Curvature Continuity of Cubic Bezier Curves in the Solid Modeling Aerospace Research Tools Design Software

Interim Report

Robert L. Roach
John R. Forrest

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

(NASA-CR-[illegible]) CURVATURE CONTINUITY OF CUBIC BEZIER CURVES IN THE SOLID MODELING AEROSPACE RESEARCH TOOLS DESIGN SOFTWARE Interim Report (Georgia Inst. of Tech.) 1[illegible] p — N90-21[illegible]4

CSCL 09B G3/61 — Unclas 0273[illegible]

Contract NAG-1-1039
NASA-Langley Research Center
Hampton, VA 23665-5225

Curvature Continuity of Cubic Bezier Curves in the Solid Modeling Aerospace Research Tools Design Software

Interim Report

Robert L. Roach and John R. Forrest
School of Aerospace Engineering
Georgia Institutue of Technology
Atlanta, GA 30332-0150

Abstract

This report presents preliminary results of an investigation into the development of a procedure to provide curvature continuity between biparametric cubic Bezier surface patches in the computer-aided design package known as SMART (Solid Modeling Aerospace Reasearch Tools). This initial effort was aimed at providing the designer with the ability to locally impose curvature continuity at the intersection of two Bezier curves without disrupting either the curvature or slope continuity that may exist at the ends of these curves. Such a method was found if the origianl Bezier control points are all coplanar. If they are not then it is possible to find a minimum deviation from exact curvature continuity. In cases where this is not sufficient, then an entire piecewise curve must be made curvature continuous simultaneously. A method was developed based on cubic splines which is very fast. The procedure returns new Bezier control points which have both slope and curvature continuity.

Introduction

The development of computer-aided design packages have advanced to the state where it is now possible to build up a geometric description of an aerospace system in a very short time using high power graphics workstations. At the same time, it becomes desirable to have the ability to apply various numerical flow solvers to the resulting shapes to test their aerodynamic efficiency. These numerical flow solvers usually need the domain partitioned into small cells to provide a computational grid suitable for the discretized equations.

The ultimate aim of this project is provide the ability of a computer-aided design package, SMART, developed at Langley Research Center, to include the generation of finite difference grids in a computational domain around the designed surfaces. Several methods for generating such grids are available but none are designed to smooth surface data. Hence if the surface has discontinuities the grid will be generated accordingly. The subsequent computation of the metrics of a coordinate transformation by using finite differences will thus have larger errors than usual. This inaccuracy of finite difference approximation to a differentioal equation will be larger than desired. To prevent these unwanted errors, it was necessary to begin with the requirement that the design surfaces must have curvature continuity over most of the domain.

Most computer-aided design packages describe the design surfaces by combining several surface elements called patches. These patches can be geometrically described by analytic functions of various kinds. Of particular interest are cubic polynomials. Here, the cubic polynomials are cast as biparametric Bezier curves. Cubic Bezier curves use control points as weights for a series of Bernstein polynomials and provide a relatively powerful mechanism for the designer to manipulate the shape, which is done by moving the control points. In the same manner, movement of the control points can allow for postprocessing of the designed shape as we wish to do here to impose curvature continuity. Thus, after the designer creates a shape, the resulting patches can then be smoothed by manipulating the location of the patch control points.

Unfortunately, curvature continuity, along with slope continuity, requires information to be used from more than one patch. This couples all patches together requiring the simultaneous solution of new control point locations. If the number of patches is large, then this would require the solution of a rather large linear system and would likely be time-consuming even on high speed graphics workstations. It was thus desirable to investigate whether it would be possible to provide the curvature continuity for patches individually or at least with some minimal effort.

The examination of the problem took the form of first looking at how the designer actually constructs a shape and seeing if there were some way of providing a priori curvature continuity rather than at the end of the process. It seems that a significant amount of time is spent in the generation of two dimensional cross sections of the design shape. Thus cubic Bezier curves are generated describing these cross sections.

Once the cross sections are described, they are assembled axially to build up the surface. A number of axial curves are then generated to connect the cross sections. These are also piecewise cubic curves with one cubic in between each cross section. In this manner, the surface is thus described by two families of piecewise cubic curves. The surface can be seen as a collection of four-sided patches with one cubic on each side. The task of providing curvature continuity between the patches is made much easier if there already exists curvature continuity in both families of curves.

To obtain curvature continuity on these curves two tasks were identified:

Task 1- Provide a means of adjusting the Bezier control points at the intersection of two cubic Bezier curves to gain slope and curvature continuity without disrupting the same at neighboring interserctions.

Task 2 - Provide a means of ensuring curvature continuity on an entire series of cubic Bezier curves.

Both of these tasks have been completed. The remainder of this report explains the approach, method, and presents an example for each task. The remaining effort of providing curvature continuity for entire network of patches will be the subject of the final report.

Task 1 - Curvature Continuity between adjacent Cubic Bezier Curves while maintaining C1 & C2 at the ends and C1 at the junction

It is desirable, in CAD systems, when using cubic curves, to be able to attain various levels of continuity between adjacent curves. Here the desire is to attain C2 at intersections without disturbing the C1 and C2 continuity at neighboring intersections. In addition, it is assumed that the C1 continuity which already exists at the intersection will not be disturbed. By this is meant not only that the Bezier control points on either side of the intersction are colinear with it, but that the ratio of their distances from it will remain constant. This ratio is also required at the neighboring junction points and does not allow for the control points nearest to neighboring junctions to be moved. It is thus allowed only to move, in space, the intersection point in question and the two control points adjacent to it.

If such a shift of control points is possible, this would allow the designer the flexibility of selecting local junctures for enhanced continuity without having to simultaneously perform the operation at all junctures where such continuity is desired. In the present effort, it will be shown that if adjacent Bezier cubics have coplanar control points, there is a unique solution to the problem. If the control points are not all coplanar, then it is possible to find the minimum C2 discontinuity at the intersection.

Analysis

Consider the intersection of two cubic Bezier curves shown in Fig. 1. The control points are coordinate vectors denoted by b_n, b_{n-1}, etc. Shown also are the requirements for C2 continuity at b_n, ie. an auxilliary point d exists which not only is the intersection of extensions from the control polygon, but that ratios of certain lengths must also be held.

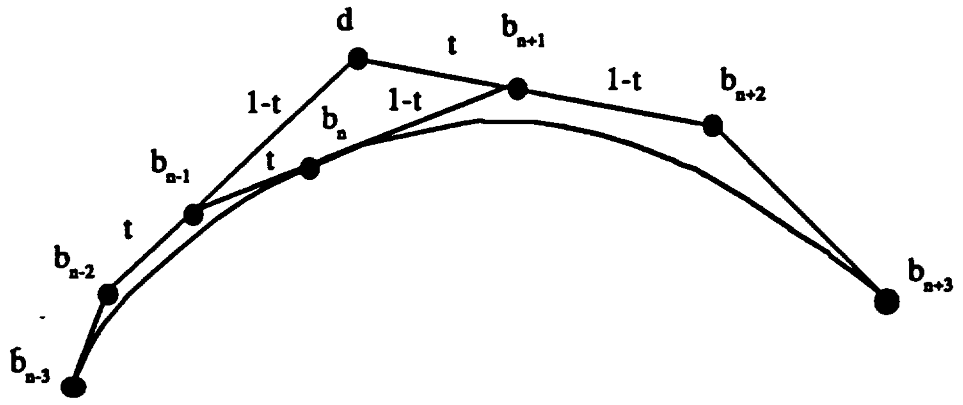

Fig. 1 Adjacent Bezier curves, associated control points and auxilliary point d. The t and 1-t are meant to indicate ratios only.

When C2 continuity does not exist, there are two points, d_1 and d_2, as shown in Fig. 2. This is the case more typically found. To gain C2 at b_n, it is necessary to move one or more of the control points so that the two points d_1 and d_2 are coincident.

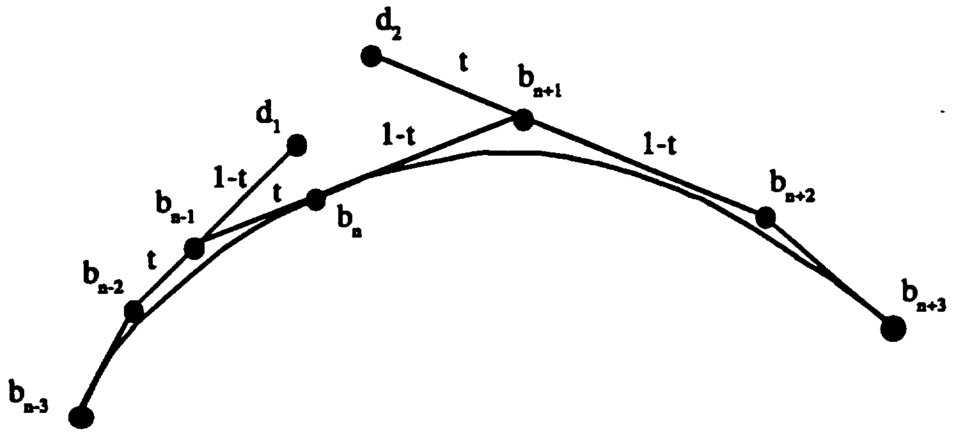

Fig. 2 Adjacent Bezier curves when C2 at the juncture does not exist. The auxilliary points d_1 and d_2 are not coincident.

This can be done if the following are satisfied:

$$b_{n-1} = (1-t)b_{n-2} + t\,d \tag{1}$$

$$b_n = (1-t)b_{n-1} + t\,b_{n+1} \tag{2}$$

$$b_{n+1} = (1-t)d + t\,b_{n+2} \tag{3}$$

To maintain C2 at the ends the following must also be satisfied:

$$b_{n-3} - 2b_{n-2} + b_{n-1} = k_1\left(b_{n-3} - 2b_{n-2} + b_{n-1}\right) \tag{4}$$

$$b_{n+3} - 2b_{n+2} + b_{n+1} = k_2\left(b_{n+3} - 2b_{n+2} + b_{n+1}\right) \tag{5}$$

Where the overbar indicates the original position. Maintenance of C1 with distance ratios at the ends means that the points b_{n-2} and b_{n+2} can not be moved. Hence, only b_{n-1},b_n, and b_{n+1} are to be relocated. Equations 1-3 further require that all but b_{n-3} and b_{n+3} must be coplanar for the possiblity of C2 at b to exist.

The geometric interpretation of the problem is shown in Fig. 3. Equations 4 and 5 require that b_{n-1} and b_{n+1} can be moved only along lines parallel to the vectors represented by the second differences of the end control points. Thus k_1 controls the location of b_{n-1} and k_2 controls the location of b_{n+1} . Equations 1 and 3 then locate the auxilliary points d_1 and d_2 . Thus different values of k_1 move b_{n-1} and d_1 through parallel lines in space. Similarly, k_2 moves b_{n+1} and d_2 through two other parallel lines in space. It should now be obvious that if all control points are coplanar, then the lines swept by d_1 and d_2 are also coplanar and their intersection represents the solution of the problem. Mathematically, we note that there are 5 equations for the 4 unknowns (the three control points and the auxilliary point d) and two undetermined parameters k_1 and k_2. This shows that if all of the control points for both curves are coplanar, then each point has two independent components. Hence the two parameters (k_1 and k_2) are sufficient to make up a 5th unknown and the system is well-posed.

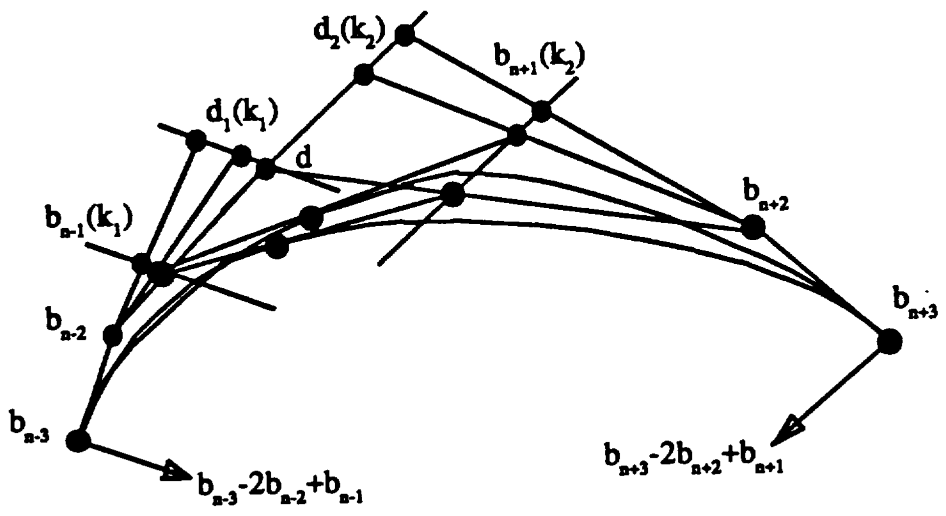

Fig. 3 Geometric interpretation of the solution of equations 1-5. As k_1 and k_2 are changed, the auxilliary points d and d move along lines parallel to the endpoint vectors. The intersection of these lines at d is the solution.

Non-Coplanar Endpoints

If b_{n-3} and b_{n+3} are not coplanar with the other control points then the lines swept by d_1 and d_2 are skew and no solution exists. However, there is an extremely useful interpretation of the closest approach of the two lines. The closest approach of the two skew lines represents the best possible match of the curvatures of the two cubic Bezier space curves. In fact, as the two endpoints approach being coplanar with the remaining control points, the two skew lines come closer to intersecting and the respective curvatures of the two lines approach being exactly C2. This means that if the endpoints are not far from being coplanar with the rest of the control points then it is possible to attain near C2 continuity.

To see why a unique solution may not exist when the endpoints are not coplanar, consider fitting two cubic splines between three points in space. Both C1 and C2 continuity are attained at the middle point but only specified slopes at the ends can be given. To attain specified curvatures at the ends, the middle point can be moved. There is a unique solution to this problem, even if the endpoint slopes are not coplanar. If they are, then this corresponds to the unique solution given in the last section. For the case when they are not, one need only examine the corresponding Bezier control points to see if those nearest the ends match the ones given in the problem. It is unlikely.

We next return to the idea of finding this nearest C2 continuity. It amounts to finding the values of k_1 and k_2 for which the two skew lines have their closest approach. To find these values it is merely sufficient to find the location in space of the points on each line at closest approach. The following is an outline of the procedure and an example to show how near C2 continuity is attained.

Procedure for Nearest C2

Consider the skew lines in space l_1 and l_2 swept out by d_1 and d_2 as the values of k_3 and k_4 are changed as shown in Fig. 4. Note that k_3 and k_4 are directly proportional to k_1 and k_2. The equations for l_1 and l_2 are given by:

$$l_1 = d_1 + k_3 c_1$$
$$l_2 = d_2 + k_4 c_2 \qquad (6)$$

Where c_1 and c_2 are the curvature vectors of the end points. Let the intersections of the line containing the shortest distance between the skew lines with the lines be called P and Q. We now seek to find P and Q and the associated values of k_3 and k_4.

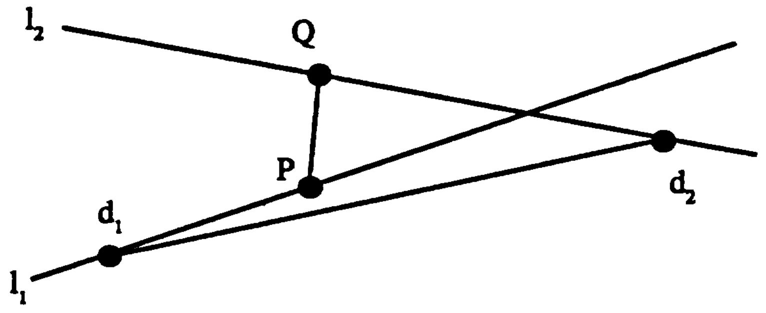

Fig. 4 Skew lines given by sweep of d_1 and d_2 due to changing k_3 and k_4.

(7)

1. The cross product of the endpoint curvature vectors c_1 and c_2 defines the direction of the line containing P and Q. The distance between P and Q is found by projecting the line segment $d_1 d_2$ on the unit vector in the direction of PQ:

$$|PQ| = (d_1 d_2) \cdot \left(\frac{c_1 \times c_2}{|c_1 \times c_2|} \right) \qquad (8)$$

2. Next l_1 is projected onto the plane perpendicular to PQ containing l_2. The equation for the projected line is:

$$l_1 = d_1 + k_3 c_1 + PQ \qquad (9)$$

The projected line crosses l_2 at Q. Hence equating the projected line with l_2 gives an expression for solving for k_3 and k_4.

3. Having found k_3 and k_4, b_{n-1} is found from equation 4, b_{n+1} is found from equation 5, and b_n is found from equation 2. Note that the original C1 and C2 at the ends and C1 at the intersection are maintained exactly while C2 at the intersection is approximate.

Example (Coplanar control points)

To insure that a coplanar set of points was generated while trying to be general, a set of points was generated all with the same z-coordinate. The set was rotated about the x-axis and then about the y-axis. The resulting set of coordinates were used:

	n-3	n-2	n-1	n	n+1	n+2	n+3
x	-1.39545	-.71244	.13878	.88660	2.00833	3.09808	3.08109
y	.76603	1.63205	2.48076	2.67128	2.95707	1.63205	.07321
z	1.58301	1.76603	1.84038	1.53564	1.07853	-.43397	-1.46340

The mismatch in initial curvature at the intersection is shown best by noting that equations 1-3 can also be expressed as requiring:

$$b_n - 2b_{n-1} + b_{n-2} = k(b_n - 2b_{n+1} + b_{n+2}) \qquad (10)$$

Hence each component of the second difference vector on one side of the intersection must be the same multiple (fraction) of the components on the other side. This means that the ratios of the components must be the same. That is:

$$\frac{b_n - 2b_{n-1} + b_{n-2}}{b_n - 2b_{n+1} + b_{n+2}} = k \qquad (11)$$

In this example the initial coordinates gives:

	k
x	3.2324
y	.4086
z	.3592

Table I. Second difference ratios of coordinates at the intersection

Thus the ratio of the second differences of the coordinates on either side of the junction are rather different. After finding the new points by the procedure above we find the ratios to be:

	k
x	.4444
y	.4444
z	.4444

Table II. Second difference ratios of the coordinates at the intersection after imposing curvature continuity

Hence for the case of coplanar control points, the C2 is attained exactly.

Example (Non coplanar)

In this example, the first point, b_{n-3}, was displaced by a distance of .8 in the z-direction before the x- and y-axis rotations. This guaranteed that all points were coplanar except the first which had co-ordinates of (-1.04904, .36603, 2.18301)and was 35.6 deg out of the plane containing the other control points. The initial curvatures were the same as the above example, since the end control point is not included in computing the curvatures. The final curvatures, however, are given in Table III below.

	k
x	.3963
y	.3996
z	.3904

Table III. Second Difference ratios for noncopanra control points after curvature continuity is imposed.

Thus, while exact C2 is not attained, it can be seen that the error in matched C2 at the juncture is small even though one of the ends made an angle of 35.6 deg with the plane containing the remaining control points.

Hence, the procedure outlined above allows for the designer to go to a single intersection of a composite cubic Bezier curve and locally impose curvature, along with slope, continuity if the control points of the two curves are coplanar. Since most cross sections do contain coplanar control points, this represents a rather useful capability. For those intersections where the control points on either side are not coplanar then the procedure allows to come as close as possible. A program listing is given in Appendix A which carries out the steps given above.

Task 2- Curvature continuity (C2) of a series of cubic Bezier curves which are at first only C0

As it has been shown, if the initial control points are not coplanar, then exact curvature continuity, is locally unobtainable, though it is possible to generate the coefficients locally of that pair for Bezier curves which are as close as possible to curvature continuous. This may suffice for many applications. But if not, it becomes necessary to attain the continuity for all cubics in the piecewise curve simultaneously.

Fortunately, this can be done relatively easily. The problem is very similar to setting up an interpolant between points in space. The procedure is one where the endpoints of each Bezier curve is kept fixed in space. Thus the outer Bezier control points of each cubic are retained. We will thus show that the problem is solved by replacing the interior control points with those computed from the corresponding cubic splines placed through the end points of each piecewise cubic. The end conditions used are thosewhich keep the original slopes at the ends of the piecewise curve.

Analysis

Consider the series of Bezier control points, control polygons, and corresponding cubic curves shown in Fig. 5 below.

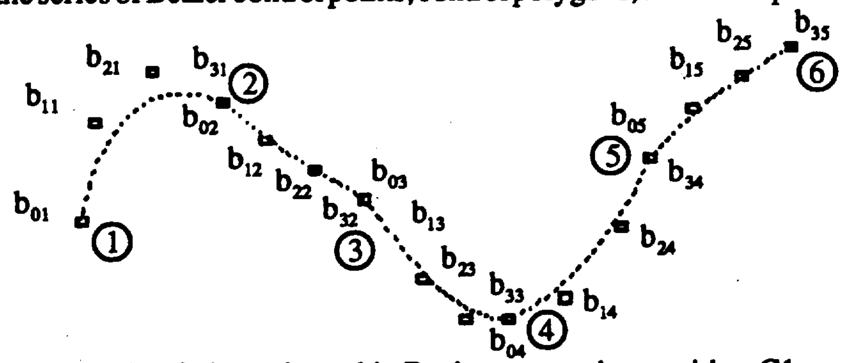

Fig. 5. Example of piecewise cubic Bezier curve where neither C1 nor C2 exist.

For clarity, the control polygon has not been drawn. Since the Bezier control points on either side of the three interior intersections are not collinear with the control point at that intersection, it is clear that slope contiuity is absent. Table IV shows the ratio of curvature vectors for each of the three interior intersections. As they are not equal, it is clear that curvature continuity is also absent.

	Intersection			
	②	③	④	⑤
x	1.0769	.8000	1.5000	-2.6364
y	-6.4000	-.1250	.7273	.1250

Table IV. Curvature ratios at the interior points of initial set of cubic Bezier curves.

Now consider the cubic spline which has been placed through the inersection points as shown in Fig. 6. The intersection points are also the outer two control points of any one cubic. Hence, only new inner control points need be found. To place a cubic spline through these points any standard cubic spline routine can

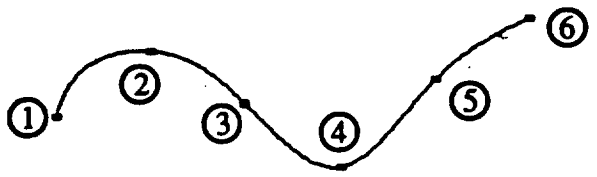

Fig. 6 Cubic spline through the intersections of the Bezier curves.

be used. The user only need specify two end conditions. These end conditions can either be those called "natural" wherein the second derivative is zero; "slope", where the slope at the end is specified; or some combination. It seemed most appropriate in this circumstance to use the slopes existing at the ends of original Bezier cubics. With these end conditions, a piecewise parametric cubic spline was passed through the ends and the intersction points.

For this effort, an optimized parametric cubic spline routine was written where a single parameter, t, varied between zero and one on each cubic. Each cubic has the form: (12)

$$x = \alpha_i t^3 + \beta_i t^2 + \gamma_i t + x_i$$

where x represents the coordinates of a point on the cubic, and x_i contains the coordinates of the ith cubic when t=0. Thus, if there are N cubics, there will be 3N unknowns. To get 3N equations we note that the condition of C0 continuity requires:

$$\alpha_{i-1} + \beta_{i-1} + \gamma_{i-1} + x_{i-1} = x_i \qquad i \in [1,N] \qquad (13)$$

C1 requires:

$$3\alpha_{i-1} + 2\beta_{i-1} + \gamma_{i-1} = \gamma_i \qquad i \in [1,N-1] \qquad (14)$$

and C2 requires:

$$3\alpha_{i-1} + \beta_{i-1} = \beta_i \qquad i \in [1,N-1] \qquad (15)$$

There are 3N equations and this leaves the two end conditions to get the remaining two equations. The usual practice is to use these equations to write a relationship between one of the coefficients at three different points. This eliminates the other two variables. Here we used eq. (13)-(15) to write and eqation for β:

$$\beta_{i-1} + 4\beta_i + \beta_{i+1} = 3(x_{i-1} - 2x_i + x_{i+1}) \qquad i \in [2,N-1] \qquad (16)$$

The linear system is computed by adding the end conditions in terms of β. Hence we have:

$$\frac{2}{3}\beta_1 + \frac{1}{3}\beta_2 = 3(x_2 - x_1) - s_1 \qquad (17)$$

$$\frac{2}{3}\beta_{N-1} + \frac{7}{3}\beta_N = 3(x_{N+1} - x_N) - 2(x_N - x_{N-1}) - s_2 \qquad (18)$$

where

$$s_1 = 3(b_{11} - b_{01})$$
$$s_2 = 3(b_{3N} - b_{2N}) \qquad (19)$$

Equations (16)-(19) form a tridiagonal system of equations for each coordinate direction and can thus be solved very quickly. With each of the β's found, the other coefficients are computed from eq. (14) and (15). This defines the cubics in the piecewise curve.

Once the cubic splines have been generated, the inner two Bezier control points for each cubic can be found from:

$$b_{1i} = b_{0i} + \frac{\gamma_i}{3} \tag{20}$$

$$b_{2i} = b_{3i} - \frac{1}{3}(3\alpha_i + 2\beta_i + \gamma_i)$$

The resulting new curves and corresponding control points are shown in Fig. 7. The dashed lines are the original cubics and the open circles are the original Bezier control points. It may be noted, as a visual check of slope continuity, that at each intersection, the two control points on either side of the intersection

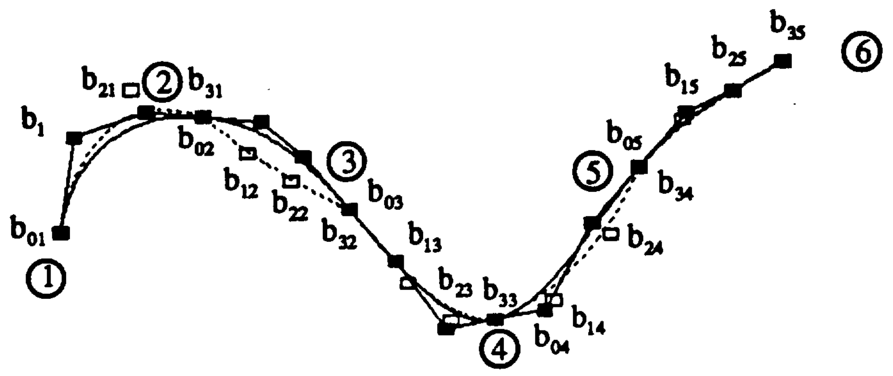

Fig. 7 New cubic Bezier curve with new control points.

point are now collinear, unlike the original control points. The control points next to those at the very ends of the total curve are the same as the original ones as a consequence of retaining the original slope at the end points. The fit of the new curve is close to the old except where the old curve was rather badly matched in slope and in curvature (ie. at intersection ②).

The new curvature ratios, shown below inTable V., show that curvature continuity has been achieved.

	Intersection			
	②	③	④	⑤
x	1.0000	1.0000	1.0000	1.0000
y	1.0000	1.0000	1.0000	1.0000

Table V. Curvature ratios at the 4 interior intersections

This completes the second task. The procedure allows for virtually any combination of cubic Bezier curves which describe a piecewise curve to be made into one which has both slope and curvature continuity. The original control points are retained at each of the curve intersections and new ones are created for the interior of each curve. The new control points are computed from the coefficients of the associated spline curve through the original intersections. The control points adjacent to the outside ends of the first and last cubic have also been retained since the end requirement of slope matching the original curve has been imposed. It may be noted that there is no requirement on the original control points to be coplanar as in task one. A program listing is given in Appendix B for a subroutine which takes the control points of a series of Bezier curves, fits a cubic spline through the intersections and endpoints, and recalculates the interior Bezier control points.

Appendix A
Program Listing for Task 1

The following program in FORTRAN will compute the values of k_1 and k_2 given an initial set of control points, and adjust the position of the control points b_{n-1}, b_n and b_{n+1}.

```
        SUBROUTINE C2(X,Y,Z)

        DIMENSION     X(4, 2),Y(4, 2), Z(4, 2)

C------------------- FIND CURVATURES OF ENDS ----------------------

        CEX1 = X(3,1) - 2*X(2,1) + X(1,1)
        CEY1 = Y(3,1) - 2*Y(2,1) + Y(1,1)
        CEZ1 =  Z(3,1) - 2*Z(2,1) +  Z(1,1)

        CEX2 = X(4,2) - 2*X(3,2) + X(2,2)
        CEY2 = Y(4,2) - 2*Y(3,2) + Y(2,2)
        CEZ2 =  Z(4,2) - 2*Z(3,2) +  Z(2,2)
C------------------- FIND t TO MAINTAIN C2 AT bn ----------------------
        DX = X(4,1) - X(3,1)
        DY = Y(4,1) - Y(3,1)
        DZ = Z(4,1) - Z(3,1)
        S1 = SQRT(DX**2 + DT**2 + DZ**2)
        DX = X(2,2) - X(3,1)
        DY = Y(2,2) - Y(3,1)
        DZ = Z(2,2) - Z(3,1)
        S2 = SQRT(DX**2 + DY**2 + DZ**2)

        T = S1/S2

C---------------------------- FIND d1  and d2 ----------------------------

        XD1 = X(2,1) + (X(3,1) - X(2,1)) / T
        YD1 = Y(2,1) + (Y(3,1) - Y(2,1)) / T
        ZD1 =  Z(2,1) + (Z(3,1) - Z(2,1)) / T

        XD2 = (X(2,2) - T * X(3,2)) / (1 - T)
        YD2 = (Y(2,2) - T * Y(3,2)) / (1 - T)
        ZD2 = (Z(2,2) - T * Z(3,2)) / (1 - T)

C---------------------------- FIND P AND Q ----------------------------

        DX = XD2 - XD1
```

```
      DY = YD2 - YD1
      DZ = ZD2 - ZD1

      CEX = CEY1 * CEZ2 - CEY2 * CEZ1
      CEY = CEX2 * CEZ1 - CEX1 * CEZ2
      CEZ = CEX1 * CEY2 - CEX2 * CEY1

      CL = SQRT ( CEX**2 + CEY**2 + CEZ**2)
      PQ = ( DX * CEX + DY * DEY + DZ * CEZ) / CL

      CEX = CEX * PQ / CL
      CEY = CEY * PQ / CL
      CEZ = CEZ * PQ / CL

      DEN = CEX1 * CEY2 - CEX2 * CEY1
      K3 = ((CEY - DY) * CEX2 - (CEX - DX) * CEY2) / DEN
      K4 = ((CEY - DY) * CEX1 - (CEX - DX) * CEY1) / DEN

      PX = XD1 + K3 * CEX1
      PY = YD1 + K3 * CEY1
      PZ = ZD1 + K3 * CEZ1

      QX = XD2 + K4 * CEX2
      QY = YD2 + K4 * CEY2
      QZ = ZD2 + K4 * CEZ2

C----------------------------- NEW K1 AND K2 ---------------------

      K1 = (T * PX - (1 + T) * X(2,1) + X(1,1)) / CEX1
      K2 = ((1 - T) * QX + (T - 2) * X(3,2) + X(4,2)) / CEX2

C--------------------- FIND NEW b , b  , and b  ----------------------

      X(3,1) = 2 * X(2,1) - X(1,1) + K1 * CEX1
      Y(3,1) = 2 * Y(2,1) - Y(1,1) + K1 * CEY1
      Z(3,1) = 2 * Z(2,1) - Z(1,1) + K1 * CEZ1

      X(2,2) = 2 * X(3,2) - X(4,2) + K2 * CEX2
      Y(2,2) = 2 * Y(3,2) - Y(4,2) + K2 * CEY2
      Z(2,2) = 2 * Z(3,2) - Z(4,2) + K2 * CEZ2

      XN = (1 - T) * X(3,1) + T * X(2,2)
      YN = (1 - T) * Y(3,1) + T * Y(2,2)
      ZN = (1 - T) * Z(3,1) + T * Z(2,2)

      X(4,1) = XN
      Y(4,1) = YN
      Z(4,1) = ZN

      X(1,2) = XN
      Y(1,2) = YN
      Z(1,2) = ZN

C------------------------------------ THAT'S ALL ------------------------------

      RETURN
      END
```

Appendix B.
Curvature Continuity on a Piecewise Cubic Curve

The following FORTRAN subroutine is designed to perform the steps listed in Task 2. That is, given an input set of N Bezier cubic control points, cubic splines are put through the intersection points and new interior control points are found.

```
      SUBROUTINE  BCRV(X,Y,Z,N)

      DIMENSION   X(4,N),Y(4,N),Z(4,N)
      DIMENSION   XX(N+1),YY(N+1),ZZ(N+1)
      DIMENSION   A(N),B(N),C(N),D(N)
      DIMENSION   AA(N),BB(N),CC(N)

C------------------------ SOME OF THE VARIABLES ------------------
C
C     N          Number of Bezier Curves
C     X,Y,Z      Coordinates of Bezier Control points
C     XX,YY,ZZ   Coordinates of the ends and intersections of the piecewise curve
C     A,B,C,D    Coefficients of the cubic spline matrix
C     AA,BB,CC   Coefficients of the cubic splines
C
C------------------------- FIND SLOPES AT THE ENDS ----------------

      S1X = 3. * (X(2,1) - X(1,1))
      S1Y = 3. * (Y(2,1) - Y(1,1))
       S1Z = 3. * (Z(2,1) - Z(1,1))

      S2X = 3. * (X(4,N) - X(3,N))
      S2Y = 3. * (Y(4,N) - Y(3,N))
      S2Z = 3. * (Z(4,N) - Z(3,N))

C------------ GET THE INTERSECTION AND END POINTS ---------

      DO 100 I = 1,N

       XX(I) = X(1,I)
       YY(I) = Y(1,I)
       ZZ(I) = Z(1,I)

  100 CONTINUE

      XX(N+1) = X(4,N)
      YY(N+1) = Y(4,N)
      ZZ(N+1) = Z(4,N)
```

```
C-------------------------- SET UP CUBIC SPLINE MATRIX ---------------

C----------- LOOP ON DIMENSIONS

      DO 1000 K = 1,3

       DO 200 I = 2,N-1

        C(I) = 1
        B(I) = 4
        A(I) = 1

        IF (K.EQ.1) THEN
          DD = XX(I+1) - 2. * XX(I) + XX(I-1)
        ELSEIF (K.EQ.2) THEN
          DD = YY(I+1) - 2. * YY(I) + YY(I-1)
        ELSE
          DD = ZZ(I+1) - 2. * ZZ(I) + ZZ(I-1)
        ENDIF

        D(I) = 3. * DD

 200    CONTINUE

        B(1) = 2. / 3.
        A(1) = 1 ./ 3.
        IF ( K.EQ.1) THEN
         D(1) = 3. * (XX(2) - XX(1)) - S1X
        ELSEIF (K.EQ.2) THEN
         D(1) = 3. * (YY(2) - YY(1)) - S1Y
        ELSE
         D(1) = 3. * (ZZ(2) - ZZ(1)) - S1Z
        ENDIF

        B(N) =  7. / 3.
        C(N) = 2. / 3.
        IF (K.EQ.1) THEN
         D(N) = 3. * (XX(N+1) - XX(N)) - 2. * (XX(N) - XX(N-1)) - S2X
        ELSEIF (K.EQ.2) THEN
         D(N) = 3. * (YY(N+1) - YY(N)) - 2. * (YY(N) - YY(N-1)) - S2Y
        ELSE
         D(N) = 3. * (ZZ(N+1) - ZZ(N)) - 2. * (ZZ(N) - ZZ(N-1)) - S2Z
        ENDIF

C---------------------- SOLVE THE MATRIX --------------------

        CALL TSOLV(A,B,C,D,BB)
```

```
C---------- FIND THE OTHER CUBIC SPLINE COEFFICIENTS --------

      IF (K.EQ.1) THEN

       DO 300 I = 1,N-1
          AA(I) = (BB(I+1) - B(I)) / 3.
          CC(I) = XX(I+1) - XX(I) - AA(I) - BB(I)
 300   CONTINUE
       CC(N) = 3. * AA(N-1) + 2. * BB(N-1) + CC(N-1)
       AA(N) = XX(N+1) - XX(N) - BB(N) - CC(N)

      ELSEIF (K.EQ.2) THEN

       DO 320 I = 1,N-1
          AA(I) = (BB(I+1) - BB(I)) / 3.
          CC(I) = YY(I+1) - YY(I) - AA(I) - BB(I)
 320   CONTINUE
       CC(N) = 3.*AA(N-1) + 2.*BB(N-1) + CC(N-1)
       AA(N) = YY(N+1) - YY(N) - BB(N) - CC(N)

      ELSE

       DO 340 I = 1,N-1
          AA(I) = ( BB(I+1) - BB(I)) / 3.
          CC(I) = ZZ(I+1) - ZZ(I) - AA(I) - BB(I)
 340   CONTINUE
       CC(N) = 3.*AA(N-1) + 2.*BB(N-1) + CC(N-1)
       AA(N) = ZZ(N+1) - ZZ(N) - BB(N) - CC(N)

      ENDIF

C------------------ NOW GET NEW BEZIER COEFFICIENTS ----------

      DO 500 I = 1 TO N

         IF (K.EQ.1) THEN

            DX0 = CC(I)
            DX1 = 3. * AA(I) + 2. * BB(I) + CC(I)
            X(2,I) = X(1,I) + DX0 / 3.
            X(3,I) = X(4,I) - DX1 / 3.

         ELSEIF (K.EQ.2) THEN

            DY0 = CC(I)
            DY1 = 3. * AA(I) + 2. * BB(I) + CC(I)
          Y(2,I) = Y(1,I) + DY0 / 3.
          Y(3,I) = Y(4,I) - DY1 / 3.

        ELSE

          DZ0 = CC(I)
          DZ1 = 3. * AA(I) + 2. * BB(I) + CC(I)
          Z(2,I) = Z(1,I) + DZ0 / 3.
          Z(3,I) = ZX(4,I) - DZ1 / 3.

        ENDIF

  500   CONTINUE

C-------------------- END OF DIMENSION LOOP --------------

 1000  CONTINUE

C------------------- THAT'S ALL ----------------

       STOP
       END
C
       SUBROUTINE TSOLV(A,B,C,D,N,BB)

       DIMENSION    A(N),B(N),C(N),D(N),BB(N)

C-------------------- ELIMINATE C'S ----------------------

       DO 100 I = 2,N

          CBI = C(I) / B(I-1)
          B(I) = B(I) - CBI * A(I-1)
          D(I) = D(I) - CBI * D(I-1)

   100  CONTINUE

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

        DO 200 I = N-1,1,-1

           BB(I) = (D(I) - A(I)*BB(I+1)) / B(I)

   200  CONTINUE

        RETURN
        END
```