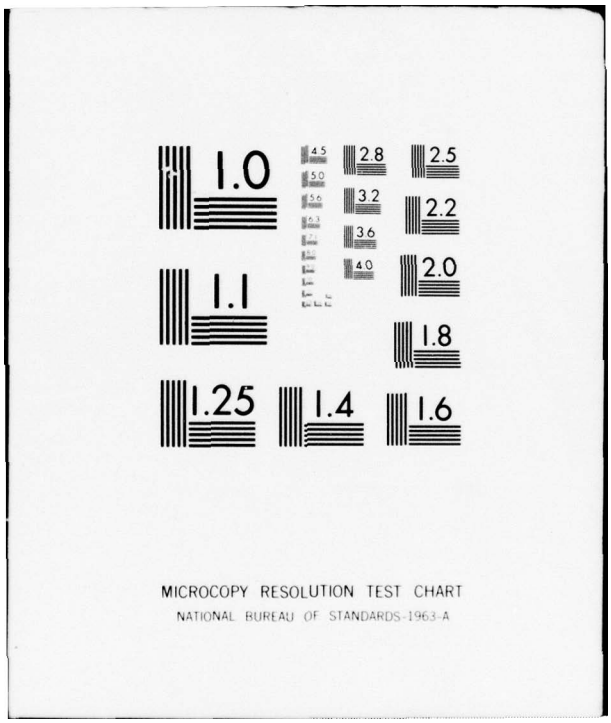
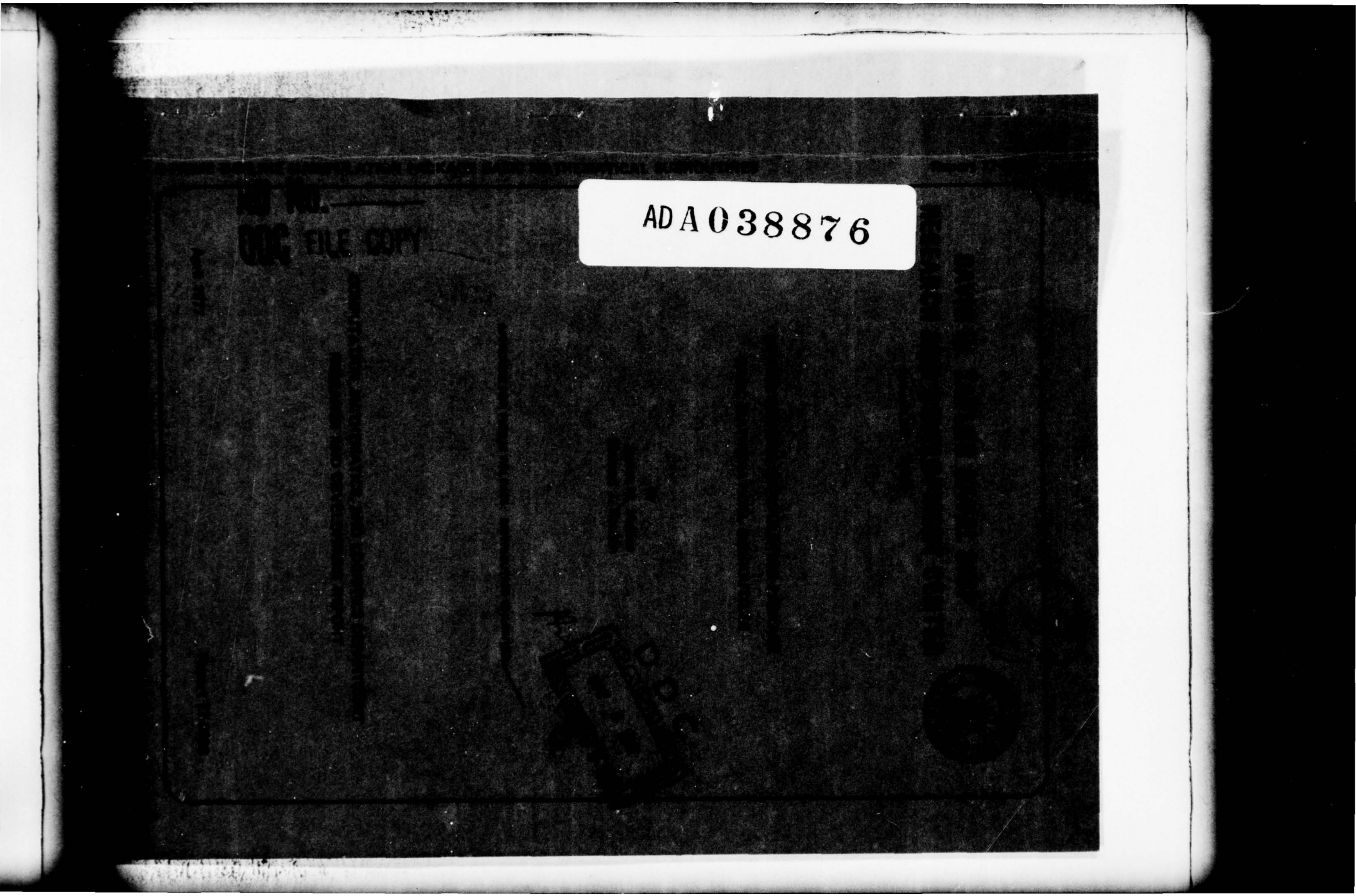
AD-A038 876 DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CE--ETC F/G 9/2
G-PRIME B-SPLINE MANIPULATION PACKAGE BASIC MATHEMATICAL SUBROU--ETC(U)
APR 77 J M MCKEE, R J KAZDEN

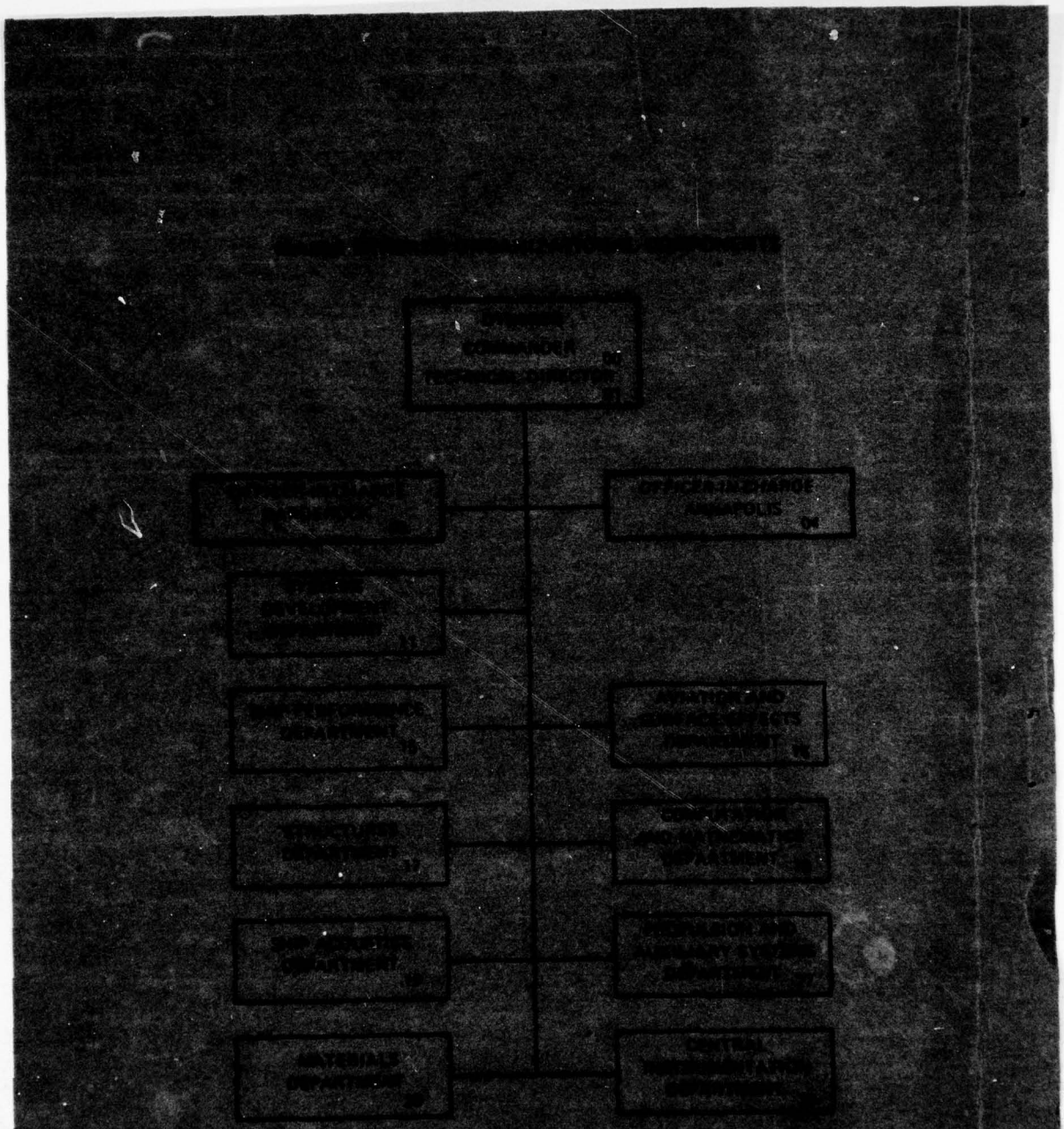
UNCLASSIFIED DTNSRDC REPORT 77-0036 NL

END

DATE FILMED 5-77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A


AD NO.
DDC FILE COPY
AD A038876



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: DTNSRDC Report 77-0036	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): G-PRIME B-SPLINE MANIPULATION PACKAGE BASIC MATHEMATICAL SUBROUTINES,	5. TYPE OF REPORT & PERIOD COVERED: Final rept.
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): James M. McKee, Richard J. Kazden	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: David W. Taylor Naval Ship Research and Development Center, Bethesda, Maryland 20084	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Task Area: 53532020; Work Unit: 1-1808-009
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE: Apr[illegible]77
(12) 93 p.	13. NUMBER OF PAGES: 93
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

B-spline functions
Curves
Intersecting curves
Intersecting surfaces
Lease-squares fitting
Mathematical subroutine library
Splines
Surfaces

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes a library of mathematical subroutines for defining and operating on B-spline curves and surfaces. This library contains subroutines for evaluating B-spline functions, for using B-splines as a basis for fitting curves and surfaces, and for finding the intersections of B-spline curves and surfaces. An explanation of

(Continued on reverse side)

next page

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(Block 20 continued)

cont → the general theory, a summary of important properties, and detailed operating instructions for each subroutine are given. A program illustrating a typical use of the basic subroutines has been included along with computer-generated plots of B-spline surfaces and intersection curves.

Although the basic evaluation subroutines use previously published techniques, the fitting and intersection procedures represent effective new approaches to the treatment of these old problems.

PREFACE

DTNSRDC is developing a programming language for describing structural geometry. This language, called G-PRIME, is designed to assist scientists and engineers in translating from concepts and blueprints to the data format required by mathematical simulation programs. G-PRIME must be able to accurately represent many geometric forms, and to reproduce, scale, move, truncate, combine, and find intersections of these various forms. One feature which greatly simplifies the design of G-PRIME, and one that probably makes it unique, is that all curve and surface geometry is represented using just one mathematical form--B-spline functions.

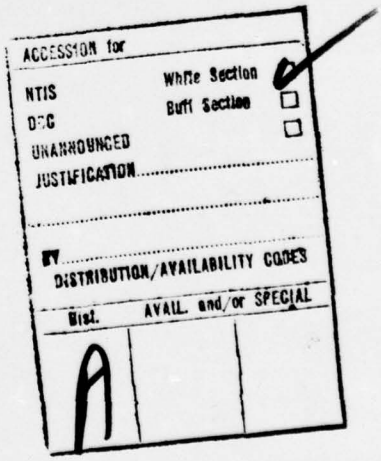
A number of projects within the Navy could benefit from a unified set of curve and surface manipulation subroutines, particularly projects in the hull representation and data fitting areas. This possibility prompted us to isolate G-PRIME's basic B-spline manipulation routines and to make them available as a separate subroutine library. This report contains a discussion of the mathematics involved in the use of B-spline functions, a description of the subroutines contained in this library, and a sample application which illustrates a typical use of the subroutines.

We have been pleased with the performance of these subroutines in our applications. It should be noted, however, that only a first increment of the intersection capability has been included at this time. Further, we now see areas in which improvements can be made and in which additional capability is necessary. This report concludes with a summary of our current work in the development of this basic B-spline capability and our recommendations for future directions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES



ABSTRACT

This report describes a library of mathematical subroutines for defining and operating on B-spline curves and surfaces. This library contains subroutines for evaluating B-spline functions, for using B-splines as a basis for fitting curves and surface data, and for finding the intersections of B-spline curves and surfaces. An explanation of the general theory, a summary of important properties, and detailed operating instructions for each subroutine are given. A program illustrating a typical use of the basic subroutines has been included along with computer-generated plots of B-spline surfaces and intersection curves.

Although the basic evaluation subroutines use previously published techniques, the fitting and intersection procedures represent effective new approaches to the treatment of these old problems.

PARAMETRIC B-SPLINE CURVES AND SURFACES

BACKGROUND

Since their introduction by I.J. Schoenberg in 1946,[1] mathematical spline techniques have become increasingly popular for smoothing and interpolating data and for functional approximation. Much of the popularity of these techniques can be attributed to the fact that mathematically smooth and aesthetically pleasing shapes are easily produced using splines. R. Riesenfeld has proposed using spline functions for curve and

---

[1] Schoenberg, I.J., "Contributions to the Problem of Approximations of Equidistant Data by Analytic Functions," Quart. Appl. Math., vol. 4 (1946), pp. 45-99; 112-141.

surface approximation in computer-aided design in his 1973 thesis.[2] The design method which he proposes is patterned after Bézier's technique,[3] but uses a well-behaved set of spline functions as the basis for approximation rather than the Bernstein polynomial basis usually associated with Bézier's method. The advantages of using these basic splines (or B-splines) include numerical stability, computational efficiency, and the ability to represent a variety of common curves and surfaces which are difficult to approximate using earlier techniques. Riesenfeld's thesis provides a good introduction to both Bézier's method and the theory of B-splines.

Bézier and Riesenfeld choose to represent curves and surfaces by vector-valued parametric functions instead of the scalar equations which are most often used for classical curves and surfaces. Although it is not necessary to use parametric equations with B-splines, the parametric form does enlarge the family of admissible shapes and makes possible several computational short cuts.

Our objective has been to produce a library of FORTRAN subroutines which can be used to perform most elementary operations involving B-spline curves and surfaces. The subroutines described in this report include (1) an implementation of the techniques described by Riesenfeld[2] for evaluating cubic B-splines having a uniform knot vector, (2) routines for fitting data with B-spline curves and surfaces, and (3) routines for finding the intersections of B-spline curves and surfaces.

For the reader who is not familiar with B-splines the following examples may help to illustrate the utility of such a library.

- A designer wishes to produce a transparent plastic canopy to enclose an instrument package. Since B-spline functions can be used directly to generate a smooth surface, the designer needs only to supply

---

[2] Riesenfeld, R., "Application of B-Spline Approximation to Geometric Problems of Computer-Aided Design," University of Utah Computer Science Report UTEC-CSC-73-126.

[3] Bézier, P., Numerical Control - Mathematics and Applications (translated by A.R. Forrest). London: John Wiley and Sons, 1972.

a rectangular mesh of points which describes the essential shape of the canopy and its boundaries in order to produce a trial surface. The evaluation routines can be invoked to calculate a number of points on the B-spline surface induced by the data given in the mesh. This surface may be graphically displayed for visual evaluation, it may be used with numerical integration techniques to calculate the volume enclosed, or it may be used to check clearances for the instrument package. If the design is not satisfactory, the designer can change it by modifying the points in the mesh. When a satisfactory design has been found, the evaluation routines can be called upon to supply enough points on the surface to produce a mold for manufacturing the canopy.

- An engineer is laying out detailed drawings for the fabrication of a plenum which involves circular pipes and an oval chamber meeting at oblique angles. He needs to know the curves of intersection of these parts for cutting the material. Points on the curves of intersection can be calculated using B-spline functions in a two-step process. First, the engineer provides data points on each of the intersecting pieces. These points are used by the fitting routines to determine B-spline surfaces which represent each of the original surfaces. The intersection subroutines will determine points on the curves of intersection of these B-spline surfaces.

- A technician would like to have experimental data plotted and have smooth curves drawn which interpolate the data points or pass close to the data points. The fitting routines can be used to determine a B-spline curve which either interpolates the data or best fits the data in the least-squares sense. The evaluation routines can then be called to obtain sufficient points on the curve for smooth plotting.

The remainder of this section describes the various mathematical procedures currently available in this library.

EVALUATION PROCEDURES

The B-Spline Curve

A parametric B-spline curve, $Q_m[\underset{\sim}{P};s]$, is said to be induced by the polygon $\underset{\sim}{P} = P_1, P_2, \ldots, P_m$ of m points, where each point is a vector with as many components as the dimensionality of the space. For a given polygon $\underset{\sim}{P}$, and parameter value s, the point on the induced B-spline curve is given by

$$Q_m[\underset{\sim}{P};s] = \sum_{i=1}^{m} N_{i,4}(s)P_i \qquad 0 \leq s \leq 1 \tag{1}$$

where $N_{i,4}(s)$ is the i^{th} normalized cubic B-spline basis function. $N_{i,4}(s)$ is computed using de Boor's recursive procedure[4]:

For M=1

$$N_{i,M}(s) = \begin{cases} 1 & \text{for } i \leq s\cdot(m-1) < i+1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

then for M = 2,3,4

$$N_{i,M}(s) = [(z-i)\cdot N_{i,M-1}(s) + (m+i-z)\cdot N_{i+1,M-1}(s)]/(m-1)$$

where $z = s\cdot(m-1)$

For closed curves a periodic basis is required which can be obtained by modifying the above procedure so that all differences and indices are computed modulo (m-1). Whenever $0 \leq s < 1/(m-1)$ or $(m-1)-1/(m-1) < s \leq 1$ the basic functions may be modified such that the end points of the polygon $\underset{\sim}{P}$ are interpolated by the B-spline curve and the slopes at the ends of the curve are the same as the slopes of the beginning and ending legs of the polygon. This modification is equivalent to introducing a *non-uniform* knot vector in the spline definition. For closed curves, values of s outside the interval $0 \leq s \leq 1$ are replaced by s modulo 1. For convenience, values of s outside the interval $0 \leq s \leq 1$ are also permitted for open curves. In this event basis functions are constructed to produce a straight line extension of the corresponding leg of the inducing

---

4 de Boor, C., "On Calculating with B-Splines," J. Approx. Theory, vol. 6 (1972), pp. 50-62.

polygon. $Q_m[\underset{\sim}{P};0]$ and $Q_m[\underset{\sim}{P};1]$ are still referred to as the "ends" of the curve, however.

A Linearly Interpolated Approximation to a B-Spline Curve

Frequently it is necessary to evaluate a B-spline curve for some set of uniformly spaced parameter values, say $(s_1,s_2,\ldots,s_K)$, and Equation (1) can be written in matrix form:

$$\underset{\sim}{Q} = [S]\underset{\sim}{P} \tag{3}$$

where

$$S_{ij} = N_{j,4}(s_i), \quad \underset{\sim}{P} = [P_1,P_2,\ldots,P_m]^T, \text{ and}$$

$$\underset{\sim}{Q} = [Q_m[\underset{\sim}{P};s_1], Q_m[\underset{\sim}{P};s_2],\ldots,Q_m[\underset{\sim}{P};s_k]]^T$$

If a matrix of B-spline coefficients, [S], has been previously computed and $Q_m[\underset{\sim}{P};s_i]$ is required where $s_1 \leq s_i \leq s_k$, $\hat{\underset{\sim}{Q}}$, an approximation to $\underset{\sim}{Q}$ may suffice for certain applications. We can obtain one such approximation by linear interpolation of the coefficients available in [S]. If the two parameter values used in computing [S], which bracket s_i, are s_j and s_{j+1}, then

$$\hat{Q}_m[\underset{\sim}{P};s_i] = Q_m[\underset{\sim}{P};s_j]+(Q_m[\underset{\sim}{P};s_{j+1}]-Q_m[\underset{\sim}{P};s_j])\cdot(s_i-s_j)/(s_{j+1}-s_j)$$

Substituting Equation (1) in the last equation gives

$$\hat{Q}_m[\underset{\sim}{P};s_i] = \sum_{\xi=1}^{m} [N_{\xi,4}(s_j)+(N_{\xi,4}(s_{j+1})-N_{\xi,4}(s_j))\cdot(s_i-s_j)/(s_{j+1}-s_j)]P_\xi,$$

or

$$\hat{Q}_m[\underset{\sim}{P};s_i] = \sum_{\xi=1}^{m} \hat{N}_{\xi,4}(s_i)P_\xi \tag{4}$$

The B-Spline Surface

A parametric B-spline surface is said to be induced by an m by n rectangular mesh of points [P], where each P_{ij} is a vector with as many components as the dimensionality of the space.

A point on a cubic B-spline surface corresponding to the pair of parameter values (s,t) is given by

$$Q_{mn}[[P];s,t] = \sum_{i=1}^{m} \sum_{j=1}^{n} N_{i,4}(s)N_{j,4}(t)P_{ij};\ 0 \le s \le 1,\ 0 \le t \le 1 \qquad (5)$$

This surface is a Cartesian product generalization of a B-spline curve and all qualifications imposed on the curve definition apply to the surface definition as well. The curves $Q_{mn}[[P];0,t]$, $Q_{mn}[[P];1,t]$, $Q_{mn}[[P];s,0]$, and $Q_{mn}[[P];s,1]$ are referred to as the "edges" of the surface.

To evaluate a B-spline surface for a set of points corresponding to the various combinations of the parameters $\{s_1,s_2,\ldots,s_k\}$ and $\{t_1,t_2,\ldots,t_\ell\}$ Equation (5) can be written in matrix form:

$$[Q] = [S][P][T^T] \qquad (6)$$

where $S_{ij} = N_{j,4}(s_i)$, $T_{ij} = N_{j,4}(t_i)$, [P] is the inducing mesh of points, and [Q] is the mesh of points on the B-spline surface.

If the B-spline coefficient matrices [S] and [T] are available, a linearly interpolated approximation to the B-spline surface can be obtained by substituting $\hat{N}_{j,4}(z)$ of Equation (4) for $N_{j,4}(z)$ in Equations (5) and (6).

FITTING PROCEDURES

The B-spline approximation to a curve or surface is particularly useful in the process of designing objects for which there are few quantitative constraints on the final form and for which the main concerns are gross dimensions, smoothness, and aesthetic acceptability. Such an approximation is usually not satisfactory as a mathematical representation of the shape of an existing object. Many of the advantages of B-spline techniques can still be brought to bear on this problem by employing least-squares fitting procedures using B-spline functions as the basis for fitting the given shape.

Curve Fitting

The problem of fitting an ordered set of data points, $\underset{\sim}{Q}$, with a B-spline curve is one of determining the inducing polygon, $\underset{\sim}{P}$, in

$$\hat{\underset{\sim}{Q}} = [S]\underset{\sim}{P}$$

such that the error,

$$e = |\underset{\sim}{Q} - \hat{\underset{\sim}{Q}}|$$

is minimized where the norm $|x| = \sum_{\text{all } i} x_i^2$, and [S] is the B-spline coefficient matrix used in Equation (3).

If we stipulate that there are at least as many points in $\underset{\sim}{Q}$ as there are in $\underset{\sim}{P}$, we can employ standard linear least-squares fitting techniques and write the following expression for $\underset{\sim}{P}$:

$$\underset{\sim}{P} = [U]\underset{\sim}{Q} \tag{7}$$

where

$$[U] = [S^T S]^{-1} S^T \tag{8}$$

Surface Fitting

The generalization of a least-squares fitted curve to a surface is more easily accomplished by introducing operator notation and zero indexing for the sums. If the cubic B-spline approximation is expressed as a linear operator, Q_m, on $C^{[d]}[0,1]$, where d is a positive integer, and the curve to be approximated is the function f(s), the approximation can be written as

$$Q_m[f;s] = \sum_{i=0}^{m} N_{i,4}(s)\ f\ (i/m)$$

The least-squares transformation [U] from Equation (8) can be used to define a least-squares fit operator F_m, such that

$$F_m[f;s] = \sum_{i=0}^{m} \sum_{\xi=0}^{k} N_{i,4}(s) U_{i\xi} f(\xi/k) \tag{9}$$

If the data point Q_ξ from $\underset{\sim}{Q}$ is substituted for each $f(\xi/k)$, Equation (9) becomes

$$F_m[\underset{\sim}{Q};s] = \sum_{i=0}^{m} N_{i,4}(s) P_i$$

where

$$\underset{\sim}{P} = [U]\underset{\sim}{Q}$$

With the same notation the Cartesian product bivariate approximation of a function f(s,t) is

$$Q_{m,n}[f;s,t] = Q_m Q_n[f;s,t] = \sum_{i=0}^{m} \sum_{j=0}^{n} N_{i,4}(s) N_{j,4}(t) f(i/m, j/n)$$

The Cartesian product generalization of the least-squares fitted curve is then

$$F_{m,n}[f;s,t] = F_m F_n[f;s,t] = \sum_{i=0}^{n} \sum_{\xi=0}^{k} \sum_{j=0}^{n} \sum_{\eta=0}^{\ell} N_{i,4}(s) N_{j,4}(t) U_{i\xi} W_{j\eta} f(\xi/k, \eta/\ell) \tag{10}$$

If the data point $Q_{\xi\eta}$ from [Q] is substituted for each $f(\xi/k,\eta/\ell)$, Equation (10) becomes

$$F_{m,n}[[Q];s,t] = \sum_{i=0}^{m} \sum_{j=0}^{n} N_{i,4}(s) N_{j,4}(t) P_{ij}$$

where

$$[P] = [U][Q][W^T] \tag{11}$$

$$[U] = [S^T S]^{-1}[S^T]$$

$$[W] = [T^T T]^{-1}[T^T] \tag{12}$$

and [S] and [T] are the B-spline coefficient matrices used in Equation (6).

For certain surface design applications it may be desirable to use fitting techniques for one parameter and a B-spline approximation for the other. This can be accomplished by substituting the identity matrix for either [U] or [W] in Equation (11), depending on the direction in which the approximation is desired. Once [U] and [W] have been computed for a problem of a given size, the fitting procedure for each set of data, [Q], involves only matrix multiplications. Some applications can exploit this property to reduce computing time.

INTERSECTION PROCEDURES

The problem of finding points of intersection of arbitrary curves and surfaces is essentially that of solving a system of nonlinear equations. We have approached the task of finding points of intersection of parametric cubic B-spline curves and surfaces by treating them as arbitrary *nonlinear functions*, rather than taking advantage of the special properties of B-splines. We have chosen this approach for two reasons: (1) the B-spline functions used for geometric applications are usually very well behaved and efficient numerical techniques for solving systems of

well-behaved nonlinear equations are readily available; and (2) one solution procedure can be used for finding intersections involving curves, surfaces, curves with surfaces, and for finding the point on a curve or surface which is closest to a given spatial point. We feel that the straightforward programming that can be achieved using this approach more than offsets any computing overhead incurred by not taking advantage of special properties.

After evaluating several iterative, nonlinear solvers we have chosen Brown's algorithm[5] for its efficiency and reliability. This algorithm employs a nonlinear, least-squares minimization technique that does not require explicit derivatives of the functions to be minimized.

The Intersection of Two Plane Curves

Although a procedure for finding the points of intersection of two planar curves is not available in the current version of B-spline subroutine library, this problem illustrates all the essential features of the existing intersection procedures.

Consider the two parametric curves

$$\underset{\sim}{f}_1(s_1) = \begin{Bmatrix} x_1(s_1) \\ y_1(s_1) \end{Bmatrix} \quad \text{and} \quad \underset{\sim}{f}_2(s_2) = \begin{Bmatrix} x_2(s_2) \\ y_2(s_2) \end{Bmatrix} \tag{13}$$

shown in Figure 1. The point of intersection may be found by finding a pair of parameter values, $(\hat{s}_1, \hat{s}_2)$, that makes the residual function

$$\underset{\sim}{R} = \begin{Bmatrix} R_1 \\ R_2 \end{Bmatrix} = \begin{Bmatrix} x_2(s_2) - x_1(s_1) \\ y_2(s_2) - y_1(s_1) \end{Bmatrix} = \underset{\sim}{f}_2(s_2) - \underset{\sim}{f}_1(s_1) \tag{14}$$

zero. $\underset{\sim}{R}$ is the function that is to be minimized by the iterative solver. For B-spline curves the $\underset{\sim}{f}_i(s)$'s are calculated using one of the B-spline evaluation procedures.

Iterative procedures require some initial value for each of the independent variables. When there is only one point of intersection, as

---

[5] Brown, K.M., "Derivative Free Analogues of the Levenberg-Marquardt and Gauss Algorithms for Nonlinear Least Squares Approximations," Numerische Mathematik, vol. 18, pp. 289-297, 1972.

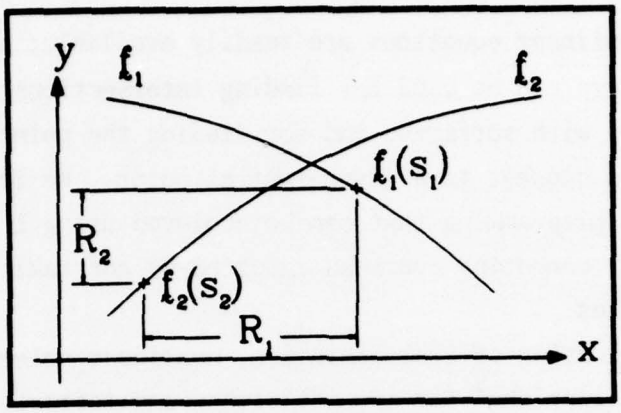


Figure 1 - Intersecting Parametric Curves

in Figure 1, almost any pair of initial values, which are not both zero, will yield convergence to the correct solution.

If the two curves have no point of intersection within the range defined by parameter values between zero and one and the linear extensions of the B-spline curves do intersect (see Evaluation Procedure section), the algorithm will quickly converge to a point outside the range. If the extended curves do not intersect, the minimization algorithm will choose the two closest points on the curves as the solution and will terminate with a non-zero residual function.

Locating the Closest Point on a Curve

If there are multiple intersection points, as in Figure 2, the user must select between them by specifying initial parameter values which are closest to the desired intersection point. If there are only two intersection points, this parameter estimate may not be an unreasonable request. Clearly, a spatial estimate of the location of the desired intersection would be more natural for the user. Further, spatial estimates seem to be mandatory if this capability is to be used effectively with computer graphics devices that have light pens or other graphical means of input.

If the user gives a spatial point, $\underset{\sim}{x}_0 = (x_0, y_0)$, the point on the curve, $\underset{\sim}{f}_1(s)$, that is closest to the point $\underset{\sim}{x}_0$ can be found by determining the parameter value, $\hat{s}_1$, that minimizes the residual function

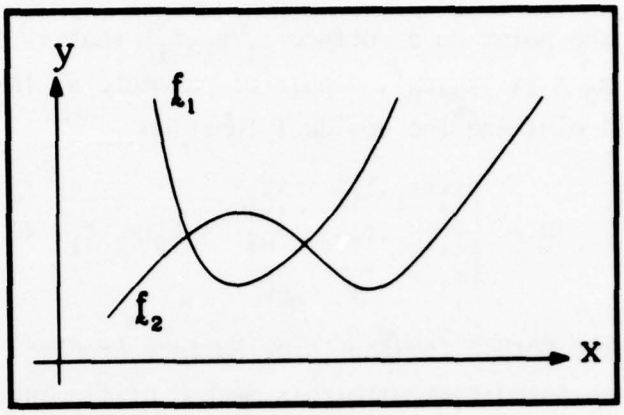


Figure 2 - Two Curves with Two Intersection Points

$$\underset{\sim}{R} = \begin{Bmatrix} x_1(s_1) - x_0 \\ y_1(s_1) - y_0 \end{Bmatrix} = \underset{\sim}{f}_1(s_1) - \underset{\sim}{x}_0 \qquad (15)$$

Again the iterative nonlinear minimization procedure is used to solve for $\hat{s}_1$. If the procedure is repeated with the second curve, in order to determine a parameter estimate, $\hat{s}_2$, the required initial parameter values will have been obtained to begin to solve for the desired intersection point.

The Intersection of Three Surfaces

The procedure for finding the point of intersection of three surfaces is the same as that described for two plane curves except that a pair of parameters, $(\hat{s}_i, \hat{t}_i)$, must be determined for each of the surface functions, $\underset{\sim}{f}_i(s_i,t_i)$, which makes residual function

$$\underset{\sim}{R} = \begin{Bmatrix} x_2(s_2,t_2) - x_1(s_1,t_1) \\ y_2(s_2,t_2) - y_1(s_1,t_1) \\ z_2(s_2,t_2) - z_1(s_1,t_1) \\ x_3(s_3,t_3) - x_2(s_2,t_2) \\ y_3(s_3,t_3) - y_2(s_2,t_2) \\ z_3(s_3,t_3) - z_2(s_2,t_2) \end{Bmatrix} = \begin{Bmatrix} \underset{\sim}{f}_2(s_2,t_2) - \underset{\sim}{f}_1(s_1,t_1) \\ \text{----------} \\ \underset{\sim}{f}_3(s_3,t_3) - \underset{\sim}{f}_2(s_2,t_2) \end{Bmatrix} \qquad (16)$$

zero.

To locate the point on a surface $\underset{\sim}{f}_1(s_1,t_1)$ that is closest to a given spatial point, $\underset{\sim}{x}_0 = (x_0,y_0,z_0)$, a pair of parameters, $(\hat{s}_1,\hat{t}_1)$, must be found which will minimize the residual function

$$\underset{\sim}{R} = \begin{Bmatrix} x_1(s_1,t_1) - x_0 \\ y_1(s_1,t_1) - y_0 \\ z_1(s_1,t_1) - z_0 \end{Bmatrix} = \underset{\sim}{f}_1(s_1,t_1) - \underset{\sim}{x}_0 \qquad (17)$$

Limitations of the Method for the Three-Surface Problem

It should be noted that with this method of finding points of intersection it is not too difficult to pose a problem that will fail. The most obvious example can be seen in Figure 3, which contains two plane curves which intersect at one point and come very close to intersecting at another (stationary point). If the initial parameter values are associated with points that are closer to the stationary point than the actual intersection, the method will usually converge to the stationary point and terminate, indicating failure to find an intersection. Problems of this type can be avoided by treating them as multiple intersection point problems. Other situations are possible which can be avoided only by subdividing the problem to restrict the region of definition.

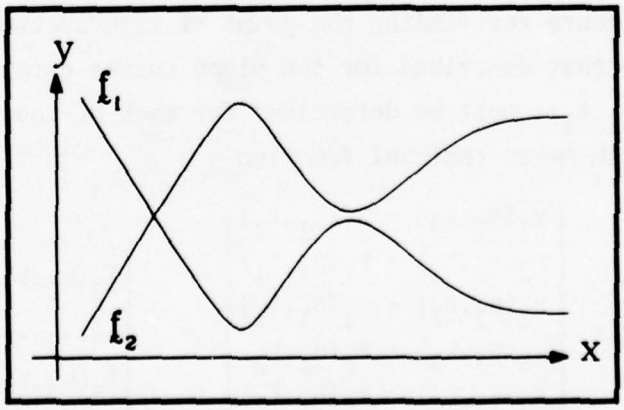


Figure 3 - An Intersection Point and a Stationary Point

The Intersection of Two Surfaces

The intersection capability of the B-spline subroutine library contains procedures for finding points on the curve of intersection of two B-spline surfaces. These points may be used directly or a fitted B-spline curve may be obtained using the procedures described earlier.

The two-surface intersection procedure has two major processing steps. First, an edge analysis is performed to determine the general location and extent of the intersection curve. The results of this analysis are in terms of ranges of parameter values on each of the surfaces. The second step involves traversing the curve by incrementing a selected parameter over the range of the curve and determining a point of intersection for each value of that parameter.

Edge Analysis

To find a curve of intersection of the two surfaces $\underset{\sim}{f}_1(s_1,t_1)$ and $\underset{\sim}{f}_2(s_2,t_2)$ some insight into the nature of such a curve can usually be gained by determining the points at which the boundary of one surface pierces the other. To accomplish this a procedure similar to the one described for finding points of intersection can be used. Determining the existence of these piercing points requires the investigation of eight possible combinations, each of which involves an edge of one surface and the other surface in its entirety. The respective residual functions for these eight combinations are as follows:

$$
\begin{aligned}
\underset{\sim}{R}_1 &= \underset{\sim}{f}_1(0,t_1)-\underset{\sim}{f}_2(s_2,t_2) & \qquad \underset{\sim}{R}_5 &= \underset{\sim}{f}_1(s_1,t_1)-\underset{\sim}{f}_2(0,t_2)\\
\underset{\sim}{R}_2 &= \underset{\sim}{f}_1(1,t_1)-\underset{\sim}{f}_2(s_2,t_2) & \qquad \underset{\sim}{R}_6 &= \underset{\sim}{f}_1(s_1,t_1)-\underset{\sim}{f}_2(1,t_2)\\
\underset{\sim}{R}_3 &= \underset{\sim}{f}_1(s_1,0)-\underset{\sim}{f}_2(s_2,t_2) & \qquad \underset{\sim}{R}_7 &= \underset{\sim}{f}_1(s_1,t_1)-\underset{\sim}{f}_2(s_2,0)\\
\underset{\sim}{R}_4 &= \underset{\sim}{f}_1(s_1,1)-\underset{\sim}{f}_2(s_2,t_2) & \qquad \underset{\sim}{R}_8 &= \underset{\sim}{f}_1(s_1,t_1)-\underset{\sim}{f}_2(s_2,1)
\end{aligned}
\tag{18}
$$

After the edge analysis, problems with two distinct edge pierce points (and certain closed surface problems with one edge pierce point) will be admitted for further processing. Any other outcome will cause the procedure to terminate, indicating failure.

Computing Points on the Curve of Intersection

The parameter values associated with the two edge pierce points are examined and the one with the greatest range is selected to control the traversing of the intersection curve. If the selected parameter, s_2, has the range $a \leq s_2 \leq b$ and the user has requested that k+1 points be found along the curve, the points are determined by submitting the following sequence of residual functions to the iterative solver:

$$\underset{\sim}{R} = \underset{\sim}{f}_1(s_1,t_1) - \underset{\sim}{f}_2(c,t_2) \tag{19}$$

where $c = a+(b-a)\cdot i/k,\ i = 0,1,\ldots,k$

When one or three edge pierce points is found and one of the surfaces is a closed surface, the closed surface parameter associated with the periodic B-spline functions is chosen to control the traversing of the curve.

Limitations of the Method for the Two-Surface Problem

Current two-surface intersection procedures admit only problems with one or two edge pierce points. This limitation excludes any problem with multiple curves of intersection and those problems in which a region of one surface bulges through the other surface. These problems can usually be treated by restricting the regions of definition or by subdividing the problem.

PROPERTIES OF B-SPLINE FUNCTIONS

The application of B-spline techniques is quite straightforward and extensive mathematical insight is not required for their use. Of course, an awareness of the major properties of B-spline functions should help the user avoid difficulties and perhaps open new avenues for their use. A short summary of properties is given below. Most of the proofs have been sketched by Risenfeld[2] or can be obtained by inspection of the B-spline definitions.

(1) If the evaluation procedures are applied to data points taken from a given curve, the induced B-spline curve will approximate that given curve with a certain degree of accuracy. As defined, B-spline functions yield a convergent approximation; hence, the more data points used, the better the approximation. The cubic B-spline approximation is also a

convergent approximation for the first three derivatives of a function.

(2) In regions of high curvature, more data points will be required to obtain a given level of approximation than in gently curving regions.

(3) A B-spline curve will either coincide with the inducing polygon or lie on the concave side of that polygon (convex hull property).

(4) The cubic B-spline curves described here have continuous second derivatives everywhere and continuous third derivatives everywhere with the possible exception of the points 0, 1/(m-1), 2/(m-1),...,1. A repeating data point in the polygon can reduce the differentiability by one at that point. Cusps or sharp corners can be induced in a B-spline curve by specifying the same data point three times.

(5) If three points of a polygon lie in a straight line, the induced curve will pass through the middle point exactly.

(6) It is apparent that Q_m can be considered a linear operator since

$$Q_m[a\underset{\sim}{P} + b\underset{\sim}{R};s] = aQ_m[\underset{\sim}{P};s] + bQ_m[\underset{\sim}{R};s]$$

This implies that one can apply a linear transformation to a polygon and use that transformed polygon to induce a B-spline curve which is equivalent to applying the linear transformation to the B-spline curve induced by the original polygon. Relevant linear transformations include rotation, translation, scaling, and projection.

Each of the above properties can be extended to B-spline surfaces using the Cartesian product generalization of the curve definition.

BASIC B-SPLINE SUBROUTINES

Our implementation of the B-spline manipulation procedures is in the form of a library of FORTRAN subroutines. The order of the previous section has been followed to group the subroutine descriptions according to the functions: Evaluation, Fitting, and Intersection. A fourth category, Utility, describes subordinate routines which are used only indirectly in the B-spline manipulation processes.

Although there are three primary evaluation subroutines, the main most general one for both curves and surfaces is BSEVL1. The other routines, BSEVL2 and BSEVL3, are somewhat more efficient under special conditions. Applications which involve several surfaces, each defined by the same size mesh, and which are to be evaluated for a fixed set of parameter values may substitute subroutine BSEVL2 for BSEVL1 to reduce computation. Subroutine BSEVL3 could also be substituted for BSEVL1, but since BSEVL3 only computes approximations to B-spline functions there are very few applications which can benefit from the use of this subroutine. The one fitting subroutine included in the library, FITBS2, is capable of performing all the fitting procedures described in the previous section for both curves and surfaces.

Only two intersection subroutines have been included at this time. One, INT3S, finds a point of intersection of three B-spline surfaces. The other, INT2S, finds points on the curve of intersection of two B-spline surfaces. INT3S is quite general, solves most problems of practical interest, and handles many pathological situations as well. INT2S is restricted to problems which have one continuous curve of intersection that includes at least one point on one of the edges of the two intersecting surfaces. This subroutine will handle a good range of practical problems, and a much broader class of problems can be treated if the user is willing to subdivide to meet the stated restrictions.

Subroutines INT2S and INT3S can also be accessed with simplified calling sequences by using subroutines INT2SX and INT3SX, respectively. The utility subroutines may be used independently to perform their various functions, but in general they have been programmed to be efficient for the tasks at hand and are not necessarily the type one would find in a general mathematical subroutine library.

GLOBAL CONVENTIONS

All the B-spline mathematical subroutines described in this report adhere to the following conventions:

(1) Each rectangular mesh of points, that contains data points taken from a surface, or points used to induce a B-spline surface, must be stored in a FORTRAN array dimensioned as follows:

P(MD, M1, N1)

where MD is the dimensionality of the space, M1 is the number of mesh points along an edge of the mesh associated with the variation of the first parameter, and N1 is the number of mesh points along an edge of the mesh associated with the variation of the second parameter. The first parameter is referred to as the "s" parameter and the second parameter is referred to as the "t" parameter (see Figure 4).

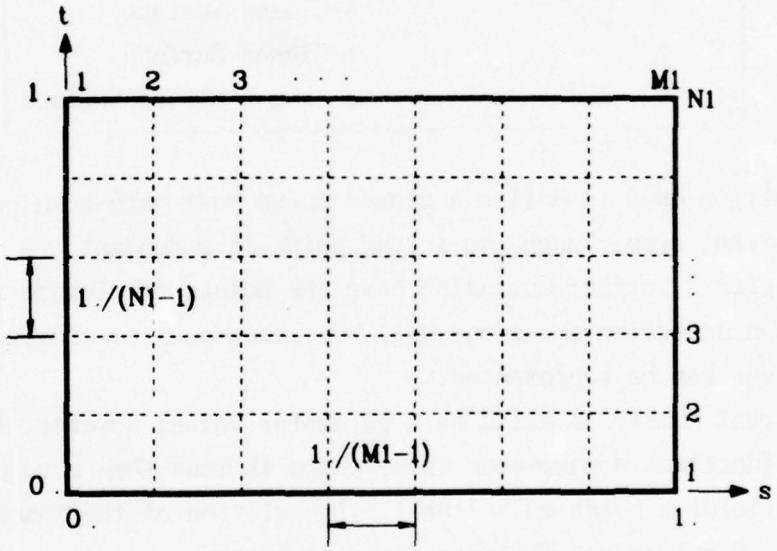


Figure 4 - Surface Mesh Conventions

(2) Each polygon of points, that contains data points taken from a curve, or points used to induce a B-spline curve, must be stored in a FORTRAN array dimensioned as follows:

P(MD, M1)

where MD is the dimensionality of the space, and M1 is the number of points in the polygon. Note that this polygon is equivalent to a surface mesh with either M1 = 1 or N1 = 1.

(3) An integer code word, ITYPE, is associated with each mesh or polygon of points used to define a B-spline curve or surface. The type of B-spline

curve or surface desired is indicated by setting ITYPE according to the values in Table 1.

TABLE 1 - CURVE AND SURFACE TYPE CODES

ITYPE	CURVE OR SURFACE TYPE
1	Open Curve
2	Closed Curve
3	Open Surface
4	s-Closed Surface
5	t-Closed Surface
6	s- and t-Closed Surface

(4) A polygon used to define a closed curve must have both points of closure given, even though the second point is redundant. A mesh used to define a closed surface must also have the points of closure repeated. For a given direction a surface will be either open or closed. No hybrid combinations can be represented.

(5) Any real number is valid as a parameter value; however, for non-periodic functions a parameter value which is less than zero or greater than one yields a point on a linear extrapolation of the function.

(6) Cubic B-spline coefficients for a given parameter value are normally packed into a four-word array, CN, and a packing index, LFT, is associated with those coefficients. Details of the packing algorithm are included with the description of subroutine MKSPLN.

For most applications only a few of the library subroutines will be called directly by the user. For example, in a program involving surfaces the user is required to read or generate the mesh of points, [P], to define each surface. He may then determine a new mesh of points which will induce a B-spline surface that best fits his original set of points by calling subroutine FITBS2. Surface meshes, regardless of their source, are all used in the same way in the application portion of the program. There, subroutine BSEVL1 can be called to obtain the coordinates of points on the surface and the intersection subroutines can be used to find intersection points and curves.

EVALUATION SUBROUTINES

* Primary Access Subroutines

BSCMAT - Subroutine Description

Function: To compute a cubic B-spline coefficient matrix.

Entry Point: BSCMAT

Calling Sequence: CALL BSCMAT(CS,M1,K1,ICLOSE)

Calling Arguments:

Name (Attributes)	Contents
CS (mixed/array/output)	Cubic B-spline coefficient matrix, [S], in packed form. Dimensioned CS(5,K1).
M1 (integer/input)	Row dimension of full [S] matrix.
K1 (integer/input)	Column dimension of [S] matrix.
ICLOSE (integer/input)	Flag word. A value of zero indicates that a non-periodic B-spline basis is to be used to compute coefficients. A value of one indicates that a periodic basis is to be used.

COMMON Areas: None

FORTRAN Data Files: None

Required Subroutines: MKSPLN

Method: Subroutine MKSPLN is called to compute rows of the coefficient matrix for the K1 parameter values 0, 1./(k1-1), 2./(K1-1),...,1. The non-zero elements of each row are stored in the first four words of each row of CS and the associated integer packing index is stored in the fifth word of each row.

Remarks:

(1) See description of subroutine MKSPLN for matrix packing algorithm.

BSEVL1 - Subroutine Description

Function: To evaluate a parametric cubic B-spline curve or surface function for a given set of parameter values, yielding the coordinates of a point on the function.

Entry Point: BSEVL1

Calling Sequence: CALL BSEVL1(P,MD,M1,N1,S,T,ITYPE,XYZ)

Calling Arguments:

Name (Attributes)	Contents
P (real/array/input)	Polygon or mesh defining a B-spline curve or surface.
MD,M1,N1 (integer/input)	Dimensions of array P.
S (real/input)	Parameter value of point to be evaluated on B-spline curve or first parameter of point on a surface.
T (real/input)	Second parameter value of point to be evaluated on B-spline surface.
ITYPE (integer/input)	Code word indicating type of B-spline curve or surface (see Table 1).
XYZ (real/array/output)	Coordinates of point evaluated. Dimensioned XYZ(MD).

COMMON Areas: None

FORTRAN Data Files: None

Required Subroutines: MKSPLN

Method: This subroutine evaluates Equation (1) if a polygon defining a B-spline curve has been given or Equation (2) if a mesh defining a B-spline surface has been given. Subroutine MKSPLN is called to compute the B-spline coefficients, $N_{i,4}(s)$ and $N_{j,4}(t)$.

Remarks:

(1) The dimension N1 and the parameter value T are not referenced if a curve is indicated by ITYPE.

BSEVL2 - Subroutine Description

Function: To evaluate a parametric cubic B-spline curve or surface function using precomputed coefficient matrices, yielding the coordinates of a point on the function.

Entry Point: BSEVL2

Calling Sequence: CALL BSEVL2(P,MD,M1,N1,K,L,CS,LS1,CT,LT1, ITYPE,XYZ)

Calling Arguments:

Name (Attributes)	Contents
P (real/array/input)	Polygon or mesh defining a B-spline curve or surface.
MD,M1,N1 (integer/input)	Dimensions of array P.
K (integer/input)	Index of the parameter value s_k for which the function is to be evaluated.
L (integer/input)	Index of the parameter value t_ℓ for which the function is to be evaluated.
CS (mixed/array/input)	B-spline coefficient matrix [S] of Equations (3) and (6) in packed form. Dimensioned CS(5,LS1).
LS1 (integer/input)	Column dimension of [S] and the number of uniformly spaced parameter values s_i for which the B-spline coefficients have been calculated.
CT (mixed/array/input)	B-spline coefficient matrix [T] of Equation (6) in packed form. Dimensioned CT(5,LT1).
LT1 (integer/input)	Column dimension of [T] and the number of uniformly spaced parameter values, t_i, for which the B-spline coefficients have been calculated.

ITYPE (integer/input)	Code word indicating type of B-spline curve or surface (see Table 1).
XYZ (real/array/output)	Coordinates of point evaluated. Dimensioned XYZ(MD).

COMMON Areas: None

FORTRAN Data Files: None

Required Subroutines: None

Method: This subroutine evaluates Equation (1) if a polygon defining a B-spline curve has been given or Equation (5) if a mesh defining a B-spline surface has been given. The coefficients $N_{i,4}(s_k)$ and $N_{i,4}(t_\ell)$ are obtained from the arrays CS and CT respectively.

Remarks:

(1) The dimension N1, the parameter index L, the coefficient matrix CT, and the dimension LT1 are not referenced if a curve is indicated by ITYPE.

(2) The coefficient matrices CS and CT can be computed by subroutine BSCMAT.

BSEVL3 - Subroutine Description

Function: To evaluate a linearly interpolated approximation to a cubic B-spline curve or surface function, yielding the coordinates of a point on the approximate function.

Entry Point: BSEVL3

Calling Sequence: CALL BSEVL3(P,MD,M1,N1,S,T,CS,LS1,CT,LT1, ITYPE,XYZ)

Calling Arguments:

Name (Attributes)	Contents
P (real/array/input)	Polygon or mesh defining a B-spline curve or surface.
MD,M1,N1 (integer/input)	Dimensions of array P.
S (real/input)	Parameter value of point to be evaluated on the approximate curve or first parameter of point on a surface.
T (real/input)	Second parameter value of point to be evaluated on approximate surface.
CS (mixed/array/input)	B-spline coefficient matrix [S] of Equations (3) and (6) in packed form. Dimensioned CS(5,LS1).
LS1 (integer/input)	Column dimension of [S] and the number of uniformly spaced parameter values s_i for which the B-spline coefficients have been calculated.
CT (mixed/array/input)	B-spline coefficient matrix [T] of Equation (6) in packed form. Dimensioned CT(5,LT1).
LT1 (integer/input)	Column dimension of [T] and the number of uniformly spaced parameter values, t_i, for which the B-spline coefficients have been calculated.

ITYPE (integer/input)	Code word indicating type of B-spline curve or surface (see Table 1).
XYZ (real/array/output)	Coordinates of point evaluated. Dimensioned XYZ(MD).

COMMON Areas: None

FORTRAN Data Files: None

Required Subroutines: PLSPLN

Method: This subroutine evaluates Equation (4) if a polygon defining a B-spline curve has been given or a variant of Equation (5) if a mesh defining a B-spline surface has been given. Subroutine PLSPLN is called to compute the approximate B-spline coefficients, $\hat{N}_{i,4}(s)$ and $\hat{N}_{j,4}(t)$.

Remarks:

(1) The dimension N1, the parameter T, the coefficient matrix CT, and the dimension LT1 will not be referenced if a curve is indicated by ITYPE.

(2) The coefficient matrices CS and CT can be computed by subroutine BSCMAT.

(3) This subroutine was designed to quickly calculate an approximation to a B-spline function for the early stages of iterative nonlinear solution routines. In tests, it has executed at about twice the speed of subroutine BSEVL1, which was not fast enough to be beneficial to the overall process.

MKSPLN - Subroutine Description

Function: To compute B-spline coefficients for a given parameter value.

Entry Point: MKSPLN

Calling Sequence: CALL MKSPLN(S,CN,M1,MBORD,ICLOSE,LFT)

Calling Arguments:

Name (Attributes)	Contents
S (real/input)	Parameter value for which coefficients are to be computed.
CN (real/array/output)	B-spline coefficients in packed form. Dimensioned CN(MBORD).
M1 (integer/input)	Length of full (unpacked) vector of coefficients.
MBORD (integer/input)	Order of B-spline basis function to be used to compute coefficients.
ICLOSE (integer/input)	Flag word. If zero, a non-periodic or open B-spline basis is to be used. If one, a periodic or closed B-spline basis is to be used.
LFT (integer/output)	A matrix packing index. See Method for description.

COMMON Areas: None

FORTRAN Data Files: None

Required Subroutines: None

Method: Non-zero B-spline coefficients are computed using Formula (2) and stored in the array CN. A matrix packing index, LFT, is computed for later reconstruction of a full vector. LFT = S*FLOAT(M1-1)+1. for a non-periodic basis. For a periodic basis the computation of LFT is made modulo (M1-1).

Remarks:

(1) For cubic B-spline functions MBORD must be set to 4.

(2) A specific B-spline coefficient used in Formula (2) can be obtained from the array CN using the following algorithm (note that the

argument MBORD is the same as the subscript M in the formula):

a. For a non-periodic basis the coefficient $N_{i,M}(S)$ is stored in CN(IPOINT) where IPOINT = i-LFT+(MBORD+1)/2 whenever $1 \leq$ IPOINT $\leq$ MBORD. Otherwise, $N_{i,M}(S) = 0$.

b. For a periodic basis the coefficient $N_{i,M}(S)$ is stored in CN(IPOINT) where IPOINT is computed as in the non-periodic case, except the computation is performed modulo (M1-1). Note that $1 \leq i \leq$ (M1-1) for a periodic basis.

If the index i is required, it may be obtained from the index IPOINT and the parameter value using the formula:

$$i = \text{IPOINT} + \text{LFT} - (\text{MBORD}+1)/2$$

For a non-periodic basis values of i outside the interval [1,M1] must be excluded. For a periodic basis i must be evaluated modulo (M1-1) when i > (M1-1) and taken as i modulo (M1-1)+(M1-1) when i < 1.

(3) Modified coefficients which cause non-periodic B-spline functions to pass through the end points of a defining polygon are calculated for cubic B-splines (MBORD=4). An open curve, for example, to be represented by data points associated with M1 uniformly spaced parameter values, requires extra data near the end points for a non-periodic B-spline basis. Specifically required are data at points associated with the parameter values

S = 1./FLOAT(3.*(M1-1)) and

S = 1.-1./FLOAT(3.*(M1-1)

Roughly, this represents data at one third the distance between the end point and its adjacent point. These values are obtained by linear interpolation. This method produces a B-spline curve which (1) interpolates the end points of the inducing polygon, and (2) has the same slope at the endpoints as does the end leg of the polygon.

(4) Although we chose to use normalized B-spline functions (curves and surfaces were defined only for parameter values between zero and one), it is sometimes useful to have smooth continuation of a curve or surface outside the range of definition. Subroutine MKSPLN does provide for linear extrapolation of open B-spline forms when parameter values outside the range zero to one are given. For parameter values which are less than zero the coefficients are packed in array CN as they would be for a parameter

value of zero. Similarly, for parameter values greater than one, the coefficients are packed as they would be for a parameter value of one. For periodic B-spline functions, the parameter value used, SS, will be computed from a given value S by the formula:

```
SS = AMOD(AMOD(S,1.)+1.,1.)
```

PLSPLN - Subroutine Description

Function: To compute approximate B-spline coefficients for a given parameter value by linearly interpolating from precomputed coefficients.

Entry Point: PLSPLN

Calling Sequence: CALL PLSPLN(S,CN,M1,MBORD,CS,K1,ICLOSE,LFT)

Calling Arguments:

Name (Attributes)	Contents
S (real/input)	Parameter value for which coefficients are to be computed.
CN (real/array/output)	B-spline coefficients in packed form. Dimensioned CN(MBORD+1).
M1 (integer/input)	Length of full (unpacked) vector of coefficients and row dimension of [S] matrix.
MBORD (integer/input)	Order of B-spline basis function to be used to compute coefficients.
CS (mixed/array/input)	B-spline coefficient matrix, [S], in packed form. Dimensioned CS(5,K1).
K1 (integer/input)	Column dimension of [S] matrix.
ICLOSE (integer/input)	Flag word. If zero, a non-periodic or open B-spline basis is to be used. If one, a periodic or closed B-spline basis is to be used.

COMMON Areas: None

FORTRAN Data Files: None

Required Subroutines: None

Method: Non-zero linearly interpolated B-spline coefficients, N, are computed as in Equation (4) and stored in the array CN. A matrix packing index, LFT, is computed for later reconstruction of a full vector. LFT = S*FLOAT(M1-1)+1, for a non-periodic basis. For a periodic basis the computation of LFT is made modulo (M1-1).

Remarks:

(1) The combination of the interpolation algorithm and the matrix packing technique requires that K1 ≥ M1.

(2) This subroutine is used in the same manner as subroutine MKSPLN and all the "Remarks" which pertain to MKSPLN also apply to this subroutine. In general, MBORD+1 non-zero interpolated coefficients are produced by a call to PLSPLN, rather than the MBORD non-zero coefficients computed by a call to MKSPLN. This fact must be accounted for in subsequent calculations and prevents the direct substitution of subroutine PLSPLN for MKSPLN.

FITTING SUBROUTINES

* Primary Access Subroutine

FITB01 - Subroutine Description

Function: To compute a least-squares-fit transformation matrix for B-spline curves and surfaces.

Entry Point: FITB01

Calling Sequence: CALL FITB01(CS,CS,M1,MORIG,U,IOPT,IFS,SCR)

Calling Arguments:

Name (Attributes)	Contents
CS (mixed/array/input)	B-spline coefficient matrix in packed form. Dimensioned CS(5,MORIG). The array CS is required as the first argument for real references and as the second argument for integer references.
M1 (integer/input)	The number of B-spline basis functions used for fitting.
MORIG (integer/input)	The number of data points to be fit.
U (real/array/output)	Least-squares fit transformation matrix. Dimensioned U(M1,MORIG).
IOPT (not used)	
IFS (integer/input)	Flag word. If zero, CS contains coefficients for a non-periodic or open B-spline function. If one, CS contains coefficients for a periodic or closed B-spline function.
SCR (array/scratch)	A working storage area KORE words long, where KORE=(M(M1+1)*M1)/2.

COMMON Areas: None

FORTRAN Data Files: None

Required Subroutines: BCKSUB, BSMULT, DECOMP

Method: This subroutine computes the least-squares transformation [U] given by Equation (8). Subroutine BSMULT is called to compute the product $[S^T][S]$, where the coefficient matrix [S] is given in packed form in array CS. Subroutines DECOMP and BCKSUB are called to solve the system of equations $[S^TS][U] = [S^T]$ for [U].

Remarks:

(1) A program stop "STOP 67" will be made whenever errors occur which cause the matrix $[S^TS]$ to be singular. This should occur only when CS, M1, and MORIG are not compatible.

FITBS2 - Subroutine Description

Function: To determine a polygon or two-dimensional mesh which will induce a B-spline curve or surface that best fits a set of curve or surface data points in the least-squares sense.

Entry Point: FITBS2

Calling Sequence: CALL FITBS2(Q,P,MD,MORIG,NORIG,M1,N1,ITYPE,RMS, IOPT,SCR,IFAIL)

Calling Arguments:

Name (Attributes)	Contents
Q (real/array/input)	Rectangular array of curve or surface data points. Dimensioned Q(MD,MORIG,NORIG).
P (real/array/output)	Polygon or mesh defining a B-spline curve or surface. Dimensioned P(MD,M1,N1).
MD,MORIG,NORIG (integer/input)	Dimensions of array Q.
M1 (integer/input/output)	The number of B-spline basis functions to be used to construct the B-spline curve or the number of basis functions to be used in the S-direction of a B-spline surface.
N1 (integer/input/output)	The number of B-spline basis functions to be used in the t-direction of a B-spline surface.
ITYPE (integer/input)	Code word indicating type of B-spline curve or surface (see Table 1).
RMS,IOPT (not used)	
SCR (array/scratch)	A working storage area which is KORE words long, where KORE=M1*MORIG+MAX0(K1,K2,K3), K1=5*MORIG+(M1*(M1+1))/2,

K2=N1*NORIG+5*NORIG+(N1*(N1+1))/2,
and
K3=N1*NORIG+MD*M1*NORIG.

IFAIL (integer/output) — Flag word. A value of zero indicates successful completion. A value of 1 indicates that the specified combination of MORIG, NORIG, M1, N1, and ITYPE is incompatible. A value of 2 indicates that an illegal value of ITYPE has been encountered.

COMMON Areas: None

FORTRAN Data Files: None

Required Subroutines: BSCMAT, FITB01, MULMAT

Method: This subroutine will evaluate Equation (7) to obtain the polygon P from a set of curve data points Q or Equation (11) to obtain the mesh [P] from a set of surface data points. Subroutine BSCMAT is called to compute the B-spline coefficient matrices [S] and [T]. Next, subroutine FITB01 is called to compute the least-squares transformations [U] and [W] using Equations (8) and (12). Then, subroutine MULMAT is called to perform the final multiplications to obtain P, or [P].

Remarks:

(1) If the value of M1 is zero, it is changed to the value of MORIG and the transformation [U] is replaced by the identity matrix. Similarly, if N1 is zero, it is set to NORIG and [W] is replaced by the identity matrix.

(2) The arrays Q and P may overlap for surface fitting applications whenever Q is not required for subsequent calculations. The arrays Q and P may not overlap for curve fitting applications.

(3) The transformation matrices [S] and [T], as computed by subroutine BSCMAT, are based on a uniform spacing of parameter values between zero and one. A weighted least-squares fit can be obtained by providing a substitute for BSCMAT which will compute a non-uniform transformation matrix.

INTERSECTION SUBROUTINES

* Primary Access Subroutines

ABC - Subroutine Description

Function: To determine where the boundary of one of the surfaces intersects the other surface.

Entry Point: ABC

Calling Sequence: *CALL ABC(I,J,K,ITOL,NGU)*

Calling Arguments:

Name (Attributes)	Contents
I (integer/input)	Surface whose boundary is to be used.
J (integer/input)	Describes which part of boundary (of surface which is being considered in terms of its boundary) is to be used. 1 = boundary for which s is constant. 2 = boundary for which t is constant.
K (integer/input)	Describes which part of boundary (of surface which is being considered in terms of its boundary) is to be used. 1 = parameter which is to be constant will be made 0.0. 2 = parameter which is to be constant will be made 1.0.
ITOL (integer/input)	Control parameter for iterative non-linear minimization subroutine.
NGU (integer/input)	Number of attempts to be made to find intersection points along each edge of the two surfaces during edge analysis.

COMMON Areas: BOPRXX, STCON, BOND, SEL

FORTRAN Data Files: None

Required Subroutines: OUTRNG, GINTR, RF2SS, RF2ST, STR

Method: Use iterative solver.

CHKR - Subroutine Description

Function: To evaluate the outcome of a call to the iterative solver.

Entry Point: CHKR

Calling Sequence: CALL CHKR(I,J,K,IRF,PARAM)

Calling Arguments:

Name (Attributes)	Contents
I (integer/input)	Specifies which surface is of concern for solution just attempted. 0 = all three surfaces i = Surface i (i=1,2, or 3)
J (integer/input)	Specifies which type of convergence is considered acceptable from solution just attempted. 0 = Accept EPS-convergence (all residuals less than the specified tolerance) or NSIG-convergence (independent variables for two consecutive iterations all agreed to within the specified tolerance). 1 = Accept EPS-convergence only.
K (integer/output)	Tells whether solution from iterative solver is acceptable. 0 = Pass; 1 = Fail.
IRF (integer/input)	Termination parameter returned by iterative solver which relates what happened when it was called.
PARAM (real/array/input/output)	Parameters (s_1,t_1,s_2,t_2,s_3,t_3). Dimensioned PARAM(6).

COMMON Areas: BOPRXX

FORTRAN Data Files: None

Required Subroutines: OUTRNG

Method: Determine whether solution from iterative solver is acceptable based on the values of the termination parameter (IRF) and the surface parameters (PARAM). Periodic surface parameters will be modified so that they fall in the range [0,1].

EFF - Function Subroutine Description

Function: To build termination code word for intersection subroutines.

Entry Point: EFF

Calling Sequence: X = EFF(I,J)

Calling Arguments:

Name (Attributes)	Contents
EFF (real/function/output)	Termination code word.
I (integer/input)	Number used to indicate where in the main subroutine a failure occurred.
J (integer/input)	Number which relates outcome of a call to subroutine GINTR.

COMMON Areas: None

FORTRAN Data Files: None

Required Subroutines: None

GINTR - Subroutine Description

Function: To iteratively minimize a set of nonlinear functions.

Entry Point: GINTR

Calling Sequence: CALL GINTR(RESFN,FAR,ITOL,PARVAL,NRES,NVAR,IFAIL)

Calling Arguments:

Name (Attributes)	Contents
RESFN (real/external function/ input)	Function used to compute residuals.
FAR (real/input)	Control parameter for iterative solver.
ITOL (integer/input)	Control parameter for iterative solver.
PARVAL (real/array/input/ output)	The independent variables for the nonlinear functions. As input: initial guesses. As output: solution. Dimensioned: PARVAL(NVAR)
NRES (integer/input)	Number of dependent variables (nonlinear functions).
NVAR (integer/input)	Number of independent variables.
IFAIL (integer/output)	Parameter which indicates outcome of a call to this subroutine. -1 = iterative solver failed to converge. 0 = iterative solver converged with all function values less than the specified tolerance. 1 = iterative solver converged when the independent variables for two consecutive iterations all agreed to within a specified tolerance. 7 = the working storage required for the iterative solver was

not available. (Increase the length of the /BOPRXX/ COMMON block.)

COMMON Areas: RESCX, BOPRXX

FORTRAN Data Files: #6 (output/formatted)

Required Subroutines: MKMARQ

Method: Determine whether sufficient memory is available and call on the iterative nonlinear minimization subroutine MKMARQ.

INT2S - Subroutine Description

Function: To find a sequence of points on the curve of intersection of two surfaces.

Entry Point: INT2S

Calling Sequence: CALL INT2S(IRESFN,FAR,ITOL,POINTS,IMULT,IFAIL, NPOINT,ICOORD,KPLANE,CORVAL)

Calling Arguments:

Name (Attributes)	Contents
IRESFN (unused)	
FAR (real/input)	Control parameter for iterative nonlinear minimization subroutine; selects how two methods used are to be mixed. In absence of any special preference, should be set at 1.0.
ITOL (integer/input)	Control parameter for iterative nonlinear minimization subroutine. A solution is considered to have been reached if either of the following occurs: (1) all functional values $<10^{-ITOL}$ (2) independent variables all agree to ITOL significant figures in any two consecutive iterations.
POINTS (real/array/output)	Polygon of points on the curve of intersection. Dimensioned POINTS(3,NPOINT).
IMULT (unused)	
IFAIL (integer/output)	

Value	Meaning
0	Curve of intersection successfully found.
1000i+j	No curve of intersection has been found; see "Remarks".

NPOINT (integer/input)	Number of points for describing curve of intersection.
ICOORD (integer/input)	Value — Meaning 0 — Do not substitute a coordinate plane for the second surface. 1 — Do substitute a coordinate plane for the second surface.
KPLANE (integer/input)	Descriptor for coordinate plane being substituted for one of the surfaces; used only if ICOORD=1. 1 = Plane of constant x 2 = Plane of constant y 3 = Plane of constant z
CORVAL (real/input)	Descriptor for coordinate plane being substituted for one of the surfaces; used only if ICOORD=1; value of constant coordinate.

COMMON Areas: BOND, BOPRXX, RESCX, SEL, STCON, TIE

FORTRAN Data Files: None

Required Subroutines: ABC, EFF, GINTR, OUTRNG, RF2SS, RF2ST

Method: Iterative nonlinear minimization scheme.

Remarks:

(1) Regarding IFAIL: If i is 1, 2, or 3, failure occurred during attempt to lay out a curve of intersection between two points. If i = 1: failed to converge; if i = 2, 3: point out of range on one of the surfaces; if i = 4: problem is inadmissible. j is always 1 for i = 2, 3, or 4. If i = 1, j indicates outcome from iterative solver as follows: If j = 1: iterative solution converged with all residuals less than a specified tolerance; if j = 2: iterative solution converged to a stationary point; if j = 3: iterative solution failed to converge; and if j = 4: insufficient memory for iterative solver.

(2) Regarding the following COMMON block: COMMON/BOPRXX/LBOPR1, LBOPR2, LBUFF, JBUFF, BUFF(1), MBOPR(8,9), BUFFX(87)

The information described below must be entered into this

block before INT2S is called. The main function of this block is to provide a storage area for the definitions of the two surfaces. The main subdivisions of this block are described in Table 2. Array MBOPR contains data about the two surfaces. The surface data region is described in Table 3. Array BUFF must contain the grid of defining points for each of the two surfaces. These grids may be entered into BUFF consecutively. BUFF is also a working storage area.

TABLE 2 - THE MAIN SUBDIVISIONS OF COMMON BLOCK BOPRXX

Name	Description
LBOPR1, LBOPR2	Dimensions of MBOPR.
LBUFF	Length of BUFF.
JBUFF	Next available location in BUFF.
BUFF	Dummy working storage area to permit both MBOPR and BUFFX to be assigned dynamically.
MBOPR	Data index containing: surface data pointers, spatial guess*, and transformation matrix data pointers*.
BUFFX	Working storage area; includes surface defining points and transformation matrices*.

* This information is not required for subroutine INT2S.

TABLE 3 - SURFACE DATA AREA OF MBOPR

Location*	Description
MBOPR(2,I)	Surface type; see Table 1
MBOPR(3,I)	Pointer to first word of surface definition in BUFF.
MBOPR(4,I),MBOPR(5,I)	Dimensions of defining mesh for surface.

* I is 1, 2, or 3 and denotes which surface is being described.

INT2SX - Subroutine Description

Function: To calculate the location of points on the curve of intersection of two B-spline surfaces.

Entry Point: INT2SX

Calling Sequence: CALL INT2SX(P1,M1,N1,ITYP1,P2,M2,N2,ITYP2,
ICOORD,KPLANE,CORVAL,POINTS,NPOINT,ITOL,
IMOVE,IFAIL)

Calling Arguments:

Name (Attributes)	Contents
Pi (real/array/input)	Mesh defining the i^{th} B-spline surface. Dimensioned Pi(3,Mi,Ni).
Mi,Ni (integer/input)	Dimensions of array Pi.
ITYPi (integer/input)	Code word indicating the type of B-spline surface stored in Pi (see Table 1).
ICOORD (integer/input)	Flag word. If ICOORD=1, intersection problem will be solved using a specified coordinate plane for the second surface. If ICOORD=0, an arbitrary surface is assumed to be present as the second surface.
KPLANE (integer/input)	Code word indicating orientation of coordinate plane, if used. If KPLANE=1, an X-plane will be used. If KPLANE=2, a Y-plane will be used. If KPLANE=3, a Z-plane will be used.
CORVAL (real/input)	Coordinate value for plane used.
POINTS (real/array/output)	Array of points calculated on the curve of intersection. Dimensioned POINTS(3,NPOINT).

NPOINT (integer/input)	Number of points to be calculated on the curve of intersection. Must be ≥ 2.
ITOL (integer/input)	Convergence tolerance for iterative solver. Procedure will terminate when an additional iteration does not change the ITOL most significant decimal digits of the solution.
IMOVE (integer/input)	Flag word. If IMOVE=1, surface meshes will be transferred to COMMON block /BOPRXX/ prior to solution. If IMOVE=0, only the memory addresses of the surface meshes will be stored in COMMON block /BOPRXX/.
IFAIL (integer/output)	Flag word. A value of IFAIL which is less than 1000 indicates a successful completion of the procedure. A value which is 1000 or greater indicates a failure condition (see description of subroutine INT2S for details).

COMMON Areas:

Name	Function
BOPRXX	Communication with subroutine INT2S (see description of subroutine INT2S for structure).

FORTRAN Data Files: None

Required Subroutines: INT2S, LOCF

Method: This subroutine is a machine dependent driver for solving the two-surface intersection problem outside of the G-PRIME environment. Operands are moved to COMMON block /BOPRXX/ and then subroutine INT2S is called to calculate the solution.

Remarks:

(1) Subroutine LOCF, used to obtain the memory addresses of the surface operands, is machine dependent. Use of this subroutine may be avoided by setting IMOVE to 1 and increasing the length of COMMON block /BOPRXX/ to accommodate copies of the surface operands.

INT3S - Subroutine Description

Function: To find a point of intersection of three surfaces.

Entry Point: INT3S

Calling Sequence: CALL INT3S(IRESFN,FAR,ITOL,PARVAL,XYZ,IMULT,IFAIL)

Calling Arguments:

Name (Attributes)	Contents
IRESFN (integer/input)	Flag word. If IRESFN=1 or IRESFN=2 solve, starting with parameter guesses, using residual functions RF3S01 or RF3S02, respectively. If IRESFN=5 or IRESFN=6 locate, then solve using residual functions RF3S01 or RF3S02, respectively.
FAR (real/input)	Control parameter for iterative nonlinear minimization subroutine.
ITOL (integer/input)	Control parameter for iterative nonlinear minimization subroutine used for "solve". A solution is considered to have been reached if either of the following occurs: (1) all functional values $<10^{-ITOL}$ (2) independent variables all agree to ITOL significant figures in any two consecutive iterations.
PARVAL (real/array/input/output)	Independent variable for nonlinear solver. Dimensioned PARVAL(6). Input: initial guesses for the parameters s_1,t_1,s_2,t_2,s_3, and t_3. If doing "Locate-Solve", use 0.5 for all six values. If doing "Solve", use best estimate of parameter values at point of intersection.

	Output: if a solution has been found, PARVAL will contain the parameter values s_1, t_1, s_2, t_2, s_3, and t_3, associated with the point of intersection of the three surfaces.
XYZ (real/array/output)	Coordinates of point of intersection of the three surfaces, if found. Dimensioned XYZ(3).
IMULT (integer/input)	Parameter which determines whether or not the program is to make any additional effort at finding a solution when the first try fails. (See "Remarks".) If IMULT=1, make additional effort. If IMULT>1, do not make additional effort.
IFAIL (integer/output)	If IFAIL=0, an intersection point has been found. If IFAIL=1000i+j, an intersection point has not been found; see "Remarks".

COMMON Areas: ARB, BOPRXX, MULT, RESCX

FORTRAN Data Files: None

Required Subroutines: CHKR, EFF, GINTR, MEK, RF3S01, RF3S02, XLOCI1, XLOCI2, XLOCI3

Method: Iterative nonlinear minimization scheme.

Remarks:

(1) If initial guess is spatial (x,y,z), it must be stored in MBOPR(3,4), MBOPR(4,4), MBOPR(5,4) in COMMON block /BOPRXX/ before calling the subroutine. If initial guess is parametric $(s_1, t_1, s_2, t_2, s_3, t_3)$, it must be put in an array which will become PARVAL(1) through PARVAL(6) when this subroutine is called.

(2) Regarding IMULT: When IMULT=1, the "additional effort" made for spatial initial guesses and for parametric initial guesses is quite

different. When the initial guess is parametric, the "additional effort" consists of substituting various sets of parametric initial guesses $(s_1,t_1,s_2,t_2,s_3,t_3)$ for the parametric initial guess supplied by the user. When the initial guess is spatial, the spatial initial guess supplied by the user remains the same; the "additional effort" is made during the "locate" process for one of the three surfaces and consists of trying a new initial guess for the parameters (s_i,t_i) for the surface on which the "locate" is being attempted. No "additional effort" is made during the "solve" phase.

(3) Regarding IFAIL: Generally, i indicates the place in INT3S at which failure occurred; j indicates the outcome of the most recent call to GINTR (which uses the iterative nonlinear minimization process).

i	Meaning
1	Failed during "solve"
2	Failed during "locate" for first surface
3	Failed during "locate" for second surface
4	Failed during "locate" for third surface
5	Failed during "solve" of "locate-solve"

j	Meaning
1	Iterative solution converged with all residuals less than 10^{-ITOL}. Usually indicates that the solution point is outside standard surface definition limits.
2	Iterative solution converged to a stationary point.
3	Iterative solution failed to converge.
4	Insufficient memory for iterative solver.

(4) Regarding the following COMMON block: COMMON/BOPRXX/LBOPR1, LBOPR2,LBUFF,JBUFF,BUFF(1),MBOPR(8,9),BUFFX(87)

The information described below must be entered into this block before INT3S is called. The main purpose of this block is to provide a storage area to contain the definitions of the three surfaces. The main subdivisions of this block are described in Table 2. Array MBOPR contains three main storage areas:

(1) data about the three surfaces

(2) a spatial guess for the intersection point

(3) data about the transformation matrices (which are required whenever the piecewise-linear approximation for the three surfaces is used).

The surface data area is described in Table 3. The spatial guess must be put into MBOPR(3,4), MBOPR(4,4), and MBOPR(5,4); however, the spatial guess must be entered into these locations as real numbers. The transformation matrix data area is described in Table 4.

TABLE 4 - TRANSFORMATION MATRIX DATA AREA OF MBOPR

Location*	Description
MBOPR(3,I)	Pointer to first word of transformation matrix in BUFF.
MBOPR(4,I)	Number of defining points.
MBOPR(5,I)	Number of output points.

* I is 8 = open, 9 = closed.

Array BUFF must contain:

(1) the grid of defining points for each of the three surfaces, and

(2) the transformation matrices.

The transformation matrices may be generated by calling subroutine BSCMAT as follows:

CALL BSCMAT(BUFF(I),J,K,L)

where I Pointer to first word of transformation matrix in BUFF, i.e., the location in BUFF where the user wants the transformation matrix to begin.

J Number of defining points

K Number of output points

L 0 = Open; 1 = Closed

Blocks of data should be entered into BUFF consecutively; JBUFF should be updated each time a new block of data is entered into BUFF. INT3S requires 87 words of working storage in the BUFFX array.

INT3SX - Subroutine Description

Function: To calculate the location of a point of intersection of three B-spline surfaces.

Entry Point: INT3SX

Calling Sequence: CALL INT3SX(P1,M1,N1,ITYP1,P2,M2,N2,ITYP2,P3, M3,N3,ITYP3,SPOINT,PARVAL,XYZ,ITOL,LOCATE, IMOVE,IFAIL)

Calling Arguments:

Name (Attributes)	Contents
Pi (real/array/input)	Mesh defining the i^{th} B-spline surface. Dimensioned Pi(3,Mi,Ni).
Mi,Ni (integer/input)	Dimensons of array Pi.
ITYPi (integer/input)	Code word indicating the type of B-spline stored in Pi (see Table 1).
SPOINT (real/array/input)	Coordinates of spatial point close to the desired intersection point. Dimensioned SPOINT(3).
PARVAL (real/array/input/ output)	On input, parametric guesses of the location of the desired point on each of the surfaces. If unknown, values of .5 are a reasonable choice. On output, contains the parametric location of the intersection for each of the three surfaces. Dimensioned PARVAL(6).
XYZ (real/array/output)	Spatial coordinates of the intersection point. Dimensioned XYZ(3).
ITOL (integer/input)	Convergence tolerance for iterative solver. Procedure will terminate when an additional iteration does not change the ITOL most significant decimal digits of the solution.

LOCATE (integer/input)	Flag word. If LOCATE=1, coordinates in array SPOINT will be used to determine a new set of initial parameter guesses for iterative solving procedure. If LOCATE=0, parameter values in PARVAL will be used as initial guesses.
IMOVE (integer/input)	Flag word. If IMOVE=1, surface meshes will be transferred to COMMON block /BOPRXX/ prior to solution. If IMOVE=0, only the memory addresses of the surface meshes will be stored in COMMON block /BOPRXX/.
IFAIL (integer/output)	Flag word. A zero value of IFAIL indicates a successful completion of the procedure. All other values indicate a failure condition (see description of subroutine INT3S for details).

COMMON Areas:

Name	Function
BOPRXX	Communication with subroutine INT3S (see description of subroutine INT3S for structure).

FORTRAN Data Files: None

Required Subroutines: INT3S, LOCF

Method: This subroutine is a machine dependent driver for solving the three-surface intersection problem outside of the G-PRIME environment. Operands are moved to COMMON block /BOPRXX/ and then subroutine INT3S is called to calculate the solution.

Remarks:

(1) Subroutine LOCF, used to obtain the memory addresses of the surface operands, is machine dependent. Use of this subroutine may be

avoided by setting IMOVE to 1 and increasing the length of COMMON block /BOPRXX/ to accommodate copies of the surface operands.

MEK - Subroutine Description

Function: Select a new parameter guess for iterative solver.

Entry Point: MEK

Calling Sequence: CALL MEK(I,J,K,L,PARAM)

Calling Arguments:

Name (Attributes)	Contents
I (integer/input/output)	Indicates which parameter guess is currently being used.
J (integer/input)	Maximum number of parameter guesses allowed.
K (integer/input)	Indicates which surfaces are involved. 0 = all three surfaces i = Surface i (i = 1, 2, or 3)
L (integer/output)	0 = A new parameter guess was made; this implies that another solution-attempt may be made. 1 = No more parameter guesses are available; a new parameter guess was not made.
PARAM (real/array/input/output)	Parameters $(s_1,t_1,s_2,t_2,s_3,t_3)$. Dimensioned PARAM(2,3).

COMMON Areas: ARB, MULT

FORTRAN Data Files: None

Required Subroutines: None

Method: Select additional parameter guesses from table.

OUTRNG - Subroutine Description

Function: To process specific values for the parameters (s,t) for a surface as follows: (a) if parameter is "open", determine whether its value is "out of range"; (b) if parameter is "closed", bring its value back into range.

Entry Point: OUTRNG

Calling Sequence: X = OUTRNG(ITYPE,PARPAR,EPS)

Calling Arguments:

Name (Attributes)	Contents
OUTRNG (logical/output)	Status indicator. If .TRUE., at least one parameter value is out of range. If .FALSE., all parameter values are in the range [0,1].
ITYPE (integer/input)	Entity type; see Table 1.
PARPAR (real/array/input/output)	Surface parameters (s,t). Dimensioned PARPAR(2).
EPS (real/input)	Tolerance value used in "out-of-range" determination.

COMMON Areas: None

FORTRAN Data Files: None

Required Subroutines: None

Method: Comparison using specified tolerance.

Remarks:

(1) When a parameter is "out of range", it lies outside the interval [0,1]. The tolerance value, EPS, is used to expand the interval by a small amount.

RF2SS - Function Subroutine Description

Function: To compute the residual functions needed to solve for one point on a curve of intersection of two surfaces.

Entry Point: RF2SS

Calling Sequence: Z = RF2SS(X,I)

Calling Arguments:

Name (Attributes)	Contents
RF2SS (real/function/output)	Residual value.
X (real/array/input)	Parameters for the two surfaces; dimensioned X(3); X(1) and X(2) are the parameters (s and t) for the surface being considered in its entirety; X(3) is the variable parameter (t) for the other surface.
I (integer/input)	Indicates which of the three residual functions is to be computed.

Value	Meaning (See Remark 3)
1	$x_1(s_1,t_1) - x_2(s_0,t_2)$
2	$y_1(s_1,t_1) - y_2(s_0,t_2)$
3	$z_1(s_1,t_1) - z_2(s_0,t_2)$

COMMON Areas: SEL, STCON, RESCX, BOPRXX

FORTRAN Data Files: None

Required Subroutines: BSEVL1

Method: Compute residual functions from evaluation of spatial coordinates.

Remarks:

(1) Regarding COMMON block STCON: COMMON/STCON/SCON, TCON
SCON must contain s_0.

(2) Regarding COMMON block SEL: COMMON/SEL/KA, KB
KA denotes the surface (Surface 1 or Surface 2) which is to be considered in its entirety.

KB denotes the surface (Surface 1 or Surface 2) which is to be considered only in terms of a curve of constant s.

(3) s_1 and t_1 are the parameters for the surface which is being considered in its entirety.

s_0 (a constant) and t_2 are the parameters for the other surface.

The spatial coordinates for the surface which is being considered in its entirety are given by

$$x_1(s_1,t_1)$$

$$y_1(s_1,t_1)$$

$$z_1(s_1,t_1)$$

The spatial coordinates for the other surface are given by

$$x_2(s_0,t_2)$$

$$y_2(s_0,t_2)$$

$$z_2(s_0,t_2)$$

RF2ST - Function Subroutine Description

Function: To compute the residual functions needed to solve for one point on a curve of intersection of two surfaces.

Entry Point: RF2ST

Calling Sequence: Z = RF2ST(X,I)

Calling Arguments:

Name (Attributes)	Contents
RF2ST (real/function/output)	Residual value.
X (real/array/input)	Parameters for the two surfaces; dimensioned X(3); X(1) and X(2) are the parameters (s and t) for the surface which is being considered in its entirety; X(3) is the variable parameter (s) for the other surface.
I (integer/input)	Indicates which of the three residual functions is to be computed.

Value	Meaning (See Remark 3)
1	$x_1(s_1,t_1) - x_2(s_2,t_0)$
2	$y_1(s_1,t_1) - y_2(s_2,t_0)$
3	$z_1(s_1,t_1) - z_2(s_2,t_0)$

COMMON Areas: SEL, STCON, RESCX, BOPRXX

FORTRAN Data Files: None

Required Subroutines: BSEVL1

Method: Compute residual functions from evaluation of spatial coordinates.

Remarks:

(1) Regarding COMMON block STCON: COMMON/STCON/SCON, TCON
TCON must contain t_0.

(2) Regarding COMMON block SEL: COMMON/SEL/KA, KB
KA denotes the surface (Surface 1 or Surface 2) which is to be considered in its entirety.

KB denotes the surface (Surface 1 or Surface 2) which is to be considered only in terms of a curve of constant t.

(3) s_1 and t_1 are the parameters for the surface which is being considered in its entirety.

s_2 and t_0 (a constant) are the parameters for the other surface.

The spatial coordinates for the surface which is being considered in its entirety are given by

$$x_1(s_1,t_1)$$

$$y_1(s_1,t_1)$$

$$z_1(s_1,t_1)$$

The spatial coordinates for the other surface are given by

$$x_2(s_2,t_0)$$

$$y_2(s_2,t_0)$$

$$z_2(s_2,t_0)$$

RF3S01 - Function Subroutine Description

Function: To compute the spatial coordinates on three surfaces corresponding to a given set of parameters and then to compute the residual functions needed to solve for a point of intersection.

Entry Point: RF3S01

Calling Sequence: Z = RF3S01(X,I)

Calling Arguments:

Name (Attributes)	Contents
RF3S01 (real/function/output)	Residual value.
X (real/array/input)	Parametric coordinates $(s_1,t_1,s_2,t_2,s_3,t_3)$ for the three surfaces. Dimensioned X(6).
I (integer/input)	Indicates which of the six residual functions is to be computed.

Value	Meaning
1	$x_1(s_1,t_1) - x_2(s_2,t_2)$
2	$y_1(s_1,t_1) - y_2(s_2,t_2)$
3	$z_1(s_1,t_1) - z_2(s_2,t_2)$
4	$x_2(s_2,t_2) - x_3(s_3,t_3)$
5	$y_2(s_2,t_2) - y_3(s_3,t_3)$
6	$z_2(s_2,t_2) - z_3(s_3,t_3)$

where the spatial coordinates of a point on Surface i are given by $x_i(s_i,t_i)$, $y_i(s_i,t_i)$, and $z_i(s_i,t_i)$.

COMMON Areas: RESCX, BOPRXX

FORTRAN Data Files: None

Required Subroutines: BSEVL3

Method: Compute residual functions from evaluation of spatial coordinates.

RF3S02 - Subroutine Description

Function: To compute the spatial coordinates on three surfaces corresponding to a given set of parameters and then to compute the residual functions needed to solve for a point of intersection.

Entry Point: RF3S02

Calling Sequence: Z = RF3S02(X,I)

Calling Arguments:

Name (Attributes)	Contents
RF3S02 (real/function/output)	Residual value
X (real/array/input)	Parametric coordinates (s_1,t_1,s_2, t_2,s_3,t_3) for the three surfaces. Dimensioned X(6).
I (integer/input)	Indicates which of the six residual functions is to be computed.

Value	Meaning
1	$x_1(s_1,t_1) - x_2(s_2,t_2)$
2	$y_1(s_1,t_1) - y_2(s_2,t_2)$
3	$z_1(s_1,t_1) - z_2(s_2,t_2)$
4	$x_2(s_2,t_2) - x_3(s_3,t_3)$
5	$y_2(s_2,t_2) - y_3(s_3,t_3)$
6	$z_2(s_2,t_2) - z_3(s_3,t_3)$

where the spatial coordinates of a point on Surface i are given by $x_i(s_i,t_i)$, $y_i(s_i,t_i)$, and $z_i(s_i,t_i)$.

COMMON Areas: RESCX, BOPRXX

FORTRAN Data Files: None

Required Subroutines: BSEVL1

Method: Compute residual functions from evaluation of spatial coordinates.

STR - Subroutine Description

Function: To build a file of points for which (1) the boundary of Surface 1 intersects Surface 2, or (2) the boundary of Surface 2 intersects Surface 1.

Entry Point: STR

Calling Sequence: CALL STR(BPP,EPSIL)

Calling Arguments:

Name (Attributes)	Contents
BPP (real/array/input)	Point for which (1) the boundary of Surface 1 intersects Surface 2, or (2) the boundary of Surface 2 intersects Surface 1; given in terms of parameters (s_1,t_1,s_2,t_2). Dimensioned BPP(2,2).
EPSIL (real/input)	Tolerance value used for checking whether a point is already in the file; if all four parameters agree to within the specified tolerance value with the parameters for a point already in the file, no new point is entered in the file.

COMMON Areas: BOPRXX, TIE, BOND

FORTRAN Data Files: None

Required Subroutines: None

Method: Compare the newly found point to the points already in the file. If the new point is different from all the points already in the file, enter the new point in the file.

Remarks:

(1) Regarding COMMON block TIE: COMMON/TIE/AB(2,2,8)
Array AB contains the aforementioned file of points.

XLOCI1 - Function Subroutine Description

Function: To compute the residual functions needed to perform the "locate" process (i.e., finding the point closest to the spatial guess).

Entry Point: XLOCI1

Calling Sequence: Z = XLOCI1(X,I)

Calling Arguments:

Name (Attributes)	Contents
XLOCI1 (real/function/output)	Residual value
X (real/array/input)	Parameters (s,t) for the surface on which the "locate" is being attempted. Dimensioned X(2).
I (integer/input)	Coordinate direction currently being called for: 1 = x 2 = y 3 = z

COMMON Areas: BOPRXX, RESCX

FORTRAN Data Files: None

Required Subroutines: BSEVL1

Method: Compute difference (in coordinate direction specified) between point on surface and spatial guess.

Remarks:

(1) This function has three entry points. When the "locate" process is to be performed for Surface i, entry point XLOCIi should be used (i is 1, 2, or 3).

UTILITY SUBROUTINES

BCKSUB - Subroutine Description

Function: To perform a vector forward backward substitution using a Cholesky factor matrix.

Entry Point: BCKSUB

Calling Sequence: CALL BCKSUB(S,X,B,N)

Calling Arguments:

Name (Attributes)	Contents
S (real/array/input)	Lower triangular factor matrix, stored: S(1,1),S(2,1),..., S(N,N-1),S(N,N). Length is (N(N+1)/2).
X (real/ array/output)	Solution vector. Length N.
B (real/array/input)	Right-hand side vector. Length N.
N (integer/input)	Order of matrix S.

COMMON Areas: None

FORTRAN Data Files: None

Required Subroutines: None

Method: The equation $[S][S^T]\underset{\sim}{X} = \underset{\sim}{B}$ is solved for $\underset{\sim}{X}$ by first solving $[S]\underset{\sim}{Y} = \underset{\sim}{B}$ for $\underset{\sim}{Y}$ and then solving $[S^T]\underset{\sim}{X} = \underset{\sim}{Y}$ for $\underset{\sim}{X}$, where [S] is a lower triangular Cholesky factor matrix.

Remarks:

(1) This subroutine is used with subroutine DECOMP to solve the equation $[A]\underset{\sim}{X} = \underset{\sim}{B}$. See description of subroutine DECOMP for details.

(2) The arrays B and X may overlap whenever B is not required for subsequent calculations.

BSMULT - Subroutine Description

Function: To compute the matrix product $[S^T][S]$, storing only the lower triangular portion of the result. [S] must be a B-spline coefficient matrix stored in packed form.

Entry Point: BSMULT

Calling Sequence: CALL BSMULT(CS,CS,M1,MORIG,IFS,STSLTR)

Calling Arguments:

Name (Attributes)	Contents
CS (mixed/array/input)	B-spline coefficient matrix in packed form. Dimensioned CS(5,MORIG). The array CS is required as the first argument for real references and as the second argument for integer references.
M1 (integer/input)	Row length of full [S] matrix.
MORIG (integer/input)	Column length of full [S] matrix.
IFS (integer/input)	Flag word. If zero, CS contains coefficients for a non-periodic or open B-spline function. If one, CS contains coefficients for a periodic or closed B-spline function.
STSLTR (real/array/output)	The product matrix stored in lower triangular form.

COMMON Areas: None

FORTRAN Data Files: None

Required Subroutines: None

Method: Each row of CS is unpacked and its contributions to the product are accumulated in the array STSLTR.

Remarks:

(1) The form of the product array STSLTR is compatible with the requirements of the Cholesky decomposition subroutine, DECOMP.

DECOMP - Subroutine Description

Function: To perform Cholesky decomposition of a real, symmetric matrix when diagonal and lower triangle of matrix are given.

Entry Point: DECOMP

Calling Sequence: CALL DECOMP(A,S,N,IOK)

Calling Arguments:

Name (Attributes)	Contents
A (real/array/input)	Lower triangle of matrix to be decomposed; stored A(1,1),A(2,1), A(2,2),A(3,1),...,A(N,N-1),A(N,N). Length of array is ((N+1)N)/2.
S (real/array/output)	Lower triangle of Cholesky factor matrix; same order of storage and length as A.
N (integer/input)	Order of matrix A.
IOK (integer/output)	Flag word. A value of -1 indicates that decomposition was terminated because the matrix A was singular.

COMMON Areas: None

FORTRAN Data Files: None

Required Subroutines: None

Method: Single precision Cholesky matrix decomposition is performed, yielding the lower triangular factor matrix [S] in the equation: $[A] = [S][S^T]$.

Remarks:

(1) This subroutine can be used with subroutine BCKSUB to solve the equation $[A]\underset{\sim}{X} = \underset{\sim}{B}$. DECOMP is called to factor [A] which permits the problem to be rewritten $[S][S^T]\underset{\sim}{X} = \underset{\sim}{B}$. BCKSUB is called to solve $[S]\underset{\sim}{Y} = \underset{\sim}{B}$ for $\underset{\sim}{Y}$ and then to solve $[S^T]\underset{\sim}{X} = \underset{\sim}{Y}$ for $\underset{\sim}{X}$.

(2) [A] must be a real, symmetric matrix.

(3) The arrays A and S may overlap whenever A is not required for subsequent calculations.

(4) The form of the factor matrix [S] is compatible with the requirements of subroutine BCKSUB.

MKMARQ - Subroutine Description

Function: To find the minimum of the sum of squares of M functions of N variables.

Entry Point: MKMARQ

Calling Sequence: CALL MKMARQ(F,EPS,NSIG,M,N,C,X,FX,ITMAX,WA,IER)

Calling Arguments:

Name (Attributes)	Contents
F (real/function/external)	Function to compute functions to be minimized, which will be called with two arguments, F(X,I), where X is the vector of N variables and I is an index to select one of the M functions.
EPS (real/input)	First stopping criterion for iterative method. If F(X,I)<EPS for all I, convergence is assumed.
NSIG (integer/input)	Second stopping criterion for iterative method. If for two successive iterations the vector of variables X is the same to NSIG significant digits, convergence is assumed.
M (integer/input)	Number of functions to be evaluated.
N (integer/input)	Number of independent variables in vector X.
C (real/input)	Algorithm modifying parameter; usually set to 1.0. See Brown[5] for details.
X (real/array/input/output)	Vector of variables; length N. Initial values for starting iteration must be stored in this array by calling routine. Contains values of variables which minimize functions on output.

FX (real/array/output)	Vector containing the values of the M functions evaluated for the final set of variables in X.
ITMAX (integer/input/output)	Third stopping criterion and iteration counter. If more than ITMAX iterations are required for convergence, the subroutine assumes that convergence can not be achieved. On output, ITMAX contains the number of iterations actually performed.
WA (array/scratch)	A working storage area KORE words long, where KORE = N(N+1)+6N+N*M+3M.
IER (integer/output)	Flag word. A value of zero indicates that subroutine has stopped by meeting first stopping criterion. A value of one indicates that second stopping criterion has been met. If greater than one, subroutine has failed.

COMMON Areas: None

FORTRAN Data Files: None

Required Subroutines: DECOMP, BCKSUB

Method: This is an implementation of Brown's Algorithm[5] which has been tailored for use with the B-spline intersection subroutines.

Remarks:

(1) Poor results have been observed when this routine has been initiated with zero or near zero independent variable vector.

(2) This implementation has evolved from a similar program called ZXMARQ which was marketed at one time by International Mathematical and Statistical Libraries, Inc. of Houston, Texas (IMSL). In its present form MKMARQ differs from the IMSL program in a number of ways and although the

calling sequences appear to be the same, the programs can not be interchanged directly.

MULMAT - Subroutine Description

Function: To compute the matrix product C = A x B for matrices which may have multidimensional elements.

Entry Point: MULMAT

Calling Sequence: CALL MULMAT(A,B,C,IAT,IBT,KAD,KAR,KAC,IAD,KBD, KBR,KBC,IBD,KCD,KCR,KCC,ICD)

Calling Arguments:

Name (Attributes)	Contents
A (real/array/input)	First matrix factor. Dimensioned A(KAD,KAR,KAC).
B (real/array/input)	Second matrix factor. Dimensioned B(KBD,KBR,KBC).
C (real/array/output)	Product matrix. Dimensioned C(KCD,KCR, KCC).
IAT (integer/input)	Flag word. A non-zero value indicates the matrix A is to be transposed prior to multiplication.
IBT (integer/input)	Flag word. A non-zero value indicates that the matrix B is to be transposed prior to multiplication.
KAD,KAR,KAC (integer/ input)	Dimensions of the array A.
IAD (integer/input)	Index to select the component of the multidimensional elements of array A.
KBD,KBR,KBC (integer/ input)	Dimensions of the array B.
IBD (integer/input)	Index to select the component of the multidimensional elements of the array B.
KCD,KCR,KCC (integer/ input)	Dimensions of the array C.
ICD (integer/input)	Index indicating the component of the multidimensional elements of the array C.

COMMON Areas: None

FORTRAN Data Files: #6 (output/formatted)

Required Subroutines: None

Method: Use formula, $c_{ij} = \sum_{\text{all } k} a_{ik} b_{kj}$, selecting components and transposing as indicated.

Remarks:

(1) Subroutine will stop program with message if matrices are not conformable.

NOBSE3 - Subroutine Description

Function: Dummy subroutine to satisfy unused external references in subroutine INT3S and to prevent loading of subroutine BSEVL3 and function subroutine RF3S01.

Entry Point: NOBSE1

Calling Sequence: CALL NOBSE1

Calling Arguments: None

Entry Point: RF3S01

Calling Sequence: CALL RF3S01

Calling Arguments: None

Entry Point: BSEVL3

Calling Sequence: CALL BSEVL3

Calling Arguments: None

Entry Point: RF3S03

Calling Sequence: CALL RF3S03

Calling Arguments: None

Entry Point: CALL RF3S04

Calling Sequence: CALL RF3S04

Calling Arguments: None

COMMON Areas: None

FORTRAN Data Files: #6 (output/formatted)

Subroutines Required: None

Method: When this subroutine is explicitly loaded with subroutine INT3SX, the library loading of the unused subroutines BSEVL3 and RF3S01 will not be performed--conserving memory.

Remarks:

(1) Subroutine will stop program with message if entry points BSEVL3 and RF3S01 are actually called.

SAMPLE APPLICATION

PLOTTING B-SPLINE SURFACES

Although we, as the developers, feel that subroutines included in the G-PRIME Basic B-Spline Library are easy to use, an example of a typical application of this capability may be helpful in establishing a frame of reference for the potential user. The subroutine PLT3DS, listed in Figure 5, draws a plot of a B-spline surface. This subroutine is from the program BHULL, written for ship hull design, which is described in a forthcoming report. (BHULL also contains examples of data fitting and simple surface and volume integration using B-spline functions.)

PLT3DS graphically represents a B-spline surface giving the user the option of (1) having the boundary curves displayed (values of zero and one for the parameters s and t), (2) having curves of constant values of either parameter displayed, or (3) having curves of constant values of one parameter displayed and then having curves of constant values of the other parameter displayed as well.

The calling sequence for PLT3DS is:

```
CALL PLT3DS(A,B,C,MD,M1,N1,ISPERD,ITPERD,IFS,JFT,ICLEAR)
```

where the calling arguments are defined as follows:

Name (Attributes)	Contents
A (real/array/input)	Mesh defining a B-spline surface. Dimensioned A(MD,M1,N1).
B (array/scratch)	Working storage for PLT3DS. Dimensioned B(MD,KORE) where KORE=MAX0(M1,N1).
C (array/scratch)	Working storage for subroutine CVPLT. Dimensioned C(2,KORE+1).
MD,M1,N1 (integer/input)	Dimensions of array A.
ITYPE (integer/input)	Code word indicating type of B-spline surface (see Table 1).
IFS (integer/input)	Flag word. If set to one, curves are drawn by varying the parameter s for a uniformly spaced sequence of values of the parameter t. If set to zero,

```
      SUBROUTINE PLT3DS(A,B,C,MD,M1,N1,ITYPE,IFS,JFT,ICLEAR)
C
C     PLOT A 3-D SURFACE ...
C        SURFACE OUTLINES ARE ALWAYS PLOTTED.  IFS AND JFT CONTROL
C        THE PLOTTING OF T AND S CURVES RESPECTIVELY.
C
      DIMENSION A(MD,M1,N1), B(MD,1), C(2,1)
C
      MULP = 6
      XLIM = 0.
      KSCL = 1.
C
C     PLOT BOUNDARIES...
C
      IF (M1 .LT. 1  .OR.  N1 .LT. 1)  GO TO 100
      MPT = MULP * M1
      DELM = 1. / FLOAT(MPT)
      MPT = MPT + 1
      NPT = MULP * N1
      DELN = 1. / FLOAT(NPT)
      NPT = NPT + 1
      JCLEAR = ICLEAR
      IF (M1 .EQ. 1)  MPT = 1
      IF (N1 .EQ. 1)  NPT = 1
      IF (M1 .EQ. 1)  GO TO 50
      IF (N1 .EQ. 1)  GO TO 22
      IF (IFS .EQ. 1)  GO TO 11
      DO 10 K=1,2
      S = -DELM
      DO 5 I=1,MPT
      S = S + DELM
    5 CALL BSEVL1 (A,MD,M1,N1,S,FLOAT(K-1),ITYPE,B(1,I) )
      CALL CVPLT (B,MD,MPT,C)
   10 JCLEAR = 0
   11 IF (JFT .EQ. 1)  GO TO 22
      DO 20 K=1,2
      T = -DELN
      DO 15 I=1,NPT
      T = T + DELN
   15 CALL BSEVL1 (A,MD,M1,N1,FLOAT(K-1),T,ITYPE,B(1,I) )
      CALL CVPLT (B,MD,NPT,C)
   20 JCLEAR = 0
   22 IF (IFS .EQ. 0)  GO TO 50
C
C
C     PLOT T-CURVES ...
C
      ISKP = 0
      T = -DELN
      DO 30 K=1,NPT
      T = T + DELN
      IF (ISKP .NE. 0)  GO TO 30
      S = -DELM
      DO 25 I=1,MPT
      S = S + DELM
   25 CALL BSEVL1 (A,MD,M1,N1,S,T,ITYPE,B(1,I) )
      CALL CVPLT (B,MD,MPT,C)
      JCLEAR = 0
   30 ISKP = MOD(ISKP+1,3)
   50 IF (JFT .EQ. 0)  GO TO 100
      IF (M1 .EQ. 1)  GO TO 100
C
C
C     PLOT S-CURVES...
C
      ISKP = 0
      S = -DELM
      DO 70 K=1,MPT
      S = S + DELM
      IF (ISKP .NE. 0)  GO TO 70
      T = -DELN
      DO 60 I=1,NPT
      T = T + DELN
   60 CALL BSEVL1 (A,MD,M1,N1,S,T,ITYPE,B(1,I) )
      CALL CVPLT (B,MD,NPT,C)
      JCLEAR = 0
   70 ISKP = MOD(ISKP+1,3)
  100 CONTINUE
      RETURN
C
      END
```

Figure 5 - Subroutine PLT3DS

Name (Attributes)	Contents
	only the curves t=0 and t=1 are drawn.
JFT (integer/input)	Flag word. If set to one, curves are drawn by varying the parameter t for a uniformly spaced sequence of values for the parameter s. If set to zero, only the curves s=0 and s=1 are drawn.
ICLEAR (not used)	

Curves are drawn as polygons with 6*M1+1 or 6*N1+1 points which lie on the given B-spline surface. The four major processing steps of this subroutine are:

(1) The "DO 10 loop" draws curves for t=0 and t=1.

(2) The "DO 20 loop" draws curves for s=0 and s=1.

(3) The "DO 30 loop" draws curves for t=1./(6*N1+1), 3./(6*N1+1),...,1.

(4) The "DO 70 loop" draws curves for s=1./(6*M1+1), 3./(6*M1+1),...,1.

Within each of the above steps, the evaluation subroutine BSEVL1 is called repeatedly to build the "curve polygon" that is to be plotted. That polygon, stored in the array B, is then passed to subroutine CVPLT for display.

Figure 6 shows three views of a B-spline surface as drawn by PLT3DS with IFS=1 and JFT=0; Figure 7 shows the same surface drawn with IFS=1 and JFT=1; and in Figures 8 and 9 the outlines of the surfaces have been drawn by PLT3DS with IFS=0 and JFT=0. The intersection curves and various symbols have been added by other subroutines.

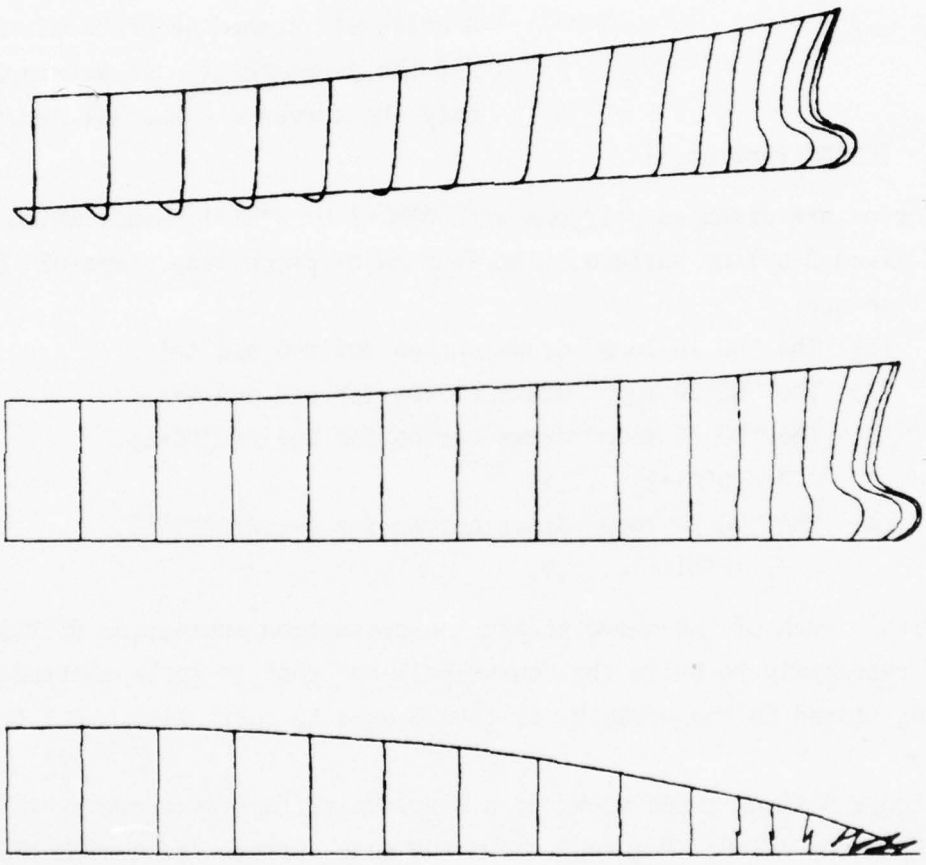


Figure 6 - Views of a B-Spline Surface Drawn by
Subroutine PLT3DS with IFS=1 and JFT=0

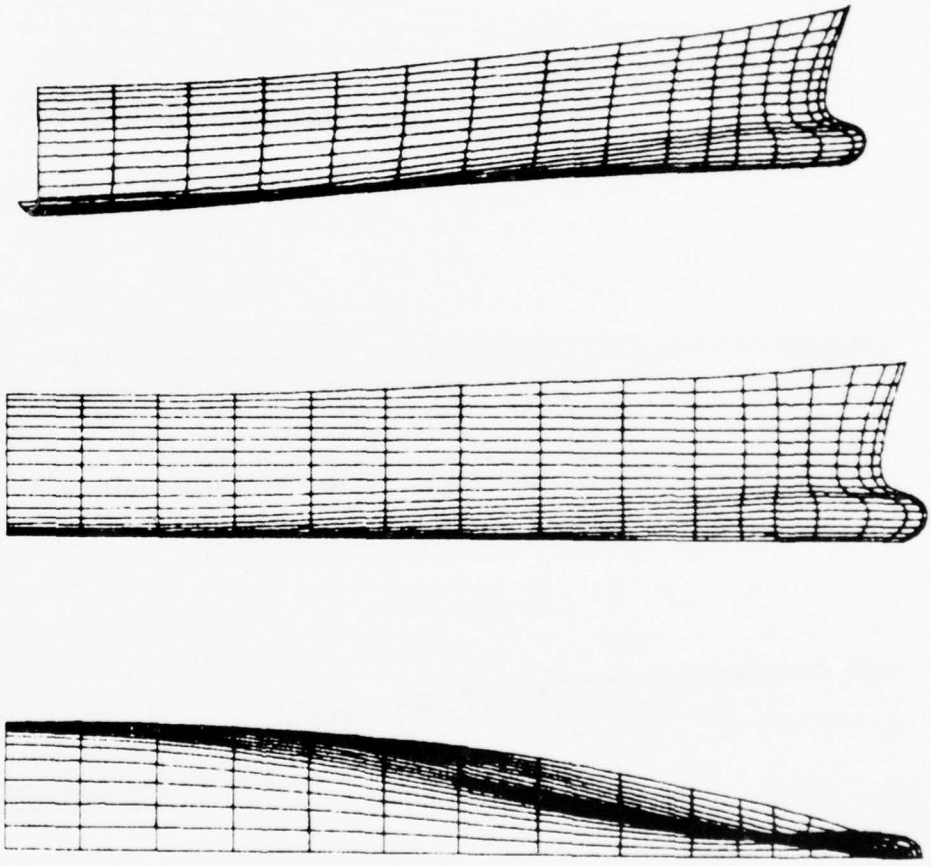


Figure 7 - Views of a B-Spline Surface Drawn by Subroutine PLT3DS with IFS=1 and JFT=1

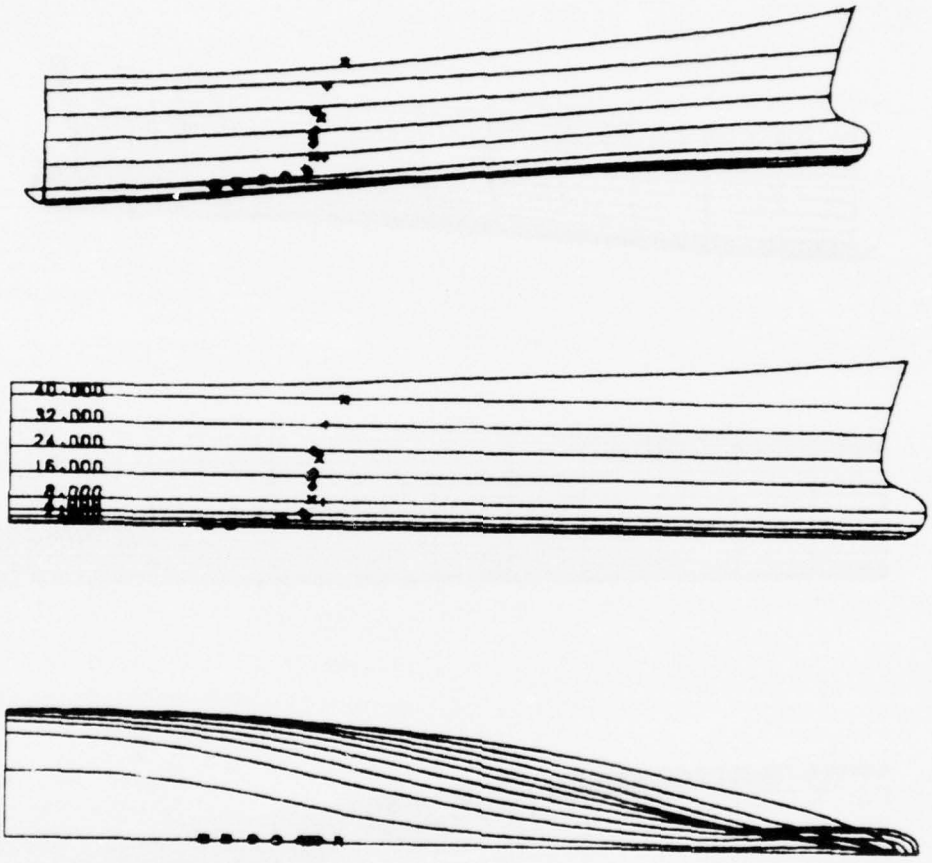


Figure 8 - B-Spline Surface Outlines Drawn by
Subroutine PLT3DS with IFS=0 and JFT=0

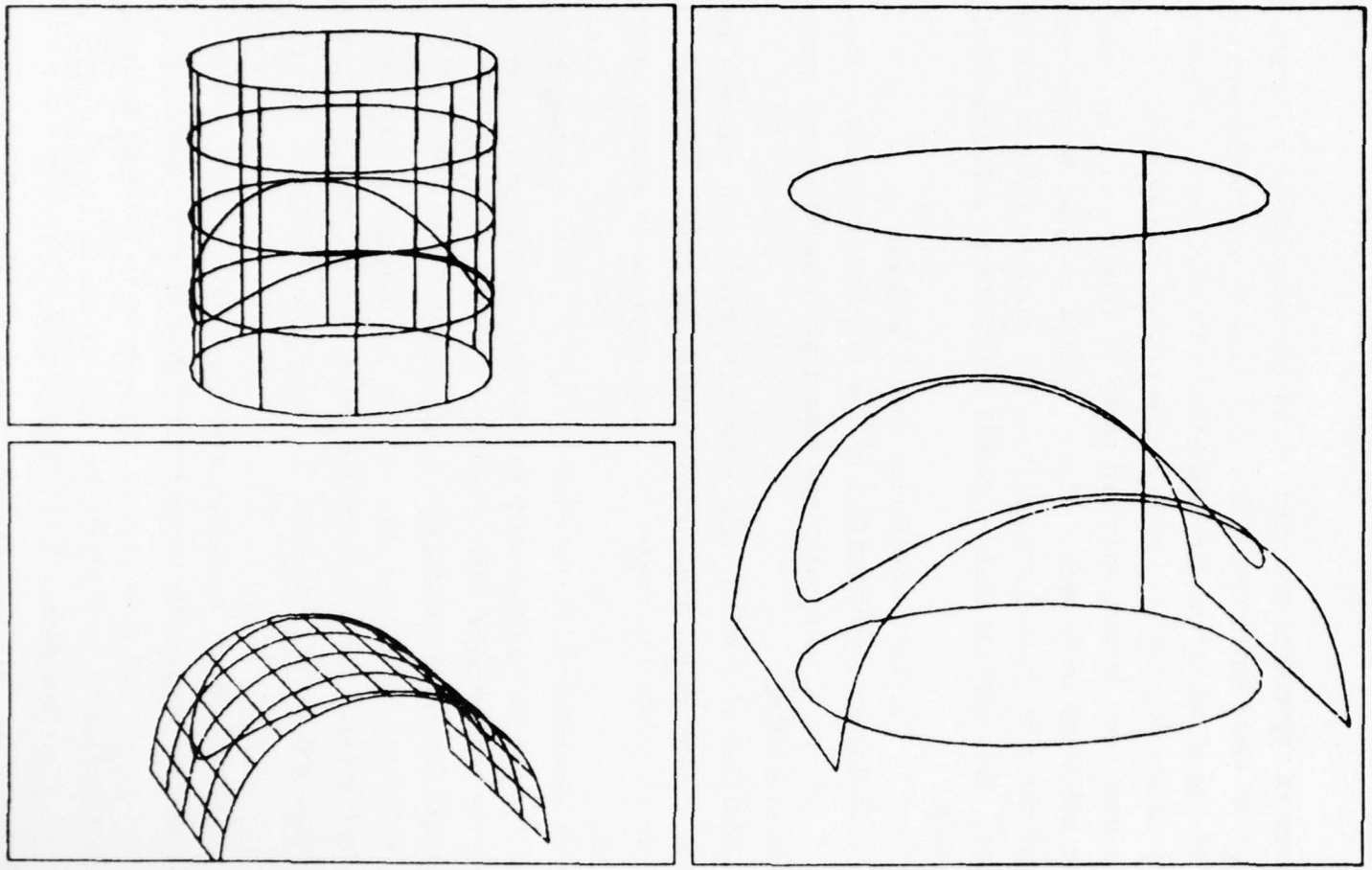


Figure 9 - Views of Intersecting B-Spline Surfaces Drawn by Subroutine PLT3DS

RECOMMENDED EXTENSIONS

This report represents one step toward the creation of a library of B-spline mathematical subroutines that will satisfy our requirements and the needs of the naval scientific community. Our work in this field is continually uncovering improved methods and new approaches to the problems we have been dealing with and there is always a list of modifications and additions to be made. Also, for lack of time, we have passed over some of the more general capabilities for which we have no current applications. This section lists possible additions to a future edition of the library.

Additional evaluation capabilities should include:

- Calculation of derivatives of the B-spline functions. This would permit the use of more efficient numerical integration and nonlinear minimization algorithms.
- Addition of a BSEVL2-like subroutine which will return a set of points along a curve rather than just one point per call. This type of subroutine could reduce the computation time required to display curves and surfaces.
- An alternate surface definition using a mesh of triangles as well as the current rectangular mesh definition. Local surface changes are more easily accomplished with a surface defined by a triangular mesh.

Additional fitting capabilities should include:

- A fitting capability which admits constraints on selected points or derivatives. This feature appears to be necessary for fitting data with cusps and for fitting data that must mesh exactly along certain boundaries.
- An option to use a uniform B-spline basis for the fitting of open curves and surfaces. The current practice for open curves and surfaces involves the use of a uniform basis which is modified to force the B-spline function to interpolate the ends of the defining polygon and the corners of a surface mesh. This practice also constrains the B-spline function to have zero curvature at those extremes. The uniform basis, without modification, will permit arbitrary curvature at the ends, which

may be more appropriate for fitting certain data.

- An option to save the least-squares transformation matrices from one fitting application so that they would not have to be recalculated for a subsequent application. Although it is evident from Equations (7) and (11) that the matrices [U] and [W] need to be computed only once for all problems of the same size, the current implementation does not take advantage of this property.

- An option to have the fitting subroutines compute the RMS error in the fit of a B-spline function to a given set of data.

Additional intersection capabilities should include:

- Calculation of the points of intersection of two curves, and the intersection of a curve and a surface. These calculations are specializations of the procedures used for finding the point of intersection of three surfaces.

- Enhancement of the procedures for finding the curve of intersection of two surfaces. The current two-surface intersection capability is not as comprehensive as we require. We hope to remove most of the restrictions currently imposed and to allow the user to select among multiple curves of intersection when they occur.

- A separate locating capability. The intersection subroutines now contain code which computes the location of the point on a B-spline curve or surface that is closest to a given spatial point. This locating capability should be made directly accessible to the user.

- Modification of the nonlinear solver to use derivatives of the B-spline functions which have been calculated by the evaluation subroutines. This should improve the overall efficiency of the nonlinear solution process, since greater accuracy can be obtained with no increase in the amount of computation.

- Include curve and surface operands as calling arguments of the intersection subroutines rather than using COMMON block /BOPRXX/ for the passing of operands. This practice, which is a hold-over from earlier iterative solution procedures, is sometimes confusing to the programmer and can now be eliminated.

ACKNOWLEDGMENTS

The authors wish to thank Dr. Feodor Theilheimer for many helpful conversations and his independent verification of the basic evaluation subroutines. We also wish to thank Donald A. Gignac who wrote the Cholesky decomposition subroutines DECOMP and BCKSUB and Suzanne Wybraniec who has been the program librarian for this project.

REFERENCES

1. Schoenberg, I.J., "Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions," Quart. Appl. Math., vol. 4 (1946), pp. 45-99; 112-141.

2. Riesenfeld, R., "Application of B-Spline Approximation to Geometric Problems of Computer-Aided Design," University of Utah Computer Science Report UTEC-CSC-73-126.

3. Bézier, P., Numerical Control - Mathematics and Applications (translated by A.R. Forrest). London: John Wiley and Sons, 1972.

4. de Boor, C., "On Calculating with B-Splines," J. Approx. Theory, vol. 6 (1972), pp. 50-62.

5. Brown, K.M., "Derivative Free Analogues of the Levenberg-Marquardt and Gauss Algorithms for Nonlinear Least Squares Approximations," Numerische Mathematik, vol. 18 (1972), pp. 289-297.

INITIAL DISTRIBUTION

Copies	
2	HARRY DIAMOND LABS 1 DRXDO-TI 1 DRXDO-NP
1	USA ABERDEEN PROVING GROUND 1 E. QUIGLEY
1	USA PICATINNY ARSENAL 1 W. BOLTE
1	NRL 1 R. PERLUT
1	DNL
2	USNA 1 DEPT MATH 1 LIB
1	NAVPGSCOL LIB
1	NROTC & NAVADMINU, MIT
1	NAVWARCOL
1	NSWC WHITE OAK 1 R.J. EDWARDS
1	NUSC NPTLAB 1 R. MESSIER
1	NUSC NLONLAB 1 A. CARLSON
2	NAVAIR 1 H. ANDREWS 1 G. HAND
1	NAVSHIPYD BREM/LIB
1	NAVSHIPYD CHASN/LIB
1	NAVSHIPYD MARE/LIB
1	NAVSHIPYD NORVA/LIB
1	NAVSHIPYD PEARL/LIB
1	NAVSHIPYD PHILA/LIB
1	NAVSHIPYD PTSMH/LIB

Copies	
9	NAVSEC 1 R. KELTIE 1 J. CLAFFEY 1 A.L. FULLER 1 M. AUGHEY 1 W.E. DIETRICH 1 R.S. JOHNSON 1 P.M. PALERMO 1 D. BILLINGSLEY 1 R.G. KEANE, JR.
12	DDC
1	AFWL KIRTLAND AFB 1 CAPT J. HANSEN
4	AFFDL 1 L. BERNIER 1 P. POURIER 1 J. JOHNSON 1 J. FOLCK
2	NASA GODDARD SFC 1 J. MASON 1 L.A. SCHMID
1	BOSTON UNIV Computing Center 111 Cummington Street Boston, MASS 02215 ATTN: Caroline Wardle
1	BATTELLE COLUMBUS LABS 505 King Avenue Columbus, Ohio 43201 ATTN: Gene Hulbert
1	GOODYEAR TIRE & RUBBER CO Dept. 100A 1144 East Market Street Akron, Ohio 44316 ATTN: David Twellman
1	JOHN DEERE, INC Dubuque Works, Dept. 614 Dubuque, IOWA 52001 ATTN: Jose Nazario

Copies

1 PDA, INC
1740 Garry Ave., Suite 201
Santa Ana, CA 92705
ATTN: E.L. Stanton

1 SPERRY SUPPORT SERVICES
716 Arcadia Circle
Huntsville, ALA 35801
ATTN: R. Schmitz

1 US STEEL RES CEN
125 Jamison Lane
Monroeville, PA 15146
ATTN: G.J. Hutchins

CENTER DISTRIBUTION

Copies	Code		Copies	Code	
1	1153	O'NEILL, W.C.	1	1809.3	HARRIS, D.
1	1170	STEVENS, R.M.	1	1820	CAMARA, A.W.
1	1182	WACHNIK, Z.G.	1	1823	CHEN, R.
1	1524	LIN, W.C.	1	1824	BERKOWITZ, S.
1	1542	YIM, B.	1	1840	LUGT, H.J.
3	1552	GROVES, N.C.	2	1843	SCHOT, J.W.
		VON KERCZEK, C.			HAAS, M.
		WHITE, N.	2	1844	DHIR, S.K.
1	1556	NORTON, R.			ZARDA, R.
1	1568	COX, G.C.	50	1844	MCKEE, J.M.
1	1572	OCHI, M.D.	5	1844	KAZDEN, R.
1	1576	SMITH, W.E.	1	1850	CORIN, T.
1	1606	DE LOS SANTOS, S.	1	1853	THOMSON, B.
1	1630	FORD, A.G.	1	1858	SMITH, B.
1	1700	MURRAY, W.W.	1	189.1	TAYLOR, N.
1	1720.2	HOM, K.	1	1892.1	STRICKLAND, J.
1	1720.3	ROTH, P.	1	1892.2	SOMMER, D.
1	1720.6	ROCKWELL, R.D.	1	1903	BROOKS, J.
1	1730	STAVOVY, A.B.	1	1962	ZALOUMIS, A.
1	1730.1	CHIU, R.H.	1	1965	FEIT, D.
1	1730.2	NAPPI, N.	1	1966	CASPAR, J.
2	1730.5	ADAMCHAK, J.C.	1	2723	COBLENZ, R.
		MEYER, P.	1	2740	WANG, Y.F.
1	1740.5	WHANG, B.	30	5214.1	REPORTS DISTRIBUTION
1	1800	GLEISSNER, G.H.	1	522.1	LIBRARY (C)
1	1802.2	FRENKIEL, F.N.	1	522.2	LIBRARY (A)
1	1802.4	THEILHEIMER, F.			
1	1805	CUTHILL, E.H.			