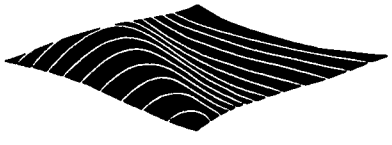
# FINAL REPORT

## NAG–1–1061

# Enhancement Of Surface Definition And Gridding In The Eagle Code

ENGINEERING RESEARCH CENTER FOR
COMPUTATIONAL FIELD SIMULATION
COMPLEX GEOMETRY / COMPLEX PHYSICS



*Dr. Joe F. Thompson, Principal Investigator*
*Brian A. Jean, Graduate Research Assistant*
*P.O. Box 6176*
*Mississippi State, MS 39762*

SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>unclassified | 1b. RESTRICTIVE MARKINGS<br>N/A |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION/AVAILABILITY OF REPORT |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE<br>N/A | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S) | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a. NAME OF PERFORMING ORGANIZATION<br>Engineering Research Center<br>Mississippi State University | 6b. OFFICE SYMBOL (*If applicable*) |
| 6c. ADDRESS (*City, State, and ZIP Code*)<br>Box 6176<br>Mississippi State, MS 39762 | 7a. NAME OF MONITORING ORGANIZATION<br>7b. ADDRESS (*City, State, and ZIP Code*) |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION<br>Langley Research Center, NASA | 8b. OFFICE SYMBOL (*If applicable*) |
| 8c. ADDRESS (*City, State, and ZIP Code*)<br>NASA Langley Research Center<br>SSD M/S 366<br>Hampton, VA 23665-5225 | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER<br>10. SOURCE OF FUNDING NUMBERS |

| PROGRAM ELEMENT NO. | PROJECT NO. | TASK NO. | WORK UNIT ACCESSION NO. |
|---|---|---|---|
| | | | |

11. TITLE (*Include Security Classification*)
Enhancement of Surface Definition and Gridding in the EAGLE Code

12. PERSONAL AUTHOR(S)
Dr. Joe F. Thompson

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (*Year, Month, Day*) | 15. PAGE COUNT |
|---|---|---|---|
| final technical | FROM Aug 89 TO Sept 91 | 27, November 1991 | 94 |

16. SUPPLEMENTARY NOTATION

| 17. COSATI CODES | | | 18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | |
| | | | |
| | | | |

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

Algorithms for smoothing of curves and surfaces for the EAGLE grid generation program are presented. The method uses an existing automated technique which detects undesirable geometric characteristics by using a local fairness criterion. The geometric entity is then smoothed by repeated removal and insertion of spline knots in the vicinity of the geometric irregularity. The smoothing algorithm is formulated for use with curves in B-spline form and tensor product B-spline surfaces.

| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT | 21. ABSTRACT SECURITY CLASSIFICATION | |
|---|---|---|
| ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS | | |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL<br>Joe F. Thompson, PhD | 22b. TELEPHONE (*Include Area Code*)<br>(601)325-7299 | 22c. OFFICE SYMBOL |

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

MISSISSIPPI STATE UNIVERSITY / National Science Foundation

# FINAL REPORT

NAG–1–1061

## ENHANCEMENT OF SURFACE DEFINITION AND GRIDDING IN THE EAGLE CODE

ENGINEERING RESEARCH CENTER FOR
COMPUTATIONAL FIELD SIMULATION
COMPLEX GEOMETRY / COMPLEX PHYSICS

*Dr. Joe F. Thompson, Principal Investigator*
*P. O. Drawer 6176*
*Mississippi State, MS 39762*

TABLE OF CONTENTS

LIST OF FIGURES

# CHAPTER I

# INTRODUCTION

Recent years have seen a dramatic increase in the use of computational field simulation (CFS) for the solution of physical field problems which can be modeled by systems of partial differential equations (PDEs). At present, computational fluid dynamics (CFD) is the most widely applied and most thoroughly understood area of CFS. Current applications in CFD include flow solutions about bodies composed of complex geometrical shapes including sculpted surfaces. Correct modeling of physical phenomena in CFD requires the use of an accurate description of the geometry involved in the problem. Therefore, discretized curves and surfaces representing the geometry of a physical field problem must be as accurate as possible.

Geometry data for CFD problems are often given as a set of digitized data points or as surface patches generated by computer aided drafting and design (CADD) software. In either case, the geometrical entities of interest are seldom analytical curves or surfaces. Digitized data sets usually contain errors resulting from the digitizing process. These errors manifest themselves as dimples, bumps, or ridges on the surface. CADD data bases may also contain errors due to lack of second or third derivative continuity at surface patch interfaces, again resulting in bumps or ridges. These errors are usually quite small and are not easily detected unless the surface or curve is plotted at full scale. However, it is not always practical or possible to examine a full scale plot or model since producing them is both expensive and time consuming.

Therefore, a method of interrogation is needed which will detect small flaws without using a full scale representation of the geometry involved. After an irregularity has been detected, it is then desirable to develop a means to correct the error without significantly perturbing the original geometry.

In order to effectively interrogate a geometric entity, a quantity describing the qualities sought must be defined. This quantity can then be calculated at various locations on the geometry and its quality assessed. Several means of interrogation have been suggested. For detection of irregularities in surfaces, Kaufmann and Klass [1] use reflection lines. Beck [2] uses a variety of methods including contour plots, shaded images, and maps of principal curvature. Hoschek [3] uses k–orthotomic curves for interrogation of planar curves, and Renz [4] uses second divided differences. Farin suggests the use of curvature plots (Refs. [5],[6],[7]) since they are highly sensitive to changes in curve shape and they allow easy detection and location of inflection points. However, signed curvature applies only to planar curves and is, by definition, nonnegative for space curves; as a more general approach, third derivative may be used (Ref [8]).

Removal of irregularities in a curve or surface basically involves smoothing or fairing the geometry. Many smoothing techniques have been put forth. Smoothing of Bezier curves and surfaces is discussed by Hoschek[3]. Kjellander provides a method for smoothing cubic spline curves in Ref[9], and bicubic parametric surface patches are discussed by Kjellander in Ref[10]. A method for curves composed of digitized data points is given by Renz[4]. Several methods for smoothing B–spline curves are presented by Klass[1], and by Farin[5],[6],[7],[8],[11]. It should be noted that curve and surface smoothers depend on the geometry model used.

Many mathematical methods of geometry description are available and they all fall into two major categories, interpolation methods and approximation methods. Interpolation techniques are characterized by the generated curve or surface passing through the physical data points which describe the object. Interpolation schemes include Lagrange and Hermite interpolation [7],[12],[13],[14],[15], Ferguson patches[13], Coon's patches[13], and transfinite interpolation[14], as well as polynomial splines[7],[12],[13],[15] and tension splines[12]. Approximation techniques are somewhat different. Instead of using points that lie on the object in question, they use a set of control points which, in general, do not lie on the curve or surface being described. This may lead to some misunderstanding. It should be noted that approximation techniques represent geometry as accurately as do interpolation methods ( in some cases, more accurately ). The difference lies in the fact that these schemes store the object in question using a set of control points which do not lie on the object. The two most widely used approximation techniques are Bezier curves and surfaces and B–splines. Excellent treatment of Bezier methods is given in Ref[7]; other works on Bezier methods include Refs[12],[13],[15],[16]. B–splines are covered in Refs[7],[12],[17],[16].

Smoothing schemes based on polynomial splines or Bezier curves require that the entire curve or surface be modified. This is due to the fact that polynomial splines do not exhibit the property of *local control*; Bezier curves do not have this property either. However B–splines do exhibit local control and allow the development of fairing algorithms which modify a curve or surface only in the region immediately surrounding the affected control points. Because B–splines exhibit the property of local control, and thus allow local smoothing algorithms to be developed, the geometry model used here will be

based on the B–spline representation. B–splines have already been incorporated into the EAGLE code (Ref[18]), thus making implementation of a B–spline smoother a logical extension of existing capabilities.

This work is organized as follows: A brief discussion of various curve modeling techniques is given in Chapter II. Chapter III is devoted to surface generation, and Chapter IV presents the interrogation and smoothing algorithms. Chapter V contains results obtained using the interrogation and smoothing routines.

# CHAPTER II

# CURVES

Many different kinds of geometry models exist. The two major categories are interpolation methods and approximation methods. Within each of those categories are methods which produce single curve or surface patches, as well as schemes which result in a piecewise description of the geometry. While methods resulting in single curve or surface patches are simple and produce smoothly varying geometric entities, they lack the flexibility necessary to describe complex shapes. Polynomial segments of the degree necessary for description of these complex geometries exhibit unwanted oscillations. High degree Bezier techniques eliminate the wiggles, however they are computationally too expensive to be practical in real–world applications Ref[7]. In addition, single Bezier curves do not have the property of local control. Piecewise techniques, on the other hand, are theoretically and conceptually more difficult, however they allow complex shapes to be represented and are computationally more tractable than high degree Bezier curves. Probably the most general of the piecewise or spline techniques is the non–uniform rational B–spline[7].

For engineering purposes, single curve segments higher than degree three need not be considered. This is because cubic curve segments are true three dimensional curves. Since engineering efforts are restricted to geometries of three dimensions, the use of higher degree curves would simply introduce unneeded complexity and computational requirements.

## The Parametric Cubic Curve

Mortenson[16] defines a parametric cubic ( PC ) curve as a continuous, single–valued mathematical function of the form:

$$x = x(t) \qquad y = y(t) \qquad z = z(t) .$$

The parameter t ranges from t=0 at the start of the curve to t=1 at the end. In vector notation, the curve is denoted as **P**(t) where the vector **P** has the components x, y, and z in three dimensions. Derivatives of **P** are taken with respect to the parametric variable t as follows.

$$P^{t}(t) = \frac{dP(t)}{dt} \qquad P^{tt}(t) = \frac{d^2P(t)}{dt^2}$$

Higher derivatives are calculated in a similar fashion, however for PC curves, derivatives higher than third order are meaningless because they are zero. Algebraically, a PC curve can be expressed as a cubic polynomial in the variable t. Equation (2.1) is the algebraic form of a PC curve. Note that the coefficients of the t terms are vectors.

$$P(t) = a_3t^3 + a_2t^2 + a_1t + a_0 \qquad (2.1)$$

A more physical definition of the PC curve is offered by the geometric form. This form is derived by taking the first derivative of Equation (2.1) with respect to the t and solving for the tangent vectors of the curve at the ends, $\mathbf{P}^t(0)$ and $\mathbf{P}^t(1)$. Then solving Equation (2.1) for the end points of the curve, **P**(0) and **P**(1). The results are given as Equations (2.2). If Equations (2.2) are solved for the algebraic coefficients **a**, Equations (2.3) result. Substituting these back into Equation (2.1) and grouping the result according to vectors **P**

and $\mathbf{P}^t$, results in Equation (2.4:)

$$\begin{aligned} P(0) &= a_0 \\ P(1) &= a_0 + a_1 + a_2 + a_3 \\ P^t(0) &= a_1 \\ P^t(1) &= a_1 + 2a_2 + 3a_3 \end{aligned} \tag{2.2}$$

$$\begin{aligned} a_0 &= P(0) \\ a_1 &= P^t(0) \\ a_2 &= -3P(0) + 3P(1) - 2P^t(0) - P^t(1) \\ a_3 &= 2P(0) - 2P(1) + P^t(0) + P^t(1) \end{aligned} \tag{2.3}$$

$$\begin{aligned} P(t) = (2t^3 - 3t^2 + 1)P(0) + (-2t^3 + 3t^2)P(1) + (t^3 - 2t^2 + t)P^t(0) \\ + (t^3 - t^2)P^t(1) \end{aligned} \tag{2.4}$$

The coefficients of the $\mathbf{P}$ and $\mathbf{P}^t$ terms can be interpreted as blending functions. These functions relate the physical curve to the parametric space of the curve. If the coefficients of $\mathbf{P}(0)$, $\mathbf{P}(1)$, $\mathbf{P}^t(0)$, and $\mathbf{P}^t(1)$ are denoted $\mathbf{F}_1$, $\mathbf{F}_2$, $\mathbf{F}_3$, and $\mathbf{F}_4$ respectively, then Equation (2.4) becomes

$$P(t) = F_1P(0) + F_2P(1) + F_3P^t(0) + F_4P^t(1), \tag{2.5}$$

where

$$\begin{aligned} F_1 &= 2t^3 - 3t^2 + 1 \\ F_2 &= -2t^3 + 3t^2 \\ F_3 &= t^3 - 2t^2 + t \\ F_4 &= t^3 - t^2 . \end{aligned}$$

In matrix form,

$$P(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_0' \\ P_1' \end{bmatrix} \quad (2.6)$$

## Bezier Curves

### The de Casteljau Algorithm

The de Casteljau algorithm generates an n–th degree polynomial curve by repeated linear interpolation. The algorithm is stated as follows.

Given points $\mathbf{b}_0,\ldots,\mathbf{b}_n$ in 3 space and a real number t,

$$b_i^r(t) = (1-t)b_i^{r-1}(t) + tb_{i+1}^{r-1}(t) \quad \begin{cases} r = 1,\ldots,n \\ i = 0,\ldots,n-r \end{cases} \quad (2.7)$$

where

$$b_i^0(t) = b_i \ .$$

The final step in the algorithm ( with r=n and i=0 ) will yield the point on the curve at the parameter value t. Curves produced in this way are Bezier curves and have several important properties, some of which are discussed in Ref [7]. These properties include:

» Affine invariance.

» Invariance under affine parameter transformations.

» Convex hull property.

» Endpoint interpolation.

A graphical interpretation of the de Casteljau algorithm for the cubic case is shown in Figure 2.1 .

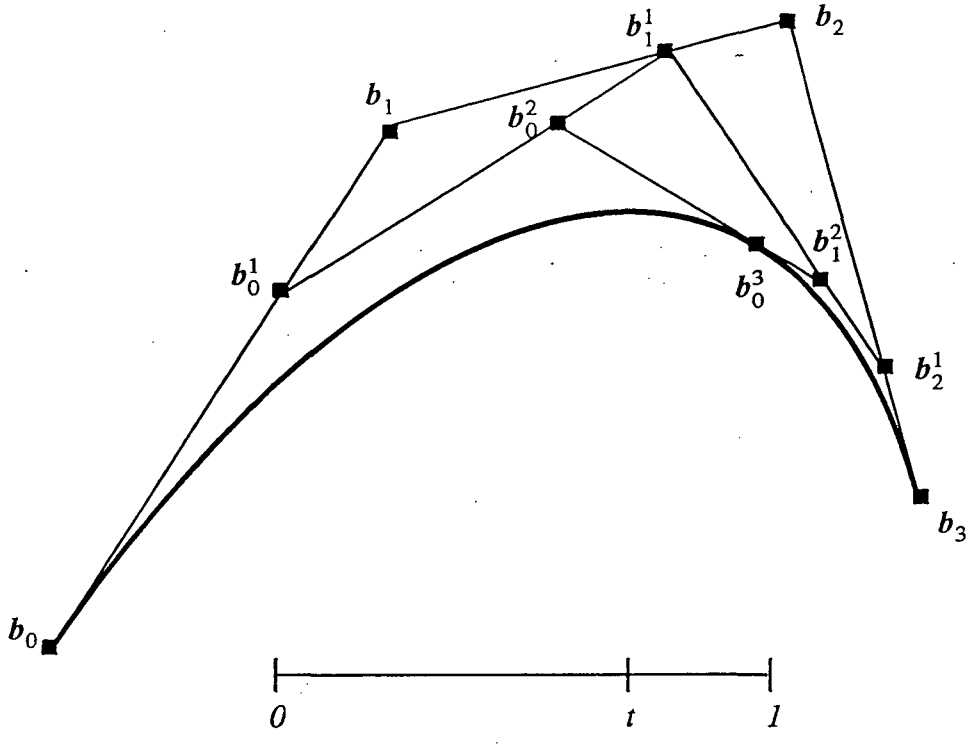



Figure 2.1 The de Casteljau algorithm for n=3.

Listed below are two examples of expansions of the de Casteljau algorithm for the quadratic and the cubic cases.

$$n = 2$$

$$b_0^1(t) = (1 - t)b_0 + tb_1$$

$$b_1^1(t) = (1 - t)b_1 + tb_2$$

$$b_0^2(t) = (1 - t)b_0^1 + tb_1^1$$

Substituting the first two Equations into the second, a closed formula results:

$$b_0^2(t) = (1-t)^2 b_0 + 2t(1-t)b_1 + t^2 b_2$$

$n = 3$

$$b_0^1(t) = (1-t)b_0 + tb_1$$
$$b_1^1(t) = (1-t)b_1 + tb_2$$
$$b_2^1(t) = (1-t)b_2 + tb_3$$

$$b_0^2(t) = (1-t)b_0^1 + tb_1^1$$
$$b_1^2(t) = (1-t)b_1^1 + tb_2^1$$

$$b_0^3(t) = (1-t)b_0^2 + tb_1^2$$

Performing a substitution process similar to that for the quadratic case results in Equation (2.8):

$$b_0^3(t) = (1-t)^3 b_0 + 3t(1-t)^2 b_1 + 3t^2(1-t)b_2 + t^3 b_3 \qquad (2.8)$$

## Bernstein Polynomials

Bernstein polynomials are defined explicitly by

$$B_i^n(t) = C(n,i)t^i(1-t)^{n-i} \qquad t \in [0\,.\,1] \qquad (2.9)$$

Where

$$C(n,i) = \binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!} & , \; i \in [\,0, n\,] \\ 0 & , \textit{otherwise.} \end{cases}$$

A recursion formula for the Bernstein polynomials can be derived as follows and the result is given as Equation (2.10):

$$\begin{aligned} B_i^n(t) &= \binom{n}{i} t^i (1-t)^{n-i} \\ &= \binom{n-1}{i} t^i (1-t)^{n-i} + \binom{n-1}{i-1} t^i (1-t)^{n-i} \\ &= (1-t) B_i^{n-1}(t) + t\, B_{i-1}^{n-1}(t) \qquad t \in [0.1] \end{aligned} \tag{2.10}$$

The results of evaluating Equation (2.9) for the cases n=2 and n=3 are given by Equations (2.11) and (2.12) respectively. These correspond to the quadratic case and cubic case respectively:

$$\begin{aligned} B_0^2(t) &= (1-t)^2 \\ B_1^2(t) &= 2t(1-t) \\ B_2^2(t) &= t^2 \end{aligned} \tag{2.11}$$

$$\begin{aligned} B_0^2(t) &= (1-t)^3 \\ B_0^2(t) &= 3t(1-t)^2 \\ B_0^2(t) &= 3t^2(1-t) \\ B_0^2(t) &= t^3 \end{aligned} \tag{2.12}$$

Some properties of Bernstein polynomials, from Refs [19], [20], [21], include the partition of unity, the positive property, endpoint interpolation, the recursive property, and symmetry:

- Partition of Unity: see Equation (2.13).
- Positive Property: For $0 \leq t \leq 1$ Bernstein polynomials are positive.
- Endpoint Interpolation
- Recursive Property: see Equation (2.10).

$$\sum_{i=0}^{n} B_i^n(t) \equiv 1 \tag{2.13}$$

*Proof:*

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

$B_i^n(0) = 1 \quad iff \quad i = 0$

$B_i^n(1) = 1 \quad iff \quad i = n$

» **Symmetry: From the symmetry property of the binomial coefficients,**

$$C(n, n-i) = C(n, i),$$

$$\begin{aligned} B_{n-i}^n(t) &= C(n, n-i) t^{n-i} (1-t)^i \\ &= C(n,i)(1-t)^i [1 - (1-t)]^{n-i} \\ &= B_i^n(1-t) . \end{aligned} \tag{2.14}$$

Derivatives of Bernstein polynomials are determined as follows:

$$\begin{aligned} \frac{d}{dt} B_i^n(t) &= \frac{d}{dt} C(n,i)\, t^i (1-t)^{n-i} \\ &= C(n,i)[\, i\, t^{i-1}(1-t)^{n-i} - (n-i)\, t^i (1-t)^{n-i-1}\,] \\ &= \frac{i\, n!}{i!\,(n-i)!}\,[\, t^{i-1}(1-t)^{n-i}\,] - \frac{(n-i)(n!)}{i!\,(n-i)!}[\, t^i (1-t)^{n-i-1}\,] \\ &= \frac{n\,(n-1)!}{(i-1)!\,(n-i)!}\,[\, t^{i-1}(1-t)^{n-i}\,] - \frac{n(n-1)!}{i!\,(n-i-1)!}[\, t^i (1-t)^{n-i-1}\,] \end{aligned}$$

$$\frac{d}{dt} B_i^n(t) = n\,[\, B_{i-1}^{n-1}(t) - B_i^{n-1}(t)\,] \tag{2.15}$$

Bezier Curves Using Bernstein Polynomials

The Bezier curve is a simple approximation technique which takes the form of Equation (2.16):

$$P(t) = \sum_{i=0}^{i=n} b_i B_i^n(t) \qquad t \in [0,1] \tag{2.16}$$

Here the $\mathbf{b}_i$ are the control vertices of the curve, and the B(t) are the Bernstein polynomials. A word about the subscript notation in the above Equations is in order. Since the Bernstein polynomials depend on the number of control vertices used in the curve description, the double subscript notation is necessary. The subscript n denotes the degree of the curve ( i.e. n=2 for quadratic, n=3 for cubic, etc. ). The subscript i indicates the control vertex associated with the particular Bernstein polynomial. If the Bezier formulation of Equation (2.16) is expanded for the cubic case, Equation (2.17) results:

$$P(t) = (1-t)^3 b_0 + 3t(1-t)^2 b_1 + 3t^2(1-t) b_2 + t^3 b_3 \tag{2.17}$$

Equation (2.17) was obtained directly using the Bernstein polynomials, however, Bezier curves were first introduced in the form of a recursion formula ( the de Casteljau algorithm ). If one compares Equation (2.17) with Equation (2.8), which was obtained from the de Casteljau algorithm with n=3, they are the same. If Equation (2.17) is cast in the form of a PC curve, the blending functions F now become

$$F_1 = (1-t)^3$$
$$F_2 = 3t(1-t)^2$$
$$F_3 = 3t^2(1-t)$$
$$F_4 = t^3$$

and Equation (2.17) becomes

$$P(t) = F_1 b_0 + F_2 b_1 + F_3 b_2 + F_4 b_3, \tag{2.18}$$

or in matrix form,

$$P(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}. \tag{2.19}$$

Because the Bernstein polynomials form a basis for Bezier curves, Bezier curves inherit several properties intrinsic to the Bernstein polynomials. The following is a partial list. Farin[7] provides a more complete listing.

- Affine invariance. Because of Equation (2.13), Bezier curves are invariant under affine maps.
- Invariance under affine parameter transformations.
- Convex hull property. Because of the positive property of the Bernstein polynomials.
- Endpoint interpolation.
- Symmetry. Due to Equation (2.14). Algebraically, this is stated as Equation (2.20):

$$\sum_{i=0}^{n} b_i B_i^n(t) = \sum_{i=0}^{n} b_{n-i} B_i^n(1-t) \tag{2.20}$$

### Derivatives of Bezier Curves

From Equations (2.15) and (2.16) the derivative of a Bezier curve can be determined:

$$\frac{d}{dt}\, \boldsymbol{b}^n(t) = n \sum_{i=0}^{n} \left( B_{i-1}^{n-1}(t) - B_i^{n-1}(t) \right) \boldsymbol{b}_i$$

This can be simplified to yield

$$\frac{d}{dt}\, \boldsymbol{b}^n(t) = n \sum_{i=0}^{n-1} \left( \Delta \boldsymbol{b}_i \right) B_i^{n-1}(t) \ . \tag{2.21}$$

Where

$$\Delta \boldsymbol{b}_i = \boldsymbol{b}_{i+1} - \boldsymbol{b}_i$$

Repeated application of Equation (2.21) produces a general formula for determining higher order derivatives of a Bezier curve. First, the forward difference operator $\delta$ is generalized in Equation (2.22). The formula for the r−th derivative of a Bezier curve is given in Equation (2.23).

$$\Delta^{\,r} \boldsymbol{b}_i = \sum_{k=1}^{r} C(r,k)(-1)^{r-k} \boldsymbol{b}_{i+k} \tag{2.22}$$

$$\frac{d^r}{dt^r}\, \boldsymbol{b}^n(t) = \frac{n!}{(n-r)!} \sum_{j=0}^{n-r} \Delta^{\,r} \boldsymbol{b}_j\, B_j^{n-r}(t) \tag{2.23}$$

Of particular interest are the derivatives at the ends of the curve. These correspond to t=0 and t=1. Evaluating Equation (2.23) at t=0 and t=1, and recalling the endpoint interpolation property of Bernstein polynomials, Equations (2.24) and (2.25) result:

$$\frac{d^r}{dt^r}\, b^n(0) = \frac{n!}{(n-r)!}\Delta^r b_0 \tag{2.24}$$

$$\frac{d^r}{dt^r}\, b^n(1) = \frac{n!}{(n-r)!}\Delta^r b_{n-r} \tag{2.25}$$

## Subdivision of a Bezier Curve

Subdivision is the process by which a single Bezier curve of degree n, defined over the interval t∈[0,1] is divided into two curves of the same degree defined over the intervals t∈[0,a] and t∈[a,1]. The following development is due to Farin [7].

The first curve segment ( t=0 to t= a ) will be denoted as $\mathbf{a}^n$ and will have control vertices $\mathbf{a}_0,\ldots,\mathbf{a}_n$. A local parameter v will be introduced such that v = t/a; therefore, as v varies from 0 to 1, t varies from 0 to a. Since the curve segment $\mathbf{a}^n$ is part of the same curve segment as $\mathbf{b}^n$, all of their derivatives at t = v = 0 must agree:

$$\frac{d^r}{dt^r}\, b^n(0) = \frac{d^r}{dv^r}\, a^n(0)\,; \quad r = 0,\ldots,n \tag{2.26}$$

From Equation (2.23), the derivatives in Equation (2.26) depend only on the control points $\mathbf{a}_0,\ldots,\mathbf{a}_r$ and $\mathbf{b}_0,\ldots,\mathbf{b}_r$. Now, if these two sets of r+1 points are taken as control points of two degree−r Bezier curves, then then two curves are the same for all v and t because of Equation (2.26):

$$a_0^r(v) = b_0^r(t) \tag{2.27}$$

Finally, from Equation (2.27) with v=1 and t=a,

$$a_0^r(1) = b_0^r(a) \tag{2.28}$$

The left hand side of Equation (2.28) is actually $\mathbf{a}_r$ ( the control points of the first Bezier curve segment ). Therefore,

$$a_j = b_0^j(v) \ . \tag{2.29}$$

Equation (2.29) is known as the *subdivision formula* for Bezier curves. This allows the de Casteljau algorithm to compute the control polygon for the curve segment corresponding to the interval [0,a]. Because of the symmetry property given by Equation (2.20), the polygon of the curve segment corresponding to the interval [a,1] is composed of the vertices $b_{n-j}^j$ .

## Parametric Spline Curves

The spline curve derives its name from a drafting tool called a spline. This is a thin piece of wood, metal, or some other flexible material which is made to pass through a series of established *control points* on the drawing of interest. While the spline is held in place by weights, the designer uses it as a guide to draw a smooth curve passing through the designated points.

The mathematical spline is a smooth piecewise polynomial curve which can be drawn through any set of points. The degree of the spline is determined by the degree of the curves which make up each span of the spline. There should be no kinks or rapid changes in curvature in the spline curve. The spline curve will be denoted by **s**(u) where u is the global parameter of the spline. On each span of the spline, the curve will be written in terms a local

parameter t. Therefore, on an particular span, the spline will be a function of the local parameter t so that s(u) can be written as

$$x = x(t) \qquad y = y(t) \qquad z = z(t) .$$

The locations in parametric space where the spans of the spline join are known as knots. Derivative continuity is prescribed at the knots. Quadratic splines are only capable of first derivative continuity ( $C^1$ ), while cubic splines may be $C^2$. The set of knot values for the entire spline is called the knot vector and will be denoted by U. Individual components of the knot vector are given by $u_i$ where i denotes the location of the knot within the vector.

## Global and Local Parameters

It will be useful to define two parameters for a spline curve, a global parameter u and a local parameter t. The global parameter u varies continuously over the entire spline curve, while the local parameter t is constrained to the interval [0,1] on each span of the spline. The relation between the global and local parameters is illustrated by Figure 2.2 and Equation (2.30).

$$t = \frac{u - u_i}{u_{i+1} - u_i} \tag{2.30}$$

It is easy to confirm that over any parametric interval $[u_i,u_{i+1}]$, t will vary over the interval [0,1]. Thus, the spline curve can now be written as $s_i(t)$ where the i denotes the i–th segment of the spline, and $s(u) = s_i(t)$.

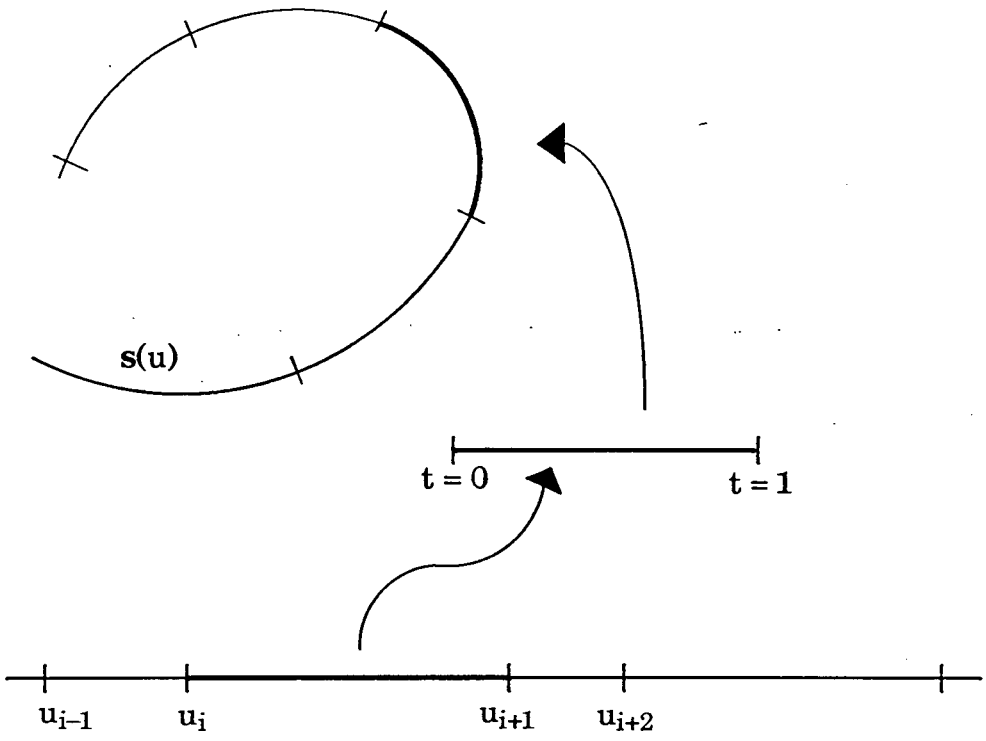



Figure 2.2 Interplay between global and local parameters for a composite curve

## The Cubic Polynomial Spline

The cubic polynomial spline splines a set of data points, $\mathbf{x}_i$, with the parameter u. On any interval $[u_i,u_{i+1}]$, the spline curve $\mathbf{s}(u) = \mathbf{s}_i(t)$ is given by

$$s_i(t) = \frac{1}{6}c_i(1-t)^3 + \frac{1}{6}c_{i+1}t^3 + (x_i - \frac{1}{6}c_i)(1-t) + (x_{i+1} - \frac{1}{6}c_{i+1})t\,. \quad (2.31)$$

The $\mathbf{c}_i$ in Equation (2.31) are found by the tridiagonal system given below in Equation (2.32).

$$c_{i-1} + 4c_i + c_{i+1} = 6(x_{i+1} - 2x_i + x_{i-1}) \qquad (2.32)$$

Since the solution of Equation (2.32) is necessary to compute the spline, any change in the data points $\mathbf{x}_i$ results in changing the *entire* spline curve. This means that the polynomial spline lacks the property of *local control.*

The ends of the spline require some special treatment. Several types of end conditions can be prescribed including zero curvature or natural end conditions, constant curvature or quadratic end conditions, and specified slopes. Details on the various types of end conditions can be found in Ref[22].

Perhaps the most widely used end condition type is the quadratic end condition. It will be used here as an example. In this case, Equation (2.32) is replaced by

$$c_2 = x_3 - 2x_2 + x_1$$
$$c_{NI-1} = x_{NI} - 2x_{NI-1} + x_{NI-2}$$

for i = 2 and i = NI where NI is the number of data points on the spline curve. The values of **c** for the first and last points are determined by extrapolation as follows:

$$c_1 = 2c_2 - c_3$$

$$c_{NI} = 2c_{NI-1} - c_{NI-2}$$

## Composite Bezier Curves

As mentioned earlier, single Bezier curves are not capable of describing complex shapes unless they are of prohibitively high degree. However, the use of composite Bezier curves will allow complex shapes to be modeled while keeping computational requirements within reason. In addition, the concepts of quadratic and cubic B–spline curves can be developed directly from composite Bezier curves. The following development is based on that of Farin [7] and uses his notation.

### Continuity at Junction Points

Since individual Bezier curves are polynomials, they are continuous in all derivatives. However, for composite Bezier curves, continuity at the junction points must be prescribed. The cases of $C^1$ and $C^2$ continuity conditions at these junction points provide a basis for the development of quadratic and cubic B–spline curves respectively.

Let two Bezier curves $\mathbf{s}_0$ and $\mathbf{s}_1$ be described by control polygons $\mathbf{b}_0,\ldots,\mathbf{b}_n$ and $\mathbf{b}_n,\ldots,\mathbf{b}_{2n}$ respectively and defined over the intervals $[u_0,u_1]$ and $[u_1,u_2]$. If these curves are the product of a subdivision process, this means that the curves $\mathbf{s}_0$ and $\mathbf{s}_1$ are part of a single polynomial curve defined over the interval $[u_0,u_2]$. Therefore the control vertices of the two curves are related by Equation (2.33):

$$b_{n+i} = b^i_{n-i}(t), \qquad i = 0,\ldots,n \tag{2.33}$$

where

$$t = \frac{u_2 - u_0}{u_1 - u_0} .$$

Since the two n–th degree Bezier curves are part of one n–th degree polynomial, the two curves must agree in all derivatives up to the n–th derivative. From Equation (2.23), the n–th derivative of a Bezier curve depends only on the surrounding n control vertices. Therefore, using that knowledge and Equation (2.33), the $C^r$ condition for the junction point of two Bezier curves can be constructed.

Given two Bezier curves defined over the intervals $[u_0,u_1]$ and $[u_1,u_2]$ and by the control polygons $\mathbf{b}_0,...,\mathbf{b}_n$ and $\mathbf{b}_n,...,\mathbf{b}_{2n}$ respectively. They are r times continuously differentiable at $u_1$ iff

$$\boldsymbol{b}_{n+i} = \boldsymbol{b}^i_{n-i}(t), \qquad i = 0,\ldots,r \tag{2.34}$$

where

$$t = \frac{u_2 - u_0}{u_1 - u_0} .$$

The $C^r$ condition can also be formulated by equating derivatives at the junction point. From Equation (2.23),

$$\left(\frac{1}{\Delta_0}\right)^i \Delta^i \boldsymbol{b}_{n-i} = \left(\frac{1}{\Delta_1}\right)^i \Delta^i \boldsymbol{b}_n, \qquad i = 0, ...,r \ . \tag{2.35}$$

Note that the derivatives here are with respect to the global parameter u and not the local parameter t, therefore the chain rule applies.

### $C^1$ Continuity

From Equation (2.35) with i = 0,1,

$$\boldsymbol{b}_n = \boldsymbol{b}_n$$

and

$$\Delta_1 \Delta \boldsymbol{b}_{n-1} = \Delta_0 \Delta \boldsymbol{b}_n \tag{2.36}$$

or, expanding the second equation,

$$(u_2 - u_1)(\boldsymbol{b}_n - \boldsymbol{b}_{n-1}) = (u_1 - u_0)(\boldsymbol{b}_{n+1} - \boldsymbol{b}_n) \ .$$

As can be seen from Equation (2.36), the three points $\mathbf{b}_{n-1}$, $\mathbf{b}_n$, $\mathbf{b}_{n+1}$ must be colinear; however, this is not sufficient to guarantee $C^1$ continuity. The additional requirement is that the three points be in the ratio $\delta_0 : \delta_1$. This is illustrated in Figure 2.3 . The $C^1$ condition can be used to determine a formula for

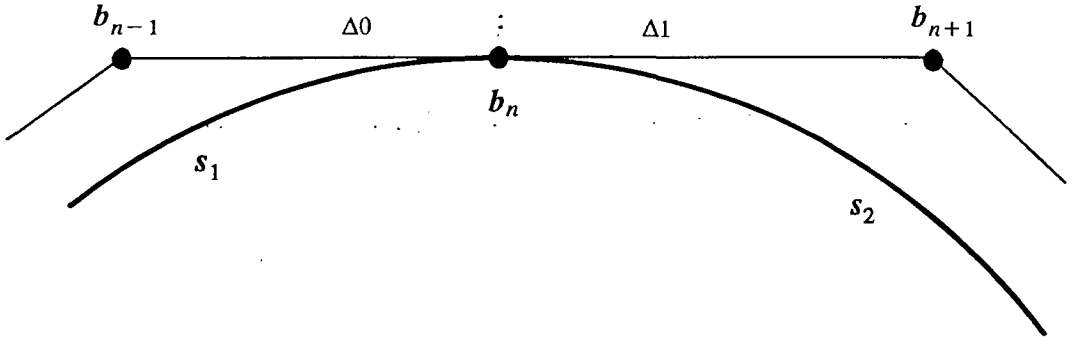



Figure 2.3 The $C^1$ condition for composite Bezier curves.

computing the location of the junction point $\mathbf{b}_n$ given the two surrounding control vertices $\mathbf{b}_{n-1}$ and $\mathbf{b}_{n+1}$. This formula is given by Equation (2.37) and will be used shortly to develop the concept of a quadratic B–spline curve.

$$b_n = \frac{\Delta_1}{\Delta_0 + \Delta_1} b_{n-1} + \frac{\Delta_0}{\Delta_0 + \Delta_1} b_{n+1} \tag{2.37}$$

or

$$b_n = \frac{u_2 - u_1}{u_2 - u_0} b_{n-1} + \frac{u_1 - u_0}{u_2 - u_0} b_{n+1}$$

## $C^2$ Continuity

If a given curve is $C^1$ at the knot $u_1$ then from Equation (2.35), the additional requirement for $C^2$ continuity is that an auxiliary point $\mathbf{d}$ be uniquely defined by Equations (2.38) and (2.39).

$$b_{n-1} = (1 - t_1)b_{n-2} + t_1 d \tag{2.38}$$

$$b_{n+1} = (1 - t_1)d + t_1 b_{n+2} \tag{2.39}$$

where

$$t_1 = \frac{u_1 - u_0}{u_2 - u_0} .$$

Note that $t_1$ is actually the local parameter of $u_1$ with respect to the interval $[u_0,u_2]$. The polygon $\mathbf{b}_{n-2}$, $\mathbf{d}$, $\mathbf{b}_{n+2}$ describes a single quadratic polynomial over the interval $[u_0,u_2]$. Figure 2.4 shows a curve that is $C^2$ at the junction point $\mathbf{b}_n$:

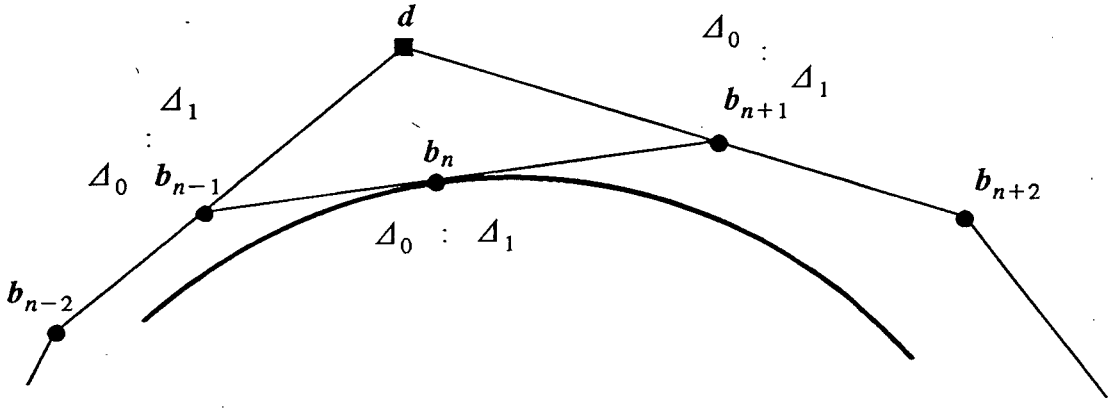



Figure 2.4 The $C^2$ condition for two degree–n Bezier curves.

## B–spline Curves

### Quadratic B–spline Curves from Composite Quadratic Bezier Curves

If some quadratic spline curve $\mathbf{s}$ defined over the interval $[u_0,u_L]$ is assumed to be $C^1$ then the spline curve can be completely defined by the knot vector u and the inner Bezier points $\mathbf{b}_0$, $\mathbf{b}_1$, $\mathbf{b}_3$,..., $\mathbf{b}_{2i+1}$,...,$\mathbf{b}_{2L-1}$, $\mathbf{b}_{2L}$. The addi-

tional points ( junction points ) required to define the composite quadratic Bezier curve can be determined by the $C^1$ condition given in Equation (2.36). Solving Equation (2.36) for the junction point $\mathbf{b}_{2i}$ in terms of the inner Bezier points, Equation (2.40) results:

$$b_{2i} = \frac{\Delta_i}{\Delta_{i-1} + \Delta_i} b_{2i-1} + \frac{\Delta_{i-1}}{\Delta_{i-1} + \Delta_i} b_{2i+1} \qquad (2.40)$$

The polygon $\mathbf{b}_0$, $\mathbf{b}_1$, $\mathbf{b}_3$,..., $\mathbf{b}_{2i+1}$,...,$\mathbf{b}_{2L-1}$, $\mathbf{b}_{2L}$ can now be called the B–spline control polygon of the spline curve **s**. This polygon, along the its knot vector u, completely describe the quadratic spline curve **s**. To indicate that the control polygon now denotes a B–spline curve, the control polygon will be denoted by $\mathbf{d}_{-1}$, $\mathbf{d}_0$, $\mathbf{d}_1$,...,$\mathbf{d}_{L-1}$, $\mathbf{d}_L$. Figure 2.5 illustrates the relationship between the B–spline control polygon and the Bezier control net for the quadratic case. Note that the polygon for the curve is stored as follows:

$$d_{-1} = b_0$$
$$d_L = b_{2L}$$

With the inner vertices given by

$$d_i = b_{2i+1} \qquad i = 0, \ldots L - 1 \ .$$

### $C^2$ Cubic B–spline Curve from Composite Cubic Bezier Curve

Consider a cubic spline curve defined over the knot sequence $[u_0,u_L]$. In order for the curve to be $C^1$, Equation (2.40) must hold. For the cubic case, the $C^1$ condition is given by Equation (2.41).

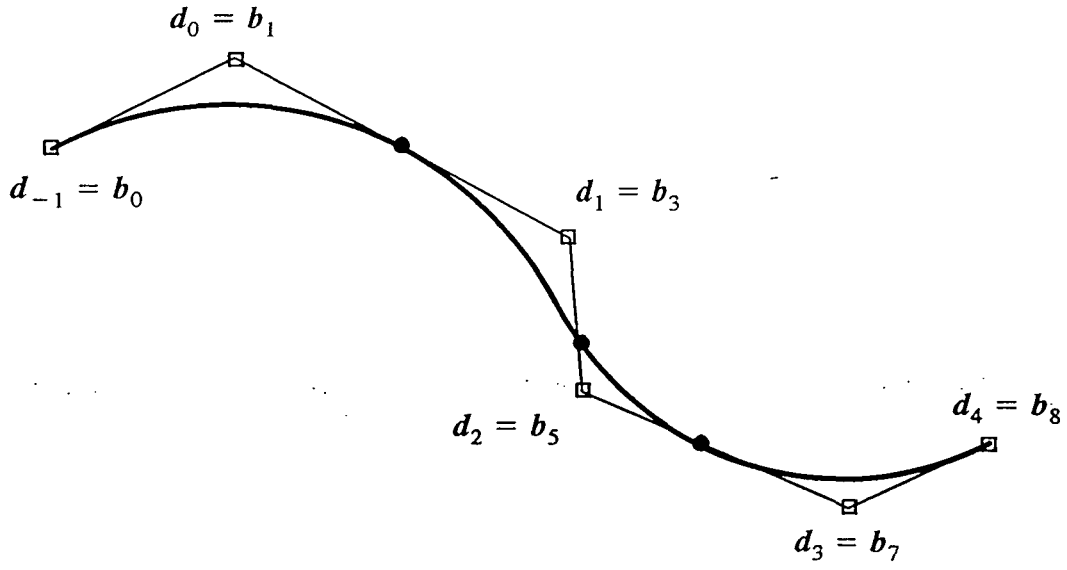



Figure 2.5 Quadratic B–spline curve with its Bezier control polygon.

The additional condition for $C^2$ continuity, previously given as Equations (2.38) and (2.39), is now given by Equations (2.42) and (2.43):

$$b_{3i} = \frac{\Delta_i}{\Delta_{i-1} + \Delta_i} b_{3i-1} + \frac{\Delta_{i-1}}{\Delta_{i-1} + \Delta_i} b_{3i+1} \tag{2.41}$$

$$b_{3i-2} = \frac{\Delta_{i-1} + \Delta_i}{\Delta_{i-2} + \Delta_{i-1} + \delta_i} d_{i-1} + \frac{\Delta_{i-2}}{\Delta_{i-2} + \Delta_{i-1} + \Delta_i} d_i \tag{2.42}$$

$$b_{3i-1} = \frac{\Delta_i}{\Delta_{i-2} + \Delta_{i-1} + \Delta_i} d_{i-1} + \frac{\Delta_{i-2} + \Delta_{i-1}}{\Delta_{i-2} + \Delta_{i-1} + \Delta_i} d_i \tag{2.43}$$

The ends of the spline are treated such that the first and last control vertices of the spline are coincident with the first and last points of the curve respec-

tively. Equations (2.41) through (2.43) are valid for i=1,...,L–1. For the end points of the curve, Equations (2.44) are used.

$$
\begin{aligned}
b_0 &= d_{-1} \\
b_1 &= d_0 \\
b_2 &= \frac{\Delta_1}{\Delta_0 + \Delta_1} d_0 + \frac{\Delta_0}{\Delta_0 + \Delta_1} d_1 \\
b_{3L-2} &= \frac{\Delta_{L-1}}{\Delta_{L-2} + \Delta_{L-1}} d_{L-1} + \frac{\Delta_{L-2}}{\Delta_{L-2} + \Delta_{L-1}} d_L \\
b_{3L-1} &= d_L \\
b_{3L} &= d_{L+1}
\end{aligned}
\tag{2.44}
$$

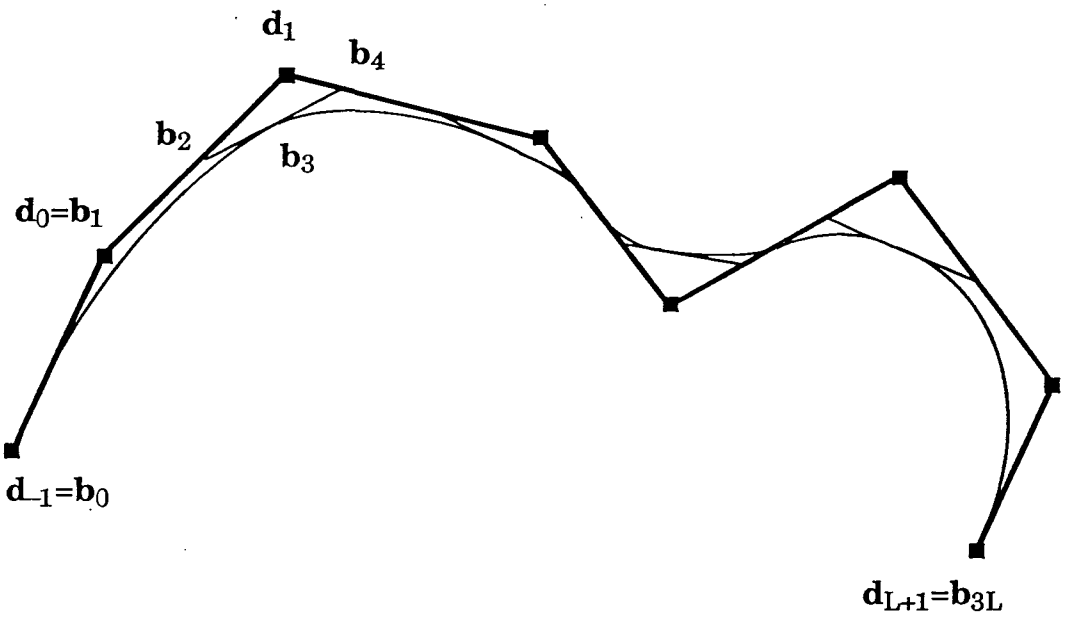



Figure 2.6 Cubic B–spline curve with its control vertices and Bezier points.

The B–Spline Basis

Further theoretical development of B–splines requires defining them in more analytical terms. Equation (2.45) is the Cox, de Boor recursion for B–spline curves. This formula can be used to calculate B–spline basis functions for splines of any degree. When Equation (2.45) is expanded for the cubic

$$N_i^n(u) = \frac{u - u_{i-1}}{u_{i+n-1} - u_{i-1}} N_i^{n-1}(u) + \frac{u_{i+n} - u}{u_{i+n} - u_i} N_{i+1}^{n-1}(u) \qquad (2.45)$$

with

$$N_i^0(u) = \begin{cases} 1 & \text{if } u_{i-1} \le u < u_i \\ 0 & else \end{cases}$$

case, Equation (2.46) results. Using Equation (2.45), the equation of a B–spline curve for a given set of control vertices and a given knot vector is given as Equation (2.47) where L is the number of interval on the spline. Note that Equations (2.45) and (2.47) are given in terms of the global parameter u of the spline. If these equations are cast in terms of the local coordinate t of the curve instead of the global coordinate u, then Equation (2.47) becomes Equation (2.48) where t is the local coordinate of the ith segment of the curve and is given by Equation (2.30). For the case of a cubic curve, the matrix form of Equation (2.48) is given by Equation (2.49). Where $M_3$ is a 3x3 matrix given by Equation (2.50).

$$N_i^3(u) = \begin{cases} \dfrac{(u-u_i)^3}{(u_{i+3}-u_i)\,(u_{i+2}-u_i)\,(u_{i+1}-u_i)} & u \in [u_i, u_{i+1}] \\ \dfrac{(u-u_i)^3}{(u_{i+3}-u_i)\,(u_{i+2}-u_i)\,(u_{i+1}-u_i)} + \dfrac{(u-u_{i+1})^3\,(u_{i+4}-u_i)}{(u_{i+1}-u_i)\,(u_{i+1}-u_{i+2})\,(u_{i+1}-u_{i+3})\,(u_{i+1}-u_{i+4})} & u \in [u_{i+1}, u_{i+2}] \\ \dfrac{(u_{i+3}-u)^3\,(u_{i+4}-u_i)}{(u_{i+3}-u_i)\,(u_{i+3}-u_i)\,(u_{i+3}-u_{i+2})\,(u_{i+3}-u_{i+4})} + \dfrac{(u_{i+4}-u)^3}{(u_{i+4}-u_{i+1})\,(u_{i+4}-u_{i+2})\,(u_{i+4}-u_{i+3})} & u \in [u_{i+2}, u_{i+3}] \\ \dfrac{(u_{i+4}-u)^3}{(u_{i+4}-u_{i+1})\,(u_{i+4}-u_{i+2})\,(u_{i+4}-u_{i+3})} & u \in [u_{i+3}, u_{i+4}] \\ 0 & u \notin [u_i, u_{i+4}] \end{cases} \tag{2.46}$$

$$P(u) = \sum_{i=0}^{L+n-1} d_i N_i^n(u) . \tag{2.47}$$

$$P_i(t) = \sum_{j=i-1}^{i+n-1} d_j N_j^n(t) . \tag{2.48}$$

$$P_i(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} M_3 \begin{bmatrix} d_{i-1} \\ d_i \\ d_{i+1} \\ d_{i+2} \end{bmatrix} . \tag{2.49}$$

Derivatives of B–spline Curves

For purposes of interrogation and smoothing, the derivatives of a curve should be calculated with respect to the curves global parameter. However, Equation (2.48) and, in particular, Equation (2.49) are given in terms of a local parameter. This requires the use of the chain rule in order to determine derivatives with respect to the global parameter u of the curve. Consider a spline curve $\mathbf{s}_i(t)$ where t is the local parameter of the curve in the interval $[u_i, u_{i+1}]$. The first derivative of the curve with respect to the global parameter u is given by Equation (2.51). Higher order derivatives are given by Equations (2.52) and (2.53).

For the case of cubic B–spline curves defined by Equation (2.49), first, second, and third derivatives are given by Equations (2.54), (2.55), and (2.56) respectively.

$$M_3 = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \tag{2.50}$$

$$m_{11} = \frac{(u_i - u_{i+1})^2}{(u_{i+1} - u_{i-2})\,(u_{i+1} - u_{i-1})} \qquad m_{12} = 1 - (\,m_{11} + m_{13}\,)$$

$$m_{21} = -3m_{11} \qquad m_{22} = -(\,m_{21} + m_{23}\,)$$

$$m_{32} = -(\,m_{31} + m_{33}\,)$$

$$m_{31} = 3m_{11} \qquad m_{42} = -(\,m_{41} + m_{43} + m_{44}\,)$$

$$m_{41} = -m_{11} \qquad m_{14} = 0$$

$$m_{13} = \frac{(u_{i-1} - u_i)^2}{(u_{i-1} - u_{i+2})\,(u_{i-1} - u_{i+1})} \qquad m_{24} = 0$$

$$m_{23} = \frac{3(u_{i+1} - u_i)\,(u_i - u_{i-1})}{(u_{i-1} - u_{i+2})\,(u_{i-1} - u_{i+1})} \qquad m_{34} = 0$$

$$m_{33} = \frac{3(u_{i+1} - u_i)^2}{(u_{i-1} - u_{i+2})\,(u_{i-1} - u_{i+1})} \qquad m_{44} = \frac{(u_{i+1} - u_i)^2}{(u_i - u_{i+3})\,(u_i - u_{i+2})}$$

$$m_{43} = -(u_{i+1} - u_i)^2$$

$$\left[\frac{1}{(u_{i-1} - u_{i+1})\,(u_{i-1} - u_{i+2})} + \frac{1}{(u_i - u_{i+2})\,(u_i - u_{i+3})} + \frac{1}{(u_{i+2} - u_{i-1})\,(u_{i+2} - u_i)}\right]$$

$$\frac{ds(u)}{du} = \frac{ds_i(t)}{dt}\frac{dt}{du} = \frac{1}{\Delta_i}\frac{ds_i(t)}{dt} \tag{2.51}$$

$$\frac{d^2s(u)}{du^2} = \frac{1}{(\Delta_i)^2}\frac{d^2s_i(t)}{dt^2} \tag{2.52}$$

$$\frac{d^3s(u)}{du^3} = \frac{1}{(\Delta_i)^3}\frac{d^3s_i(t)}{dt^3} \tag{2.53}$$

$$\frac{d}{du}\boldsymbol{P}_i(t) = \frac{1}{\Delta_i}\begin{bmatrix}0 & 1 & 2t & 3t^2\end{bmatrix} M_3 \begin{bmatrix}\boldsymbol{d}_{i-1}\\ \boldsymbol{d}_i\\ \boldsymbol{d}_{i+1}\\ \boldsymbol{d}_{i+2}\end{bmatrix} \tag{2.54}$$

$$\frac{d^2}{du^2}\boldsymbol{P}_i(t) = (\frac{1}{\Delta_i})^2\begin{bmatrix}0 & 0 & 2 & 6t\end{bmatrix} M_3 \begin{bmatrix}\boldsymbol{d}_{i-1}\\ \boldsymbol{d}_i\\ \boldsymbol{d}_{i+1}\\ \boldsymbol{d}_{i+2}\end{bmatrix} \tag{2.55}$$

$$\frac{d^3}{du^3}\boldsymbol{P}_i(t) = (\frac{1}{\Delta_i})^3\begin{bmatrix}0 & 0 & 0 & 6\end{bmatrix} M_3 \begin{bmatrix}\boldsymbol{d}_{i-1}\\ \boldsymbol{d}_i\\ \boldsymbol{d}_{i+1}\\ \boldsymbol{d}_{i+2}\end{bmatrix} \tag{2.56}$$

# CHAPTER III

# SURFACES

Curves are defined in terms of a one–dimensional parameter space with global parameter u and local parameter t. Surfaces, however, are defined in terms of a two–dimensional parameter space with global parameters u and v and local parameters s and t. There are many ways to represent surface patches, however, this work will concentrate only on the tensor product form.

## Bicubic Hermite Surface Patches

Surface patches of the kind found in the early versions of the EAGLE surface code are bicubic Hermite patches. These are given by equation (3.1):

$$P(s,t) = B(s)HB^T(t) \tag{3.1}$$

where

$$H = \begin{bmatrix} r(0,0) & r(1,0) & r_u(0,0) & r_u(1,0) \\ r(0,1) & r(1,1) & r_u(0,1) & r_u(1,1) \\ r_v(0,0) & r_v(1,0) & r_{uv}(0,0) & r_{uv}(1,0) \\ r_v(0,1) & r_v(1,1) & r_{uv}(0,1) & r_{uv}(1,1) \end{bmatrix}$$

$$B(a) = \begin{bmatrix} 1 - 3a^2 + 2a^3 \\ 3a^2 - 2a^3 \\ a - 2a^2 + a^3 \\ -a^2 + a^3 \end{bmatrix} \quad , a = s,t$$

Here, s and t are local coordinates of the spline and the global coordinates are taken as the integer indices of the grid point. The parametric space of the spline is shown in figure Figure 3.1 .

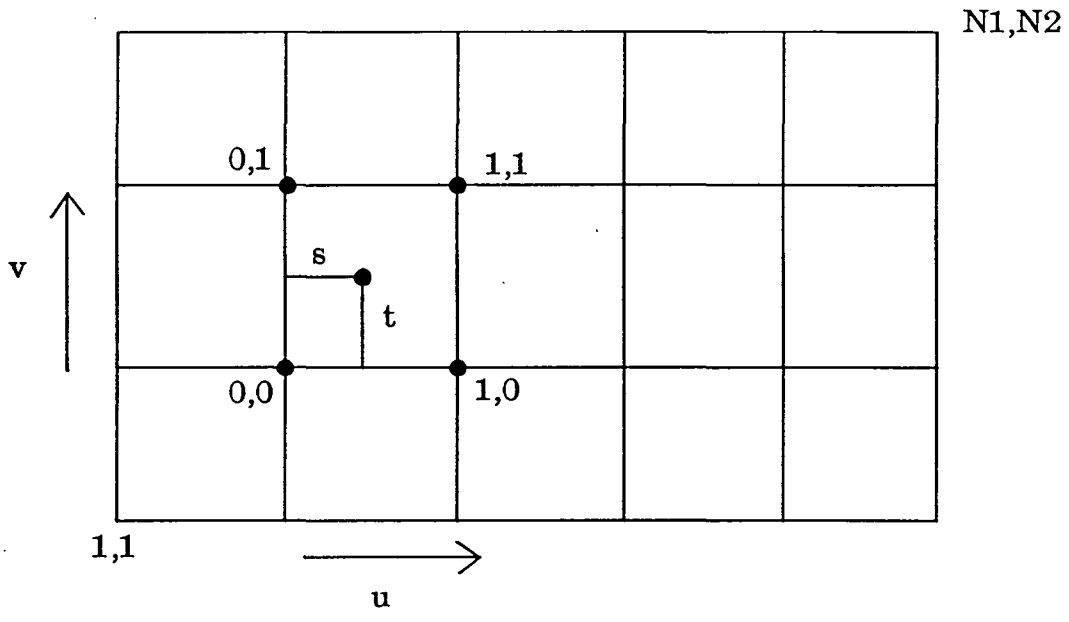



Figure 3.1 Parametric space of a bicubic Hermite surface patch

Tensor Product Bezier Surfaces

Recent versions of the EAGLE code include the capability of generating tensor product Bezier surfaces (Ref[18]). This type of surface patch is a simple extension of the Bezier curve technique. The mathematical form of a tensor product Bezier surface patch is given by equation (3.2).

$$b^{m,n}(s,t) = \sum_{i=0}^{m}\sum_{j=0}^{n} b_{i,j} B_i^m(s) B_j^n(t) \qquad (3.2)$$

A set of (n+1)x(m+1) control points define the surface. The surface may be

generated by treating each row of the control net as a degree–n Bezier curve, then treating each column of the result as a degree–m Bezier curve. This amounts to generating a surface by using a curve moving through space and changing its shape. The matrix form of equation (3.2) for the cubic case is given by equation (3.3):

$$P(s,t) = SM_{bz}BM_{bz}^{T}T^{T} \tag{3.3}$$

$$B = \begin{bmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{10} & b_{11} & b_{12} & b_{13} \\ b_{20} & b_{21} & b_{22} & b_{23} \\ b_{30} & b_{31} & b_{32} & b_{33} \end{bmatrix}, \qquad M_{bz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & -3 & 1 \end{bmatrix}$$

$$S = [\,1 \;\; s \;\; s^2 \;\; s^3] \qquad T = [\,1 \;\; t \;\; t^2 \;\; t^3]$$

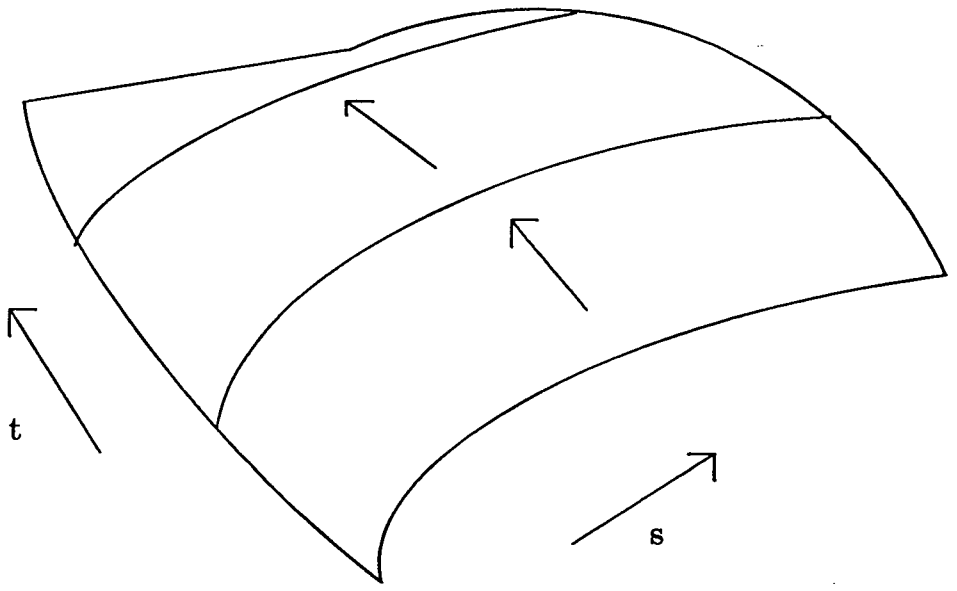



Figure 3.2 Surface created by a moving and deforming curve.

Tensor Product B–spline Surfaces

Recent versions of the EAGLE code also have the capability of generating tensor product B–spline surfaces (Ref[18]). The relationship between B–spline surfaces and B–spline curves is the same as that between Bezier surfaces and Bezier curves. Equation (3.4) gives the mathematical form for the surface patch:

$$P(s,t) = \sum_{i=0}^{m}\sum_{j=0}^{n} d_{i,j} N_i^m(s) N_j^n(t) \tag{3.4}$$

Note that the basis functions N are the tensor product of the set of univariate basis functions given by equation (2.45). The matrix form for the cubic case is given by equation (3.5):

$$P(s,t) = SM_3DM_3^TT^T \tag{3.5}$$

where **D** is a 4 x 4 matrix of control points and $M_3$ is the same as in equation (2.50). The vectors S and T are composed of the *local* parameters s and t of the point to be generated.

B–spline surfaces inherit all the properties possessed by B–spline curves. This is due to the fact that the tensor product method can be broken into a series of univariate operations each of which can be interpreted as generating a B–spline curve. Note that there are limitations to the tensor product method. Perhaps the most restrictive of these is the necessity that only one knot vector can be prescribed for each direction on the surface. This is to say that all u isoparametric curves must have the same knot vector and likewise, all curves of constant v must be defined by a single knot sequence. Figure 3.3

illustrates the parametric space of a surface which is definable by a tensor product method, while Figure 3.4 shows the parameter space of one which cannot be represented by a tensor product:

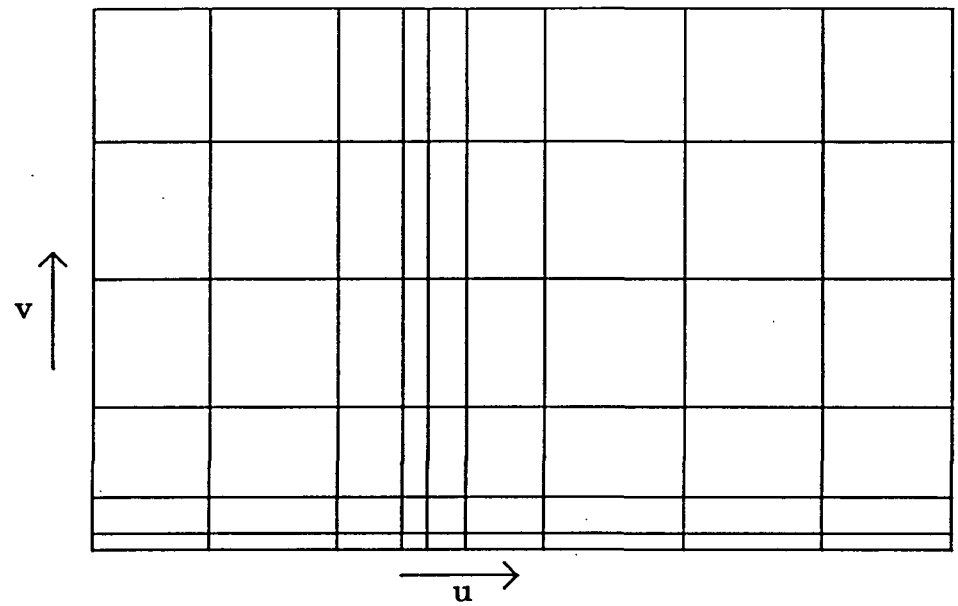



Figure 3.3 Knot matrix of a surface spline which can be represented by a tensor product

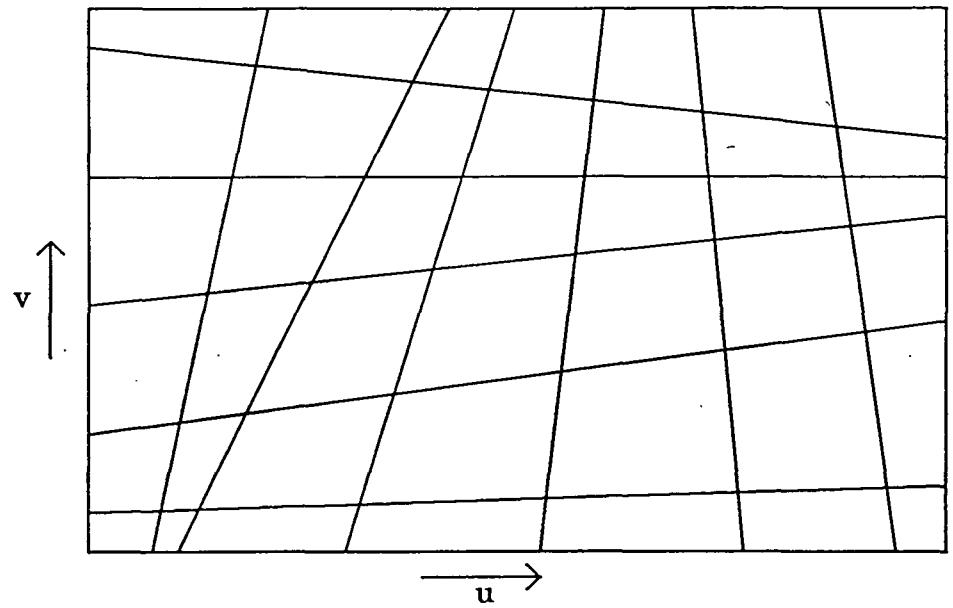



**Figure 3.4 Knot matrix of a surface spline which cannot be represented by a tensor product**

# CHAPTER IV

# INTERROGATION AND SMOOTHING

Chapters II and III have described some curve and surface modeling techniques. However, if given a particular geometry, how can one access the quality of that geometry, in terms of derivative continuity, and if necessary, improve upon it? Attempts to provide a practical solution to this problem have resulted in a variety of techniques ( see Chapter I ). This work, however, will focus on techniques provided by Farin[6],[7].

## Interrogation of B–spline Curves

Several means of curve interrogation have been suggested. Curvature plots provide a great deal of information about the shape of a curve. However, their use is limited to planar or two dimensional geometries because curvature is always nonnegative for space curves. The same two–dimensional limitation applies to k–orthotomic curves. What is needed is a quantity which is highly sensitive to small perturbations in the geometry but is not limited by spatial dimension. The third derivative provides such a quantity and will be used for interrogation of space curves. For planar curves however, signed curvature will be used.

Calculation of derivatives is effected by use of equations (2.54) through (2.56). For planar curves, curvature is calculated via equation (4.1). For determining derivative continuity, derivatives or curvature must only be calculated at the spline knots. All points on the interior of each span of the spline are defined by polynomials and thus are continuous in all derivatives.

Farin[7] suggests that a fair curve should be composed of segments with smoothly varying curvature ( i.e. there should be no slope discontinuities in the curvature plot of a curve ). For planar curves, this definition is sufficient, however, for space curves, the second derivative will be used in lieu of signed curvature. Slope discontinuities in second derivative are actually third derivative discontinuities.

$$\varkappa(u) = \frac{x''(u)y'(u) - y''(u)x'(u)}{[(x'(u))^2 + (y'(u))^2]^{3/2}} \tag{4.1}$$

The method used to locate discontinuities is to calculate the left and right hand limits of the third derivative at each spline knot and then compare them. This will be the basis for a quantity known as *local fairness* and leads to the following definition:

Definition: Let x(t) be a $C^2$ parametric cubic piecewise curve with u as the global parameter. Then the local fairness ε is defined as

$$\varepsilon = \| x'''(u_+) - x'''(u_-) \|. \tag{4.2}$$

Note that ε is a local quantity since it may vary with the value of the parameter u. It is reasonable to say that the point most in need of smoothing is the point with the largest value of ε.

**A separate routine for interrogation of curves was not produced. The method of interrogation presented here was incorporated into the smoothing routine outlined below.**

## Interrogation of Surfaces

**The maximal and minimal curvatures of a surface at a point P are known as the principle curvatures and are denoted by $\varkappa_1$ and $\varkappa_2$. These curva-**

tures are calculated by finding the roots of equation (4.3) where the superscripts on **P** denote differentiation with respect to the parameter indicated.

$$(EG - F^2)\varkappa^2 - (EN + GL - 2FM)\varkappa + (LN - M^2) = 0 \tag{4.3}$$

where

$$E = \boldsymbol{P}^u \cdot \boldsymbol{P}^u \qquad L = \boldsymbol{P}^{uu} \cdot \boldsymbol{n}$$
$$F = \boldsymbol{P}^u \cdot \boldsymbol{P}^v \qquad M = \boldsymbol{P}^{uv} \cdot \boldsymbol{n}$$
$$G = \boldsymbol{P}^v \cdot \boldsymbol{P}^v \qquad N = \boldsymbol{P}^{vv} \cdot \boldsymbol{n}$$

and

$$\boldsymbol{n} = \frac{\boldsymbol{P}^u \times \boldsymbol{P}^v}{|\boldsymbol{P}^u \times \boldsymbol{P}^v|} .$$

The two major types of surface curvature are known as mean curvature H and Gaussian curvature K. These are given by equations (4.4) and (4.5) respectively.

$$K = \varkappa_1 \varkappa_2 = \frac{LN - M^2}{EG - F^2} \tag{4.4}$$

$$H = \frac{1}{2}(\varkappa_1 + \varkappa_2) = \frac{EN + GL - 2FM}{2\,(EG - F^2)} \tag{4.5}$$

While these quantities provide a great deal of information, they have limitations. Gaussian curvature is always zero for cylinders of the form given in equation (4.6) while mean curvature is always zero for minimal surfaces.

$$c(u,v) = (1 - u)X(v) + u[X(v) + V] \tag{4.6}$$

Absolute curvature, however, will always detect curvature in a surface. Absolute curvature is given by equation (4.7):

$$\varkappa_{abs} = |\varkappa_1| + |\varkappa_2| \qquad (4.7)$$

A subroutine for evaluation of principal, Gaussian, mean, and absolute curvatures was developed for the EAGLE code. The routine is capable of evaluating surface qualities of either B–spline surfaces or surfaces produced by the original EAGLE surface spline routine, SPLSUR. In the latter case, derivatives on the surface are calculated using the SPLINT subroutine ( see Ref [23] ). The result is two data files, one containing points on the surface and the other containing values of the five curvatures at those points. These files are written in formatted PLOT3D single grid form.

Smoothing of B–spline Curves

In order to fair or smooth at a particular spline knot, the method used must be local in nature. That is, when the method is applied, it only affects the curve in a small region surrounding the point which was smoothed. The technique used here is one proposed by Farin [6], [7]. First, the local fairness is calculated at each knot in the spline; then the knot with the largest value of e is chosen as the knot at which smoothing will take place. Let this knot be the knot associated with the B–spline control point $\mathbf{d}_i$. The knot is then removed from the knot sequence and a new location $\bar{\mathbf{d}}_i$ for the control point $\mathbf{d}_i$ is calculated. Then the knot is reinserted into the knot sequence so that the number of spline segments remains the same as in the original curve. The criterion for the selection of a new location for the control point $\mathbf{d}_i$ is third derivative continuity at the knot. Mathematically, this amounts to equating the

left and right hand limits of the third derivative at the spline knot ( e.g. driving e to zero ). The left and right hand limits of third derivative of a B–spline curve at the ith knot are given by

$$P_i^{uuu}(0) = \frac{6}{(u_{i+1} - u_i)^3}[m_{41}d_{i-1} + m_{42}d_i + m_{43}d_{i+1} + m_{44}d_{i+2}],$$
$$P_{i-1}^{uuu}(1) = \frac{6}{(u_i - u_{i-1})^3}[m_{41}d_{i-2} + m_{42}d_{i-1} + m_{43}d_i + m_{44}d_{i+1}] .$$

The values of coefficients of the **d** terms are calculated by equation (2.50). The new location of the control point $\mathbf{d}_i$ is given by equation (4.8):

$$\bar{d}_i = \frac{(u_{i+2} - u_i)l_i + (u_i - u_{i-2})r_i}{(u_{i+2} - u_{i-2})}, \tag{4.8}$$

with the points $\mathbf{l}_i$ and $\mathbf{r}_i$ given by

$$l_i = \frac{(u_{i+1} - u_{i-3})d_{i-1} - (u_{i+2} - u_{i-2})d_{i-2}}{(u_i - u_{i-3})},$$
$$r_i = \frac{(u_{i+1} - u_{i-3})d_{i+1} - (u_{i+1} - u_i)d_{i+2}}{(u_i - u_{i-3})}.$$

Figure 4.1 illustrates the geometry behind equation (4.8).

The curve smoothing routine developed for EAGLE takes, as input, the control vertices and the knot vector which define a B–spline curve. The routine then smooths the spline as directed. The user may specify the number of smoothing passes to make, as well as a movement tolerance for points on the curve. The user may also specify whether or not the routine is to interrogate the curve and smooth only at points with large discontinuities. If the entire curve is to be smoothed, the user instructs the routine accordingly and all vertices on the spline, with the exception of the first two, and last two are smoothed according to equation (4.8).

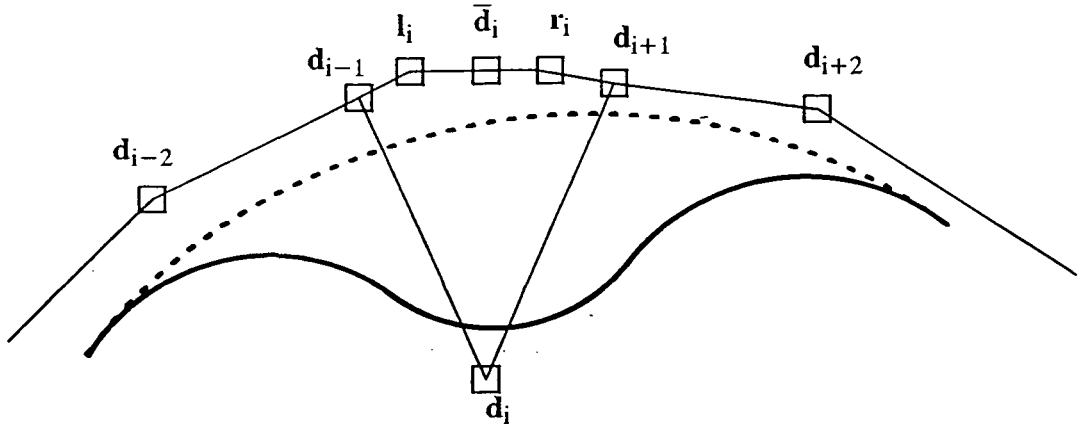



Figure 4.1 Geometric interpretation of the B–spline smoother.

The movement tolerance, mentioned earlier, enables the user to specify the maximum distance which any point on the curve is allowed to move. This option may be used where geometry perturbations must be kept within certain limits. Since the smoother moves the control vertices of the spline, maximum allowable movement of a point on the curve must be translated to an allowable perturbation limit for the curves control vertices. This is accomplished by use of equation (4.9):

$$e_i = \frac{1}{c_i}\delta x_i \tag{4.9}$$

where

$$c_i = \frac{1}{u_{i+1} - u_{i-1}}\left(\delta_i \frac{u_i - u_{i-2}}{u_{i+1} - u_{i-2}} + \delta_{i-1}\frac{u_{i+2} - u_i}{u_{i+2} - u_{i-1}}\right)$$

Here, $\mathbf{e}_i$ is the displacement vector for a control vertex necessary to displace a point on the curve by the vector $\delta\mathbf{x}_i$. Using equation (4.9), a maximum displacement is calculated for each control vertex on the spline. The movement of the control vertices is monitored during the smoothing process to ensure that none of the vertices are displaced more than the allowable tolerance. If equation (4.9) results in moving a control point farther than is allowed by the prescribed tolerance, the point is moved in the desired direction, but only as far as the tolerance allows.

Smoothing of Tensor Product B–spline Surfaces

Fairing of surfaces is accomplished by using a tensor product method. This amounts to using the curve smoothing routine to smooth the curve net that defines the surface. Each curve of constant v is smoothed and the results stored. Then, using the result of the first smoothing pass, each curve of constant u is smoothed. This procedure constitutes one smoothing pass. The resulting surface will be smoother than the original. The tensor product method is discussed in detail in Ref [7], and its application to surface smoothing is discussed in Refs [6] and [7].

The surface smoothing routine developed for EAGLE takes, as input, the knot vectors and control vertices which define a tensor product B–spline surface. The surface is then smoothed using the tensor product method described above. The user may specify that only certain regions of the surface are to be smoothed. A region is defined by giving the indices of the vertices on two opposing corners of the patch. If the patch lies on the interior of the surface, the entire patch is smoothed. However, if one or more of the edges of the patch are coincident with one or more edges of the surface, the first two vertices from the edge of the surface are not smoothed. The user may, as with

curves, specify a maximum allowable displacement. However, for surfaces, this value is the maximum allowable movement for a control vertex, not for a point which lies on the surface.

# CHAPTER V

# RESULTS AND CONCLUSIONS

The material in this chapter is concerned with results obtained using the routines outlined in Chapter IV. Several examples of both curve and surface smoothing are provided as well as the results obtained from the respective interrogation algorithms. Additional information is provided by results obtained from a numerical simulation effected on a NACA 0012 airfoil both before and after the smoothing algorithm had been applied.

## Curve Smoothing Results

### Waisted Body

The waisted body represents a simple geometric entity described by four polynomial segments. Data for this geometry were given as a set of 120 raw data points obtained from evaluation of the piecewise polynomial function describing the curve. The data points were then splined using a non–uniform B–spline; this resulted in a spline curve consisting of 120 control vertices and the associated knot vector. The curve smoothing routine was applied and the results are presented in Figure 5.1 through Figure 5.4.

Figure 5.1 shows the control polygon of the spline both before and after smoothing. As evidenced by Figure 5.1, the geometry is changed very little by the smoothing process, however, Figure 5.2 illustrates the change in the splines curvature plot. Note how slope discontinuities in the curvature plot are virtually eliminated by the smoothing process. Figure 5.3 demonstrates the effect of smoothing on the metric coefficient $\eta_y$. The degree by which the

geometry was perturbed is indicated by Figure 5.4. Note that the maximum point displacement was only about 1.5 percent of the total arclength of the curve. If Figure 5.1 is examined closely, it is hard to distinguish ( visually ) any change in the geometry due to smoothing; however, Figure 5.2 and Figure 5.3 show that the continuity of the spline was substantially improved.

## Digitized Curve

Figure 5.5 is a spline curve which was obtained from points digitized from an physical model. Examination of the curve reveals visible discontinuities in first derivative. A better indication of the splines condition is given by the curvature plots in Figures 5.7 and 5.9.

This data set was smoothed in two different ways. First, the smoother was allowed to pick the point with the largest third derivative discontinuity and smooth there. The results after ten passes using interrogation and point selection are shown in Figures 5.6. and 5.7. The actual spline curve is shown in Figure 5.6, while curvature plots before and after smoothing are shown in Figure 5.7. For the second case, the curve was smoothed without using the interrogation algorithm. In both cases the same number of smoothing passes were used. Figure 5.8 shows the spline after smoothing while Figure 5.9 shows plots of curvature both before and after smoothing.

## NACA 0012 Airfoil

The purpose of including these routines in a numerical grid generation program is to help enhance the quality of the geometry model used for numerical simulation. Therefore, it is appropriate that some indication be given of how smoothing affects the results obtained from simulation programs. With this purpose in mind, data for a NACA 0012 airfoil were taken from Ref[24]

and splined. The data set for the airfoil consisted of 121 data points generated from the spline. The spline was then smoothed and another 121 data points generated. The smoothed and unsmoothed data sets are presented in Figure 5.10. Again, close examination is required in order to visually detect changes in the geometry ( see Figure 5.11 ). Figure 5.12 shows curvatures for the smoothed and unsmoothed geometries.

To test how smoothing effects numerical simulations, an incompressible flow solution was run on both geometries. The development of the code used in to obtain the solution is documented in Ref[25]. Plots of pressure coefficient (CP) vs. X/C at zero degrees angle of attack for the two cases are shown in Figures 5.13 and 5.14. Notice the smooth variation of CP for the modified geometry. Also note that the CP curve for the smoothed airfoil still agrees with experimental data. Color contour plots of the pressure field for the two cases are given in Figures 5.15 and 5.16. The simulation was also run at five degrees angle of attack. Plots of the pressure field near the leading edge of the wing section are shown in Figures 5.17 and 5.18. Notice that, for the unsmoothed geometry, the low pressure region on the upper surface exhibits ripples, whereas the same region on the smoothed geometry does not.

## Surface Smoothing Results

### Flat Plate

A flat plate represents the simplest test for the surface smoother. Figures 5.19 through 5.22 show a plane to which perturbations have been applied. The surface in Figure 5.19 appears to be smooth and free of geometric irregularities. However, examination of its absolute curvature, shown in Figure 5.20, reveals that the surface is actually rippled. After applying the fair-

ing algorithm, the new surface, shown in Figure 5.21 appears unchanged, but the curvature plot in Figure 5.22 shows that the irregularities have been removed.

### Sculpted Surface

Figure 5.23 shows a surface patch digitized from a model of a lifting body vehicle. The surface is fairly smooth as evidenced by the plot of absolute curvature in Figure 5.24. The geometry was smoothed and the results shown in Figures 5.25 and 5.26. Additional options available for surface interrogation are shown by plots of first principal curvature and Gaussian curvature in Figures 5.27 and 5.28 respectively.

### F–15 Aft Quarter

The last example uses a digitized surface from the aft section of an F–15 fighter aircraft. The original surface is plotted in Figure 5.29 and contour plots of Gaussian and absolute curvature are shown in Figures 5.30 and 5.31 respectively. The smoothed surface is shown in Figure 5.32 along with plots of Gaussian and absolute curvature in Figures 5.33 and 5.34 respectively.

## Conclusions and Recommendations

The interrogation and smoothing routines, outlined herein, represent a powerful tool for evaluation of curve and surface quality as well as integrity enhancement of geometry data. These routines are limited, however, since they are unable to account for intended discontinuities. Research is still needed in order to produce an automated method for improving the quality of curve and surface representations without destroying desired geometric irregulari-

ties. At present, the only viable method is the use of these routines in an interactive environment.

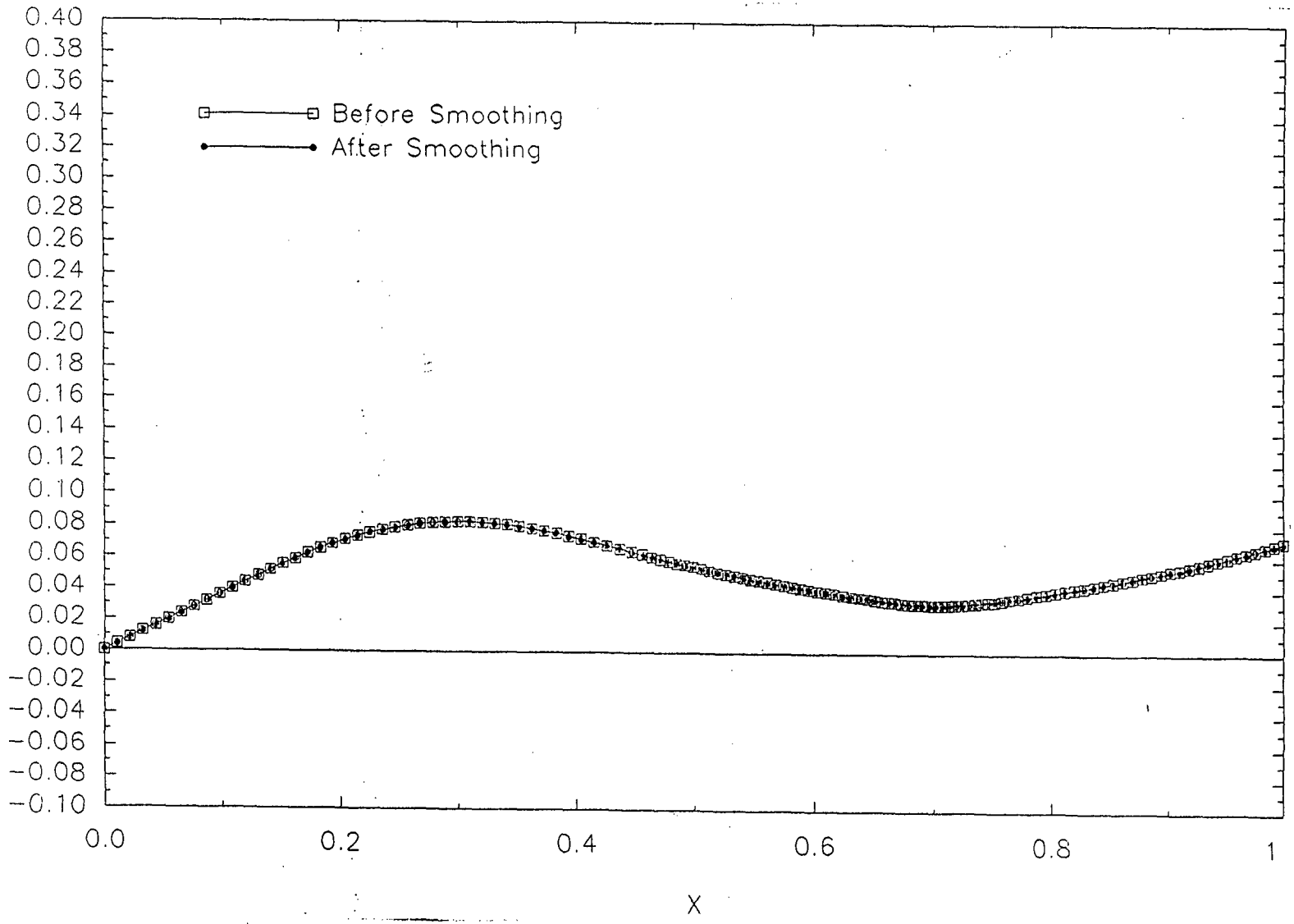



Figure 5.1 B–spline control polygon for the waisted body geometry before and after smoothing

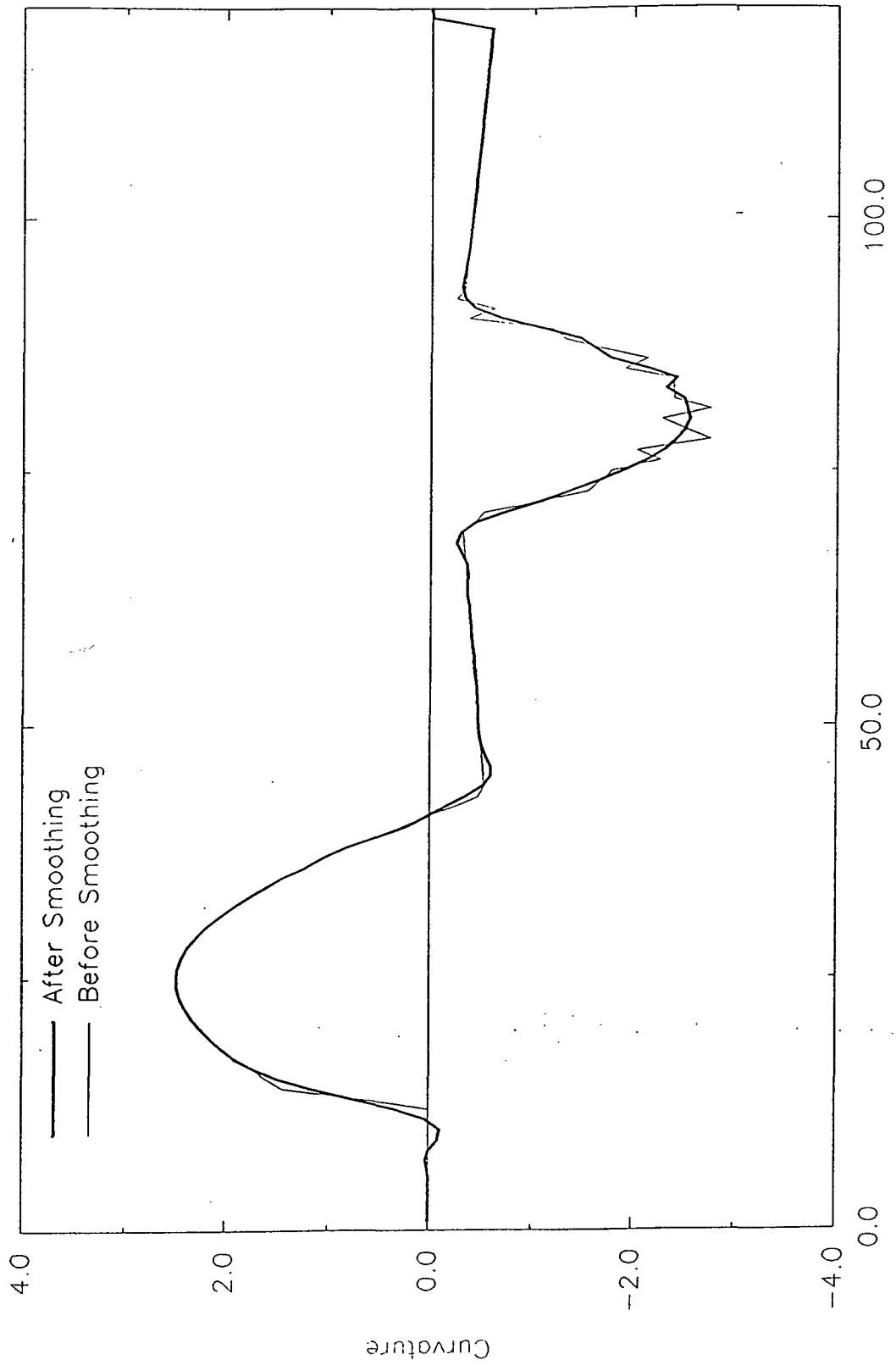



**Figure 5.2 Curvature plots before and after fairing for the waisted body**

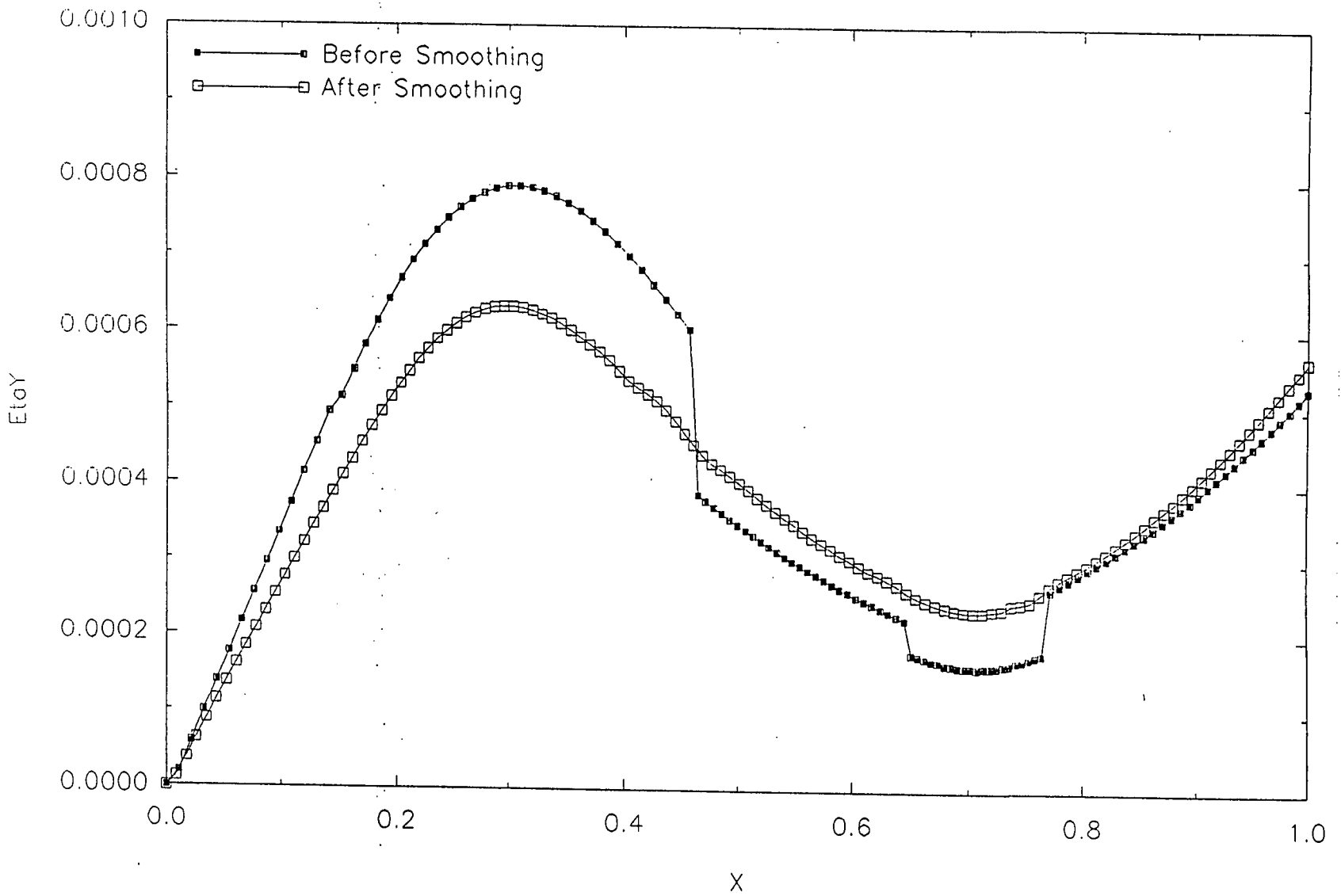



Figure 5.3 Effects of smoothing on the metric coefficient $\eta_y$ for the waisted body

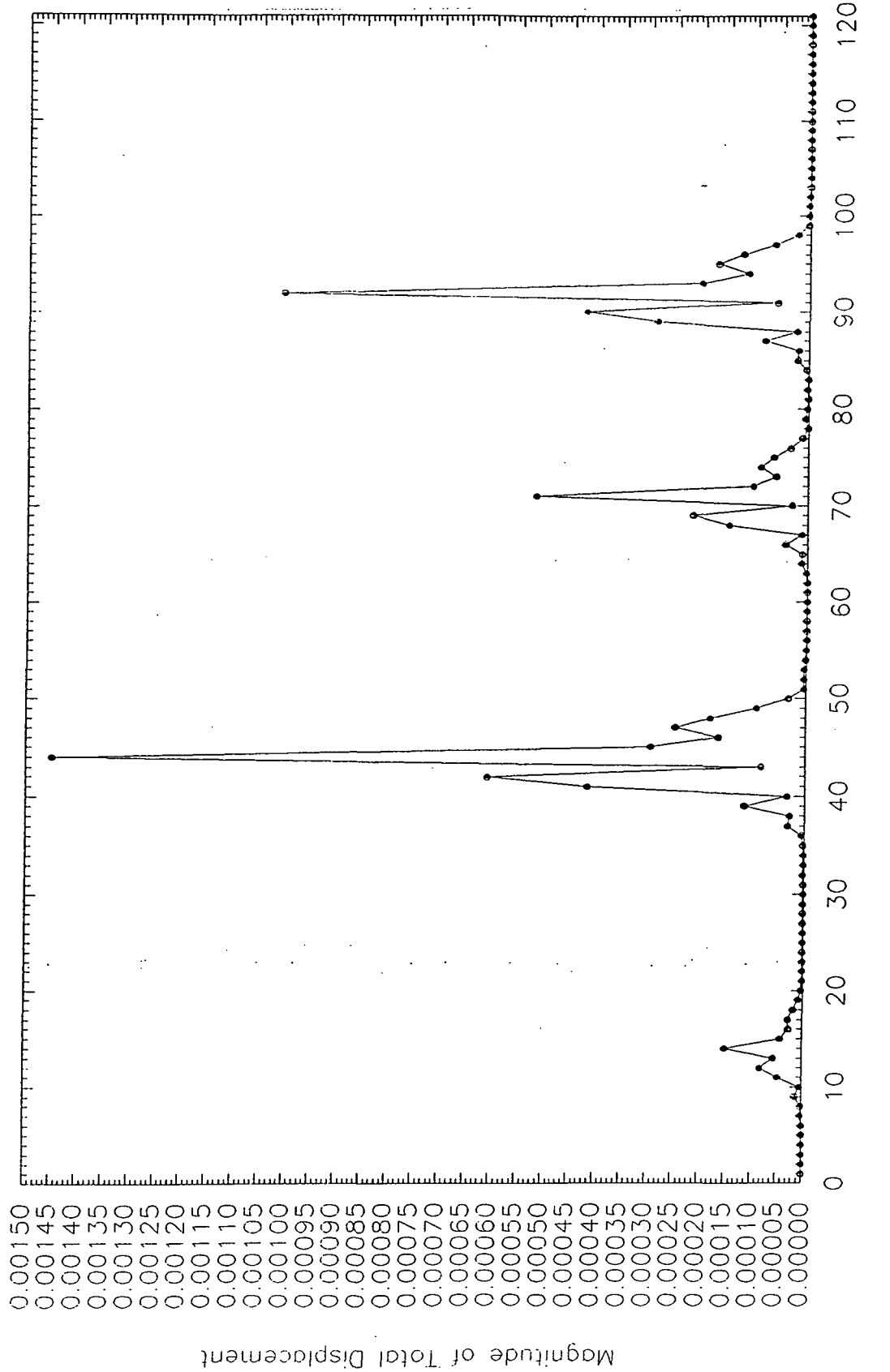



**Figure 5.4 Magnitude of total point displacement on the waisted body due to smoothing**

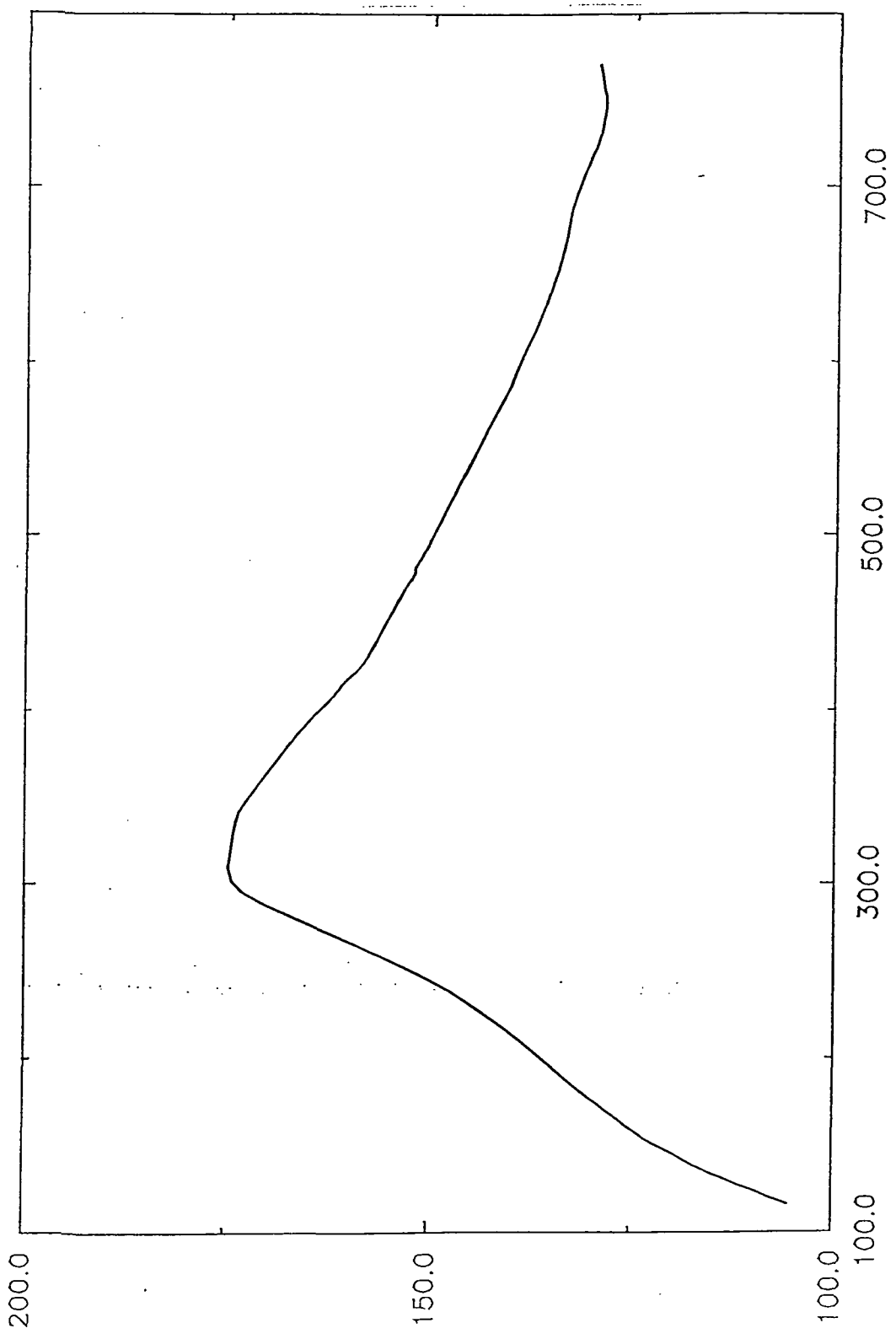



Figure 5.5 Original digitized curve data

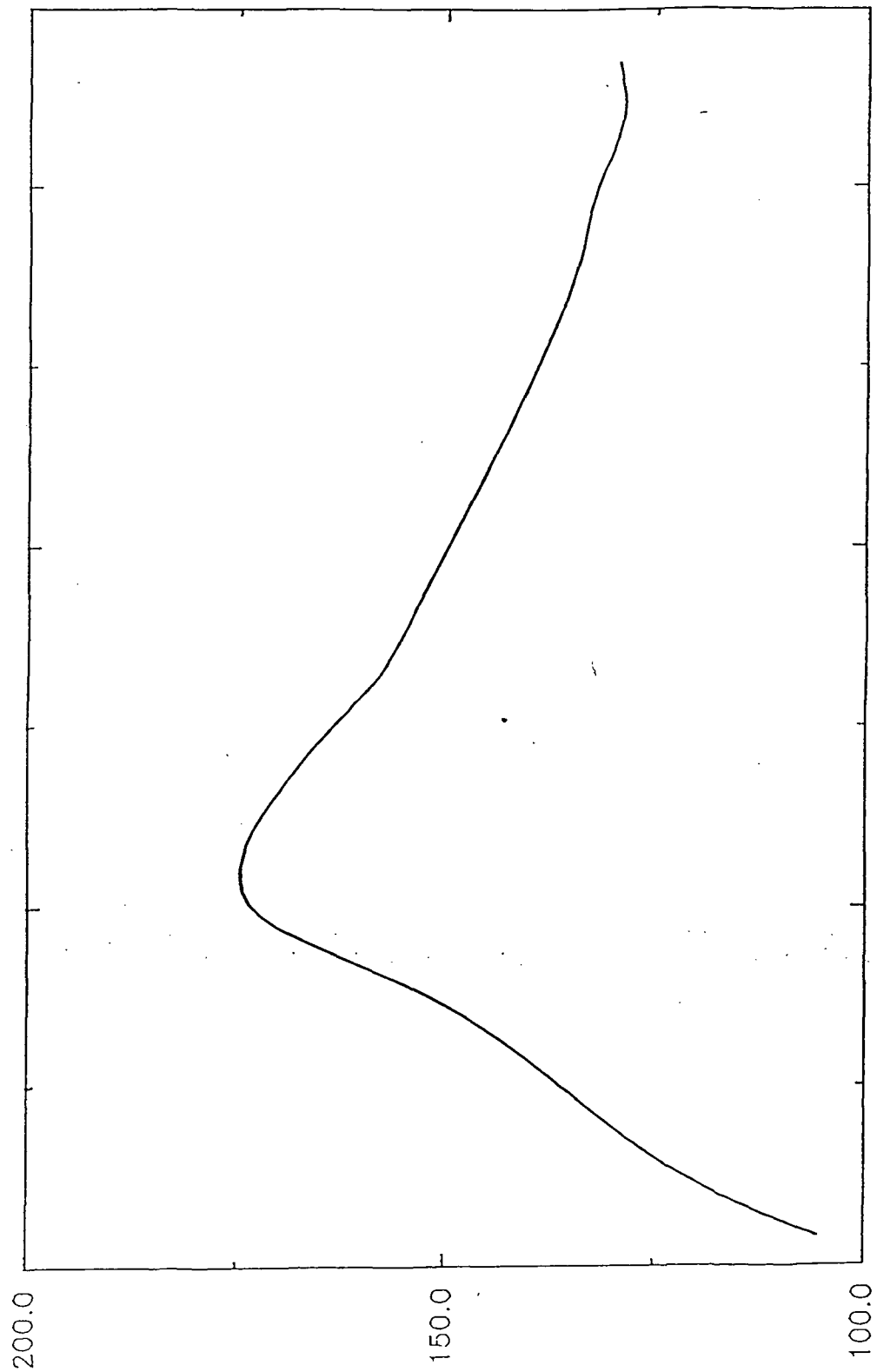



Figure 5.6 Results of smoothing with interrogation and point picking

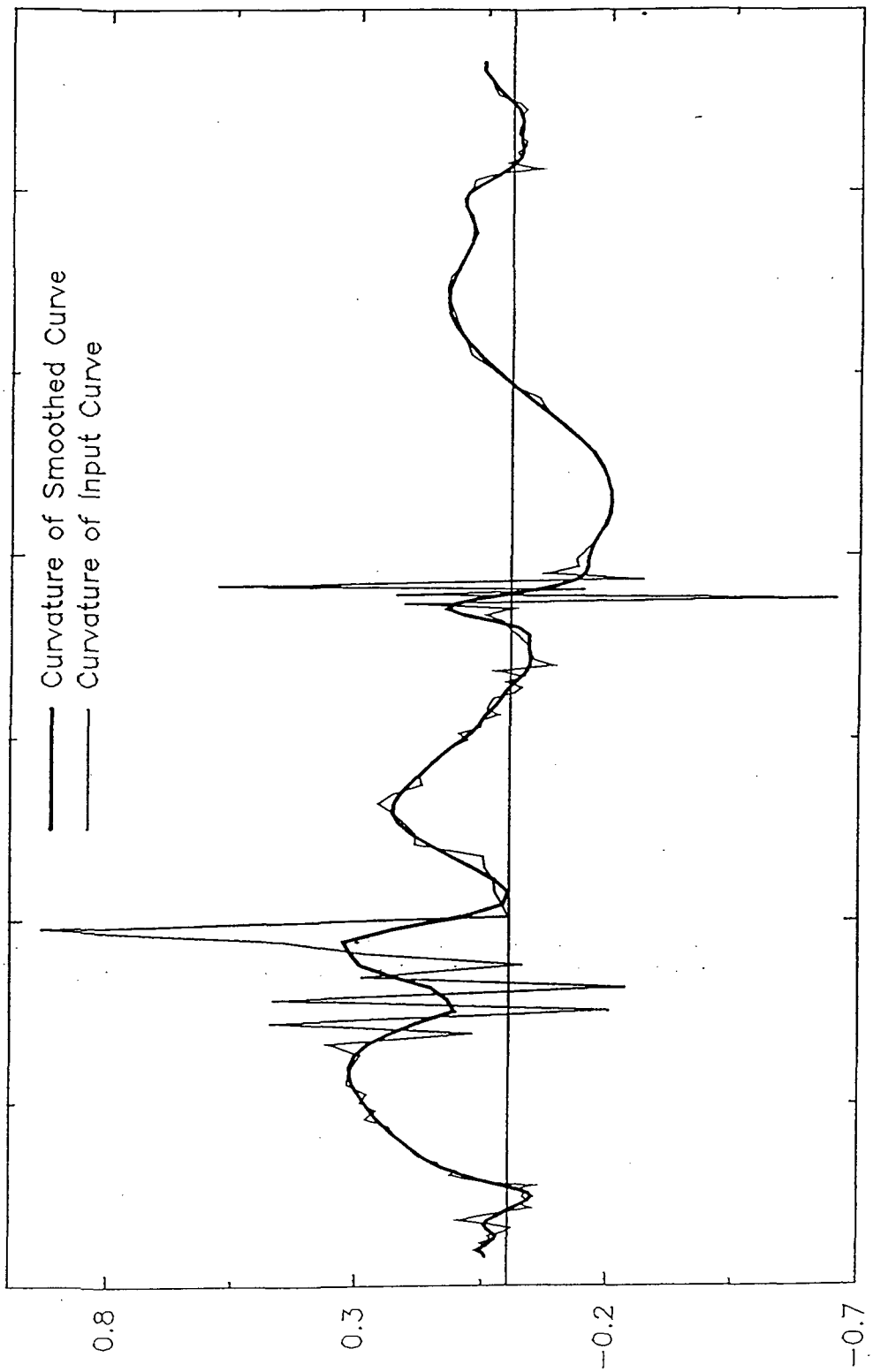



Figure 5.7 Curvature plots for unsmoothed data and data smoothed using the interrogation algorithm

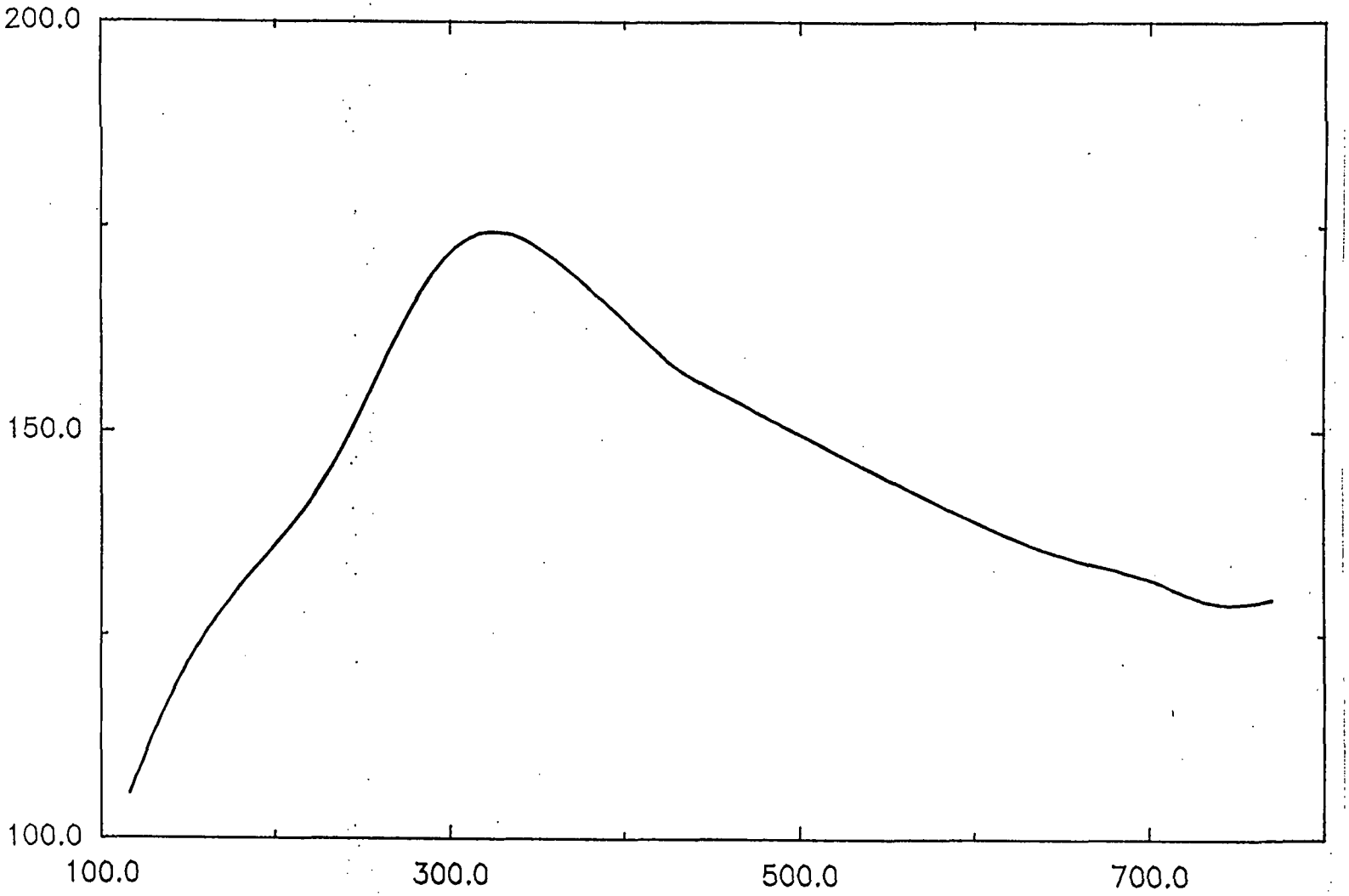



Figure 5.8 Results of smoothing without using interrogation and point picking

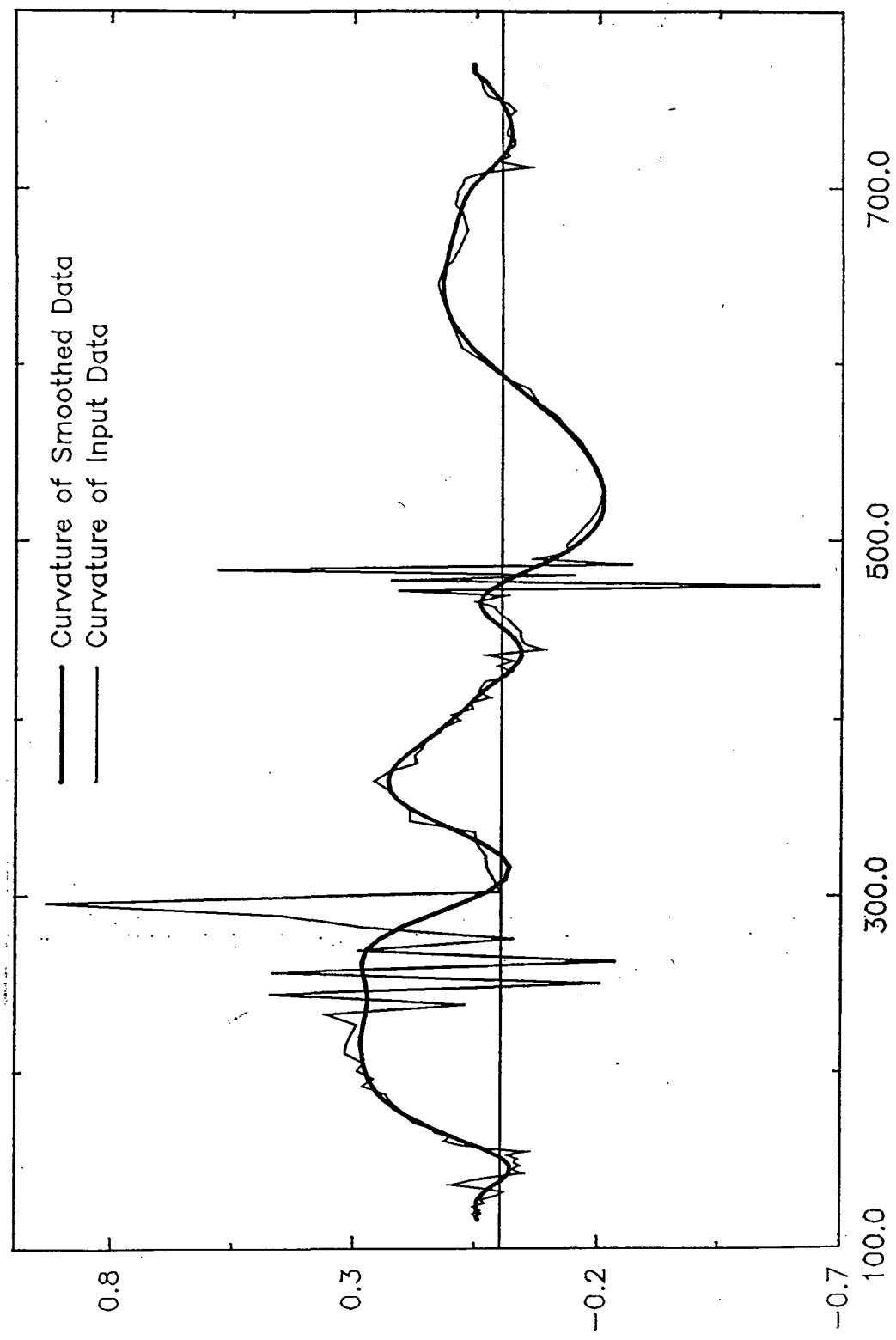



Figure 5.9 Curvature plots for original data and data smoothed without interrogation and picking

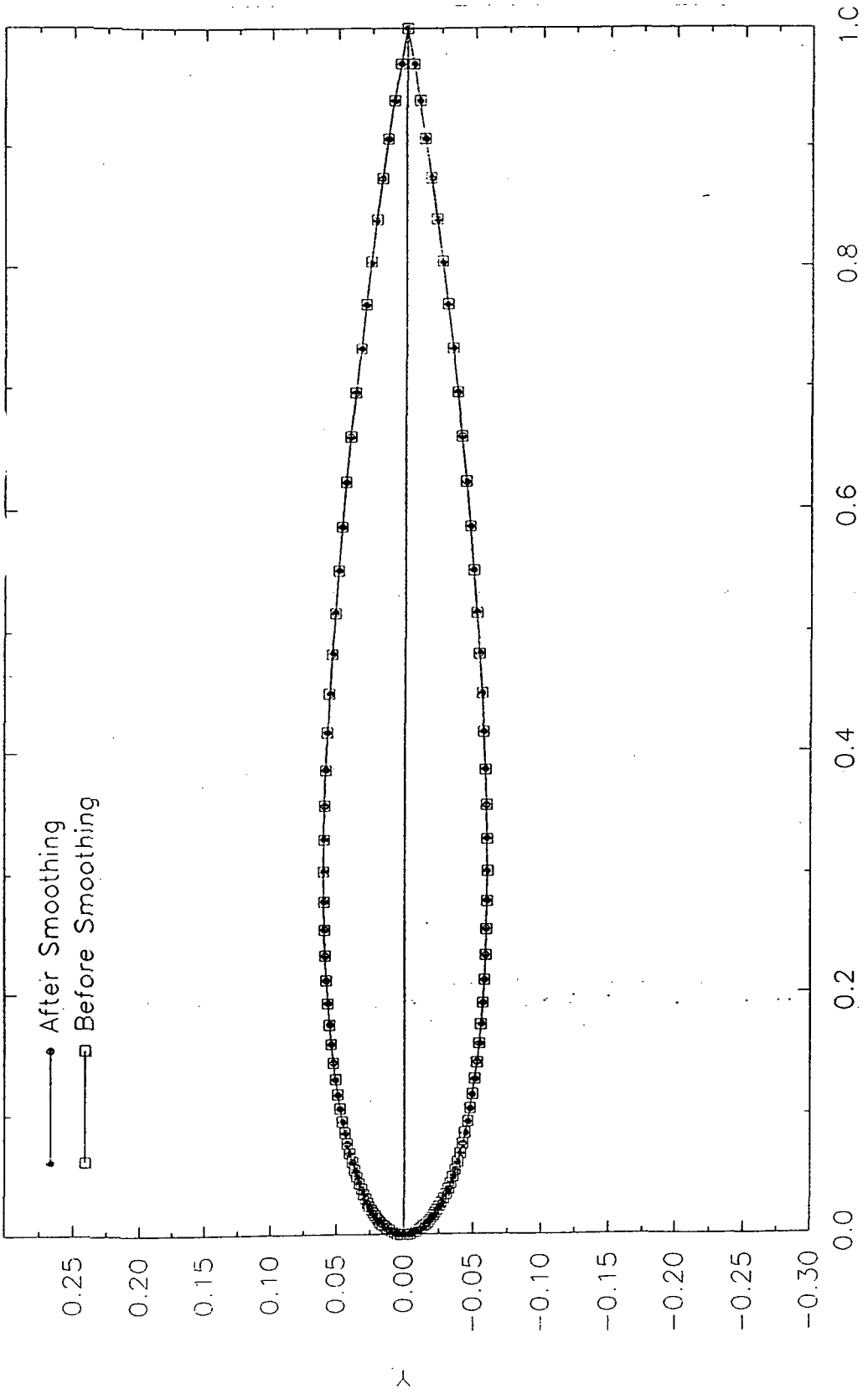



Figure 5.10 B–spline control polygon for original and smoothed NACA 0012 airfoil

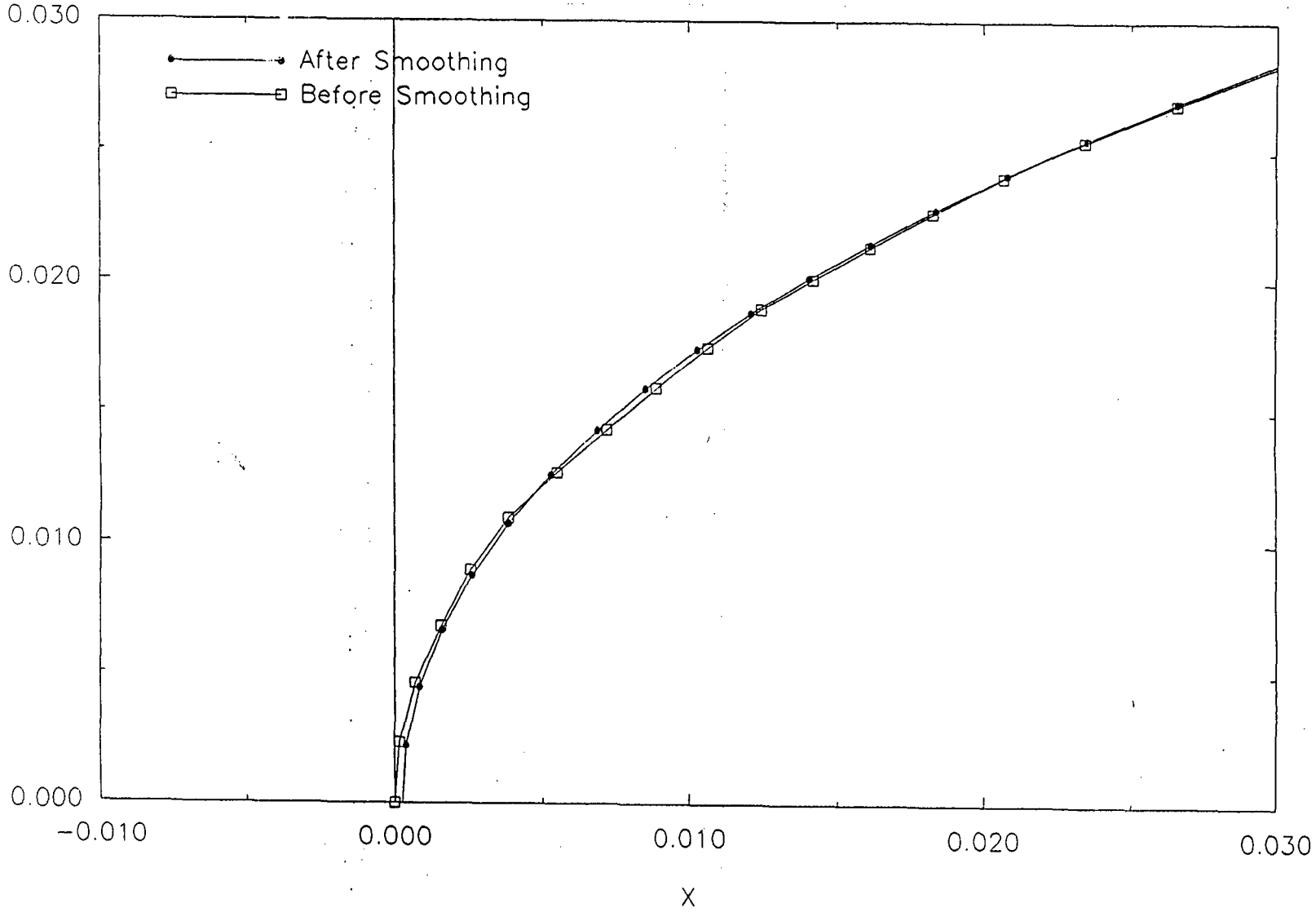



**Figure 5.11 Closeup of leading edge of the NACA 0012 airfoil showing effects of smoothing on the shape of the curve**

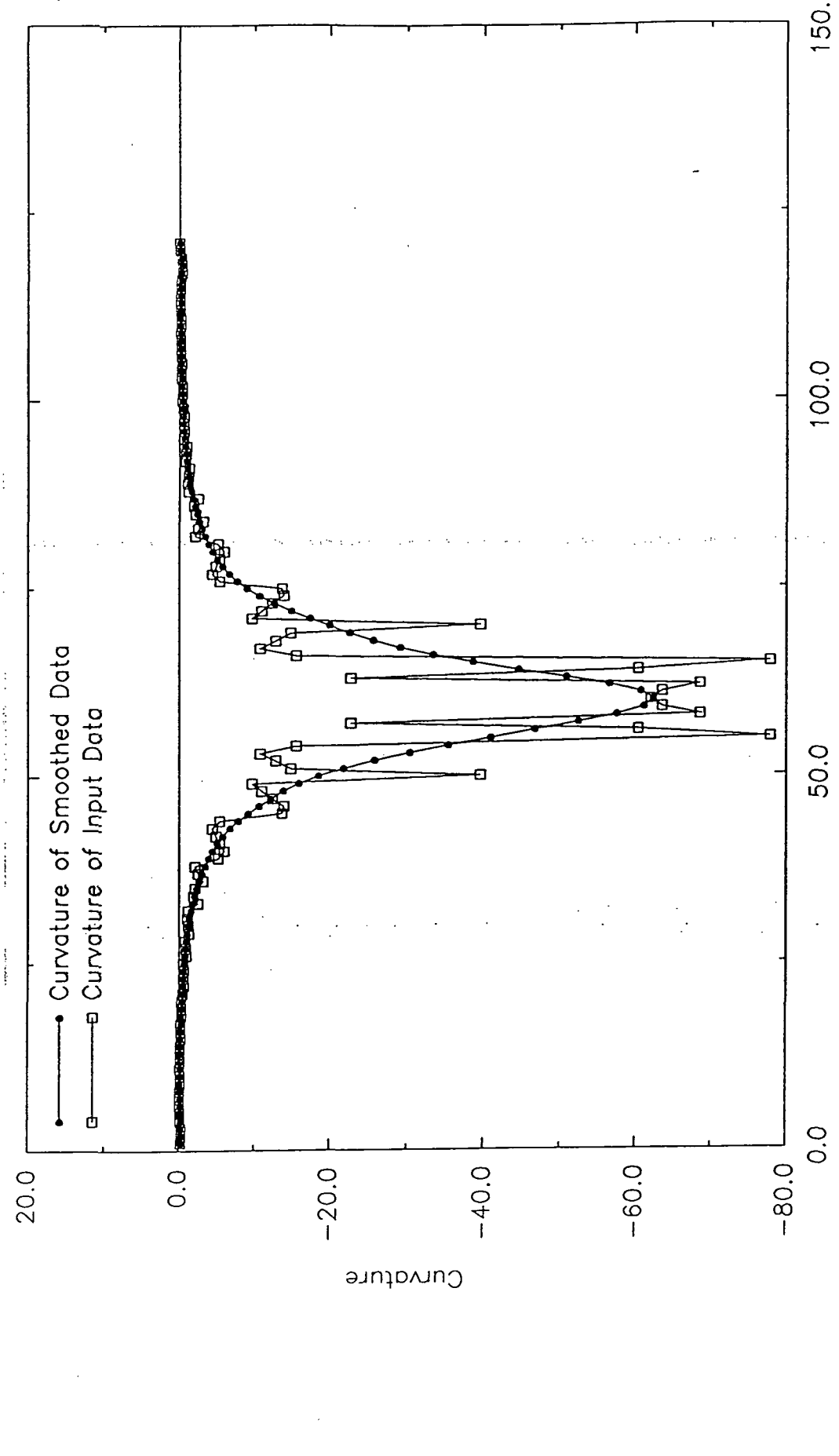



Figure 5.12 Curvature plots for the original data and smoothed data from the NACA 0012 wing section

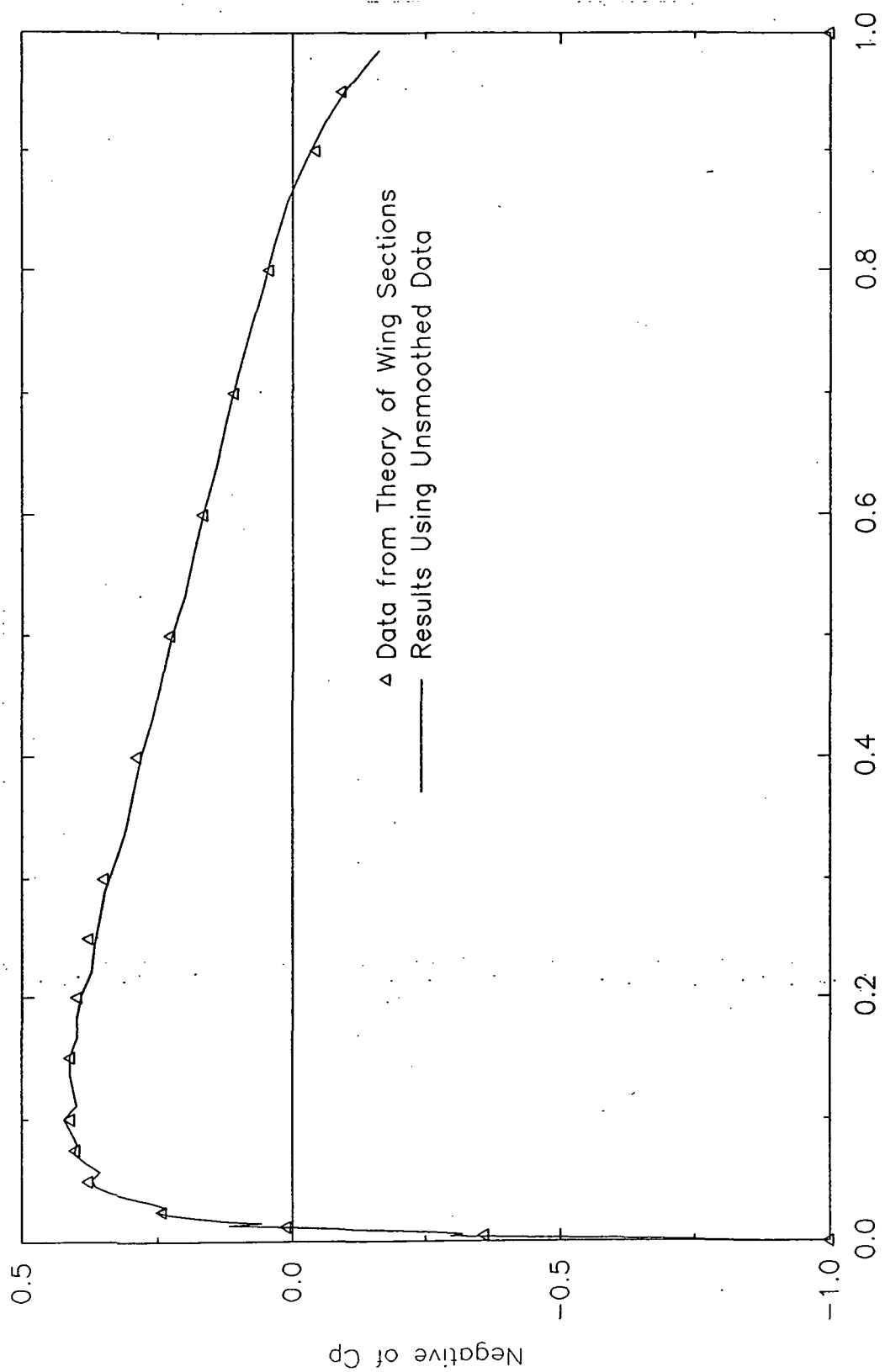



**Figure 5.13 Cp vs. X/C plot for original NACA 0012 data**

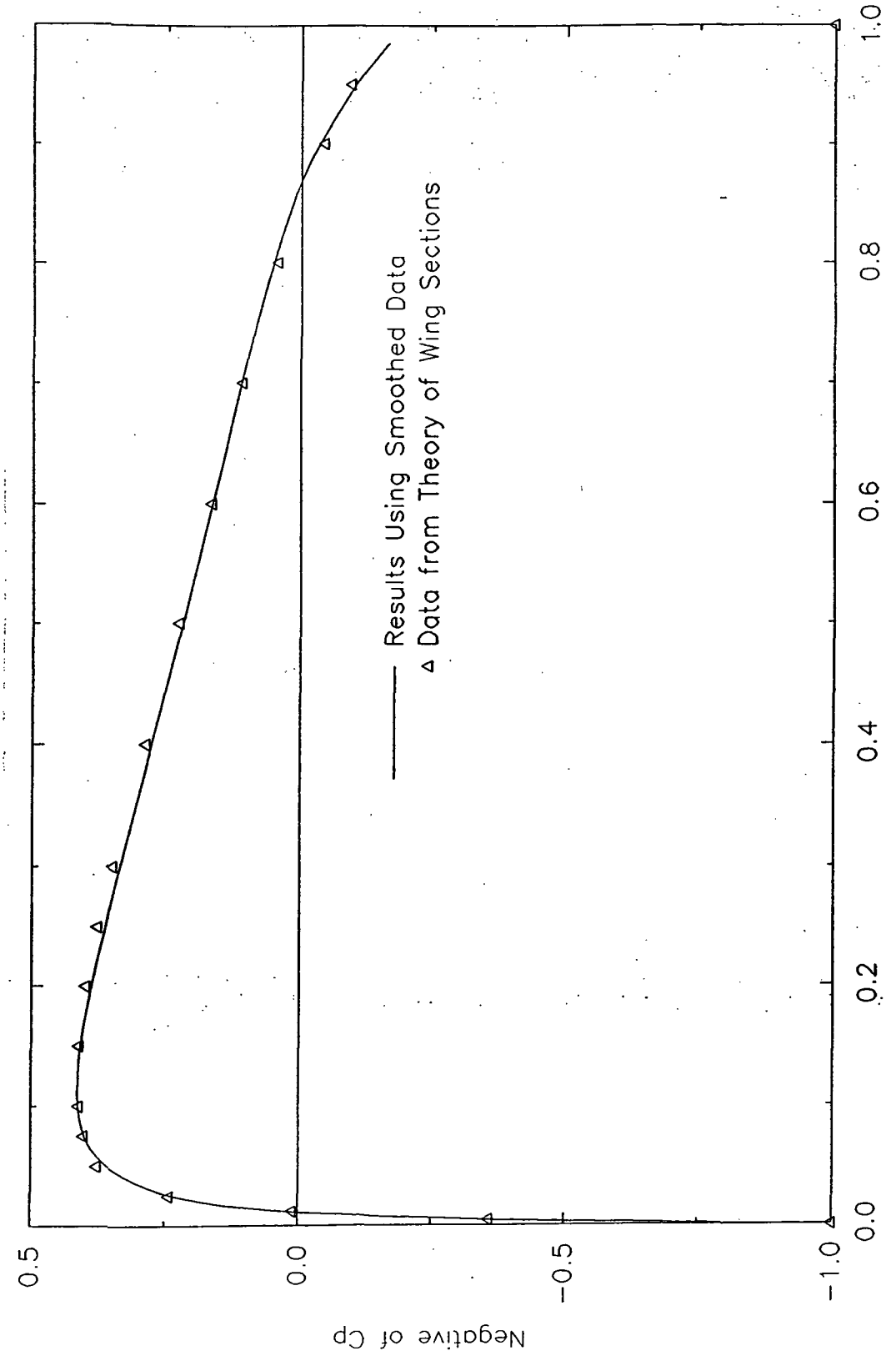



**Figure 5.14 Cp vs. X/C plot for the smoothed NACA 0012 data**

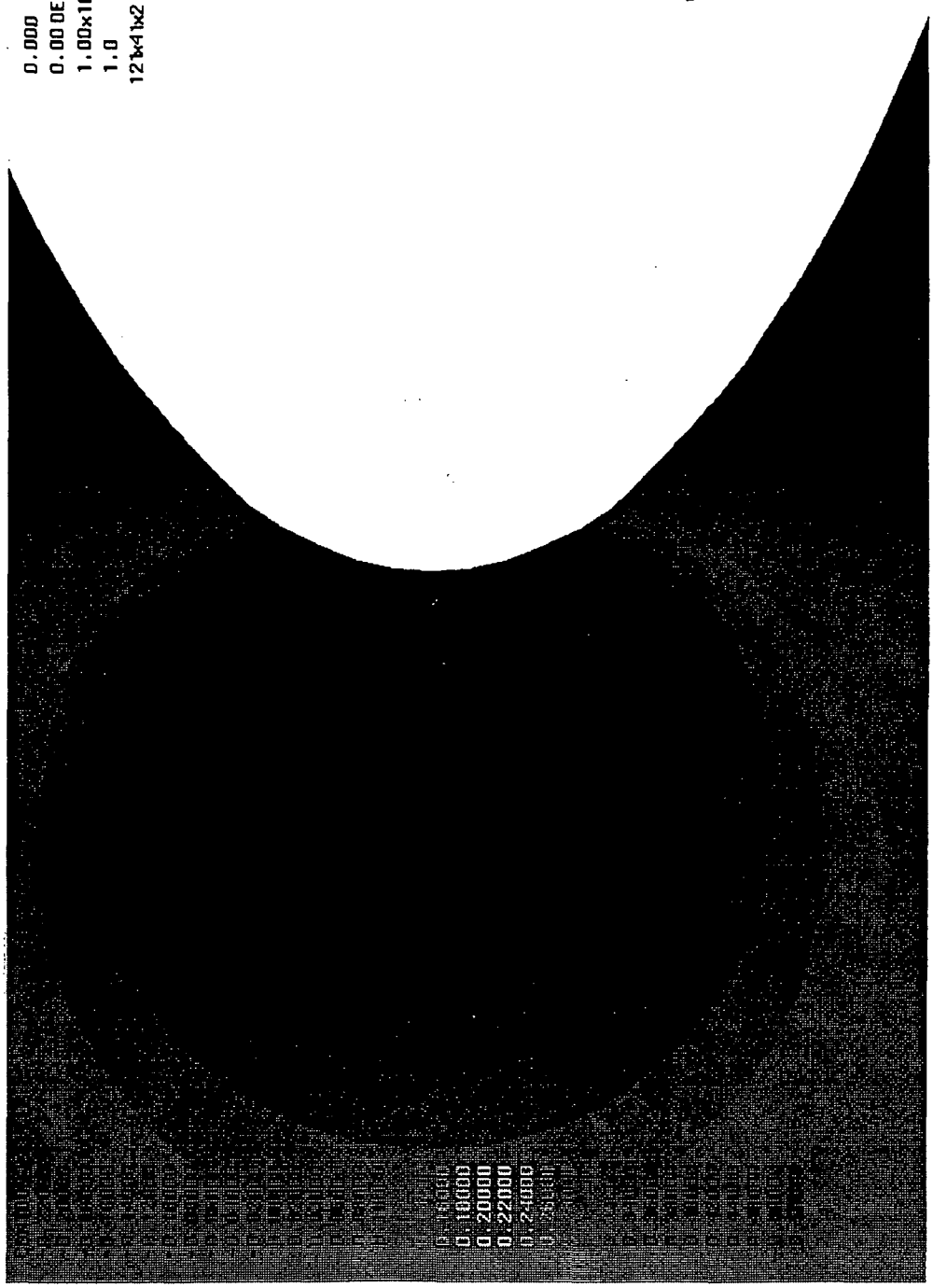



Figure 5.15 Pressure field near the leading edge of the original NACA 0012 airfoil at zero degrees A.O.A.

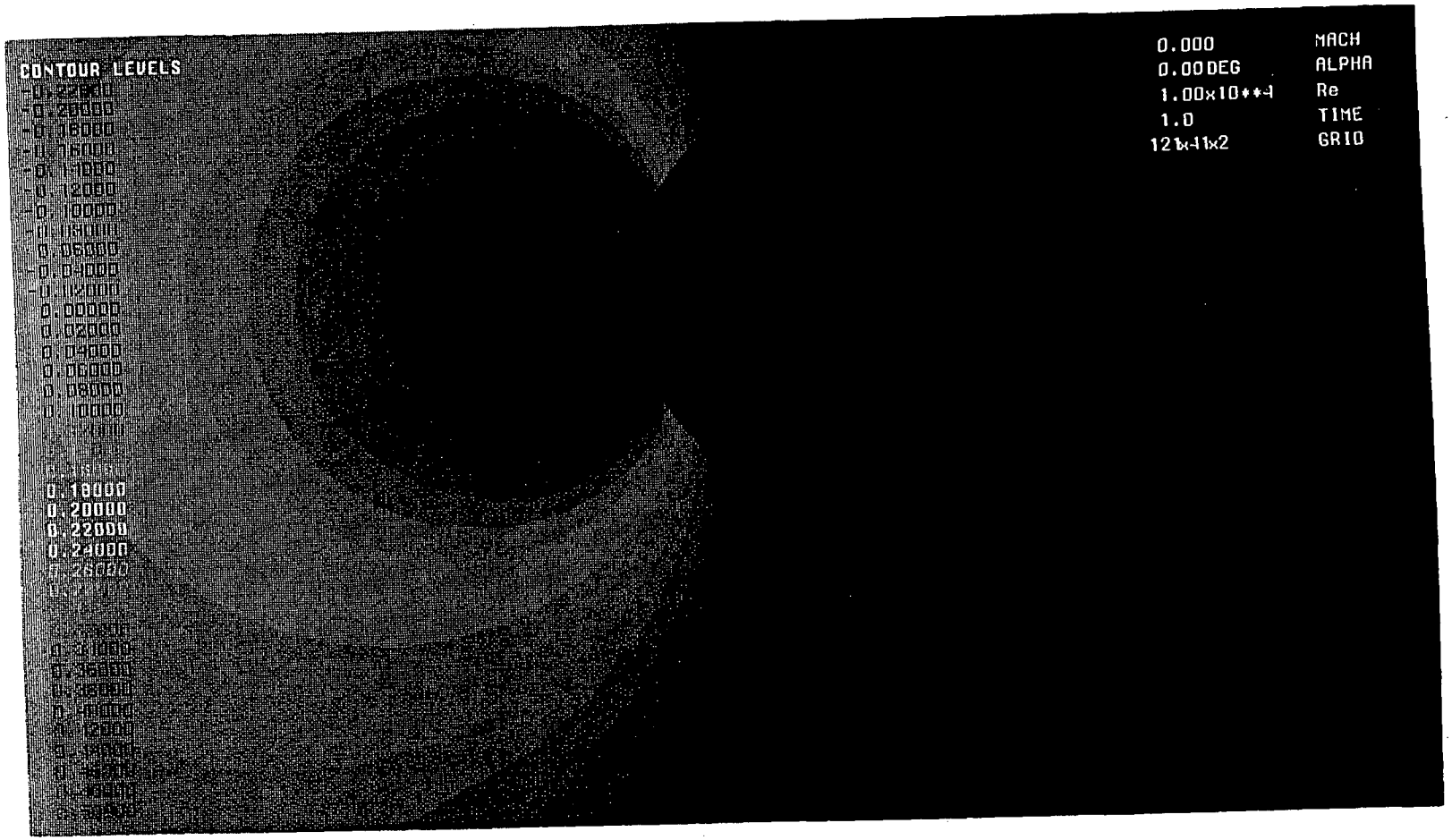



Figure 5.16 Pressure field near the leading edge of the smoothed NACA 0012 airfoil at zero degrees A.O.A.

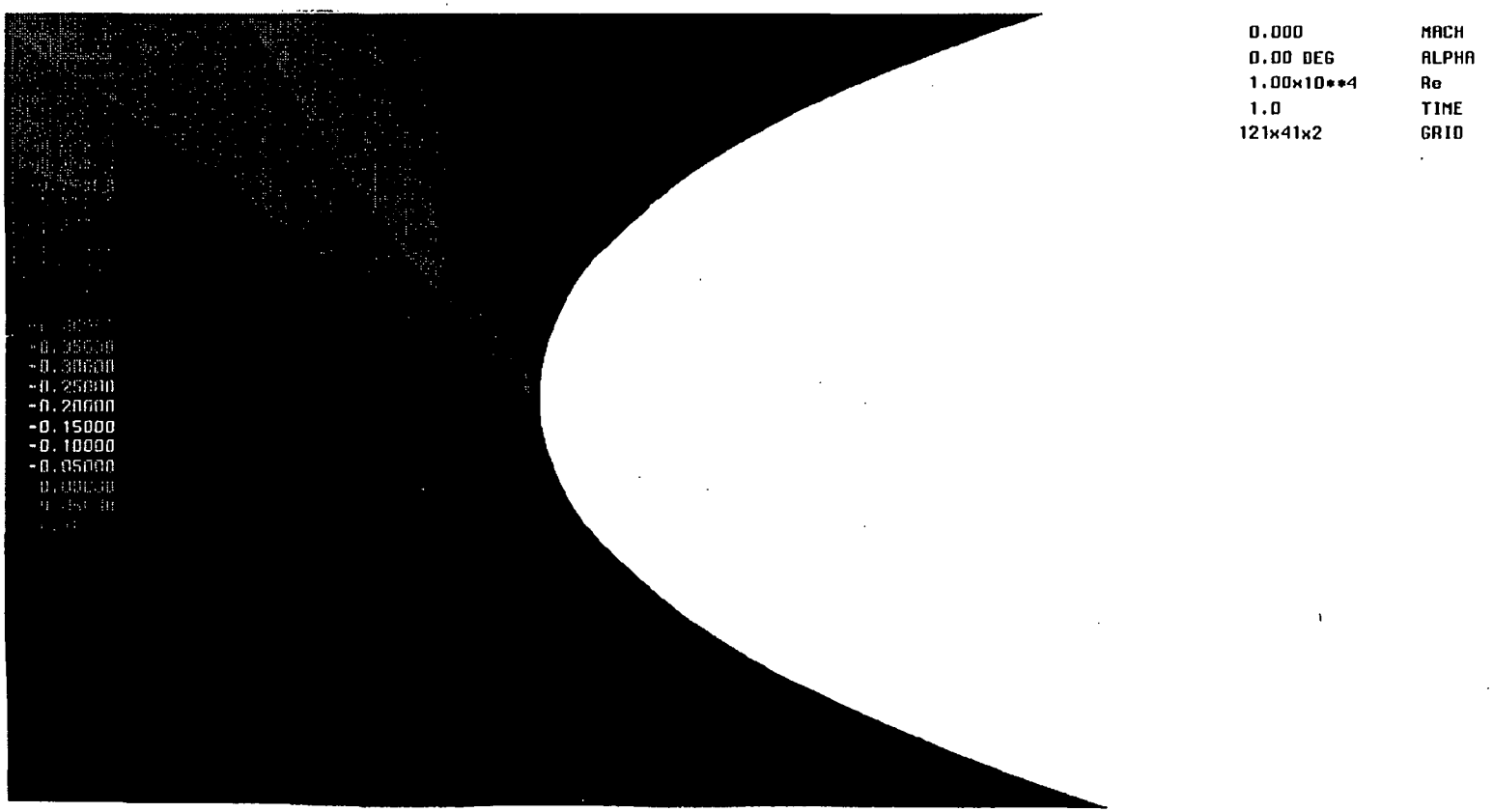



Figure 5.17 Pressure field near the leading edge of the original NACA 0012 airfoil at five degrees A.O.A.

Figure 5.18 Pressure field near the leading edge of the smoothed NACA 0012 airfoil at five degrees A.O.A.

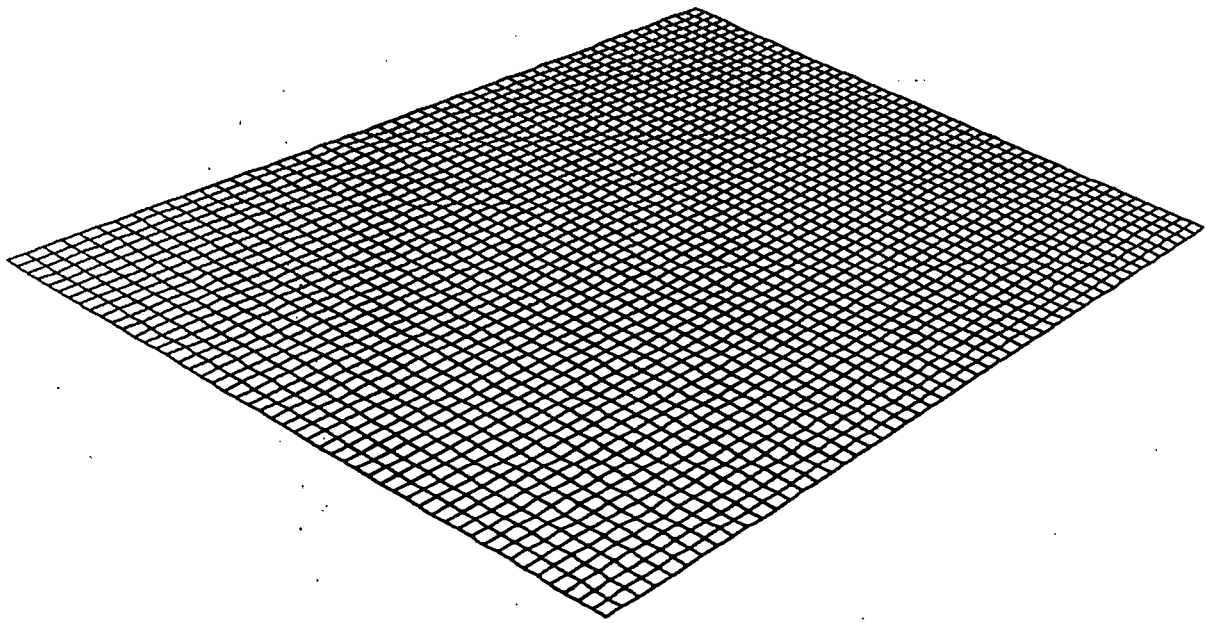


Figure 5.19 Planar surface with geometric irregularities

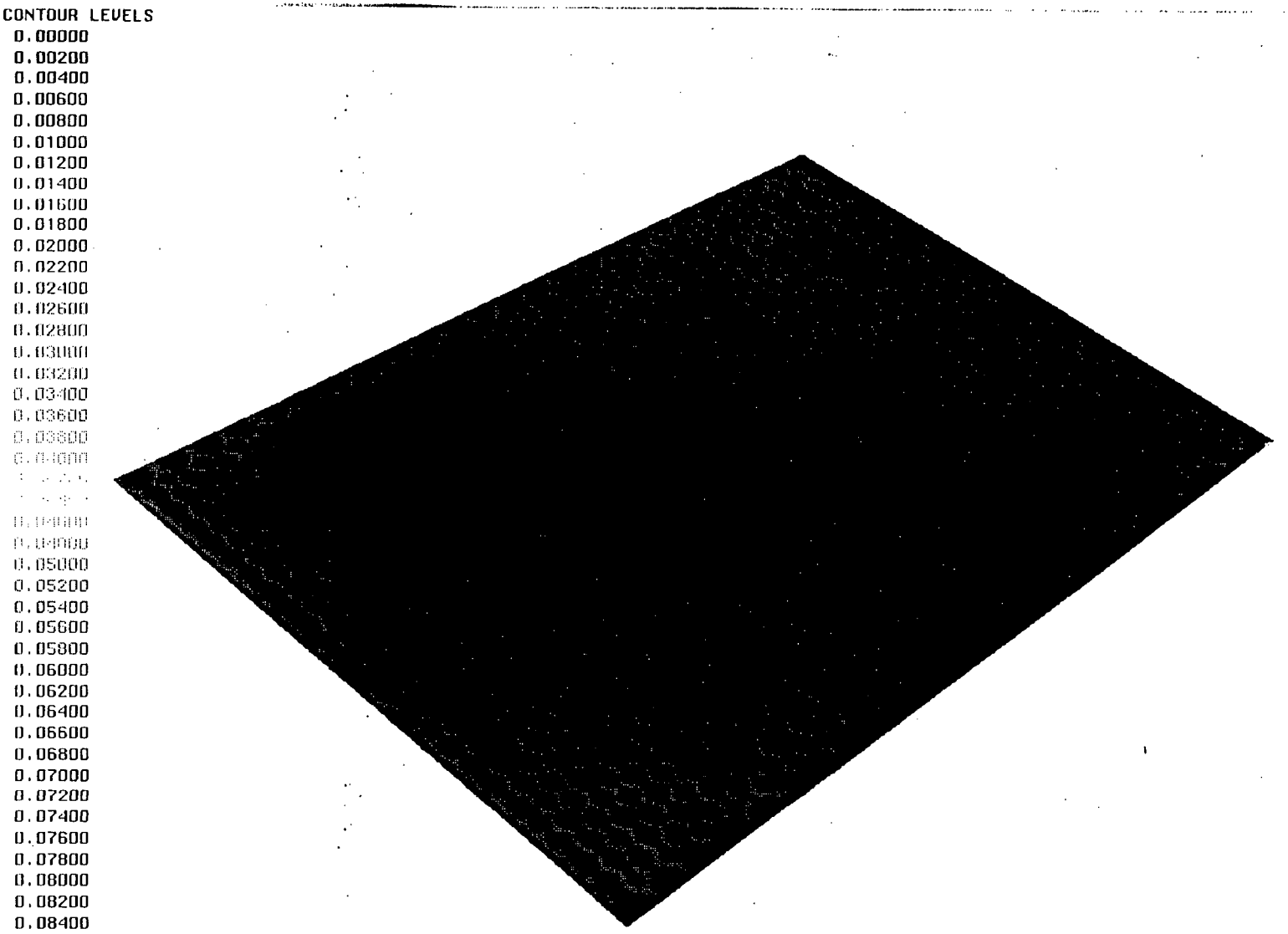



Figure 5.20 Color contours of absolute curvature on the perturbed planar surface

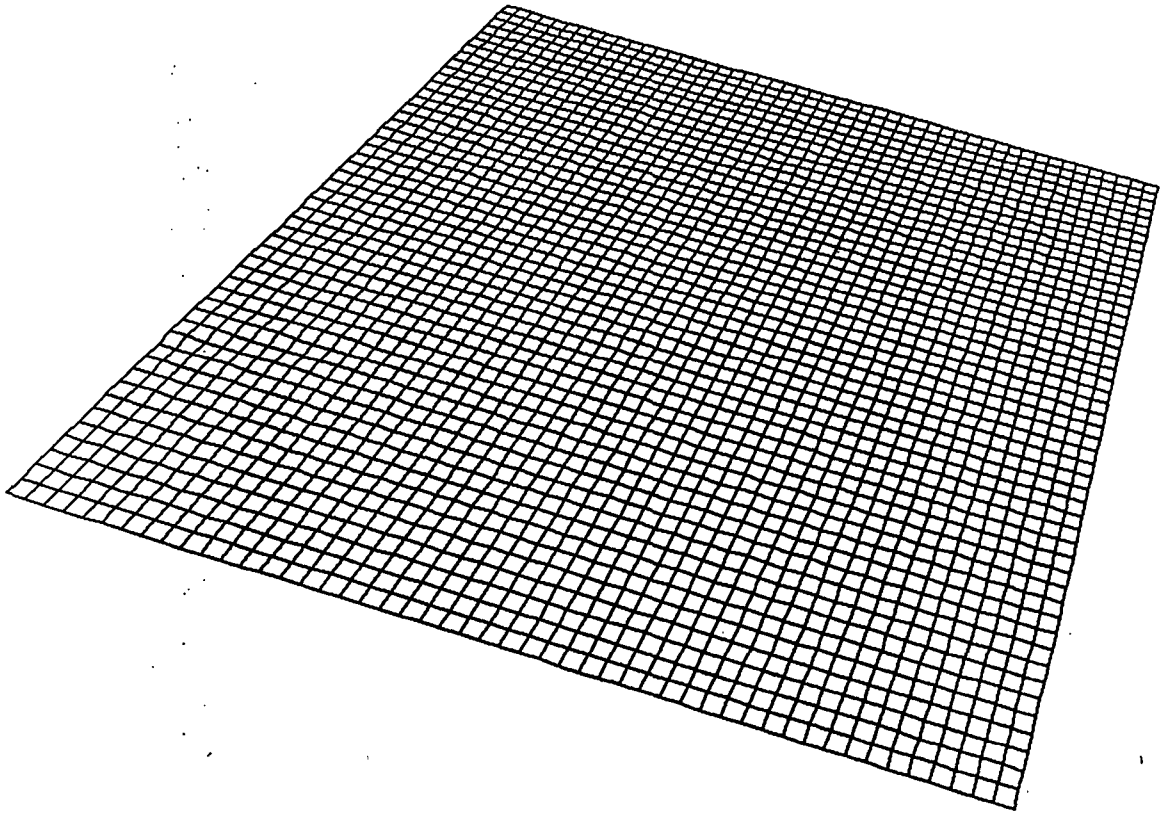


Figure 5.21 Planar surface after smoothing

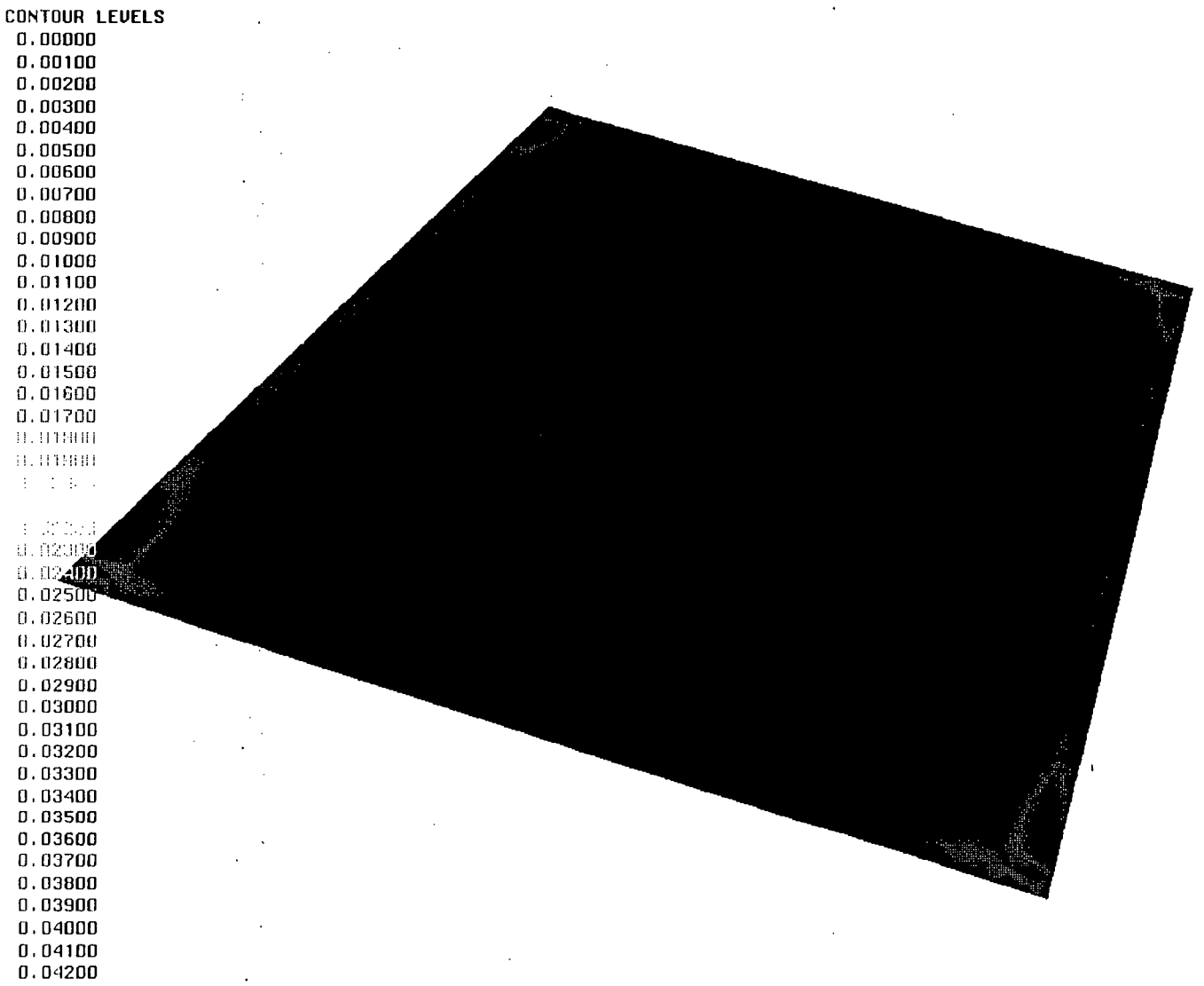



Figure 5.22 Color contours of absolute curvature on the smoothed planar surface

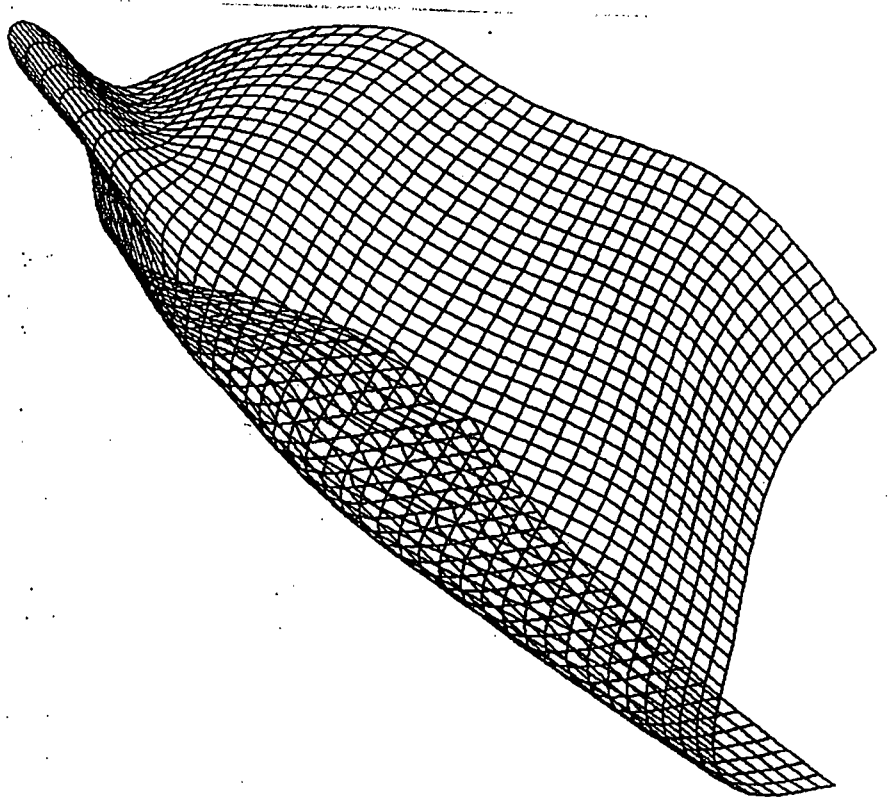


Figure 5.23 Original surface data for the lifting body surface patch

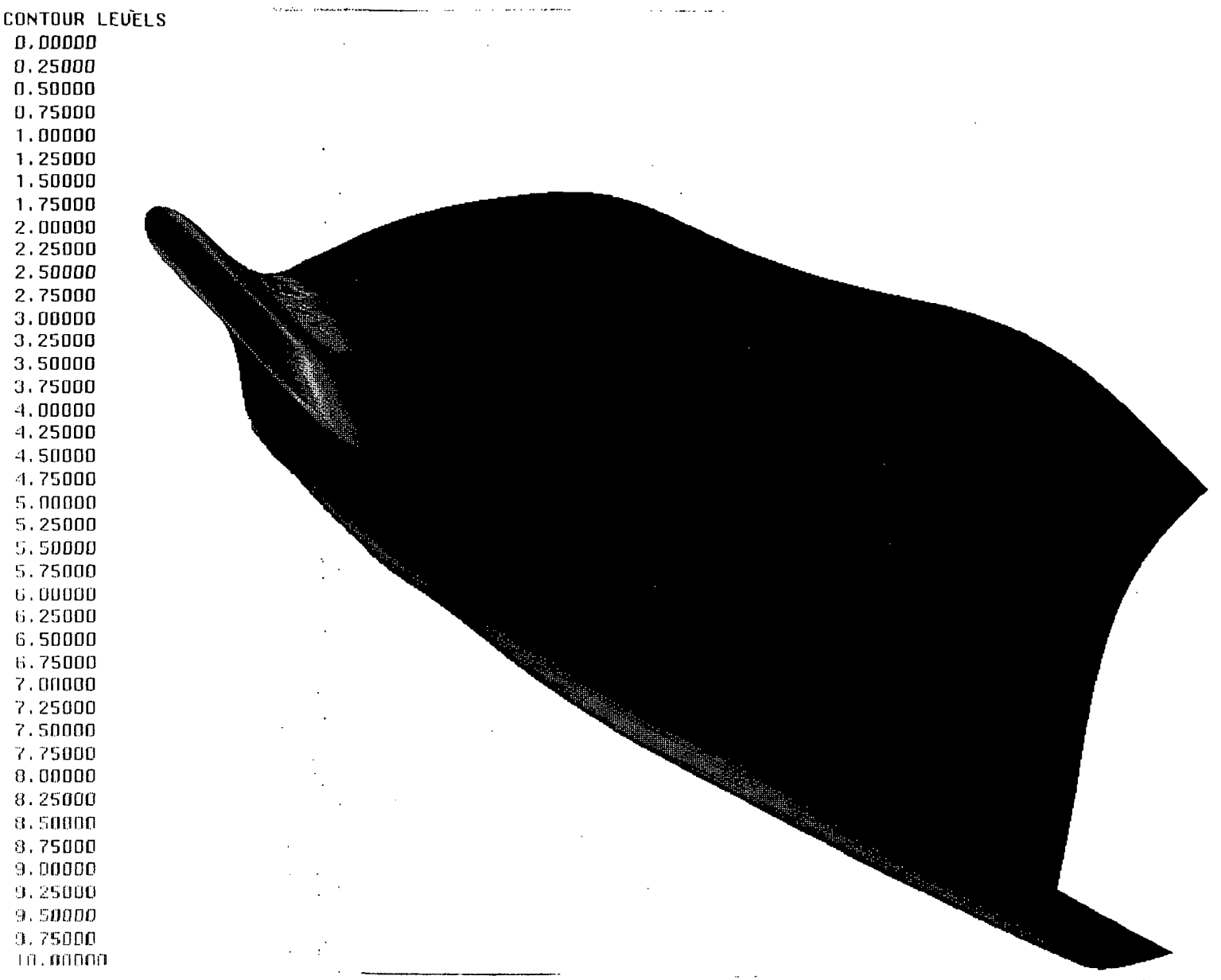



Figure 5.24 Absolute curvature on the original lifting body surface spline

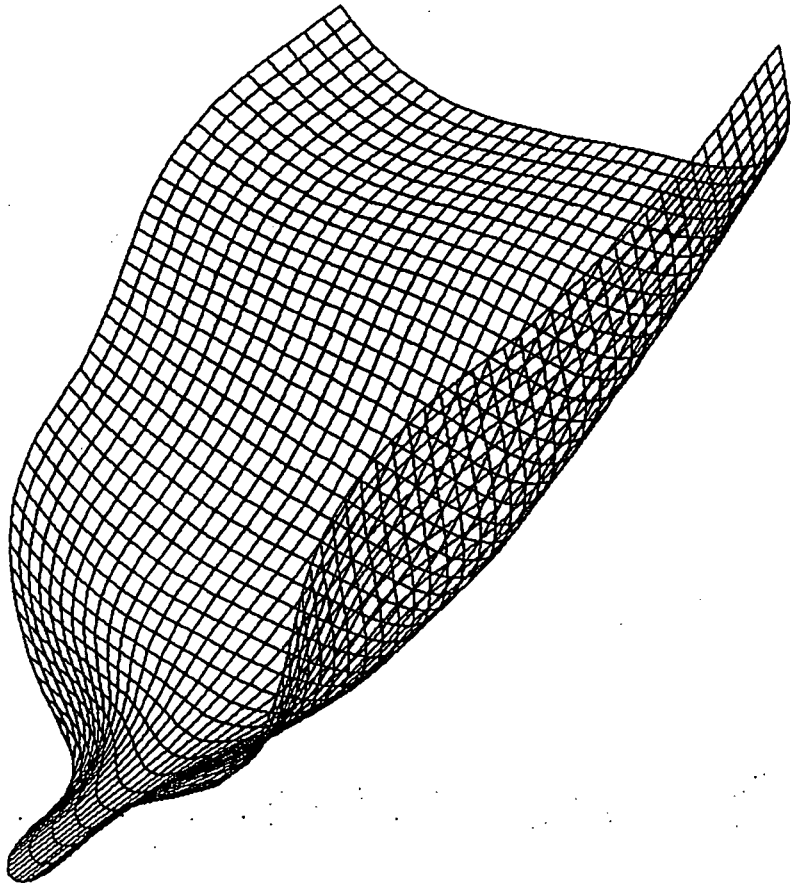


Figure 5.25 Smoothed data for the lifting body surface patch

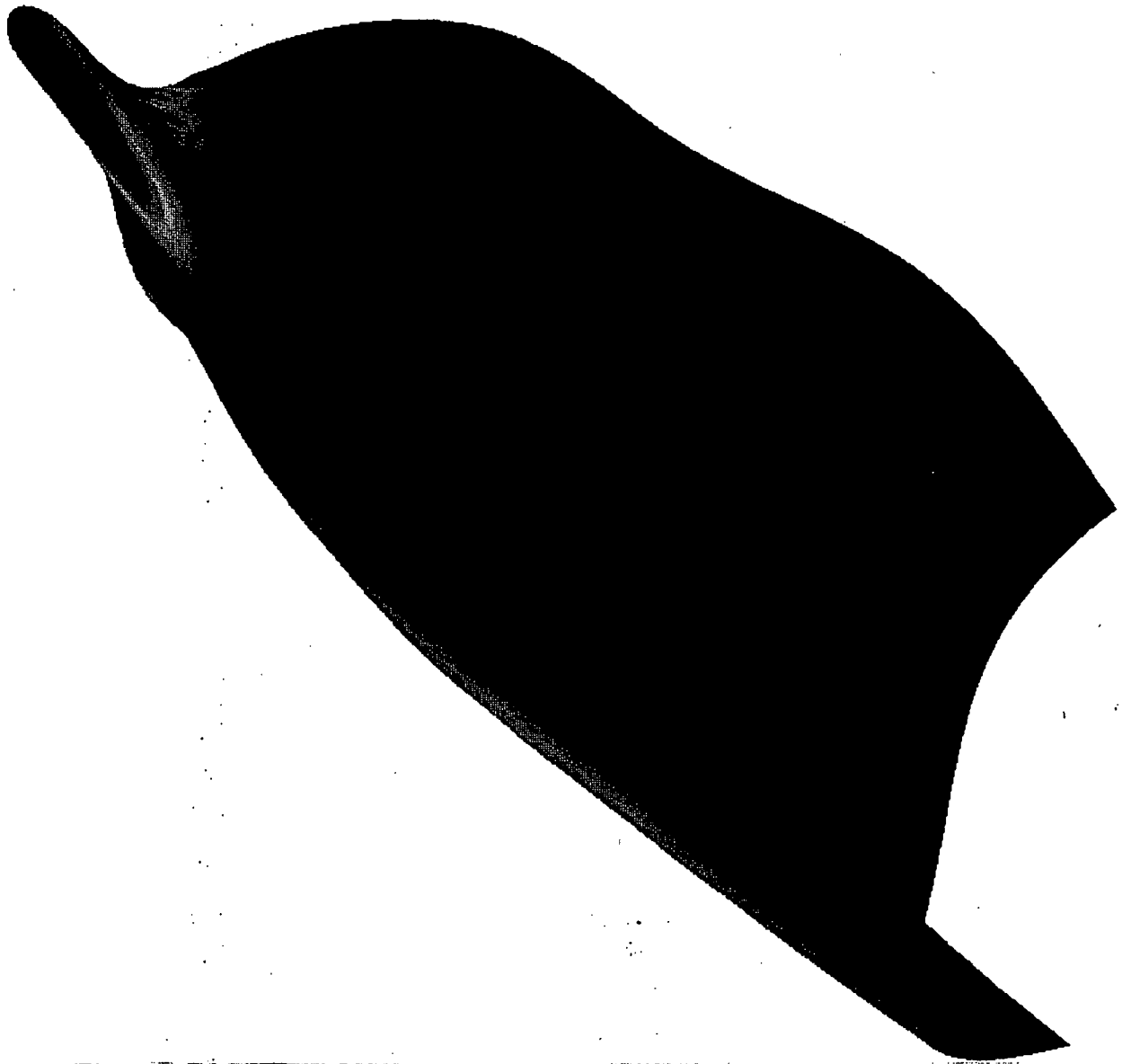


Figure 5.26 Absolute curvature on the smoothed lifting body surface spline

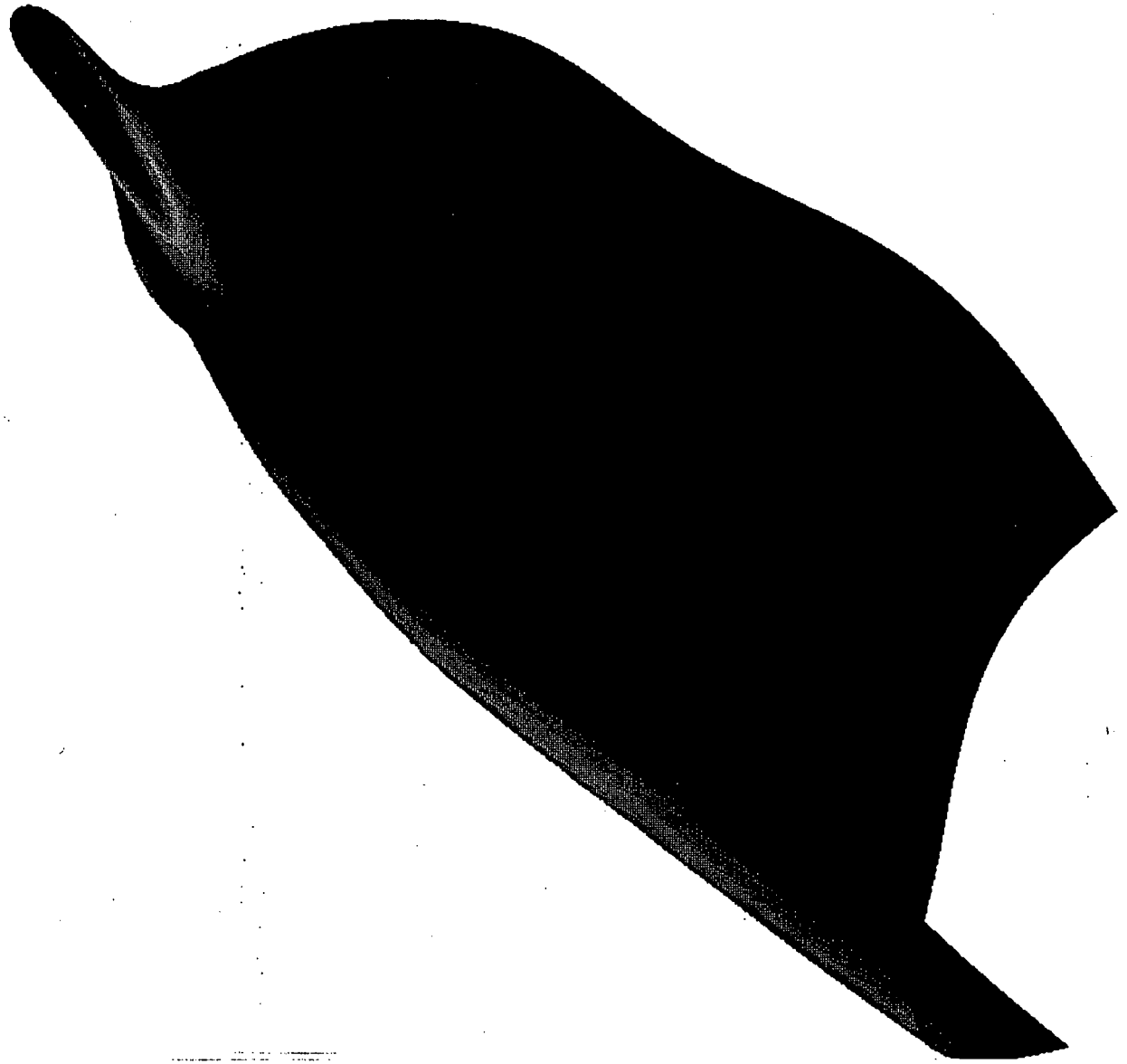


Figure 5.27 First principal curvature on the smoothed lifting body surface spline

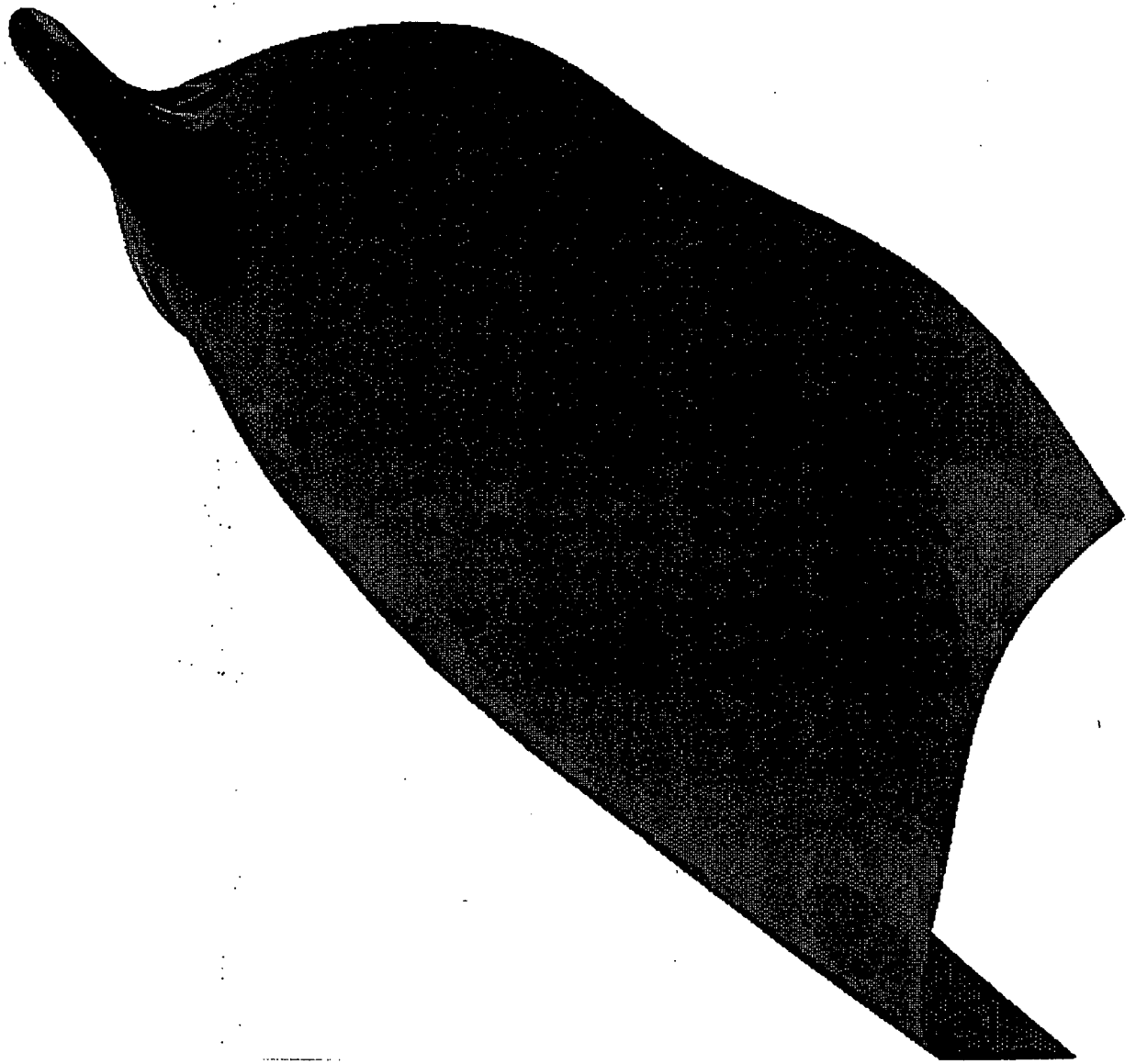


Figure 5.28 Gaussian curvature on the smoothed lifting body surface spline

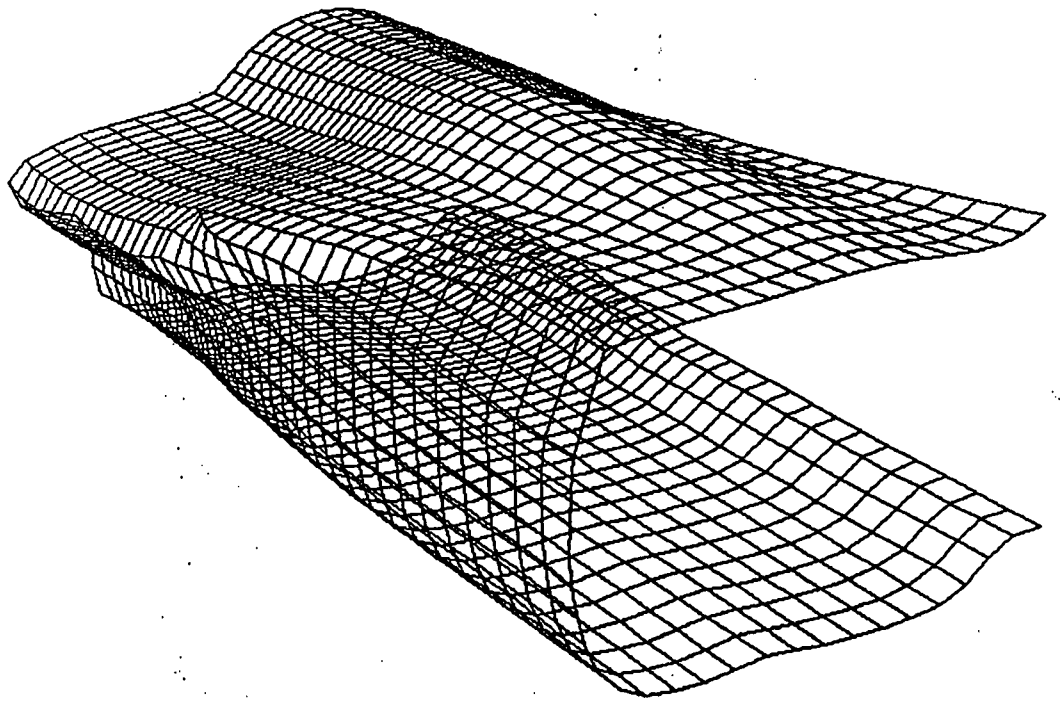


Figure 5.29 Original surface data from the digitized F–15 aft quarter

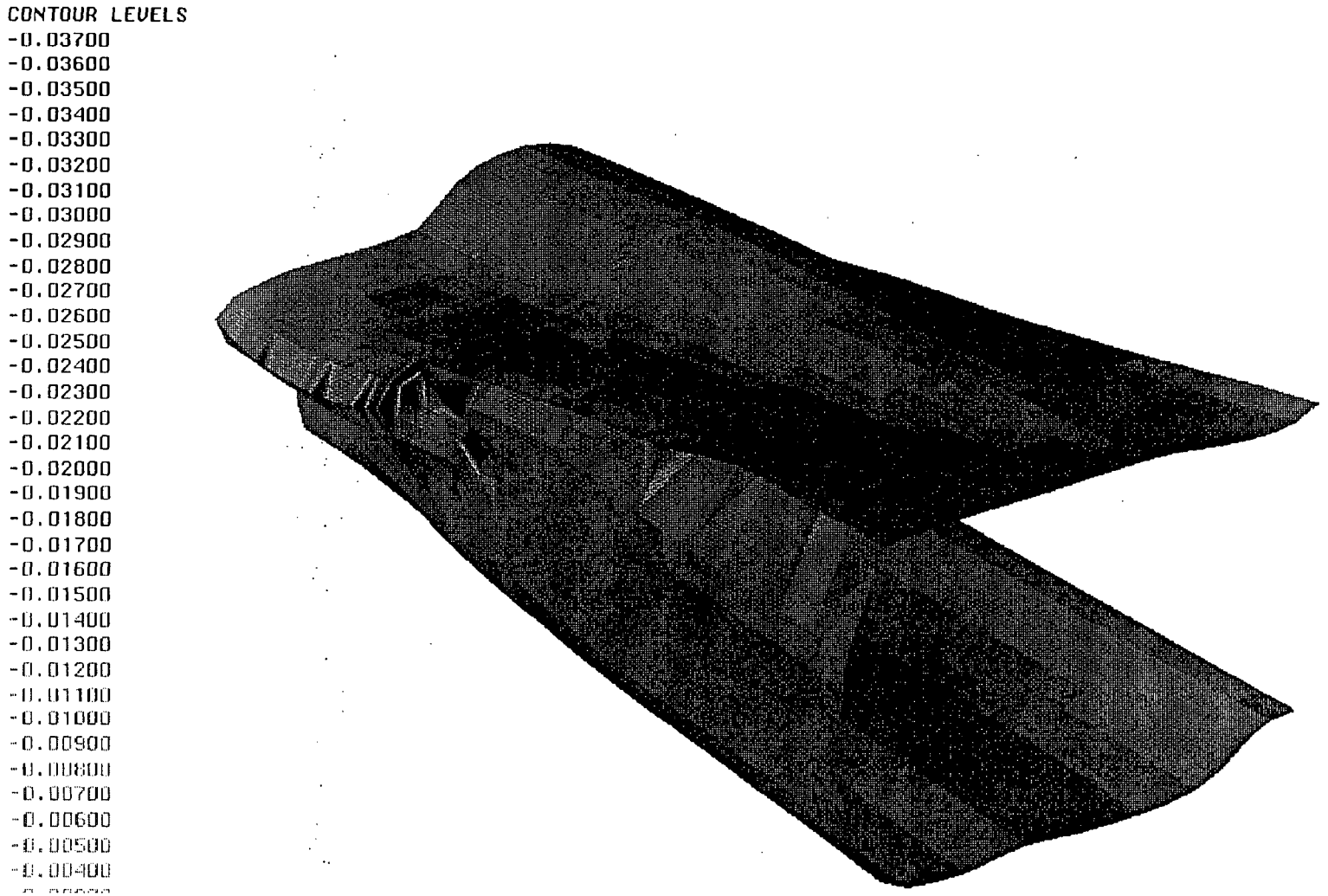



Figure 5.30 Gaussian curvature on the original F–15 aft quarter surface spline

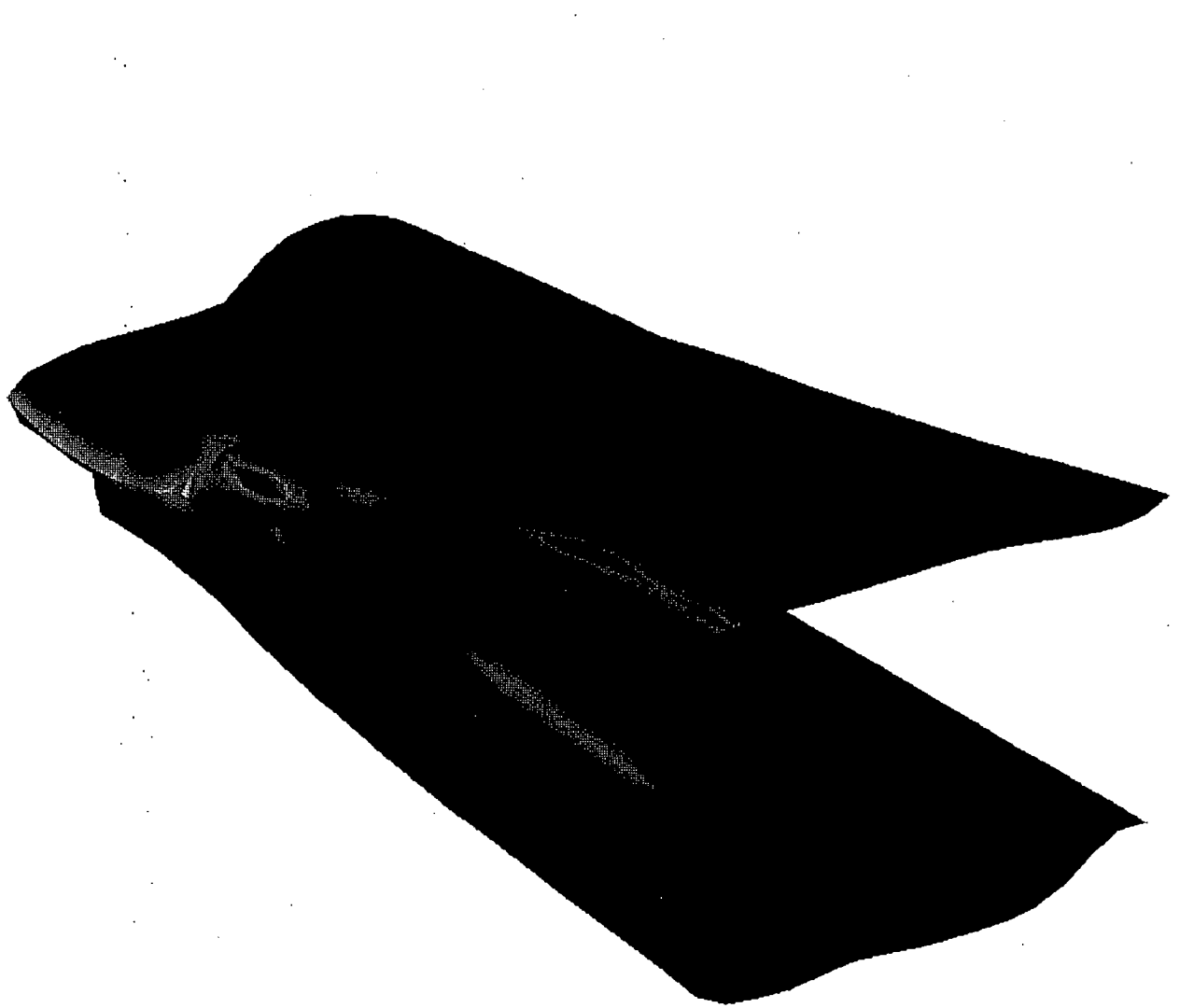



Figure 5.31 Absolute curvature on the original F–15 aft quarter surface spline

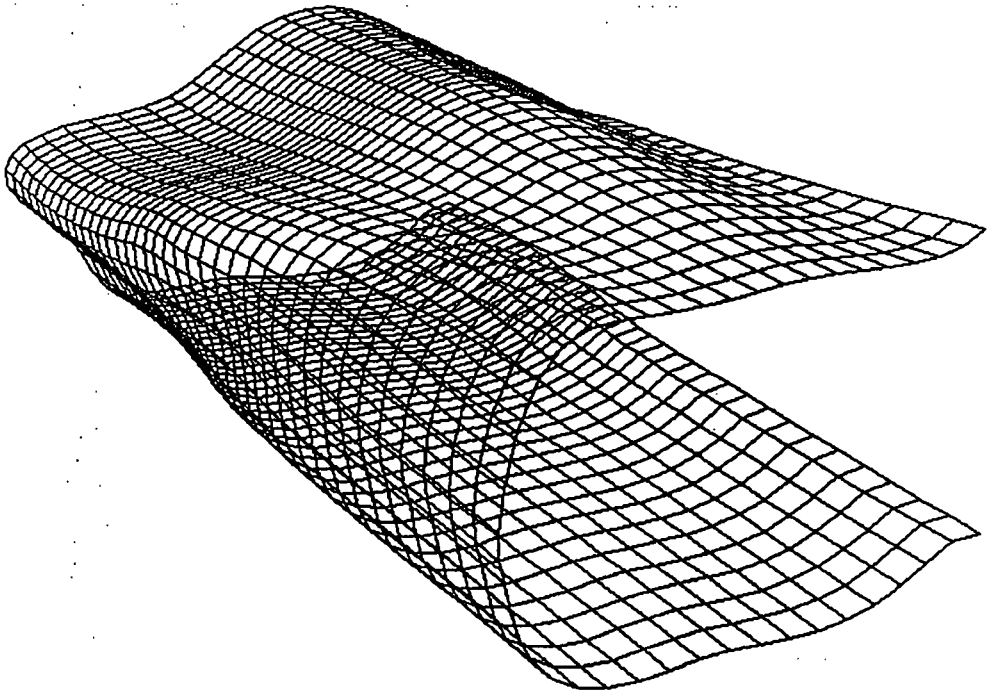


Figure 5.32 Smoothed F–15 aft quarter surface data

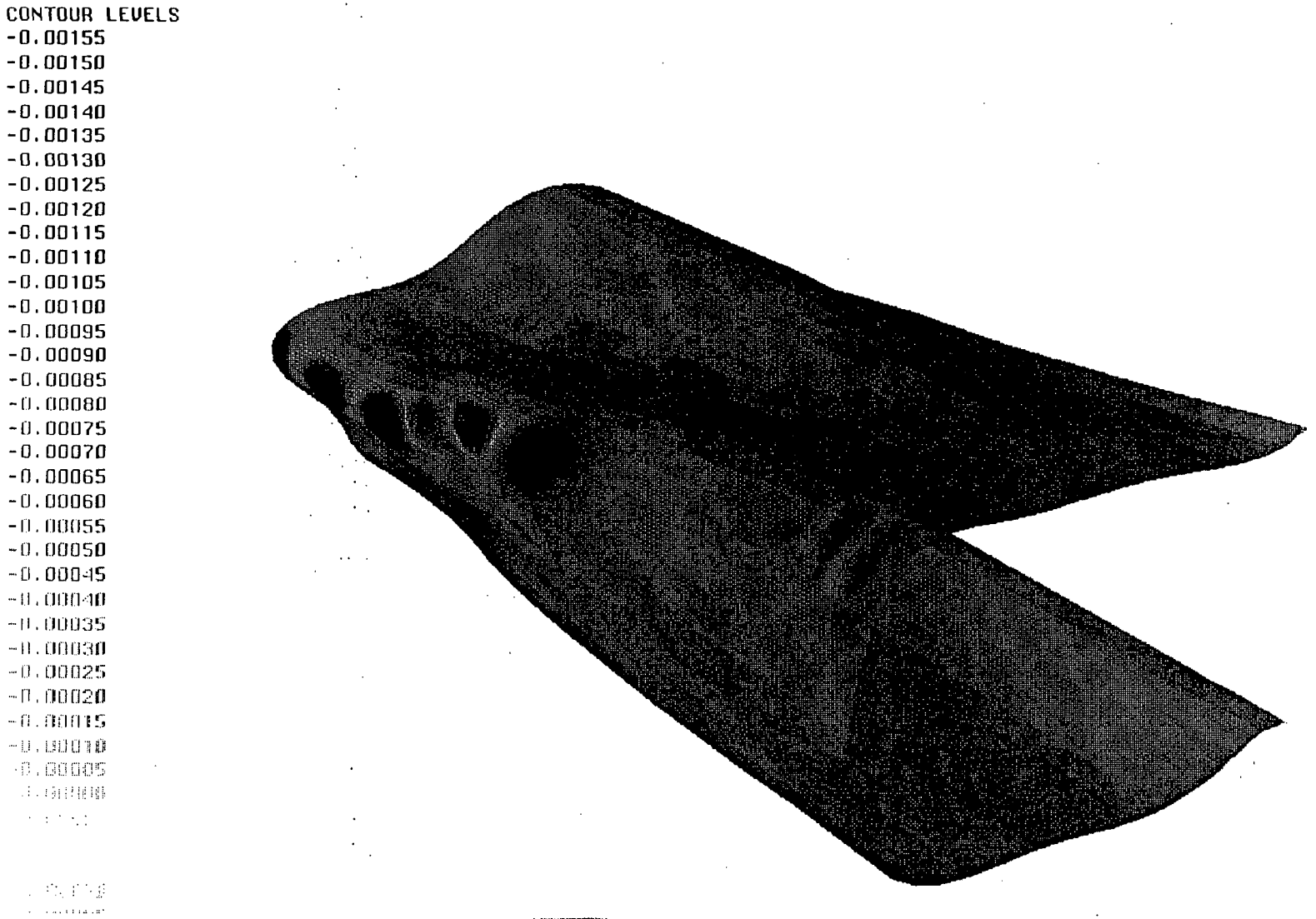



Figure 5.33 Gaussian curvature on the smoothed F–15 aft quarter surface spline

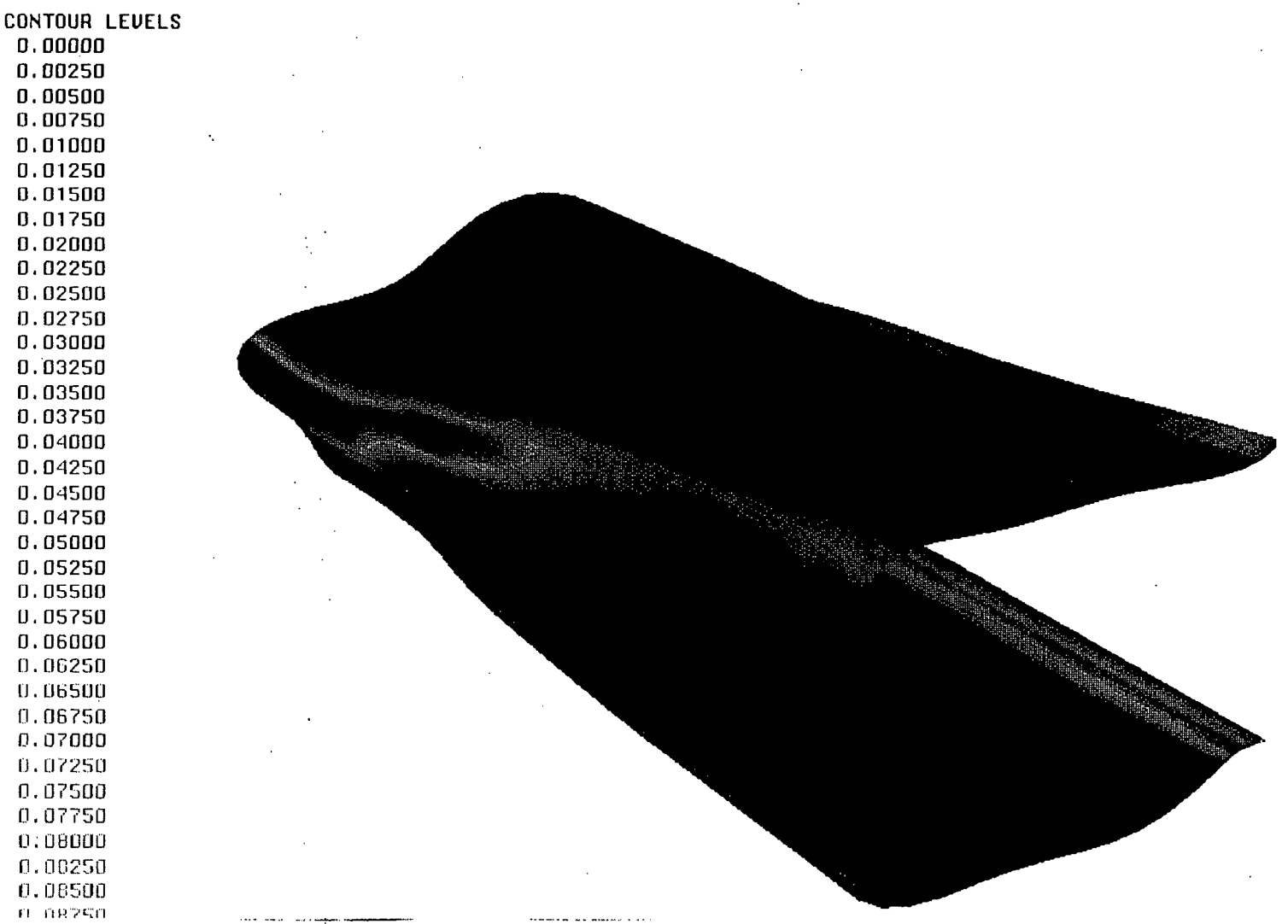



Figure 5.34 Absolute curvature on the smoothed F–15 aft quarter surface spline

## REFERENCES


[1] Kaufmann, E. and Klass, R. "Smoothing Surfaces Using Reflection Lines for Families of Splines" Computer–Aided Design, Vol 20, no. 6, 1988. pp 312–316.

[2] Beck, J.M. et al., "Surfaces Analysis Methods" IEEE Computer Graphics and Applications, December 1986.

[3] Hoschek, J. "Smoothing of Curves and Surfaces" Computer–Aided Geometric Design, Vol 2, 1985. pp 97–105.

[4] Renz, W. "Interactive Smoothing of Digitized Data Point Data" Computer–Aided Design, Vol 14, no 5, September 1982. pp 267–269.

[5] Sapidis, N. and Farin, G. "Automatic Faring Algorithm for B–spline Curves" Comp. Aided Design Vol 22, March 1990, pp 121–129.

[6] Farin, G. and Sapidis, N. "Curvatures and Fairness of Curves and Surfaces" IEEE Computer Graphics and Applications, Vol 9, no. 4, March 1989 pp 52–57.

[7] Farin, G. Curves and Surfaces for Computer Aided Geometric Design, Academic Press, 1988.

[8] Farin, G. Rein, G. Sapidis, N. and Worsey, A.J. "Fairing Cubic B–spline Curves" Computer Aided Geometric Design, Vol. 4, no. 1–2, July 1987, pp 91–93.

[9] Kjellander, J.A.P. "Smoothing of cubic splines"

[10] Kjellander, J.A.P. "Smoothing of bicubic parametric surface"

[11] Gerald Farin, Personal Communication.

[12] Su Bu–Qing and Liu Ding–Yaun. Computational Geometry Academic Press 1989.

[13] Faux, I.D. and Pratt, M. J. Computational Geometry for Design and Manufacture, Ellis Horwood Ltd. 1979.

[14] Thompson, J.F., Warsi, Z.U.A., Mastin, C.W. Numerical Grid Generation Foundations and Application, North Holland, 1985.

[15] Lancaster, P., Salkauskas, K. Curve and Surface Fitting An Introduction, Academic Press, 1986.

[16] Mortenson, M.E., Geometric Modeling, Wiley, 1985.

[17] Bartels, R., Beatty, J., Barsky, B. An Introduction to Splines for Use in computer Graphics and Geometric Modeling, Morgan Kaufmann Inc., 1987.

[18] Yoon, Y.H. "Enhancements and Extensions of the EAGLE Grid Generation System," Doctoral Dissertation, Miss. State Univ., 1991.

[19] Rivlin, J.T. An Introduction to the Approximation of Functions, Dover Publications, Inc., 1969.

[20] Davis, P.J. Interpolation and Approximation, Dover Publications, Inc., 1974.

[21] Lorentz, G. Bernstein Polynomials, Toronto Press, 1953.

[22] EAGLE Surface Generation Code Users Manual

[23] EAGLE Grid Generation Code Users Manual

[24] Abbot, I.H., von Doenhoff, A.E., Theory of Wing Sections, Dover Publications 1959.

[25] Taylor, L.K., "Unsteady Three–Dimensional Incompressible Algorithm Based on Artificial Compressibility," Doctoral Dissertation, Miss. State Univ., 1991.