ELSEVIER

Computer-Aided Design 34 (2002) 807–814

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

Biarc approximation of NURBS curves

Les A. Piegl[a,*], Wayne Tiller[b]

[a]Department of Computer Science and Engineering, University of South Florida, 4202 Fowler Avenue, ENG 118, Tampa, FL 33620, USA
[b]GeomWare, Inc., 3035 Great Oak Circle, Tyler, TX 75703, USA

Received 29 May 2001; revised 23 July 2001; accepted 26 July 2001

Abstract

An algorithm for approximating arbitrary NURBS curves with biarcs is presented. The main idea is to approximate the NURBS curve with a polygon, and then to approximate the polygon with biarcs to within the required tolerance. The method uses a parametric formulation of biarcs appropriate in geometric design using parametric curves. The method is most useful in numerical control to drive the cutter along straight line or circular paths.

Keywords: Curve approximation; Biarcs; Curves and surfaces; Algorithms

1. Introduction

Approximating data, points, lines or arbitrary curves, with curves of various forms is an essential operation in engineering design. Approximation by straight lines and circular arcs is of particular importance because of the simplicity of these curves, and because of the capability of the milling machines to move along straight line and circular paths [12,19]. Approximations to data by C^0 [16], and by G^1 arcs [7,9,15,18,21,22,25,26] have been investigated in the past. Approximations to given curves [1,4,8,10,11,14,15,22,24,27] have also been the subject of extensive research. Because of the restrictions inherent in circle approximation, biarc approximation is by far not as well studied as the approximation of data by splines.

This paper presents a method of approximating a NURBS curve by a set of biarc curves to within a user specified tolerance. The main ideas of the method are summarized as follows:

- approximate the NURBS curve by a polygon to within a certain tolerance;
- approximate the polygon with biarcs whose end points lie on the curves, and whose end tangents are the tangents of the curve taken at the end points of the biarc; and
- hook up the biarcs into a G^1 curve that lies from the given curve within tolerance.

The organization of the paper is as follows. In Section 2 a biarc formulation is given, followed by Section 3 where details on the approximation method are presented. Some tests and examples are found in Section 4 before the paper is closed with some conclusions.

2. Biarc formulation

Biarcs appeared in the engineering design literature as early as the 1970s [2,3,21], and research has continued, although quite modestly, until the present day [9,13,15,20,22,23,26]. The idea of using two arcs instead of one is due to the fact that a circular arc cannot match two end-point and two end-tangent conditions. Choosing two curves, i.e. a piecewise curve, these conditions are nicely met. More precisely, given two points $\mathbf{P}_s$ and $\mathbf{P}_e$, and two unit end tangents $\mathbf{T}_s$ and $\mathbf{T}_e$, $|\mathbf{T}_s| = |\mathbf{T}_e| = 1$, a piecewise circular arc is sought so that:

- it passes through $\mathbf{P}_s$ and $\mathbf{P}_e$;
- it is tangential at $\mathbf{P}_s$ to $\mathbf{T}_s$, and at $\mathbf{P}_e$ to $\mathbf{T}_e$; and
- the arcs join in G^1 continuity.

In the general case two arcs are satisfactory, however, some special cases require four arcs.

The great majority of biarc formulations are coordinate system dependent, calculating the respective radii based on angles and trigonometry. While this is a correct approach, it is not appropriate in the world of parametric curves. Below we give a derivation that is coordinate systems independent and relies on the well established NURBS formulation.

* Corresponding author. Tel.: +1-813-974-5234; fax: +1-813-974-5456.
E-mail addresses: piegl@aol.com (L.A. Piegl), geomware@gower.net (W. Tiller).

0010-4485/02/$ - see front matter
PII: S0010-4485(01)00160-9

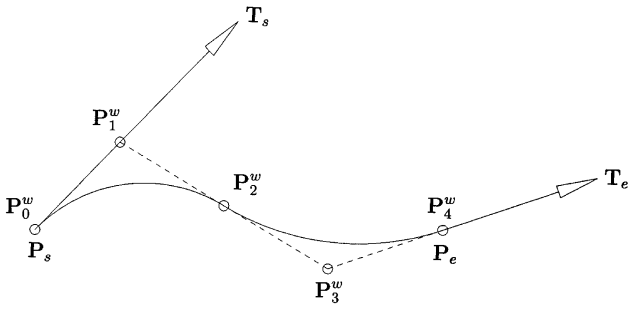


Fig. 1. Biarc formulation.

Referring to Fig. 1, the unknown control points $\mathbf{P}_1^w$, $\mathbf{P}_2^w$ and $\mathbf{P}_3^w$ are sought. Below, the Euclidean projections $\mathbf{P}_1$, $\mathbf{P}_2$ and $\mathbf{P}_3$ are computed first, then appropriate weights are assigned to them. Since the end tangents are assumed to be of unit lengths, we get

$$\mathbf{P}_1 = \mathbf{P}_0 + \alpha\mathbf{T}_s \qquad \mathbf{P}_2 = \frac{\beta}{\alpha+\beta}\mathbf{P}_1 + \frac{\alpha}{\alpha+\beta}\mathbf{P}_3$$
$$\mathbf{P}_3 = \mathbf{P}_4 - \beta\mathbf{T}_e \tag{1}$$

Because the control triangle of a circle must be an isosceles triangle [17], the following must hold

$$|\mathbf{P}_1\mathbf{P}_2| = \alpha \qquad |\mathbf{P}_2\mathbf{P}_3| = \beta \tag{2}$$

which is equivalent to

$$(\mathbf{P}_1 - \mathbf{P}_2)\cdot(\mathbf{P}_1 - \mathbf{P}_2) = \alpha^2 \qquad (\mathbf{P}_2 - \mathbf{P}_3)\cdot(\mathbf{P}_2 - \mathbf{P}_3) = \beta^2 \tag{3}$$

Substituting $\mathbf{P}_1$ and $\mathbf{P}_3$ into the equation of $\mathbf{P}_2$, the difference vectors are computed as follows:

$$\mathbf{P}_1 - \mathbf{P}_2 = \frac{\alpha}{\alpha+\beta}[\mathbf{V} + \alpha\mathbf{T}_s + \beta\mathbf{T}_e] \tag{4}$$

$$\mathbf{P}_2 - \mathbf{P}_3 = \frac{\beta}{\alpha+\beta}[\mathbf{V} + \alpha\mathbf{T}_s + \beta\mathbf{T}_e] \tag{5}$$

where $\mathbf{V} = \mathbf{P}_0 + \mathbf{P}_4$. Computing the dot products in Eq. (3), after simplifications one gets

$$\mathbf{V}\cdot\mathbf{V} + 2\alpha\mathbf{V}\cdot\mathbf{T}_s + 2\beta\mathbf{V}\cdot\mathbf{T}_e + 2\alpha\beta(\mathbf{T}_s\cdot\mathbf{T}_e - 1) = 0 \tag{6}$$

The only unknowns in Eq. (6) are the constants α and β. Various conditions can be imposed on the ratio α/β to have a unique solution. The choice that was made in this research is to set $\alpha = \beta$, after which the general equation reduces to

$$\mathbf{V}\cdot\mathbf{V} + 2\alpha\mathbf{V}\cdot(\mathbf{T}_s + \mathbf{T}_e) + 2\alpha^2(\mathbf{T}_s\cdot\mathbf{T}_e - 1) = 0 \tag{7}$$

Eq. (6) is similar to the one found in Ref. [13] and the choice of $\alpha = \beta$ is consistent with that of Ref. [20]. This choice not only makes computations simple, it becomes critical in reproducing circular data. In the general case, Eq. (7) always has a positive root which, together with Eq. (1), uniquely determines the unknown control points.

The attractive feature of the above formulation is that special cases are easily identified and computed. These cases appear when

$$\mathbf{T}_s\cdot\mathbf{T}_e = \{1, -1\} \qquad \mathbf{V}\cdot(\mathbf{T}_s + \mathbf{T}_e) = 0 \tag{8}$$

The first condition implies that the end tangents are parallel, whereas the second states that the vector sum of the end tangents is perpendicular to the chord joining the end points. In the case of parallel end tangents biarcs, can be constructed without solving the quadratic equation. The second type of special case is solved by piecing two biarcs together at the midpoint of the chord joining the end points, using the tangent as the bisector of the end tangent directions [18].

To express the biarcs in NURBS form, in addition to the control points, knot values and weights are also needed. For a two-arc biarc curve let t be a parameter value computed as

$$t = \frac{|\mathbf{P}_0\mathbf{P}_2|}{|\mathbf{P}_0\mathbf{P}_2| + |\mathbf{P}_2\mathbf{P}_4|} \tag{9}$$

then the knot vector is simply

$$U = \{0, 0, 0, t, t, 1, 1, 1\} \tag{10}$$

which can be rescaled to any other span to yield a more general form

$$U = \{u_0 = u_1 = u_2, u_3 = u_4, u_5 = u_6 = u_7\} \tag{11}$$

The weights are assigned as follows [17]:

$$w_0 = w_2 = w_4 = 1 \qquad w_1 = \cos(\alpha_1) \qquad w_3 = \cos(\alpha_2) \tag{12}$$

where α_1 and α_2 denote the half sweep angles of the first and the second arc, respectively.

3. NURBS approximation

For notational convenience, we start with curve definition. A general NURBS curve of degree p is defined as follows [17]:

$$\mathbf{C}(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_i^w \tag{13}$$

where $\mathbf{P}_i^w$ are the weighted control points, and $N_{i,p}(u)$ are the normalized B-splines defined over the knot vector

$$U = \left\{ \underbrace{u_0 = \cdots = u_p}_{p+1}, u_{p+1}, \ldots, u_n, \underbrace{u_{n+1} = \cdots = u_{n+p+1}}_{p+1} \right\} \tag{14}$$

Given a user defined tolerance ϵ, a piecewise G^1 biarc curve is sought so that the biarc does not deviate from the curve more than ϵ. That is, for any given point $\mathbf{P}$ lying on the piecewise biarc,

$$\left|\mathbf{C}(t_p) - \mathbf{P}\right| < \epsilon \qquad \left(\mathbf{C}(t_p) - \mathbf{P}\right)\cdot\mathbf{C}'(t_p) = 0 \tag{15}$$

where t_p is the parameter corresponding to the projection of

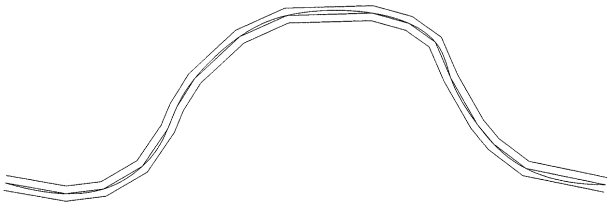


Fig. 2. Polygonal decomposition.

P onto the NURBS curve. To obtain such an approximation, the following conceptual procedure is applied, Fig. 2:

- obtain a polygonal approximation of the NURBS curve to within the tolerance $\epsilon/2$;
- offset the polygon by $\epsilon/2$ in each direction to obtain a fat polygon of width ϵ; and
- construct the piecewise G^1 biarc curve that lies within the fat polygon.

Because of the $\epsilon/2$ polygonal approximation, the original NURBS curve lies entirely within the fat polygon. If the G^1 biarc also lies within the fat polygon, it approximates the curve to within ϵ. Therefore, the first objective is to get a fast and reliable polygonal decomposition of NURBS curves.

3.1. Polygonal decomposition

There are essentially two kinds of polygonal decompositions: (1) parametric; and (2) geometric. The parametric decomposition computes a set of parameters where the vertices of the polygon are assumed. To avoid the computation of expensive quantities such as curvature, equally spaced parameters are computed. If the curve is parameterized on [0, 1], the number of points is computed as follows [5]:

$$n = \sqrt{\frac{1}{8}\frac{M}{\epsilon}} \tag{16}$$

where M is a bound on the second derivative. Given n, the polygonal decomposition entails computing points at $1/n$ increments. While this is a very simple method, and it works quite well for large tolerances, it has a number of shortcomings:

- it is parameterization dependent, i.e. different parameterizations of the same curve may give drastically different derivatives and hence number of points;
- computing a sharp bound on the second derivative is a challenging problem, especially for rational curves; and
- the formula tends to oversample the curve quite a bit causing a slow down in the algorithm (see Table 1).

The geometric decomposition is essentially a classical subdivision scheme based on the convex hull property of NURBS [6]. The process is outlined as follows.

1. Decompose the NURBS curve into piecewise Bézier segments.
2. For each Bézier piece $\mathscr{B}(\mathbf{P}_0, \ldots, \mathbf{P}_p)$ do:
   2.1. Initialize a stack with the curve $\mathscr{B}$
   2.2. While the stack is not empty do
      2.2.1. $\mathscr{B} \leftarrow$ Pop the stack
      2.2.2. If $\mathscr{B}$ is straight, output line ($l[\mathbf{P}_0, \mathbf{P}_p]$.
      2.2.3. Else, subdivide $\mathscr{B}$ at one-half the parameter range.
      2.2.4. Add the right and the left pieces to the stack.

The subcurve $\mathscr{B}[\mathbf{P}_0, \ldots, \mathbf{P}_p]$ is declared straight if the distances

$$d\Big(\mathbf{P}_i, l\Big[\mathbf{P}_0, \mathbf{P}_p\Big]\Big) < \epsilon \qquad i = 1, \cdots, p-1 \tag{17}$$

where $l[\mathbf{P}_0, \mathbf{P}_p]$ denotes the straight line between $\mathbf{P}_0$ and $\mathbf{P}_p$. The reason why the NURBS is decomposed into Bézier is twofold: (1) the individual Bézier pieces tend to be simple; and (2) subdividing the Bézier is cheaper than subdividing the NURBS.

A comparison of the two methods is found in Table 1 using the test curve shown in Fig. 2. n_p and n_g denote the number of polygon vertices required by the parametric and the geometric processes, respectively.

It is evident that for high precision manufacturing the derivative based method provides a prohibitively large number of points. Even in the range of 10^{-3}–10^{-4} it oversamples by an average factor of 25. Therefore, in this research, the geometric method was chosen.

3.2. Error control

Given a subset of the vertices $\mathbf{P}_s, \ldots, \mathbf{P}_e$, the objective is to check if the biarc, fitted to $\mathbf{P}_s$, $\mathbf{P}_e$ and the tangents there, is acceptable for approximation. That is, the biarc must be within $\epsilon/2$ of the polygon $\mathbf{P}_s, \ldots, \mathbf{P}_e$. The central element of the error control is the computation of the distance between a line segment **PQ** and a circular arc, Fig. 3. Assuming that both points are within the sweep angle of the arc, the distance is defined as follows:

Table 1

ϵ	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}
n_p	56	224	889	3521	14,000	57,741	221,886	883,344	3,516,646
n_g	7	14	38	127	422	1238	4002	13,356	39,427
n_p/n_g	8.0	16.0	23.4	27.7	33.2	46.6	55.4	66.1	89.2

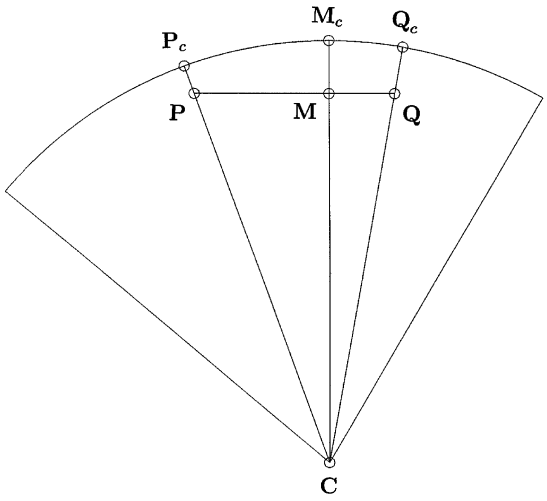


Fig. 3. Distance of line segment and circular arc.

$$d = \max\left(d_{\mathrm{p}}, d_{\mathrm{q}}, d_{\mathrm{m}}\right) \tag{18}$$

where

$$d_{\mathrm{p}} = d(\mathbf{P}, \mathbf{P}_{\mathrm{c}}) \qquad d_{\mathrm{q}} = d(\mathbf{Q}, \mathbf{Q}_{\mathrm{c}})$$

$$d_{\mathrm{m}} = \begin{cases} d(\mathbf{M}, \mathbf{M}_{\mathrm{c}}) & \text{if } \mathbf{M} \in \mathbf{PQ} \\ -1.0 & \text{otherwise} \end{cases} \tag{19}$$

The point **M** is the perpendicular projection of the center **C** onto the line segment **PQ**. If one of the points is not within the sweep angle, the line segment is clipped with respect to the bounding rays and the new segment is tested. If both points are outside the sweep angle, the line segment is not accepted, and a predefined distance, called UNDEFINED, is returned.

Now, given a biarc curve and a polygon, the algorithm proceeds as follows.

1. For each vertex, find the closest biarc. Note that a polygon vertex can lie within the sweep angles of more than one biarc.
2. For each polygon leg compute the distances from the participating circles obtained in step one.
3. If the maximum of all non UNDEFINED distances is less than the tolerance, the subset is accepted.

Several examples are shown in Fig. 4. Fig. 4(a) shows a C-shaped biarc in which almost all points lie within the sweep angles of both biarcs. The line segment toward the middle of the biarc is split so that the first part is tested against the first arc, whereas the second one is tested against the second arc. An S-shaped example is given in Fig. 4(b). Only one arc is visible from each point, and two polygon legs must be split for successful error check. Fig. 4(c) shows the special case when the vector sum of the end tangents is perpendicular to the chord. Several points are visible from three arcs, and only one leg split is necessary as all points lie within the respective closest arc, except one. The special case of a straight line, defined by collinear end tangents, is illustrated in Fig. 4(d). All points must project to the internal part of the line segment and must be within tolerance.

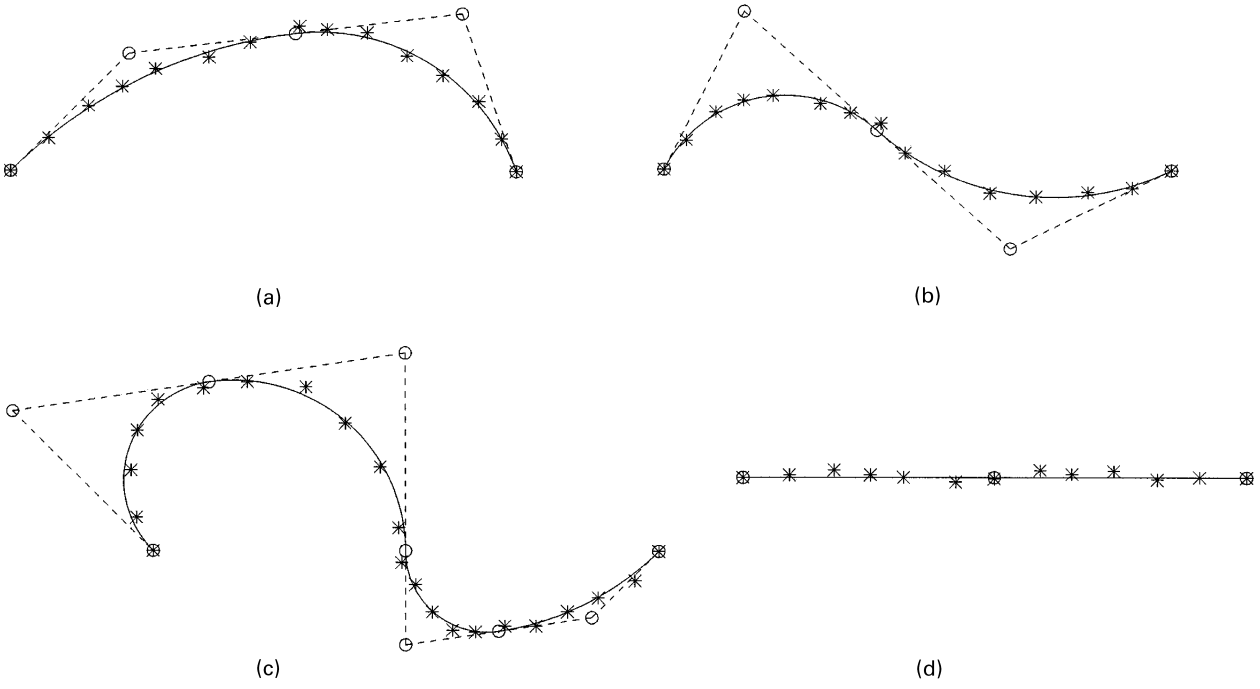


Fig. 4. (a) C-shaped biarc with point subset. (b) S-shaped biarc with point subset. (c) Four-segment biarc with point subset. (d) Degenerate biarc with point subset.

3.3. Biarc fitting

Given the biarc formulation and the error control mechanism, a very robust algorithm can be designed to approximate arbitrary NURBS curves. The major steps of the algorithm are outlined below.

1. Decompose the NURBS curve into segments of at least C^1 continuity. That is, extract segments from the input curve that have knots of multiplicity less than the degree. The rationale behind this step is to capture local phenomena such as cusps or straight line segments. If a profile curve contains corners, lines and arcs, these segments are normally pieced together by knots of multiplicity equal to the degree.
2. For each at least C^1 smooth curve get a biarc approximation, and piece the biarcs together. The important steps are as follows.
   2.1. Get a polygonal decomposition for $\epsilon/2$. Store both the polygon vertices as well as the parameters where these vertices are assumed.
   2.2. Compute the unit tangents at the polygon vertices.
   2.3. Segment the vertex set into a subset so that the fitted biarcs satisfy the tolerance requirement.
   2.4. Piece the biarcs together into a G^1 NURBS curve for compact storage. The knot vector is computed from the chord length parameterization of the junction points as outlined for two segments above.

The critical part of the algorithm is the segmentation of the data set. The method is explained as follows. Let us denote the biarc by $\mathscr{B} = \{\mathbf{P}_s, \mathbf{T}_s, \mathbf{P}_e, \mathbf{T}_e\}$. Now, given a set of vertices $\mathbf{P}_0, \ldots, \mathbf{P}_K$, an index set $I_0, \ldots, I_r$ is sought so that the biarcs

$$\mathscr{B}_l = \left\{\mathbf{P}_{I_l}, \mathbf{T}_{I_l}, \mathbf{P}_{I_{l+1}}, \mathbf{T}_{I_{l+1}}\right\}, l = 0, \ldots, r-1 \tag{20}$$

are all within tolerance from the polygonal approximation, and their number r is optimum in some way. Optimality can be in many forms, e.g. minimum number of arcs, minimum average error, minimum range of curvature, etc. Experience shows that computing such an optimum is fairly expensive and may not even yield aesthetically acceptable curves. The method that was applied is a simple search technique that tries to find the longest arc once a start point and a start tangent are computed. The algorithm is explained as follows.

1. $ks = 0$; $kd = K$;
2. While ($ks < K$) do
   2.1. $ke = \min(ks + kd, K)$; $kl = ks$; $kr = ke$; $ext = yes$;
   2.2. While ($kl \neq ke$) do
      2.2.1. Get $\mathscr{B} = \{\mathbf{P}_{ks}, \mathbf{T}_{ks}, \mathbf{P}_{ke}, \mathbf{T}_{ke}\}$.
      2.2.2. Check error of approximation
      2.2.3. If within tolerance
         2.2.3.1. If ($ext = yes$ and $ke < K$) $ke = \min(ks + kd, K)$; $kr = ke$;
         2.2.3.2. Else, save control points; $kl = ke$;
      2.2.4. Else, kr = ke; ext = no;
      2.2.5. If ($ext = no$) $ke = (kl + kr)/2$;
   2.3. $kd = \max(1, ke - ks)$; $ks = ke$;

The method first finds the longest arc starting from the beginning. Once the first arc is found, the increment kd is set to be the difference between the start and end indexes. The idea is borrowed from computer graphics and is called the *coherence principle*. That is, it is expected that the i-th segment approximates about the same number of vertices as the $(i-1)$-th one did. If it happens that the i-th segment is immediately within tolerance, the end index ke is extended (hence the use of the flag *ext*). Binary search commences once the point set $\mathbf{P}_{ks}, \ldots, \mathbf{P}_{ke}$ cannot be approximated by one arc within the required tolerance.

Even though the above segmentation method is not strictly optimal, practical experience shows that the distribution of biarcs is in line with the designer's expectations. Additionally, the coherence principle based search makes the algorithm quite fast as each new biarc construction requires no more than a few iterations.

Fig. 5 shows the working of the algorithm. The biarcs with control points (solid curve) and the original curve (dashed) are presented in Fig. 5(a) using $\epsilon = 0.02$. For better visualization, the control points are turned off in Fig. 5(b). Fig. 5(c) is the critical figure that shows all three curves, the original curve (dashed), the biarc approximation (solid) as well as the polygonal approximation (solid), are within the fat polygon. Fig. 5(d)–(f) present curvature plots for tolerances 0.02, 10^{-4} and 10^{-5}, respectively. Notice the tremendous change in curvature along the original curve. The objective of biarc approximation is to reproduce the shape of this curve with a step function. This clearly indicates that the curves that are more suitable for biarc approximation are the ones that have smoothly changing curvature. It took a tight tolerance of 10^{-5} to recapture the details of the curvature plot, Fig. 5(f). Notice that the two plots do not overlap due to the difference in parameterization of the two curves (the original NURBS curve and its biarc approximation). Nevertheless, both the shape of the curvature plot as well as the peaks are reasonably well reproduced.

4. Tests and examples

The above algorithm has been tested on a variety of data sets, including both open and closed curves. We were able to approximate arbitrary NURBS curves in real time up to 10^{-7} tolerance on a Windows workstation. Error checks have also been performed and found that the approximation was well within the given tolerance, i.e. in most cases the maximum error was found to be slightly more than half the required tolerance.

Two more test cases are presented in Figs. 6 and 7. Fig. 6(a) shows the biarc approximation of a straight curve and a

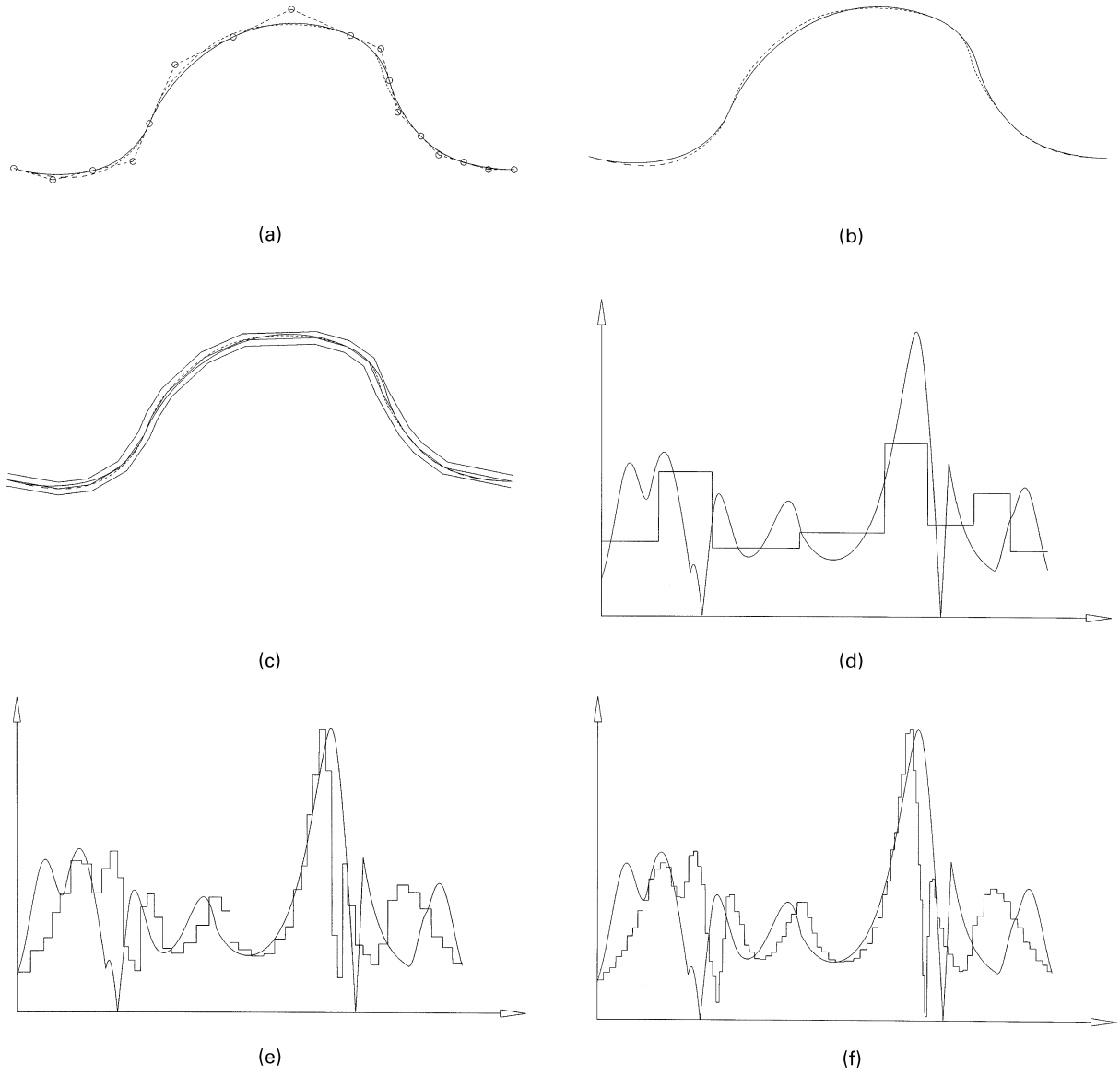


Fig. 5. (a) Biarc approximation: $\epsilon = 0.02$. (b) Biarc approximation, control points turned off: $\epsilon = 0.02$. (c) Biarc approximation with fat polygon: $\epsilon = 0.02$. (d) Curvature plots; $\epsilon = 0.02$. (e) Curvature plots: $\epsilon = 0.0001$. (f) Curvature plots: $\epsilon = 0.00001$.

cusp using $\epsilon = 0.01$. The curvature plots are shown in Fig. 6(b) and (c) for tolerances of 10^{-2} and 10^{-5}, respectively. Notice how well the peaks as well as the shape of the curvature plot is approximated.

A shoe insole design is illustrated in Fig. 7. The insole was obtained by interpolation of digitized points, and subsequently it was approximated by biarc. Fig. 7(a) shows the original curve and its biarc approximation (with control points) for $\epsilon = 0.005$. The curvature plots are in Fig. 7(b) and (c) for tolerances 0.005 and 10^{-4}, respectively. Notice the overlap in the plots. This is because both curves are parameterized the same way, i.e. using chord length parameterization.

A legitimate question arises as to why not parameterize the biarc the same way the original curve is parameterized. The answer is twofold: (1) the biarc does not need any parameterization, the one that is used is given for compact storage only; and (2) many input curves are not well parameterized and it is a bad idea to inherit a not so good parameterization.

An interesting question is 'how does the number of circular arcs depend on the tolerance?'. Table 2 shows the result of an empirical study on the curves of Figs. 5–7.

Table 2

ϵ	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
Fig. 5	8	20	48	100	188	376	788
Fig. 6	8	20	36	52	116	220	432
Fig. 7	16	36	72	148	284	604	1302

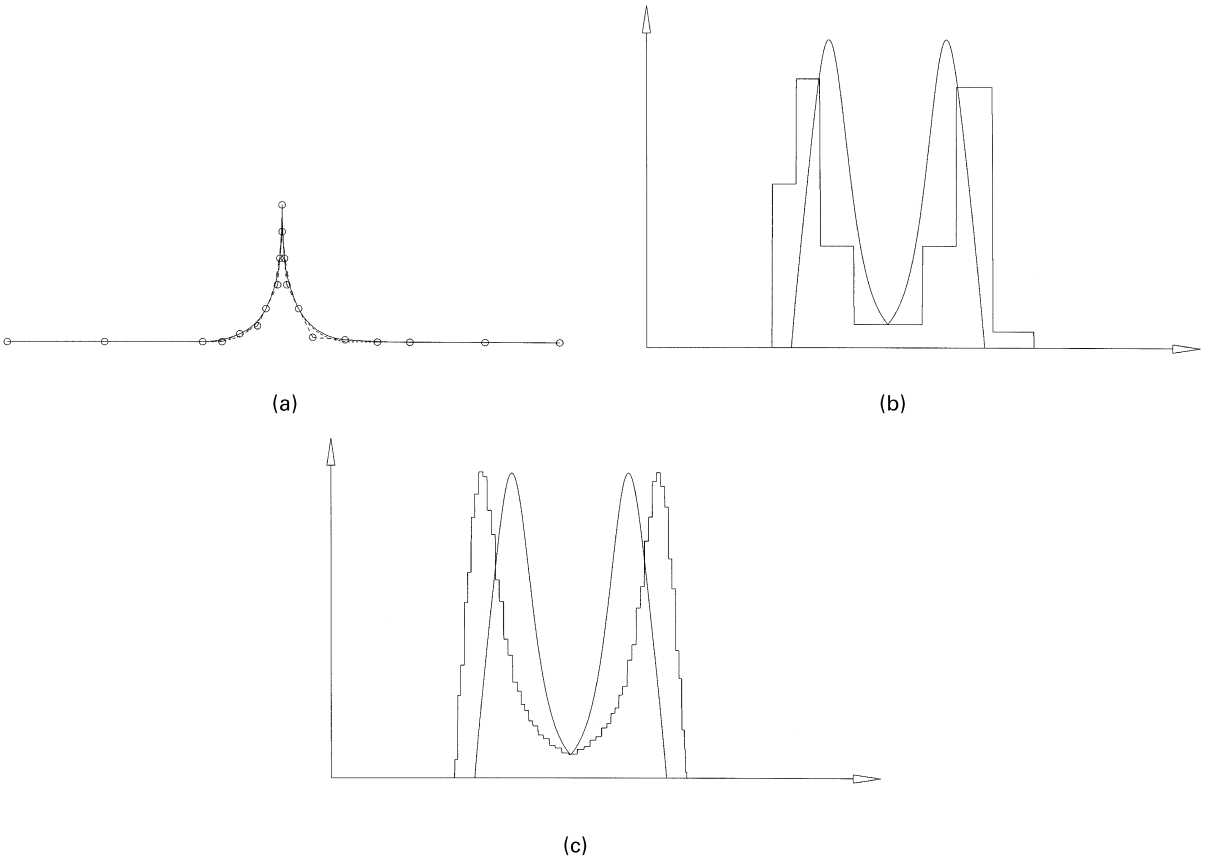


Fig. 6. (a) Biarc approximation example: $\epsilon = 0.01$. (b) Curvature plots: $\epsilon = 0.01$; curvature plots: $\epsilon = 0.00001$.

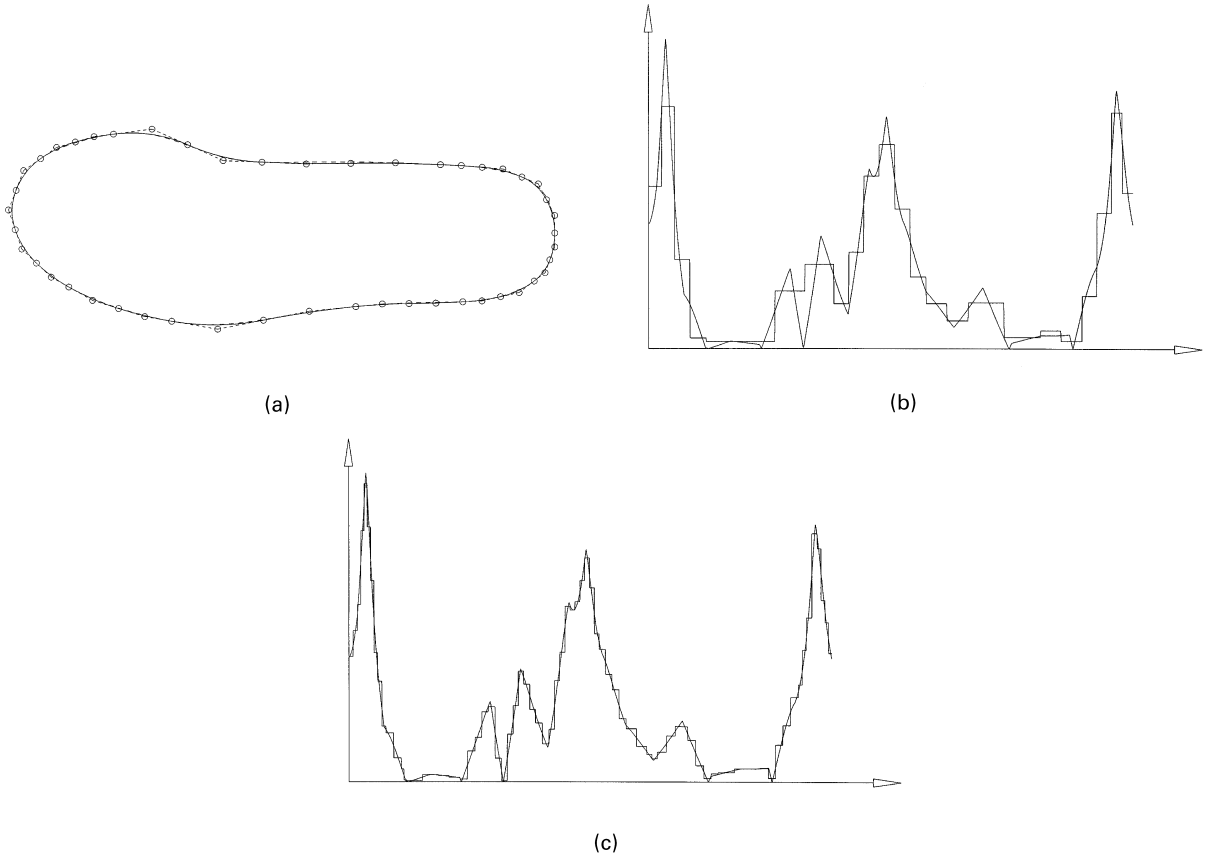


Fig. 7. (a) Biarc approximation example: $\epsilon = 0.005$. (b) Curvature plots: $\epsilon = 0.001$. (c) Curvature plots: $\epsilon = 0.0001$.

Practical tests confirmed that as the tolerance drops by an order of magnitude, the number of segments roughly double. This is actually quite a reasonable increase considering that the decrease is tenfold.

5. Conclusions

A method to approximate arbitrary NURBS curves with G^1 biarcs is presented. A special parametric formulation of the biarc curves makes the method suitable in a NURBS modeling environment. The algorithm is robust as virtually all operations are geometric calculations, e.g. line intersections, point projections, and various vector operations such as dot and cross products. Because of the simplicity of computations, the algorithm performs in real time up to about 10^{-7} even on a regular Windows workstation. The method is most appropriate in engineering applications where the original curves have smoothly changing curvature, and the output curves are required to be lines or arcs, or where the application necessitates a piecewise curve with piecewise constant curvature.

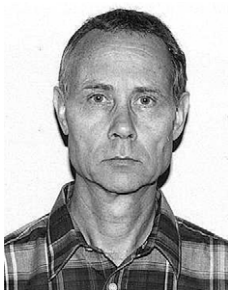
Acknowledgements

The authors thank Professors D. Walton, I.-K. Lee and M.-S. Kim for their many valuable comments and criticisms.

References

[1] Ahn YJ, Kim HO, Lee K-Y. G^1 arc spline approximation of quadratic Bézier curves. Computer-Aided Design 1998;30(8):615–20.
[2] Bézier PE. Numerical control: mathematics and applications. New York: John Wiley & Sons, 1972.
[3] Bolton KM. Biarc curves. Computer-Aided Design 1975;7(2):89–92.
[4] Chuang S-H, Kao CZ. One-sided arc approximation of B-spline curves for interference-free offsetting. Computer-Aided Design 1999;31(2):111–8.
[5] Filip D, Magedson R, Markot R. Surface algorithms using bounds on derivatives. Computer-Aided Geometric Design 1986;3(4):295–311.
[6] Lane JM, Riesenfeld RF. A theoretical development for the computer generation and display of piecewise polynomial surfaces. IEEE Transactions on Pattern Analysis and Machine Intelligence 1980;PAMI-2(1):35–46.
[7] Hoschek J. Circular splines. Computer-Aided Design 1992;11:611–8.
[8] Marciniak R, Putz B. Approximation of spirals by piecewise curves of fewest circular arc segments. Computer-Aided Design 1984;16(2):87–90.
[9] Meek DS, Walton DJ. Approximation of discrete data by G^1 arc splines. Computer-Aided Design 1992;24(6):301–6.
[10] Meek DS, Walton DJ. Approximating quadratic NURBS curves by arc splines. Computer-Aided Design 1993;25(6):371–6.
[11] Meek DS, Walton DJ. Approximating smooth planar curves by arc splines. J Computing and Applied Mathematics 1995;59:221–31.
[12] Moreton DN, Parkinson DB. The application of biarc technique in CNC machining. Computer-Aided Engineering Design Journal 1991;8:54–60.
[13] Nutbourne AW, Martin RR. Differential geometry applied to curve and surface design, vol. 1: foundations. Chichester, UK: Ellis Horwood, 1988.
[14] Ong CJ, Wong YS, Loh HT, Hong XG. An optimization approach for biarc curve-fitting of B-spline curves. Computer-Aided Design 1996;28(12):951–9.
[15] Parkinson DB, Moreton DN. Optimal biarc-curve fitting. Computer-Aided Design 1991;23(6):411–9.
[16] Piegl L. Curve fitting algorithm for rough cutting. Computer-Aided Design 1986;18(2):79–82.
[17] Piegl L, Tiller W. The NURBS book. 2nd ed.. New York: Springer-Verlag, 1997.
[18] Piegl L, Tiller W. Data approximation using biarcs. Technical Report, Department of Computer Science, University of South Florida (also submitted for publication).
[19] Qiu H, Cheng K, Li Y. Optimal circular arc interpolation for NC tool path generation in curve contour manufacturing. Computer-Aided Design 1997;29(11):751–60.
[20] Rossignac JR, Requicha AA. Piecewise circular curves for geometric modeling. IBM Journal of Research and Development 1987;31(3):296–313.
[21] Sabin MA. The use of piecewise forms for the numerical representation of shape. Reports 1977;60:1977.
[22] Schönherr J. Smooth biarc curves. Computer-Aided Design 1993;25(6):365–70.
[23] Su B-Q, Liu D-Y. Computational geometry—curve and surface modeling. New York: Academic Press, 1989.
[24] Yang X. Approximating NURBS curves by arc splines. In: Martin RR, Wang W, editors. Geometric modeling and processing 2000, Los Alamitos, CA: IEEE Computer Society, 2000. p. 175–83.
[25] Yeung MK, Walton DJ. Curve fitting with arc splines for NC toolpath generation. Computer-Aided Design 1994;26(11):845–9.
[26] Yong J-H, Hu S-M, Sun J-G. A note on approximation of discrete data by G^1 splines. Computer-Aided Design 1999;31(14):911–5.
[27] Yong J-H, Hu S-M, Sun J-G. Bisection algorithm for approximating quadratic Bézier curves by G^1 arc splines. Computer-Aided Design 2000;32(4):253–60.

Les A. Piegl is a Professor in the Department of Computer Science and Engineering, University of South Florida, Tampa, Florida, USA. His research interests are in CAD/CAM, geometric modeling, data structures and alogrithms, computer graphics and software engineering. He spent many years researching and implementing geometric algorithms in academia as well as in industry. He is the co-author of the text book *The NURBS Book*, published by Springer-Verlag. He serves as Editor for *Computer-Aided Design*.



Wayne Tiller is President of GeomWare, Inc., a company specializing in NURBS technology and software. He has 28 years experience in applied mathematics, computer science and software development. He has worked in the area of NURBS geometry since 1981, conducting research and implementing software. He has published numerous papers on this topic and is co-author of the book *The NURBS Book*. He received a PhD in mathematics from Texas Christian University, USA.