Corners in Planar Cubic B-spline and Bézier Curve Segments

D.J. Walton
D.S. Meek
Department of Computer Science
University of Manitoba
Winnipeg, Canada R3T 2N2

Abstract

A cubic B-spline or Bézier curve segment is determined by four control vertices. Assuming that three of the vertices have already been positioned, the locus of the fourth vertex which would cause the curve segment to have a corner is determined. The corner curve lies in a region which can easily be determined and rendered efficiently in an interactive graphics environment.

Keywords: Splines, Bézier Curves, Geometric Modelling.

Introduction

Cubic Bézier curves as well as cubic B-spline curves and their variations such as Beta-spline curves are frequently used in computer-aided design and computer-aided geometric design applications. Although geometric continuity can be assured at the joints of such piecewise cubic parametric curves prior to their rendering, it is possible for the curvature of individual spline curve segments to become undefined [Wang '81]. The curve can have a corner or cusp at such a point.

In the design of highway and railway routes, a curve with pre-specified maximum curvature is required. A recent algorithm [Walton and Meek '87] finds an accurate bound on the curvature of any cubic B-spline curve segment, but the algorithm only works on segments which have defined and finite curvature. Thus, it would be useful for a curve designer to know *a priori* which positions to avoid when choosing control vertices. Preliminary highway or railway routes are typically designed from an initial starting point by placing successive guiding points in order along an approximate proposed horizontal alignment. Ideally, for such applications in which it is of primary importance to ensure that the curvature of the designed curve is below a specified maximum, and in which the shape of the curve is of secondary importance, it would be desirable for the user to input the specified maximum curvature, so that the region in which the user may safely position successive points, could be displayed on an interactive graphics screen. This problem is still under investigation. The preliminary results presented in this paper enable designers of such applications to avoid placing successive guiding points in regions in which the curvature may become undefined.

It is shown in [Lau '88] that the user may avoid cusps by preventing the Bézier control polygon from turning through an angle of π radians. It is shown in this paper that if three control vertices of a guided cubic B-spline or Bézier curve segment are already positioned, then a simple curve, called the corner curve, can be determined *a priori* such that if the fourth control vertex is positioned off that curve, the curvature is defined and finite, and if positioned on that curve, the curvature of the B-spline or Bézier curve will become undefined at an interior point. This is a strong *a priori* condition which will be useful for computer-aided design or computer-aided geometric design applications since if the corner curve is drawn, the designer will know how to position the next vertex without loss of geometric continuity between knots.

A frame of reference

Properties of B-spline and Bézier curves may be found in [Böhm, Farin and Kahmann '84]. Let the four control vertices $\mathbf{V}_0, \mathbf{V}_1, \mathbf{V}_2$ and $\mathbf{V}_3$ define a cubic B-spline curve segment. Although results for B-splines are perhaps of more interest, it is easier to work with Bézier control vertices and convert the results back to B-spline control vertices afterwards. The corresponding Bézier control vertices for the same curve segment, $\mathbf{Q}(u)$ parametrized over $u \in [0,1]$, are $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ and $\mathbf{P}_3$ [Barsky and De Rose '85] where

$$\mathbf{Q}(u) = \mathbf{P}_0(1-u)^3 + 3\mathbf{P}_1(1-u)^2u + 3\mathbf{P}_2(1-u)u^2 + \mathbf{P}_3u^3, \quad (1)$$

$$\begin{aligned} \mathbf{P}_0 &= (\mathbf{V}_0 + 4\mathbf{V}_1 + \mathbf{V}_2)/6, \\ \mathbf{P}_1 &= (2\mathbf{V}_1 + \mathbf{V}_2)/3, \\ \mathbf{P}_2 &= (\mathbf{V}_1 + 2\mathbf{V}_2)/3, \quad (2) \\ \mathbf{P}_3 &= (\mathbf{V}_1 + 4\mathbf{V}_2 + \mathbf{V}_3)/6. \quad (3) \end{aligned}$$

In the sequel it is assumed that $\mathbf{V}_0 \neq \mathbf{V}_2$, $\mathbf{V}_1 \neq \mathbf{V}_2$, $\mathbf{V}_1 \neq \mathbf{V}_3$ and $\mathbf{V}_0, \mathbf{V}_1, \mathbf{V}_2$ not collinear. These conditions are equivalent to $\mathbf{P}_0 \neq \mathbf{P}_1$, $\mathbf{P}_1 \neq \mathbf{P}_2$, $\mathbf{P}_2 \neq \mathbf{P}_3$ and $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ not collinear. It follows from Equation (1) that

$$\mathbf{Q}'(u) = 3(\mathbf{P}_1-\mathbf{P}_0)(1-u)^2+6(\mathbf{P}_2-\mathbf{P}_1)(1-u)u+3(\mathbf{P}_3-\mathbf{P}_2)u^2. \quad (4)$$

It is convenient to choose an orthogonal frame, which remains fixed as $\mathbf{P}_3$ varies, as

$$\mathbf{H}_1 = 3(\mathbf{P}_1 - \mathbf{P}_0), \quad (5)$$

$$\mathbf{H}_2 = 6(\mathbf{P}_2 - \mathbf{P}_1) - c_0\mathbf{H}_1, \quad (6)$$

where

$$c_0 = 6[(\mathbf{P}_2 - \mathbf{P}_1)\cdot\mathbf{H}_1]/(\mathbf{H}_1\cdot\mathbf{H}_1). \quad (7)$$

In this particular orthogonal frame, $\mathbf{H}_1$ is parallel to the unit tangent vector of the Bézier curve at its beginning point and $\mathbf{H}_2$ is parallel to the principal normal as defined by the Frenet frame

[Faux and Pratt '79]. The vector $3(\mathbf{P}_3 - \mathbf{P}_2)$ may be written in this frame as

$$3(\mathbf{P}_3 - \mathbf{P}_2) = c_1\mathbf{H}_1 + c_2\mathbf{H}_2 \qquad (8)$$

where

$$c_1 = [3(\mathbf{P}_3 - \mathbf{P}_2)\bullet\mathbf{H}_1]/(\mathbf{H}_1\bullet\mathbf{H}_1), \qquad (9)$$

$$c_2 = [3(\mathbf{P}_3 - \mathbf{P}_2)\bullet\mathbf{H}_2]/(\mathbf{H}_2\bullet\mathbf{H}_2). \qquad (10)$$

It also follows from Equation (6) that $6(\mathbf{P}_2 - \mathbf{P}_1)$ may be written as

$$6(\mathbf{P}_2 - \mathbf{P}_1) = c_0\mathbf{H}_1 + \mathbf{H}_2. \qquad (11)$$

Using Equations (5)-(11), Equation (4) may thus be re-written as

$$\mathbf{Q}'(u) = g_1(u)\mathbf{H}_1 + g_2(u)\mathbf{H}_2 \qquad (12)$$

where

$$g_1(u) = (c_1 - c_0 + 1)u^2 + (c_0 - 2)u + 1 \qquad (13)$$

and

$$g_2(u) = (c_2 - 1)u^2 + u. \qquad (14)$$

The corner curve

Any point $\mathbf{Q}(u)$ for which $\mathbf{Q}'(u) = \mathbf{0}$ is called a singular point [Do Carmo '76], and the curvature is undefined at such a point. Conversely, at any point $\mathbf{Q}(u)$ for which $\mathbf{Q}'(u) \neq \mathbf{0}$ there is a defined and finite curvature. In this paper, the locus of the points $\mathbf{P}_3$ or $\mathbf{V}_3$ which give a cubic Bézier or B-spline curve with a singular point is called the corner curve. Under the conditions described in the previous section, it can be shown that a cusp is produced in the resulting Bézier or B-spline curve if and only if $\mathbf{P}_3$ or $\mathbf{V}_3$ is on the corner curve.

Suppose $\mathbf{P}_3$ is chosen so that there is an interior singular point $\mathbf{Q}(\lambda)$ for some $\lambda \in (0,1)$, then $g_1(\lambda)$ and $g_2(\lambda)$ must be zero. From Equation (14) it follows that $\lambda = 1 / (1 - c_2)$ where $c_2 < 0$. The mean value theorem gives

$$g_1(u) = g_1(\lambda) + (u - \lambda)g_1'(m_1)$$

and

$$g_2(u) = g_2(\lambda) + (u - \lambda)g_2'(m_2).$$

Hence, for $\mathbf{Q}(\lambda)$ a singular point,

$$g_1(u) = (u - \lambda)g_1'(m_1)$$

and

$$g_2(u) = (u - \lambda)g_2'(m_2)$$

where m_1 and m_2 are between u and λ. Now, $g_2'(\lambda) = -1 \neq 0$, so it is clear that $\mathbf{Q}''(\lambda)$ is a non-zero vector. Although the unit tangent is not defined at $u = \lambda$, the factor $u - \lambda$ in the above equations causes the tangent vector to $\mathbf{Q}(u)$ to become parallel to $\mathbf{Q}''(\lambda)$ and reverse direction as u passes through λ. Thus, with restrictions on the control vertices, as stated in the previous section, a cusp must occur at a singular point. It is interesting to note that if $c_2 > 0$, or $(\mathbf{P}_3 - \mathbf{P}_2)\bullet\mathbf{H}_2 > 0$, there is no corner or cusp, and the curvature is defined and finite. This is the result obtained by Lau ['88].

A stronger condition for the occurrence of corners can be obtained by requiring that $g_1(\lambda) = 0$, hence

$$(c_1 - c_0 + 1)/(1 - c_2)^2 + (c_0 - 2)/(1 - c_2) + 1 = 0,$$

which leads to

$$c_1 = c_0c_2 - c_2^2. \qquad (15)$$

If $\mathbf{P}_3$ is a point which causes a cusp, then it must make $c_2 < 0$ and $c_1 = c_0c_2 - c_2^2$. From Equations (8) and (15) it follows that if $\mathbf{P}_3$ is on the corner curve,

$$\mathbf{P}_3 = \mathbf{P}_2 + (c_1\mathbf{H}_1 + c_2\mathbf{H}_2)/3$$

$$= \mathbf{P}_2 + \{(c_0c_2 - c_2^2)\mathbf{H}_1 + c_2\mathbf{H}_2\}/3.$$

Since only c_2 depends on $\mathbf{P}_3$ in the above right hand side, and $\mathbf{P}_3$ can be chosen to give c_2 any negative value, a parametric representation of the corner curve for the first three given cubic Bézier control vertices is

$$\mathbf{C}_1(t) = \mathbf{P}_2 + \{(c_0t - t^2)\mathbf{H}_1 + t\mathbf{H}_2\}/3, \quad t < 0. \qquad (16)$$

If $\mathbf{V}_3$ is a point which causes a cusp, then it must make $c_2 < 0$ and $c_1 = c_0c_2 - c_2^2$. From Equations (2), (3), (7), and (15) it follows that if $\mathbf{V}_3$ is on the corner curve,

$$\mathbf{V}_3 = \mathbf{V}_1 + 6(\mathbf{P}_3 - \mathbf{P}_2)$$

or

$$\mathbf{V}_3 = \mathbf{V}_1 + 2(c_1\mathbf{H}_1 + c_2\mathbf{H}_2)$$

$$= \mathbf{V}_1 + 2(c_0c_2 - c_2^2)\mathbf{H}_1 + 2c_2\mathbf{H}_2.$$

Since only c_2 depends on $\mathbf{V}_3$ in the above right hand side, and $\mathbf{V}_3$ can be chosen to give c_2 any negative value, a parametric representation of the corner curve for the first three given cubic B-spline control vertices is

$$\mathbf{C}_2(t) = \mathbf{V}_1 + 2(c_0t - t^2)\mathbf{H}_1 + 2t\mathbf{H}_2, \quad t < 0. \qquad (17)$$

The corner curve is traced by computing and plotting the points of $\mathbf{C}_1(t)$ or $\mathbf{C}_2(t)$ for negative values of t. Since the corner curve extends to infinity, it is desirable to clip it at the boundary of the drawing region, as described in the Appendix.

Examples

An algorithm to predict the locus of the fourth B-spline or Bézier vertex which would cause corners was implemented in Pascal on a Macintosh computer for illustrative purposes. The results are illustrated in Figures 1 - 6.

Figure 1 shows the corner curve as a thick line for three given B-spline control vertices. The control polygon is shown as a dotted line. An attempt was then made to create a corner by positioning the fourth vertex on the corner curve. The resulting curve is shown in Figure 2.

Figures 3 and 4 demonstrate similar results for a cubic Bézier curve. Figures 5 and 6 demonstrate the sectors of the Drawing Window which contain the corner curve for a cubic B-spline curve segment and a cubic Bézier curve respectively. The clipping segment, $\mathbf{S}(\sigma)$, as described in the Appendix, was respectively chosen to coincide with each of the boundaries of the Drawing Window.

Conclusion

A simple but effective method has been demonstrated for locating the position of the last point of a segment which would cause a corner in a B-spline or Bézier curve. A designer may request a display of the corner curve if it is desired to design a curve with corners, or the designer may request a display of the sector of the drawing window which should be avoided if it is desirable not to have any corners. The results presented in this

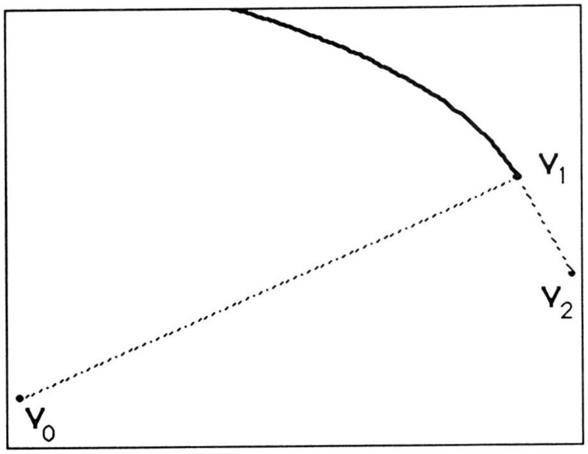

Figure 1. Corner curve for a B-spline curve segment.

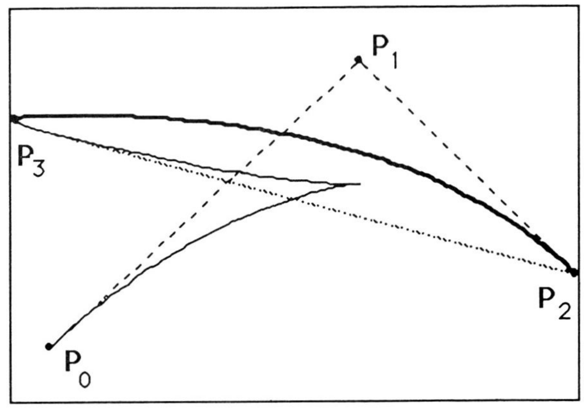

Figure 4. Bézier curve segment with a corner.

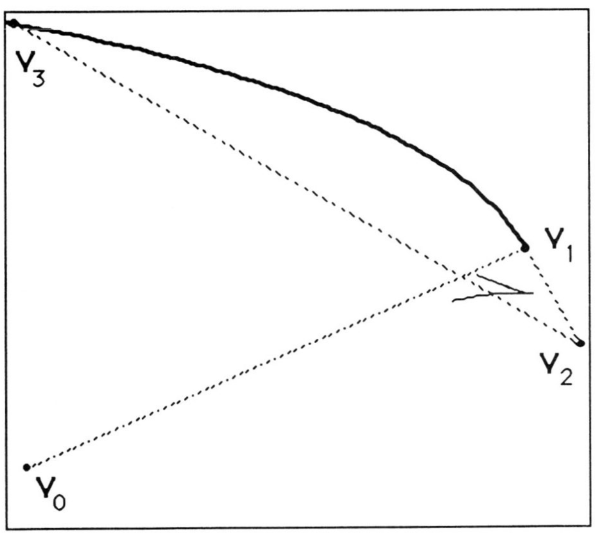

Figure 2. B-spline curve segment with a corner.

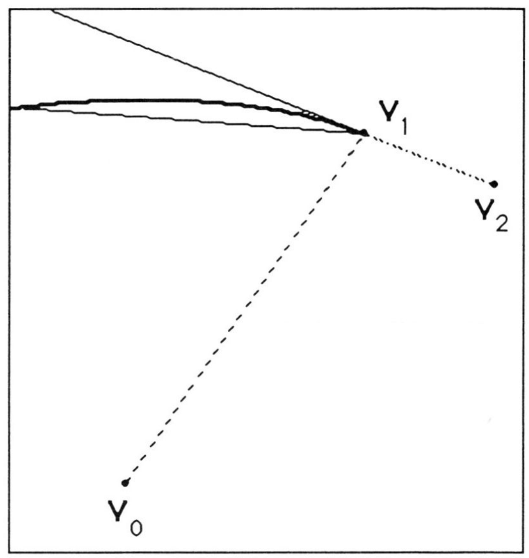

Figure 5. Corner sector for a B-spline curve segment.

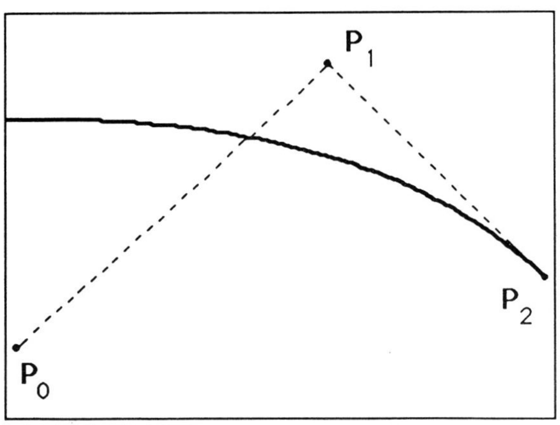

Figure 3. Corner curve for a Bézier curve segment.

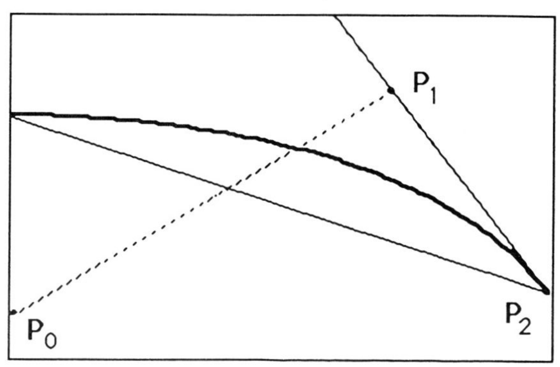

Figure 6. Corner sector for a Bézier curve segment.

article are easily extended to cubic Beta-splines [Barsky '88], Nu-splines [Nielson '85] and rational geometric cubic splines [Böhm '87] since their control polygons can be determined in terms of guided cubic B-spline or Bézier vertices.

The corner curve, in conjunction with the results of Wang ['81], can be used to anticipate the occurrence of loops, S-bends or double S-bends. It is for example well known that if a cubic Bézier control intersects itself, then a loop usually occurs in the corresponding cubic curve segment; however, if the polygon intersects itself such that the fourth vertex lies between the corner curve and the straight line which passes through the second and third vertices, then a curve with a double S-bend occurs as illustrated in Figure 7. Unfortunately, simple conditions, such as that which has been determined for the corner curve, which determine the regions in which loops, S-bends or double S-bends may occur, seem to be elusive; however, work is in progress to search for simple conditions for these regions.

The analysis in this paper was done for the last control vertex of a B-spline or Bézier curve segment. The same analysis can be applied to the first control vertex by simply re-labelling the vertices in reverse order. Analyses for the interior vertices are underway but are more complicated since a change to an interior vertex of the B-spline segment would cause all the vertices of the corresponding Bézier segment to change.

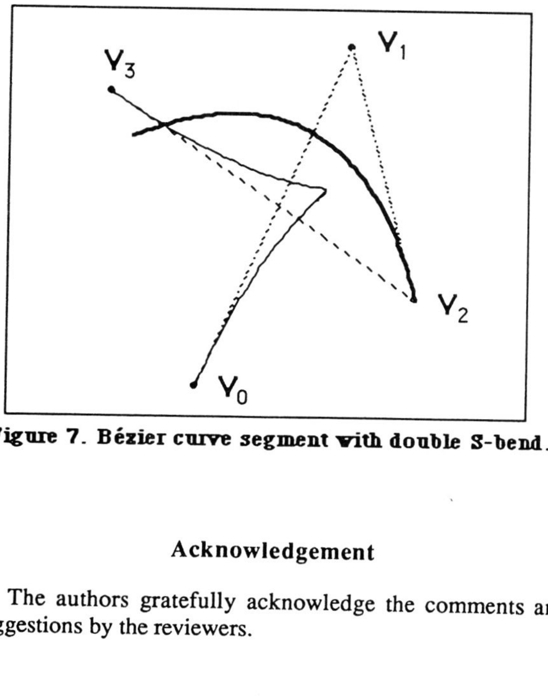

Figure 7. Bézier curve segment with double S-bend.

Acknowledgement

The authors gratefully acknowledge the comments and suggestions by the reviewers.

References

Barsky, B.A. (1988), *Computer Graphics and Geometric Modeling Using Beta-splines,* Springer-Verlag, New York.

Barsky, B.A. and De Rose, T.D. (1985), The Beta2-spline: A special case of the Beta-spline curve and surface representation, *IEEE Computer Graphics and Applications* **5**, No. 5, 46-58.

Böhm, W. (1987), Rational geometric splines, *Computer Aided Geometric Design* **4**, 67-77.

Böhm, W., Farin, G. and Kahmann, J. (1984), A survey of curve and surface methods in CAGD, *Computer Aided Geometric Design* **1**, 1-60.

Do Carmo, M.P. (1976), *Differential Geometry of Curves and Surfaces,* Prentice-Hall, Englewood Cliffs, New Jersey.

Faux, I.D. and Pratt, M.J. (1985), *Computational Geometry for Design and Manufacture,* Ellis Horwood, Chichester, U.K.

Lau, K.H. (1988), Conditions for avoiding loss of geometric continuity on spline curves, *Computer Aided Geometric Design* **5**, 209-214.

Nielson, G.M. (1986), Rectangular ν-splines, *IEEE Computer Graphics and Applications* **6**, No. 2, 35-40.

Walton, D.J. and Meek, D.S. (1987), Curvature bounds for planar B-spline curve segments, *Computer Aided Design* **20**, 146-150.

Wang, C.Y. (1981), Shape classification of the parametric cubic curve and parametric B-spline curve, *Computer Aided Design* **13**, 199-206.

Appendix

Clipping The Corner Curve

Designed curves are usually restricted to regions, the boundaries of which are polygonal. The most common case is the restriction of a drawing to a particular rectangular region, usually called the Drawing Window. It would thus be necessary to clip the drawing. The corner curve is also easily clipped as follows. Consider the straight line segment

$$S(\sigma) = (1-\sigma)\mathbf{S}_0 + \sigma\mathbf{S}_1, \quad 0 \le \sigma \le 1, \tag{18}$$

joining the two screen points, $\mathbf{S}_0$ and $\mathbf{S}_1$. This line will clip the corner curve for given Bézier control vertices whenever $S(\sigma) = \mathbf{C}_1(t)$. From Equation (18),

$$\begin{aligned} S(\sigma) - \mathbf{P}_2 &= (1-\sigma)\mathbf{S}_0 + \sigma\mathbf{S}_1 - \mathbf{P}_2 \\ &= (1-\sigma)(\mathbf{S}_0 - \mathbf{P}_2) + \sigma(\mathbf{S}_1 - \mathbf{P}_2). \end{aligned}$$

Now

$$\begin{aligned} \mathbf{S}_0 - \mathbf{P}_2 &= \alpha_1\mathbf{H}_1 + \alpha_2\mathbf{H}_2, \\ \alpha_1 &= (\mathbf{S}_0 - \mathbf{P}_2)\cdot\mathbf{H}_1/(\mathbf{H}_1\cdot\mathbf{H}_1), \\ \alpha_2 &= (\mathbf{S}_0 - \mathbf{P}_2)\cdot\mathbf{H}_2/(\mathbf{H}_2\cdot\mathbf{H}_2), \end{aligned}$$

and

$$\begin{aligned} \mathbf{S}_1 - \mathbf{P}_2 &= \beta_1\mathbf{H}_1 + \beta_2\mathbf{H}_2, \\ \beta_1 &= (\mathbf{S}_1 - \mathbf{P}_2)\cdot\mathbf{H}_1/(\mathbf{H}_1\cdot\mathbf{H}_1), \\ \beta_2 &= (\mathbf{S}_1 - \mathbf{P}_2)\cdot\mathbf{H}_2/(\mathbf{H}_2\cdot\mathbf{H}_2). \end{aligned}$$

Hence, using Equation (16) the condition $S(\sigma) = \mathbf{C}_1(t)$ becomes

$$\begin{aligned} S(\sigma) - \mathbf{P}_2 &= \{(1-\sigma)\alpha_1 + \sigma\beta_1\}\mathbf{H}_1 + \{(1-\sigma)\alpha_1 + \sigma\beta_1\}\mathbf{H}_2. \\ &= \{(c_0t - t^2)\mathbf{H}_1 + t\mathbf{H}_2\}/3 \end{aligned}$$

The two equations,

$$(1-\sigma)\alpha_1 + \sigma\beta_1 = (c_0t - t^2)/3, \tag{19}$$

$$(1-\sigma)\alpha_2 + \sigma\beta_2 = t/3, \tag{20}$$

are thus obtained. Elimination of σ from the Equations (19) and (20) produces

$$a_2t^2 + (a_1 - c_0a_2)t + 3(\alpha_1a_2 - \alpha_2a_1) = 0, \qquad (21)$$

where

$$a_i = \alpha_i - \beta_i, \qquad i = 1,2.$$

Equation (21) may have 0, 1 or 2 solutions. Valid clipping points are only obtained for those solutions which produce a value of σ in the range [0,1] when substituted into Equation (20).

The same analysis can be used to obtain the clipping points for given B-spline control vertices by using Equation (17) instead of (16).

Evaluation of the first derivatives of Equations (16) and (17) at t=0 and comparison of the results with Equation (11) show that the vector $\mathbf{P}_2 - \mathbf{P}_1$ is tangent to the corner curve, which is actually part of a parabola. Since only negative values of c_2 are used in generating the corner curve, it will start from $\mathbf{P}_2$ or $\mathbf{V}_1$ and proceed in a direction opposite to $\mathbf{H}_2$. The axis of symmetry of the corner parabola is parallel to $\mathbf{H}_1$. If the corner curve intersects the drawing region more than once, the sector to be avoided is the smallest contiguous sector which includes all the straight lines which join $\mathbf{P}_2$ or $\mathbf{V}_1$ to each of the clipping points as well as that part of the ray emanating from $\mathbf{P}_2$ or $\mathbf{V}_1$ in the direction of $\mathbf{P}_1 - \mathbf{P}_2$ which lies inside the drawing region.

This is a very useful result for curve designers who wish to avoid corners or near corners since the region to be avoided is easily shown by simply drawing two straight lines on the screen.