed by the procedure described above. Once the final set of control polyhedron vertices is obtained, the B-spline interpolation control polyhedron vertices, $PT$'s, are calculated using the error adding technique described in Chapter 3. This set of control vertices is again checked for smoothness and unfair region is modified in the original set of vertices. It is observed that error sign matrix of this new set of control vertices enhances any potential undulations that might exist within the original vertices. To ensure that the result from the second correction does not disturb the fairness obtained through the first fairing procedure, the newly modified vertices go through the first fairing process again. The whole procedure is repeated until all the irregular distributions of this new vertex error sign matrix disappear.

## 5.7 Other Remarks

The fairing procedure described above can modify the undesired regions while preserving the feature of the original surface. Additionally, in this method, a range for a sub-block can be defined by the user to deal with unforeseen situations. This feature is useful for smoothing data with large errors. In Table 5.8, although the number of contiguous dominant elements are greater than four and the sub-block is convex, the manual override feature can be used to fair the control polyhedron and obtain the total convex surface.

If $x$, $y$ coordinates of the data vertices of the control polyhedron are sparsely spaced, there are two possible ways to smooth the surface. One is to check and modify the error sign matrices of $x$, $y$ and $z$ simultaneously. The second is to first compute the surface interpolating the original data

vertices, then to obtain a new set of control vertices, which have linearly spaced $x$, $y$ coordinates. Next, a new error sign matrix (in $z$) is calculated and the fairing procedure described above is performed.

## 5.8 Surface Smoothing Illustration

During the course of this work, many examples were tested to demonstrate the effectiveness of the technique. Four representative examples are discussed below.

In Figure 5.1, the control polyhedron vertices before the correction are derived from a known analytical function, and a small error was introduced at one vertex in $z$ coordinate. Table 5.9 shows the $z$ error sign matrix. It can be seen that the whole matrix has the same value except at one element. The opposite error sign indicates the position of the unfair spot. The control polyhedron vertex after the adjustments eliminate the error, and the new error sign matrix is given in Table 5.10.

The control polyhedron vertices given in Figure 5.2 were also obtained from a known analytical equation, and small perturbations were introduced at two vertices. As one can see from the the error sign matrix shown in Table 5.11, there are five sub-blocks. Two sub-blocks with dominant elements value 2 and 4 are modified. The local corrections are made iteratively only in these two sub-blocks, until the irregularities disappear. The error sign

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -2 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | -1 | -1 | -1 | 1 | 1 | 1 | 1 |
| 1 | 1 | -1 | -1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| -2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -2 |

Table 5.8: Error sign matrix with irregular distribution.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -2 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| -2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -2 |

Table 5.9: Error sign matrix with irregular distribution.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -2 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| -2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -2 |

Table 5.10: Error sign matrix after correction is made to Table 5.9.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -2 | -1 | 2 | 2 | 2 | -1 | -1 | -1 | -2 |
| 2 | 2 | 2 | 2 | 2 | 2 | -1 | -1 | 3 |
| 2 | 2 | -1 | 2 | 2 | -1 | -1 | -1 | 3 |
| -1 | 2 | 2 | 2 | 2 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | 2 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | 4 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | 5 | 5 | -1 | -1 | -1 | -1 | -1 |
| 5 | 5 | 5 | 5 | 5 | -1 | -1 | -1 | 6 |
| -2 | -1 | 5 | 5 | 5 | -1 | -1 | -1 | -2 |

Table 5.11: Error sign matrix with irregular distribution.

matrix after modification is shown in Table 5.12.

The original data vertices given in Figure 5.3 are developed by linearly blending a parabola and a straight line. Errors are introduced at certain vertices. The corresponding error sign matrix have four sub-blocks, in which two sub-blocks, whose dominant elements are 4 and 5, are classified as need modifications. Appropriate corrections are made iteratively and the new error matrix is given in Table 5.14.

The fairing procedure developed in this chapter was used to design the surface of a kayak. Data points were digitized from rough sketch showing the feature of the hull. Figure 5.4 shows the front part of the kayak. As surface data was unevenly spaced in $y$ and $z$, their error matrices were faired and the modified polyhedron was used to generate the hull surface. The single and double kayak hulls (half) are shown in Figure 5.5 and 5.6 respectively.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -2 | -1 | 2 | 2 | 2 | -1 | -1 | -1 | -2 |
| 2 | 2 | 2 | 2 | 2 | 2 | -1 | -1 | 3 |
| 2 | 2 | 2 | 2 | 2 | -1 | -1 | -1 | 3 |
| -1 | 2 | 2 | 2 | 2 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | 2 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | 4 | 4 | -1 | -1 | -1 | -1 | -1 |
| 4 | 4 | 4 | 4 | 4 | -1 | -1 | -1 | 5 |
| -2 | -1 | 4 | 4 | 4 | -1 | -1 | -1 | -2 |

Table 5.12: Error sign matrix after corrections are made to Table 5.11.

| -2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 3 |
| 2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 3 |
| 2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 3 |
| 2 | -1 | -1 | -1 | -1 | -1 | 4 | 4 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 3 |
| -1 | -1 | -1 | -1 | -1 | -1 | 4 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 5 | 5 | 5 | 5 | 5 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | 5 | 5 | 5 | 5 | 5 | -1 | 5 | 5 | 5 | 5 | 5 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | -1 | -1 | -1 |
| -2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -2 |

Table 5.13: Error sign matrix with irregular distribution.

| -2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 3 |
| 2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 3 |
| 2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 3 |
| 2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 3 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 4 | 4 | 4 | 4 | 4 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | -1 | -1 | -1 |
| -2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -2 |

Table 5.14: Error sign matrix after corrections are made to Table 5.13.

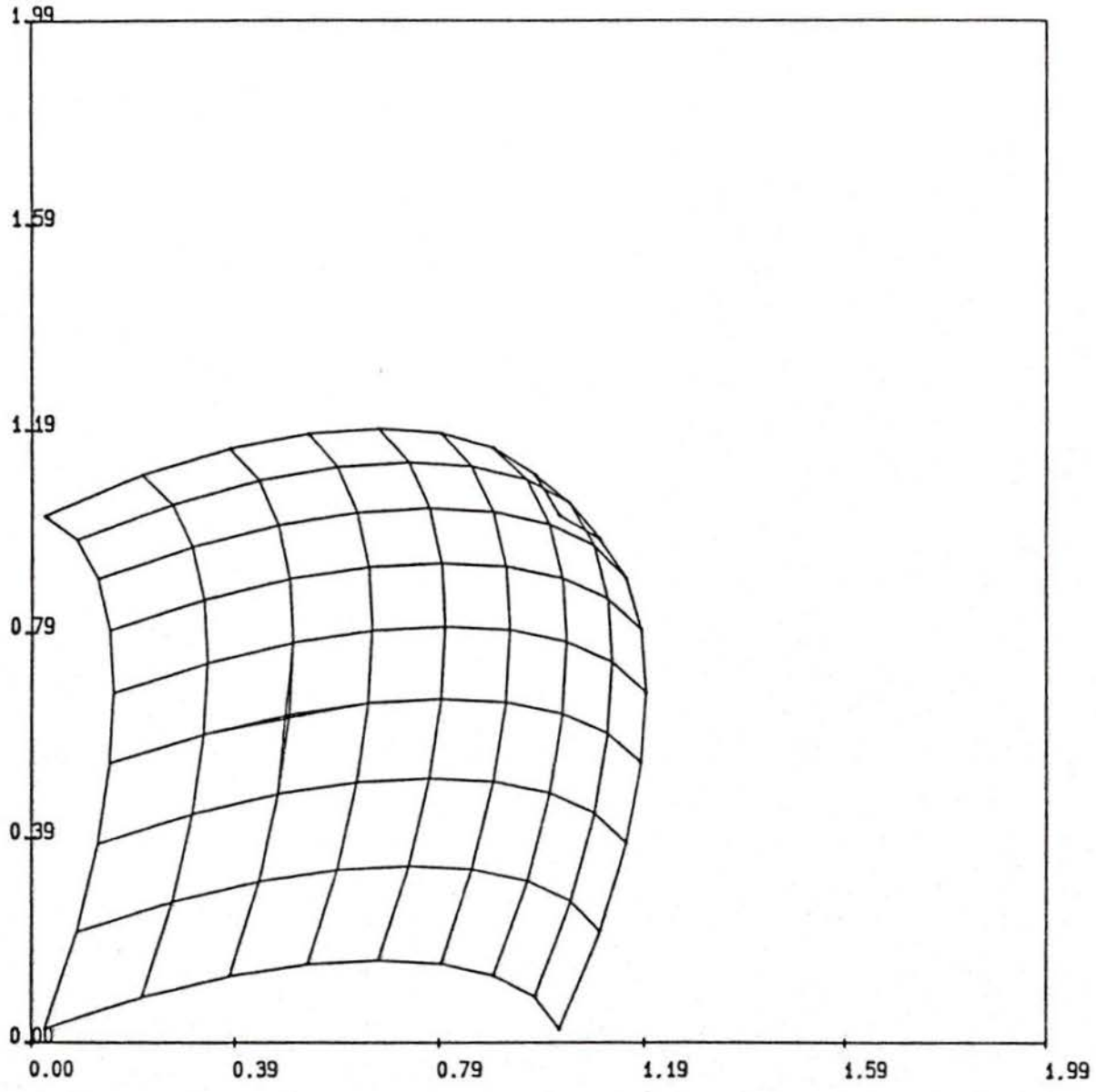


Figure 5.1: Perspective views of the control polyhedrons before and after the corrections.

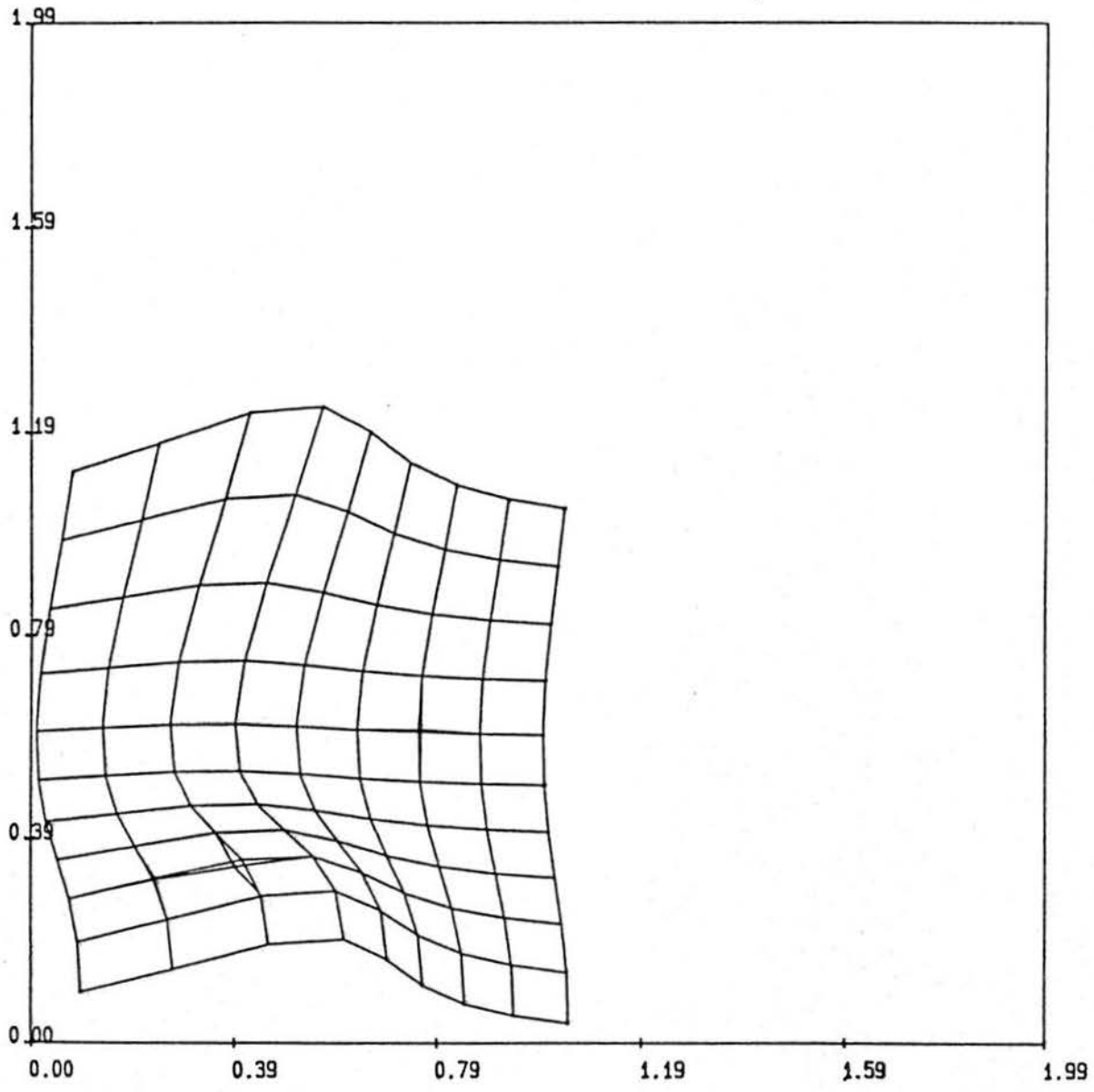


Figure 5.2: Perspective views of the control polyhedrons before and after the corrections.

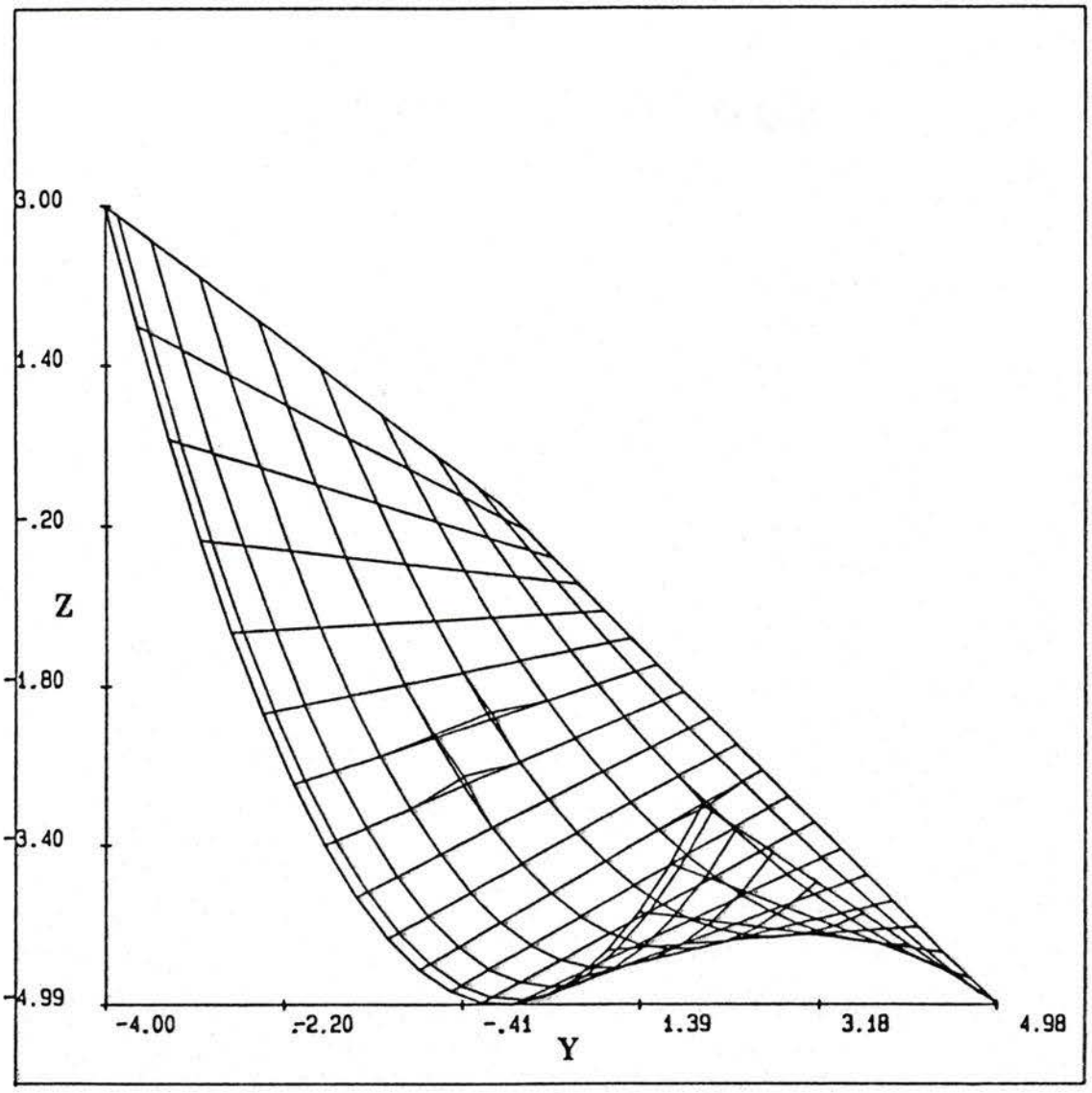


Figure 5.3: $(y - z)$ views of the control polyhedrons before and after the corrections.

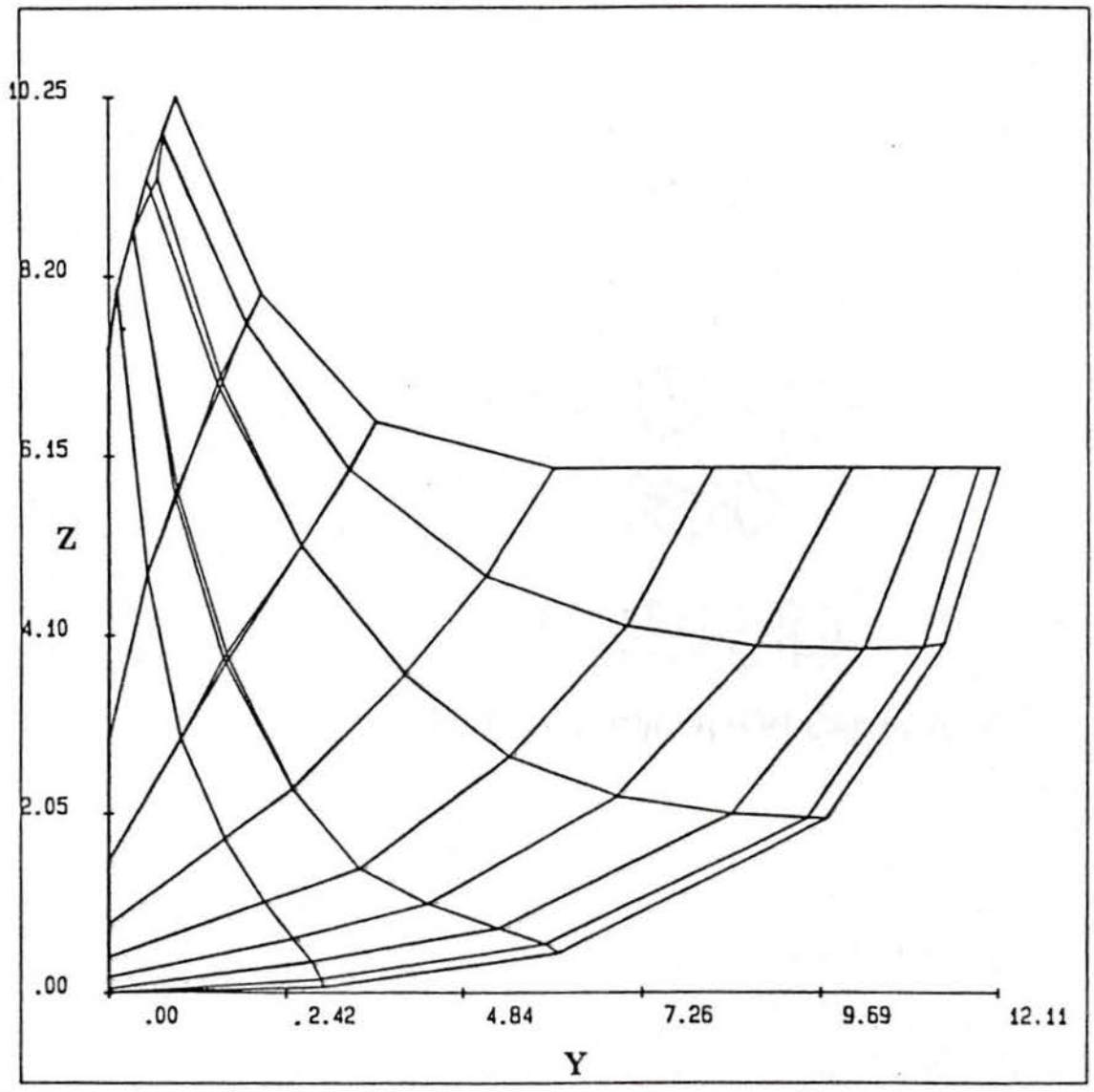


Figure 5.4: $(y - z)$ views of the kayak hull before and after the corrections.

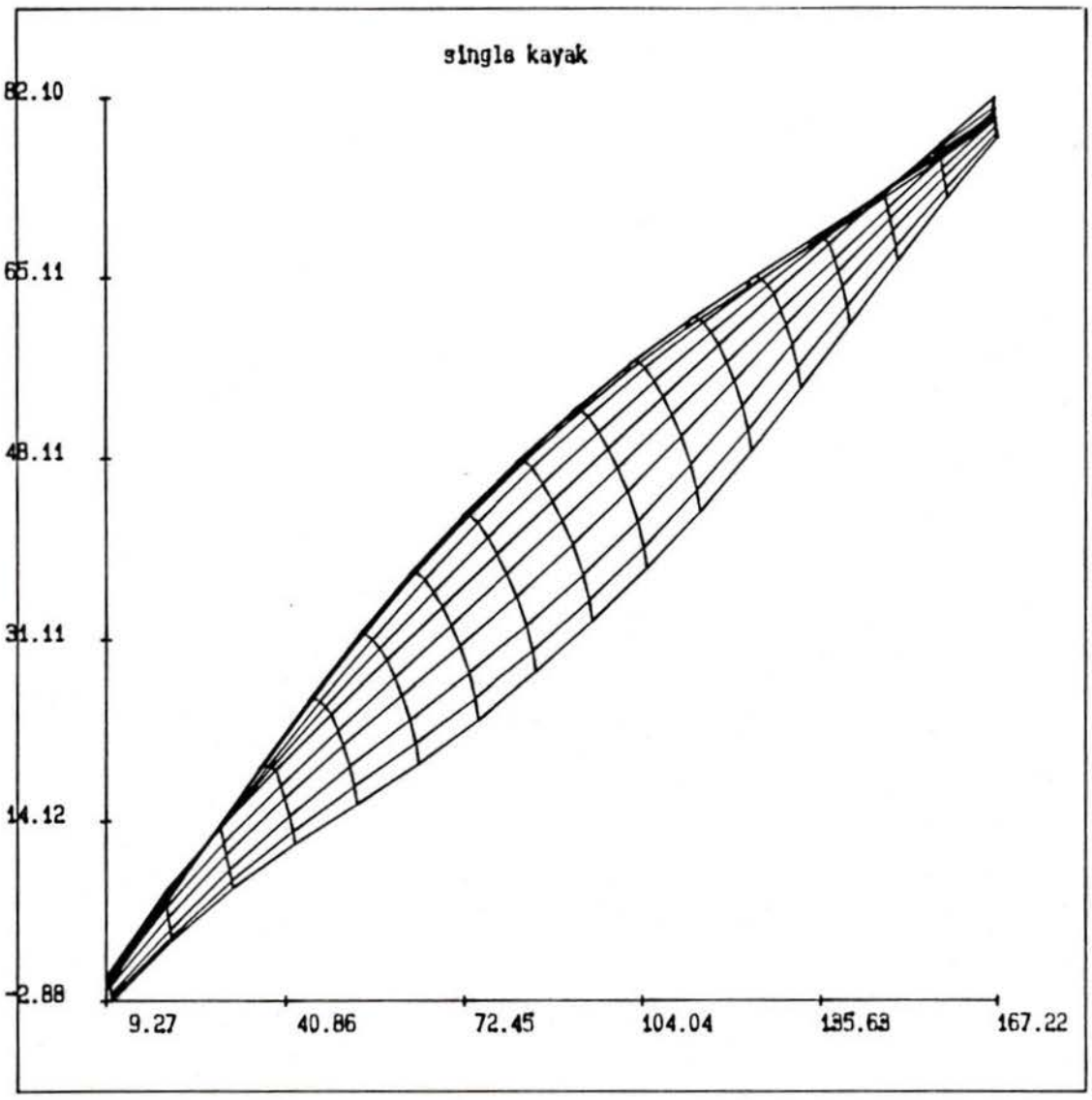


Figure 5.5: Perspective view of a single kayak hull.

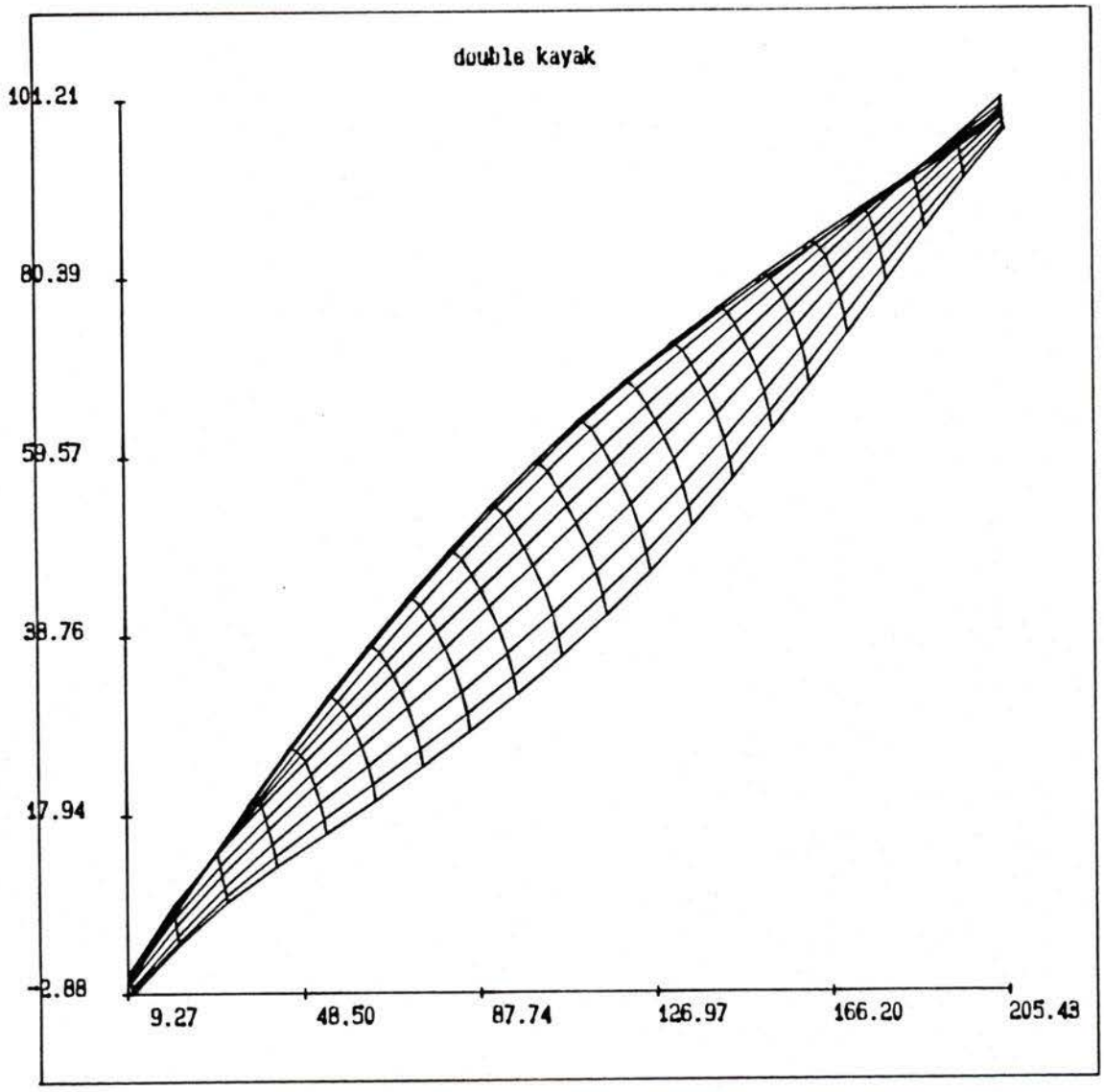


Figure 5.6: Perspective view of a double kayak hull.

# Chapter 6

# Conclusions and Further Work

## 6.1 Conclusions

In the first part of this work, a novel and efficient B-spline interpolation algorithm was developed. Instead of using the conventional matrix inversion method, it employs B-spline errors to update the control vertices iteratively. The method not only gives a complete interpolation solution, but also can generate a number of intermediate solutions between the approximation and the interpolation. Furthermore, according to the requirements, the B-spline errors can be used to modify, in a controlled manner, any particular area of the curve or surface.

In the second part of the work, simple iterative fairing algorithms based on checking the nature of the control vertices were developed. Using the sign of the cross-product at control polygon vertices, the characteristics of the control polygon are determined. This leads to the prediction of the fairness of the resulting curve.

Similarly, with the aid of the control polyhedron vertex error sign matrix, the unfair areas of the control polyhedron can be identified. In either case, all the modifications needed are made automatically and iteratively. The work enables the B-spline method to be applied more effectively to curve and surface definitions in engineering, specially in computer-aided design and manufacture applications.

## 6.2 Further Work

Presently, only the fourth order B-spline interpolation solution has been solved efficiently. Although the concept of adding the B-spline error to update the control vertices can be extended to any order of B-spline interpolation problems, a simple and effective algorithm to calculate the error needs to be developed. In addition, in order to apply the error sign matrix method more efficiently when $x$, $y$ coordinates of the control polyhedron vertices are arbitrarily located, non-uniformly spaced knot vectors may be used and the corresponding error calculation formula is required to be developed.

# Bibliography

[1] Barnhill, R. E., "Surfaces in CAGD: A Survey", *Computer Aided Geometric Design,* Vol. 2, No. 1-3, pp. 1-17, September 1985.

[2] Barsky, B. A., "End Conditions and Boundary Conditions for Uniform B-spline Curve and Surface Representations", *Computers in Industry,* Vol. 3, pp. 17-29, 1982.

[3] Barsky, B. A., "A Description and Evaluation of Various 3-D Models", *IEEE Computer Graphics and Applications,* pp. 38-52, 1984.

[4] Bohm, W., Farin, G., and Kahmann, J., "A Survey of Curves and Surface Methods in CAGD", *Computer Aided Geometric Design,* Vol. 1, No. 1, pp. 1-60, 1984.

[5] Beck, J. M., Farouki, R .T., and Hinds, J. K., "Surface Analysis Methods", *IEEE Computer Graphics and Applications,* pp. 18-36, December 1986.

[6] Cohen, E., Lyche, T., and Riesenfeld, R., "Discrete B-spline and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics ", *Computer Graphics and Image Processing,* Vol. 14, pp. 87-111, 1980.

[7] DeBoor, C., "On Calculating with B-splines", *Journal of Approximation Theory,* Vol. 6., 1972.

[8] DeBoor, C., *A Guide to Splines*, Applied Mathematical Sciences, 27, 1978.

[9] Dierckx, P., "An Algorithm for Least-Squares Fitting of Cubic Spline Surfaces to Functions on a Rectilinear Mesh over a Rectangle", *Journal of Computational and Applied Mathematics,* Vol. 3., No. 2, 1977.

[10] Dill, J. C., "An Application of Color Graphics to the Display of Surface Curvature", *Computer Graphics,* Vol. 15., No. 3, pp. 153-161, August 1981.

[11] Duncan, J. P., and Mair, S., *Sculptured Surfaces in Engineering and Medicine*, Cambridge University Press, Cambridge, 1983.

[12] Farin, G., "A Modified Clough-Tocher Interpolation", *Computer Aided Geometric Design,* Vol. 2, No. 1-3, pp. 19-27, September 1985.

[13] Farouki, R. T., "The Approximation of Non-Degenerate Offset Surfaces", *Computer Aided Geometric Design,* Vol. 3, pp. 15-43, 1986.

[14] Faux, I. D., and Pratt, M. J., *Computational Geometry for Design and Manufacture*, Ellis Horwood, Chichester, England, 1985.

[15] Flutter, A. G., and Rolph, R. N., "POLYSURF: an Interactive System for the Computer-Aided Design and Manufacture of Components", Proceedings of the CAD-76 Conference, London (UK), pp. 150-158, 1976.

[16] Gordon, W. J., and Riesenfeld, R. E., "B-spline Curves and Surfaces", in: Barnhill, R. E., ed., *Computer Aided Geometric Design*, Academic Press, New York, 1974.

[17] Hoschek, J., "Detecting Regions with Undesirable Curvature", *Computer Aided Geometric Design,* Vol. 1, No. 2, pp. 183-192, November 1984.

[18] Hoschek, J., "Smoothing of Curves and Surfaces", *Computer Aided Geometric Design*, Vol. 2, No. 1-3, pp. 97-105, September 1985.

[19] Huitric, H., and Nahas, M., "B-spline Surfaces: A Tool for Computer Painting", *Computer Aided Geometric Design, IEEE Computer Graphics and Applications*, pp. 39-47, March 1985.

[20] Izumida, K., and Y. Mattida, "Ship Hull Definition by Surface Techniques for Production Use", *Computer Application in the Automation of Shipyard Operations and Ship Design* 3, North-Holland Publishing Co., 1979.

[21] Munchmeyer, F. C., Schubert, C., and Nowacki, H., "Interactive Design of Fair Hull Surfaces Using B-splines", *Computer Applications in the Automation of Shipyard Operation and Ship Design* 3, North-Holland Publishing Company, 1979.

[22] Munchmeyer, F. C., and Lau G. K. H., "On the Interactive Design of Smooth Patched Surfaces", Interactive Techniques in Computer-Aided Design of Smooth Patched Surfaces, Palazzodei Congressi, Bologna, Italy, 1978.

[23] Pere, B., "Increasing the Smoothness of Bicubic Spline Surfaces", *Computer Aided Geometric Design,* Vol. 2, No. 1-3, pp. 157-164, September 1985.

[24] Poeschl, T., "Detecting Surface Irregularities using Isophotes", *Computer Aided Geometric Design*, Vol. 1, No. 2, pp. 163-168, November 1984.

[25] Pratt, M. J., "Smooth Parametric Surface Approximation to Discrete Data", *Computer Aided Geometric Design*, Vol. 2, No. 1-3, pp. 165-171, September 1985.

[26] Renz, W., "Interactive Smoothing of Digitized Point Data", *Computer Aided Design 14*, pp. 267-269, 1982.

[27] Riesenfeld, R. F., "Applications of B-spline Approximation to Geometric Problems of Computer-Aided Design", Ph.D. Thesis, Syracuse University, Syracuse, New York, 1973.

[28] Rogers, D. J., Satterfield, S. G., and Rodriguez, F. A., "Ship Hulls, B-spline Surfaces and CAD/CAM", *Computer Graphic Applications*, Vol. 3, No. 9, 1983.

[29] Sabin, M. A., "An Existing System in the Aircraft Industry. The British Aircraft Corporation Numerical Master Geometry System", Proceedings of the Royal Society of London, Vol. A321, pp. 197-205, 1971.

[30] Schmitt, F. J. M., Barsky, B. A., and Du, W. H., "An Adaptive Subdivision Method for Surface-Fitting from Sampled Data", Computer Graphics, SIGGRAPH'86 Proceedings, pp. 179-188, August 1986.

[31] Sederberg, T. W., and Anderson D. C., "Steiner Surface Patches", *IEEE Computer Graphics and Applications*, pp. 23-36, May 1985.

[32] Worsey, A. J., "$C^2$ Interpolation over Hypercubes", *Computer Aided Geometric Design*, Vol. 2, No. 1-3, pp. 107-115, September 1985.

# Appendix A

# Hermite Interpolation and Coons Patch

## A.1 Hermite Interpolation

It is known that interpolation to both position and slope constrains is called *Hermite interpolation*. Assume $\{P_0, P_1, \cdots, P_n\}$ be $n+1$ points to be interpolated and $\{P_0^1, P_1^1, \cdots, P_n^1\}$ be the corresponding values of the first derivative vector. Parametrically the $i^{th}$ curve segment is described as

$$R_i(t) = \sum_{j=0}^{1} \sum_{k=0}^{1} h_{jk}(t) P_{i-1+k}^j. \tag{A.1}$$

The functions $h_{jk}(t)$ are the cubic Hermite basis functions

$$h_{00}(t) = 2t^3 - 3t^2 + 1, \tag{A.2}$$

$$h_{01}(t) = -2t^3 + 3t^2, \tag{A.3}$$

$$h_{10}(t) = t^3 - 2t^2 + t, \tag{A.4}$$

$$h_{11}(t) = t^3 - t^2. \tag{A.5}$$

## A.2 Coons Patches

One of the first method for surface representation was proposed by Coons. The basic idea is to create a patch by blending four boundary

curves. A simple Coons patch can be expressed as

$$\begin{aligned} Q(u,v) &= \sum_{i=0}^{1} \alpha_i(u)F(i,v) + \sum_{j=0}^{1} \alpha_j(v)F(u,j) \\ &\quad - \sum_{i=0}^{1}\sum_{j=0}^{1} \alpha_i(u)\alpha_j(v)F(i,j). \end{aligned} \tag{A.6}$$

Here, $F(u,0)$, $F(u,1)$, $F(0,v)$ and $F(1,v)$ are the boundary curves; $F(0,0)$, $F(0,1)$, $F(1,0)$ and $F(1,1)$ are the corner points; and $\alpha_0(t)$ and $\alpha_1(t)$ are the blending functions. The blending functions satisfy $\alpha_i(j) = \delta_{ij}$, where $\delta_{ij}$ is the Kronecker delta. This simple Coons patch does not constrain the cross boundary derivatives; thus it is not possible to ensure continuity higher than positional when using composite patches.

For first derivative continuity this method is extended so that the cross-boundary derivative can be specified. This requires four blending functions, $h_{00}(t)$, $h_{01}(t)$, $h_{10}(t)$, and $h_{11}(t)$. The patch is now written

$$\begin{aligned} Q(u,v) &= \sum_{i=0}^{1}\sum_{r=0}^{1} F^{r,0}(i,v)h_{r,i}(u) \\ &\quad + \sum_{j=0}^{1}\sum_{s=0}^{1} F^{0,s}(u,j)h_{s,j}(v) \\ &\quad - \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{r=0}^{1}\sum_{s=0}^{1} F^{r,s}(i,j)h_{r,i}(u)h_{s,j}(v), \end{aligned} \tag{A.7}$$

where

$$F^{a,b}(u_i,v_j) = \frac{\partial^{a+b}F(u,v)}{\partial u \partial v}|u = u_i, v = v_j.$$

One way to simplify Equation (A.7) is to use the following boundary func-

tions

$$F^{0,s}(u,j) = \sum_{i=0}^{1}\sum_{r=0}^{1} F^{r,s}(i,j)h_{i,r}(u), \tag{A.8}$$

and

$$F^{r,0}(i,v) = \sum_{j=0}^{1}\sum_{s=0}^{1} F^{r,s}(i,j)h_{j,s}(v). \tag{A.9}$$

Substituting Equation (A.8) and (A.9) into (A.7), the three terms are now equal, and thus Equation (A.7) reduces to

$$Q(u,v) = \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{r=0}^{1}\sum_{s=0}^{1} F^{r,s}(i,j)h_{r,i}(u)h_{s,j}(v), \tag{A.10}$$

The blending functions have to satisfy

$$h_{0i}(j) = h'_{1i}(j) = \delta_{ij}, \tag{A.11}$$

$$h_{1i}(j) = h'_{0i}(j) = 0. \tag{A.12}$$

these conditions are satisfied by the cubic Hermite basis functions given in Equations from (A.2) to (A.5).

The simplified Coons patch is completely defined in terms of four points, eight first partial derivatives and four second partial derivatives at the outermost four corners, also known as the position, slope and twist. Large surfaces are defined by a mesh of such patches. Point and slope continuity for bicubic patches can be ensured by only matching the corner quantities.

# Appendix B

# Bezier Curves and Surfaces

## B.1 Bezier Curves

A Bezier curve is associated with the "vertices" of a polygon which define the curve shape. In general, an $n^{th}$ degree Bezier curve is defined by $n+1$ vertices as

$$R(t) = \sum_{i=0}^{n} P_i B_{i,n}(t), \quad 0 \leq t \leq 1. \tag{B.1}$$

Where $P_i$ is the $i^{th}$ control vertex, and

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

is the so called Bernstein polynomial basis function.

## B.2 Bezier Surfaces

A Bezier surface is a tensor product of Bezier curves. It is defined by a set of control vertices, $V_{i,j}$'s $(0 \leq i \leq n, 0 \leq j \leq m)$, in three-dimensional $x-y-z$ space. A point on the surface is a weighted average of these control vertices. That is,

$$Q(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} V_{i,j} B_{i,n}(u) B_{j,m}(v), \quad 0 \leq u, v \leq 1. \tag{B.2}$$

# Appendix C

# Divided Difference

*Definition*

The $k^{th}$ divided difference of a function $g(x)$ at the points $x_i, \cdots, x_{i+k}$, is the leading coefficient (i.e., the coefficient of $x^k$ ) of the polynomial of order $k+1$ which agree with $g(x)$ at the points $x_i, \cdots, x_{i+k}$ [8]. It is denoted by

$$[x_i, \cdots, x_{i+k}]g.$$

*Theorem C.1*

The $k^{th}$ divided differences of a polynomial $g(x)$ of degree $k$ are all equal, and the $(k+1)^{th}$ divided differences are consequently all zero (see Faux [14] for proof).

*Theorem C.2*

The computation of a divided difference can be carried out recursively [14], that is

$$[x_s, \cdots, x_t]g = \frac{[x_{s+1}, \cdots, x_t]g - [x_s, \cdots, x_{t-1}]g}{x_t - x_s}. \tag{C.1}$$

*Leibniz' Formula*

If $f = gh$, i.e., $f(t) = g(t)h(t)$, for all $t$, then

$$[x_i, \cdots, x_{i+k}]f = \sum_{r=i}^{i+k}([x_i, \cdots, x_r]g)([x_r, \cdots, x_{i+k}]h). \tag{C.2}$$

see de Boor [8] for the proof.

# Appendix D

# List of Symbols

$B_{i,k}(t)$ the $i^{th}$ normalized B-spline basis function of order $k$

$x_i$ the $i^{th}$ element of the knot vector

$P_i$ the $i^{th}$ control polygon vertex

$R(t)$ the parametric B-spline curve approximation

$k$ the order of the B-spline basis function

$M$ the multiplicity of the knot

$R_j$ the $j^{th}$ point on the B-spline approximation curve

$Q(u,v)$ the B-spline surface approximation

$PT_i$ the $i^{th}$ temporary control vertex

$PT_{yi}$ the $i^{th}$ temporary control vertex value in $y$

$V_{i,j}$ the control polyhedron vertex

$Q_{i,j}$ the $i^{th}$ and $j^{th}$ B-spline approximation surface point corresponding to $V_{i,j}$

$C\vec{P}_i$ the vector cross-product on a vertex $P_i$

$\vec{P_i}$ the $i^{th}$ control polygon vertex vector

$P\vec{N}_i$ the $i^{th}$ new control polygon vertex vector after a correction is made to $\vec{P_i}$

$S_{max}$ the most frequently occurring element value in the error sign matrix

$S_{mid}$ the second most frequently occurring element value in the error sign matrix

$S_{min}$ the least frequently occurring element value in the error sign matrix

$S_d$ the dominant element value

# VITA

*Surname:* WANG *Given Name:* HUI PING

*Place of Birth:* BEIJING, CHINA
*Date of Birth:* March 12, 1962.

*Educational Institutions Attended, with dates of Entering and Leaving:*

| | | | |
|---|---|---|---|
| Beijing Institute of Telecommunications, Beijing. | 1980 | to | 1984 |
| University of Victoria,Victoria. | 1985 | to | 1987 |

*Degrees, Diplomas, Etc., Awarded, with Dates ans Names of Institutions:*

B.Eng. 1984 Beijing Institute of Telecommunications, Beijing.

*Honors and Awards:*

University of Victoria Fellowship, 1986.

Charles S. Humphrey Award, 1986.

*Publications:*

Wang, H. P., G. W. Vickers and S. Bedi, "An Efficient Evaluation Algorithm for B-spline Interpolation", submitted to *Computer Aided Geometric Design*, 1987.

## PARTIAL COPYRIGHT LICENSE

I hereby grant the right to lend my thesis (the title of which is shown below) to users of the University of Victoria Library, and to make *single copies only* for such users or in the response to a request from the library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my permission.

Title of Thesis

Definiing and Fairing Curves and Surfaces

Using an Iterative B-spline Error Technique

Author [redacted]

*Signature*

Hui Ping Wang

*Name*

May 26, 1987

*Date*