Curvature Continuity in Arbitrary Bicubic Bezier Patches

Final Report

Robert L. Roach

School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0150

Contract NAG-1-1039
NASA-Langley Research Center
Hampton, VA 23665-5225
Curvature Continuity in Arbitrary Bicubic Bezier Patches

Abstract

The following document outlines two methods for imposing interpatch curvature continuity in existing Bezier bicubic patch surfaces. Each method assumes that coordinates of the corners of the patches can not be altered but the interior Bezier control points can. Each method also preserves outer edge slope and outer corner twist derivatives. Neither method requires intersection or C0 continuity nor slope or C1 continuity at the start. A computer program for each method is given in the appendices.

Background

Computer-aided geometric design uses many forms of surface representation. Among the most popular to date are those which some use of cubic polynomials. The cubic polynomials can be easily manipulated in many instances and makes the difficult numerical problem of finding patch intersections more tractable than would some more complex functions. The cubic polynomial is also chosen since it is the minimum order polynomial which can satisfy curvature continuity constraints desirable in many applications [1]. Curvature continuity is important in many applications since the smoothness of a surface demands a small rates of change of curvature except at corners.

Given surfaces are frequently represented by patches consisting of smaller pieces of a whole surface. Each patch may be represented by a 2 dimensional array of coordinate points which lie on the surface. The designer then frequently needs to be able to faithfully interpolate between these given points. This brings in the frequent use of splines for surface representation and will be used here. However, the designer frequently does want to know all the spline details and would prefer to have a representation of the surfaces such that he may easily manipulate the shape. This can be done easily with Cubic Bezier surfaces, the control points of which locally control the shape of the surface. Hence the designer may "lock on" to one of the control points and "drag" it to a new location, causing a controllable distortion of the nearby surface. We next examine some detail of the bicubic Bezier patch.

In a given patch with n x m points, there are (n-1) x (m-1) subpatches. Each subpatch has four points at its corners. This may be all the data that the designer has to start with. If the overall patch is to be mated with other patches then the surface slope normal to the edge would likely be specified. Within each subpatch, a bicubic Bezier surface has 16 control points in a 4 x 4 array. The bicubic surface and all properties are completely specified by these control points since the basis functions are also specified. Four of these control points lie at the corners. Eight more lie along the edges and four are in the interior of the subpatch. Locations of the control points not on the corners control the surface curvature, slope, and position.

The Bezier representation, while simple to formulate and manipulate, does not necessarily give the user first or second derivative continuity between adjacent patches. Trying to provide such continuity by eye will be approximate at best for the slope continuity and not doable for the other. As a consequence a post processing capability is envisioned in which the user has generated the surface as close as possible to his own specifications and then his surface is modified in some minimal way to ensure slope and curvature continuity. This minimal way would most likely correspond to leaving those features intact that the designer would also choose, such as the boundary points and
particular surface points at the corners of each subpatch. Further, the outer edge slope may be important and should be kept. Thus the post processing subroutine would only change the relative positions of the subpatch interior control points and those between the subpatch corner points.

This still leaves a number of degrees of freedom all of which can be shown to be related to the manner in which the "twist" derivatives are computed. The twist derivatives are second derivatives of the components of the position vector with respect to the parameters \( s \) and \( t \), i.e. \( x_{st} \) at corners. An original method by Ferguson [1] required these twist derivatives to simply be zero. While this gives the requisite slope and curvature continuity, apparently flat spots existed at the subpatch corners. Thus, this method was not deemed suitable.

Two other methods suggest themselves. In each, nonzero twist vectors result, but only the second method specifically uses them. The methods are similar in that in each, cubic splines are first placed through all subpatch corners as will be described. They thus both seek to determine the elements of the biparametric surface cubic for each subpatch. Cubic splines through the subpatch corners determine 12 of the 16 elements. The methods differ in the next step, that of computing the remaining elements of the biparametric cubic matrices.

**Analysis**

Consider the large patch shown in Fig. 1. The patch consists of \( m \) subpatches in the \( t \) direction and \( n \) subpatches in the \( s \) direction. On each subpatch, there exists the 16 Bezier control points numbered 0-15. The numbering of the control points and the corresponding directions are consistent with the numbering system used by the NASA-Langley SMART program. The four on the corners are coincident with the corners of the subpatch and are to be retained.

The main idea of the procedures to be described is that slope and curvature continuity can be attained by first switching from a Bezier representation to a cubic spline representation for the surfaces. Cubic spline curves through data points in space have such continuity at all points. It is also fortunate that a rather convenient set of relations exist between the Bezier curves and cubic spline curves of the same order making it a simple matter to switch back and forth. By using cubic splines in both directions, it should be possible to effect the same for the surfaces. The biparametric cubic spline surface representation of a single subpatch surface is characterized by a 4x4 coefficient matrix. Once the 16 elements of this matrix are determined for each subpatch, the Bezier coefficients can be determined.

Thus, cubic splines are placed through all the subpatch corners, from one edge of the large patch to the other in both the \( t \) and \( s \) directions. The slope of the large patch around the edges is also retained as the extra information required of the ends of the cubic splines. Once the cubic splines are determined along the subpatch edges, 12 of the 16 matrix elements are known. This leaves 4 unknown and corresponds to not knowing the 4 interior Bezier control point locations. At this point, we describe two methods for determining these coefficients in such a manner that slope and curvature continuity are assured across subpatch boundaries.
Method 1. Splines Fit through Second Derivatives

It is well known that curvature is related to second derivatives. It is also known that the cubic splines through the subpatch corners provide second derivative continuity tangential to subpatch edges. What is not guaranteed is second derivative continuity normal to the edges. Thus, this method is based on putting cubic splines through the second derivatives of one parameter in the direction of the other, i.e. putting splines through $x_{tt}$ in the $s$ direction. This allows the computation of the missing elements of the biparametric cubic coefficient matrix and directly assures second derivative continuity across subpatch edges. It also turns out that if splines had been placed on $x_{ss}$ in the $t$ direction, the same result would have been obtained. A program written in QuickBasic which performs this task is given in Appendix A.

Method 2. Twist Derivative Method

In this method, the large patch corner twist derivatives, $x_{st}$, are computed from the original Bezier coefficients. Next, cubic splines are placed through $x_t$ in the $s$ direction on the outer two $s$ boundaries of the large patch. The original twist derivatives are used as slope end conditions for these two splines. With $x_{st}$ now available on these two edges, they are used as the slope end conditions of cubic splines placed through $x_s$ in the $t$ direction. The remaining twist derivatives are then computed from these splines. Knowing the twist derivatives at each of the subpatch corners allows the completion of the biparametric cubic coefficient matrices. A QuickBasic program written to effect this computation is given in Appendix B.

Computation of the New Bezier Coefficients

Once the biparametric cubic surfaces are known from either method above, standard relationships are used to compute the new Bezier control point locations. These are then returned to the in place of the original set. The programs in the appendices perform this computation. The new set has changed all Bezier control point locations except those along the large patch edges, those immediately adjacent to the outer edges, and those at all subpatch corners.

Results

Each of the subpatches in the $3 \times 3$ patch shown in Fig. 1 were originally flat surfaces with Bezier control point locations coplanar with the subpatch edges. The second method was used to generate the new set of control points which is shown in Fig. 2. The outside edges are all still nearly flat as these were left intact in the procedure. This gives the highest curvature at the vertical intersections between the subpatches. That the second derivative continuity has been accomplished is shown in Fig. 3-8. Each of these is a contour plot of lines of constant second derivative on the large patch. It can be seen that each of the contours is continuous with no breaks. There are corners on some of them indicating a lack of $C^3$ continuity at these points. These occur only at subpatch edges.
References


Fig. 1 3x3 large patch with flat subpatches

Fig. 2 New control point locations on large patch and some smoothed surface lines.
Fig. 3 xtt contours

Fig. 4 ytt contours
Fig. 5 \( z_{tt} \) contours

Fig. 6 \( x_{ss} \) contours
Fig. 7 Yss contours

Fig. 8 Zss contours
Appendix A.

Method 1. Cubic Splines through Second Derivatives
This program computes a bi-cubic interpolating function through a rectangular array of coordinate data with curvature continuity along all interior patches. This is done by fitting cubic splines along rows of points in each direction. Then, the second derivatives of the interpolating functions are "splined" in the other parametric direction.

Coordinate data
Data through which a spline is fit
# of coords in T-direction, S-direction
# of points fed to PC Spline subroutine
# of space dimensions (ie. = 3 for 3D)

Coeff's of splines for X from PCSSUB
Coeff's of splines for Y from PCSSUB
Coeff's of splines for Z from PCSSUB

Coeff's of T-lines for X
Coeff's of T-lines for Y
Coeff's of T-lines for Z

Coeff's of S-lines for X
Coeff's of S-lines for Y
Coeff's of S-lines for Z

Coeff's of biparametric patches

DIM XX(21,21), YY(21,21), ZZ(21,21)
DIM X(21), Y(21), Z(21)
DIM A(21), B(21), C(21)
DIM AX(21), BX(21), CX(21)
DIM AY(21), BY(21), CY(21)
DIM AZ(21), BZ(21), CZ(21)
DIM AXT(21,15), BXT(21,15), CXT(21,15)
DIM AYT(21,15), BYT(21,15), CYT(21,15)
DIM AZT(21,15), BZT(21,15), CZT(21,15)
DIM AXS(21,15), BXS(21,15), CXS(21,15)
DIM AYS(21,15), BYS(21,15), CYS(21,15)
DIM AZS(21,15), BZS(21,15), CZS(21,15)
DIM D(21), E(21), F(21)
DIM XTT(21,15), YTT(21,15), ZTT(21,15)
DIM XSS(21,15), YSS(21,15), ZSS(21,15)
DIM XV(16,14,14), YV(16,14,14), ZV(16,14,14), CC(4)
DIM KX(16,14,14), KY(16,14,14), KZ(16,14,14)

SCREEN 9
WINDOW (0,-2)-(2,5)
XVP = -10
YVP = 10
ZVP = 10
ND = 3
LOCATE 13,20: INPUT "Enter choice: ",ICD

IF ICD = 1 THEN
    LOCATE 15,20: INPUT "Enter data file name: ",DATNAMS
    LOCATE 17,20: PRINT USING "Reading XX,YY,and ZZ from &...."; DATNAMS

    OPEN DATNAMS$ FOR INPUT AS #1
    INPUT #1,II,JJ
    LOCATE 18,20: PRINT USING "Surface has ## x ## points..."; II,JJ
    XL = 10000: YL = 10000: ZL = 10000
    XM = -10000: YM = -10000: ZM = -10000

    FOR J = 1 TO JJ
        FOR I = 1 TO II
            INPUT #1,XX(I,J),YY(I,J),ZZ(I,J)
            IF XX(I,J) < XL THEN XL = XX(I,J)
            IF XX(I,J) > XM THEN XM = XX(I,J)
            IF YY(I,J) < YL THEN YL = YY(I,J)
            IF YY(I,J) > YM THEN YM = YY(I,J)
            IF ZZ(I,J) < ZL THEN ZL = ZZ(I,J)
            IF ZZ(I,J) > ZM THEN ZM = ZZ(I,J)
        NEXT
    NEXT
    CLOSE #1

ELSE

    LOCATE 15,20: PRINT "Generating XX,YY,and ZZ...."
    II = 5
    JJ = 5
    LOCATE 18,20: PRINT USING "Surface has ## x ## points..."; II,JJ
    XL = 10000: YL = 10000: ZL = 10000
    XM = -10000: YM = -10000: ZM = -10000

    FOR J = 1 TO JJ
        YYY = (J - 1)/(JJ - 1)
        FOR I = 1 TO II
            XXX = (I - 1)/(II - 1)
            XX(I,J) = XXX
            YY(I,J) = YYY
            R = (XXX -.5)^2 + (YYY -.5)^2
            E = EXP(-3*SQR(R))
            ZZ(I,J) = 4*(YYY -.5)^2 - 4*(XXX -.5)^2
            IF XX(I,J) < XL THEN XL = XX(I,J)
            IF XX(I,J) > XM THEN XM = XX(I,J)
            IF YY(I,J) < YL THEN YL = YY(I,J)
            IF YY(I,J) > YM THEN YM = YY(I,J)
            IF ZZ(I,J) < ZL THEN ZL = ZZ(I,J)
            IF ZZ(I,J) > ZM THEN ZM = ZZ(I,J)
        NEXT
    NEXT
    END IF

XPL = YVP + (YVP - YL)*XVP/(XL - XVP)
XPM = YVP + (YVP - YM)*XVP/(XM - XVP)
YPL = ZVP + (ZVP - ZL)*XVP/(XL - XVP)
YPM = ZVP + (ZVP - ZM)*XVP/(XM - XVP)

DXW = XPM - XPL
\[
YWM = YPM + 0.1 \cdot DYW
\]

REM------------------- DRAW COORDS IN SPACE -------------------

CLS
WINDOW (XWL, YWL)-(XWM, YWM)

FOR J = 1 TO JJ
    XP = YVP + (YVP - YY(I, J)) \cdot XVP/(XX(I, J) - XVP)
    YP = ZVP + (ZVP - ZZ(I, J)) \cdot XVP/(XX(I, J) - XVP)
    PSET (XP, YP)
FOR I = 2 TO II
    XP = YVP + (YVP - YY(I, J)) \cdot XVP/(XX(I, J) - XVP)
    YP = ZVP + (ZVP - ZZ(I, J)) \cdot XVP/(XX(I, J) - XVP)
    LINE -(XP, YP), II
NEXT
NEXT
FOR I = 1 TO II
    XP = YVP + (YVP - YY(I, 1)) \cdot XVP/(XX(I, 1) - XVP)
    YP = ZVP + (ZVP - ZZ(I, 1)) \cdot XVP/(XX(I, 1) - XVP)
    PSET (XP, YP)
FOR J = 2 TO JJ
    XP = YVP + (YVP - YY(I, J)) \cdot XVP/(XX(I, J) - XVP)
    YP = ZVP + (ZVP - ZZ(I, J)) \cdot XVP/(XX(I, J) - XVP)
    LINE -(XP, YP), II
NEXT
NEXT
DO: LOOP WHILE INKEY$ = ""

REM----------------- GET PC SPLINES THROUGH THE DATA -----------------

REM----------------- T-LINES (I-DIRECTION) -----------------

CLS
LOCATE 9, 20: PRINT "Computing splines in T-direction.."
LOCATE 10, 20: PRINT USING " (there are ## pts on each T line)"
LOCATE 12, 20: PRINT "Which end condition do you want:";
LOCATE 13, 20: PRINT 1 = Natural (x'', y'', z'' = 0)
LOCATE 14, 20: PRINT 2 = Periodic (matched slopes)
LOCATE 15, 20: PRINT 3 = Slope (specified at ends)
LOCATE 16, 20: INPUT "Enter choice: ", ICE
IF ICE = 1 THEN CASE$ = "NATURAL"
IF ICE = 2 THEN CASE$ = "PERIODIC"
IF ICE = 3 THEN CASE$ = "SLOPE"
N = II
FOR J = 1 TO JJ
    LOCATE 20, 20: PRINT USING "Now doing T-Line ##";
    FOR I = 1 TO II
        X(I) = XX(I, J)
        Y(I) = YY(I, J)
        Z(I) = ZZ(I, J)
    NEXT
GOSUB 1000
FOR I = 1 TO II - 1
    AXT(I, J) = AX(I)
BYT(I,J) = BY(I)
CYT(I,J) = CY(I)
AZT(I,J) = AZ(I)
BZT(I,J) = BZ(I)
CZT(I,J) = CZ(I)

IF J = JJ GOTO 70

KX(4,I,J) = AX(I)
KX(8,I,J) = BX(I)
KX(12,I,J) = CX(I)

KY(4,I,J) = AV(I)
KY(8,I,J) = BV(I)
KY(12,I,J) = CV(I)

KZ(4,I,J) = AZ(I)
KZ(8,I,J) = BZ(I)
KZ(12,I,J) = CZ(I)

NEXT

NEXT

REM-------- S-LINES (J-DIRECTION)

CLS

LOCATE 9,20: PRINT "Computing splines in S-direction.."
LOCATE 10,20: PRINT USING " (there are ## pts on each S line)"; JJ
LOCATE 12,20: PRINT "Which end condition do you want:"
LOCATE 13,20: PRINT " 1 - Natural (x'',y'',z'' = 0)"
LOCATE 14,20: PRINT " 2 - Periodic (matched slopes)"
LOCATE 15,20: PRINT " 3 - Slope (specified at ends)"
LOCATE 16,20: INPUT "Enter choice: ", ICE

IF ICE = 1 THEN CASE$ = "NATURAL"
IF ICE = 2 THEN CASE$ = "PERIODIC"
IF ICS = 3 THEN CASE$ = "SLOPE"

N = JJ
FOR I = 1 TO II
   LOCATE 20,20: PRINT USING "Now doing S-Line ##"; I
   FOR J = 1 TO JJ
      X(J) = XX(I,J)
      Y(J) = YY(I,J)
      Z(J) = ZZ(I,J)
   NEXT

GOSUB 1000

FOR J = 1 TO JJ - 1
   AXS(I,J) = AX(J)
   BXS(I,J) = BX(J)
   CXS(I,J) = CX(J)

   AYS(I,J) = AY(J)
   BYS(I,J) = BY(J)
   CYS(I,J) = CY(J)
IF I = II GOTO 76

KX(13,I,J) = AX(J)
KX(14,I,J) = BX(J)
KX(15,I,J) = CX(J)

KY(13,I,J) = AY(J)
KY(14,I,J) = BY(J)
KY(15,I,J) = CY(J)

KZ(13,I,J) = AZ(J)
KZ(14,I,J) = BZ(J)
KZ(15,I,J) = CZ(J)

76 NEXT

NEXT

REM----------- NOW DRAW THE CUBIC SPLINES -----------

CLS
XWL = XWL + .3*DXW
XWM = XWM - .3*DXW
YWL = YWL + .3*DYW
YWM = YWM - .3*DYW

REM WINDOW (XWL,YWL)-(XWM,YWM)

FOR J = 1 TO JJ
FOR I = 1 TO II - 1
XP = YVP + (YVP - YY(I,J))*XVP/(XX(I,J) - XVP)
YP = ZVP + (ZVP - ZZ(I,J))*XVP/(XX(I,J) - XVP)
PSET (XP,YP)
FOR T = 0 TO 1 STEP .099
XXX = XX(I,J) + ((AXT(I,J)*T + BXT(I,J))*T + CXT(I,J))*T
YYY = YY(I,J) + ((AYT(I,J)*T + BYT(I,J))*T + CYT(I,J))*T
ZZZ = ZZ(I,J) + ((AZT(I,J)*T + BZT(I,J))*T + CZT(I,J))*T
XP = YVP + (YVP - YYY)*XVP/(XXX - XVP)
YP = ZVP + (ZVP - ZZZ)*XVP/(XXX - XVP)
LINE -(XP,YP),11
NEXT
NEXT

NEXT

FOR I = 1 TO II
FOR J = 1 TO JJ - 1
XP = YVP + (YVP - YY(I,J))*XVP/(XX(I,J) - XVP)
YP = ZVP + (ZVP - ZZ(I,J))*XVP/(XX(I,J) - XVP)
PSET (XP,YP)
FOR S = 0 TO 1 STEP .099
XXX = XX(I,J) + ((AXS(I,J)*S + BXS(I,J))*S + CXS(I,J))*S
YYY = YY(I,J) + ((AYS(I,J)*S + BYS(I,J))*S + CYS(I,J))*S
ZZZ = ZZ(I,J) + ((AZS(I,J)*S + BZS(I,J))*S + CZS(I,J))*S
XP = YVP + (YVP - YYY)*XVP/(XXX - XVP)
YP = ZVP + (ZVP - ZZZ)*XVP/(XXX - XVP)
LINE -(XP,YP),11
NEXT
NEXT
NEXT

DO: LOOP WHILE INKEY$ = ""
REM-------- S-DIRECTION FOR Xtt

FOR J = 1 TO JJ
    Xtt(II,J) = 6*AXT(II - 1,J) + 2*BXT(II - 1,J)
    Ytt(II,J) = 6*AYT(II - 1,J) + 2*BYT(II - 1,J)
    Ztt(II,J) = 6*AZT(II - 1,J) + 2*BZT(II - 1,J)
FOR I = 1 TO II - 1
    Xtt(I,J) = 2*BXT(I,J)
    Ytt(I,J) = 2*BYT(I,J)
    Ztt(I,J) = 2*BZT(I,J)
NEXT

NEXT

FOR I = 1 TO II
    FOR J = 1 TO JJ
        X(J) = Xtt(I,J)
        Y(J) = Ytt(I,J)
        Z(J) = Ztt(I,J)
    NEXT
    GOSUB 1000
NEXT

FOR J = 1 TO JJ - 1
    IF I = JJ GOTO 200
    KX(5,I,J) = .5*AX(J)
    KX(6,I,J) = .5*BX(J)
    KX(7,I,J) = .5*CX(J)
    KY(5,I,J) = .5*AY(J)
    KY(6,I,J) = .5*BY(J)
    KY(7,I,J) = .5*CY(J)
    KZ(5,I,J) = .5*AZ(J)
    KZ(6,I,J) = .5*BZ(J)
    KZ(7,I,J) = .5*CZ(J)
    IF I = 1 GOTO 210
200
    KX(1,I - 1,J) = (AX(J) - 2*KX(5,I - 1,J))/6
    KX(2,I - 1,J) = (BX(J) - 2*KX(6,I - 1,J))/6
    KX(3,I - 1,J) = (CX(J) - 2*KX(7,I - 1,J))/6
    KY(1,I - 1,J) = (AY(J) - 2*KY(5,I - 1,J))/6
    KY(2,I - 1,J) = (BY(J) - 2*KY(6,I - 1,J))/6
    KY(3,I - 1,J) = (CY(J) - 2*KY(7,I - 1,J))/6
    KZ(1,I - 1,J) = (AZ(J) - 2*KZ(5,I - 1,J))/6
    KZ(2,I - 1,J) = (BZ(J) - 2*KZ(6,I - 1,J))/6
    KZ(3,I - 1,J) = (CZ(J) - 2*KZ(7,I - 1,J))/6
210
NEXT

REM---------------- GET REMAINDER OF THE K'S -----------------

FOR J = 1 TO JJ - 1
    FOR I = 1 TO II - 1
        KX(9,I,J) = AXS(I + 1,J) - AXS(I,J) - KX(1,I,J) - KX(5,I,J)
        KX(10,I,J) = BXS(I + 1,J) - BXS(I,J) - KX(2,I,J) - KX(6,I,J)
    NEXT
NEXT
KY(11, I, J) = CYS(I + 1, J) - CYS(I, J) - KY(3, I, J) - KY(7, I, J)
KZ(9, I, J) = AZS(I + 1, J) - AZS(I, J) - KZ(1, I, J) - KZ(5, I, J)
KZ(10, I, J) = BZS(I + 1, J) - BZS(I, J) - KZ(2, I, J) - KZ(6, I, J)
KZ(11, I, J) = CZS(I + 1, J) - CZS(I, J) - KZ(3, I, J) - KZ(7, I, J)

NEXT
NEXT

REM----------------- NOW DRAW SOME LINES IN SOME PATCHES -----------------

DX = .01
DY = .02
NODRW = 1
IF NODRW = 1 GOTO 450

REM---------------------- PATCH IP,JP ---------------------

SCR = 1
FOR JP = 1 TO JJ - 1
FOR IP = I TO II - 1
FOR T = 0 TO 1.01 STEP .05

XXX = XX(IP, JP)
YYY = YY(IP, JP)
ZZZ = ZZ(IP, JP)
FOR JT = 0 TO 3
   TP = T^(3 - JT)
   FOR JS = 0 TO 3
      SP = S^(3 - JS)
      K = (JS + 1) + 4*JT
      IF K > 15 GOTO 300
      XXX = XXX + KX(K, IP, JP)*TP*SP
      YYY = YYY + KY(K, IP, JP)*TP*SP
      ZZZ = ZZZ + KZ(K, IP, JP)*TP*SP
   NEXT
   XP = YVP + (YVP - YYY)*XVP/(XXX - XVP)
   YP = ZVP + (ZVP - ZZZ)*XVP/(XXX - XVP)
   IF T = 0 THEN PSET (XP, YP)
   LINE -(XP, YP), SCR
   NEXT
NEXT

FOR T = .25 TO .76 STEP .25

FOR S = 0 TO 1.01 STEP .05

XXX = XX(IP, JP)
YYY = YY(IP, JP)
ZZZ = ZZ(IP, JP)
FOR JT = 0 TO 3
   TP = T^(3 - JT)
   FOR JS = 0 TO 3
      SP = S^(3 - JS)
      K = (JS + 1) + 4*JT
      NEXT
   NEXT
320  \[ XP = YVP + (YVP - YYY) \times XVP/(XXX - XVP) \]
    \[ YP = ZVP + (ZVP - ZZZ) \times XVP/(XXX - XVP) \]
    IF S = 0 THEN PSET (XP,YP)
    LINE -(XP,YP),SCR
    NEXT
    NEXT
    NEXT
    NEXT
    
    DO: LOOP WHILE INKEY$ = ""

450 REM----------------------- NOW GET BEZIER CONTROL POINTS -----------
REM----------------------- CC -----------------------
    NO = 4
    NF = 1
    FOR IP = 1 TO NO - 1
        NF = NF*IP
    NEXT
    FOR IP = 0 TO NO - 1
        IFC = 1
        FOR JP = 1 TO IP
            IFC = IFC*JP
        NEXT
        FF = 1
        FOR JP = 1 TO NO - IP - 1
            FF = FF*JP
        NEXT
        CC(IP + 1) = NF/(IFC*FF)
    NEXT
REM----------------------- CONTROL POINTS -----------------------
    FOR J = 1 TO JJ
        FOR I = 1 TO II
            PRINT USING "Computing bazier control points for ##,##..."; I,J
REM-------- CORNERS
    XV(0,I,J) = XX(I,J)
    YV(0,I,J) = YY(I,J)
    ZV(0,I,J) = ZZ(I,J)
    XV(3,I,J) = XX(I + 1,J)
    YV(3,I,J) = YY(I + 1,J)
    ZV(3,I,J) = ZZ(I + 1,J)
    XV(12,I,J) = XX(I,J + 1)
    YV(12,I,J) = YY(I,J + 1)
    ZV(12,I,J) = ZZ(I,J + 1)
    XV(15,I,J) = XX(I + 1,J + 1)
    YV(15,I,J) = YY(I + 1,J + 1)
    ZV(15,I,J) = ZZ(I + 1,J + 1)
REM-------- ON SIDE 1 (S=0)
    XV(1,I,J) = XV(0,I,J) + CXT(I,J)/3
DYL = 3*AYT(I,J) + 2*BYT(I,J) + CYT(I,J)
DZL = 3*AZT(I,J) + 2*BZT(I,J) + CZT(I,J)

XV(2,I,J) = XV(3,I,J) - DXI/3
YV(2,I,J) = YV(3,I,J) - DYI/3
ZV(2,I,J) = ZV(3,I,J) - DZI/3

REM-------- ON SIDE 2 (S=1)

XV(13,I,J) = XV(12,I,J) + CXT(I,J + 1)/3
YV(13,I,J) = YV(12,I,J) + CYT(I,J + 1)/3
ZV(13,I,J) = ZV(12,I,J) + CZT(I,J + 1)/3

DXI = 3*AXT(I,J + 1) + 2*BXT(I,J + 1) + CXT(I,J + 1)
DYI = 3*AYT(I,J + 1) + 2*BYT(I,J + 1) + CYT(I,J + 1)
DZI = 3*AZT(I,J + 1) + 2*BZT(I,J + 1) + CZT(I,J + 1)

XV(14,I,J) = XV(15,I,J) - DX1/3
YV(14,I,J) = YV(15,I,J) - DY1/3
ZV(14,I,J) = ZV(15,I,J) - DZ1/3

REM-------- ON SIDE 3 (T=0)

XV(4,I,J) = XV(0,I,J) + CXS(I,J)/3
YV(4,I,J) = YV(0,I,J) + CYS(I,J)/3
ZV(4,I,J) = ZV(0,I,J) + CZS(I,J)/3

DXI = 3*AXS(I,J) + 2*BXS(I,J) + CXS(I,J)
DYI = 3*AYS(I,J) + 2*BYS(I,J) + CYS(I,J)
DZI = 3*AZS(I,J) + 2*BZS(I,J) + CZS(I,J)

XV(8,I,J) = XV(12,I,J) - DXI/3
YV(8,I,J) = YV(12,I,J) - DYI/3
ZV(8,I,J) = ZV(12,I,J) - DZI/3

REM-------- ON SIDE 4 (T=1)

XV(7,I,J) = XV(3,I,J) + CXS(I + 1,J)/3
YV(7,I,J) = YV(3,I,J) + CYS(I + 1,J)/3
ZV(7,I,J) = ZV(3,I,J) + CZS(I + 1,J)/3

DXI = 3*AXS(I + 1,J) + 2*BXS(I + 1,J) + CXS(I + 1,J)
DYI = 3*AYS(I + 1,J) + 2*BYS(I + 1,J) + CYS(I + 1,J)
DZI = 3*AZS(I + 1,J) + 2*BZS(I + 1,J) + CZS(I + 1,J)

XV(11,I,J) = XV(15,I,J) - DX1/3
YV(11,I,J) = YV(15,I,J) - DY1/3
ZV(11,I,J) = ZV(15,I,J) - DZ1/3

REM-------- INTERIOR POINTS

REM-------- POINT 5

XST = KX(11,I,J)
YST = KY(11,I,J)
ZST = KZ(11,I,J)

XV(5,I,J) = XV(1,I,J) + XV(4,I,J) - XV(0,I,J) + XST/9
YV(5,I,J) = YV(1,I,J) + YV(4,I,J) - YV(0,I,J) + YST/9
ZV(5,I,J) = ZV(1,I,J) + ZV(4,I,J) - ZV(0,I,J) + ZST/9

REM-------- POINT 6
XV(6, I, J) = XV(2, I, J) + XV(7, I, J) - XV(3, I, J) - XST/9
YV(6, I, J) = YV(2, I, J) + YV(7, I, J) - YV(3, I, J) - YST/9
ZV(6, I, J) = ZV(2, I, J) + ZV(7, I, J) - ZV(3, I, J) - ZST/9

REM------- POINT 9

XST = 3*KX(9, I, J) + 2*KX(10, I, J) + KX(11, I, J)
YST = 3*KY(9, I, J) + 2*KY(10, I, J) + KY(11, I, J)
ZST = 3*KZ(9, I, J) + 2*KZ(10, I, J) + KZ(11, I, J)

XV(9, I, J) = XV(8, I, J) + XV(13, I, J) - XV(12, I, J) - XST/9
YV(9, I, J) = YV(8, I, J) + YV(13, I, J) - YV(12, I, J) - YST/9
ZV(9, I, J) = ZV(8, I, J) + ZV(13, I, J) - ZV(12, I, J) - ZST/9

REM------- POINT 10

XST = XST + 9*KX(1, I, J) + 6*(KX(2, I, J) + KX(5, I, J))
XST = XST + 4*KX(6, I, J) + 3*KX(3, I, J) + 2*KX(7, I, J)
YST = YST + 9*KY(1, I, J) + 6*(KY(2, I, J) + KY(5, I, J))
YST = YST + 4*KY(6, I, J) + 3*KY(3, I, J) + 2*KY(7, I, J)
ZST = ZST + 9*KZ(1, I, J) + 6*(KZ(2, I, J) + KZ(5, I, J))
ZST = ZST + 4*KZ(6, I, J) + 3*KZ(3, I, J) + 2*KZ(7, I, J)

XV(10, I, J) = XV(11, I, J) + XV(14, I, J) - XV(15, I, J) + XST/9
YV(10, I, J) = YV(11, I, J) + YV(14, I, J) - YV(15, I, J) + YST/9
ZV(10, I, J) = ZV(11, I, J) + ZV(14, I, J) - ZV(15, I, J) + ZST/9

NEXT

REM--------- DRAW CONTROL POINTS ---------
REM
REM CLS
DX = .02
DY = .03

FOR J = 1 TO JJ
FOR I = 1 TO II
FOR IBP = 0 TO 15
XXX = XV(IBP, I, J)
YYY = YV(IBP, I, J)
ZZZ = ZV(IBP, I, J)
XP = YVP + (YVP - YYY)*XVP/(XXX - XVP)
YP = ZVP + (ZVP - ZZZ)*XVP/(XXX - XVP)
LINE (XP + DX, YP + DY)-(XP - DX, YP - DY), 11, B
NEXT
FOR IBP = 0 TO 3
FOR JBP = 0 TO 3
IB = IBP + JBP*4
XXX = XV(IB, I, J)
YYY = YV(IB, I, J)
ZZZ = ZV(IB, I, J)
XP = YVP + (YVP - YYY)*XVP/(XXX - XVP)
YP = ZVP + (ZVP - ZZZ)*XVP/(XXX - XVP)
IF JBP = 0 THEN PSET (XP, YP)
LINE -(XP, YP), 11
NEXT
FOR JBP = 0 TO 3
FOR IBP = 0 TO 3
IB = IBP + JBP*4
YP = ZVP + (ZVP - ZZZ)*XVP/(XXX - XVP)
IF IBP = 0 THEN PSET (XP,YP)
LINE -(XP,YP),11
NEXT
NEXT
NEXT
NEXT

REM----------- NOW FILL IN SURFACE -----------

FOR J = 1 TO JJ
FOR I = 1 TO II
NLP = 11
FOR S = .25 TO .76 STEP .25
FOR IP = 1 TO NLP
T = (IP - 1)/(NLP - 1)
XXX = XV(0,I - 1,J - 1)
YYY = YV(0,I - 1,J - 1)
ZZZ = ZV(0,I - 1,J - 1)
FOR L1 = 0 TO 3
L = L1 + 1
B2 = CC(L)*S^(L - 1)*(1 - S)^(NO - L)
FOR K1 = 0 TO 3
K = K1 + 1
IBP = K1 + L1*4
B1 = CC(K)*T^(K - 1)*(1 - T)^(NO - K)
XXX = XXX + B1*B2*XV(IBP,I,J)
YYY = YYY + B1*B2*YV(IBP,I,J)
ZZZ = ZZZ + B1*B2*ZV(IBP,I,J)
NEXT
NEXT
XD = YVP + (YVP - YYY)*XVP/(XXX - XVP)
YD = ZVP + (ZVP - ZZZ)*XVP/(XXX - XVP)
IF IP = 1 THEN PSET (XD,YD)
LINE -(XD,YD),10
NEXT
NEXT

FOR T = .25 TO .76 STEP .25
FOR IP = 1 TO NLP
S = (IP - 1)/(NLP - 1)
XXX = XV(0,I - 1,J - 1)
YYY = YV(0,I - 1,J - 1)
ZZZ = ZV(0,I - 1,J - 1)
FOR L1 = 0 TO 3
L = L1 + 1
B2 = CC(L)*S^(L - 1)*(1 - S)^(NO - L)
FOR K1 = 0 TO 3
K = K1 + 1
IBP = K1 + L1*4
B1 = CC(K)*T^(K - 1)*(1 - T)^(NO - K)
XXX = XXX + B1*B2*XV(IBP,IP,JP)
NEXT
NEXT
XD = YVP + (YVP - YYY)*XVP/(XXX - XVP)
YD = ZVP + (ZVP - ZZZ)*XVP/(XXX - XVP)
IF IP = 1 THEN PSET (XD,YD)
LINE -(XD,YD),10
NEXT
NEXT
This subroutine takes the \( N \) coordinates in the arrays \( X, Y, \) and \( Z, \) and generates the coefficients \( AX, BX, CX, AY, BY, CY, AZ, BZ, CZ \) of the corresponding cubic spline through the data.

\[
\begin{align*}
T1 &= \text{TIMER} \\
\text{FOR KKK} &= 1 \text{ TO ND} \\
\text{FOR IT} &= 2 \text{ TO } N - 2 \\
C(\text{IT}) &= 1 \\
B(\text{IT}) &= 4 \\
A(\text{IT}) &= 1 \\
\text{NEXT} \\
\text{REM RHS} \\
X(0) &= X(N): Y(0) = Y(N): Z(0) = Z(N) \\
\text{FOR IT} &= 1 \text{ TO } N - 1 \\
\text{IF KKK} &= 1 \text{ THEN} \\
DD &= X(\text{IT} + 1) - 2*X(\text{IT}) + X(\text{IT} - 1) \\
\text{ELSEIF KKK} &= 2 \text{ THEN} \\
DD &= Y(\text{IT} + 1) - 2*Y(\text{IT}) + Y(\text{IT} - 1) \\
\text{ELSE} \\
DD &= Z(\text{IT} + 1) - 2*Z(\text{IT}) + Z(\text{IT} - 1) \\
\text{END IF} \\
D(\text{IT}) &= 3*DD \\
\text{NEXT} \\
\text{REM CASES$ = "NATURAL"} \\
\text{IF CASES$} &= \text{"NATURAL" THEN} \\
B(1) &= 1 \\
A(1) &= 0 \\
D(1) &= 0 \\
C(N - 1) &= 1 \\
B(N - 1) &= 4 \\
\text{GOTO 2040} \\
\text{END IF} \\
\text{REM CASES$ = "PERIODIC"} \\
\text{IF CASES$} &= \text{"PERIODIC" THEN} \\
B(1) &= 4 \\
A(1) &= 1 \\
F(1) &= 1 \\
E(1) &= 1 \\
C(N - 1) &= 1 \\
B(N - 1) &= 4 \\
\text{FOR IT} &= 2 \text{ TO } N - 1 \\
E(\text{IT}) &= 0 \\
F(\text{IT}) &= 0 \\
\text{END IF}
REM---------------- CASES = "SLOPE" -------------------

IF CASES = "SLOPE" THEN

END IF

2040 REM----------------- SOLVE MATRIX ---------------------

IF CASES = "PERIODIC" THEN
  GOSUB 2100
ELSE
  GOSUB 2000
END IF

REM---------------- NOW GET COEFS ------------------------

IF KKK = 1 THEN
  FOR IT = 1 TO N - 1
    BX(IT) = D(IT)
  NEXT
  FOR IT = 1 TO N - 2
    AX(IT) = (BX(IT + 1) - BX(IT))/3#
    CX(IT) = X(IT + 1) - X(IT) - AX(IT) - BX(IT)
  NEXT
  CX(N - 1) = 3*AX(N - 2) + 2*BX(N - 2) + CX(N - 2)
  AX(N - 1) = X(N) - BX(N - 1) - CX(N - 1) - X(N - 1)
ELSEIF KKK = 2 THEN
  FOR IT = 1 TO N - 1
    BY(IT) = D(IT)
  NEXT
  FOR IT = 1 TO N - 2
    AY(IT) = (BY(IT + 1) - BY(IT))/3#
    CY(IT) = Y(IT + 1) - Y(IT) - AY(IT) - BY(IT)
  NEXT
  CY(N - 1) = 3*AY(N - 2) + 2*BY(N - 2) + CY(N - 2)
  AY(N - 1) = Y(N) - BY(N - 1) - CY(N - 1) - Y(N - 1)
ELSE
  FOR IT = 1 TO N - 1
    BZ(IT) = D(IT)
  NEXT
  FOR IT = 1 TO N - 2
    AZ(IT) = (BZ(IT + 1) - BZ(IT))/3#
    CZ(IT) = Z(IT + 1) - Z(IT) - AZ(IT) - BZ(IT)
  NEXT
  CZ(N - 1) = 3*AZ(N - 2) + 2*BZ(N - 2) + CZ(N - 2)
  AZ(N - 1) = Z(N) - BZ(N - 1) - CZ(N - 1) - Z(N - 1)
END IF

NEXT

T2 = TIMER

RETURN

END

2000 REM---------- SUBROUTINE TSOLV ----------

FOR IT = 2 TO N - 1
  CBI = C(IT)/B(IT - 1)
D(N - 1) = D(N - 1)/B(N - 1)

FOR IR = 2 TO N - 1
    IT = N - IR + 1
    D(IT) = (D(IT) - A(IT)*D(IT + 1))/B(IT)
NEXT
RETURN

2100 REM--------- SUBROUTINE TSOLVP ---------
REM
REM This routine is used for periodic tridiagonal systems
REM
REM-----------------------------------------------

FOR IT = 2 TO N - 2
    CBI = C(IT)/B(IT - 1)
    B(IT) = B(IT) - CBI*A(IT - 1)
    D(IT) = D(IT) - CBI*D(IT - 1)
    EBI = E(IT - 1)/B(IT - 1)
    E(IT) = E(IT) - EBI*A(IT - 1)
    F(IT) = F(IT) - EBI*F(IT - 1)
    D(N - 1) = D(N - 1) - EBI*D(IT - 1)
NEXT

CBI = C(N - 2)/B(N - 3)
B(N - 2) = B(N - 2) - CBI*A(N - 3)
A(N - 2) = A(N - 2) - CBI*F(N - 3)
D(N - 2) = D(N - 2) - CBI*D(N - 3)
EBI = E(N - 3)/B(N - 3)
C(N - 1) = C(N - 1) - EBI*A(N - 3)
B(N - 1) = B(N - 1) - EBI*F(N - 3)
D(N - 1) = D(N - 1) - EBI*D(N - 3)
CBI = C(N - 1)/B(N - 2)
B(N - 1) = B(N - 1) - CBI*A(N - 2)
D(N - 1) = D(N - 1) - CBI*D(N - 2)
F(N - 1) = 0
F(N - 2) = 0

D(N) = D(N)/B(N)

FOR IR = 2 TO N
    IT = N - IR + 1
    D(IT) = (D(IT) - A(IT)*D(IT + 1) - F(IT)*D(N))/B(IT)
NEXT
RETURN
Appendix B.

Method 2. Twist Derivative Method
REM PROGRAM BPCS4-Bi-Parametric Cubic Spline V.4
REM--------------------------------------------
REM
REM This program replaces an array of Cubic Bezier patch
REM control points with another set which possess gradient and
REM curvature continuity across all patch boundaries.
REM This is done by fitting cubic splines along rows of
REM points in each direction. This provides C2 continuity along
REM patch boundaries. Twist derivatives are found on every corner
REM by fitting splines through the t-derivative of the cubics in
REM the s-direction.
REM
REM V.4 No graphics version.
REM
REM NPT       # of patches in the t-direction
REM NPS       # of patches in the s-direction
REM N         # of points fed to PC Spline subroutine
REM ND        # of space dimensions (ie. = 3 for 3D)
REM XV,YV,ZV  Bezier Control point coordinates
REM XX,YY,ZZ  Coordinate data from input patch corners
REM X,Y,Z     Data through which a spline is fit
REM II,JJ     # of coords in T-direction,S-direction
REM (II = NPT+1,JJ = NPS+1)
REM AX,BX,CX  Coeff’s of splines for X from PCSSUB
REM AY,BY,CY  Coeff’s of splines for Y from PCSSUB
REM AZ,BZ,CZ  Coeff’s of splines for Z from PCSSUB
REM FX,FY,FZ  Coeff’s of tensor-product matrix
REM--------- USEFUL STUFF ---------------------

DEFDBL A-H,O-Z
DEFINT I-N

ID = 16: JD = 7: KD = 7
DIM XX(ID,JD),YY(JD,KD),ZZ(ID,JD)
DIM X(ID),Y(JD),Z(KD)
DIM A(ID),B(ID),C(ID)
DIM AX(ID),BX(ID),CX(ID)
DIM AY(JD),BY(JD),CY(JD)
DIM AZ(KD),BZ(KD),CZ(KD)
DIM XV(ID,JD,KD),YV(ID,JD,KD),ZV(ID,JD,KD)
DIM FX(ID,JD,KD),FY(ID,JD,KD),FZ(ID,JD,KD)

REM--------- COORD DATA ---------------------

ND = 3

REM--------- COORD DATA ---------------------

DATNAM$ = "PATCHES"
PRINT USING "Reading Bezier control points from &......"; DATNAM$

OPEN DATNAM$ FOR INPUT AS #1
INPUT #1,NPT,NPS

PRINT USING "There are ## x ## patches..."; NPT,NPS

FOR J = 1 TO NPS
CLOSE #1

II = NPT + 1
JJ = NPS + 1

REM---------------- GET TWIST VECTORS ON OUTERMOST CORNERS ----------------

REM---- Corner at 0,0

FX(10,1,1) = 9*(XV(0,1,1) - XV(1,1,1) - XV(4,1,1) + XV(5,1,1))
FY(10,1,1) = 9*(YV(0,1,1) - YV(1,1,1) - YV(4,1,1) + YV(5,1,1))
FZ(10,1,1) = 9*(ZV(0,1,1) - ZV(1,1,1) - ZV(4,1,1) + ZV(5,1,1))

REM---- Corner at NPT,0

FX(11,NPT,1) = 9*(XV(2,NPT,1) - XV(3,NPT,1) - XV(6,NPT,1) + XV(7,NPT,1))
FY(11,NPT,1) = 9*(YV(2,NPT,1) - YV(3,NPT,1) - YV(6,NPT,1) + YV(7,NPT,1))
FZ(11,NPT,1) = 9*(ZV(2,NPT,1) - ZV(3,NPT,1) - ZV(6,NPT,1) + ZV(7,NPT,1))

REM---- Corner at 0,NPS

FX(14,1,NPS) = 9*(XV(8,1,NPS) - XV(9,1,NPS) - XV(12,1,NPS) + XV(13,1,NPS))
FY(14,1,NPS) = 9*(YV(8,1,NPS) - YV(9,1,NPS) - YV(12,1,NPS) + YV(13,1,NPS))
FZ(14,1,NPS) = 9*(ZV(8,1,NPS) - ZV(9,1,NPS) - ZV(12,1,NPS) + ZV(13,1,NPS))

REM---- Corner at NPT,NPS

K = NPT
L = NPS

FX(15,K,L) = 9*(XV(10,K,L) - XV(11,K,L) - XV(14,K,L) + XV(15,K,L))
FY(15,K,L) = 9*(YV(10,K,L) - YV(11,K,L) - YV(14,K,L) + YV(15,K,L))
FZ(15,K,L) = 9*(ZV(10,K,L) - ZV(11,K,L) - ZV(14,K,L) + ZV(15,K,L))

END IF

REM---------------- GET CORNERS -----------------------------

FOR J = 1 TO JJ-1
  FOR I = 1 TO II-1
    XX(I,J) = XV(0,I,J)
    YY(I,J) = YV(0,I,J)
    ZZ(I,J) = ZV(0,I,J)
  NEXT
  XX(II,J) = XV(3,NPT,J)
  YY(II,J) = YV(3,NPT,J)
  ZZ(II,J) = ZV(3,NPT,J)
NEXT

FOR I = 1 TO II-1
  XX(I,JJ) = XV(12,I,NPS)
  YY(I,JJ) = YV(12,I,NPS)
  ZZ(I,JJ) = ZV(12,I,NPS)
NEXT

XX(II,JJ) = XV(15,NPT,NPS)
FOR I = 1 TO NPT
  FX(0,I,J) = XX(I,J)
  FX(1,I,J) = XX(I+1,J)
  FX(4,I,J) = XX(I,J+1)
  FX(5,I,J) = XX(I+1,J+1)

  FY(0,I,J) = YY(I,J)
  FY(1,I,J) = YY(I+1,J)
  FY(4,I,J) = YY(I,J+1)
  FY(5,I,J) = YY(I+1,J+1)

  FZ(0,I,J) = ZZ(I,J)
  FZ(1,I,J) = ZZ(I+1,J)
  FZ(4,I,J) = ZZ(I,J+1)
  FZ(5,I,J) = ZZ(I+1,J+1)

NEXT
NEXT

REM----------------- GET PC SPLINES THROUGH THE DATA -----------------

REM---------------- T-LINES (I-DIRECTION)

CASE$ = "BEZIER"

N = II
FOR J = 1 TO JJ
  IF J = JJ THEN
    SIX = 3*(XV(13,I,NPS)-XV(12,I,NPS))
    SIY = 3*(YV(13,I,NPS)-YV(12,I,NPS))
    SIZ = 3*(ZV(13,I,NPS)-ZV(12,I,NPS))
    S2X = 3*(XV(15,NPT,NPS)-XV(14,NPT,NPS))
    S2Y = 3*(YV(15,NPT,NPS)-YV(14,NPT,NPS))
    S2Z = 3*(ZV(15,NPT,NPS)-ZV(14,NPT,NPS))
  ELSE
    SIX = 3*(XV(I,I,J)-XV(0,I,J))
    SIY = 3*(YV(I,I,J)-YV(0,I,J))
    SIZ = 3*(ZV(I,I,J)-ZV(0,I,J))
    S2X = 3*(XV(3,NPT,J)-XV(2,NPT,J))
    S2Y = 3*(YV(3,NPT,J)-YV(2,NPT,J))
    S2Z = 3*(ZV(3,NPT,J)-ZV(2,NPT,J))
  END IF

FOR I = 1 TO II
  X(I) = XX(I,J)
  Y(I) = YY(I,J)
  Z(I) = ZZ(I,J)
NEXT

GOSUB 1000

FOR I = 1 TO II-1
  IF J = 1 GOTO 68

  FX(6,I,J-1) = CX(I)
  FY(6,I,J-1) = CY(I)
  FZ(6,I,J-1) = CZ(I)

  FX(7,I,J-1) = 3*AX(I) + 2*BX(I) + CX(I)
  FY(7,I,J-1) = 3*AY(I) + 2*BY(I) + CY(I)
FX(2,I,J) = CX(I)
FY(2,I,J) = CY(I)
FZ(2,I,J) = CZ(I)

FX(3,I,J) = 3*AX(I) + 2*BX(I) + CX(I)
FY(3,I,J) = 3*AY(I) + 2*BY(I) + CY(I)
FZ(3,I,J) = 3*AZ(I) + 2*BZ(I) + CZ(I)

NEXT

REM---------- S-LINES (J-DIRECTION)

CASE$ = "BEZIER"

N = JJ
FOR I = 1 TO II
IF I = II THEN
   S1X = 3*(XV(7,NPT,1)-XV(3,NPT,1))
   S1Y = 3*(YV(7,NPT,1)-YV(3,NPT,1))
   S1Z = 3*(ZV(7,NPT,1)-ZV(3,NPT,1))
   S2X = 3*(XV(15,NPT,NPS)-XV(11,NPT,NPS))
   S2Y = 3*(YV(15,NPT,NPS)-YV(11,NPT,NPS))
   S2Z = 3*(ZV(15,NPT,NPS)-ZV(11,NPT,NPS))
ELSE
   S1X = 3*(XV(4,I,1)-XV(0,I,1))
   S1Y = 3*(YV(4,I,1)-YV(0,I,1))
   S1Z = 3*(ZV(4,I,1)-ZV(0,I,1))
   S2X = 3*(XV(12,I,NPS)-XV(8,I,NPS))
   S2Y = 3*(YV(12,I,NPS)-YV(8,I,NPS))
   S2Z = 3*(ZV(12,I,NPS)-ZV(8,I,NPS))
END IF
FOR J = 1 TO JJ
   X(J) = XX(I,J)
   Y(J) = YY(I,J)
   Z(J) = ZZ(I,J)
NEXT
GOSUB 1000

FOR J = 1 TO JJ-1
IF I = II GOTO 74
   FX(8,I,J) = CX(J)
   FY(8,I,J) = CY(J)
   FZ(8,I,J) = CZ(J)
   FX(12,I,J) = 3*AX(J) + 2*BX(J) + CX(J)
   FY(12,I,J) = 3*AY(J) + 2*BY(J) + CY(J)
   FZ(12,I,J) = 3*AZ(J) + 2*BZ(J) + CZ(J)
74 IF I = 1 GOTO 76
   FX(9,I-1,J) = CX(J)
   FY(9,I-1,J) = CY(J)
   FZ(9,I-1,J) = CZ(J)
   FX(13,I-1,J) = 3*AX(J) + 2*BX(J) + CX(J)
   FY(13,I-1,J) = 3*AY(J) + 2*BY(J) + CY(J)
NEXT

REM------- NEXT PUT SPLINES THROUGH THE FIRST DERIVATIVES -------

REM------- PUT SPLINE THROUGH S-DERIV'S ALONG S=0 & S=I -------

REM------------ S=0 (J=1) EDGE

N = II
CASE$ = "BEZIER"
S1X = FX(10,1,1)
S1Y = FY(10,1,1)
S1Z = FZ(10,1,1)
S2X = FX(11,NPT,1)
S2Y = FY(11,NPT,1)
S2Z = FZ(11,NPT,1)
FOR I = 1 TO NPT
    X(I) = FX(8,I,1)
    Y(I) = FY(8,I,1)
    Z(I) = FZ(8,I,1)
NEXT
X(II) = FX(9,NPT,1)
Y(II) = FY(9,NPT,1)
Z(II) = FZ(9,NPT,1)
GOSUB 1000
FOR I = 1 TO NPT
    IF I = 1 GOTO 412
    FX(10,I,1) = CX(I)
    FY(10,I,1) = CY(I)
    FZ(10,I,1) = CZ(I)
412  IF I = NPT GOTO 414
    FX(11,I,1) = 3*AX(I) + 2*BX(I) + CX(I)
    FY(11,I,1) = 3*AY(I) + 2*BY(I) + CY(I)
    FZ(11,I,1) = 3*AZ(I) + 2*BY(I) + CZ(I)
414    NEXT

REM------------- S=1 (J=JJ) EDGE

S1X = FX(14,1,NPS)
S1Y = FY(14,1,NPS)
S1Z = FZ(14,1,NPS)
S2X = FX(15,NPT,NPS)
S2Y = FY(15,NPT,NPS)
S2Z = FZ(15,NPT,NPS)
FOR I = 1 TO NPT
X(II) = FX(13,NPT,NPS)
Y(II) = FY(13,NPT,NPS)
Z(II) = FZ(13,NPT,NPS)

GOSUB 1000

FOR I = 1 TO NPT

IF I = 1 GOTO 422

FX(14,I,NPS) = CX(I)
FY(14,I,NPS) = CY(I)
FZ(14,I,NPS) = CZ(I)

422 IF I = NPT GOTO 424

FX(15,I,NPS) = 3*AX(I) + 2*BX(I) + CX(I)
FY(15,I,NPS) = 3*AY(I) + 2*BY(I) + CY(I)
FZ(15,I,NPS) = 3*AZ(I) + 2*BZ(I) + CZ(I)

424 NEXT

REM------- NOW SPLINE T-DERIV'S IN S-DIRECTION

FOR I = 1 TO II

IF I = II THEN

S1X = FX(11,NPT,1)
S1Y = FY(11,NPT,1)
S1Z = FZ(11,NPT,1)

S2X = FX(15,NPT,NPS)
S2Y = FX(15,NPT,NPS)
S2Z = FZ(15,NPT,NPS)

ELSE

S1X = FX(10,I,1)
S1Y = FY(10,I,1)
S1Z = FZ(10,I,1)

S2X = FX(14,I,NPS)
S2Y = FX(14,I,NPS)
S2Z = FZ(14,I,NPS)

END IF

FOR J = 1 TO NPS

IF I = II THEN

X(J) = FX(3,NPT,J)
Y(J) = FY(3,NPT,J)
Z(J) = FZ(3,NPT,J)

ELSE

X(J) = FX(2,I,J)
Y(J) = FY(2,I,J)

END IF

NEXT
NEXT

IF I = II THEN

\[ X(JJ) = FX(7, NPT, NPS) \]
\[ Y(JJ) = FY(7, NPT, NPS) \]
\[ Z(JJ) = FZ(7, NPT, NPS) \]

ELSE

\[ X(JJ) = FX(6, I, NPS) \]
\[ Y(JJ) = FY(6, I, NPS) \]
\[ Z(JJ) = FZ(6, I, NPS) \]

END IF

GOSUB 1000

FOR J = 1 TO NPS

IF I = 1 GOTO 432

\[ FX(11, I-1, J) = CX(J) \]
\[ FY(11, I-1, J) = CY(J) \]
\[ FZ(11, I-1, J) = CZ(J) \]

\[ FX(15, I-1, J) = 3*AX(J) + 2*BX(J) + CX(J) \]
\[ FY(15, I-1, J) = 3*AY(J) + 2*BY(J) + CY(J) \]
\[ FZ(15, I-1, J) = 3*AZ(J) + 2*BZ(J) + CZ(J) \]

432 IF I = II GOTO 434

\[ FX(10, I, J) = CX(J) \]
\[ FY(10, I, J) = CY(J) \]
\[ FZ(10, I, J) = CZ(J) \]

\[ FX(14, I, J) = 3*AX(J) + 2*BX(J) + CX(J) \]
\[ FY(14, I, J) = 3*AY(J) + 2*BY(J) + CY(J) \]
\[ FZ(14, I, J) = 3*AZ(J) + 2*BZ(J) + CZ(J) \]

434 NEXT

NEXT

REM----------------- COMPUTE THE BEZIER CONTROL POINTS -----------------

DX = .01
DY = .02

FOR JP = 1 TO NPS
FOR IP = 1 TO NPT

REM------- CORNERS

\[ XV(0, IP, JP) = XX(IP, JP) \]
\[ YV(0, IP, JP) = YY(IP, JP) \]
\[ ZV(0, IP, JP) = ZZ(IP, JP) \]

\[ XV(3, IP, JP) = XX(IP+1, JP) \]
\[ YV(3, IP, JP) = YY(IP+1, JP) \]
\[ ZV(3, IP, JP) = ZZ(IP+1, JP) \]
XV(15, IP, JP) = XX(IP+1, JP+1)
YV(15, IP, JP) = YY(IP+1, JP+1)
ZV(15, IP, JP) = ZZ(IP+1, JP+1)

REM--------- ON SIDE 1 (S=0)

XV(1, IP, JP) = XV(0, IP, JP) + FX(2, IP, JP)/3
YV(1, IP, JP) = YV(0, IP, JP) + FY(2, IP, JP)/3
ZV(1, IP, JP) = ZV(0, IP, JP) + FZ(2, IP, JP)/3

XV(2, IP, JP) = XV(3, IP, JP) - FX(3, IP, JP)/3
YV(2, IP, JP) = YV(3, IP, JP) - FY(3, IP, JP)/3
ZV(2, IP, JP) = ZV(3, IP, JP) - FZ(3, IP, JP)/3

REM--------- ON SIDE 2 (S=1)

XV(13, IP, JP) = XV(12, IP, JP) + FX(6, IP, JP)/3
YV(13, IP, JP) = YV(12, IP, JP) + FY(6, IP, JP)/3
ZV(13, IP, JP) = ZV(12, IP, JP) + FZ(6, IP, JP)/3

XV(14, IP, JP) = XV(15, IP, JP) - FX(7, IP, JP)/3
YV(14, IP, JP) = YV(15, IP, JP) - FY(7, IP, JP)/3
ZV(14, IP, JP) = ZV(15, IP, JP) - FZ(7, IP, JP)/3

REM--------- ON SIDE 3 (T=0)

XV(4, IP, JP) = XV(0, IP, JP) + FX(8, IP, JP)/3
YV(4, IP, JP) = YV(0, IP, JP) + FY(8, IP, JP)/3
ZV(4, IP, JP) = ZV(0, IP, JP) + FZ(8, IP, JP)/3

XV(8, IP, JP) = XV(12, IP, JP) - FX(12, IP, JP)/3
YV(8, IP, JP) = YV(12, IP, JP) - FY(12, IP, JP)/3
ZV(8, IP, JP) = ZV(12, IP, JP) - FZ(12, IP, JP)/3

REM--------- ON SIDE 4 (T=1)

XV(7, IP, JP) = XV(3, IP, JP) + FX(9, IP, JP)/3
YV(7, IP, JP) = YV(3, IP, JP) + FY(9, IP, JP)/3
ZV(7, IP, JP) = ZV(3, IP, JP) + FZ(9, IP, JP)/3

XV(11, IP, JP) = XV(15, IP, JP) - FX(13, IP, JP)/3
YV(11, IP, JP) = YV(15, IP, JP) - FY(13, IP, JP)/3
ZV(11, IP, JP) = ZV(15, IP, JP) - FZ(13, IP, JP)/3

REM--------- INTERIOR POINTS

REM-------- POINT 5

XST = FX(10, IP, JP)
YST = FY(10, IP, JP)
ZST = FZ(10, IP, JP)

XV(5, IP, JP) = XV(1, IP, JP) + XV(4, IP, JP) - XV(0, IP, JP) + XST/9
YV(5, IP, JP) = YV(1, IP, JP) + YV(4, IP, JP) - YV(0, IP, JP) + YST/9
ZV(5, IP, JP) = ZV(1, IP, JP) + ZV(4, IP, JP) - ZV(0, IP, JP) + ZST/9

REM-------- POINT 6

XST = FX(11, IP, JP)
YST = FY(11, IP, JP)
ZST = FZ(11, IP, JP)
PC SPLINE SUBROUTINE

This subroutine takes the N coordinates in the arrays X, Y, and Z, and generates the coefficients AX, BX, CX, AY, BY, CY, AZ, BZ, CZ of the corresponding cubic spline through the data.

--- SET UP MATRIX ---

FOR KKK = 1 TO ND
  FOR IT = 2 TO N-2
    C(IT) = 1
    B(IT) = 4
    A(IT) = 1
  NEXT

--- RHS ---

X(0) = X(N): Y(0) = Y(N): Z(0) = Z(N)

FOR IT = 1 TO N-1
  IF KKK = 1 THEN
    DD = X(IT + 1) - 2*X(IT) + X(IT-1)
  ELSEIF KKK = 2 THEN
    DD = Y(IT + 1) - 2*Y(IT) + Y(IT-1)
  ELSE
    DD = Z(IT + 1) - 2*Z(IT) + Z(IT-1)
  END IF
  D(IT) = 3*DD
NEXT

REM------ CASE$ = "NATURAL" ---
C(N-1) = 1
B(N-1) = 4
GOTO 2040
END IF

REM-------- CASE$ = "BEZIER" ---------------

IF CASE$ = "BEZIER" THEN
B(1) = 2/3
A(1) = 1/3
B(N-1) = 7/3
C(N-1) = 2/3
IF KKK = 1 THEN
D(1) = (X(2)-X(1))-S1X
D(N-1) = 3*(X(N)-X(N-1)) - 2*(X(N-1)-X(N-2)) - S2X
ELSEIF KKK = 2 THEN
D(1) = (Y(2)-Y(1))-S1Y
D(N-1) = 3*(Y(N)-Y(N-1)) - 2*(Y(N-1)-Y(N-2)) - S2Y
ELSE
D(1) = (Z(2)-Z(1))-S1Z
D(N-1) = 3*(Z(N)-Z(N-1)) - 2*(Z(N-1)-Z(N-2)) - S2Z
END IF

END IF

2040 REM-------- SOLVE MATRIX ---------------

GOSUB 2000

REM-------- NOW GET COEFFS -----------

IF KKK = 1 THEN
FOR IT = 1 TO N-1
BX(IT) = D(IT)
NEXT
FOR IT = 1 TO N-2
AX(IT) = (BX(IT + 1)-BX(IT))/3
CX(IT) = X(IT + 1)-X(IT)-AX(IT)-BX(IT)
NEXT
CX(N-1) = 3*AX(N-2) + 2*BX(N-2) + CX(N-2)
AX(N-1) = X(N)-BX(N-1)-CX(N-1)-X(N-1)
ELSEIF KKK = 2 THEN
FOR IT = 1 TO N-1
BY(IT) = D(IT)
NEXT
FOR IT = 1 TO N-2
AY(IT) = (BY(IT + 1)-BY(IT))/3
CY(IT) = Y(IT + 1)-Y(IT)-AY(IT)-BY(IT)
NEXT
CY(N-1) = 3*AY(N-2) + 2*BY(N-2) + CY(N-2)
AY(N-1) = Y(N)-BY(N-1)-CY(N-1)-Y(N-1)
ELSE
FOR IT = 1 TO N-1
BZ(IT) = D(IT)
NEXT
FOR IT = 1 TO N-2
AZ(IT) = (BZ(IT + 1)-BZ(IT))/3
CZ(IT) = Z(IT + 1)-Z(IT)-AZ(IT)-BZ(IT)
NEXT
CZ(N-1) = 3*AZ(N-2) + 2*BZ(N-2) + CZ(N-2)
AZ(N-1) = Z(N)-BZ(N-1)-CZ(N-1)-Z(N-1)
END IF
SUBROUTINE TSOLV

FOR IT = 2 TO N-1
    CBI = C(IT)/B(IT-1)
    B(IT) = B(IT) - CBI*A(IT-1)
    D(IT) = D(IT) - CBI*D(IT-1)
NEXT

D(N-1) = D(N-1)/B(N-1)

FOR IR = 1 TO N-2
    IT = N-IR-1
    D(IT) = (D(IT) - A(IT)*D(IT + IR))/B(IT)
NEXT

RETURN