NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER
Bethesda, Maryland 20034

SURFACE MODELING AND ANALYSIS SUBROUTINE PACKAGE

by

P. M. Rosenshine

COMPUTATION AND MATHEMATICS DEPARTMENT

RESEARCH AND DEVELOPMENT REPORT

April 1973

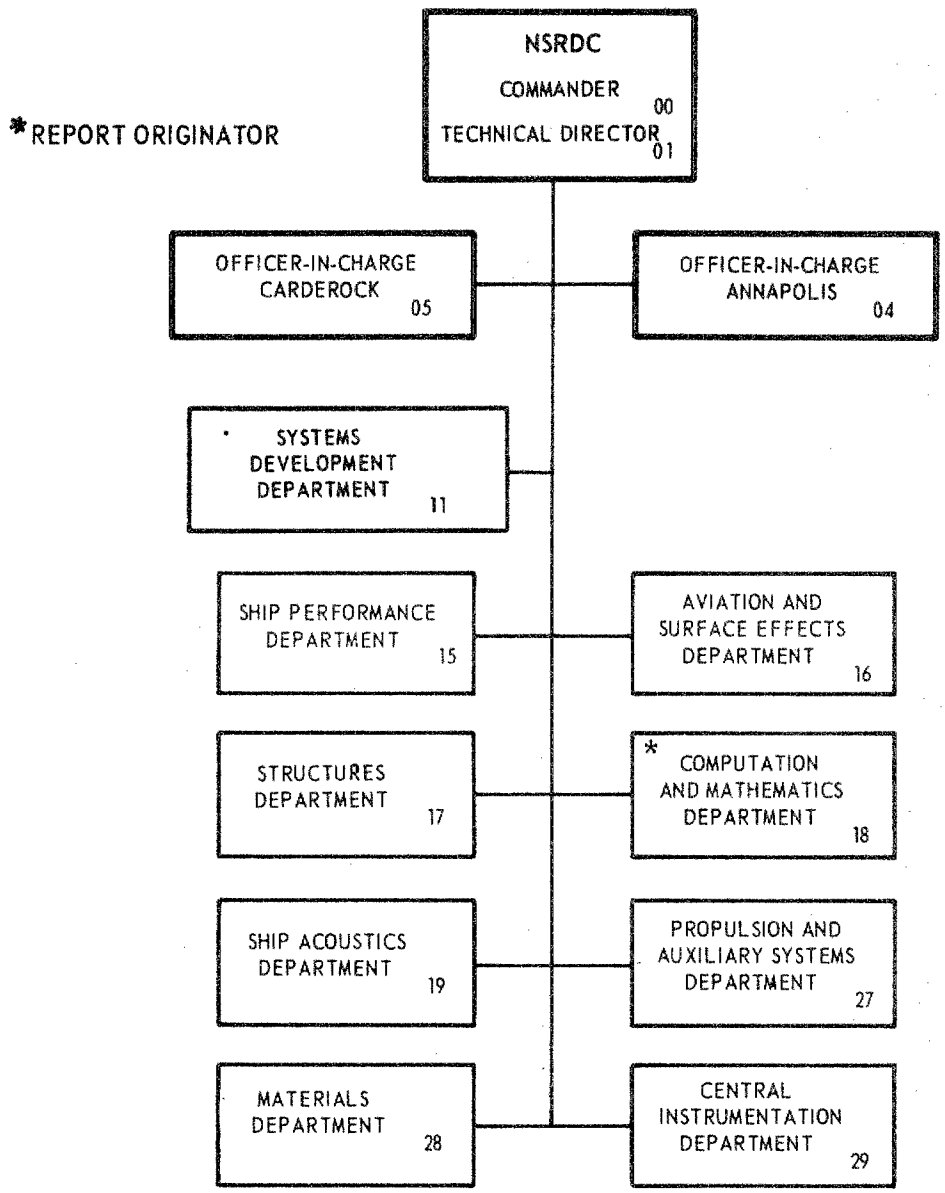
Report 4063

SURFACE MODELING AND ANALYSIS SUBROUTINE PACKAGE

The Naval Ship Research and Development Center is a U. S. Navy center for laboratory effort directed at achieving improved sea and air vehicles. It was formed in March 1967 by merging the David Taylor Model Basin at Carderock, Maryland with the Marine Engineering Laboratory at Annapolis, Maryland.

Naval Ship Research and Development Center
Bethesda, Md. 20034

MAJOR NSRDC ORGANIZATIONAL COMPONENTS



NDW-NSRDC 3960/44 (REV. 8/71)
GPO 917-872

DEPARTMENT OF THE NAVY
NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER
WASHINGTON, D. C. 20007

SURFACE MODELING AND ANALYSIS SUBROUTINE PACKAGE

by

P. M. Rosenshine

April 1973

Report 4063

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The design of surfaces of vehicles is a lengthy and tedious process. Advances in interactive computer graphics and mathematical representation of curves and surfaces have made interactive graphics an attractive medium for design. The Surface and Modeling and Analysis program provides a set of computer subroutines for (1) calculating descriptive information about a surface and (2) manipulating data (by providing the interface between the programmer and the data structure). These routines are based upon parametric cubics and the Coons' Patch technique. A brief description of the representation is included.

The SMA subroutine package was implemented on the CDC 6700 Computer.

ADMINISTRATIVE INFORMATION

This work was performed within the Graphic Systems Development Group, Computer Sciences Division, under Task 15324, Subtask Area SR0140301, Job Order 1-1802-001 the Mathematical Sciences Subroutine Library Project.

INTRODUCTION

Traditionally, surface design has been carried out on the drawing board. With recent developments in the area of interactive computer graphics, and with improvement in plotters and numerically controlled machine tools, new methods of computer-aided geometrical design have emerged. These methods are based on mathematical descriptions of shape from which drawings or instructions for a numerically controlled machine tool can be produced. The non-mathematical representation of a geometric shape--for example, by a grid of points--is both wasteful of storage space in the computer and ambiguous, since a complete description of the object is not available. The method of mathematical representation selected must be capable of describing the kinds of shapes which are encountered in design. The method must have the ability to generate rapidly any particular view of a surface, and it must be axis-independent. Some mathematical representations do not permit multiply-valued curves with large slopes relative to the coordinate axis, or closed curves. Parametric methods of curve and surface description have been widely used to overcome these restrictions. Curves are represented by vector-valued functions of a parameter, t, for example

$$P(t) = [f(t),g(t),h(t)]$$

and surfaces are represented by functions of two parameters, u and v, for example

$$P(u,v) = [f(u,v),g(u,v),h(u,v)]$$

In general, the shape of a real object is non-analytic so that local conditions do not have an overall effect on shape. For this reason various piece-wise representations of curves and surfaces have been devised by which the extent of the influence of a local change in shape can be controlled.

A mathematical representation can be obtained in either of two basic ways: by fitting, or by design. In the former method, the problem is to obtain a mathematical representation for a shape that has not been mathematically defined but which exists either as a physical model from which data points can be measured or as the result of some procedure. Since the shape is already defined, the fitting process can be largely automatic. In the design method, however, the problem is either to create a shape which satisfies design constraints, or to modify an existing mathematically defined shape. In this case, human intervention is often essential.

The Surface Modeling and Analysis (SMA) subroutine package uses both parametric cubics and "Coons' Patch" Techniques[1,2,3]. The first version of SMA (Mod1, described here) may be used in first-order design of a surface (mathematical modeling). A subsequent version (SMA-Mod2)

---

[1] Coons, S. A., "Surface for Computer-Aided Design of Space Forms," MIT Project MAC, MAC-TR-41 (Jun 1967).

[2] Forrest, A. R., "Curves and Surfaces for Computer-Aided Design," Cambridge University, CAD Group, Ph.D. Thesis (Jul 1968).

[3] Armit, A. P., "A Multipatch Design System for Coons' Patch," IEE International Conference of Computer-Aided Design, Conference Publication 51, Southampton (Apr 1969).

planned for development later in 1973 will enable the user to design a new surface under certain constraints and will provide a simplified method of modification of an existing mathematical model. It will also provide capabilities for analysis of the model (area, moments, etc.) created by SMA.

SMA SUBROUTINE PACKAGE

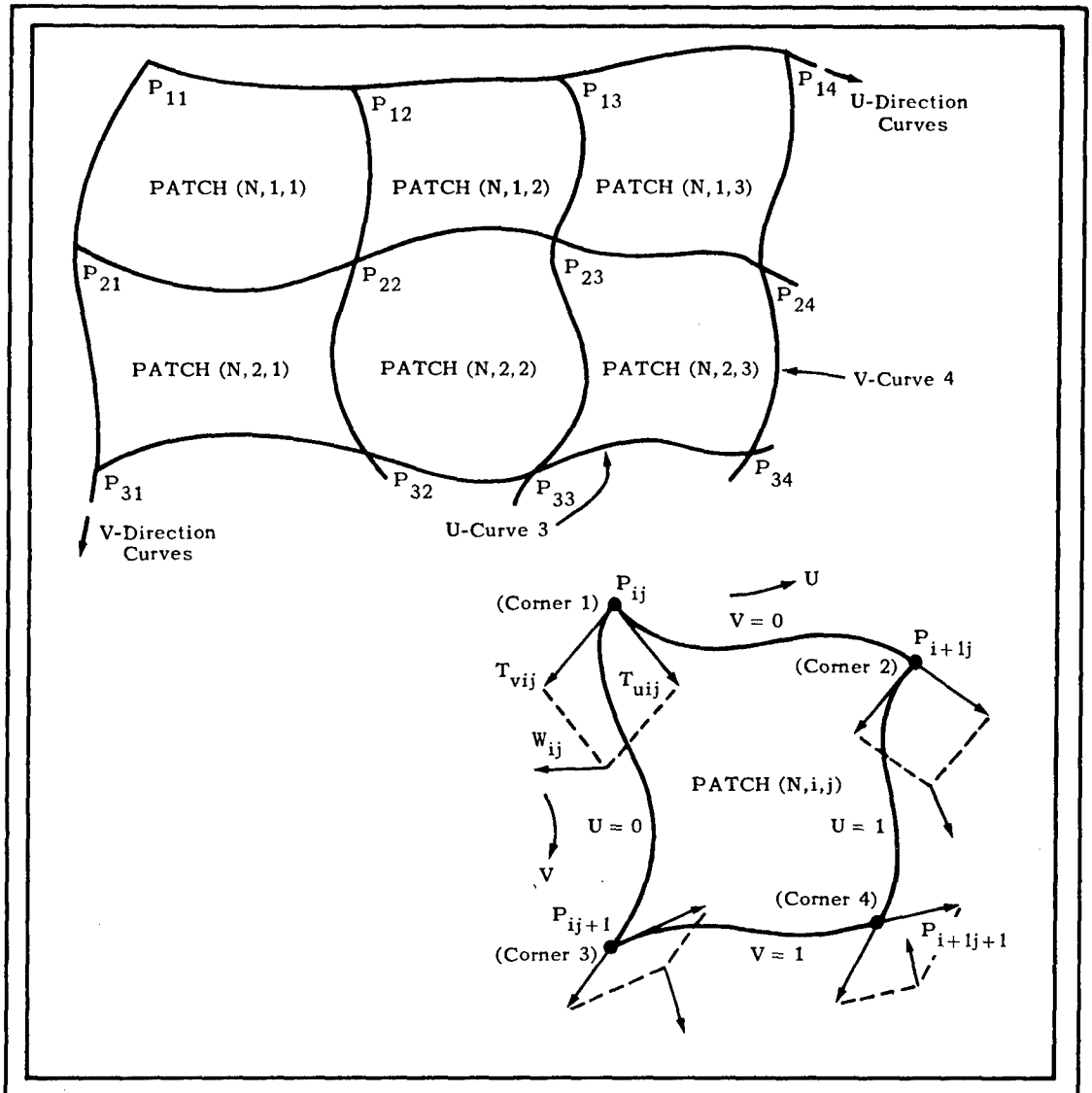
The functions performed by the various SMA component routines are summarized in Table 1 on the opposite page. The package enables (1) the manipulation of data (via the interface it provides between the programmer and the data structure), and (2) calculation of descriptive information about the surface (interior and boundary points, tangents, and twist vectors). At present, the data structure resides in a programmer-defined array; later versions of SMA will use disk files.

A typical surface and an enlargement of a single patch are illustrated in Figure 1, page 6, using SMA patch notation.

Calling parameters and variables used extensively in the SMA subroutines are explained in Table 2, page 7.

TABLE 1 - SUMMARY OF SMA SUBROUTINE CAPABILITIES

INTERFACE ROUTINES	
LGETV,LSETV	Retrieves (or stores) surface patch data from (or into) the array PATCH according to given patch numbers.
IGETV,ISETV	Retrieves (or stores) surface patch data from (or into) the array PATCH using given curve numbers at intersections.
MAIN COMPUTATIONAL ROUTINES	
SPLINE	Computes endpoint rates (related to tangents) and puts them into the data structure.
GENSURF	Generates boundary and interior points of a patch.
GENERAL PURPOSE SUBROUTINES	
CUBIT	Fits a parametric cubic spline through a series of points.
ICONP ICONT	Compare 3-dimensional vectors for proportionality and identicality.
EPTC	Checks for user-defined tangents and sets up an endpoint tangent condition array.
MZCAM	Compares a 3-dimensional vector to (-0., -0., -0.)
MULT	Performs matrix multiplication.
BNDRYPT	Determines points on the boundary of a patch.
FGBLEND	Determines the Coons' blending function value at a given parameter value.



Patch Notation: PATCH(N,IP,JP) where N denotes an entire patch
IP,JP describe the topological position of the patch

Point Notation: P_{ij} where i is the U-curve number
j is the V-curve number

Tangent Vectors: $T_{uij} = \frac{\partial}{\partial u}(P(u,v)) \Big|_{u,v = 0 \text{ or } 1}$

$T_{vij} = \frac{\partial}{\partial v}(P(u,v)) \Big|_{u,v = 0 \text{ or } 1}$

Twist Vectors: $W_{ij} = \frac{\partial^2}{\partial u \partial v}(P(u,v)) \Big|_{u,v = 0 \text{ or } 1}$

Figure 1 - SMA Notation

TABLE 2 - EXPLANATIONS OF SURFACE PATCH NOTATION

Variable or Array	Description
MAX_u	The number of curves in the U-direction to be contained in the data structure. (See Figure 1.)
MAX_v	The number of curves in the V-direction to be contained in the data structure. (See Figure 1.)
M1	$MAX_u - 1$
M2	$MAX_v - 1$
PATCH	The name of the array containing the surface patch data structure. The array is dimensioned PATCH(48, M1, M2). (See SMA Data Structure).
IP, JP	Subscripts of specific patch to be referenced: (PATCH(n, IP, JP) where n denotes a vector component. $1 \leq IP \leq M1$, $1 \leq JP \leq M2$. When referring to an entire patch, N will be in for n.

DATA STRUCTURE INTERFACE ROUTINES

Each Coons' patch consists of 16 vectors: four point-position vectors, eight tangent vectors, and four twist vectors. An SMA patch consists of 48 sequential locations in an array. The patch is divided into four partitions, one for each corner point. Each corner point partition is subdivided into four 3-dimensional vectors. Within the partition, the vectors are stored in the following order: (1) point; (2) tangent along a U-curve; (3) tangent along a V-curve; (4) twist

vector. The components of the vectors can be stored in any order, but the order must be consistent throughout the structure. The entire system of patches is stored in a 3-dimensional array, PATCH(48, M1, M2). Figure 2 illustrates the way in which the associated data are stored. Each partition represents all the data associated with the corresponding corner--for example, Partition 2 contains all the data for Corner 2.

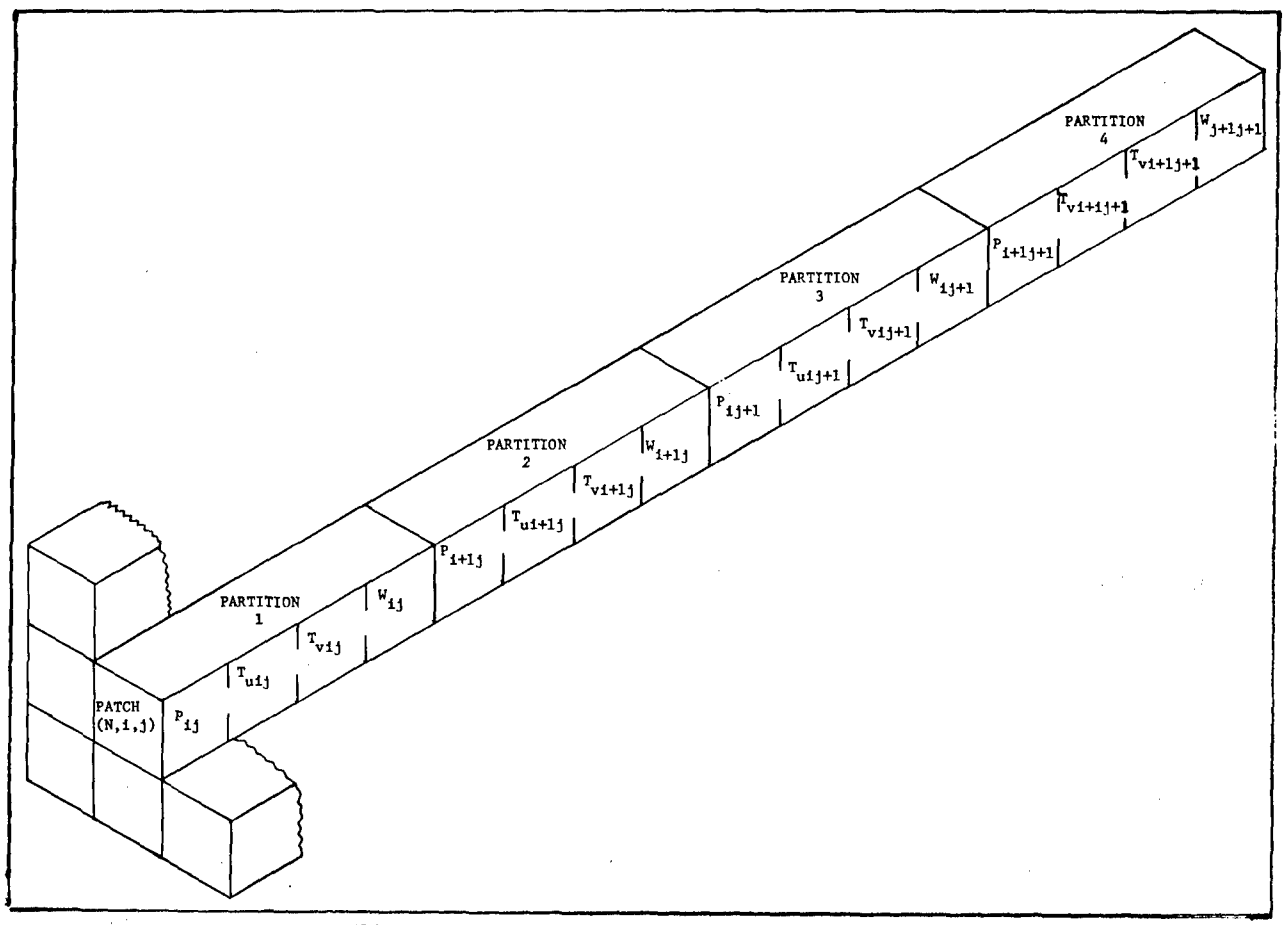


Figure 2 - SMA Data Structure

FUNCTION LGETV

Entry Points:

LGETV, LSETV

This function retrieves (or stores) surface patch information from (or within) the PATCH data structure, given the patch reference numbers.

Calling Sequence:

N = LGETV(PATCH, M1, M2, IP, JP, NPT, KODE, VECTOR)

Parameter Definitions:

(Input) NPT — Number of the corner to be referenced (Figure 1). If NPT = 0, all four corner points will be referenced.

KODE — Specific vector at corner NPT to be referenced.

0	All four vectors
1	Point
2	Tangent along U-curve
3	Tangent along V-curve
4	Twist

(Output) VECTOR — Array to receive patch data, or that containing information to be stored into the data structure.

LGETV

= 0	Function is completed
= 1	Error has developed

Detailed Description:

When LGETV is called, with NPT = 0, the entire patch (48 numbers) is returned in VECTOR. Otherwise, NPT indicates the specific corner to be referenced. If all 12 numbers associated with a corner are desired, KODE is given as 0. Otherwise, KODE denotes the specific 3-dimensional

vector to be referenced. The vector obtained is positioned immediately following PATCH(3*(KODE-1) + 12*(NPT-1), IP, JP) in the Patch array.

If an attempt is made to reference a surface patch outside of the bounds of the data structure, or if there are any other illegal calling parameters, LGETV is returned with a value of 1. Otherwise, its value is 0.

The description of entry LSETV applies also for LGETV, except that LSETV updates data in the structure whereas LGETV retrieves data from the structure.

Future Modifications:

In the MOD2 version of the SMA subroutines, LGETV and LSETV will use the Interactive Data Manager for the PATCH data structure storage.

Subroutines Called: None.

FUNCTION IGETV

Entry Points:

IGETV, ISETV

This function retrieves (or stores) surface patch information from (or within) the PATCH data structure, given the curve numbers of an intersection.

Calling Sequence:

N = IGETV(PATCH, M1, M2, KNU, KNV, NPT, KODE, VECTOR)

Parameter Definitions:

(Input)	KNU	U-curve number (Figure 1)
	KNV	V-curve number (Figure 1)
	NPT	Corner point number to be referenced (Figure 3)
	KODE	Specific vector to be referenced (See LGETV)
(Output)	VECTOR	Array to receive patch data, or that containing information to be stored into the data structure.
	IGETV	= 0 Function is completed = 1 Error has developed

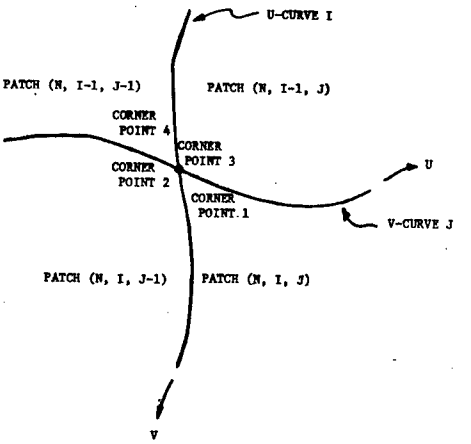
Detailed Description:

This routine is similar to LGETV; thus, only differences between the two will be noted.

(1) The U- and V-curve numbers are specified instead of the PATCH reference numbers.

(2) NPT may not equal 0

Subroutines Called: LGETV, LSETV



The corner point is found in the corresponding partition of each of the four patches.

Figure 3 - A Shared Interior Corner Point

MAIN COMPUTATIONAL ROUTINES

A mathematical model of a surface through a given set of points can be constructed using the two routines described in this section. There is one condition: the set of points must form a quadrilateral grid network. However, since one or two of the sides of the patch may be degenerate, a two- or three-sided figure can be obtained. The quantities determined in subroutine SPLINE are endpoint tangent rates. After the endpoint tangent rates have been determined, interior points of patches can be computed with GENSURF.

The techniques used in the construction of the surface are described in Appendix A which contains a paraphase of sections of a paper by I. M. Yuille.[4]

SUBROUTINE SPLINE

This routine coordinates all of the operations needed to be performed in order to fit a SPLINE curve with data from an array of surface patches.

Calling Sequence:

```
CALL SPLINE(PATCH, M1, M2, CURVPT, RATES, ENDS, SPACE, ID, KRV,
            IERR)
```

[4] Yuille, I. M., "A System for On-Line Computer Aided Design of Ships - Prototype System and Future Possibilities," paper presented at meeting of Royal Institution of Naval Architects, London (Apr 1970).

Parameter Definitions:

CURVPT(J)	An array used to store points retrieved from the data structure ($J \leq 3*$ MAX(M1, M2))
RATES(J)	An array in which parametric tangents are stored before entering them into the data structure.
ENDS(8)	An array used for storing endpoint tangent conditions.
SPACE(K)	A scratch array used in the solution of simultaneous equations in tangent calculations. ($K \leq 4*$ MAX(M1, M2, 3))
ID	Type of Curve. ID = 1 for U-curve, ID = 2 for V-curve
KURV	Curve Number.
IERR	Number of retrieval and updating errors that occurred during manipulation of patch information. (Should be 0 if calling parameters are good.)

Detailed Description:

SPLINE assumes that the patch array was initialized to -0 (all bits on—777777777777777777B) at the programs start. The reason for this initialization is that SPLINE checks each tangent vector on the curve KURV to determine if any tangents were predefined by the user. If tangents have been defined, or if SPLINE finds any point discontinuities, the curve is broken into segments with the defined tangents as endpoint tangent conditions. If SPLINE encounters a defined tangent at the beginning of an interval but not at the end of the previous interval, the curve is broken and a free end is assumed for that previous interval. The same is done for a defined tangent at the end of an interval and an undefined beginning of the next interval (See Figure 4). If SPLINE encounters point discontinuities with no defined tangents at the points, free ends are assumed.

Figure 4a - If there are no defined tangents (except possibly the curve endpoints), the first derivative will be continuous.

Figure 4b - If a tangent is defined at the final point of a segment, a free (hinged) end is assumed for the succeeding segment (unless it, too, is defined). In general, slope will be discontinuous.

Figure 4c - If a tangent is defined at the initial end of a segment, the final end of the preceeding segment is assumed to have a free (hinged) end (unless it is defined). In general, slope will be discontinuous.

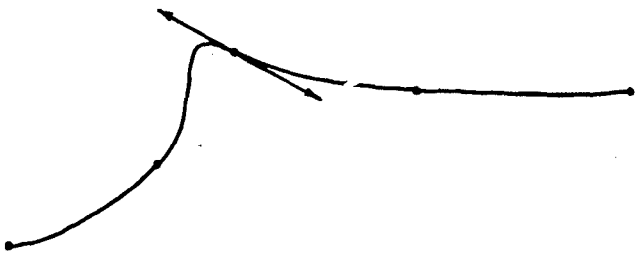


Figure 4d - If both tangents at a single point are defined, the segments will have the corresponding slope; if the tangents are scalar multiples of each other, the slope will be continuous; if the tangents have equal magnitude and opposite direction, the first derivative will be continuous.

Figure 4 - Spline Segments with Interior Tangents Defined

Cross-curve point discontinuities are handled by spline fitting the curve twice. For example, if a U-curve was being splinefit, the V = 0 boundary curve would be fit first, then the V = 1 boundary curve would be fit. (Figure 5).

In general, SPLINE will accommodate any kind of surface or slope discontinuity as long as the discontinuity is on a patch boundary or at a corner point.

Subroutines Called: CUBIT, INCONP, ISETV, IGETV, EPTC, MZCAM

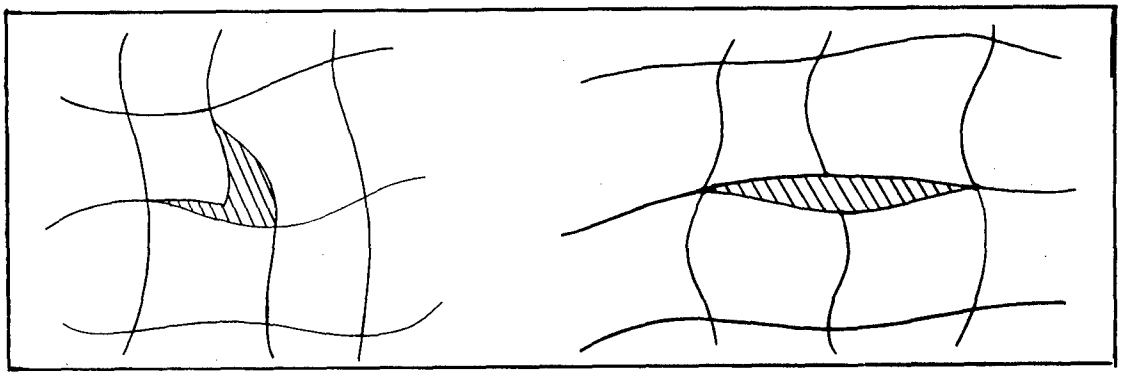


Figure 5 - Examples of Discontinuities SPLINE can Handle

SUBROUTINE GENSURF

This routine generates points on a surface within a particular patch, given the surface tensor of the patch.

Calling Sequence:

CALL GENSURF (PATCH, M1, M2, IP, JP, XYZ, JVSP, IVSP)

Parameter Definitions:

(Input)	JVSP, IVSP	Number of sub-intervals in the V and U directions, respectively +1.
	XYZ(3, JVSP, IUSP)	Array containing intermediate boundary and surface points.

Detailed Description:

This routine computes each point on the surface by evaluation of the surface equation: $(X, Y, Z) = UMBM^tV^t$

where $U = [U^3\ U^2\ U\ 1] \quad 0 \le U \le 1$

$V = [V^3\ V^2\ V\ 1] \quad 0 \le V \le 1$

$$M = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 00 & 01 & 00_v & 01_v \\ 10 & 11 & 10_v & 11_v \\ 00_u & 01_u & 00_{uv} & 01_{uv} \\ 10_u & 11_v & 10_{uv} & 11_{vv} \end{bmatrix}$$ Patch Surface Tensor

First, MBM^t is computed. Then V is varied from 0 to 1 for each value of U from 0 to 1. The increments in the U and V direction are 1/IUSP and 1/JVSP, respectively. GENSURF uses a labelled COMMON/SCRATCH/ (that is 184 words in length) for working space.

Subroutines Called:

LGETV, MULT

- - - - - - - - - - - - - - - -

SUBROUTINE CUBIT

This routine fits a parametric cubic spline through a series of points in an n-dimensional space such that specified end conditions are satisfied.

Calling Sequence:

CALL CUBIT(PTS, RATES, NDIM, ENDS, SPACE, MAXPTS, PRMNT)

Parameter Definitions:

(Input)	PTS (NDIM, MAXPTS)	Array containing points through which a cubic spline will be fitted. Points must be in succession.
	NDIM	Number of dimensions in point space (and first dimension in the rates and PTS arrays).
	ENDS(J)	Endpoint conditions for the spline fit. (J = 2* NDIM +2) See note on endpoint condition setup.
	SPACE(K)	Array used as a working area for the solution of MAXPTS simultaneous equations. ($K \leq R^*$ MAXPTS).
	MAXPTS	The number of points to be fitted.
	PRMNT	The range of the parameter over a curve segment as it goes from the beginning point to the end point.
(Output)	RATES (NDIM, MAXPTS)	Array containing endpoint tangent rates.

Detailed Description:

The SPLINE approximation is a technique which attempts to reproduce mathematically a draftsman's method of drawing fair curves.

The differential equation for the mechanical spline curve is

$$\frac{Y''}{(1 + Y'^2)^{3/2}} = \frac{M(x)}{EI} \qquad (1)$$

where $Y = f(x)$

$M(x)$ Bending moment in the Spline.

E Youngs modulus.

I Cross sectional moment.

In linerized beam theory, Y' is assumed to be small compared to 1. Therefore, Equation 1 reduces to

$$Y'' = f''(x)$$

Clearly, Y can be expressed as a cubic polynomial. Generalizing this, Y and X could be expressed as cubic polynomial functions of a parameter U. This routine fits each component of an n-dimensional spatial set of points with a set of parametric cubic polynomials that is, for example,

$$X_i = X_i(u) \quad i = 1, 2, \ldots n-1$$
$$X_i = Y_i(u) \quad 0 \leq U \leq 1$$
$$Z_i = Z_i(u)$$

for Cartesian coordinates.

A parametric spline function with joints U_1, U_2, U_3, ... U_n is a function $\overline{S}(U)$ constructed such that:

1) In each of the intervals $[U_1, U_2]$, $[U_2, U_3]$, ... $[U_{n-1}, U_n]$, $\overline{S}(U)$ is a cubic polynomial,

2) $\overline{S}(U)$ has continuous first and second derivatives over the interval $[U_1, U_n]$,

3) If $S_x(u)$ is a spline approximation to $x = f(u)$, then $S_x(u) = f(u)$ at all joints in the considered interval.

A cubic polynomial $\bar{S}_{ji}(u)$, (the i^{th} interval of the spline function for the j^{th} dimensional components of the points), can be found in a straight-forward manner if $\bar{S}_{ji}(U_i)$, $\bar{S}_{ji}(U_{i+1})$, $\bar{S}_{ji}(U_i)$, and $\bar{S}_{ji}(U_{i+1})$ are known, where $\bar{S}_{ji}(U_i)$ and $\bar{S}_{ji}(U_{i+1})$ are point components and $\bar{S}_{ji}(U_{i+1})$ are endpoint rate or tangent components. For "PRMNT" = 1., the following results are obtained: (See Figure 6, page 22, for notation)

$$C_{i1} = PTS(j, i)$$

$$C_{i2} = RATES(j, i)$$

$$C_{i3} = 3* (PTS(j, i+1) - PTS(j,i)) - 2* RATES(j,i) - RATES(j,i+1)$$

$$C_{i4} = 2.* (PTS(j,i) - PTS(j, i+1)) + RATES(j, i) + RATES(j,i+1)$$

$$\bar{S}_{ji}(u) = \sum_{K=1}^{4} \bar{C}_{ik} U^{(k-1)} \qquad i = 1, 2, \ldots n-1$$

$$0 \le U \le 1.$$

where:
$$\bar{S}_{ji}(U_i) = PTS(j,i)$$
$$\bar{S}_{ji}(U_{i+1}) = PTS(j,i+1)$$
$$\bar{S}_{ji}(U_i) = RATES(j,i)$$
$$\bar{S}_{ji}(U_i) = RATES(j,i+1) \qquad (2)$$

For this program, the lengths of all of the intervals $[U_i, U_{i+1}]$ are equal to "PRMNT".

A brief description of the calculation of the entries of the rates array will now be given. A necessary condition for S(u) to have continuous first and second derivatives is that the equation below must be satisfied.

$$S'(U_{i-1}) + 4\,S'(U_i) + S'(U_{i+1}) = \frac{3}{\ell}\,[S(U_{i+1}) - S(U_{i-1})]$$

$$\text{where } \ell = \text{PRMNT} \qquad (3)$$

$$2 \le i \le n-1$$

For a problem with n joints, these n-2 equations must be supplemented with two endpoint tangent conditions. If the ends are free, $S''(U_1) = S''(U_n) = 0$. This condition is equivalent to the following two equations:

$$4\,S'(U_1) + S'(U_2) = \frac{3}{\ell}\,[S(U_2) - S(U_1)]$$

$$4\,S'(U_n) + S'(U_{n-1}) = \frac{3}{\ell}\,[S(U_n) - S(U_{n-1})] \qquad (4)$$

If the ends are fixed, then the values from ENDS array are used as the two supplementary conditions.

The matrices associated with the simultaneous linear equations are tridiagonal and diagonally dominant. The equations can be written in the following form:

$$C_{21}\,S'(U_1) + C_{31}\,S'(U_2) = C_{41}$$

$$C_{1j}\,S'(U_{j-1}) + C_{2j}\,S'(U_j) + C_{3j}\,S'(U_{j+1}) = C_{4j}$$

$$j = 2, 3, \ldots n-1$$

$$C_{1n}\,S'(U_{n-1}) + C_{2n}\,S'(U_n) = C_{4n} \qquad (5)$$

The coefficients of Equation 5 correspond to those shown in Equations 3 and 4.

A Gaussian elimination method can be used to solve the equations. First, eliminate the lower diagonal in the following way:

$$C'_{21} = C_{21},\ C'_{41} = C_{41}$$

$$C'_{2j} = C_{3j-1} - C_{zj} * [C_{2j-1}/C'_{1j}]$$

$$C'_{3j} = - C_{3j} * [C_{2j-1}/C'_{1j}] \qquad j = 2,\ldots n$$

$$C'_{4j} = C'_{4j-1} - C_{4j} * [C_{2j-1}/C'_{1j}) \qquad (6)$$

Then the nth equation can be solved by

$$S'(U_n) = C'_{2n}/C'_{4n}$$

Finally, the remaining unknowns can be found by

$$S'(U_i) = [C'_{4i} - C'_{3i} * S'(U_{i+1})]/C'_{2i} \qquad i = n-1,\ n-2,\ldots 1$$

This equation yields all of the $S'(U_i)$'s or RATES (j,i) known.

If PRNMT $\neq$ 1, then the RATES array is scaled by PRNMT.

This procedure is carried out for each dimension of PTS array.

Subroutines Called:

None

Endpoint condition setup for CUBIT. The number of endpoint conditions is dependent on the number of dimensions in the point space. For example, for 2-dimensional space (x,y), there would be six end conditions. The following table gives information for 3-dimensional space stored as (x,y,z) in the PTS array, with the parameter U, in the interval (a,b).

	Free Ends	Fixed Ends
Pa	ENDS(1) = 1.	ENDS(1) = 2.
	ENDS(3 through 5) = 0.	ENDS(3) = $\frac{\partial x}{\partial u}\Big\|$Pa
		ENDS(4) = $\frac{\partial y}{\partial u}\Big\|$Pa
		ENDS(5) = $\frac{\partial z}{\partial u}\Big\|$Pa
Pb	ENDS(2) = 1.	ENDS(2) = 2.
	ENDS(6 through 8) = 0.	ENDS(6) = $\frac{\partial x}{\partial u}\Big\|$Pb
		ENDS(7) = $\frac{\partial y}{\partial u}\Big\|$Pb
		ENDS(8) = $\frac{\partial z}{\partial u}\Big\|$Pb

Note that infinite slopes can be produced since, for example, $\frac{\partial y/\partial u}{\partial x/\partial u} = \frac{\partial y}{\partial x}$ and $\partial x/\partial u$ could be zero. Also note that scalar multiples of the endpoint conditions will, in general, produce different results, even though the derivatives such as $\frac{\partial y}{\partial x}$ would be the same.

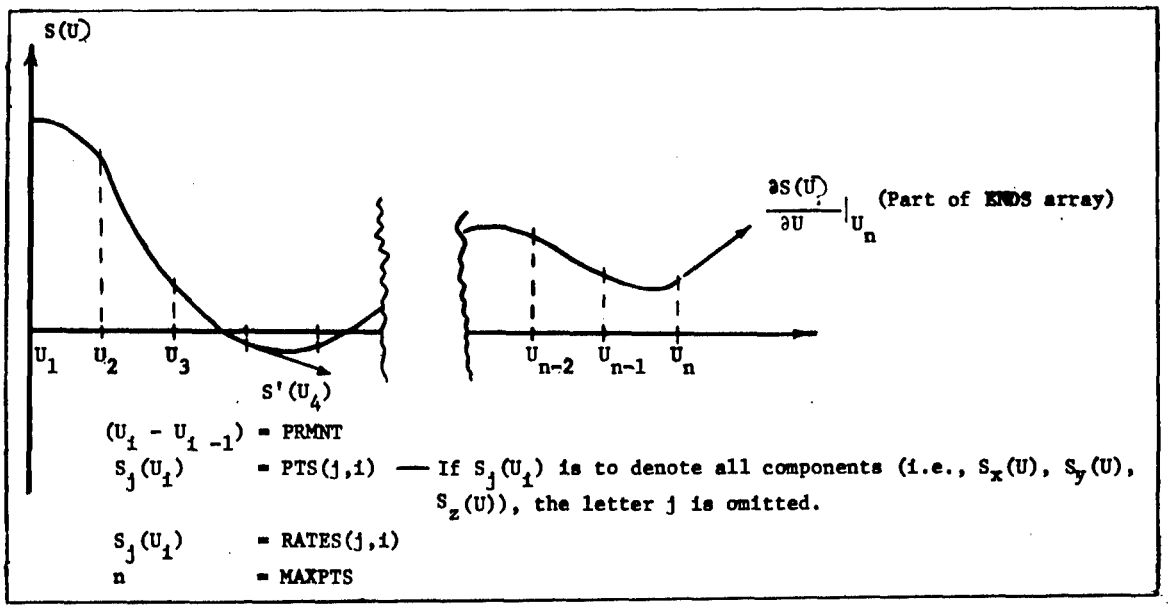


Figure 6 - Notation used in Subroutine CUBIT

FUNCTION ICONP

Entry Points:

ICONP, ICONT

ICONP compares two 3-dimensional vectors to determine if they are identical. ICONT determines whether the two vectors are scalar multiples of each other.

Calling Sequence:

J = ICONP (R1, R2)

Parameter Definitions:

Input: R1, R2 - Vectors to be compared

Output: ICONP = 0 if vectors are identical

= 1 if vectors are distinct

or

= 0 if vectors are scalar multiples

= 1 if vectors have different directions

Detailed Description:

ICONP compares the like components of the vectors to determine whether they are equal.

ICONT first looks for any zero components in R2 that do not have corresponding zero components in R1. If there are no non-corresponding zero components, a scale factor D = R1/R2 is computed and the rest of the components of R1 and R2 are checked.

Subroutines Called:

None

SUBROUTINE EPTC

This routine sets up the endpoint tangent conditions array for CUBIT.

Calling Sequence:

```
CALL EPTC (ENDPT, VECTOR, IAORB)
```

Parameter Definitions:

ENDPT(8) - Endpoint tangent conditions array

VECTOR(3) - Endpoint tangent vector

IAORB - Beginning or end of a segment

IAORB = 1 for beginning

IAORB = 2 for end

Detailed Description:

EPTC tests vector to determine if it has been defined by the user. If vector has all components of "-0." (or 7777777777777777777B), it has not been predefined, and the segment is assumed to have a free end. Otherwise, the end is assumed to be fixed. EPTC, then, sets up the ENDPT array in accordance with the convention outlined in CUBIT.

Subroutines Called:

MZCAM

- -

FUNCTION MZCAM

This function compares a three dimensional vector to "-0." or 7777777777777777777B.

Calling Sequence:

```
J = MZCAM(R)
```

Parameter Definitions:

Input: R(3) - Vector to be compared

Output: MZCAM = 1 if R = (-0., -0., -0.).

= 0 otherwise

Detailed Description:

Since a -0. = 0. in the computer, this function is needed to distinguish between them. The routine compares R to -0. by taking one half a word (30 bits) at a time and comparing that to 7777777777B.

Subroutines Called:

None

- -

SUBROUTINE MULT

This routine performs matrix multiplication of any compatible arrays.

Calling Sequence:

```
CALL MULT (ARR1, ARR2, ARR3, NR1, NC1R2, NC2)
```

Parameter Definitions:

Input: ARR1(NR1, NC1R2) - Left array in multiplication

ARR2(NC1R2, NC2) - Right array in multiplication

NR1 - Number of rows in ARR1 and ARR3

NC1R2 - Number of columns in ARR1 and number of rows in ARR2

NC2 - Number of columns in ARR2 and ARR3

Output: ARR3(NR1, NC2) = [ARR1] X [ARR2]

Subroutines Called:

None

- -

SUBROUTINE BNDRYPT

This routine returns the boundary point position vector at a given parameter value.

Calling Sequence:

CALL BNDRYPT (PATCH, M1, M2, IP, JP, ISIDE, UV, CURVAL)

Parameter Definitions:

Input: ISIDE - The number of the boundary on which an intermediate value is desired (see Figure 2)

UV - Parameter value, $0 \leq UV \leq 1$

Output: CURVAL - Array to receive point position vector

Detailed Description:

Subroutine BNDRYPT retrieves point and tangent information from the data structure. The coefficients of a cubic polynomial describing the boundary are then computed. (Formulas appear in detailed

description of subroutine CUBIT). The polynomial is then evaluated at the parameter value "UV".

Subroutines Called:

LGETV

- - - - - - - - - - - - - - - - - - - -

FUNCTION FGBLEND

This function returns the Coons' Surface Blending Function Value at a given parametric value.

Calling Sequence:

X = FGBLEND (IFG, I01, C)

Parameter Definitions:

IFG	Type of blending function IFG = 1 for F-type IFG = 2 for G-type
I01	Type of F or G function I01 = 0 for 0-type I01 = 1 for 1-type
C	Parameter value at which the function will be evaluated ($0 \leq C \leq 1$)

Detailed Description:

FGBLEND returns the value of one of the following functions:

$$F_0(C) = 2*C^3 - 3*C^2 + 1.$$

$$F_1(C) = -2*C^3 + 3*C^2$$

$$G_0(C) = C^3 - 2*C^2 + C$$

$$G_1(C) = C^3 - C^2$$

Subroutine Called: None

ACKNOWLEDGMENTS

The author would like to express his appreciation to the following people: Dr. E. Cuthill for her encouragement, guidance, and support of this work; Dr. F. Theilheimer for his enlightening discussions of some mathematical problems; and Mr. Ruey Chen for his technical assistance.

APPENDIX A

COONS' SURFACE PATCH REPRESENTATION TECHNIQUE

The following material was taken from the paper by I. M. Yuille entitled "A System for On-Line Computer Aided Design of Ships--Prototype System and Future Possibilities," presented at the meeting of the Royal Institution of Naval Architects in London in April 1970. The letter v has been substituted for the letter w throughout, to allow the notation to be consistent with the rest of the report.

"The complete surface is represented by a series of smaller surfaces, called patches, which join together at their edges with any desired order of continuity. Local changes can therefore be made to individual patches or to the whole surface depending on the extent and nature of the continuity conditions.

"Each patch is represented by a three-dimensional vector equation in two parameters u and v so that a point in the surface is given by

$$x = X(u,v)$$
$$y = Y(u,v) \quad 0 \leq u \leq 1$$
$$z = Z(u,v) \quad 0 \leq v \leq 1$$

or in vector notation by $P(x,y,z) = P(u,v)$. It is convenient to omit the commas and write $P(uv)$ or simply (uv). Derivatives will be denoted by vector expressions such as

$$(u0)_u = \left[\frac{\partial (uv)}{\partial u}\right] v = 0$$

"The basis of Coons' work is that each patch is represented by an equation of the form

$$P(uv) = f(uv) + g(uv) + h(uv) + \ldots$$

"A patch has four boundary curves denoted vectorially by (u0), (u1), (0v), (1v) and four corner points (00), (01), (10), (11).

" The first kind of Coons' surface f(uv) is represented in matrix form by

$$f(uv) = -\,[-1\ F_0(u)\ F_1(u)] \begin{bmatrix} 0 & (u0) & (u1) \\ (0v) & (00) & (01) \\ (1v) & (10) & (11) \end{bmatrix} \begin{bmatrix} -1 \\ F_0(v) \\ F_1(v) \end{bmatrix}$$

where the functions $F_0(u)$, $F_1(u)$, $F_0(v)$, $F_1(v)$ are called blending functions. To ensure that the surface passes through the four boundaries and the corner points the blending functions must satisfy the conditions

$$F_0(0) = 1 \qquad F_1(0) = 0$$
$$F_0(1) = 0 \qquad F_1(1) = 1$$

The blending functions are also chosen to satisfy the conditions

$$F_0'(0) = F_0'(1) = F_1'(0) = F_1'(1) = 0$$

so that the derivative with respect to v at v = 0, for example, is

$$f(u0)_v = (00)_v F_0(u) + (10)_v F_1(u)$$

It is seen that the value of the derivative across a boundary of the patch depends only on the tangent vectors at the two ends of the boundary and on the form of the blending functions. Note that the derivative is independent of the shape of the boundary curves.

" The second kind of Coons' surface is represented in matrix form by

$$g(uv) = -\,[-1\ G_0(u)\ G_1(u)] \begin{bmatrix} 0 & g(u0)_v & g(u1)_v \\ g(0v)_u & (00)_{uv} & (01)_{uv} \\ g(1v)_u & (10)_{uv} & (11)_{uv} \end{bmatrix} \begin{bmatrix} -1 \\ G_0(v) \\ G_1(v) \end{bmatrix}$$

The purpose of this surface is to provide a correction to be added to a surface of the first kind in such a way that the derivatives

across the boundaries may be controlled without altering the shapes of the boundaries already defined. The derivative with respect to v at the boundary (uo) is

$$(u0)_v = f(u0)_v + g(u0)_v$$

and the elements such as $g(u0)_v$ are chosen so that for example

$$g(u0)_v = (u0)_v - (00)_v F_0(u) - (10)_v F_1(u)$$

In order that the blending functions shall not change the positions of the boundaries they must satisfy the conditions

$$G_0(0) = G_0(1) = G_1(0) = G_1(1) = 0$$

In order to provide the required derivatives across the boundary they must also satisfy the following end conditions

$$G_0'(0) = 1 \qquad G_1'(0) = 0$$

$$G_0'(1) = 0 \qquad G_1'(1) = 1$$

Note that the cross derivatives at the corners arise in the definition of this second kind of surface.

"In an analogous manner a third function h(uv) can be defined which gives control of the second derivatives across patch boundaries, and so on.

" There is no restriction on the shape of the boundaries of a patch except that they should intersect at its corners and can be represented in parametric form. It is convenient to describe the boundaries in terms of the blending functions. For example, the boundary (u0) is given by

$$(u0) = (00) F_0(u) + (10) F_1(u) + (00)_u G_0(u) + (10)_u G_1(u)$$

" The shape of the surface represented by a patch is a function of the position and the derivatives at the corner points only and of the

equations chosen for the blending functions. The important feature of this work is that, if a surface is represented by a number of contiguous patches having common values of position and derivatives at the corner points and using similar blending function equations, there will be continuity across the patch boundaries of position and slope if the first and second kind of blending functions are used, and of second derivatives as well if second order continuity is maintained along the patch boundaries. In the Master Geometry programs, the unconstrained tangent vectors are chosen by fitting spline curves through the corner points so that second order continuity is achieved along the boundaries at those corners and across adjacent surface patch boundaries.

"In ship, aircraft and car design it is required that the surface used should be fair. No precise definition is available but it is generally agreed that there should be continuity of first and second derivatives (and no unintended points of inflection). It has been found in practice that a fair surface may be represented very well indeed by using blending functions of the first and second kinds only and by taking the form of these functions to be cubic polynomials as follows:

$$F_o(s) = 2s^3 - 3s^2 + 1$$

$$F_1(s) = -2s^3 + 3s^2$$

$$G_o(s) = s^3 - 2s^2 + s$$

$$G_1(s) = s^3 - s^2$$

"The equations may then be combined and written in the form

$$(uv) = [F_0(u)\ F_1(u)\ G_0(u)\ G_1(u)] \begin{bmatrix} (00) & (01) & (00)_v & (01)_v \\ (10) & (11) & (10)_v & (11)_v \\ (00)_u & (01)_u & (00)_{uv} & (01)_{uv} \\ (10)_u & (11)_u & (10)_{uv} & (11)_{uv} \end{bmatrix} \begin{bmatrix} F_0(v) \\ F_1(v) \\ G_0(v) \\ G_1(v) \end{bmatrix}$$

$$= \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij}\ u^i\ v^j$$

This form of Coons' surface may thus be described by a bicubic parametric equation. By including second order continuity along the patch boundaries and using the cross derivative terms fair surfaces may be described for which both the first and second derivatives are continuous throughout. In general this is necessary for the satisfactory representation of doubly curved surfaces. Note that the coefficients in the square matrix include derivatives with respect to the parameters u and v and do not represent physical tangents at points on the surface. The latter may easily be obtained, however, by means of equations of the type

$$\frac{dy}{dx} = \frac{\partial y}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial y}{\partial v}\frac{\partial v}{\partial x}$$

Surfaces described in this way may have virtually any shape and the method is eminently suited to engineering design. Surface description by parametric equations has several advantages compared with non-parametric description. In particular the equations are axis-independent, so that transformations of axes or scale are easily computed, and avoid the possibility of infinite slopes such as may occur if non-parametric

equations are used. The analytical and arithmetical techniques used, for example to calculate intersections, are rather more complex than with other methods. The technique is intended for use on a computer, however, and once it has been implemented the designer is relieved of the necessity to follow the analytical geometry in detail. . ."

APPENDIX B

TEST RUNS OF SMA SUBROUTINES

The performance of the SMA routines was tested and verified by an application program. This program consists mainly of calls to SPLINE, GENSURF, and THREED (a routine to do 3-dimensional plotting on the CalComp 765 plotter). Figures 7, 8, and 9 are output of this test program.

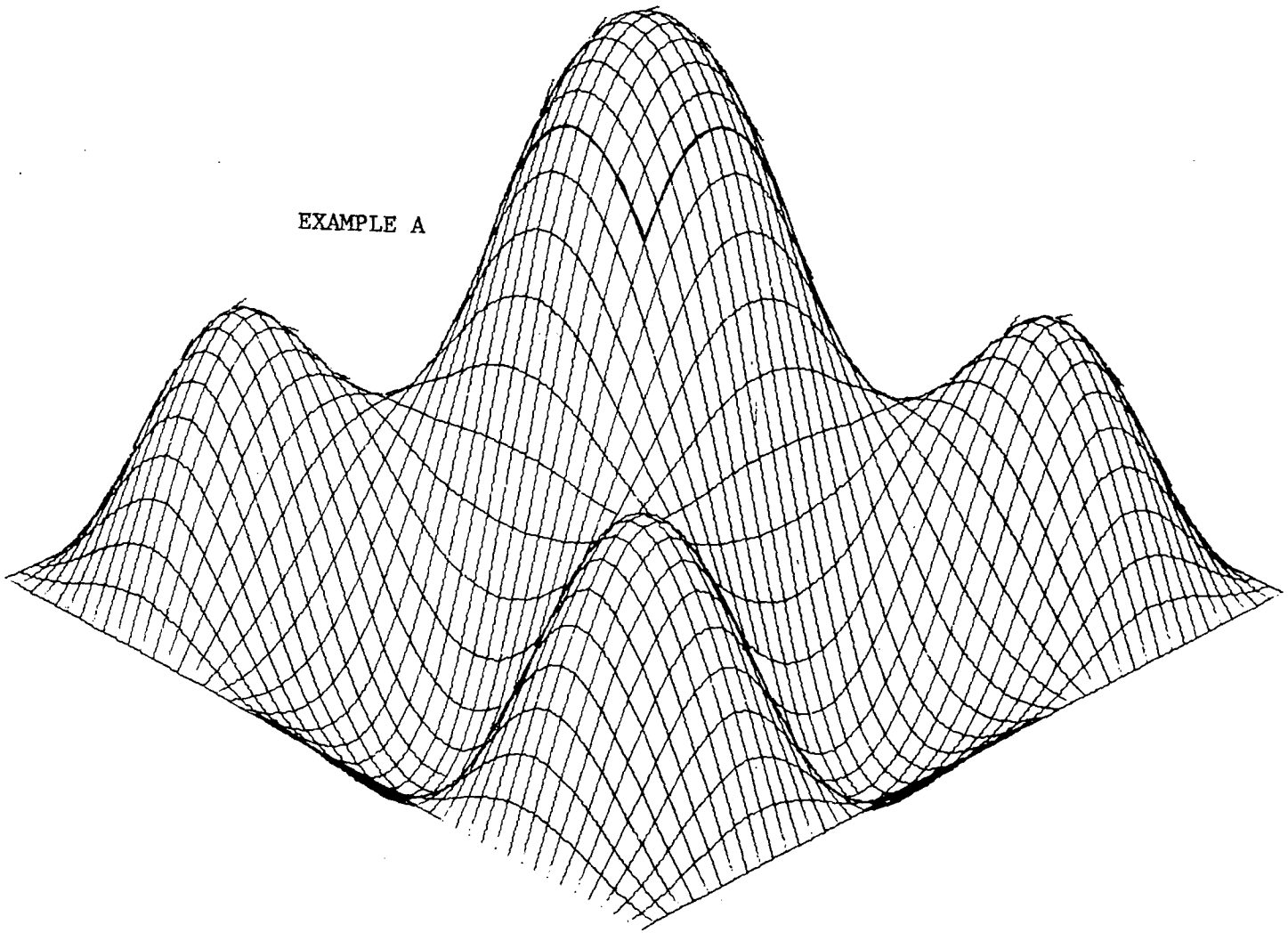


Figure 7 - An Arbitrary 25-Patch Surface Produced by SMA (Example A)

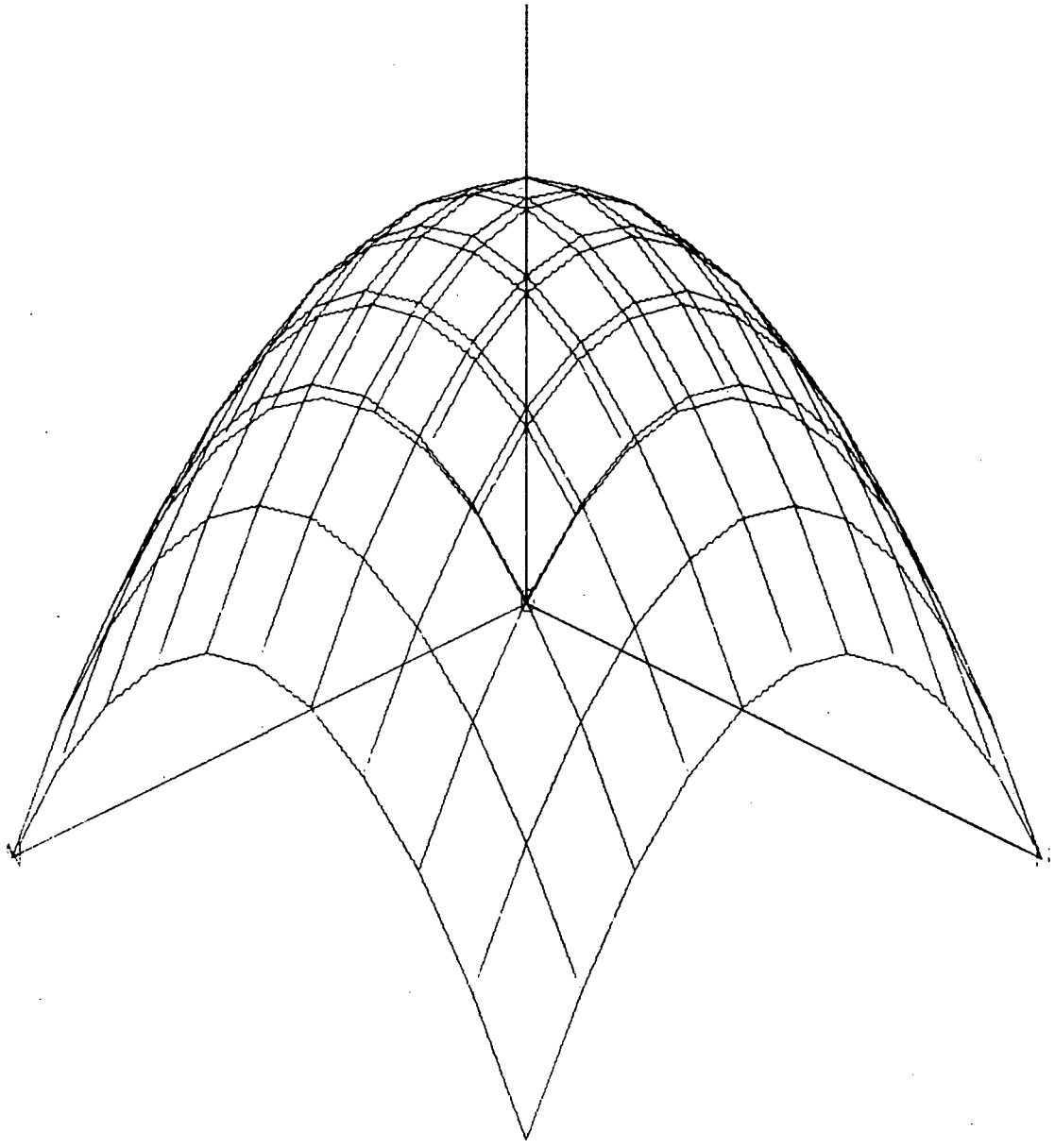


Figure 8 - Enlargement of Example A Center Patch
(Hidden Lines Included)

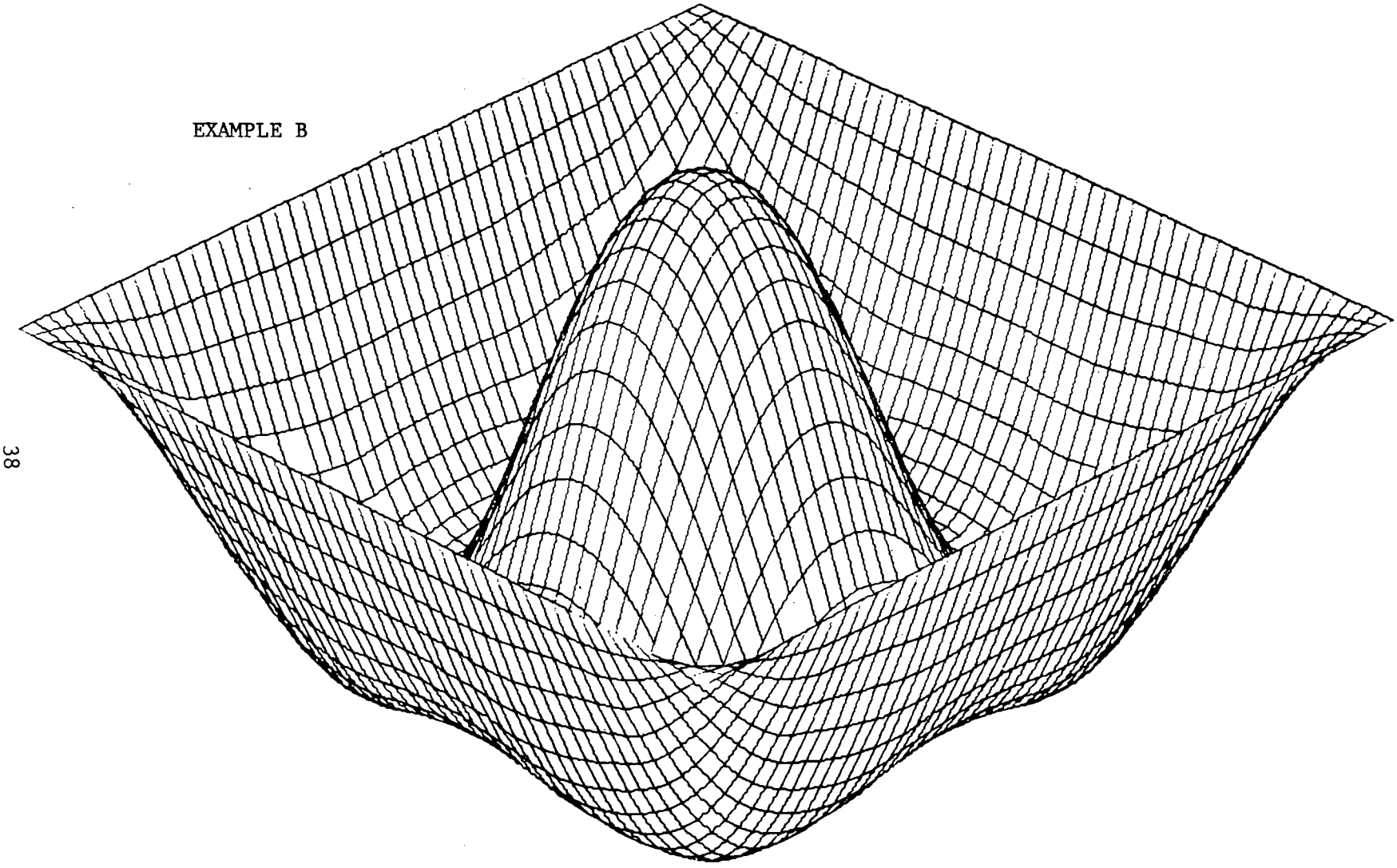


Figure 9 - An Arbitrary 25-Patch Surface Produced by SMA (Example B)

REFERENCES

1. Coons, S. A., "Surface for Computer-Aided Design of Space Forms," MIT Project MAC, MAC-TR-41 (Jun 1967).

2. Forrest, A. R., "Curves and Surfaces for Computer-Aided Design," Cambridge University, CAD Group, Ph.D. Thesis (Jul 1968).

3. Armit, A. P., "A Multipatch Design System for Coons' Patch," IEE International Conference of Computer-Aided Design, Conference Publication 51, Southampton (Apr 1969).

4. Yuille, I. M., "A System for On-Line Computer Aided Design of Ships - Prototype System and Future Possibilities," paper presented at meeting of Royal Institution of Naval Architects, London (Apr 1970).

INITIAL DISTRIBUTION

Copies	
1	DODCI (T. Braithwaite)
1	DARPA
	1 (L. Roberts)
4	CNO
	1 OP-916
	1 OP-916C1 (LCDR W. Poteat)
	1 OP-916D
	1 OP-098TD (L. Aarons)
6	CHONR
	1 ONR 400R (R. Ryan)
	1 ONR 430 (R. Lundegard)
	1 ONR 432 (L. Bram)
	1 ONR 437 (M. Denicoff)
	1 ONR 437 (G. Goldstein)
	1 Tech Lib
12	DDC
1	ONR Boston
1	ONR Chicago (R. Buchal)
1	ONR London
1	ONR New York Area
1	ONR Pasadena (R. Lau)
1	ONR San Francisco Area
4	NRL
	1 5030 (S. Wilson)
	1 5400 (B. Wald)
	1 7810 (A. Bligh)
	1 Tech Lib

Copies	
15	CHNAVMAT
	1 MAT 0141E (R. Jeske)
	1 MAT 03
	1 MAT 03A (CDR R. Booth)
	1 MAT 03L (J. Lawson)
	1 MAT 03L3 (D. Hughes)
	1 MAT 03L4 (J. Huth)
	1 MAT 03L4A (M. Crook)
	1 MAT 03P2 (P. Newton)
	1 MAT 03P21 (S. Atchison)
	1 MAT 03P21 (J. Buhl)
	1 MAT 03T (CAPT D. Keach)
	1 MAT 03T1 (H. Law)
	1 MAT 0316 (CDR D. Brown)
	1 MAT 0342 (CDR Petzrick)
4	USNA
	1 (D. Rogers)
	1 (A. Adams)
	1 Dept of Math
	1 Tech Lib
5	NAVPGSCOL
	1 (M. Woods)
	1 (D. Williams)
	1 (G. Barksdale)
	1 (C. Comstock)
	1 Tech Lib
1	CMC
1	CGMCDEC
1	NAVCOSSACT
1	ADPESO

Copies	
1	COMNAVINT
1	NAVELECSYSCOM
7	NAVSHIPSYSCOM
	1 SHIPS 03 (RADM Andrews)
	1 SHIPS 0311 (B. Orleans)
	1 SHIPS 03414 (A. Chaikin)
	1 SHIPS 03423 (C. Pohler)
	1 SHIPS 0719 (L. Rosenthal)
	1 SHIPS 08 (Nuclear Power Directorate)(NC-2)
	1 Tech Lib
3	NAVAIRSYSCOM
	1 NAVAIR 5033 (R. Saenger)
	1 NAVAIR 5333F4 (R. Entner)
	1 NAVAIR 5375A (J. Polgren)
1	NAVFACENGCOM
2	NAVORDSYSCOM
	1 NAVORD 032C (C. McGuigan)
	1 Tech Lib
1	NAVAIRDEVCEN
1	NCEL
1	NAVWARCOLLEGE
10	NELC
	3 5000 (A. Beutel)
	3 5200 (M. Lamendola)
	3 5300 (J. Dodds)
	1 Tech Lib
1	NAVUSEACEN
1	NAVWPNSCEN
1	NAVCOASTSYSLAB
2	NOL
	1 (H. Stevens)
	1 Tech Lib

Copies	
6	NWL
	1 Code K
	1 Code K-1
	1 Code KO
	1 Code KP
	1 Code KPS
	1 Tech Lib
1	NUSC NPT
1	NAVSHIPYD BSN
1	NAVSHIPYD CHASN
1	NAVSHIPYD HUNTERS PT
1	NAVSHIPYD LBEACH
1	NAVSHIPYD MARE ISLAND
1	NAVSHIPYD NORVA
1	NAVSHIPYD PEARL
1	NAVSHIPYD PHILA
1	NAVSHIPYD PTSMH
1	NAVSHIPYD BREM
15	NAVSEC
	1 SEC 6102 (CDR Burnett)
	3 SEC 6102C (W. Dietrich)
	1 SEC 6102C (P. Bono)
	1 SEC 6105C1 (Y. Park)
	1 SEC 6110.01 (R. Leopold)
	1 SEC 6114 (R. Johnson)
	1 SEC 6114E (A. Fuller)
	1 SEC 6128 (J. O'Brien)
	1 SEC 6129
	1 SEC 6133E (E. Straubinger)
	1 SEC 6178D03 (L. Biscomb)
	1 SEC 6179A20 (J. Singer)
	1 Tech Lib

Copies	
1	AFOSR Code 423
1	Rome Air Development Center
1	WPAFB Code AFFDL
1	NASA Langley Research Center (R. Fulton)
1	MIT
1	VPI
1	Syracuse University Computation Center (Professor S. A. Coons)

CENTER DISTRIBUTION

Copies	Code	Copies	Code
1	1802.1	5	184
1	1802.2	5	1844 (Attn: B. Kelly)
2	1802.3	15	185
1	1802.4	6	186
3	1805	4	188
1	18/1809	5	189
5	183	2	1891
1	1832	10	1892.1
25	1833	1	1892.3
1	1834	3	1745 (J.Potts)
1	1835		

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Naval Ship Research and Development Center Bethesda, Maryland 20034	UNCLASSIFIED
	2b. GROUP

3. REPORT TITLE

SURFACE MODELING AND ANALYSIS SUBROUTINE PACKAGE

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(First name, middle initial, last name)*

Philip M. Rosenshine

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
April 1973	45	4

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO. SR0140301	NSRDC Report 4063
c. 15324	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d. 1-1802-001	

10. DISTRIBUTION STATEMENT

Approved for Public Release: Distribution Unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

The design of surfaces of vehicles is a lengthy and tedious process. Advances in interactive computer graphics and mathematical representation of curves and surfaces have made interactive graphics an attractive medium for design. The Surface Modeling and Analysis (SMA) program provides a set of computer subroutines for (1) calculating descriptive information about a surface and (2) manipulating data (by providing the interface between the programmer and the data structure). These routines are based upon parametric cubics and the Coons' Patch technique. A brief description of the representation is included.

The SMA subroutine package was implemented on the CDC 6700 computer.

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Parametric Cubics						
Coons' Surface Patch						
Interactive Computer Graphics						
Computer-Aided Geometrical Design						
Data Structures						
Cubic Spline						
Blending Functions						