University of Cambridge

C.A.D. Group Document No. 46

SURFACE FITTING FOR COONS' PATCHES

A report on two programs for fitting surfaces to curvilinear grids of points and transforming the resulting surfaces for display and manipulation using the program MultiPatch Design Program.

by Robert E. Marks

Computer Aided Design Group

University Computer Laboratory

October 1970

Introduction

If a computer is used to design the shape of an object, the object is in some sense modelled in the computer. This model consists of data arranged according to various relationships, and is known as the data structure. One of the basic items in the data structure or model is a description of the geometry of the object itself: if one is to apply computers to engineering design one must be able to describe the curves and surfaces which define engineering shapes.

In industries where surface shape is of critical importance methods of surface description have already been developed. The approach taken is largely that of fitting surfaces to a grid of points defined in some other way.

The traditional method of describing complex shapes is by draughting a series of sections along the object. This has two drawbacks: firstly, curves other than straight lines and circular arcs are rarely used by the draftsman, although the object described may be almost free form - this leads to obvious errors in description. Secondly, a drawing may not define a curve shape fully - interpolations between the given sections may lead to inaccuracies in the completed prototype. One aim of storing surface descriptions in the computer is to provide unambiguous definitions.

The computer must be able to deal both with analytic shapes such as conic sections, and with free form surfaces. (Free form surfaces being surfaces of abitrary shape, both smooth or rippled.) It will be an added advantage if the computer can deal with surface fitting from point data such as coordinate information, and with surface design from the start in itself.

Several ways of geometric surface description are possible: description by arrays of coordinate data, by arrays of analytic section curves, or by purely analytic means.

The first has several serious disadvantages: excessively large storage needed, cumbersome transformation computations, and inaccurate interpolations. The second, known as "lofting" in the trade, stores the intersecting curves as sets of orthogonal plane sections with the curves as a series of analytic equations. Storage is reduced and there is a greater degree of analytic definition, but some interpolation is still necessary to obtain points and curves not lying on the plane sections.

Analytic surface description overcomes many of the drawbacks of the other methods - no interpolation is necessary, the exact shape is immediately known, and changes can be made easily with the exact description always present.

To summarize: the most general way to describe a surface is to list sufficient points on the surface to define it to the desired accuracy. This approach is too cumbersome for normal use and in practice it is common to employ some form of interpolation scheme and list fewer points. The more elaborate the interpolation scheme and the fewer the points listed, the nearer one approaches total analytic definition of a surface which is more convenient for computer-aided design and manipulation. In mechanical engineering surfaces are generally smooth and regular, and analytic methods are convenient. But for surfaces of a very arbitrary, almost random, nature (terrain contours, for instance) the analytic methods of description become unwieldy and tedious as well, and other methods are better suited for their description.

Parametric Curves

To describe the three-dimensional curves and surfaces found in mechanical engineering, one attempts to achieve several objectives: to obtain axis independence to be able to describe finite shapes, and to be able to decouple the representation. The curve or surface is an actual physical entity being modelled mathematically, and the model is inaccurate if dependent on the directions arbitrarily decided for the axis directions. As finite objects the curves and surfaces are closed, and this may lead to infinite slopes in the analytic description - very awkward to deal with.

Forrest[1] has shown that using parametric representation has several advantages over the previously more general use of non-parametric representation. Each special coordinate is evaluated and manipulated separately, and a particular value of the parameter defines only one point of the curve. The bounds imposed on the parameter ensure that the desired portion of the curve is selected. Infinities of slope can easily be avoided.

Forrest noted that sophisticated interpolation techniques enable representation of a single curve by a series of piecewise curves with degrees of continuity between adjoining segments. He suggested using parametric polynomial functions as curve segments, and noted that the parametric cubic spline segment is the simplest curve which allows slope continuity with adjoining segments. It is also the simplest non-planar curve: it is fully 3D in the general sense, and it is also the lowest order curve with points of inflexion. It permits tangent vector continuity at both ends. However it reduces to conic curves in special cases only and never to a circular arc, and it is not accurate for describing asymptotic curves. But once end points and tangent vectors are fixed the cubic is completely defined.

$$\text{Let } P = \text{point vector, } [x\ y\ z]$$

$$= au^3 + bu^2 + cu + d$$

$$= [u^3\ u^2\ u\ 1] \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

then

$$P_o = [x\ y\ z]_{u=0} = d$$

$$P_1 = [x\ y\ z]_{u=1} = a + b + c + d$$

$$P'_o = \frac{\partial}{\partial u}[x\ y\ z]_{u=0} = c$$

$$P'_1 = \frac{\partial}{\partial u}[x\ y\ z]_{u=1} = 3a + 2b + c$$

Rewriting

$$\begin{bmatrix} P_o \\ P_1 \\ P'_o \\ P'_1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

and hence

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} P_o \\ P_1 \\ P'_o \\ P'_1 \end{bmatrix} = M \begin{bmatrix} P_o \\ P_1 \\ P'_o \\ P'_1 \end{bmatrix}$$

Extending the homogeneous coordinate approach to space curves using the basis vector $r = [u^3\ u^2\ u\ 1]$ we obtain the rational cubic curve, which in general is a twisted space curve with points of inflexion. (The homogeneous coordinate approach can be considered as describing an n-dimensional curve in (n+1) hyper-space and projecting the resulting hypercurve on the n-dimensional space.) By adding the fourth or homogeneous coordinate, one obtains a very flexible curve - the rational cubic provides a standard form for describing straight lines, conics, cubics, et cetera, and is fully defined by its two end points, its two end tangent vectors, and an intermediate shoulder point.

$$\text{Let } V = [x\ y\ z\ 1]$$

$$wV = [wx\ wy\ wz\ w] = r = [u^3\ u^2\ u\ 1]$$

Then transform the primitive curve by a 4 x 4 matrix A

$$\text{so that } wV = [u^3\ u^2\ u\ 1]\ A$$

Substitution of end points and tangent vectors and "homogeneous coordinates" ((w) u=0, (w)u=1, (w)'u=0, (w)'u=1) yields a specific curve.

Coon's Patches

Coons[(2)] has developed the idea of representing a surface by using a quilt of "patches" with the desired degree of surface and slope continuity across their boundaries. A patch has four edges which meet at four corners. The surface between the four edges is defined by a weighting of the boundary functions defining the edges. The whole surface is thus described in terms of its edge properties and one can thus ensure that the internal patch shape is satisfactory by so designing the edge properties

If each patch edge is a segment of a 3D parametric cubic curve the data structure is simplified since only a small amount of information is sufficient to define the boundary curves. But the range of shapes a single patch can adopt is restricted, as discussed above, unless rational cubic curves are used. These "bicubic" patches, as they are called, can be defined by the four corner coordinates, the eight slope vectors of the ends of the edges, and the four twist vectors at each corner.

Using Forrest's[(3)] notation, a patch is represented by the vector expression (tu), which in the three dimensional case is $(tu) = [x\ (tu)\ y(tu)\ z(tu)]$ A patch is typically the sum of one or more types of Coons surface, denoted here by f(tu), g(tu), etc. Then $(tu) = f(tu) + g(tu) + \ldots..$ The patch will always consist of at least the component f(tu). A patch has four boundary curves: (t0), (t1), (0u), (1u) and four corner points (00), (01), (10), (11). These are, of course, vectors. Figure 1 shows the basic Coons surface patch.

The basic form is $$f(tu) = -[-1\ F_0(t)\ \ F_1(t)] \begin{bmatrix} 0 & (t0) & (t1) \\ (0u) & (00) & (01) \\ (1u) & (10) & (11) \end{bmatrix} \begin{bmatrix} -1 \\ F_0(u) \\ F_1(u) \end{bmatrix}$$

This surface form is composed of the boundary curve vectors and the corner points, and the four functions $F_0(t)$, $F_1(t)$, $F_0(u)$ and $F_1(u)$ which are sometimes called the blending functions. Other than the restriction that they should meet at the four corner points, there is no restriction on the boundary curve functions, provided they can be represented in a parametric form. To ensure that the surface f(tu) includes the boundary curves and corner points, the four F blending functions have the properties that:

$$F_i(j) = \delta_{ij},\ \forall\ i,j \quad \text{(for all i,j)}$$

and further, to ensure that the slope across the boundaries depends only upon the two end tangent vectors across the boundary and blending functions :

$$F_i'(j) = 0, \quad \forall\ i,j$$

It can be shown[(3)] that the corner cross derivations are all zero.

By introducing a second form g(tu) such that

$$g(tu) = [-1\ G_0(t)\ G_1(t)] \begin{bmatrix} 0 & g(t0)u & g(t1)u \\ g(0u)_t & (00)tu & (01)tu \\ g(1u)_t & (10)tu & (11)tu \end{bmatrix} \begin{bmatrix} -1 \\ G_0(u) \\ G_1(u) \end{bmatrix}$$

where $G_0(t)$, $G_1(t)$, $G_0(u)$, $G_1(u)$ are slope blending functions which have the following end conditions to ensure that the correction surface has zero positioned value on the patch edge but has the required cross-boundary slopes

$$G_i\ (j) = 0 \qquad \forall i,\ j$$

and

$$G_i'\ (j) = \delta ij \qquad \forall i,\ j$$

one can match patches with given cross-boundary slopes, or non-zero twist vectors.

One now has a general expression for a slope-matching, slope continuous surface patch with entirely arbitrary boundaries and entirely arbitrary slopes across these boundaries.

It is often convenient to use particular boundary curve functions defined by curve end-points and end-point tangent vectors. If the blending functions themselves are used by writing

$$(t_0) = [F_0(t)\quad F_1(t)\quad G_0(t)\quad G_1(t)] \begin{bmatrix} (00) \\ (10) \\ (00)t \\ (10)t \end{bmatrix} \qquad \text{etc.}$$

one obtains for (tu) = f(tu) + g(tu)

$$(tu) = [F_0(t)\ F_1(t)\ G_0(t)\ G(t)] \begin{bmatrix} (00) & (01) & (00)u & (01)u \\ (01) & (11) & (10)u & (11)u \\ (00)t & (01)t & (00)tu & (01)tu \\ (10)t & (11)t & (10)tu & (11)tu \end{bmatrix} \begin{bmatrix} F_0(u) \\ F_1(u) \\ G_0(u) \\ G_1(u) \end{bmatrix}$$

where the 4 x 4 matrix is the boundary condition matrix B. Given the blending functions, the surface is defined purely in terms of the four corner coordinates, the eight tangent vectors, and the four "twist" or cross tangent vectors.

If the blending functions are chosen to be the primitive cubic blending functions (the simplest functions to obey the necessary conditions above) then one gets the bicubic form

$$(tu) = [A_0(t)\ A_1(t)\ B_0(t)\ B_1(t)] \left|\begin{array}{cc|cc} (00) & (01) & (00)u & (01)u \\ (10) & (11) & (10)u & (11)u \\ \hline (00)t & (01)t & (00)tu & (01)tu \\ (10)t & (11)t & (10)tu & (11)tu \end{array}\right| \left|\begin{array}{c} A_0(u) \\ A_1(u) \\ B_0(u) \\ B_1(u) \end{array}\right|$$

where

$$A_0(x) = 2x^3 - 3x^2 + 1$$
$$A_1(x) = -2x^3 + 3x^2$$
$$B_0(x) = x^3 - 2x^2 + x$$
$$B_1(x) = x^3 - x^2$$

It is this form that Armit's program "Multi-Patch Design Program" uses. (4)

A Program for Fitting

In dealing with curves and surfaces there can be two basic ways of treating the surface: fitting or designing. Fitting is the problem, given the existing shape, of obtaining a mathematical representation, or perhaps simplifying a known complex analytic expression of a shape. Designing involves creating a shape from scratch or perhaps modifying an existing shape and its mathematical description.

The aim of the work described below was to tie together fitting a surface to a curvilinear grid of data points using the program "Multi-Patch Design Program" (MPDP) written by Armit of the C.A.D. group, Cambridge, for designing surfaces from scratch using a PDP-7 and associated display.

Previous attempts to use the MPDP to fit surfaces to point data have(5) proven unsatisfactory: the methods were long, tedious, and did not give really smooth surface fits. Given that there are many mechanical engineering cases where a design would proceed by first fitting a curve or surface and then refining the shape, using design techniques, it is obvious that a surface fitting method compatible with MPDP would be of great benefit.

The program OADA/REM/SUBROUTS was written to obtain the boundary condition matrices B of a quilt of patches covering the grid of data points on the surface. Th method is limited in that the data points must lie on the corner points of the patche and must thus form a curvilinear grid pattern in x, y, z space (a rectilinear grid on the t, u plane).

Forrest provided a simple method of calculating the tangent vector based on the F-Mesh,(6) devised to fit a surface to an array of data points homeomorphic to a rectangular grid of points.

For tangent vectors there are four possible cases

i) only one point on line - error message

ii) only two points on line

$$P'(1) = P'(2) = P(2) - P(1)$$

iii) three or more points on line, tangent vectors at ends. Computed to make ends parabolas.

$$P'(1) = 2\{P(2) - P(1)\} - P'(2)$$

and $$P'(m) = 2\{P(m) - P(m-1)\} - P'(m-1)$$

where $P(1)$ and $P(m)$ are at ends.

iv) three or more points on line, tangent vectors at interior points

$$P'(i) = \frac{P(i+1) - P(i-1)}{|P(i+1) - P(i-1)|} \times \text{min. of } \left\{ |P(i+1) - P(i)|, \quad |P(i) - P(i-1) \right.$$

For the cross derivative or twist vector at each corner, a formula suggested by Birkhoff and de Boor (7) is used

$$P''(ij) = \frac{1}{4}\left[P(i+1, j+1) + P(i-1, j-1) - P(i+1, j-1) - P(i-1, j+1)\right]$$

If any of these four surrounding points is not specified, its coordinate is interpolat linearly from the two closest points. (For instance, in the case of $P(i+1, j+1)$, from $P(i+1, j)$ and $P(i, j+1)$). If either of these is not specified, the twist vector is set to zero. The effect of the approximation made in the linear interpolation is found to be negligible, as the value of the vector is not much changed, and the surface is not very sensitive to the value of the twist vector at any rate.

The data read in are

i) the number of rows of the rectangle in which the surface grid fits, the number of columns, (format: 2 I 4);

ii) the key element to each point, row by row. This is set to 1. if the coordinates are specified,
+1 if the horizontal tangent vectors of the point are specified
+2 " " vertical " " " " " " "
+4 " " twist " " " " " "
(format: 18F 4.0) - Note that the grid may appear as in Fig. II;

iii) Point by point, row by row, the x, y, and z coordinates of each data point specified are read, (format: 9F7.1) and the specified tangent or twist vectors, if any, are read (format: 9F7.1).

This is all the data needed.

The output is

i) row by row the 13 elements (the key element, the three coordinate components, the nine vector components) are written;

ii) row by row the boundary conditions matrices B of each bicubic patch are written. The convention of writing the vectors is as follows:

for a given point C (Diagram I) the matrix is written as -

x components				y components				z components			
C_c	S_c	C_V	S_V								
E_c	SE_c	E_V	SE_V								
C_H	S_H	C_T	S_T								
E_H	SE_H	E_T	SE_T								

where the points C, S, E and SE are shown in Diag.I and where the subscripts C, H, V and T stand for position, horizontal(ie. along row), vertical (ie. along column), and twist vectors respectively. Horizontal and vertical vectors are positive in the directions of increasing columns and rows respectively.

A more detailed summary of the internal data structure is given in Appendix A.

The tape punching program CADA/REM/BINARY is described in Appendix B.

Results

Coons[2] shows that it is possible to choose a parametric cubic which very nearly approximates a circle for one quadrant. Consider Figure III

$$x = [u^3 \; u^2 \; u \; 1] \; M \begin{bmatrix} 0 \\ 1 \\ a \\ 0 \end{bmatrix}$$ where the column vector is the end conditions

Then for $u = \frac{1}{2}$ this becomes

$$x = \tfrac{1}{8}[1 \; 2 \; 4 \; 8] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ a \\ 0 \end{bmatrix}$$

Solving $a = 8x - 4$
But at $u = \frac{1}{2}$, $x = \frac{\sqrt{2}}{2}$, by $\tan(\pi/4)$.

thus $a = 4(\sqrt{2} - 1) = 1.656$
and $a^2 = 2.752$

Coons shows that for a quasi-sphere, using the quasi-circle as boundary curves, the boundary conditions matrix B is given by

for z components $B = \begin{bmatrix} 1 & 0 & 0 & -a \\ 1 & 0 & 0 & -a \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

for x components $B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & a & 0 \\ 0 & a & a^2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

for y components $B = \begin{bmatrix} 0 & 1 & a & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -a & -a^2 & 0 \end{bmatrix}$

where, as shown above, $a = 1.656$
$a^2 = 2.752$

Thus for a quasi-sphere of radius = 100 (see Figure IV)

$$B = \left[\begin{array}{cc|cc|cc|cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 100 & 166 & 0 & 100 & 0 & 0 & -166 \\ 0 & 100 & 166 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & -166 \\ \hline 0 & 166 & 275 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -166 & -275 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

By dividing the octant into a 20 x 20 grid, with additional rows and columns on all sides to obtain accurate interpolation of tangent and twist vectors, it was possible to obtain values of the tangent and twist vectors at the four corners of the quadran patch. By multiplying the tangent vectors by 20 and the twist vectors by 20 x 20, the values obtained from the program CADA/REM/SUBROUTS are scaled up to enable comparison with Coons' values above. (The purpose for dividing the quadrant into a 20 x 20 grid is to achieve better accuracy; the values of the tangent vectors at any point are only dependent on the position vectors of the surrounding points.- the closer these are to the point considered, <u>if the surface is regular</u>, the better the approximation.)

The boundary conditions matrix obtained thus is

$$B = \left[\begin{array}{cc|cc|cc|cc|cc|cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 100 & 157 & 0 & 100 & 0 & 0 & -157 \\ 0 & 100 & 157 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & -157 \\ 0 & 157 & 248 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -157 & -248 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

This shows not only that the program is working to a good degree of accuracy ($9/166 \simeq 6\%$, $27/275 \simeq 10\%$) with only twenty divisions, but also that the formula for the twist vector is acceptable, not bringing in any non-zero elements where they are not expected, and keeping the apparent error to within 10% with this relatively coarse grid. Subsequent display of both the results with MPDP on the PDP-7 showed that they were both very good approximations to the eye of a true quadrant.

Following this reassuring result, the data points previously taken from a wooden shoe-last by other members of the CAD Group were obtained and fed into the fitting program CADA/REM/SUBROUTS. The resulting values of twist and tangent vectors were found to be in good agreement with results obtained earlier using a trial-and-error method to design the tangent vectors by looking at the display screen to see their effects on the smoothness of the shape.

At the time of termination of the project the author was attempting to display part of the results obtained. A difficulty here is that the PDP-7 can only take about ten patches at a time and the complete shoe last needed ninety-six for full definition. However a reasonable view of the toe of the shoe was obtained using the program CADA/REM/BINARY to convert the patch data structures obtained from CADA/REM/SUBROUTS into a binary form on 8-track tape suitable for digestion by the Multi-Patch Design Program on the PDP-7.

This transformation and tape-punching program is described in Appendix B.

Appendix A - CADA/REM/SUBROUTS

The program is written in FORTRAN IV for implementation on Titan, the prototype Atlas II at the Computer Laboratory, University of Cambridge. Its purpose is to fit a quilt of boundary condition matrices B to a curvilinear grid of data points for use with the program MULTI-PATCH DESIGN PROGRAM written by Armit of the Computer-Aided Design Group in the Laboratory. As such it should be used in conjunction with the program CADA/REM/BINARY which punches the patch data in binary form on 8-track paper tape for input to the design program in the PDP-7.

Each data point in the smallest rectangular grid into which the shape can fit, whether specified or not, requires 13 elements of a 3D floating-point array, ARRAY (see Figure II)

Input is:

```
    i)  MR, number of rows    )
        NC, number of columns) format 2I4
   ii)  ARRAY(I, J, 1) the key element, row by row, where the key element
                equals 1. if the point coordinates are specified
                      +1.  "  "  horizontal tangent vectors of the point are spe
                      +2.  "  "  vertical       "        "    "  "    "    "  "f
                      +4.  "  "   twist                  "    "  "    "    "  "
  iii)  the x, y and z coordinates of each specified point and the specified
        tangent or twist vectors, if any.  Row by row.  Format for both ii)
        and iii) : 9F7.1
```

The data is held in the m x n x 13 array ARRAY. The thirteen elements for each point are the key element, the three position coordinates, the three horizontal tangen coordinates, the three vertical tangent coordinates and the three twist coordinates. Horizontal (ie. along the rows) and vertical(ie.along the columns) tangents are taken positive with increasing columns and rows respectively.

The program is split into five subroutines:
the main routine which inputs, calls the subroutines and outputs;
HORTANV, which calculates the horizontal tangent vectors;
VERTANV, which calculates the vertical tangent vectors;
TWISTV, which calculates the twist vectors; and
PATCHES, which reorganises the contents of the array ARRAY into the boundary conditions matrices B of the patches.

The main routine dimensions ARRAY, reads the rectangle dimensions, checks them, sets ARRAY to zero, reads the key elements, reads the position coordinates and any specified tangent vectors, calls the three calculating subroutines, outputs the contents of ARRAY, and calls the reorganising subroutine.

HORTANV, virtually identical to VERTANV, row by row calculates the horizontal tangent vectors, if not specified. To do this properly it looks two points on either side of the current point to determine which method of calculation to use (see main text and Diagram II).

TWISTV row by row calculates the twist vectors. To do this it must look at each of the points SE, NW, SW and NE (see Diagram III) to see whether they are specified. If one is not then its position vector is interpolated from the two points between it and C, the current point. (For instance, in the case of point SE, points S and E). If either one of these is not specified, then the twist vector is set to zero.

PATCHES sets 5 x 12 array, PATCH, to zero and writes into it the components of the boundary conditions matrix associated with the current patch. This is partitione and each partition transposed to give the structure of the main level. PATCH is read out and the next point, row by row, considered. (See "A Program for Fitting," main text).

Appendix B - CADA/REM/BINARY

The program is written in FORTRAN IV for implementation on Titan. Its purpose is to punch an eight-track binary tape of the patch data geverated by CADA/REM/SUBROUTS to enable this data to be read into the program MULTI-PATCH DESIGN PROGRAM in the PDP-7.

The form of the tape is shown in Diagram IV. The PDP-7 program demands that the two left-most holes be always punched and that the number be in 18-digit binary. This means splitting the number into three parts: the first , the significant figures in binary greater than 4096 ($=2^{12}$), the second, the significant figures in binary between 4096 and 64 ($=2^{6}$), and the third, the rest.

Each patch has associated with it fifty four numbers. The first is the (unique) number of the patch. The second specifies the number of internal lines specified (ABU, BCW). The next four are associated with such things as the numbers of the surrounding patches, their edges and cyclicity, the type of cyclicity of the patch (low or high), and continuity. These four have been set to zero in CADA/REM/BINARY.

The next sixteen are the x-components, row by row, of the boundary condition matrix B, the next sixteen numbers the y-components, and the last sixteen the z-components.

In addition, in the first two numbers are encoded such characteristics as invisibility on/off, corner labels on/off, patch and corner labels on/off, hedge-hog on/off, hedge-hog edge/full, ABU parameters edge/full, BCW parameters edge/full. In the present program all options are set at off.

There is a sign discrepancy between Coons(1) and the MPDP system (see Diagram V). This means that the signs of elements S_V, SE_V, E_H, and SE_H must be reversed (see main text).

The final number on the tape must be 777777_8.

Input is the exact output of CADA/REM/SUBROUTS : row by row, the boundary conditions matrix B of each patch is read in, format: 4(4F7.0, F10.0, 3F7.0, F10.0, 3F7.0)/.

1. Forrest, A.R. "Analytic Curves for Describing Engineering Shapes" Brunel Conference on Computer Graphics, Specialists' Session, July 1968

2. Coons, S.A. "Surfaces for Computer-Aided Design of Space Forms" M.I.T. Project MAC, MAC-TR-41, June 1967

3. Forrest, A.R. "Curves and Surfaces for Computer-Aided Design Joint C.A.D. Group Thesis, Cambridge, July 1968

4. Armit, A.P. "Multipatch Design Program - User's Guide" Cambridge C.A.D. Group Document, September 1968

5. Forrest, A.R. "Notes on Topics in Curve and Surface Fitting" Cambridge C.A.D. Group Document 36, February 1970

6. Ferguson, J.C. "Multivariate Curve Interpolation" The Boeing Company, Document D2-22504, July 1963: Also in J.A.C.M., 11(2), 221-228, April 1964

7. Birkhoff, G. and de Boor, C.R. "Piecewise Polynomial Interpolation and Approximation" in "Approximations of Function" ed. Garebedian, H.L. Elsevir, Amsterdam, 1965.

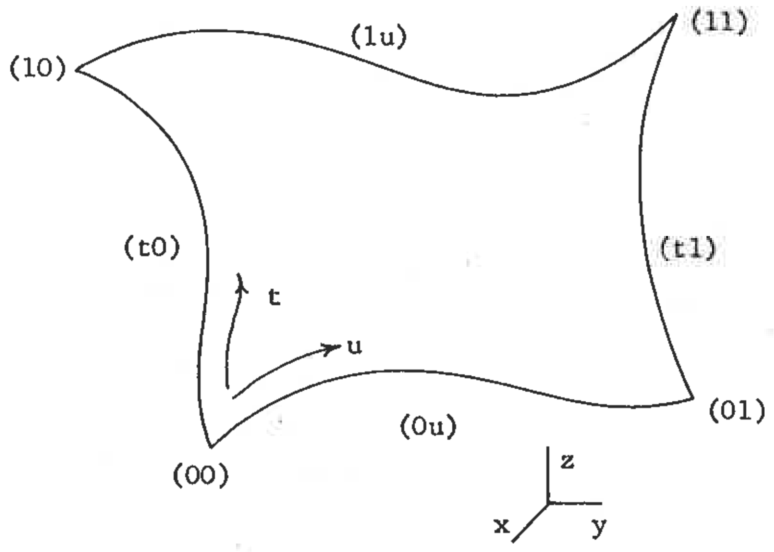


Figure I

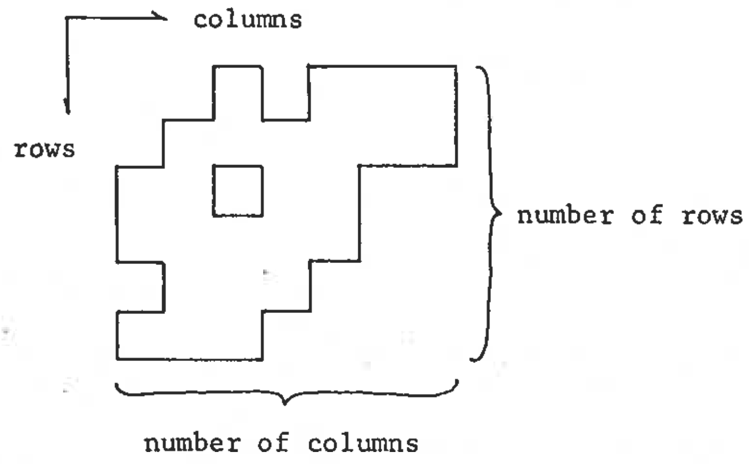


Figure II

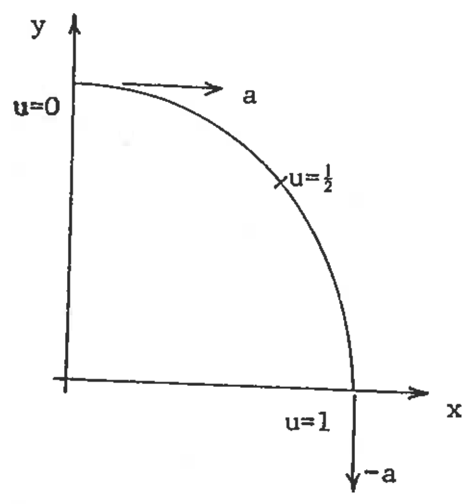
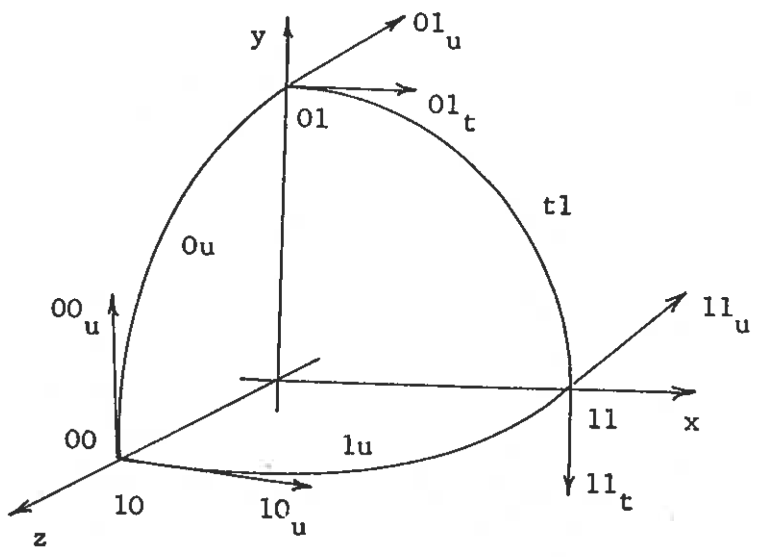


Figure III



t0 is a degenerate boundary

Figure IV

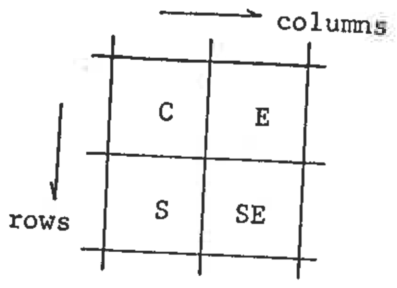


Diagram I

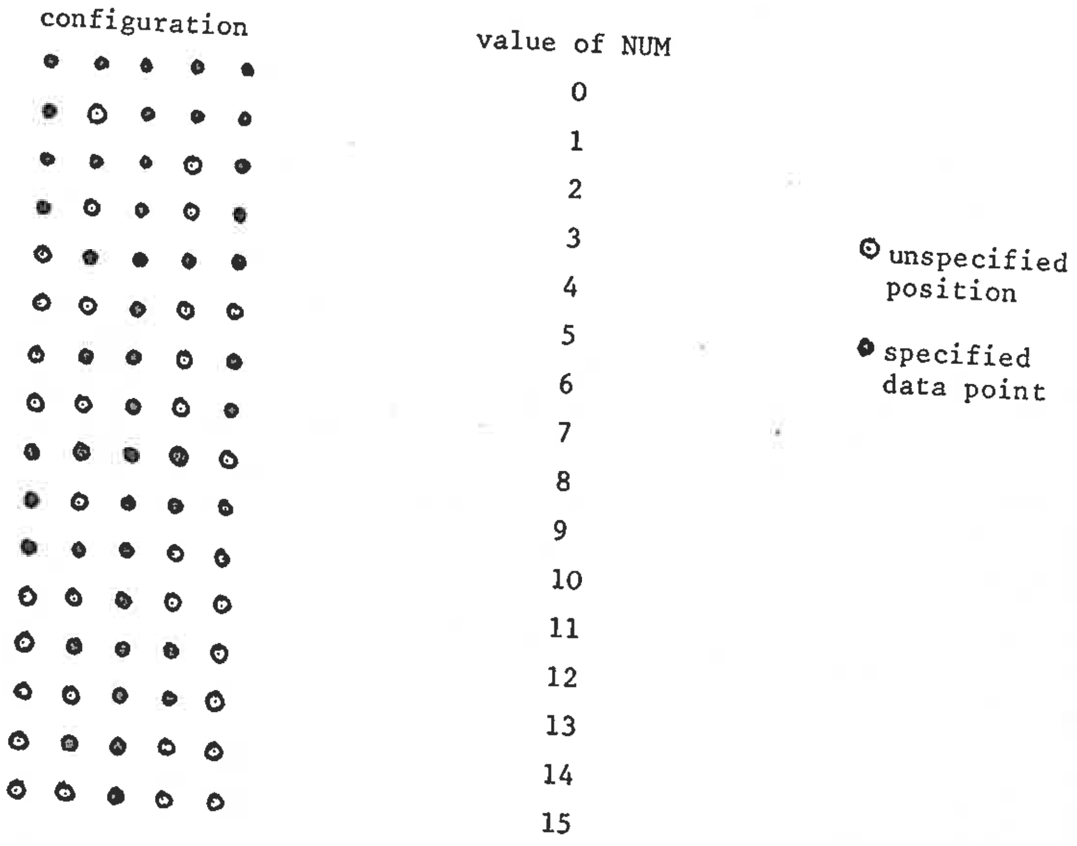


Diagram II

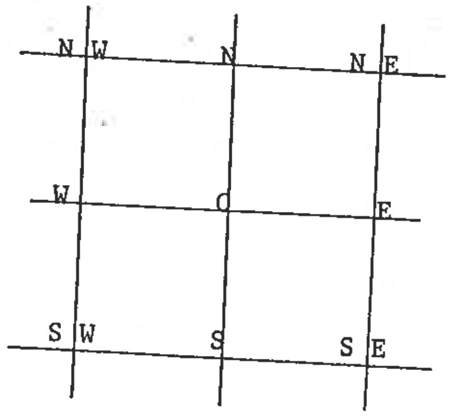


Diagram III

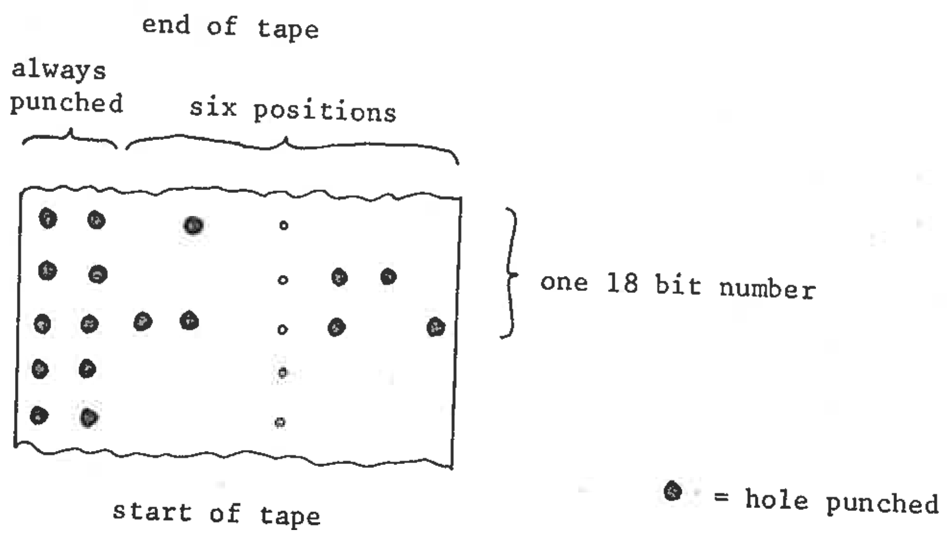


Diagram IV

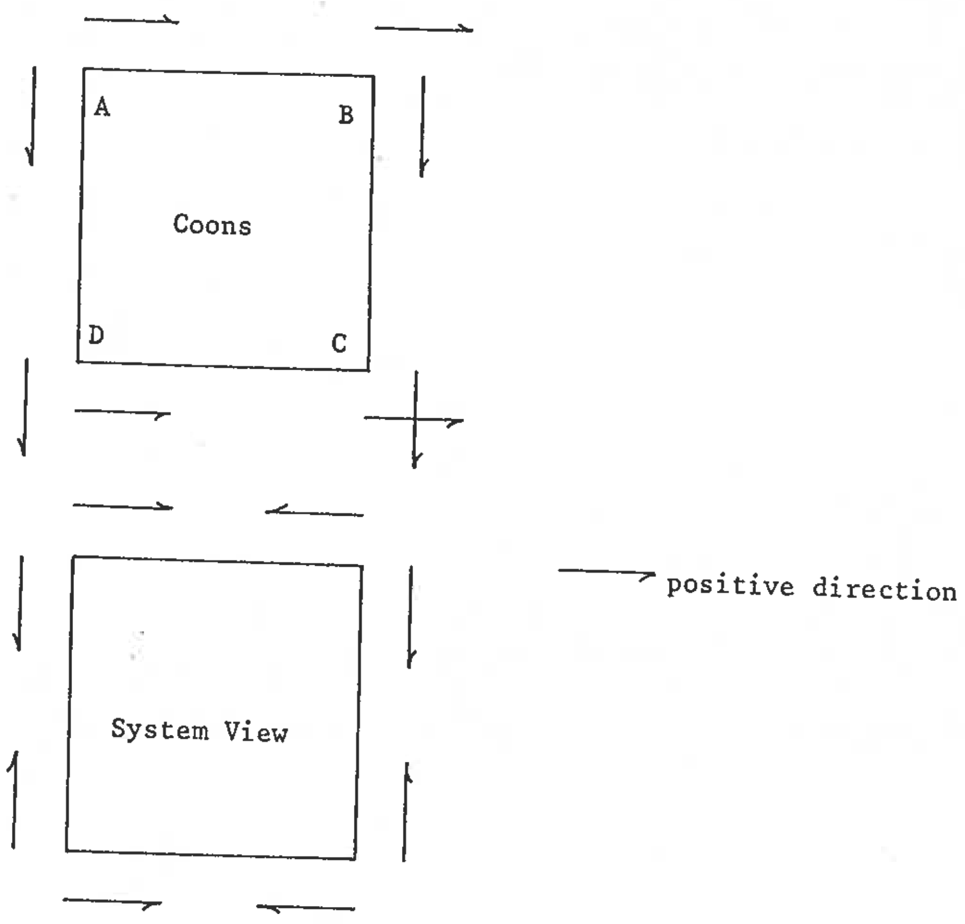


Diagram V